Data Preprocessing with Python Pandas — Part 3 Normalisation

Angelica Lo Duca Nov 30, 2020 · 5 min read ★

Image by mohamed Hassan from Pixabay

This tutorial explains how to preprocess data using the Pandas library. Preprocessing is the process of doing a pre-analysis of data, in order to transform them into a standard

and normalised format. Preprocessing involves the following aspects:

- missing values
- data formatting
- data normalisation
- data standardisation
- data binning

In this tutorial we deal only with normalisation. In my previous tutorials I dealt with missing values and data formatting.

Data Normalisation involves adjusting values measured on different scales to a common scale. When dealing with dataframes, data normalization permits to adjust values referred to different columns to a common scale. This operation is strongly recommended when the columns of a dataframe are considered as input features of a machine learning algorithm, because it permits to give all the features the same weight.

Normalization applies only to columns containing numeric values. Five methods of normalization exist:

- single feature scaling
- min max
- z-score
- log scaling
- clipping

In the remainder of the tutorial, we apply each method to a single column. However, if you wanted to use each column of the dataset as input features of a machine learning algorithm, you should apply the same normalisation method to all the columns.

In this tutorial, we use the `pandas` library to perform normalization. As an alternative, you could use the preprocessing methods of the `scikit-learn` libray. **A little note for readers**: if you wanted to learn how to use the preprocessing package of `scikit-learn`, please drop me a message or a comment to this post :)

You can download the source code of this tutorial as a Jupyter notebook from my Github Data Science Repository.

Data Import

As example dataset, in this tutorial we consider the dataset provided by the Italian Protezione Civile, related to the number of COVID-19 cases registered since the beginning of the COVID-19 pandemic. The dataset is updated daily and can be downloaded from this link.

First of all, we need to import the Python `pandas` library and read the dataset through the `read_csv()` function. Then we can drop all the columns with `NaN` values. This is done through `dropna()` function.

```
import pandas as pd

df = pd.read_csv('https://raw.githubusercontent.com/pcm-dpc/COVID-19/master/dati-regioni/dpc-covid19-ita-regioni.csv')
df.dropna(axis=1,inplace=True)
df.head(10)
```


Single Feature Scaling

Single Feature Scaling converts every value of a column into a number between 0 and 1. The new value is calculated as the current value divided by the max value of the column. For example, if we consider the column `tamponi`, we can apply the single feature scaling by applying to the column the function `max()`, whic calculates the maximum value of the column:

```
df['tamponi'] = df['tamponi']/df['tamponi'].max()
```

Min Max

Similarly to Single Feature Scaling, Min Max converts every value of a column into a number between 0 and 1. The new value is calculated as the difference between the current value and the min value, divided by the range of the column values. For example, we can apply the min max method to the column `totale_casi`.

```
df['totale_casi'] = (df['totale_casi'] -
df['totale_casi'].min())/(df['totale_casi'].max() -
df['totale_casi'].min())
```

z-score

Z-Score converts every value of a column into a number around 0. Typical values obtained by a z-score transformation range from -3 and 3. The new value is calculated as the difference between the current value and the average value, divided by the standard deviation. The average value of a column can be obtained through the `mean()` function, while the standard deviation through the `std()` function. For example, we can calculate the z-score of the column `deceduti`.

```
df['deceduti'] = (df['deceduti']-
df['deceduti'].mean())/df['deceduti'].std()
```

Now we can calculate the minimum and maximum value obtained by the z-score transformation:

```
df['deceduti'].min()
```

which gives the following output:

```
-0.4329770144818199
```

and:

```
df['deceduti'].max()
```

which gives the following output:

```
5.945028962275545
```

Log Scaling

Log Scaling involves the conversion of a column to the logarithmic scale. If we want to use the natural logarithm, we can use the `log()` function of the `numpy` library. For example, we can apply log scaling to the column `dimessi_guariti`. We must deal with `log(0)` because it does not exist. We use the `lambda` operator to select the single rows of the column.

```
import numpy as np
df['dimessi_guariti'] = df['dimessi_guariti'].apply(lambda x:
np.log(x) if x != 0 else 0)
```

which gives the following output:

```
0        0.000000
1        0.000000
2        0.000000
3        0.000000
4        0.000000
          ...
5812     9.846388
5813    10.794296
5814     9.474088
5815     8.372861
5816    10.922389
Name: dimessi_guariti, Length: 5817, dtype: float64
```

Clipping

Clipping involves the capping of all values below or above a certain value. Clipping is useful when a column contains some outliers. We can set a maximum `vmax` and a minimum value `vmin` and set all outliers greater than the maximum value to `vmax` and

all the outliers lower than the minimum value to `vmin`. For example, we can consider the column `ricoverati_con_sintomi` and we can set `vmax = 10000` and `vmin = 10`.

```
vmax = 10000
vmin = 10
df['ricoverati_con_sintomi'] =
df['ricoverati_con_sintomi'].apply(lambda x: vmax if x > vmax else
vmin if x < vmin else x)
```

Summary

In this tutorial, I have shown you the different techniques used to perform data normalisation: single feature scaling: min max, z-score, log scaling, clipping. Thus, the question is: what is the best technique? Actually, there is not a technique better than the others, the choice of a method rather than another depends on what we want as output. Thus:

- if you want an output between 0 and 1, you should use single feature scaling or min max
- if you want an output around 0, which includes also negative values, you should use z-score
- if your data contain many outliers, you can use clipping
- if you want to change the scale of your data, you can use the log scaling.

If you would like to learn about the other aspects of data preprocessing, such as data binning and data standardisation, stay tuned…

If you wanted to be updated on my research and other activities, you can follow me on Twitter, Youtube and Github.

Data Analysis Data Science Python Pandas Normalization