notes

April 14, 2024

0.1 Pandas

Pandas is an open-source Python Library used for `high-performance` data manipulation and data analysis using its powerful data structures. `Python with pandas is in use in a variety of academic and commercial domains, including Finance, Economics, Statistics, Advertising, Web Analytics, and more.`Using Pandas, we can accomplish many steps in the processing and analysis of data, — load, organize, manipulate, model, and analyze the data.

****Key Features of Pandas****

- Fast and efficient DataFrame `object with default and customized indexing`.
- Can load many file formats.
- Data alignment and handling of missing data.
- Data manipulation like reshaping and pivoting and many more.
- Columns can be inserted and deleted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.

Series - One Dimensional with homogeneous data. - One type of data

Dataframe - Two dimensional - rows and columns - with heterogeneous data

0.2 NumPy

NumPy is a python package which stands for 'Numerical Python'.

Key Features

- Mathematical and Logical operations
- NumPy has in-built functions for linear algebra and random number generation.
- Shape Manipulation

NumPy is used along with scipy package [Scientific Python] and matplotlib [plotting]

0.2.1 ndarray

It is the most important object defined in NumPy and it is the N-dimensional array or ndarray. It describes the collection of items of the same type. Every item in an ndarray takes the same size of block in the memory. Each element in ndarray is an object of data-type object (called **dtype**). Any item extracted from ndarray object (by slicing) is represented by a Python object of one of array scalar types.

0.3 SciPy

SciPy library works together with NumPy and provides many user-friendly and efficient numerical practices such as routines for numerical integration and optimization.

