Pandas for Machine Learning

AcadView

May 20, 2018

---

1 Overview

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures.
Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

- Fast and efficient DataFrame object with 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 date 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.

2 Pandas Data structures

Pandas deals with the following three data structures

- Series

- DataFrame
- Panel

These data structures are built on top of Numpy array, which means they are fast.

Data Structure	Dimensions	Description
Series	1	1D labeled homogeneous array, sizeimmutable.
Data Frames	2	General 2D labeled, size-mutable tabular structure with potentially heterogeneously typed columns.
Panel	3	General 3D labeled, size-mutable array.

Mutability All Pandas data structures are value mutable (can be changed) and except Series all are size mutable. Series is size immutable.
Note: DataFrame is widely used and one of the most important data structures. Panel is used much less.

2.1 Series

Series is a one-dimensional array like structure with homogeneous data. For example, the following series is a collection of integers 10, 23, 56,

10	23	56	17	52	61	73	90	26	72

2.2 DataFrame

DataFrame is a two-dimensional array with heterogeneous data. For example,

Name	Age	Gender	Rating
Steve	32	Male	3.45
Lia	28	Female	4.6
Vin	45	Male	3.9
Katie	38	Female	2.78

The table represents the data of a sales team of an organization with their overall performance rating. The data is represented in rows and columns. Each column represents an attribute and each row represents a person.
The data types of the four columns are as follows:

Column	Type
Name	String
Age	Integer
Gender	String
Rating	Float

3 Pandas DataFrame

3.1 Creating a DataFrame

Data frames are the central concept in pandas. In essence, a data frame is table with labeled rows and columns. Data frames can be created from multiple sources - e.g. CSV files, excel files, and JSON.

3.1.1 Hardcoded Dataframes

Hardcoded data frames can be constructed by providing a hash of columns and their values.

3.1.2 Loading CSV files

Loading a CSV file as a data frame is pretty easy:
Let's say you have the following table in form of csv in the system as "example.csv".

	first_name	last_name	age	preTestScore	postTestScore
0	Jason	Miller	42	4	25,000
1	Molly	Jacobson	52	24	94,000
2	Tina	.	36	31	57
3	Jake	Milner	24	.	62
4	Amy	Cooze	73	.	70

Loading the data from the csv using pd.read csv() function:

```
df = pd.read_csv('pandas_dataframe_importing_csv/example.csv')
df
```

	Unnamed: 0	first_name	last_name	age	preTestScore	postTestScore
0	0	Jason	Miller	42	4	25,000
1	1	Molly	Jacobson	52	24	94,000
2	2	Tina	.	36	31	57
3	3	Jake	Milner	24	.	62
4	4	Amy	Cooze	73	.	70

Load a csv with no headers:

```
df = pd.read_csv('pandas_dataframe_importing_csv/example.csv', header=None)
df
```

	0	1	2	3	4	5
0	NaN	first_name	last_name	age	preTestScore	postTestScore
1	0.0	Jason	Miller	42	4	25,000
2	1.0	Molly	Jacobson	52	24	94,000
3	2.0	Tina	.	36	31	57
4	3.0	Jake	Milner	24	.	62
5	4.0	Amy	Cooze	73	.	70

3.1.3 Create a DataFrame from Dict of ndarrays / Lists

All the ndarrays must be of same length. If index is passed, then the length of the index should equal to the length of the arrays.
If no index is passed, then by default, index will be range(n), where n is the array length.
Example

```
import pandas as pd
data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]}
df = pd.DataFrame(data)
print df
```

Its output is as follows:

```
   Age   Name
0   28    Tom
1   34   Jack
2   29  Steve
3   42  Ricky
```

Note:Observe the values 0,1,2,3. They are the default index assigned to each using the function range(n).
Example
Let us now create an indexed DataFrame using arrays.

```
import pandas as pd
data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]}
df = pd.DataFrame(data, index=['rank1','rank2','rank3','rank4'])
print df
```

Its output is as follows:

```
        Age    Name
rank1    28     Tom
rank2    34    Jack
rank3    29   Steve
rank4    42   Ricky
```

3.2 DataFrame Basic Functionality

Let us now understand what DataFrame Basic Functionality is. The following tables lists down the important attributes or methods that help in DataFrame Basic Functionality.

S.No.	Attribute or Method	Description
1	T	Transposes rows and columns.
2	axes	Returns a list with the row axis labels and column axis labels as the only members.
3	dtypes	Returns the dtypes in this object.
4	empty	True if NDFrame is entirely empty [no items]; if any of the axes are of length 0.
5	ndim	Number of axes / array dimensions.
6	shape	Returns a tuple representing the dimensionality of the DataFrame.
7	size	Number of elements in the NDFrame.
8	values	Numpy representation of NDFrame.
9	head()	Returns the first n rows.
10	tail()	Returns last n rows.

Let us now create a DataFrame and see how some of the above mentioned attributes operate.

