What is Leave-One-Out Cross-Validation and how can it be implemented in Python?

Authored by

stats writer

April 21, 2024

Leave-One-Out Cross-Validation (LOOCV) is a technique used for evaluating the performance of a machine learning model. It involves partitioning the available data into training and testing sets, with one data point being held out for testing while the remaining data is used for training the model. This process is repeated for each data point, with each point being held out once. The model's performance is then evaluated by averaging the results from all iterations.

In Python, LOOCV can be implemented using the "LeaveOneOut" function from the "sklearn.model_selection" module. This function creates a cross-validator object that can be used in conjunction with other machine learning algorithms. It takes in the dataset and returns an iterator that generates indices to split the data into training and testing sets. This iterator can then be used in a loop to train and test the model on each iteration, and the final performance can be evaluated by averaging the results. LOOCV is a useful technique for evaluating the generalizability of a model and can help in avoiding overfitting.

Leave-One-Out Cross-Validation in Python (With Examples)

To evaluate the performance of a model on a dataset, we need to measure how well the predictions made by the model match the observed data.

One commonly used method for doing this is known as leave-one-out cross-validation (LOOCV), which uses the following approach:

1. Split a dataset into a training set and a testing set, using all but one observation as part of the training set.

2. Build a model using only data from the training set.

3. Use the model to predict the response value of the

one observation left out of the model and calculate the mean squared error (MSE).

4. Repeat this process *n* times. Calculate the test MSE to be the average of all of the test MSE's.

This tutorial provides a step-by-step example of how to perform LOOCV for a given model in Python.

Step 1: Load Necessary Libraries

First, we'll load the necessary functions and libraries for this example:

```
from sklearn.model_selectionimport train_test_split
from sklearn.model_selectionimport LeaveOneOut
from sklearn.model_selectionimport cross_val_score
from sklearn.linear_modelimport LinearRegression
from numpy import mean
from numpy import absolute
from numpy import sqrt
import pandas as pd
```

Step 2: Create the Data

Next, we'll create a pandas DataFrame that contains two

predictor variables, x1 and x2, and a single response variable y.

```
df = pd.DataFrame({'y': ,
'x1': ,
'x2': })
```

Step 3: Perform Leave-One-Out Cross-Validation

Next, we'll then fit a multiple linear regression model to the dataset and perform LOOCV to evaluate the model performance.

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

#define cross-validation method to use
cv = LeaveOneOut()

#build multiple linear regression model
model = LinearRegression()

#use LOOCV to evaluate model
scores = cross_val_score(model, X, y,
scoring='neg_mean_absolute_error',
```

```
cv=cv, n_jobs=-1)

#view mean absolute error
mean(absolute(scores))

3.1461548083469726
```

From the output we can see that the mean absolute error (MAE) was 3.146. That is, the average absolute error between the model prediction and the actual observed data is 3.146.

Another commonly used metric to evaluate model performance is the root mean squared error (RMSE). The following code shows how to calculate this metric using LOOCV:

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

#define cross-validation method to use
cv = LeaveOneOut()

#build multiple linear regression model
```

```
model = LinearRegression()

#use LOOCV to evaluate model
scores = cross_val_score(model, X, y,
scoring='neg_mean_squared_error',
cv=cv, n_jobs=-1)

#view RMSE
sqrt(mean(absolute(scores)))

3.619456476385567
```

From the output we can see that the root mean squared error (RMSE) was 3.619. The lower the RMSE, the more closely a model is able to predict the actual observations.

In practice we typically fit several different models and compare the RMSE or MAE of each model to decide which model produces the lowest test error rates and is therefore the best model to use.

A Quick Intro to Leave-One-Out Cross-Validation (LOOCV)

A Complete Guide to Linear Regression in Python