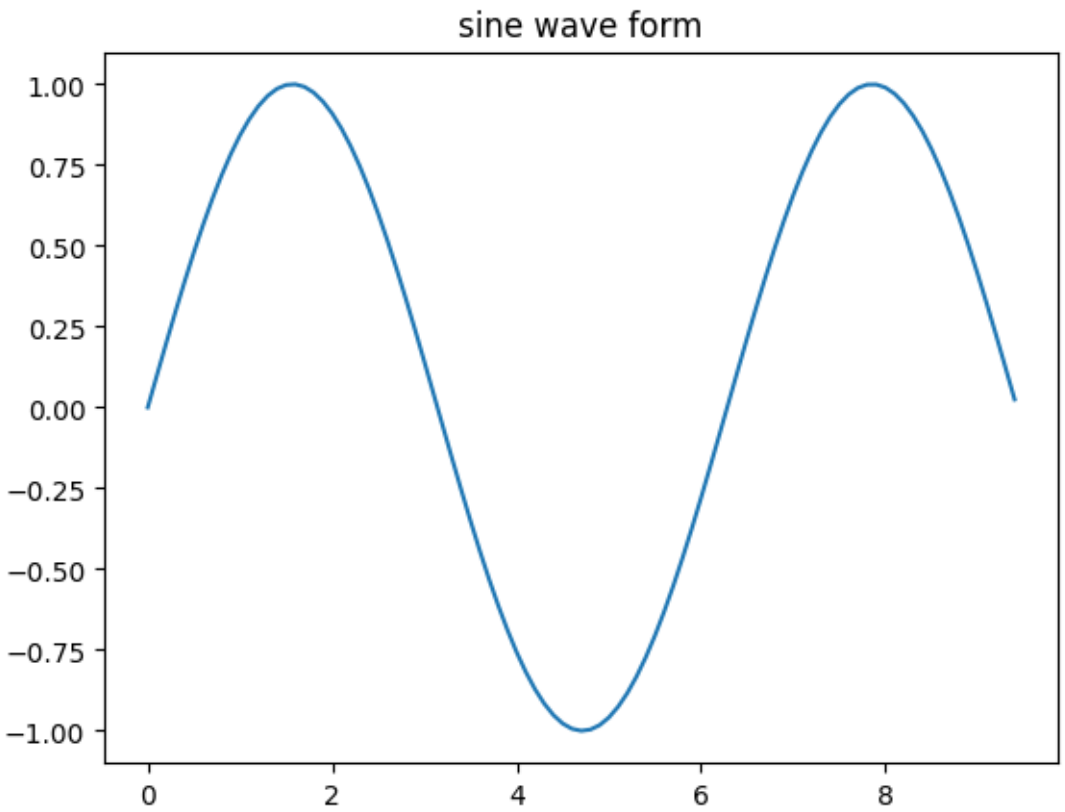
0.4 Matplotlib

It is used to create 2D graphs using python scripts. It has a module called pyplot which make it easy to plot and it also helps to control the line style, font styles and formatting axis. It supports a very wide variety of graphs and plots namely - histogram, bar charts, power spectra, error charts etc.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt

# Compute the x and y coordinates for points on a sine curve
x = np.arange(0, 3 * np.pi, 0.1)
y = np.sin(x)
plt.title("sine wave form")

# Plot the points using matplotlib
plt.plot(x, y)
plt.show()
```



0.5 Data Processing

Two main libraries: Pandas and NumPy to process various data formats ### Data Operations

0.5.1 Pandas Series

[1]:
```
import pandas as pd
import numpy as np
data = np.array(['a','s','c'])
d = pd.Series(data)
print(d)
```

```
0    a
1    s
2    c
dtype: object
```

0.5.2 Pandas Dataframe

pandas.DataFrame(data, index, columns, dtype, copy)

[2]:
```
import pandas as pd
data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]}
df = pd.DataFrame(data, index=['rank1','rank2','rank3','rank4'])
df
```

[2]:

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

0.5.3 Data Cleansing

When collecting data or fetching data, most of the times people don't share all of their information so missing data happens and it needs to be fixed.

Example: Few people share their experience, but not how long they are using the product; few people share how long they are using the product, their experience but not their contact information.

For dataframe, it is `ALWAYS pd.DataFrame`

[3]:
```
#import pandas as pd
#import numpy as np
#index is a total of 6 and columns = 3 hence random.randn = 6,3
df = pd.DataFrame(np.random.randn(6,3), index = ['a', 'c', 'e', 'f','h','g'],␣
 ↪columns = ['one','two','three'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
```

```
df
```

[3]:
```
        one       two     three
a -2.011379 -0.804862  1.687874
b       NaN       NaN       NaN
c -0.773984 -0.730600  1.291006
d       NaN       NaN       NaN
e  2.147558  0.602790 -0.131916
f -0.044347 -0.597930  0.682180
g -1.552349  0.120235 -0.060360
h  1.780655  0.816198 -0.054545
```

[9]:
```
#checking for null values
#the entire df
df.isnull()
```

[9]:
```
     one    two  three
a  False  False  False
b   True   True   True
c  False  False  False
d   True   True   True
e  False  False  False
f  False  False  False
g  False  False  False
h  False  False  False
```

[33]:
```
#checking for null values
#the one column
df['one'].isnull()
```

[33]:
```
a    False
b     True
c    False
d     True
e    False
f    False
g    False
h    False
Name: one, dtype: bool
```

[34]:
```
#filling the nan values with 0
df.fillna(0)
```

[34]:
```
        one       two     three
a  1.610199 -0.433307  0.050328
b  0.000000  0.000000  0.000000
c -0.761050  0.579514 -0.258856
```

