# Normalize columns of pandas data frame

130

41

I have a dataframe in pandas where each column has different value range. For example:

df:

```
A     B   C
1000  10  0.5
765   5   0.35
800   7   0.09
```

Any idea how I can normalize the columns of this dataframe where each value is between 0 and 1?

My desired output is:

```
A     B    C
1     1    1
0.765 0.5  0.7
0.8   0.7  0.18(which is 0.09/0.5)
```

python pandas normalize

edited Feb 2 '18 at 15:03

asked Oct 16 '14 at 22:24

ahajib
**3,059** 13 47 89

1 there is an apply function, e.g. frame.apply(f, axis=1) where f is a function that does something with a row... – tschm Oct 16 '14 at 22:30

1 Normalization might not be the most appropriate wording, since scikit-learn documentation defines it as "the process of **scaling individual samples to have unit norm**" (i.e. row by row, if I get it correctly). – Skippy le Grand Gourou Mar 5 at 16:58

I do not get it, why min_max scaling is considered normalization! normal has got to have meaning in the sense of normal distribution with mean zero and variance 1. – No Lie Apr 21 at 2:21

## 13 Answers

You can use the package sklearn and its associated preprocessing utilities to normalize the data.

125

```
from sklearn import preprocessing

x = df.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
df = pandas.DataFrame(x_scaled)
```

For more information look at the scikit-learn documentation on preprocessing data: scaling features to a range.

edited Dec 8 '16 at 20:25
Praveen
3,840 2 25 48

answered Oct 16 '14 at 23:34
Sandman
2,020 4 15 19

24 i think this will get rid of the column names, which might be one of the reasons op is using dataframes in the first place. – pietz Jan 16 '17 at 21:02

25 This will normalize the rows and not the columns, unless you transpose it first. To do what the Q asks for: `pd.DataFrame(min_max_scaler.fit_transform(df.T), columns=df.columns, index=df.index)` – hobs Jan 20 '17 at 23:47

14 @pietz to keep column names, see this post. Basically replace the last line with , `df=pandas.DataFrame(x_scaled, columns=df.columns)` – ijoseph Jun 26 '17 at 18:52

1 @hobs This is not correct. Sandman's code normalizes column-wise and per-column. You get the wrong result if you transpose. – petezurich Apr 1 '18 at 14:10

4 @petezurich It looks like Sandman or Praveen corrected their code. Unfortunately, it's not possible to correct comments ;) – hobs Apr 3 '18 at 21:25

238

one easy way by using **Pandas**: (here I want to use mean normalization)

```
normalized_df=(df-df.mean())/df.std()
```

to use min-max normalization:

```
normalized_df=(df-df.min())/(df.max()-df.min())
```

edited Jan 19 '17 at 10:26

answered Jan 8 '17 at 11:25
Cina
3,111 3 11 23

12 i like this one. it's short, it's expressive and it preserves the header information. but i think you need to subtract the min in the denominator as well. – pietz Jan 16 '17 at 20:49

3 thank you for your comment. I edit the denominator. – Cina Jan 19 '17 at 10:27

This solution is beautiful, concise, and wrong. The mean() and str() methods don't return a dataframe, but a series. That produces ridiculous error messages ("ValueError: cannot reindex from a duplicate axis") To get something that works, you have to uglify it in the following way: normalized_df=(df-df.mean().to_frame().T)/df.std().to_frame().T – MightyCurious Oct 6 '18 at 18:40

5 I don't think it's wrong. Works beautifully for me - I don't think mean() and std() need to return a dataframe in order for this to work and your error message does not imply that them not being a dataframe is a problem. – Strandtasche Oct 23 '18 at 9:19

this is not column wise normalization. this is normalizing entire matrix as whole which will provide wrong results. – Nguai al Apr 26 at 23:19

37

Based on this post: https://stats.stackexchange.com/questions/70801/how-to-normalize-data-to-0-1-range

You can do the following:

```
def normalize(df):
    result = df.copy()
    for feature_name in df.columns:
        max_value = df[feature_name].max()
        min_value = df[feature_name].min()
        result[feature_name] = (df[feature_name] - min_value) / (max_value -
min_value)
    return result
```

You don't need to stay worrying about whether your values are negative or positive. And the values should be nicely spread out between 0 and 1.

edited Apr 13 '17 at 12:44
Community ♦
1 1

answered Apr 15 '15 at 13:25

Michael Aquilina
3,223 1 25 33

Be careful when min and max values are same, your denominator is 0 and you will get a NaN value. – Hrushikesh Dhumal Feb 1 at 6:02

23

If you like using the sklearn package, you can keep the column and index names by using pandas `loc` like so:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaled_values = scaler.fit_transform(df)
df.loc[:,:] = scaled_values
```

answered Apr 21 '17 at 15:06

j sad
406 5 9

Your problem is actually a simple transform acting on the columns:

19

```
def f(s):
    return s/s.max()

frame.apply(f, axis=0)
```

Or even more terse:

```
frame.apply(lambda x: x/x.max(), axis=0)
```

edited Feb 22 '17 at 13:52

answered Oct 17 '14 at 9:57

tschm

**1,340** 2 19 32

1 The `lambda` one is the best :-) – Abu Shoeb Dec 8 '18 at 23:49

isn't this supposed to be axis=1 since the question is column wise normalization? – Nguai al Apr 26 at 23:27

No, I don't think so. axis=1 would run f over the rows. To run f across each column you have to state axis=0 or don't specify it at all, – tschm Apr 28 at 13:20

---

Simple is Beautiful:

16

```
df["A"] = df["A"] / df["A"].max()
df["B"] = df["B"] / df["B"].max()
df["C"] = df["C"] / df["C"].max()
```

answered Feb 6 '18 at 20:03

Basil Musa

**3,835** 5 43 47

Great and in my opinion the best solution ! – Maciej A. Bednarz Apr 16 '18 at 7:44

2 Note, that OP asked for [0..1] range and this solution scales to [-1..1] range. Try this with the array [-10, 10]. – Alexander Sosnovshchenko Apr 28 '18 at 9:20

1 @AlexanderSosnovshchenko not really. Basil Musa is assuming the OP's matrix is always non-negative, that's why he has given this solution. If some column has a negative entry then this code does NOT normalize to the [-1,1] range. Try it with the array [-5, 10]. The correct way to normalize to [0,1] with negative values was given by Cina's answer `df["A"] = (df["A"]-df["A"].min()) / (df["A"].max()-df["A"].min())` – facuq Nov 9 '18 at 13:24

simple AND explicit – joshi123 Dec 12 '18 at 16:17

---

**You can create a list of columns that you want to normalize**

13

```
column_names_to_normalize = ['A', 'E', 'G', 'sadasdsd', 'lol']
x = df[column_names_to_normalize].values
x_scaled = min_max_scaler.fit_transform(x)
```

```
df_temp = pd.DataFrame(x_scaled, columns=column_names_to_normalize, index =
df.index)
df[column_names_to_normalize] = df_temp
```

***Your Pandas Dataframe is now normalized only at the columns you want***

**However**, if you want the **opposite**, select a list of columns that you **DON'T** want to normalize, you can simply create a list of all columns and remove that non desired ones

```
column_names_to_not_normalize = ['B', 'J', 'K']
column_names_to_normalize = [x for x in list(df) if x not in
column_names_to_not_normalize ]
```

edited Sep 29 '18 at 22:10

answered Sep 29 '18 at 21:34

raullalves

**446** 3 16

---

8

I think that a better way to do that in pandas is just

```
df = df/df.max().astype(np.float64)
```

**Edit** If in your data frame negative numbers are present you should use instead

```
df = df/df.loc[df.abs().idxmax()].astype(np.float64)
```

edited Oct 24 '14 at 17:52

answered Oct 17 '14 at 13:58

Daniele

**334** 1 10

1 In case all values of a column are zero this won't work – ahajib Sep 2 '15 at 23:23

dividing the current value by the max will not give you a correct normalisation unless the min is 0. – pietz Jan 16 '17 at 21:16

I agree, but that is what the OT was asking for (see his example) – Daniele Feb 21 '17 at 14:33

---

6

The solution given by Sandman and Praveen is very well. The only problem with that if you have categorical variables in other columns of your data frame this method will need some adjustments.

My solution to this type of issue is following:

```
from sklearn import preprocesing
x = pd.concat([df.Numerical1, df.Numerical2,df.Numerical3])
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
x_new = pd.DataFrame(x_scaled)
df = pd.concat([df.Categoricals,x_new])
```

edited Nov 26 '17 at 21:33

answered Nov 26 '17 at 20:09

cyber-math

**129** 1 6

1 This answer is useful because most examples on the internet apply one scaler to all the columns, whereas this actually addresses the situation where one scaler, say the MinMaxScaler, should not apply to all columns. – demongolem Sep 10 '18 at 17:07

3

You might want to have some of columns being normalized and the others be unchanged like some of regression tasks which data labels or categorical columns are unchanged So I suggest you this pythonic way (It's a combination of @shg and @Cina answers ):

```
features_to_normalize = ['A', 'B', 'C']
# could be ['A','B']

df[features_to_normalize] = df[features_to_normalize].apply(lambda x:(x-x.min())
/ (x.max()-x.min()))
```

answered Apr 25 at 20:33

Masoud Masoumi Moghadam

**334** 4 23

1

```
def normalize(x):
    try:
        x = x/np.linalg.norm(x,ord=1)
        return x
    except :
        raise
data = pd.DataFrame.apply(data,normalize)
```

From the document of pandas,DataFrame structure can apply an operation (function) to itself .

```
DataFrame.apply(func, axis=0, broadcast=False, raw=False, reduce=None, args=(),
**kwds)
```

> Applies function along input axis of DataFrame. Objects passed to functions are Series objects having index either the DataFrame's index (axis=0) or the columns (axis=1). Return type depends on whether passed function aggregates, or the reduce argument if the DataFrame is empty.

You can apply a custom function to operate the DataFrame .

edited Apr 14 '18 at 2:49

answered Apr 13 '18 at 9:21

shg

**16** 4

2 It would be good to explain, why your code solves the OPs problem, so people can adapt the strategy rather

than just copy your code. Please read How do I write a good answer? – Mr. T Apr 13 '18 at 9:53

---

1

The following function calculates the Z score:

```
def standardization(dataset):
  """ Standardization of numeric fields, where all values will have mean of zero
  and standard deviation of one. (z-score)

  Args:
    dataset: A `Pandas.Dataframe`
  """
  dtypes = list(zip(dataset.dtypes.index, map(str, dataset.dtypes)))
  # Normalize numeric columns.
  for column, dtype in dtypes:
      if dtype == 'float32':
          dataset[column] -= dataset[column].mean()
          dataset[column] /= dataset[column].std()
  return dataset
```

edited Jan 19 at 1:11

answered Jan 18 at 22:55

user260826

**3,758** 2 40 75

---

0

You can do this in one line

```
DF_test = DF_test.sub(DF_test.mean(axis=0), axis=1)/DF_test.mean(axis=0)
```

it takes mean for each of the column and then subtracts it(mean) from every row(mean of particular column subtracts from its row only) and divide by mean only. Finally, we what we get is the normalized data set.

edited Apr 12 at 6:39

answered Apr 12 at 6:13

Rishi Bansal

**882** 1 2 19

---

**protected** by eyllanesc Apr 14 '18 at 2:51

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?