Example:

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']),
   'Age':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

#Create a DataFrame
df = pd.DataFrame(d)
print ("Our data series is:")
print df
```

Its output is as follows:

```
Our data series is:
    Age   Name    Rating
0   25    Tom     4.23
1   26    James   3.24
2   25    Ricky   3.98
3   23    Vin     2.56
4   30    Steve   3.20
5   29    Smith   4.60
6   23    Jack    3.80
```

- **T (Transpose)**

  Returns the transpose of the DataFrame. The rows and columns will interchange.

```
import pandas as pd
import numpy as np

# Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']),
   'Age':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

# Create a DataFrame
df = pd.DataFrame(d)
print ("The transpose of the data series is:")
print df.T
```

Its output is as follows:

```
The transpose of the data series is:
         0     1       2      3      4      5       6
Age      25    26      25     23     30     29      23
Name     Tom   James   Ricky  Vin    Steve  Smith   Jack
Rating   4.23  3.24    3.98   2.56   3.2    4.6     3.8
```

- **axes**

Returns the list of row axis labels and column axis labels.

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']),
   'Age':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

#Create a DataFrame
df = pd.DataFrame(d)
print ("Row axis labels and column axis labels are:")
print df.axes
```

Its output is as follows:

```
Row axis labels and column axis labels are:


[RangeIndex(start=0, stop=7, step=1), Index([u'Age', u'Name', u'Rating'],
dtype='object')]
```

- **dtypes**

Returns the data type of each column.

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']),
   'Age':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

#Create a DataFrame
df = pd.DataFrame(d)
print ("The data types of each column are:")
print df.dtypes
```

Its output is as follows:

```
The data types of each column are:
Age     int64
Name    object
Rating  float64
dtype: object
```

- **shape**

Returns a tuple representing the dimensionality of the DataFrame. Tuple (a,b), where a represents the number of rows and b represents the number of columns.

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']),
   'Age':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

#Create a DataFrame
df = pd.DataFrame(d)
print ("Our object is:")
print df
print ("The shape of the object is:")
print df.shape
```

Its output is as follows:

```
Our object is:
   Age   Name    Rating
0  25    Tom     4.23
1  26    James   3.24
2  25    Ricky   3.98
3  23    Vin     2.56
4  30    Steve   3.20
5  29    Smith   4.60
6  23    Jack    3.80

The shape of the object is:
(7, 3)
```

- **values**

Returns the actual data in the DataFrame as an NDarray.

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']),
   'Age':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

#Create a DataFrame
df = pd.DataFrame(d)
print ("Our object is:")
print df
print ("The actual data in our data frame is:")
print df.values
```

Its output is as follows:

```
Our object is:
    Age   Name    Rating
0   25    Tom     4.23
1   26    James   3.24
2   25    Ricky   3.98
3   23    Vin     2.56
4   30    Steve   3.20
5   29    Smith   4.60
6   23    Jack    3.80
The actual data in our data frame is:
[[25 'Tom' 4.23]
[26 'James' 3.24]
[25 'Ricky' 3.98]
[23 'Vin' 2.56]
[30 'Steve' 3.2]
[29 'Smith' 4.6]
[23 'Jack' 3.8]]
```

- **Head & Tail**

To view a small sample of a DataFrame object, use the head() and tail() methods. head() returns the first n rows (observe the index values). The default number of elements to display is five, but you may pass a custom number.

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']),
   'Age':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

#Create a DataFrame
df = pd.DataFrame(d)
print ("Our data frame is:")
print df
print ("The first two rows of the data frame is:")
print df.head(2)
```

Its output is as follows:

```
Our data frame is:
    Age   Name    Rating
0   25    Tom     4.23
1   26    James   3.24
2   25    Ricky   3.98
3   23    Vin     2.56
4   30    Steve   3.20
5   29    Smith   4.60
6   23    Jack    3.80

The first two rows of the data frame is:
   Age   Name   Rating
0  25    Tom    4.23
1  26    James  3.24
```

tail() returns the last n rows (observe the index values). The default number of elements to display is five, but you may pass a custom number.

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']),
   'Age':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

#Create a DataFrame
df = pd.DataFrame(d)
print ("Our data frame is:")
print df
print ("The last two rows of the data frame is:")
print df.tail(2)
```

Its output is as follows:

```
Our data frame is:
    Age   Name    Rating
0   25    Tom     4.23
1   26    James   3.24
2   25    Ricky   3.98
3   23    Vin     2.56
4   30    Steve   3.20
5   29    Smith   4.60
6   23    Jack    3.80

The last two rows of the data frame is:
    Age   Name    Rating
5   29    Smith    4.6
6   23    Jack     3.8
```

3.3 Descriptive Statistics

A large number of methods collectively compute descriptive statistics and other related operations on DataFrame. Most of these are aggregations like sum(), mean(), but some of them, like sumsum(), produce an object of the same size. Generally speaking, these methods take an axis argument, just like ndarray.{sum, std, ...}, but the axis can be specified by name or integer
DataFrame : "index" (axis=0, default), "columns" (axis=1)

Let us create a DataFrame and use this object throughout this chapter for all the operations.

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack',
   'Lee','David','Gasper','Betina','Andres']),
   'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

#Create a DataFrame
df = pd.DataFrame(d)
print df
```

