Python 3 cheat sheet

Ermano Buikis

import sys
print(sys.path[0]+'/folder/file.py') ← improta file path
runfile('/File/Path/hello.py', wdir=r'/File/Path') ← run file
%reset ← resetta le varialbili

% matplotlib inline Show inline results
% matplotlib notebook Interactive plots

pwd ← *where I am on the file directory*

dir() ← *lista tutte le variabili della console*

df.DataFrame(dictionary)
df.**describe**()
df.**index**
df.**columns**

sns.heatmap(df.isnull(), ytickslabels=False, cbar=False, cmap=’viridis’)

df .**load_csv**(path) ← *read*
df .**to_csv**(path) ← *write*

df**.dropna**(thresh = 2, axis =1)
thresh : scarta tutte le righe che hanno almeno 2 NaN
axis : agisce su: 1 colonne, 0 righe

df.**fillna**(value = ‘fill’) ← value : valore con il quale i NaN vengono sostituiti
df[‘A’].fillna(value = df[‘A’].mean())

df**.iloc**[index_name_i] ← **Selecting**

df**.loc**[[index_name_i],[column_name_i]]
df**.loc**[[row1, row2], [col1, col2]]

new = df **[**‘A’**]** **← Subsetting**
new = df.A

df.**drop**([‘col1’,’col2] , axis = 1 , inplace = True)
← *axis: 1 colonne, 0 righe | inplace: substitute new df with the old one*

df.**xs**(‘A‘, level = ‘B’) ← **Multi Index** *level : nome colonna su cui agire*

Conditional DataFrame

new = df [df [‘A’] > 7] [[‘A‘,’B’]] ← *filter and subsetting*
new = df [(df [‘A’] > 7) **&** (df[‘B] ==’NaN’) **|** (df.C > 2)] ← *triple filter* { & : and , | : or }

data = [(‘A’,’C’),(‘B’,’D’)]
pd.MultiIndex.from_tuples(data)

Group by

```
diz = {'company':['a','b','a','b','c'], 'sales':[2,3,5,7,3]}
df = pd.DataFrame(diz)

df2 = df.groupby('company').count().loc['a']          ←  select only one column
                                   .mean()
                                   .std()
                                   .describe()
                                   .describe().transpose()
```

pd.**concat**([df1, df2] , axis = 0)   ← *Concatenating two different df*

pd.**merge**(df_left, df_rigth, how = ‘inner’ , on=’key’)   ← *merge* : *join on same column keys*

df_left.**join**(df_right)   ← join on same index keys

df[‘col1’]**.value_count()**   ← *count element for that column*

df[‘col1’]**.nunique()**   ← *Number of unique values*

df[(df[‘col1’] > 2) & (df[‘col2’] < 5)]   ← *Conditional selection*

df[‘col1’]**.apply(name_function)**   ← *Apply function*

df[‘col1’]**.apply(lambda x: x*2)**   ← *Apply function with lambda*

df**.sort_values(by=’col1’, axis=0)**   ← *Apply function with lambda*

df.isnull()

df.**pivot_table(values = ‘Col1’ , index = [‘Col2’,’Col3’] , columns = [‘Col4’])**

DATA SOURCE

- **XML**
- **CSV**
  - df = pd.**read**_csv(‘file.csv’, sep=’ ’, index_col=’col1’ , parse_date = True)
  - df.**to_csv**(‘new_name‘, index=False)
- **beautifulsoup**
- **SQL**
  - from sqlalchemy import create_engine
  - engine = create_engine(‘ sqlite:///:memory:’)   ← *create ligth temporary SQL engine*
  - df.to_sql(‘my_table’,engine)   ← *post data frame into sql object*
  - sqldf = pd.read_sql(‘my_table’, con = engine)   ← *read data*
- **EXCEL**
  - df = pd.read_xcels(‘file.xlsx’, sep=’ ’)
- **HTML**
  - df = pd.read_html(URL)

pd.get_dummies(df[‘sex’)

```
F M
0 1
1 0
1 0
...
```

PLOT

```
import seaborn as sns
sns.joinplot( x = ‘col1’ , y =’col2’ , data = df, kind = ’hex’)
                                                  kind = ’reg’
                                                  kind = ’kde’
```

```
sns.pairplot( df , hue = ‘col_categorical’ , palette=’coolware’ )

sns.rugplot( df[‘col1’] )  ← only one column

sns.kdeplot( df[col1] )


df.plot.hist()

df.plot.area()

df.plot.bar( Stacked=True)

df.plot.line( x, y , figsize=(a,b) , lw= )

df.plot.scatter(x , y, cmap=’coolwarm’, c=’column3’ )

df.plot.hexbin( x, y, gridsize= , cmap= )
```

Kernel Density Estimator

```
df.plot.kde()

df.plot.density()
```

PLOTLY & CUFFLIN

(interactive plot)

```
%matplotlib inline
from pyplot import iplot                               ← import

df.iplot(kind=’hist’ )                                 ← histogram

.iplot( kind = ‘scatter’, x=’col1’, y=’col2’, mode=’markers’ )

.iplot( kind = ‘scatter’, x=’col1’, y=’col2’, mode=’markers’ )              ← scatter plot

.iplot( kind = ‘bar’, x=’Category_colum’, y=’Numerical_colun’)              ← barplot

.iplot( kind = ‘box’)                                                       ← boxplot

.iplot( kind = ‘surface’ , colorscale=’rdylbu’ )                            ← 3D surface plot

df[[‘A’,’B’]].iplot( kind = ‘spread’ )                                      ← Spread plot
.iplot( kind = ‘bubble’ ,x=’col1’, y=’col2’, size=’col3’ )                  ← Scatter plot with dimension point

df.scatter_matrix()                      ← Scatter matrix
```

Aggregate function

```
df.count().iplot()
df.sum().iplot()
```

HEATMAP

sns.heatmap

sns.clustermap( )

Dates formatting

import matplotlib.dates as dates

idx = data.index

idx = data.iloc['2007-01-01':'2008-01-01']   Filter rows by date

fig, ax = plt.subplot()
ax.plot_date(idx, stock, '-')
plt**.tigth_layout**()   - Format size
fig**.autofmt_xdate**()   - Format date label

ax.xaxis.grid(True)   - Set Grid

Change date label

fig, ax = plt.subplot()
ax.plot_date(idx, stock, '-')

ax**.xaxis.set_major_locator**(dates.MonthLocator())   Major
ax**.xaxis.set_major_formatter**(dates.DateFormatter('%b%y'))

ax**.xaxis.set_minor_locator**(dates.MonthLocator())   Minor
ax**.xaxis.set_minor_formatter**(dates.DateFormatter('%b%y'))

Style

plt.style('ggplot')

Pandas Datareader

from **pandas_datareader.data** import **Options**

facebook_stock_option = **Options**(‘FB’ , ‘google’)

option_df = facebook_stock_option**.get_options_data**(expiry = facebook_stock_option**.expiry_dates**[0])

QUANDL

Retrive data stocks with Python API

import **quandl**

data = quandl**.get**(“WIKI/FB.1”)**.get_table**(‘’)   WIKI/FB.1 take just the first column, use WIKI/FB for whole dataset

DateTime

from datetime import datetime

date_string = datetime(2016, 1, 1)   convert to datetime

df[‘date’] = pd.**to_datetime**(df[‘date’])   convert column to datetime

df.**set_index**(‘Date’ , inplace = True)   set date column as index

df = pd.read_csv(‘file.csv’, **index_col**= ‘Date’ , **parse_date** =True)   read data and parse date while reading

Resample

df. **resample**(**rule** = , **how** =, **axis** = , **fill_method** =)

Shift data

df.**shift**(**periods** = -7)   int(+1, -7,+3)

df.**tshift(freq** = ‘M’)   Shift index of 1 month

Rolling and Expanding

Rolling mean = Moving average

df**.rolling**(**window** = 7).mean()

df**.expanding**()

LINEAR REGRESSION

```
sns.lmplot(x, y, data = DF )
```

MULTIPLE REGRESSION

```
X = df[“col_1”, … ,“col_N”] ← several features
Y = df[“Target”]

from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X , y, test_size=0.1, random_state = 101 )


from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit( X_train, y_train)
lm.intercept_
lm.coef_

corr_df = pd.DataFrame(lm.coef_, X.columns, columns = [‘Coeff’]

predictions = lm.predict( X_test )

plt.scatter( y_test, predictions )  ← must be a straigth line

sns.distplot( (y_test-predictions) ) ← Histogram of residuals

from sklearn import metrics
metrics.mean_absolute_error( t_test, predictions )

metrics.mean_sqared_error ← mean sqared error
np.sqrt metrics(.mean_sqared_error) ← Root mean sqare error
```