How can I create a train and test set from a Pandas DataFrame?

Authored by

stats writer

June 28, 2024

Creating a train and test set from a Pandas DataFrame is a common practice in machine learning and data analysis. This allows for the evaluation of the model's performance on new, unseen data. The process involves splitting the original dataset into two separate sets - a training set and a testing set. The training set is used to build and train the model, while the testing set is used to evaluate its performance. This can be achieved by using the "train_test_split" function from the "sklearn.model_selection" module. It randomly divides the data into the specified proportions and returns two new DataFrames. The train and test sets can then be used for further analysis and model development. It is important to properly balance the sizes of the two sets and to ensure that they represent the original data accurately in order to obtain reliable results.

Create a Train and Test Set from a Pandas DataFrame

When fitting to datasets, we often split the dataset into two sets:

1. Training Set: Used to train the model (70-80% of original dataset)

2. Testing Set: Used to get an unbiased estimate of the model performance (20-30% of original dataset)

In Python, there are two common ways to split a pandas DataFrame into a training set and testing set:

Method 1: Use train_test_split() from sklearn

from sklearn.model_selectionimport train_test_split

train, test = train_test_split(df, test_size=0.2,

```
random_state=0)
```

Method 2: Use sample() from pandas

```
train = df.sample(frac=0.8,random_state=0)
test = df.drop(train.index)
```

The following examples show how to use each method with the following pandas DataFrame:

```
import pandas as pd
import numpy as np

#make this example reproducible
np.random.seed(1)

#create DataFrame with 1,000 rows and 3 columns
df = pd.DataFrame({'x1': np.random.randint(30,
size=1000),
'x2': np.random.randint(12, size=1000),
'y': np.random.randint(2, size=1000)})

#view first few rows of DataFrame
df.head()
```

x1 x2 y

0 5 1 1

1 11 8 0

2 12 4 1

3 8 7 0

4 9 0 0

Example 1: Use train_test_split() from sklearn

The following code shows how to use the train_test_split() function from sklearn to split the pandas DataFrame into training and test sets:

```
from sklearn.model_selectionimport train_test_split

#split original DataFrame into training and testing sets
train, test = train_test_split(df, test_size=0.2, random_state=0)

#view first few rows of each setprint(train.head())
```

x1 x2 y

687 16 2 0

500 18 2 1

332 4 10 1

979 2 8 1

817 11 1 0

print(test.head())

x1 x2 y
993 22 1 1
859 27 6 0
298 27 8 1
553 20 6 0
672 9 2 1

#print size of each setprint(train.shape, test.shape)

(800, 3) (200, 3)

From the output we can see that two sets have been created:

Training set: 800 rows and 3 columnsTesting set: 200 rows and 3 columns

Note that test_size controls the percentage of observations from the original DataFrame that will belong to the testing set and the random_state value makes the split reproducible.

Example 2: Use sample() from pandas

```
#split original DataFrame into training and testing sets
train = df.sample(frac=0.8,random_state=0)
test = df.drop(train.index)

#view first few rows of each setprint(train.head())

x1 x2 y
993 22 1 1
859 27 6 0
298 27 8 1
553 20 6 0
672 9 2 1

print(test.head())

x1 x2 y
9 16 5 0
11 12 10 0
19 5 9 0
23 28 1 1
28 18 0 1

#print size of each setprint(train.shape, test.shape)
```

(800, 3) (200, 3)

From the output we can see that two sets have been created:

Training set: 800 rows and 3 columnsTesting set: 200 rows and 3 columns

Note that frac controls the percentage of observations from the original DataFrame that will belong to the training set and the random_state value makes the split reproducible.

Additional Resources

The following tutorials explain how to perform other common tasks in Python:

How to Calculate Balanced Accuracy in Python