Its output is as follows:

```
    Age  Name   Rating
0   25   Tom     4.23
1   26   James   3.24
2   25   Ricky   3.98
3   23   Vin     2.56
4   30   Steve   3.20
5   29   Smith   4.60
6   23   Jack    3.80
7   34   Lee     3.78
8   40   David   2.98
9   30   Gasper  4.80
10  51   Betina  4.10
11  46   Andres  3.65
```

Functions & Description Let us now understand the functions under Descriptive Statistics in Python Pandas. The following table list down the important functions

S.No.	Function	Description
1	count()	Number of non-null observations
2	sum()	Sum of values
3	mean()	Mean of Values
4	median()	Median of Values
5	mode()	Mode of values
6	std()	Standard Deviation of the Values
7	min()	Minimum Value
8	max()	Maximum Value
9	abs()	Absolute Value
10	prod()	Product of Values
11	cumsum()	Cumulative Sum
12	cumprod()	Cumulative Product

Note Since DataFrame is a Heterogeneous data structure. Generic operations dont work with all functions.

- Functions like sum(), cumsum() work with both numeric and character (or) string data elements without any error. Though n practice, character aggregations are never used generally, these functions do not throw any exception.
- Functions like abs(), cumprod() throw exception when the DataFrame contains character or string data because such operations cannot be performed.

3.3.1 sum()

Returns the sum of the values for the requested axis. By default, axis is index (axis=0).

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack',
   'Lee','David','Gasper','Betina','Andres']),
   'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

#Create a DataFrame
df = pd.DataFrame(d)
print df.sum()
```

Its output is as follows:

```
Age                                               382
Name      TomJamesRickyVinSteveSmithJackLeeDavidGasperBe...
Rating                                          44.92
dtype: object
```

Each individual column is added individually (Strings are appended).

axis=1

This syntax will give the output as shown below.

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack',
   'Lee','David','Gasper','Betina','Andres']),
   'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

#Create a DataFrame
df = pd.DataFrame(d)
print df.sum(1)
```

Its output is as follows:

```
0    29.23
1    29.24
2    28.98
3    25.56
4    33.20
5    33.60
6    26.80
7    37.78
8    42.98
9    34.80
10   55.10
11   49.65
dtype: float64
```

3.3.2 mean()

Returns the average value

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack',
   'Lee','David','Gasper','Betina','Andres']),
   'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

#Create a DataFrame
df = pd.DataFrame(d)
print df.mean()
```

Its output is as follows:

```
Age       31.833333
Rating     3.743333
dtype: float64
```

3.3.3 std()

Returns the Bressel standard deviation of the numerical columns.

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack',
   'Lee','David','Gasper','Betina','Andres']),
   'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

#Create a DataFrame
df = pd.DataFrame(d)
print df.std()
```

Its output is as follows:

```
Age       9.232682
Rating    0.661628
dtype: float64
```

3.3.4 Summarizing Data

The describe() function computes a summary of statistics pertaining to the DataFrame columns.

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack',
   'Lee','David','Gasper','Betina','Andres']),
   'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

#Create a DataFrame
df = pd.DataFrame(d)
print df.describe()
```

Its output is as follows:

```
             Age        Rating
count   12.000000     12.000000
mean    31.833333      3.743333
std      9.232682      0.661628
min     23.000000      2.560000
25%     25.000000      3.230000
50%     29.500000      3.790000
75%     35.500000      4.132500
max     51.000000      4.800000
```

This function gives the mean, std and IQR values. And, function excludes the character columns and given summary about numeric columns. 'include' is the argument which is used to pass necessary information regarding what columns need to be considered for summarizing. Takes the list of values; by default, 'number'.

- **object**: Summarizes String columns
- **number** : Summarizes Numeric columns
- **all**: Summarizes all columns together (Should not pass it as a list value)

Now, use the following statement in the program and check the output

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack',
   'Lee','David','Gasper','Betina','Andres']),
   'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

#Create a DataFrame
df = pd.DataFrame(d)
print df.describe(include=['object'])
```

Its output is as follows:

```
         Name
count      12
unique     12
top     Ricky
freq        1
```

Now, use the following statement in the program and check the output

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack',
   'Lee','David','Gasper','Betina','Andres']),
   'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

#Create a DataFrame
df = pd.DataFrame(d)
print df. describe(include='all')
```

Its output is as follows:

```
          Age          Name       Rating
count   12.000000        12    12.000000
unique        NaN        12          NaN
top           NaN     Ricky          NaN
freq          NaN         1          NaN
mean    31.833333       NaN     3.743333
std      9.232682       NaN     0.661628
min     23.000000       NaN     2.560000
25%     25.000000       NaN     3.230000
50%     29.500000       NaN     3.790000
75%     35.500000       NaN     4.132500
max     51.000000       NaN     4.800000
```