```
d  0.000000  0.000000  0.000000
e  0.543653 -1.848697 -1.399187
f -0.046237  1.276574  0.195437
g  1.025117 -0.839656  0.433096
h  1.535614  0.795355  0.618780
```

[40]:
```
df.fillna(method='pad')
#since row d was null, method pad uses the previous row's data [c column to␣
 ↪fill in the d column]
```

[40]:
```
        one       two     three
a  1.610199 -0.433307  0.050328
b  1.610199 -0.433307  0.050328
c -0.761050  0.579514 -0.258856
d -0.761050  0.579514 -0.258856
e  0.543653 -1.848697 -1.399187
f -0.046237  1.276574  0.195437
g  1.025117 -0.839656  0.433096
h  1.535614  0.795355  0.618780
```

[39]:
```
df.dropna()
```

[39]:
```
        one       two     three
a  1.610199 -0.433307  0.050328
c -0.761050  0.579514 -0.258856
e  0.543653 -1.848697 -1.399187
f -0.046237  1.276574  0.195437
g  1.025117 -0.839656  0.433096
h  1.535614  0.795355  0.618780
```

[129]:
```
df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',␣
 ↪'h'],columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
df
```

[129]:
```
        one       two     three
a  0.347242 -0.110605  1.516642
b       NaN       NaN       NaN
c  0.063763  0.460191  1.347027
d       NaN       NaN       NaN
e  0.031764 -1.142955 -0.447163
f -0.171389  0.973598 -0.227122
g       NaN       NaN       NaN
h -0.615443 -0.645140  0.374451
```

[67]:
```
df = pd.read_csv(r'C:\Users\KAREN J FERNANDES\Downloads\Sheet1.csv')
```

[68]:
```
df
```

[68]:

	id	name	salary	start_date	dept	Unnamed: 5	Unnamed: 6	Unnamed: 7	id,name,salary,start_date,dept
0	1	Rick	623.30	2012-01-01	IT	NaN	NaN	NaN	1,Rick,623.3,2012-01-01,IT
1	2	Dan	515.20	2013-09-23	Operations	NaN	NaN	NaN	2,Dan,515.2,2013-09-23,Operations
2	3	Tusar	611.00	2014-11-15	IT	NaN	NaN	NaN	3,Tusar,611,2014-11-15,IT
3	4	Ryan	729.00	2014-05-11	HR	NaN	NaN	NaN	4,Ryan,729,2014-05-11,HR
4	5	Gary	843.25	2015-03-27	Finance	NaN	NaN	NaN	5,Gary,843.25,2015-03-27,Finance
5	6	Rasmi	578.00	2013-05-21	IT	NaN	NaN	NaN	6,Rasmi,578,2013-05-21,IT
6	7	Pranab	632.80	2013-07-30	Operations	NaN	NaN	NaN	7,Pranab,632.8,2013-07-30,Operations
7	8	Guru	722.50	2014-06-17	Finance	NaN	NaN	NaN	8,Guru,722.5,2014-06-17,Finance

[69]:
```
df.columns
```

[69]:
```
Index(['id', 'name', 'salary', 'start_date', 'dept', 'Unnamed: 5',
       'Unnamed: 6', 'Unnamed: 7', 'id,name,salary,start_date,dept'],
      dtype='object')
```

[74]:
```
#to drop columns
df = df.drop(['Unnamed: 6'], axis = 1)
```

[75]:
```
df.columns
```

[75]:
```
Index(['id', 'name', 'salary', 'start_date', 'dept'], dtype='object')
```

[77]:
```
df
```

[77]:

	id	name	salary	start_date	dept
0	1	Rick	623.30	2012-01-01	IT
1	2	Dan	515.20	2013-09-23	Operations
2	3	Tusar	611.00	2014-11-15	IT
3	4	Ryan	729.00	2014-05-11	HR
4	5	Gary	843.25	2015-03-27	Finance
5	6	Rasmi	578.00	2013-05-21	IT
6	7	Pranab	632.80	2013-07-30	Operations
7	8	Guru	722.50	2014-06-17	Finance

[82]:
```
#to return specific columns
df.loc[:,['name', 'salary']]
```

[82]:

	name	salary
0	Rick	623.30
1	Dan	515.20
2	Tusar	611.00
3	Ryan	729.00
4	Gary	843.25
5	Rasmi	578.00
6	Pranab	632.80
7	Guru	722.50

[83]:
```
#to return specific columns and rows
df.loc[[1,2,5],['name','salary']]
```

[83]:

	name	salary
1	Dan	515.2
2	Tusar	611.0
5	Rasmi	578.0

[86]:
```
#to return a range
df.loc[3:6,['name','salary']]
```

[86]:

	name	salary
3	Ryan	729.00
4	Gary	843.25
5	Rasmi	578.00
6	Pranab	632.80

0.5.4 How to convert json to dataframe

https://sparkbyexamples.com/pandas/pandas-convert-json-to-dataframe/#:~:text=You%20can%20convert%20JS

[118]:
```
#Convert json to dataframe
import json
from pandas import json_normalize

