What is Data Science?

Data science ~ computer science + mathematics/statistics + visualization

Outline of a data science project

- Harvesting
- Cleaning
- Analyzing
- Visualizing
- Publishing

Actively used Python tools for Data Analytics

- Pandas
- Numpy
- Matplotlib

What is Pandas

- Pandas 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.

Using Pandas, we can accomplish five typical steps

- Load
- Prepare
- Manipulate
- Model
- Analyze

Pandas is used for

- Finance
- Economics
- Statistics
- Analytics

Pandas deals with the following three data structures

- Series - 1D homogenous size immutable
- Data Frame - 2D Heterogenous size mutable
- Panel - 3D Size mutable

Series

- Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called index.
- A pandas Series can be created using the following constructor − **pandas.Series(data, index, dtype, copy)**

In [5]:
```
#import the pandas library and aliasing as pd
import pandas as pd
s = pd.Series()
print (s)
```

```
Series([], dtype: float64)
```

In [6]:
```
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data)
print (s)
```

```
0    a
1    b
2    c
3    d
dtype: object
```

In [7]:
```
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data,index=[1,12,123,145])
print (s)
```

```
1      a
12     b
123    c
145    d
dtype: object
```

In [1]:
```
#Series from a dictionary
import pandas as pd
import numpy as np
data = {'Mammal' : 'Tiger', 'Snake' : 'Python', 'Bird' : 'Peacock'}
s = pd.Series(data)
print (s)
```

```
Mammal      Tiger
Snake      Python
Bird      Peacock
dtype: object
```

In [9]:
```
#Creating series out of scalar
import pandas as pd
import numpy as np
s = pd.Series(5, index=[0, 1, 2, 3])
print (s)
```

```
0    5
1    5
2    5
3    5
dtype: int64
```

In [16]:
```
#Accessing Data from Series with Position
import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve the first element
print (s['a']) #Retrieve Data Using Label
print (s[3]) #Retrieve Data Using index
print(s[:4]) #Retrieve data using slice
print(s[-3:]) #Retrieve data using slice
print (s[['a','c']]) #Retrieve multiple elements using a list of index label values.
```

```
1
4
a    1
b    2
c    3
d    4
dtype: int64
c    3
d    4
e    5
dtype: int64
a    1
c    3
dtype: int64
```

Working with Data Frames

- **Two-dimensional size-mutable**
- **potentially heterogeneous tabular data structure with labeled axes (rows and columns)**
- **Can be thought of as a dict-like container for Series objects**
- **class pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)**

In [3]:
```
#Constructing DataFrame from a dictionary
import pandas as pd
d = {'col1': [1, 2], 'col2': [3, 4]}
df = pd.DataFrame(data=d)
print(df)
```

```
   col1  col2
0     1     3
1     2     4
```

In [8]:
```
#Access the types
import pandas as pd
d = {'col1': [1, 2], 'col2': [3, 4]}
df = pd.DataFrame(data=d)
print(df.dtypes)
```

```
col1    int64
col2    int64
dtype: object
```

In [9]:
```
#Enforcing the types
import pandas as pd
import numpy as np
d = {'col1': [1, 2], 'col2': [3, 4]}
df = pd.DataFrame(data=d, dtype=np.int8)
print(df.dtypes)
```

```
col1    int8
col2    int8
dtype: object
```

In [14]:
```
#Creating a custom data frame
import pandas as pd
import numpy as np
dates = pd.date_range('20130101', periods=6)
df = pd.DataFrame(np.random.randint(low=0, high=10, size=(6, 4)), index=dates, columns
=list('ABCD'))
print(df)
```

```
            A  B  C  D
2013-01-01  9  7  6  2
2013-01-02  7  3  0  9
2013-01-03  2  4  2  7
2013-01-04  9  9  1  8
2013-01-05  2  9  1  2
2013-01-06  9  0  4  2
```

In [15]:
```
#Creating a custom data frame
import pandas as pd
import numpy as np
dates = pd.date_range('20130101', periods=6)
df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list('ABCD'))
print(df)
```

```
                   A         B         C         D
2013-01-01  0.735414  0.371794  2.181287 -0.256224
2013-01-02  1.041834  0.560393  0.368102  1.606140
2013-01-03 -0.419954 -0.939574 -0.765878 -0.725393
2013-01-04 -0.358126 -0.422891  1.829044 -0.808074
2013-01-05  1.287520 -1.021095  1.477140 -1.016435
2013-01-06 -1.840260  0.616133 -1.273795  1.205932
```

