How do I create test and train samples from one dataframe with pandas?

I have a fairly large dataset in the form of a dataframe and I was wondering how I would be able to split the dataframe into two random samples (80% and 20%) for training and testing.

Thanks!

python python-2.7 pandas dataframe

asked Jun 10 '14 at 17:24
tooty44
802 2 11 30

12 Answers

I would just use numpy's `randn` :

```
In [11]: df = pd.DataFrame(np.random.randn(100, 2))

In [12]: msk = np.random.rand(len(df)) < 0.8

In [13]: train = df[msk]

In [14]: test = df[~msk]
```

And just to see this has worked:

```
In [15]: len(test)
Out[15]: 21

In [16]: len(train)
Out[16]: 79
```

edited Jun 11 '14 at 0:30

answered Jun 10 '14 at 17:29
Andy Hayden
103k 22 260 293

Since `msk` returns an array of bools, perhaps `df.iloc` should be `df.loc` lest True/False be treated as 1,0 indices. – unutbu Jun 10 '14 at 17:37

@unutbu hmmmmmm good point, I was thinking the same about the loc ambiguity (if they are labelled with 0 or 1... maybe best not to use at all? – Andy Hayden Jun 10 '14 at 17:51

2 Sorry, my mistake. As long as `msk` is of dtype `bool` , `df[msk]` , `df.iloc[msk]` and `df.loc[msk]` always return the same result. – unutbu Jun 10 '14 at 18:32

2 I think you should use `rand` to `< 0.8` make sense because it returns uniformly distributed random numbers between 0 and 1. – Rolando Jun 10 '14 at 18:43

1 @user3712008: this doesn't convert anything into a numpy array, but rather uses numpy to create the mask for indices. the construction of the dataframe at the start uses numpy as well, but that is incidental – watsonic Jan 20 '16 at 19:38

|

SciKit Learn's `train_test_split` is a good one.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

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

edited Nov 7 '16 at 10:27
Peque
3,353 2 20 41

answered Jun 10 '14 at 22:19
gobrewers14
4,553 7 18 36

12 This will return numpy arrays and not Pandas Dataframes however – Bar Oct 22 '14 at 15:10

30 Btw, it does return a Pandas Dataframe now (just tested on Sklearn 0.16.1) – Julien Marrec Jul 8 '15 at 10:30

2 If you're looking for KFold, its a bit more complex sadly. `kf = KFold(n, n_folds=folds) for train_index, test_index in kf: X_train, X_test = X.ix[train_index], X.ix[test_index]` see full example here: quantstart.com/articles/… – ihadanny Feb 23 '16 at 13:13

4 In new versions (0.18, maybe earlier), import as `from sklearn.model_selection import train_test_split` instead. – Mark Oct 19 '16 at 17:24

See official docs here – Noel Evans Nov 7 '16 at 21:08

|

Pandas random sample will also work

```
train=df.sample(frac=0.8,random_state=200)
test=df.drop(train.index)
```

answered Feb 21 '16 at 1:28
ParagM
949 5 17

5 This seems to me as even more cleaner way how to do that than current top answer. It's shorter and clearer. – kotrfa Apr 21 '16 at 12:27

What does .index mean / where is the documentation for .index on a DataFrame? I can't find it. – dmonopoly Feb 13 at 16:47

@dmonopoly, it is exactly what it looks like. df.index retruns index object of that dataframe. pandas.pydata.org/pandas-docs/stable/generated/… also some discussion at stackoverflow.com/questions/17241004/… – ParagM Feb 14 at 3:28

I would use scikit-learn's own training_test_split, and generate it from the index

```
from sklearn.cross_validation import train_test_split


y = df.pop('output')
X = df

X_train,X_test,y_train,y_test = train_test_split(X.index,y,test_size=0.2)
X.iloc[X_train] # return dataframe train
```

edited Oct 13 '15 at 11:11

answered May 26 '15 at 9:33
Napitupulu Jon
1,216 1 10 16

1 The `cross_validation` module is now deprecated: `DeprecationWarning: This module was deprecated in version 0.18 in favor of the model_selection module into which all the refactored classes and functions are moved. Also note that the interface of the new CV iterators are different from that of this module. This module will be removed in 0.20.` – Harry Nov 5 '16 at 23:23

