In [1]:

```
%load_ext watermark
%watermark
%matplotlib inline
```

```
2019-05-30T21:24:43+02:00

CPython 3.6.5
IPython 6.4.0

compiler   : GCC 7.2.0
system     : Linux
release    : 5.1.5-arch1-2-ARCH
machine    : x86_64
processor  :
CPU cores  : 4
interpreter: 64bit
```

VISUALIZACIÓN DE DATOS AVANZADA

En este capitulo se trataran los métodos para poder personalizar las gráficas de Matplotlib y como cargar Estilos ya predefinidos. Además se realizaran ejemplos con otras librerías interesantes para la visualización de datos:

- IPyWidgets.
- Cartopy.
- Seaborn.
- BokehJS

Carga de Datos y Preparacion de DataSet

Como en el apartado de visualización básica de datos utilizaremos el Boston Housing Dataset. Recopilado en 1976 y publicado en Berkeley

In [ ]:

```
import pandas as pd
df = pd.read_csv("boston_dataset.csv")

#renombramos las variables
df = df.rename(columns={
    "TOWN":"CIUDAD",
    "CRIM":"INDICE_CRIMEN",
    "ZN":"PCT_ZONA_RESIDENCIAL",
    "INDUS":"PCT_ZONA_INDUSTRIAL",
    "CHAS":"RIO_CHARLES",
    "NOX":"OXIDO_NITROSO_PPM",
    "RM":"N_HABITACIONES_MEDIO",
    "AGE":"PCT_CASAS_40S",
    "DIS_EMPLEO":"DISTANCIA_CENTRO_EMPLEO",
    "RAD":"DIS_AUTOPISTAS",
    "TAX":"CARGA_FISCAL",
    "PTRATIO":"RATIO_PROFESORES",
    "B":"PCT_NEGRA",
    "MEDV":"VALOR_MEDIANO",
    "LSTAT":"PCT_CLASE_BAJA"
})

df.head()
```

Out[ ]:

	CIUDAD	LON	LAT	VALOR_MEDIANO	INDICE_CRIMEN	PCT_ZONA_RESIDENCIAL	PCT_ZONA_INDUSTRIAL
0	Nahant	-70.955	42.2550	24.0	0.00632	18.0	2.31
1	Swampscott	-70.950	42.2875	21.6	0.02731	0.0	7.07

	CIUDAD	LON	LAT	VALOR_MEDIANO	INDICE_CRIMEN	PCT_ZONA_RESIDENCIAL	PCT_ZONA_INDUSTRIAL
2	Swampscott	-70.936	42.2887	34.7	0.01729	0.0	2.18
3	Marblehead	-70.928	42.2930	33.4	0.03237	0.0	2.18
4	Marblehead	-70.922	42.2980	36.2	0.06905	0.0	2.18

Personalización de Gráficos

Mediante los metodos de la librería **PyPlot** de **MatplotLib** veremos como especifciar titulos para los gráficos y como personalizar la forma de los punteros en gráficos, su tamaño y su color.

In [3]:

```
%matplotlib notebook
```

In [4]:

```
import matplotlib.pyplot as plt
```

In [5]:

```
#Cambiamos el punto con marker, color y el tamanio tan solo llamando a los parametros
df.plot.scatter(x="N_HABITACIONES_MEDIO", y="VALOR_MEDIANO", marker="*", color="pink", figsize=(8,8
))

plt.title("Relacion entre el numero de habitaciones y el valor de las viviendas")

plt.xlabel("Numero medio de habitaciones")

plt.ylabel("Valor mediano de las viviendas ($1000s)")
```

Out[5]:

```
Text(0,0.5,'Valor mediano de las viviendas ($1000s)')
```

Damos un tamanio para las figuras por defecto en las librerias

In [6]:

```
import matplotlib as mpl

mpl.rcParams['figure.figsize'] = (8,8)
```

In [7]:

```
df.plot.scatter(x="N_HABITACIONES_MEDIO", y="VALOR_MEDIANO", marker="*", color="pink")

plt.title("Relacion entre el numero de habitaciones y el valor de las viviendas")

plt.xlabel("Numero medio de habitaciones")

plt.ylabel("Valor mediano de las viviendas ($1000s)")
```

