How can the predict function be used with the glm function in R? Can you provide some examples?

Authored by

stats writer

April 27, 2024

The predict function in R is a powerful tool that can be used in conjunction with the glm (generalized linear model) function to make predictions based on a fitted model. This function allows users to generate predicted values for a particular response variable based on the input of explanatory variables. This can be useful in various statistical analyses, such as regression and classification, to make inferences or forecasts.

To use the predict function with the glm function, first a model must be fitted using the glm function and saved as an object. Then, the predict function can be applied to this object, with the appropriate arguments specified, to generate the predicted values.

For example, if we have a dataset with a binary response variable (0 or 1) and several explanatory variables, we can fit a logistic regression model using the glm function. After saving the fitted model as an object, we can use the predict function to generate predicted probabilities for the response variable based on the explanatory variables. These probabilities can then be used to make predictions about the likelihood of an event occurring.

In another example, if we have a dataset with a continuous response variable and multiple explanatory variables, we can fit a multiple linear regression model using the glm function. We can then use the predict function to generate predicted values for the response variable based on the input of the explanatory variables. These predicted values can be compared to the actual values to assess the accuracy of the model.

In conclusion, the predict function can be a useful tool when used with the glm function in R, allowing for the generation of predicted values based on fitted models. This can aid in making predictions and evaluating the performance of the model.

Use the predict function with glm in R (With Examples)

The glm() function in R can be used to fit generalized linear models. This function is particularly useful for fitting , , and other complex models.

Once we've fit a model, we can then use the predict() function to predict the response value of a new observation.

This function uses the following syntax:

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

where:

object: The name of the model fit using the glm() functionnewdata: The name of the new data frame to make predictions fortype: The type of prediction to make.

The following example shows how to fit a generalized linear model in R and how to then use the model to predict the response value of a new observation it hasn't seen before.

Example: Using the predict function with glm in R

For this example, we'll use the built-in R dataset called mtcars:

#view first six rows of *mtcars* data frame
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

We'll fit the following logistic regression model in which we use the variables disp and hp to predict the response variable am (the transmission type of the car: 0 = automatic, 1 = manual).

```
#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

We can then use this model to predict the probability that a new car has an automatic transmission (am=0) or a manual transmission (am=1) by using the following code:

```
#define new observation
newdata = data.frame(disp=200, hp= 100)

#use model to predict value of am
```

```
predict(model, newdata, type="response")

1
0.00422564
```

The model predicts the probability of the new car having a manual transmission (am=1) to be 0.004. This means it's highly like that this new car has an automatic transmission.

Note that we can also make several predictions at once if we have a data frame that has multiple new cars.

For example, the following code shows how to use the fitted model to predict the probability of a manual transmission for three new cars:

```
#define new data frame of three cars
newdata = data.frame(disp=c(200, 180, 160),
hp=c(100, 90, 108))

#view data frame
newdata

disp hp
1 200 100
```

2 180 90

3 160 108

#use model to predict value of *am* for all three cars
predict(model, newdata, type="response")

1 2 3
0.004225640 0.008361069 0.335916069

Here's how to interpret the output:

The probability that car 1 has a manual transmission is .004.The probability that car 2 has a manual transmission is .008.The probability that car 3 has a manual transmission is .336.

Notes

The names of the columns in the new data frame should exactly match the names of the columns in the data frame that were used to build the model.

Notice that in our previous example, the data frame we used to build the model contained the following column names for our predictor variables:

disphp

Thus, when we created the new data frame called newdata we made sure to also name the columns:

disphp

If the names of the columns do not match, you'll receive the following error message:

Error in eval(predvars, data, env)

Keep this in mind when using the predict() function.