You may also consider stratified division into training and testing set. Startified division also generates training and testing set randomly but in such a way that original class proportions are preserved. This makes training and testing sets better reflect the properties of the original dataset.

```
import numpy as np

def get_train_test_inds(y,train_proportion=0.7):
    '''Generates indices, making random stratified split into training set and testing
sets
    with proportions train_proportion and (1-train_proportion) of initial sample.
    y is any iterable indicating classes of each observation in the sample.
    Initial proportions of classes inside training and
    testing sets are preserved (stratified sampling).
    '''

    y=np.array(y)
    train_inds = np.zeros(len(y),dtype=bool)
    test_inds = np.zeros(len(y),dtype=bool)
    values = np.unique(y)
    for value in values:
        value_inds = np.nonzero(y==value)[0]
        np.random.shuffle(value_inds)
        n = int(train_proportion*len(value_inds))

        train_inds[value_inds[:n]]=True
        test_inds[value_inds[n:]]=True

    return train_inds,test_inds
```

df[train_inds] and df[test_inds] give you the training and testing sets of your original DataFrame df.

answered Dec 10 '14 at 23:11
Apogentus
1,989 14 22

This is the preferable strategy for supervised learning tasks. – vincentmajor Mar 2 at 19:16

When trying to use this I am getting an error. ValueError: assignment destination is read-only in the line "np.random.shuffle(value_inds)" – Markus W Mar 17 at 18:25

This is what I wrote when I needed to split a DataFrame. I considered using Andy's approach above, but didn't like that I could not control the size of the data sets exactly (i.e., it would be sometimes 79, sometimes 81, etc.).

```
def make_sets(data_df, test_portion):
    import random as rnd

    tot_ix = range(len(data_df))
    test_ix = sort(rnd.sample(tot_ix, int(test_portion * len(data_df))))
    train_ix = list(set(tot_ix) ^ set(test_ix))

    test_df = data_df.ix[test_ix]
    train_df = data_df.ix[train_ix]

    return train_df, test_df


train_df, test_df = make_sets(data_df, 0.2)
test_df.head()
```

edited Dec 25 '14 at 20:59

answered Dec 25 '14 at 20:52
Anarcho-Chossid
497 6 20

There are many valid answers. Adding one more to the bunch. from sklearn.cross_validation import train_test_split

```
#gets a random 80% of the entire set
X_train = X.sample(frac=0.8, random_state=1)
#gets the left out portion of the dataset
X_test = X.loc[~df_model.index.isin(X_train.index)]
```

answered Dec 9 '16 at 22:18
Abhi
133 3 12

You can make use of df.as_matrix() function and create Numpy-array and pass it.

```
Y = df.pop()
X = df.as_matrix()
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2)
model.fit(x_train, y_train)
model.test(x_test)
```

answered Nov 27 '15 at 8:50
kiran6
225 3 13

I think you also need to a get a copy not a slice of dataframe if you wanna add columns later.

```
msk = np.random.rand(len(df)) < 0.8
train, test = df[msk].copy(deep = True), df[~msk].copy(deep = True)
```

answered Aug 4 '15 at 4:16
Hakim
422 1 4 15

How about this? df is my dataframe

```
total_size=len(df)

train_size=math.floor(0.66*total_size) (2/3 part of my dataset)

#training dataset
train=df.head(train_size)
```

```
#test dataset
test=df.tail(len(df) -train_size)
```

answered Oct 13 '16 at 16:34

Akash Jain
1 1

If your wish is to have one dataframe in and two dataframes out (not numpy arrays), this should do the trick:

```
def split_data(df, train_perc = 0.8):

   df['train'] = np.random.rand(len(df)) < train_perc

   train = df[df.train == 1]

   test = df[df.train == 0]

   split_data ={'train': train, 'test': test}

   return split_data
```

answered Jul 19 '15 at 21:29

Johnny V
71 1 4

I have the same issue in c++, I do not know how to split the matrix of images to train and test. Does anybody have experience in this?

deleted by owner Jan 2 '16 at 22:56

answered Jan 2 '16 at 22:28

ga97rasl
39 11

This does not really answer the question. If you have a different question, you can ask it by clicking Ask Question. You can also add a bounty to draw more attention to this question once you have enough reputation. - From Review – elo80ka Jan 2 '16 at 22:45