Out[7]:

```
Text(0,0.5,'Valor mediano de las viviendas ($1000s)')
```

Establecer estilos en Matplotlib

Por defecto Matplotlib tiene un estilo definido, un aspecto muy característico y facilmente reconocible. Pero también permite personalizar los estilos de gráficas de una forma muy sencilla, utilizando hojas de estilos predefinidas y que vienen incluidas con Matplotlib

In [8]:

```
#mostramos la lista disponible de estilos en pyplot.
plt.style.available
```

Out[8]:

```
['fivethirtyeight',
 'bmh',
 'seaborn-talk',
 'seaborn-notebook',
 'seaborn',
 'seaborn-whitegrid',
 'ggplot',
 'tableau-colorblind10',
```

```
 'dark_background',
 'seaborn-deep',
 '_classic_test',
 'seaborn-colorblind',
 'seaborn-darkgrid',
 'seaborn-poster',
 'grayscale',
 'seaborn-dark',
 'seaborn-paper',
 'fast',
 'seaborn-pastel',
 'seaborn-white',
 'seaborn-ticks',
 'classic',
 'seaborn-bright',
 'Solarize_Light2',
 'seaborn-muted',
 'seaborn-dark-palette']
```

Podemos encontrar mas estilos aqui

In [9]:

```
plt.style.use("fivethirtyeight")
```

In [10]:

```
df.plot.scatter(x="N_HABITACIONES_MEDIO", y="VALOR_MEDIANO")

plt.title("Relacion entre el numero de habitaciones y el valor de las viviendas")

plt.xlabel("Numero medio de habitaciones")

plt.ylabel("Valor mediano de las viviendas ($1000s)")
```

Out[10]:

```
Text(0,0.5,'Valor mediano de las viviendas ($1000s)')
```

IPyWidgets

IpyWidgets es una librería que nos permite importar widgets FrontEnd para poder interactuar con las gráficos. Podemos invocarlo con interact.

In [11]:

```
from ipywidgets import interact
```

In [12]:

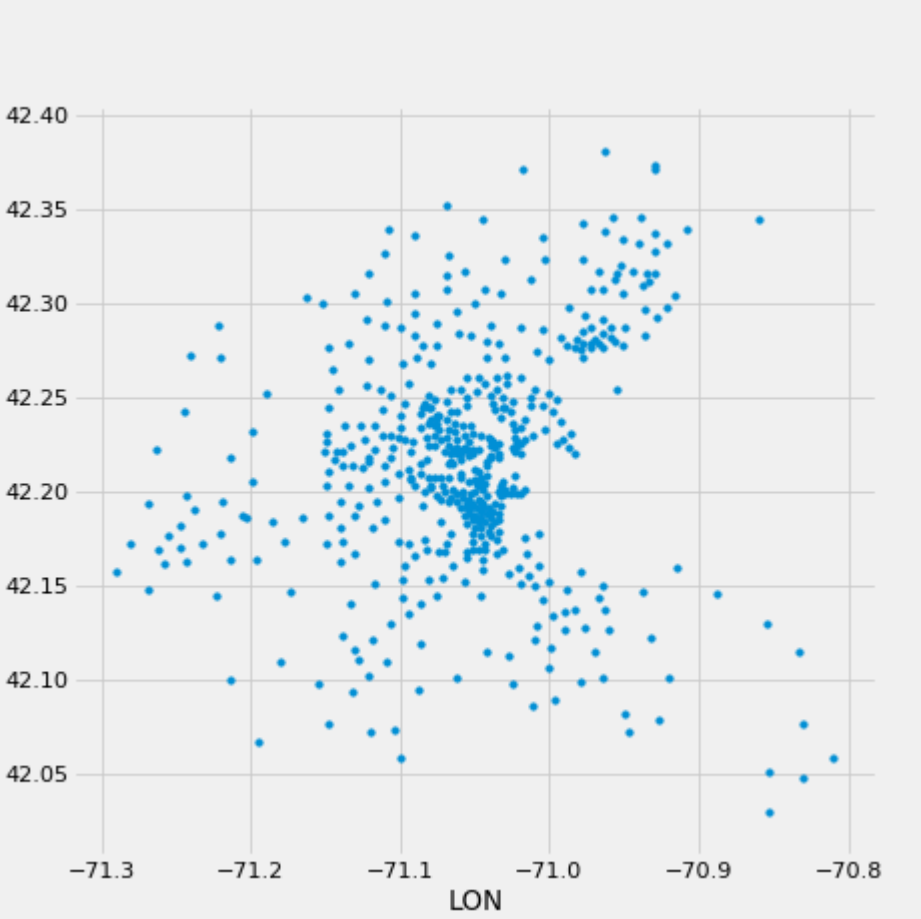
```
#creamos la funcion grafico varible para comparar la columna 1 seleccionable desde el ComboBox con el Valor Mediano
@interact(col1=df.columns.tolist())
def grafico_variable(col1):
    df.plot.scatter(x=col1, y="VALOR_MEDIANO")
    plt.title("{} vs VALOR_MEDIANO".format(col1))
```

