



Práctica de laboratorio: Delitos en San Francisco

Nombres: Armas Paul, Burbano Rodrigo, Guachamin Jemery

Objetivos

Demuestre sus conocimientos acerca del ciclo activo del análisis de datos mediante un conjunto de datos determinado, herramientas, Python y la libreta de anotaciones de Jupyter.

Parte 1: Importar los paquetes de Python

Parte 2: Cargar los datos

Parte 3: Preparar los datos

Parte 4: Analizar los datos

Parte 5: Visualizar los datos

Segundo plano/situación

En esta práctica de laboratorio, importará algunos paquetes de Python que nos necesarios para analizar un conjunto de datos que contiene información sobre delitos en San Francisco. Luego utilizará Python y Jupyter Notebook para preparar estos datos para su análisis, analizarlos, graficarlos y comunicar sus conclusiones.

Recursos necesarios

- 1 computadora con acceso a Internet
- Raspberry Pi versión 2 o superior
- Bibliotecas de Python: pandas, numpy, matplotlib, folium, datetime y csv
- Archivos de datos: Map-Crime_Incidents-Previous_Three_Months.csv

Parte 1: Importar los paquetes de Python

En esta sección, importará los paquetes de Python que son necesarios para el resto de esta práctica de laboratorio.

numpy

NumPy es el paquete fundamental para computación científica con Python. Contiene entre otras cosas: un potente objeto de matriz N dimensional y funciones sofisticadas (difusión).

pandas

Pandas es un código abierto, una biblioteca con licencia BSD autorizada que proporciona un alto rendimiento, estructuras de datos fáciles de usar y herramientas de análisis de datos para el lenguaje de programación de Python.

matplotlib

Matplotlib es una biblioteca gráfica para el lenguaje de programación de Python y su extensión matemática numérica NumPy.

folium

Folium es una biblioteca para crear mapas interactivos.

```
In [2]: pip install folium
```

```
Collecting folium
  Downloading folium-0.20.0-py2.py3-none-any.whl.metadata (4.2 kB)
Collecting branca>=0.6.0 (from folium)
  Downloading branca-0.8.1-py3-none-any.whl.metadata (1.5 kB)
Requirement already satisfied: jinja2>=2.9 in c:\users\lco\anaconda3\lib\site-packages (from folium) (3.1.6)
Requirement already satisfied: numpy in c:\users\lco\anaconda3\lib\site-packages (from folium) (2.1.3)
Requirement already satisfied: requests in c:\users\lco\anaconda3\lib\site-packages (from folium) (2.32.3)
Requirement already satisfied: xyzservices in c:\users\lco\anaconda3\lib\site-packages (from folium) (2022.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\lco\anaconda3\lib\site-packages (from jinja2>=2.9->folium) (3.0.2)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\lco\anaconda3\lib\site-packages (from requests->folium) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\lco\anaconda3\lib\site-packages (from requests->folium) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\lco\anaconda3\lib\site-packages (from requests->folium) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\lco\anaconda3\lib\site-packages (from requests->folium) (2025.6.15)
  Downloading folium-0.20.0-py2.py3-none-any.whl (113 kB)
  Downloading branca-0.8.1-py3-none-any.whl (26 kB)
Installing collected packages: branca, folium

----- 2/2 [folium]
```

```
Successfully installed branca-0.8.1 folium-0.20.0
Note: you may need to restart the kernel to use updated packages.
```

```
In [46]: # Code cell 1
%matplotlib inline
import numpy as np
```

```
import pandas as pd
import matplotlib.pyplot as plt
import folium
```

Parte 2: Cargar los datos

En esta sección, cargará el conjunto de datos de delitos en San Francisco y los paquetes de Python necesarios para analizarlo y visualizarlo.

Paso 1: Cargar los datos de delitos en San Francisco en un marco de datos.

En este paso, se importarán los datos de delitos en San Francisco de un archivo de valores separado por comas (csv) en una trama de datos.

```
In [47]: # code cell 2
# This should be a local path
dataset_path = './Data/Map-Crime_Incidents-Previous_Three_Months.csv'

# read the original dataset (in comma separated values format) into a DataFrame
SF = pd.read_csv(dataset_path)
```

In []: Para ver las primeras cinco líneas `del` archivo csv, se utilizará el comando '`head`'

```
In [48]: # code cell 3
!head -n 5 ./Data/Map-Crime_Incidents-Previous_Three_Months.csv
```

"head" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

Paso 2: Ver los datos importados.

