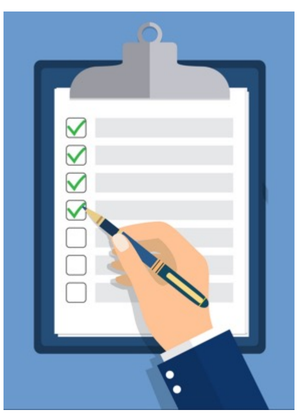
Data Collection, Cleaning and Analysis with Pandas

Session-3

1

Agenda



Lecture -3: Review Python and Jupyter notebook

- Use NumPy arrays and functions for basic statistical analyses
- Use Pandas to build, extract, filter, and transform data frames
- Describe Pandas data structures: data frames and series
- Use Pandas objects for analyses

2

Use Numpy Arrays

NumPy

3

Numpy

- Short for of Numerical Python - A grid of values all of the same type
- Most common data science packages are built on Numpy
- Numpy operations are about ten times or more faster than a simple list operation
- The number of dimensions is the rank of the array
- The shape of the array is a tuple of integers denoting the size of the array along each dimension

4

Key Features of numpy

- ndarrays: n-dimentional arrays for rapid processing of data without using loops
- Broadcasting: defines implicit behavior between multi-dimensional arrays of different sizes.
- Vectorization: enables numeric operations on ndarrays.
- Input/Output: simplifies reading and writing of data from/to file.

5

Creating and Accessing Arrays by Index

```
# Creating a rank 2 or multidimentional array
x=np.array([[1,2],[2,4]])
print(x)
# Convert a  list to a numpy array
list1 = [1, 2, 3, 4, 5]
array1 = np.array(list1)
print (array1)
print("Shape of x: ", x.shape)
print("Shape of array1: ", array1.shape)
array1.dtype
```

```
array([[1, 2],
       [2, 4]])
[1 2 3 4 5]
Shape of x: (2, 2)
Shape of array1: (5,)
dtype('int64')
```

6

Creating and Accessing Arrays by Index

```
# Access by passing the index of each dimension
print(array2)
print(array2[0,0])
print(array2[1, 2])
print(array2[1, :])
print(array2[:, 3])
print(array2[:, 1:])
print(array2[:, -2:])
```

array2 = [[1 2 3 4 5]
[100 200 300 400 500]]

array2[0,0]= 1

array2[1, 2]= 300

array2[1, :]= [100 200 300 400 500]

array2[:, 3]= [4 400]

array2[:, 1:]= [[2 3 4 5]
[200 300 400 500]]

array2[:, -2:]= [[4 5]
[400, 500]]

ASCEND towards a sustainable future KIRON e-Learning for Future

7

Slicing Indexes

```
# One dimensional array with 10 elements
a = np.arange(10)
print(a)
b = a[2:7:2] #One dimensional (start:stop:step)
print(b)
c = a[2:]
print (c)
```

[0 1 2 3 4 5 6 7 8 9]

[2 4 6]

[2 3 4 5 6 7 8 9]

ASCEND towards a sustainable future KIRON e-Learning for Future

8

Quiz

What would these print commands slice from the following array a?

[[1 2 3]
[4 5 6]
[7 8 9]]

print(a[1:])

a. [[4 5 6]
[7 8 9]]

b. [[2 3]
[5 6]
[8 9]]

9

Quiz

What would these print commands slice from the following array b?

[[1 2 3]
[4 5 6]
[7 8 9]]

print(b[:,1])

a. [7 8 9]

b. [2 5 8]

c. [4 5 6]

d. None of the above

10

Numpy : Arithmetic Operation

Numpy Array - Addition

```
x=np.array([[1,2],[2,4]])
y=np.array([[1,3],[3,5]])
print(x)
print (y)
print(x+y)
```

array([[1, 2],
       [2, 4]])

array([[1, 3],
       [3, 5]])

array([[2, 5],
       [5, 9]])

Multiplication

```
print(x*y)
```

array([[1, 6],
       [6, 20]])

You can use subtraction, division, square etc.

Unlike list you cannot mix data types

11

Basic Statistical Operations

```
# setup a random 3 x 5 matrix
narray1 = 10 * np.random.randn(2,5)
print(narray1)
print(narray1.mean())
print(narray1.mean(axis = 1)) #mean by row
print(narray1.mean(axis = 0)) #mean by col
```

[[10.117 -6.251 1.923 -9.875 -3.118]
 [2.367 14.119 -4.002 -5.276 0.879]]

-0.0385

[-0.921 0.844]

[1.338 -4.636 6.227 -9.264 5.555]

```
# sum all the elements
print(narray1.sum())
print(np.median(narray1, axis = 1))
```

-0.385

[-2.827 -1.306]

12

Broadcasting

```
start = np.zeros((4,3))
print(start)
# create a rank 1 ndarray with 3 values
add_rows = np.array([1, 0, 2])
print(add_rows)
```

[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
[1 0 2]

```
y = start + add_rows # add to each
row of 'start' using broadcasting
```

[[1. 0. 2.]
 [1. 0. 2.]
 [1. 0. 2.]
 [1. 0. 2.]]

Similarly you can add to each column

13

Dot Products on Matrices

```
p=np.array([[1,2,3],[2,4,5]])
print (p)
q=np.array([[1,3],[3,5],[4,6]])
print(q)
p@q
```

array([[1, 2, 3],
 [2, 4, 5]])

array([[1, 3],
 [3, 5],
 [4, 6]])

array([[19, 31],
 [34, 56]])

```
A = np.array([[2., 3.], [3., 4.]])
B = np.linalg.inv(A) # create its inverse
print (A)
print (B)
print(A@B)
print(B@A) # A @ B =I
I = np.identity(2) # Identity matrix
```

[[2. 3.]
 [3. 4.]]

[[1. 0.]
 [0. 1.]]

[[-4. 3.]
 [3. -2.]]

[[1. 0.]
 [0. 1.]]

14

Transposing A Matrix

```
A = np.array([[2, 3, 6], [5, 7, 9]])
A
```

```
array([[2, 3, 6],
       [5, 7, 9]])
```

```
C=A.T
C
```

```
array([[2, 5],
       [3, 7],
       [6, 9]])
```

```
C.T
```

```
array([[2, 3, 6],
       [5, 7, 9]])
```

15

Build, Extract, Filter, and Transform data

16

Pandas

Most commonly used Python package for data analysis

Why use Pandas?

- Offers a number of essential data exploration, cleaning and transformation operations
- Read and write data between in-memory data structures and different formats
- Easily manipulate messy data
- Label-based slicing, indexing, and subsetting of large data sets
- Open Source!

17

Pandas

Tow main data structures *pandas* provides are *Series* and *DataFrames:*

```
ser1=pd.Series([1, 2, 3], index=['a', 'b', 'c'])
ser2=pd.Series([2, 4, 6, 7], index=['a', 'b', 'c', 'd'])
print (ser1)
print (ser2)
```

```
a    1
b    2
c    3
dtype: int64
a    2
b    4
c    6
d    7
dtype: int64
```

```
d = {'one' : ser1,
     'two' : ser2}
df = pd.DataFrame(d)
print(df)
```

```
   one  two
a  1.0    2
b  2.0    4
c  3.0    6
d  NaN    7
```

18

Introduction to DataFrames

```
data = np.array([['','Col1','Col2'], ['Row1',1,2], ['Row2',3,4]])

print(pd.DataFrame(data=data[1:,1:], index=data[1:,0], columns=data[0,1:]))
```

Output:

```
      Col1 Col2
Row1    1    2
Row2    3    4
```

ASCEND towards a sustainable future KIRON e-Learning for Future

19

Pandas Dataframe

```
df
df.index
```

```
   one  two
a  1.0   2
b  2.0   4
c  3.0   6
d  NaN   7
```

Index(['a', 'b', 'c', 'd'], dtype='object')

```
df.columns
```

Index(['one', 'two'], dtype='object')

```
# # Creating a dataframe with selected indexes
pd.DataFrame(df, index=[ 'b', 'c'])
```

df

	one	two
b	2	4
c	3	6

```
# Projecting a column
df['one']
```

```
a   1.0
b   2.0
c   3.0
d   NaN
```

ASCEND towards a sustainable future KIRON e-Learning for Future

20

Quiz

Which of the following input can be accepted by a Pandas DataFrame?

a. Structured ndarray
b. Series
c. DataFrame
d. All of these

Answer: d. All of these

21

Setting Pandas Options

- Suppress scientific notation e.g. 1.000094e+11

```
pd.set_option('display.float_format', '{:.0f}'.format)
```

- Enable scientific notation

```
pd.set_option('display.float_format', '{:.6E}'.format)
```

- Set row and column options

```
pd.set_option('display.max_rows', 5)
pd.set_option('display.max_colwidth', 20)
pd.set_option('display.max_columns', None)
```

- For productivity save all your preferences in a file that you can reuse

22

Quiz

How do you Suppress scientific notation ?

a. pd.set_option('display.float_format', '{:.0f}'.format)
b. pd.set_option('display.float_format', '{:.6E}'.format)
c. pd.set_option('display.float_format = {:.6E}.format')
d. pd.set_option('display.float_format = {:.0f}.format')

Write to Different File Types and Formats

- Pandas supports a variety of different file formats
  - CSV Files
  - SAS Files
  - JSON Files
  - HTML Files
  - Excel Files
  - SQL Files
  - Parquet files
- For larger files you can use compression option as below

```
df.to_csv('file2', compression = 'zip')
df.to_parquet('df.parquet.gzip', compression='gzip')
```

Quiz

Which of the following is not a valid Pandas (pd) file load option?

a. pd.read_csv
b. pd.read_table
c. pd.read_data
d. pd.read_clipboard

Answer: c. pd.read_data

25

Restructure data into a tidy form

- **Tidying when two or more values are stored in the same cell**

Name, Address

John,Washington D.C. 20003

Rob,Brooklyn NY 11211-1755

Sandy,Omaha NE 68154

Katy,Pittsburgh PA 15211

```
Addresses = clients.Address.str.split(pat=' ', expand=True)
Addresses.columns = ['City', 'State','Zip']
```

City	State	Zip
Washington	D.C.	20003
Brooklyn	NY	11211-1755

26

Quiz

pandas.Series.str.split split strings around given separator/delimiter.

a. Ture

b. False

27

Restructure data into a tidy form (contd.)

- Tidying when column names are values, not variable names

State,Iphone,Galaxy,Others
DC,230,210,340
NY,480,170,215
CA,900,140,180

```
df_phone.stack().rename_axis(['State', 'Phone']).reset_index(name='Quantity_Sold')
```

State	Phone	Quantity_Sold
DC	Iphone	230
DC	Galaxy	210
DC	Others	340

28

Quiz

The stack() function acts like a collection of books being reorganized from being side by side to a horizontal position (the columns of the dataframe) to being stacked vertically on top of each other.

a. True
b. False

Answer: True

29

Joins with Pandas

Pandas offers concatenation, appendation, not to be confused with joining. To join, Pandas uses the function merge()

```
pd.merge(left, right, how='inner', on=None, left_on=None, right_on=None,
         left_index=False, right_index=False, sort=True,
         suffixes=('_x', '_y'), copy=True, indicator=False,
         validate=None)
```

- left = dataframe
- right = dataframe
- on = column/s to join on
- left_on/right_on = join keys of the dataframes
- how = inner/outer/left/right

30

Join Examples

Left & Right joins

```
#Left Join
left = pd.merge(df1, df2,
on='user_id', how='left')

#Right Join
right = pd.merge(df1,
df2, on='user_id',
how='right')
```

Inner & Outer joins

```
#Inner Join
inner = pd.merge(df1, df2,
on='user_id', how='inner')

#Outer Join
outer = pd.merge(df1, df2,
on='user_id', how='outer')
```

31

pandas.cut

- pandas.cut is used to bin values into discrete intervals.
- This function is also useful for going from a continuous variable to a categorical variable.
- For example, cut could convert ages to groups of age ranges.

```
data=np.array([1, 7, 5, 4, 6, 3, 8, 9])
data
```

```
array([1, 7, 5, 4, 6, 3, 8, 9])
```

```
binned=pd.cut(data, 3, labels=["bad", "medium", "good"])
binned
```

```
['bad', 'good', 'medium', 'medium', 'medium', 'bad', 'good', 'good']
Categories (3, object): ['bad' < 'medium' < 'good']
```

```
binned.value_counts()
```

```
bad       2
medium    3
good      3
```

32

Pivot

- Pandas provide the pivot_table() function to create spreadsheet-style pivot tables.
- The pivot_table() function enables aggregation of data values across row and column dimensions.
- This function can be a convenient method to apply a MultiIndex to a DataFrame

```
df.pivot_table(index =   ['Year', 'ProductLine'],
               columns = ['Territory'],
               values = ['Amount'])
```

33

Pandas Timestamp

- Pandas replacement for python datetime. datetime object.
- Timestamp is the pandas equivalent of python's Datetime and is interchangeable with it in most cases.
- It's the type used for the entries that make up a DatetimeIndex, and other timeseries oriented data structures in pandas.

34

Formatting Datetime/Timestamp as String

- Within python we can format dates using strftime and parse dates using strptime.
- The two classes use the same parameters for its output format:

Directive	Meaning	Example
%a	Weekday as locale's abbreviated name.	Sun, Mon, …, Sat (en_US);
		So, Mo, …, Sa (de_DE)
%A	Weekday as locale's full name.	Sunday, Monday, …, Saturday (en_US);
		Sonntag, Montag, …, Samstag (de_DE)
%w	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.	0, 1, …, 6
%d	Day of the month as a zero-padded decimal number.	01, 02, …, 31
%b	Month as locale's abbreviated name.	

35

Thank You

36