How can I perform Ridge Regression in Python, and what are the step-by-step instructions?

Authored by

stats writer

April 22, 2024

Ridge Regression is a popular technique used in data analysis for reducing the impact of multicollinearity and overfitting in linear regression models. It is implemented in Python using the scikit-learn library, which provides a wide range of machine learning tools. To perform Ridge Regression in Python, one can follow these simple steps:

1. Import the necessary libraries: The first step is to import the required libraries, including numpy, pandas, and sklearn.

2. Prepare the data: Ridge Regression requires a dataset with numerical values, so make sure to convert any categorical or text data into numerical values.

3. Split the data: Divide the dataset into training and testing sets to evaluate the performance of the model.

4. Fit the model: Use the Ridge() function from sklearn to create an instance of the Ridge Regression model and fit it to the training data.

5. Tune the hyperparameters: Ridge Regression has a hyperparameter called alpha that controls the amount of regularization. Experiment with different values of alpha to find the optimal value for your dataset.

6. Make predictions: Use the trained model to make predictions on the test data.

7. Evaluate the model: Use metrics such as mean squared error (MSE) or R-squared to evaluate the performance of the Ridge Regression model.

By following these steps, one can easily perform Ridge Regression in Python and improve the accuracy of their linear regression models.

Ridge Regression in Python (Step-by-Step)

Ridge regression is a method we can use to fit a regression model when **multicollinearity** is present in the data.

In a nutshell, least squares regression tries to find coefficient estimates that minimize the sum of squared

residuals (RSS):

RSS = Σ(yi - ?i)2

where:

Σ: A greek symbol that means *sum*
yi: The actual response value for the ith observation
?i: The predicted response value based on the multiple linear regression model

Conversely, ridge regression seeks to minimize the following:

RSS + λΣβj2

where *j* ranges from 1 to *p* predictor variables and λ ≥ 0.

This second term in the equation is known as a *shrinkage penalty*. In ridge regression, we select a value for λ that produces the lowest possible test MSE (mean squared error).

This tutorial provides a step-by-step example of how to perform ridge regression in Python.

Step 1: Import Necessary Packages

First, we'll import the necessary packages to perform ridge regression in Python:

```
import pandas as pd
from numpy import arange
from sklearn.linear_modelimport Ridge
from sklearn.linear_modelimport RidgeCV
from sklearn.model_selectionimport RepeatedKFold
```

Step 2: Load the Data

For this example, we'll use a dataset called mtcars, which contains information about 33 different cars. We'll use hp as the response variable and the following variables as the predictors:

mpg
wt
drat
qsec

The following code shows how to load and view this dataset:

```
#define URL where data is located
url = "https://raw.githubusercontent.com/Statology/Python-Guides/main/mtcars.csv"

#read in data
data_full = pd.read_csv(url)

#select subset of data
data = data_full]

#view first six rows of data
data

mpg wt drat qsec hp
0 21.0 2.620 3.90 16.46 110
1 21.0 2.875 3.90 17.02 110
2 22.8 2.320 3.85 18.61 93
3 21.4 3.215 3.08 19.44 110
4 18.7 3.440 3.15 17.02 175
5 18.1 3.460 2.76 20.22 105
```

Step 3: Fit the Ridge Regression Model

Next, we'll use the **RidgeCV()** function from sklearn to fit the ridge regression model and we'll use the

RepeatedKFold() function to perform k-fold cross-validation to find the optimal alpha value to use for the penalty term.

Note: The term "alpha" is used instead of "lambda" in Python.

For this example we'll choose k = 10 folds and repeat the cross-validation process 3 times.

Also note that RidgeCV() only tests alpha values .1, 1, and 10 by default. However, we can define our own alpha range from 0 to 1 by increments of 0.01:

```
#define predictor and response variables
X = data]
y = data

#define cross-validation method to evaluate model
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)

#define model
model = RidgeCV(alphas=arange(0, 1, 0.01), cv=cv, scoring='neg_mean_absolute_error')
```

#fit model
model.fit(X, y)

#display lambda that produced the lowest test MSE
print(model.alpha_)

0.99

The lambda value that minimizes the test MSE turns out to be 0.99.

Step 4: Use the Model to Make Predictions

Lastly, we can use the final ridge regression model to make predictions on new observations. For example, the following code shows how to define a new car with the following attributes:

mpg: 24
wt: 2.5
drat: 3.5
qsec: 18.5

The following code shows how to use the fitted ridge regression model to predict the value for *hp* of this new observation:

#define new observation

new =

#predict hp value using ridge regression model

model.predict()

array()

Based on the input values, the model predicts this car to have an *hp* value of 104.16398018.

You can find the complete Python code used in this example here.