jsonstr = '''{
   "ID":["1","2","3","4","5","6","7","8" ],
   "Name":["Rick","Dan","Michelle","Ryan","Gary","Nina","Simon","Guru" ],
   "Salary":["623.3","515.2","611","729","843.25","578","632.8","722.5" ],

   "StartDate":[ "1/1/2012","9/23/2013","11/15/2014","5/11/2014","3/27/2015","5/
 ↪21/2013",
      "7/30/2013","6/17/2014"],
   "Dept":[ "IT","Operations","IT","HR","Finance","IT","Operations","Finance"]
```

```
}'''
```

[119]:
```
jsonstr
```

[119]:
```
'{ \n   "ID":["1","2","3","4","5","6","7","8" ],\n
"Name":["Rick","Dan","Michelle","Ryan","Gary","Nina","Simon","Guru" ],\n
"Salary":["623.3","515.2","611","729","843.25","578","632.8","722.5" ],\n   \n
"StartDate":[
"1/1/2012","9/23/2013","11/15/2014","5/11/2014","3/27/2015","5/21/2013",\n
"7/30/2013","6/17/2014"],\n   "Dept":[
"IT","Operations","IT","HR","Finance","IT","Operations","Finance"]\n}'
```

[140]:
```
import pandas as pd
jsonStrr = '''{
   "ID":["1","2","3","4","5","6","7","8" ],
   "Name":["Rick","Dan","Michelle","Ryan","Gary","Nina","Simon","Guru" ],
   "Salary":["623.3","515.2","611","729","843.25","578","632.8","722.5" ],
   "StartDate":[ "1/1/2012","9/23/2013","11/15/2014","5/11/2014","3/27/2015","5/
 ↪21/2013",
      "7/30/2013","6/17/2014"],
   "Dept":[ "IT","Operations","IT","HR","Finance","IT","Operations","Finance"]
}'''

# Convert JSON to DataFrame Using read_json()
json_data = pd.read_json(jsonStrr, orient ='index')
json_data
```

[140]:

	0	1	2	3	4	5	6	7
ID	1	2	3	4	5	6	7	8
Name	Rick	Dan	Michelle	Ryan	Gary	Nina	Simon	Guru
Salary	623.3	515.2	611	729	843.25	578	632.8	722.5
StartDate	1/1/2012	9/23/2013	11/15/2014	5/11/2014	3/27/2015	5/21/2013	7/30/2013	6/17/2014
Dept	IT	Operations	IT	HR	Finance	IT	Operations	Finance

[143]:
```
json_data.to_csv('json_data.csv')
```

[146]:
```
json = pd.read_csv(r'C:\Users\KAREN J␣
 ↪FERNANDES\AppData\Local\Programs\Python\Python311\Scripts\Files\Study␣
 ↪Notes\json_data.csv')
```

[147]:
```
json
```

[147]:
```
  Unnamed: 0         0           1           2           3           4  \
0         ID         1           2           3           4           5
1       Name      Rick         Dan    Michelle        Ryan        Gary
2     Salary     623.3       515.2         611         729      843.25
3  StartDate  1/1/2012   9/23/2013  11/15/2014   5/11/2014  3/27/2015
4       Dept        IT  Operations          IT          HR     Finance

