(ML-I/P)
MS. SHWETA TIWARI
September 7 , 2022

ML-Implementation/Practice Machine Learning

By SHWETA TIWARI

CSV: Pandas Basics (Reading Data Files, DataFrames, Data Selection)

Let's start with this ML tutorial!

The first question is:

Pandas Introduction

What is the Pandas?

Pandas is a Python library used for working with data sets.

It has functions for analyzing, cleaning, exploring, and manipulating data.

The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Why Use Pandas?

Pandas allows us to analyze big data and make conclusions based on statistical theories.

Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science.

What Can Pandas Do?

Pandas gives you answers about the data. Like:

- **Is there a correlation between two or more columns?**
- **What is the average value?**
- **Max value?**
- **Min value?**

Pandas are also able to delete rows that are not relevant, or contain wrong values, like empty or NULL values. This is called *cleaning* the data.

How to open data files in pandas

You might have your data in `.csv` files or SQL tables. Maybe Excel files. Or `.tsv` files. Or something else. But the goal is the same in all cases. If you want to analyze that data using pandas, the first step will be to read it into a *data structure* that's compatible with pandas.

Pandas data structures

There are two types of data structures in pandas:

- Series
- DataFrames.

Pandas Series

A pandas Series is a one-dimensional data structure (“*a one-dimensional ndarray*”) that can store values — and for every value, it holds a unique index, too. You can think of it as a *single column* of a bigger table. And it’s just enough if you know this much about Series for now, I’ll get back to it later.

```
In [4]: test_set_series

Out[4]: 0      15
        1      36
        2      41
        3      14
        4      69
        5      73
        6      92
        7      56
        8     101
        9     120
        10    175
        11    191
        12    215
        13    306
        14    241
        15    392
        dtype: int64
```

Pandas Series example

Pandas DataFrame

A pandas *DataFrame* is a two (or more) dimensional data structure – basically a table with rows and columns. The columns have names and the rows have indexes. Compared to a pandas Series (which was one labeled column only), a DataFrame is practically the whole data table. You can think of it as a collection of pandas Series (columns next to each other).

Either way: DataFrame is the primary pandas DataStructure!

```
In [12]: big_table
```

Out[12]:

	user_id	phone_type	source	free	super
0	1000001	android	invite_a_friend	5.0	0.0
1	1000002	ios	invite_a_friend	4.0	0.0
2	1000003	error	invite_a_friend	37.0	0.0
3	1000004	error	invite_a_friend	0.0	0.0
4	1000005	ios	invite_a_friend	6.0	0.0

Pandas DataFrame example

Loading a .csv file into a pandas DataFrame

Okay, time to put things into practice! Let's load a .csv data file into pandas!

There is a function for it, called `read_csv()`.

Start with a simple demo data set, called `zoo`! This time – for the sake of practicing – you will create a .csv file for yourself! Here's the raw data:

```
animal,uniq_id,water_need
elephant,1001,500
elephant,1002,600
elephant,1003,550
tiger,1004,300
tiger,1005,320
tiger,1006,330
tiger,1007,290
tiger,1008,310
zebra,1009,200
zebra,1010,220
zebra,1011,240
zebra,1012,230
zebra,1013,220
zebra,1014,100
zebra,1015,80
lion,1016,420
lion,1017,600
lion,1018,500
lion,1019,390
kangaroo,1020,410
kangaroo,1021,430
kangaroo,1022,410
```

Go back to your Jupyter Home tab and create a new text file...

...then copy-paste the above zoo data into this text file...

... and then rename this text file to zoo.csv!

Okay, this is our .csv file!

Now, go back to your Jupyter Notebook (that I named *ml_tutorial_1*) and open this freshly created .csv file in it!

Again, the function that you have to use for that is read_csv()

Type this to a new cell:

```
pd.read_csv('zoo.csv', delimiter = ',')
```

Step #0 – The data file we'll use in this tutorial

STEP #1 – Download it to the dataset!

STEP #2 – loading the .csv file with .read_csv into a DataFrame

Now, go back again to your Jupyter Notebook and use the same .read_csv() function that we have used before (but don't forget to change the file name and the delimiter value):

```
pd.read_csv('dataset.csv', delimiter=';')
```

Done! The data is loaded into a pandas DataFrame:

Does something feel off? Yes, this time we didn't have a header in our .csv file, so we have to define it manually! Add the names parameter to your .read_csv() function:

pd.read_csv('pandas_tutorial_dataset.csv', delimiter=';', names = ['m', 'e'])

Better!

And with that, we finally loaded our .csv data into a pandas DataFrame!

Note 1: Just so you know, there is an alternative method. (I don't prefer it though.) You can load the .csv data using the URL directly. In this case, the data won't be downloaded to your data server.

Selecting data from a DataFrame in pandas

#1 How to print the whole DataFrame
The most basic method you can do in pandas is to just simply print your whole DataFrame to your screen. Nothing special.

Although it's good to get a grasp on a concept right here at the beginning:

To work with a specific dataset, you don't have to run the pd.read_csv() function again and again and again. You can just store its output into a variable the first time you run it!

#2 How to print a sample of your dataframe (e.g. first 5 rows) – by .head()

Or the last few rows by typing:

.tail()

Or a few random rows by typing:

.sample(5)

#3 How to select specific columns of your DataFrame

This one is a bit tricky — but very often used, so better learn it now! Let's say you want to print the country and the user_id columns only.

You should use this syntax:

article_read[['country', 'user_id']]

By the way, if you change the order of the column names, the order of the returned columns will change, too:

#4 How to filter for specific values in your DataFrame