Python for Data Science!!!

**Finding Data:**

Kaggle: Great for practice and challeges

SQL: everywhere in industry

Site like Kaggale: DrivenData, CrowdANALYTIX, innocentive, codaLab, DataHack.

**Loading Data:**

1. Manually: Last resort for spaghetti gles
2. Numpy.loadtxt: simple, homogenous files
3. Numpy.genfromtxt: smple,heterogeneous file
4. Pandas.read\_csv: highly flexiable reader
5. Pickle: save actual objects

**Loading Data Coding:**

Loading datasets

* Manual Loading
* np.loadtxt()
* np.loadfromtxt()
* np.read\_csv()
* pd.read\*
* pickel

**#import section**

import pandas as pd

import numpy as np

import pickle

filename = 'heart.csv'

**#Best way to load database read\_csv()**

df = pd.read\_csv(filename)

df.head()

**#LoadText Methods**

data= np.loadtxt(fname=filename,delimiter=',',skiprows=1)

print(data)

**#Loadgenfromtxt**

data= np.genfromtxt(filename,delimiter=',',dtype=None,names=True,encoding='utf-8-sig')

data

#Read file manually (Default option in pythong)

def load\_file(filename):

with open(filename,encoding='utf-8-sig') as f:

data, cols = [],[]

for i, line in enumerate(f.read().splitlines()):

if i == 0:

cols += line.split(',')

else:

data.append([float(x) for x in line.split(',')])

df = pd.DataFrame(data,columns=cols)

return df

load\_file(filename).head()

**#Pandas Vs Numpy**

data = df.to\_numpy()

#Will remove soon in future

data = df.values

**DataFrame**

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

**pandas.DataFrame**

A pandas DataFrame can be created using the following constructor −

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

**data:**

Data takes various forms like ndarray, series, map, lists, dict, constants and also another DataFrame.

**index**

For the row labels, the Index to be used for the resulting frame is Optional Default np.arange(n) if no index is passed.

**columns**

For column labels, the optional default syntax is - np.arange(n). This is only true if no index is passed.

**dtype**

Data type of each column.

**copy**

This command (or whatever it is) is used for copying of data, if the default is False.

**Create DataFrame**

A pandas DataFrame can be created using various inputs like −

* Lists
* dict
* Series
* Numpy ndarrays
* Another DataFrame

In the subsequent sections of this chapter, we will see how to create a DataFrame using these inputs.

#Create an emply dataframe

import pandas as pd

df = pd.DataFrame()

print df

#Create Data from from List

import pandas as pd

data = [1,2,3,4,5]

df = pd.DataFrame(data)

print df

#Example 2

import pandas as pd

data = [['Alex',10],['Bob',12],['Clarke',13]]

df = pd.DataFrame(data,columns=['Name','Age'])

print df

#Example 3

import pandas as pd

data = [['Alex',10],['Bob',12],['Clarke',13]]

df = pd.DataFrame(data,columns=['Name','Age'],dtype=float)

print df

#Create DataFrame using dictionary

import pandas as pd

data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]}

df = pd.DataFrame(data)

print df

#Index DataFrame using Array

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

#Dataframe by dtype

dtype = [("A",np.int),("B",(np.str,20))]

data = np.array([(1,'Sam'),(2,'Alex'),(3,'John')], dtype=dtype)

df = pd.DataFrame(data,)

print df

**Save DataFrames:**

#save to CSV file

df.to\_csv('save.csv',index=False)

#save to pickle file

df.to\_pickle('save.pkl')

#Save in HDF (Hirarchical Data Format)

df.to\_hdf('save.hdf',key='data',format='table')

#Save in Feather format

df.to\_feather('save.fth')

**Inspecting DataFrame**

Df = pd.read\_csv(‘myfie.csv’)

#Shows top 5 rows

Df.head()

#Shows bottom 5 rows

Df.tail()

#Provides summery of DF

Df.info()

#Provide Corelations

Df.corr()

#Provides relations between columns

Df.describe()

#Shows sample row

Df.sample()

#Show count of values in year colums

Df.Year.value\_count()

#Show max value from all colums

Df.max()

#Shows unique values in columns

Df.Year.uniques()

#Shows Columns name of DF

Df.colums

**Basics of python Visualisations**

**#import section**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read\_csv('heart.csv')

**##BAR PLOTS**

**chest\_pain = df.groupby(by='cp').median().reset\_index()**

#pandas inbuild library

chest\_pain.plot.bar(x='cp',y='age');

chest\_pain.plot.bar(x='cp');