In [13]:

```
#Indicamos a matplotlib que estamos trabajando con notebook para que se reescale mejor.
%matplotlib notebook
```

In [17]:

```
df.plot.scatter(x="LON", y="LAT")
```



Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd8027c7400>
```

Cartopy

Cartopy es una librería diseñada para procesar datos geoespaciales en orden para "plottear" mapas y poder realizar analisis de datos.

In [18]:

```
import cartopy.crs as ccrs

from cartopy.io import img_tiles
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-18-fca1b84c8aa4> in <module>()
----> 1 import cartopy.crs as ccrs
      2
      3 from cartopy.io import img_tiles

ModuleNotFoundError: No module named 'cartopy'
```

In [19]:

```
df.VALOR_MEDIANO.plot.kde()
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd8029c4208>
```

In [20]:

```
primer_quintil = df.VALOR_MEDIANO.quantile(0.2)
primer_quintil
```

Out[20]:

```
15.3
```

In [21]:

```
cuarto_quintil = df.VALOR_MEDIANO.quantile(0.8)
cuarto_quintil
```

Out[21]:

```
28.2
```

In [22]:

```
imagery = img_tiles.GoogleTiles()


ax = plt.axes(projection=imagery.crs)

limites_mapa = (-71.38 ,-70.77,42.03 , 42.47)

ax.set_extent(limites_mapa)

ax.add_image(imagery, 10)

df_primer_qt = df[df.VALOR_MEDIANO<primer_quintil]

df_tercer_qt = df[df.VALOR_MEDIANO>cuarto_quintil]


plt.plot(df_primer_qt.LON, df_primer_qt.LAT, transform=ccrs.Geodetic(), marker=".",
         markersize=10, color="red", linewidth=0, alpha=0.5)

plt.plot(df_tercer_qt.LON, df_primer_qt.LAT, transform=ccrs.Geodetic(), marker=".",
         markersize=10, color="green", linewidth=0, alpha=0.5)

plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-22-e458cdb2207d> in <module>()
----> 1 imagery = img_tiles.GoogleTiles()
      2
      3
      4 ax = plt.axes(projection=imagery.crs)
      5

NameError: name 'img_tiles' is not defined
```

Seaborn

Basada en matplotlib, se usa para hacer más atractivos los gráficos e información estadística en Python. Su objetivo es darle una mayor relevancia a las visualizaciones, dentro de las tareas de exploración e interpretación de los datos.

In [ ]:

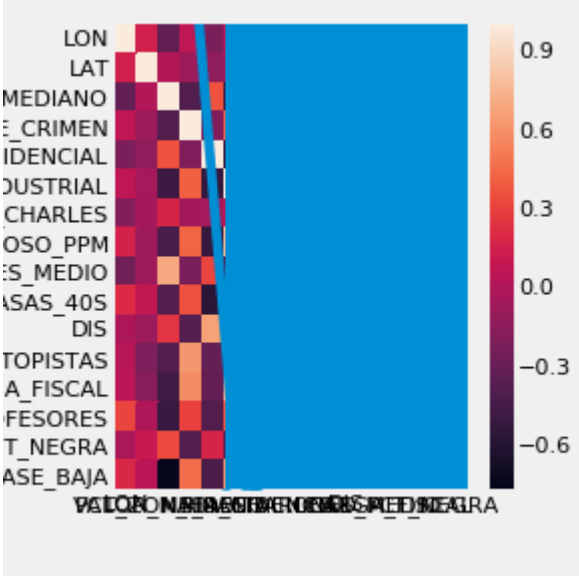
```
#especificamos a matplotlib para incluir los gráficos en el notebook
%matplotlib inline
```

In [23]:

```
import seaborn as sns
```

In [24]:

```
sns.lmplot(x="N_HABITACIONES_MEDIO", y="VALOR_MEDIANO", data=df)
```



Out[24]:

```
<seaborn.axisgrid.FacetGrid at 0x7fd7f99a5a58>
```

In [25]:

```
sns.heatmap(df.corr())
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd7f92aa668>
```

BokehJS

BokehJS es una librería que nos permitirá realizar graficas pensadas para mostrar gráficos en un navegador. Bokeh es una librería para visualizaciones interactivas diseñada para funcionar en los navegadores web modernos. Su objetivo es proporcionar una construcción elegante y concisa de gráficos modernos al estilo de D3.js, y para ampliar esta capacidad con la interactividad y buen rendimiento sobre grandes volúmenes de datos. Bokeh puede ayudar a cualquier persona a crear en forma rápida y sencilla gráficos interactivos, dashboards y aplicaciones de datos

In [26]:

```
import bokeh.plotting as bk

bk.output_notebook()
```

Loading BokehJS ...

In [27]:

```
df.INDICE_CRIMEN
```

Out[27]:

```
0       0.00632
1       0.02731
2       0.02729
3       0.03237
4       0.06905
5       0.02985
6       0.08829
```

```
7       0.14455
8       0.21124
9       0.17004
10      0.22489
11      0.11747
12      0.09378
13      0.62976
14      0.63796
15      0.62739
16      1.05393
17      0.78420
18      0.80271
19      0.72580
20      1.25179
21      0.85204
22      1.23247
23      0.98843
24      0.75026
25      0.84054
26      0.67191
27      0.95577
28      0.77299
29      1.00245
         ...
476     4.87141
477    15.02340
478    10.23300
479    14.33370
480     5.82401
481     5.70818
482     5.73116
483     2.81838
484     2.37857
485     3.67367
486     5.69175
487     4.83567
488     0.15086
489     0.18337
490     0.20746
491     0.10574
492     0.11132
493     0.17331
494     0.27957
495     0.17899
496     0.28960
497     0.26838
498     0.23912
499     0.17783
500     0.22438
501     0.06263
502     0.04527
503     0.06076
504     0.10959
505     0.04741
Name: INDICE_CRIMEN, Length: 506, dtype: float64
```

In [28]:

```
df["CRIMEN_QUINTIL"] = pd.qcut(df.INDICE_CRIMEN, 5)
```

In [29]:

```
df.CRIMEN_QUINTIL.cat.categories
```

Out[29]:

```
IntervalIndex([(0.00532, 0.0642], (0.0642, 0.15], (0.15, 0.55], (0.55, 5.581], (5.581, 88.976]],
              closed='right',
              dtype='interval[float64]')
```

In [30]:

```
from bokeh.palettes import brewer

colors = brewer["Spectral"][len(df.CRIMEN_QUINTIL.unique())]
colors
```

Out[30]:

```
['#2b83ba', '#abdda4', '#ffffbf', '#fdae61', '#d7191c']
```

In [31]:

```
p = bk.figure(
plot_width=600,
    plot_height=600,
    title="Habitaciones vs Valor vivienda vs crimen"
)

for i, quintil in enumerate(df.CRIMEN_QUINTIL.cat.categories):
    df_q = df[df.CRIMEN_QUINTIL==quintil]
    p.scatter(df_q.N_HABITACIONES_MEDIO, df_q.VALOR_MEDIANO, color=colors[i],
              legend="({}-{})".format(quintil.left, quintil.right)
              )

bk.show(p);
```

In [32]:

```
df.VALOR_MEDIANO.plot.hist()
```

Out[32]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd7f92aa668>
```

In [33]:

```
import numpy as np

hist, edges = np.histogram(df.VALOR_MEDIANO, bins=20)
```

In [34]:

```
hist
```

Out[34]:

```
array([ 9, 12, 18, 36, 41, 41, 84, 71, 72, 12, 23, 18, 17, 14,  6,  1,  5,
        5,  2, 19])
```

In [35]:

```
edges
```

Out[35]:

```
array([ 5.  ,  7.25,  9.5 , 11.75, 14.  , 16.25, 18.5 , 20.75, 23.  ,
       25.25, 27.5 , 29.75, 32.  , 34.25, 36.5 , 38.75, 41.  , 43.25,
       45.5 , 47.75, 50.  ])
```

In [36]:

```
p1 = bk.figure(title="Histograma valor viviendas", tools="save,hover", background_fill_color="#E8DD
CB")



p1.quad(top=hist,bottom=0, left=edges[:-1], right=edges[1:], fill_color="#026560")

bk.show(p1)
```