How do I perform K-Fold Cross Validation in Python, step-by-step?

Authored by

stats writer

April 22, 2024

K-Fold Cross Validation is a commonly used technique in machine learning to evaluate the performance of a model on unseen data. It involves dividing the dataset into k equal subsets, also known as folds, and using one fold as the test set while training the model on the remaining k-1 folds. This process is repeated k times, with each fold being used as the test set exactly once. The final performance metric is then calculated by averaging the results from each iteration.

To perform K-Fold Cross Validation in Python, follow these steps:

1. Import the necessary libraries: Begin by importing the required libraries such as NumPy, Pandas, and Scikit-Learn.

2. Load the dataset: Use the Pandas library to load the dataset into your Python program.

3. Split the dataset into k folds: Use the KFold function from the Scikit-Learn library to split the dataset into k folds.

4. Define the model: Choose the appropriate model for your dataset and define it using the Scikit-Learn library.

5. Train and test the model: Use a for loop to iterate through the folds and train the model on the training data, then evaluate its performance on the test data.

6. Calculate the performance metric: After all iterations are complete, calculate the average performance metric using the results from each fold.

7. Repeat with different values of k: You can repeat this process with different values of k to see how the performance of the model varies.

By following these steps, you can easily perform K-Fold Cross Validation in Python and obtain a more accurate evaluation of your model's performance.

K-Fold Cross Validation in Python (Step-by-Step)

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

k-fold cross-validation, which uses the following approach:

1. Randomly divide a dataset into *k* groups, or "folds", of roughly equal size.

2. Choose one of the folds to be the holdout set. Fit the model on the remaining k-1 folds. Calculate the test MSE on the observations in the fold that was held out.

3. Repeat this process *k* times, using a different set each time as the holdout set.

4. Calculate the overall test MSE to be the average of the *k* test MSE's.

This tutorial provides a step-by-step example of how to perform k-fold cross validation 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 KFold
```

```
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 K-Fold 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=KFold(n_splits=10,random_state=1,shuffle=True)#build multiple linear regression model
model = LinearRegression()

#use k-fold CV 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))

3.6141267491803646
```

From the output we can see that the mean absolute error (MAE) was 3.614. That is, the average absolute error between the model prediction and the actual observed data is 3.614.

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=KFold(n_splits=5,random_state=1,shuffle=True)#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)))

4.284373111711816
```

From the output we can see that the root mean squared error (RMSE) was 4.284.

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.

Also note that in this example we chose to use k=5 folds, but you can choose however many folds you'd like.

In practice, we typically choose between 5 and 10 folds because this turns out to be the optimal number of folds that produce reliable test error rates.

You can find the complete documentation for the KFold() function from sklearn here.

An Introduction to K-Fold Cross-Validation
A Complete Guide to Linear Regression in Python
Leave-One-Out Cross-Validation in Python