**#matplotlib version**

fig, ax = plt.subplots()

ax.bar(chest\_pain.cp,chest\_pain.age,label='age',color=['green','blue','pink','yellow'],edgecolor='k')

ax.set\_xlabel('CP')

ax.set\_ylabel('Age')

ax.legend(bbox\_to\_anchor=(1,1))

plt.show()

**#Seaborn version of born plot**

sns.set\_style('dark')

sns.barplot(data=df,x='cp',y='age',errcolor='white', label='age');

plt.legend();

**#Scatter Plots**

**#pandas inbuild libraries**

df.plot.scatter('age','trestbps',c='black');

**#Matplotlib version**

fig, ax = plt.subplots()

ax.scatter(data=df,x='age',y='trestbps',marker='\*',s=40,c='age',edgecolor='k',alpha=0.5,label='age')

ax.set\_xlabel('Age')

ax.set\_ylabel('trestbps')

ax.legend(bbox\_to\_anchor = (1,1))

plt.show()

**#Seaborn version of scatter plot**

sns.set\_style('white')

sns.scatterplot(x='age',y='trestbps',data=df,hue='age',edgecolor='k');

**#Line Plots**

ages = df.groupby('age').median().reset\_index()

ages.head()

**#Default Pandas version**

ages.plot.line('age',['chol','cp','trestbps']);

**#Pandas version of line**

fig, ax = plt.subplots()

ax.plot(ages['age'],ages['chol'],ls='--')

ax.set\_xlabel("age",fontsize=20)

ax.set\_ylabel('Cholestrol',fontsize=20);

**#seaborn version of line**

sns.lineplot('age','chol',data=ages);

**Plotting Methods**

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv('heart.csv')

**#plot scatter plots with two options**

fig, axes = plt.subplots(ncols=2,)

df.plot.scatter(x= 'age',y='chol',ax=axes[0])

df.plot.scatter(x= 'age',y='trestbps',ax=axes[1])

fig.tight\_layout()

**#Access and save plot**

fig2 = axes[1].get\_figure()

fig.savefig('Output.png',transperent=True,bbox\_inches= 'tight')

**#Exaples**

with plt.style.context('default'):

fig, axes = plt.subplots(ncols=3,sharey=True,gridspec\_kw={"width\_ratios":[2,2,2],"wspace":0})

y = 'age'

xs = ['chol','trestbps','thalach']

for x, ax in zip(xs,axes):

ax.scatter(x= df[x],y=df[y])

ax.set\_xlabel(x)

axes[0].set\_ylabel(y)

#save the file

fig.savefig('output1.png',dpi=100,bbox\_inches='tight')

fig.savefig('output1.pdf',dpi=100,bbox\_inches='tight')

**1D Visualisation:**

#import section

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read\_csv('heart.csv')

**#Histogram**

**#pandas version**

df.age.plot.hist(bins=20.);

**#matplot version**

plt.hist(df.age,bins=20);

**#Example 2: with vusual effects**

fig, ax = plt.subplots()

ax.hist(df.trestbps,bins=20,histtype='step',label='trestbps',edgecolor='r');

ax.hist(df.thalach,bins=20,histtype='stepfilled',label='thalach',alpha=0.3,edgecolor='g');

ax.legend();

**#BOX Plot**

**#pandas box version**

df[['trestbps','thalach']].plot.box();

**#matplotlibe version of**

plt.boxplot(df[['trestbps','thalach']].to\_numpy());

plt.boxplot()

**#seaborn version of boxplot**

sns.boxplot(data=df[['trestbps','thalach']]);

**#Example2**

sns.boxplot(data=df,x='cp',y='trestbps');

**#Violen Plot**

**#Matplotlib Version**

**#Example 1**

fig, ax = plt.subplots()

ax.violinplot(df[['trestbps','thalach']].to\_numpy());

ax.set\_xlabel('Data',fontsize=20);

ax.set\_ylabel('Numbers',fontsize=20);

**#Eample 2:**

fig, ax = plt.subplots()

ax.violinplot(df[['trestbps','thalach']].to\_numpy(),bw\_method=0.2);

ax.set\_xlabel('Data',fontsize=20);

ax.set\_ylabel('Numbers',fontsize=20);

**#sea born version**

sns.violinplot(data=df[['trestbps','thalach']],inner='quartile',bw=0.2);

**#BEE SWARM PLOT**

*If you want to go fancy, these can be fun for presentation. but let concise than other plot.*

sns.swarmplot(data=df[['trestbps','thalach']],size=3.5,color='g');

#Example 2:

sns.violinplot(data=df[['trestbps','thalach']],inner=None);

sns.swarmplot(data=df[['trestbps','thalach']],size=3.5,color='g',alpha=0.5);

**2D Visualisation:**

* plt.hist2d()
* plt.hexbin()
* df.plot.hexbin()
* plt.contour()
* plt.contourf()
* sns.kdeplot()
* sns.jointplot()
* sns.pairplot()

**#Import Sections**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read\_csv('meteorite-landings.csv')

**#Removing NA values for dataframes**

df = df.dropna(subset=['reclat','reclong'])

df = df[df.reclong < 300]

**#2D Histogram**

plt.hist2d(df.reclong,df.reclat,bins=200,vmax=4);

plt.colorbar();

**#2D HexPlot**

plt.hexbin(df.reclong,df.reclat,bins=200,vmax=4,lw=0.0)

plt.colorbar();

**#pandas version of hexbin**

df.plot.hexbin(x='reclong',y='reclat',vmax=2, gridsize=100,linewidth=None);

**CONTOUR**  
A contour line of a function of two variables is a curve along which the function has a constant value, so that the curve joins points of equal value. It is a plane section of the three-dimensional graph of the function f parallel to the-plane.

**#Lets create some data for contour**

spacing = np.linspace(0,10,200)

x,y = np.meshgrid(spacing,spacing)

z = (np.sin(x)+np.cos(y) + 2 \* np.arcsinh(x\*y))\*\*2

plt.contour(x,y,z,levels=20);

plt.colorbar();

#Example 2:

c =plt.contour(x,y,z,levels=20);

plt.clabel(c, inline=True,fmt="%0.1f",colors='black')

plt.colorbar();

#Example 3:

plt.contourf(x,y,z,levels=20);

plt.colorbar();

#Example 4:

plt.contourf(x,y,z,levels=20);

c = plt.contour(x,y,z,levels=20,colors='black');

plt.clabel(c, inline=True,fmt="%0.1f")

plt.colorbar();

**KDE Plot** (Kernal Dencity Estimation)

**#this is the called rejection sampling. a way to brute force sample any surface**

n = 50000

xs, ys = np.random.uniform(0, 10, n), np.random.uniform(0, 10,n)

zs = (np.sin(xs) + np.cos(ys) + 2 \* np.arcsinh(xs \* ys))\*\*2

zs /= zs.max()

passed = np.random.uniform(0,1,n) < zs

xs,ys = xs[passed],ys[passed]

plt.scatter(xs,ys, s=1, alpha=0.2);

**#Example 2:**

sns.kdeplot(xs,ys,bw=3.0,shade=True,shade\_lowest=True);

sns.kdeplot(xs,ys,bw=3.0,shade\_lowest=True);

**#JointPlot**

sns.jointplot(data=df,x='reclong',y='reclat');

Example 2:

sns.jointplot(data=df,x='reclong',y='reclat',kind='hex',gridsize=100,vmax=3,linewidth=0,marginal\_kws={'bins':100});

Example 3:

sns.jointplot(x=xs,y=ys, kind='kde')

#Example 4: sns.jointplot(xs,ys,kind='hex',gridsize=20,cmap='magma');

#Pairplot

sns.pairplot(data=df[['reclong','reclat','mass']])

**Basics of data manipulations:**

1. **Indexing:-** (.set\_index())  
   Indexing in pandas means simply selecting particular rows and columns of data from a DataFrame. Indexing could mean selecting all the rows and some of the columns, some of the rows and all of the columns, or some of each of the rows and columns. Indexing can also be known as Subset Selection.

import numpy as np

import pandas as pd

#import datafiles

df = pd.read\_csv('AB\_NYC\_2019.csv')

df.head(3)

df2 = df.set\_index('id')

Example:

df2.name[2539]

df2.host\_name[3647]

df3.reset\_index()

1. **Sorting:**

* sort\_index()
* sort\_values()

df.sort\_index(ascending = False).head(3)

df.sort\_values(['neighbourhood\_group','host\_name'],ascending=[False,True]).head(3)