           5           6          7
0          6           7          8
1       Nina       Simon       Guru
2        578       632.8      722.5
3  5/21/2013   7/30/2013  6/17/2014
4         IT  Operations    Finance
```

[155]:
```
df_long = json.melt(id_vars=["ID", "Name", "StartDate", "Dept"],␣
 ↪var_name="Variable", value_name="Value")
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[155], line 1
----> 1 df_long =␣
 ↪json.melt(id_vars=["ID", "Name", "StartDate", "Dept"], var_name="Variable", v lue_name="Va

File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\frame
 ↪py:9124, in DataFrame.melt(self, id_vars, value_vars, var_name, value_name,␣
 ↪col_level, ignore_index)
   9113 @Appender(_shared_docs["melt"] % {"caller": "df.melt(", "other": "melt"})
   9114 def melt(
   9115     self,
   (…)
   9121     ignore_index: bool = True,
   9122 ) -> DataFrame:
-> 9124     return melt(
   9125         self,
   9126         id_vars=id_vars,
   9127         value_vars=value_vars,
   9128         var_name=var_name,
   9129         value_name=value_name,
   9130         col_level=col_level,
   9131         ignore_index=ignore_index,
   9132     ).__finalize__(self, method="melt")
```

```
File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\reshape\melt.
 ↪py:77, in melt(frame, id_vars, value_vars, var_name, value_name, col_level,␣
 ↪ignore_index)
     75         missing = Index(com.flatten(id_vars)).difference(cols)
     76         if not missing.empty:
---> 77             raise KeyError(
     78                 "The following 'id_vars' are not present "
     79                 f"in the DataFrame: {list(missing)}"
     80             )
     81 else:
     82     id_vars = []

KeyError: "The following 'id_vars' are not present in the DataFrame: ['Dept',␣
 ↪'ID', 'Name', 'StartDate']"
```

[121]:
```
import pandas as pd
jsonStr = '''{"Index0":{"Courses": "Pandas","Discount": "1200"},
           "Index1":{"Courses": "Hadoop","Discount": "1500"},
           "Index2":{"Courses": "Spark","Discount": "1800"}
          }'''

# Convert JSON to DataFrame Using read_json()
df2 = pd.read_json(jsonStr, orient ='index')
df2
```

[121]:

	Courses	Discount
Index0	Pandas	1200
Index1	Hadoop	1500
Index2	Spark	1800

[157]:
```
import pandas as pd
with pd.ExcelFile('C:/Users/Rasmi/Documents/pydatasci/input.xlsx') as xls:
    df1 = pd.read_excel(xls, 'Sheet1')
    df2 = pd.read_excel(xls, 'Sheet2')

print("****Result Sheet 1****")
print (df1[0:5]['salary'])
print("")
print("***Result Sheet 2****")
print (df2[0:5]['zipcode'])
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[157], line 2
      1 import pandas as pd
```

```
----> 2 with pd.ExcelFile('C:/Users/Rasmi/Documents/pydatasci/input.xlsx') as␣
 ↪xls:
      3     df1 = pd.read_excel(xls, 'Sheet1')
      4     df2 = pd.read_excel(xls, 'Sheet2')

File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\io\excel\_base.
 ↪py:1652, in ExcelFile.__init__(self, path_or_buffer, engine, storage_options)
   1650     ext = "xls"
   1651 else:
-> 1652     ext = inspect_excel_format(
   1653         content_or_path=path_or_buffer, storage_options=storage_options
   1654     )
   1655     if ext is None:
   1656         raise ValueError(
   1657             "Excel file format cannot be determined, you must specify "
   1658             "an engine manually."
   1659         )

File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\io\excel\_base.
 ↪py:1525, in inspect_excel_format(content_or_path, storage_options)
   1522 if isinstance(content_or_path, bytes):
   1523     content_or_path = BytesIO(content_or_path)
-> 1525 with get_handle(
   1526     content_or_path, "rb", storage_options=storage_options, is_text=Fal e
   1527 ) as handle:
   1528     stream = handle.handle
   1529     stream.seek(0)

