What is the process for implementing Multivariate Adaptive Regression Splines in R?

Authored by

stats writer

April 22, 2024

The process for implementing Multivariate Adaptive Regression Splines (MARS) in R involves several steps. First, the data must be prepared and cleaned to ensure its suitability for MARS analysis. Next, the "earth" package must be installed in R, which contains the necessary functions for MARS analysis. Once the package is installed, the data can be imported into R and the MARS model can be built using the "earth" function. This function allows for various model settings and options to be specified, such as the maximum number of terms and the degree of interaction. After the model is built, it can be evaluated and validated using techniques such as cross-validation. Finally, the model can be used to make predictions on new data. It is important to note that the process for implementing MARS in R may vary slightly depending on the specific goals and data of the analysis, but these are the general steps involved.

Multivariate Adaptive Regression Splines in R

Multivariate adaptive regression splines (MARS) can be used to model nonlinear relationships between a set of predictor variables and a response variable.

This method works as follows:

1. Divide a dataset into *k* pieces.

2. Fit a regression model to each piece.

3. Use k-fold cross-validation to choose a value for *k*.

This tutorial provides a step-by-step example of how to fit a MARS model to a dataset in R.

Step 1: Load Necessary Packages

For this example we'll use the Wage dataset from the

ISLR package, which contains the annual wages for 3,000 individuals along with a variety of predictor variables like age, education, race, and more.

Before we fit a MARS model to the data, we'll load the necessary packages:

library(ISLR) #contains Wage datasetlibrary(dplyr) #data wranglinglibrary(ggplot2) #plottinglibrary(earth) #fitting MARS modelslibrary(caret) #tuning model parameters

Step 2: View Data

Next, we'll view the first six rows of the dataset we're working with:

#view first six rows of data
head(Wage)

year age maritl race education region
231655 2006 18 1. Never Married 1. White 1. < HS Grad 2. Middle Atlantic
86582 2004 24 1. Never Married 1. White 4. College Grad 2. Middle Atlantic
161300 2003 45 2. Married 1. White 3. Some College 2. Middle Atlantic

155159 2003 43 2. Married 3. Asian 4. College Grad 2. Middle Atlantic
11443 2005 50 4. Divorced 1. White 2. HS Grad 2. Middle Atlantic
376662 2008 54 2. Married 1. White 4. College Grad 2. Middle Atlantic
jobclass health health_ins logwage wage
231655 1. Industrial 1. <=Good 2. No 4.318063 75.04315
86582 2. Information 2. >=Very Good 2. No 4.255273 70.47602
161300 1. Industrial 1. <=Good 1. Yes 4.875061 130.98218
155159 2. Information 2. >=Very Good 1. Yes 5.041393 154.68529
11443 2. Information 1. <=Good 1. Yes 4.318063 75.04315
376662 2. Information 2. >=Very Good 1. Yes 4.845098 127.11574

Step 3: Build & Optimize the MARS Model

Next, we'll build the MARS model for this dataset and perform **k-fold cross-validation** to determine which model produces the lowest test RMSE (root mean

squared error).

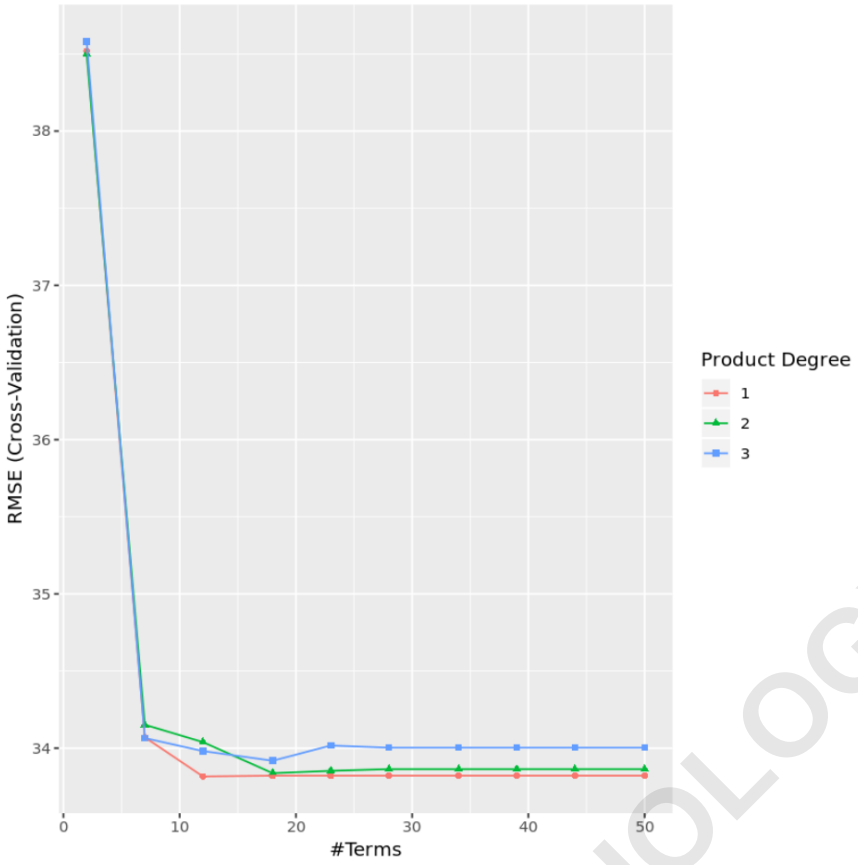
```
#create a tuning grid
hyper_grid <- expand.grid(degree = 1:3,
nprune = seq(2, 50, length.out = 10) %>% floor())

#make this example reproducible
set.seed(1)

#fit MARS model using k-fold cross-validation
cv_mars <- train(
x = subset(Wage, select = -c(wage, logwage)),
y = Wage$wage,
method = "earth",
metric = "RMSE",
trControl = trainControl(method = "cv", number = 10),
tuneGrid = hyper_grid)

#display model with lowest test RMSE
cv_mars$results
%>%filter(nprune==cv_mars$bestTune$nprune, degree
=cv_mars$bestTune$degree)
degree nprune RMSE Rsquared MAE RMSESD
RsquaredSD MAESD
1 12 33.8164 0.3431804 22.97108 2.240394 0.03064269
```

1.4554

From the output we can see that the model that produced the lowest test MSE was one with only first-order effects (i.e. no interaction terms) and 12 terms. This model produced a root mean squared error (RMSE) of 33.8164.

We can also create a plot to visualize the test RMSE based on the degree and the number of terms:

#display test RMSE by terms and degree
ggplot(cv_mars)



In practice we would fit a MARS model along with several other types of models like:

Multiple Linear RegressionPolynomial RegressionRidge RegressionLasso RegressionPrincipal Components RegressionPartial Least Squares

We would then compare each model to determine which one lead to the lowest test error and choose that model as the optimal one to use.

The complete R code used in this example can be found

here.