ReCap:

* set\_index()
* reset\_index()
* Sort\_index()
* soft\_values()
* unique()
* value\_count()
* rank()

**Slicing:**

**-Slicing Rows**

df['price'].head()

#Multiple columns

df[['host\_name','host\_id']].head()

**-Filtering Rows**

df.host\_name == 'John'

mask = df.host\_name == 'Tazz'

df[mask]

quick\_and\_cheap = (df.price < 300) & (df.minimum\_nights < 3)

df[quick\_and\_cheap].head(2)

reviews\_consistent = df[(df.reviews\_per\_month > 3) | (df.number\_of\_reviews > 50)]

reviews\_consistent.head(2)

mask = np.logical\_or((df.reviews\_per\_month > 3),(df.number\_of\_reviews > 50))

#'~' Logical inversion True becomes False and False become True.

df[~mask].count()

**-Filtering Rows and Columns togethers**

* .loc
* .iloc

***property*DataFrame.loc**

Access a group of rows and columns by label(s) or a boolean array.

.loc[] is primarily label based, but may also be used with a boolean array.

Allowed inputs are:

* A single label, e.g. 5 or 'a', (note that 5 is interpreted as a *label* of the index, and **never** as an integer position along the index).
* A list or array of labels, e.g. ['a', 'b', 'c'].
* A slice object with labels, e.g. 'a':'f'.

Also

* [**DataFrame.at**](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.at.html#pandas.DataFrame.at)
* Access a single value for a row/column label pair.
* [**DataFrame.iloc**](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.iloc.html#pandas.DataFrame.iloc)
* Access group of rows and columns by integer position(s).
* [**DataFrame.xs**](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.xs.html#pandas.DataFrame.xs)
* Returns a cross-section (row(s) or column(s)) from the Series/DataFrame.
* [**Series.loc**](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.loc.html#pandas.Series.loc)
* Access group of values using labels.

**#loc example**

df.loc[mask,['name','host\_name']].head()

df.loc[:,'name']

df.loc[mask,:].head(2)

**#Iloc exaples**

df.iloc[0,1]

df.iloc[1:5, 5:6]

**#Provide mask helper**

df.loc[df.price.between(100,500),'price'].head()

df.loc[df.price.isin([100,200]),'price'].head()

df == 'John'

(df == 'John').any()

#Shows True if any columns contains Value 'John'

### Basic of adding and removing Data

### -Modifying type of column:

### Common for time series, categoricals, or converting string to numeric

**#Date**

birthdate = pd.to\_datetime(df['Birth Date'],format='%m/%d/%Y')

birthdate.dt.year

zarya = pd.to\_datetime('1998-11-20')

df['age\_at\_zarya'] = (zarya - birthdate).astype('timedelta64[Y]')

df.head(2)

**#Categoricals**

*“Why use? information can be utilised by other libraries that pandas interfaces with, you can provie explicit sorting order rather than lexical order, and huge speed imporvements if you group on categories”*

df['Military Rank'].unique()

**#Change Datatype to chategor**

**#First Way**

df['Military Rank'] = df['Military Rank'].astype('category')

df['Military Rank'].dtype

**#second way**

pd.Categorical(df['Military Rank'])

**#Numerical and String conversations**

df.age\_at\_zarya.astype('str').astype('float').astype('int')

**#Removing & Adding Columns and Rows**

#Creating dummy dataframe for operations

df2 = df[['Name','Year','Group']].copy()

df2.head(2)

**#removing Columns**

df2.drop('Group',axis=1).head()

df2.drop(columns='Group').head()

df2.drop(columns=['Year','Name']).head()

**#removing first rows**

df2.drop(1).head()

**#Adding Rows**

df2.append({"Name":"Sonu","Year":2010,"Group":20.0},ignore\_index=True)

df\_sis = pd.DataFrame({"Name":['Didi'],"Year":[2019],"Group":[15.0]})

df\_sis

df2.append(df\_sis,ignore\_index=True)

**#Adding Columns**

**#Way 1**

df2['Col1'] = 'Sonu'

df2.head()

**#Way 2**

df2.assign(col2='Shaikh').head()

**#Way 3**

df2.insert(0,'FirstName',df2.Name.str.split(' ',1,expand=True)[0])

df2.head()

**#Transpose Rows to Columns and Columns to Rows**

**Df2.T**