File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\io\common.
 ↪py:865, in get_handle(path_or_buf, mode, encoding, compression, memory_map,␣
 ↪is_text, errors, storage_options)
    856         handle = open(
    857             handle,
    858             ioargs.mode,
   (…)
    861             newline="",
    862         )
    863     else:
    864         # Binary mode
--> 865         handle = open(handle, ioargs.mode)
    866     handles.append(handle)
    868 # Convert BytesIO or file objects passed with an encoding

FileNotFoundError: [Errno 2] No such file or directory: 'C:/Users/Rasmi/
 ↪Documents/pydatasci/input.xlsx'
```

[16]:
```
#import numpy as np
#import pandas as pd

#create DataFrame
percentile = pd.DataFrame({'var1': [25, 12, 15, 14, 19, 23, 25, 29, 33, 35],
                   'var2': [5, 7, 7, 9, 12, 9, 9, 4, 14, 15],
                   'var3': [11, 8, 10, 6, 6, 5, 9, 12, 13, 16]})

#percentile is the 'of the total lot or population', so if it is 95 percentile,␣
 ↪you are in the among the 95% of the lot or
#population

#numpy.percentile(a, q)
#a: Array of values
#q: Percentile or sequence of percentiles to compute, which must be between 0␣
 ↪and 100 inclusive.

#for all the columns
percentile.quantile(.95)
```

[16]:
```
var1    34.10
var2    14.55
var3    14.65
Name: 0.95, dtype: float64
```

[15]:
```
#selected columns
percentile[['var1','var2']].quantile(.95)
```

[15]:
```
var1    34.10
var2    14.55
Name: 0.95, dtype: float64
```

[ ]:
```
#find the difference in the highest salaries between two depts ie; dept_id 4␣
 ↪and 1
# Import your libraries
import pandas as pd

# Start writing code
db_employee.head()

db_dept.head()

db_employee[db_employee['department_id']== 4]['salary'].
 ↪max()-db_employee[db_employee['department_id']==1]['salary'].max()
```

[ ]:
```
#Popularity of Hack
```

```
#Based on the above, find the average popularity of the Hack per office␣
 ↪location.
#Output the location along with the average popularity.

import pandas as pd
import numpy as np

merged = pd.merge(facebook_employees,facebook_hack_survey, left_on = 'id',␣
 ↪right_on = 'employee_id', how = 'inner')
result = merged.groupby(['location'])['popularity'].mean().reset_index()
```

[ ]:
```
#sort in ascending order of id
# Import your libraries
import pandas as pd
import numpy as np
# Start writing code
ms_employee_salary.head()

ms_employee_salary.groupby('id').max().reset_index()

#or

ms_employee_salary.sort_values(['id','salary'], ascending = (True, False)).
 ↪groupby(['id']).first().reset_index()
```

[ ]:
```
#Compare each employee's salary with the average salary of the corresponding␣
 ↪department.
#Import your libraries
import pandas as pd

# Start writing code
employee.head()

df = employee
df

df["avg_sal"] = employee.groupby('department')['salary'].transform('mean')
df[['department','first_name','salary','avg_sal']]
```

[ ]:
```
#count the no of users who use macbook pro or Count the number of user events␣
 ↪performed by MacBookPro users.

#Import your libraries
import pandas as pd

# Start writing code
df = playbook_events
```

```
df
df[df['device'] == 'macbook pro']['event_name'].value_counts().reset_index()
#by default, meaning without many parameters in the brackets, it is␣
 ↪ascending=False meaning descending
Syntax: Series.value_counts(normalize=False, sort=True, ascending=False,␣
 ↪bins=None, dropna=True)
```

[1]:
```
py = {1:'Apple',2:'Mango'}
```

[2]:
```
py
```

[2]:
```
{1: 'Apple', 2: 'Mango'}
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: