MSPC Workshop

pandas
Powerful python Data Analysis Toolkit

—

By Hritik Bhandari

INTRODUCTION

Python is an interpreted, high-level, general-purpose programming language. It has many different libraries that are used for different purposes like web dev, machine learning, deep learning, data analysis, visualization, etc.

Pandas is basically one such python library. It is an open-source Python library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

https://pandas.pydata.org/index.html

Key Features of Pandas

- Fast and efficient DataFrame object with the default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, indexing, and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High-performance merging and joining of data.
- Time Series functionality.

Installing pandas

Using pip install pandas

We can use the above command to download pandas through command line or we can prefer downloading the library directly from the navigator.

Importing pandas in our code

To use pandas in our code, we will need to import it using the given command

```
import pandas as pd
```

Working with pandas

A pandas DataFrame can be created using various inputs like −

- Array
- Dict
- Importing from files like json, csv, etc.

A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

Features of DataFrame

- Potentially, columns are of different types
- Size – Mutable

- Labeled axes (rows and columns)
- Can Perform Arithmetic operations on rows and columns

Data from a csv file

A comma-separated values (csv) file is a delimited text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas.

Data from a Dictionary

Dictionary in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key: value pair. Key-value is provided in the dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon :, whereas each key is separated by a 'comma'.

https://www.geeksforgeeks.org/python-dictionary/

Exporting Data

Using df.to_csv()

Say that you want to export pandas DataFrame to a CSV file. How would you go about it?

Simply use the given command

```
out_csv = 'df.csv'

df.to_csv(out_csv)
```

You can also provide the required address by using df.to_csv(r'.....')

Now, we will work on the iris dataset using pandas

- ➔ **type()**- type() method returns the class type of the argument(object) passed as parameter. type() function is mostly used for debugging purposes.
- ➔ **df.shape**- The shape attribute for NumPy arrays returns the dimensions of the array. If Y has n rows and m columns, then Y.shape is (n,m).
- ➔ **df.dtypes**- This returns a Series with the data type of each column. The result's index is the original DataFrame's columns
- ➔ **df.info()**- dataframe.info() function is used to get a concise summary of the dataframe. It comes really handy when doing an exploratory analysis of the data. To get a quick overview of the dataset we use the dataframe.info() function.
- ➔ **print()**
- ➔ **Setting column Names** -
    - ◆ First creating a list
    - ◆ Then, df.columns = col_name
- ➔ **df.head()**- This function returns the first n rows for the object based on position. It is useful for quickly testing if your object has the right type of data in it.
    - We can specify the number of rows to view by giving the argument.
- ➔ **df.tail()**- This function returns last n rows from the object based on position. It is useful for quickly verifying data, for example, after sorting or appending rows.
    - We can specify the number of rows to be viewed from the bottom.
- ➔ **df.iloc[: , :]**- Purely integer-location based indexing for selection by position. .iloc[] is primarily integer position based (from 0 to length-1 of the axis), but may also be used with a boolean array.

Statistical Data Analysis

➔ **df.describe()** - Generate descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding NaN values. Analyzes both numeric and object series, as well as DataFrame column sets of mixed data types. The output will vary depending on what is provided. Refer to the notes below for more detail.

➔ **df.count()** - Count non-NA cells for each column or row. The values None, NaN, NaT, and optionally numpy.inf (depending on pandas.options.mode.use_inf_as_na) are considered NA.

- Also we can give columns specifically.
- iris['Sepal length'].count()

➔ **df.mean()** - Average of values of all the columns.

➔ **df.std()**- In statistics, the standard deviation is a measure of the amount of variation or dispersion of a set of values. A low standard deviation indicates that the values tend to be close to the mean of the set, while a high standard deviation indicates that the values are spread out over a wider range.

- Pandas dataframe.std() function return sample standard deviation over requested axis. By default, the standard deviations are normalized by N-1. It is a measure that is used to quantify the amount of variation or dispersion of a set of data values.

➔ **df.median()** - Pandas dataframe.median() function return the median of the values for the requested axis. If the method is applied on a pandas series object, then the method returns a scalar value which is the median value of all the observations in the dataframe. If the method is applied on a pandas dataframe object, then the method returns a pandas series object which contains the median of the values over the specified axis.

➔ **Ranges**- **df.min()** ‘OR’ **df.max** can be used to get the minimum and maximum values respectively for every column in the dataset.

Missing Values

Missing Data can occur when no information is provided for one or more items or for a whole unit. Missing Data is a very big problem in real-life scenarios. Missing Data can also refer to as NA(Not Available) values in pandas. In DataFrame sometimes many datasets simply arrive with missing data, either because it exists and was not collected or it never existed

In Pandas missing data is represented by two value:

- None: None is a Python singleton object that is often used for missing data in Python code.
- NaN: NaN (an acronym for Not a Number), is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation

df.isnull()

df.fillna()

We’ll learn more about missing values and analyzing time series data in python in the coming workshops.

Conclusion

This was my time guys, we have successfully completed how to analyze data using pandas.

Now my colleague will take you forward with visualizing datasets in python.