In [21]:
```
# creating a DataFrame by passing a dict..
import pandas as pd
df2 = pd.DataFrame({ 'A' : 1.,
                     'B' : pd.Timestamp('20130102'),
                     'C' : pd.Series(data=[1,2,3,4],index=list(range(4)),dtype='float3
2'),
                     'D' : np.array([3,4,5,6],dtype='int32'),
                     'E' : np.random.randint(low=0, high=10, size=(4)),
                     'F' : 'foo' })
print(df2)
```

```
     A          B    C  D  E    F
0  1.0 2013-01-02  1.0  3  9  foo
1  1.0 2013-01-02  2.0  4  8  foo
2  1.0 2013-01-02  3.0  5  1  foo
3  1.0 2013-01-02  4.0  6  7  foo
```

In [23]:
```
#Printing the first tow rows of the previous Data Frame
print(df2.head(2))
```

```
     A          B    C  D  E    F
0  1.0 2013-01-02  1.0  3  9  foo
1  1.0 2013-01-02  2.0  4  8  foo
```

In [24]:
```
 # Printing last 2 rows from end
print(df2.tail(2))
```

```
     A          B    C  D  E    F
2  1.0 2013-01-02  3.0  5  1  foo
3  1.0 2013-01-02  4.0  6  7  foo
```

In [25]:
```
print(df2.index)
```

```
Int64Index([0, 1, 2, 3], dtype='int64')
```

In [26]:
```
print(df2.columns)
```

```
Index(['A', 'B', 'C', 'D', 'E', 'F'], dtype='object')
```

In [27]:
```
# Underlying NumPy array
print(df2.values)
```

```
[[1.0 Timestamp('2013-01-02 00:00:00') 1.0 3 9 'foo']
 [1.0 Timestamp('2013-01-02 00:00:00') 2.0 4 8 'foo']
 [1.0 Timestamp('2013-01-02 00:00:00') 3.0 5 1 'foo']
 [1.0 Timestamp('2013-01-02 00:00:00') 4.0 6 7 'foo']]
```

In [28]:
```
# A quick statistic summary of our data# A quic
print(df2.describe())
```

```
         A         C         D         E
count  4.0  4.000000  4.000000  4.000000
mean   1.0  2.500000  4.500000  6.250000
std    0.0  1.290994  1.290994  3.593976
min    1.0  1.000000  3.000000  1.000000
25%    1.0  1.750000  3.750000  5.500000
50%    1.0  2.500000  4.500000  7.500000
75%    1.0  3.250000  5.250000  8.250000
max    1.0  4.000000  6.000000  9.000000
```

Let us work on some interesting csv data set

In [44]:
```
import pandas as pd
import numpy as np
import os
os.chdir("D:/dataset")
data = pd.read_csv("transfer.csv")
```

```
c:\python-36\lib\site-packages\IPython\core\interactiveshell.py:2785: DtypeWarning: Co
lumns (41,87,92,96,101) have mixed types. Specify dtype option on import or set low_me
mory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Indexing Dataframes using pandas

iloc method allows us to retrieve rows and columns by position.

In [50]:
```
data['id'] = [np.random.randint(0,1000) for x in range(data.shape[0])]
data.iloc[0:5,0:4]
```

Out[50]:

	Application No.	Employee Name	Post Code	Subject Code
0	104015146	UDAI KISHORA	LBR	NaN
1	100618307	RAM PRATAP TIWARI	TGT	SANS
2	100814080	KESHAV GOPAL	LBR	NaN
3	103810494	RAJBANSH PAUL	PGT	HIST
4	103810430	VINOD KUMAR PATHAK	PGT	PHYS

Here are some indexing examples:

- **data.iloc[:5,:] — the first 5 rows, and all of the columns for those rows.**
- **data.iloc[:,:] — the entire DataFrame.**
- **data.iloc[5:,5:] — rows from position 5 onwards, and columns from position 5 onwards.**
- **data.iloc[:,0] — the first column, and all of the rows for the column.**
- **data.iloc[9,:] — the 10th row, and all of the columns for that row.**

Indexing Using Labels in Pandas

In [52]:
```
data.loc[0:5:,:]
```

Out[52]:

	Application No.	Employee Name	Post Code	Subject Code	Employee Code	Region Code	Ro Name	Zone Name	Present Station Code	Stat
0	104015146	UDAI KISHORA	LBR	NaN	1423	1	Ahmedabad	WEST ZONE	23	SURA
1	100618307	RAM PRATAP TIWARI	TGT	SANS	1575	1	Ahmedabad	WEST ZONE	2	ANKL
2	100814080	KESHAV GOPAL	LBR	NaN	2361	1	Ahmedabad	WEST ZONE	3	BARC
3	103810494	RAJBANSH PAUL	PGT	HIST	4517	1	Ahmedabad	WEST ZONE	23	SURA
4	103810430	VINOD KUMAR PATHAK	PGT	PHYS	5588	1	Ahmedabad	WEST ZONE	23	SURA
5	100317578	Praveen Kumar Khandelwal	PGT	MATH	6269	1	Ahmedabad	WEST ZONE	1	AHME

6 rows × 114 columns

In [53]:
```
print(data.index)
```

```
RangeIndex(start=0, stop=43528, step=1)
```

In [54]:
```
data.loc[:5, 'Post Code']
```

Out[54]:
```
0    LBR
1    TGT
2    LBR
3    PGT
4    PGT
5    PGT
Name: Post Code, dtype: object
```

In [57]:
```
data.loc[:5,['Post Code','Subject Code']]
```

Out[57]:

	Post Code	Subject Code
0	LBR	NaN
1	TGT	SANS
2	LBR	NaN
3	PGT	HIST
4	PGT	PHYS
5	PGT	MATH

In [59]:
```
data['Total Displacement Count'].mean()
```

Out[59]:
```
-3.4011900385958462
```

In [61]:
```
# Boolean indexing in Pandas
score_filter = data["Total Displacement Count"] > 9
score_filter.head(10)
```

Out[61]:
```
0    False
1    False
2    False
3     True
4     True
5    False
6    False
7    False
8     True
9    False
Name: Total Displacement Count, dtype: bool
```

In [63]:
```
# select rows in data where score is greater than 9
score_filter = data["Total Displacement Count"] > 9
filtered_data = data[score_filter]
filtered_data.head()
```

Out[63]:

	Application No.	Employee Name	Post Code	Subject Code	Employee Code	Region Code	Ro Name	Zone Name	Present Station Code	
3	103810494	RAJBANSH PAUL	PGT	HIST	4517	1	Ahmedabad	WEST ZONE	23	SUR
4	103810430	VINOD KUMAR PATHAK	PGT	PHYS	5588	1	Ahmedabad	WEST ZONE	23	SUR
8	102621872	V. RAMESH	PGT	BIOL	7946	1	Ahmedabad	WEST ZONE	14	JAM
12	101515680	Kaluram Tanwar	TGT	PETR	8220	1	Ahmedabad	WEST ZONE	7	DAN BSF
17	103613514	ATUL KUMAR TIWARI	PGT	CHEM	8324	1	Ahmedabad	WEST ZONE	21	RAJ

5 rows × 114 columns

Let us try to use multiple conditions

- The post should be PGT
- The Subject should be COMP
- The Displacement count should be 10 or more

In [65]:
```
multi_filter = (data['Post Code']=='PGT') & (data['Subject Code']=='COMP') & (data["To
tal Displacement Count"] > 9)
filtered_data = data[multi_filter]
filtered_data.head()
```

Out[65]:

	Application No.	Employee Name	Post Code	Subject Code	Employee Code	Region Code	Ro Name	Zone Name	Present Station Code
577	102114103	RAVAL VISHNUBHAI RAVJIBHAI	PGT	COMP	52004	1	Ahmedabad	WEST ZONE	12
598	100812364	MAYURI PATEL	PGT	COMP	53964	1	Ahmedabad	WEST ZONE	3
600	103810468	S R WATTEMWAR	PGT	COMP	53976	1	Ahmedabad	WEST ZONE	23
602	101514327	Satish Chandra Jangir	PGT	COMP	53979	1	Ahmedabad	WEST ZONE	7
604	104122532	MANISHKUMAR KANTILAL PARMAR	PGT	COMP	53990	1	Ahmedabad	WEST ZONE	24

5 rows × 114 columns