SEABORN

```
import seaborn as sns
```

Instogramma sovrapposto

```
data = np.random.multivariate_normal([0, 0], [[5, 2], [2, 2]], size=2000)
data = pd.DataFrame(data, columns=['x', 'y'])
for col in 'xy':
    plt.hist(data[col], normed=True, alpha=0.5)
```

Gaussiana sovrapposta

```
for col in 'xy':
    sns.kdeplot(data[col], shade=True)
grafico=sns.lmplot(data, x='nome_col_x',y=) #linear model

sns.distplot(data['colname'],bins=numbins) #instogramma
sns.boxplot(data,x,y)

sns.distplot(data['x']) #applico la gaussiana all'instogramma
sns.distplot(data['y']);

tips = sns.load_dataset('tips')  # LOAD
tips.head()

with sns.axes_style(style='ticks'):
    g = sns.factorplot("x", "y", "sex", data=tips, kind="box")
    g.set_axis_labels("x", "y");
```

Violin Plot

```
men = (data.gender == 'M')
women = (data.gender == 'W')
with sns.axes_style(style=None):
    sns.violinplot("age_dec", "split_frac", hue="gender", data=data,
                   split=True, inner="quartile",
                   palette=["lightblue", "lightpink"]);
```

PANDAS

```
pd.info() Info
df.describe(include='all') summarize
.colums Columns
df.values Values
.head(5) Head
.tail(5) Tail

.drop(nome_colonna,axis=0)  elimina colonna

.drop(nome_riga,axis=1)  elimina riga
.drop("Year", axis=1, inplace=True)

.lock(label,index) lock
.pylock(index) pylock
.copy()      copy
len(df)       len

 data = pd.read_csv('mydata.csv') Read CSV
 data = pd.DataFrame(np.random.rand(10,5), columns = list('abcde')) DataFrame
 data=pd.DataFrame(data=[f,m], index=['a','b'], columns=[3,4,5,6], dtype=None, copy=False)

DF.to_csv("datasets/DF.csv") DF Save CSV
df[["COL3", "COL5"]].to_csv("datasets/df.csv", index=False)   DF Columns Save CSV


df[1][2]==3 Assegnazione

newDf= df[(df['Year']== 2015) & (df["Month"] == "February")] Extract and filter

df.["nomeColonna"].head()
df.nomeColonna.head()

std(), var(), mean(), median(), max(), min(), abs()

df.head(10)['ColName1'].values

count()  Count  non NaN
```

clip(lower=-10,upper=10) **Clip** Trim values at input thresholds

df.plot.hist() **Histogram**
df.plot.scatter(x='w',y='h') **Scatter**
plt.scatter(x,y,c='green',s=100) #color, size
df.rolling(n) **Rolling**
pd.merge(ydf, zdf) **Merge Df**

df.dropna() **Dropna**
df.dropna(inplace=True)

df.drop('nomeColonna') **Drop**

.loc([nome colonna]) **Seleziona LOC**
df.loc[:,'x2':'x4'] #Select all columns between x2 and x4
df.loc[df['a'] > 10, ['a','c']] #Select rows meeting logical condition,and only the specific columns (a,c)
df.loc[:, 'foo':'sat'] select all rows, and all colums between 'foo' and 'sat'

.iloc([indiceRiga : indiceColonna]) **Seleziona ILOC**
df.iloc[:,[1,2,5]] # Select columns in positions 1, 2 and 5 (first column is 0).
df.iloc[10:20] #Select rows by position
df.iloc[1:5, 2:4]

Integer slicing
df.ix[:4] #row
df.ix[:,'A'] #col
df['A'] #col

df[['width','length','species']] **Select colums**

df.filter(state='Italia') **Filter**

df.drop_duplicates() **Drop duplicates**
df.sample(frac=0.5) **Select sample**

df.dtypes **column types**
df.empty **Empty** return True if obj is empty
df.ndim **Ndim**
df.size **Size** df (n rows * n col)
df.shape **shape** tuple (r,c)

examples
df['w'].value_counts() **Count values frequencies** with each unique value of variable

df_new1= df.copy()[df['nomeColonna1':'nomeColonna2']]
df_new1= df.copy()[df['nomeColonna1']=='Italia']

df['Date']=pd.datetime(df['Date']) **DateTime format column**

df_new2=df_new.set_index('NomeColonna') **Set Index** prende la colonna come indice

df['date'].assign(a=lambda df: df.a / 2)

df_new2['NomeColonna'].plot() **Plot**

d_new['NomeColonnaNuova']=df_new2['NomeColonna'].rolling(20).mean().plot(figsize=(8,5),legend=False) **add new column**

df.sort_index(inplace=True) **Sort Index**
df.sort_values(by='nomeColonna1',ascending=True,inplace=True) **Sort Values**

df = df.join(df_new1['nomeCol']) **Join**
df.rename(columns={"county": "County", "st": "State"}, inplace=True) **Rename**

state_abbv_dict = state_abbv.to_dict()['Postal Code'] **Df to Dict**
dic=df.to_dict() **Df to Dict**

df1['State'] = df1['State'].map(df2) **Map**

Sostituire valori nominali con numeri
color_dict = {"J": 1,"I": 2,"H": 3,"G": 4,"F": 5,"E": 6,"D": 7}
df['color'] = df['color'].map(color_dict)

list_nome_colonna=list(set(df['nomeColonna'].values.tolist())) **DF To List**

df2 = df.groupby("State") **Group By**
df2.get_group("Alabama").set_index("Year").head() **Get Group, Set Index**

issue_df = df[df['NomeColonna']==0] **Individuare NaN**
issue_df['State'].unique() **Unique**

act_min_wage.replace(0, np.NaN).dropna(axis=1).corr().head() **Replace Nan**
(axis 1 == columns. 0 is default,0 is for rows)

df [['Col1' , 'Col2']].corr() **Correlation**
df [['Col1' , 'Col2']].cov() **Covariance**

Trasformo i dati di una colonna in colonne diverse secondo le categorie della colonna

```
g_df=pd.DataFrame()
for col in df['nomeColonna'].unique():
  new_df = df.copy()[df['NomeColonna']==col] # crea nuovo df
  new_df = new_df.set_index("date", inplace=True)
  new_df = new_df.sort_index(inplace=True)
  new_df = new_df.join(df['{}_colonna'.format(str(col))])
```

Heat Map Correlation

```
corr=df.corr().head()
import matplotlib.pyplot as plt
plt.matshow(corr)
plt.show()
```

Rinominare le colonne

```
labels = [c[:2] for c in min_wage_corr.columns]
fig = plt.figure(figsize=(12,12))  # figure so we can add axis
ax = fig.add_subplot(111)  # define axis, so we can modify
ax.matshow( Df , cmap=plt.cm.RdYlGn)
ax.set_xticks(np.arange(len(labels)))
ax.set_yticks(np.arange(len(labels)))
ax.set_xticklabels(labels)
ax.set_yticklabels(labels)
ax1.xaxis.label.set_color('c')
ax1.yaxis.label.set_color('r')
ax1.set_yticks([0,25,50,75])
plt.show()
```

New Dfs from one Df using Group By

```
for name, group in df.groupby("State"):
    if act_min_wage.empty:
        act_min_wage = group.set_index("Year")[["Low.2018"]].rename(columns={"Low.2018":name})
    else:
        act_min_wage = act_min_wage.join(group.set_index("Year")
[["Low.2018"]].rename(columns={"Low.2018":name}))

grouped_issues.get_group("Alabama")['Low.2018'].sum()

for state, data in grouped_issues: # another way
    if data['Low.2018'].sum() != 0.0:
        print("Some data found for", state)
```

Rielabora Df, da 1 df a molti diversi, raggruppati per una features

```
df1 = pd.DataFrame()
for name, group in df.groupby("State"): # raggruppa il DF secondo gli Stati
    if df1.empty:
         # se vuoto, setta l' indice agli anni, e la colonna visualizzata e' Col1, con nome==name
        df1 = group.set_index("Year")[["Col1"]].rename(columns={"Col1":name})
    else: # sedf e' pieno, aggiungi la colonna Col1 con nome==name e indice Year
        df1 = df1.join(group.set_index("Year")[["Col1"]].rename(columns={"Col1":name}))
# cosi' tutti gli stati sono in colonne diverse, mentre i loro valori della colonna1 sono visualizzati, ognuno in ordine
crescente della data
```

Replace

```
df1 = df1.replace(0, np.NaN).dropna(axis=1)
```

NUMPY

```
a=np.array()
a.ndim
.shape
.stype
.itemsize
.size()
.shape(row,col)
.zeroes(row,col)
.flatten() #return array = 1 Dim
.random.rand(row,col)
.random.randint(max, (row,col))
.random.normal(mu, sigma)(row,col)
.random.seed(seed_value)

.dot(v1,v2) #prodotto scalare
.sum(axis=1) # somma sulle righe
.sum(axis=0) # somma sulle col

np.arange(start,stop)
np.linspace(start,stop,size)
```

MATPLOTLIB

```
import matplotlib.plotlib as plt
with open('example.txt','r') as csvfile: load CSV
    plots = csv.reader(csvfile, delimiter=',')
    for row in plots:
        x.append(int(row[0]))
        y.append(int(row[1]))

x, y = np.loadtxt('example.txt', delimiter=',', unpack=True) load TXT
plt.plot(x,y, label='Loaded from file!')

bins = [0,10,20,30,40,50,60,70,80,90,100,110,120,130] # sono da dove partono le barre
plt.hist(population_ages, bins, histtype='bar', rwidth=0.8) HIST

plt.scatter(x,y, label='skitscat', color='k', s=25, marker="o") SCATTER

days = [1,2,3,4,5] #X        STACK PLOT
sleeping = [7,8,6,11,7] #Y0
eating =   [2,3,4,3,2] #Y1
working =  [7,8,7,2,2] #Y2
playing =  [8,5,7,8,13] #Y3
plt.plot([],[],color='m', label='Sleeping', linewidth=5) # legend features
plt.plot([],[],color='c', label='Eating', linewidth=5)
plt.plot([],[],color='r', label='Working', linewidth=5)
plt.plot([],[],color='k', label='Playing', linewidth=5)
plt.stackplot(days, sleeping,eating,working,playing, colors=['m','c','r','k'])
plt.legend()

slices = [7,2,2,13] PIE CHART
activities = ['sleeping','eating','working','playing']
cols = ['c','m','r','b']
plt.pie(slices,
        labels=activities,
        colors=cols,
        startangle=90,
        shadow= True,
        explode=(0,0.1,0,0), # Explode: If we wanted to pull out the first slice a bit, we would do 0.1,0,0,0.
        autopct='%1.1f%%')
fig = plt.figure(figsize=plt.figaspect(2.0))
fig.add_axes()
```

Subplot

```
fig = plt.figure()
ax1 = fig.add_subplot(221)# row-col-num #221 means 2 tall, 2 wide, plot number 1.
ax2 = fig.add_subplot(222)
ax3 = fig.add_subplot(212)
ax1 = plt.subplot2grid((6,1), (0,0), rowspan=1, colspan=1)
ax2 = plt.subplot2grid((6,1), (1,0), rowspan=4, colspan=1)
ax3 = plt.subplot2grid((6,1), (5,0), rowspan=1, colspan=1)
fig3, axes = plt.subplots(nrows=2,ncols=2)
plt.plot(x, y)
plt.legend('ABCDEF', ncol=2, loc='upper left');
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(x, y, color='lightblue', linewidth=3)
ax.margins(x=0.0,y=0.1)
ax.set(xlim=[0,10.5],ylim=[-1.5,1.5])
plt.setp(lines,color='r',linewidth=4.0)
fig.colorbar(im, orientation='horizontal')
plt.savefig('foo.png')
fig.tight_layout()
plt.show()
plt.close()
plt.title()
plt.xlabel()
plt.bar(nomi, pesi, colori)
plt.axes([x,y,larghezza,altezza])
plt.subplot(n_righe_plot, n_col_plot, indice_del_grafico=(row,col)) # 0<x<1
```

`plt.subplots_adjust(left=0.09, bottom=0.20, right=0.94, top=0.90, wspace=0.2, hspace=0)` **subplots adjust**

```
ax1.plot([],[],linewidth=5, label='loss', color='r',alpha=0.5)
```

`plt.savefig(path,nomefile)` **savefig**
`plt.rcparams['figure.figuresize']=[12,8]` **parametri immagine**
`img=mpimg.imread(path)` **legge un immagine**
`plt.tight.layout()` **ottimizza dimensione**

`plt.legend()` **legend**

Styles

```
from matplotlib import style
print(plt.style.available)
style.use('dark_background')
```

`plt.style.aviable` **lista stili grafici disponibili**

```
plt.style.use("nome stile")
```

Caratteristiche immagine

```
img.shape
img.imshow(img,cmap=)
```

`img.xtricks([])` **Tricks**
`img.xlim()` **Limits**
`img.grid()` **Grid**
`img.Annotate("string", xy=(num,num), xytest=("string"))` **Annotation**

```
ax1.annotate('Bad News!',(date[9],highp[9]),
             xytext=(0.8, 0.9), textcoords='axes fraction',
             arrowprops = dict(facecolor='grey',color='grey'))
```

`img.text((x,y),"text", size)` **Text**

Fill immagine

```
ax1.fill_between(date, 0, closep)
  ax1.fill_between(date, closep, closep[0],where=(closep > closep[0]), facecolor='g', alpha=0.5)
  ax1.fill_between(date, closep, closep[0],where=(closep < closep[0]), facecolor='r', alpha=0.5)
```

`ax1.grid(True)#, color='g', linestyle='-', linewidth=5)` **Griglia immagine**

CLASS

```
class cibo:
  def __init__(self,carboidrati=0,proteine=0,grassi=0)
      self.proteine=proteine
      self.carboidrati=carboidrati
  def calcolacalorie(self):
      return(self.carboidrati *4 + self.proteine *4 + self.grassi *9)

pasta=cibo(proteine=12,carboidrati=32)
print(pasta.carboidrati)
cibo.calcolacalorie=calcolacalorie
```

FILE

```
file_handler=open('/Users\Ermano\Desktop\Python\sequenzagfp.fasta','a')
for i in list(range(0,10)):
    n=rnd.choice(list(diz.keys()))
    (diz[n])= diz[n]+1
    file_handler.write(str(rnd.choice(list(diz.keys()))))
```