A) Al ingresar el nombre del marco de datos en una celda, puede visualizar las filas superior e inferior de manera estructurada.

```
In [49]: # Code cell 4
pd.set_option('display.max_rows', 10) #Visualize 10 rows
SF
```

Out[49]:

	IncidentNum	Category	Descript	DayOfWeek	Date	Time
0	NaN	LARCENY/THEFT	GRAND THEFT FROM UNLOCKED AUTO	Sunday	08/31/2014	07:00:00 AM +0000
1	NaN	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Sunday	08/31/2014	07:00:00 AM +0000
2	NaN	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Sunday	08/31/2014	07:00:00 AM +0000
3	NaN	DRUG/NARCOTIC	POSSESSION OF METH-AMPHETAMINE	Sunday	08/31/2014	07:00:00 AM +0000
4	NaN	DRUG/NARCOTIC	POSSESSION OF COCAINE	Sunday	08/31/2014	07:00:00 AM +0000
...
30755	NaN	LARCENY/THEFT	PETTY THEFT SHOPLIFTING	Sunday	06/01/2014	07:00:00 AM +0000
30756	NaN	OTHER OFFENSES	DRIVERS LICENSE, SUSPENDED OR REVOKED	Sunday	06/01/2014	07:00:00 AM +0000
30757	NaN	ASSAULT	BATTERY	Sunday	06/01/2014	07:00:00 AM +0000
30758	NaN	ASSAULT	ASSAULT WITH CAUSTIC CHEMICALS	Sunday	06/01/2014	07:00:00 AM +0000
30759	NaN	OTHER OFFENSES	DRIVERS LICENSE, SUSPENDED OR REVOKED	Sunday	06/01/2014	07:00:00 AM +0000

30760 rows × 12 columns



b) Utilice la función 'columns' para ver el nombre de las variables en el marco de datos.

In [50]:

```
# Code cell 5
SF.columns
```

Out[50]: Index(['IncidentNum', 'Category', 'Descript', 'DayOfWeek', 'Date', 'Time', 'PdDistrict', 'Resolution', 'Address', 'X', 'Y', 'Location'], dtype='object')

¿Cuántas variables se incluyen en el marco de datos de SF (ignore el índice)?

El número de variables (columnas) es: 12

c) Utilice la función 'len' para determinar la cantidad de filas en el conjunto de datos.

```
In [17]: # Code cell 6
len(SF)
```

```
Out[17]: 30760
```

Parte 3: Preparar los datos

Ahora que cuenta con los datos cargados en el entorno de trabajo y determinó el análisis que desea realizar, es momento de preparar los datos para el análisis.

Paso 1: Extraer el mes y el día del campo de Date (Fecha).

'lambda' es una palabra clave de Python que define las reconocidas *funciones anónimas*.

'lambda' le permite especificar una función en una línea de código sin utilizar 'def' y sin definir un nombre específico para ella. La sintaxis para una expresión 'lambda' es:

'lambda' *parámetros: expresión*.

En este caso, la función 'lambda' se utiliza para crear una función en línea que seleccione solo los dígitos del mes de la variable Date (Fecha) e 'int' para convertir una representación de secuencia en un valor entero. Luego, la función *pandas* 'apply' se utiliza para aplicar esta función a una columna entera (en la práctica, 'apply' define implícitamente un bucle 'for' y pasa una por una las filas a la función 'lambda'). El mismo procedimiento se puede hacer para el día.

```
In [55]: # Code cell 7
SF['Month'] = SF['Date'].apply(lambda row: int(row[0:2]))
SF['Day'] = SF['Date'].apply(lambda row: int(row[3:5]))
```

Para verificar que estas dos variables se agreguen al marco de datos de SF, use la función 'print' para imprimir algunos de los valores de estas columnas y 'type' para verificar que estas columnas nuevas contengan, de hecho, valores numéricos.

```
In [56]: # Code cell 8
print(SF['Month'][0:2])
print(SF['Day'][0:2])
```

```
0    8
1    8
Name: Month, dtype: int64
0    31
1    31
Name: Day, dtype: int64
```

```
In [57]: # Code cell 9
print(type(SF['Month'][0]))
```

```
<class 'numpy.int64'>
```

Paso 2: Eliminar las variables del marco de datos de SF.

a) La columna 'IncidentNum' contiene varias celdas con NaN. En este caso, faltan los datos. Además, 'IncidentNum' no proporciona ningún valor al análisis. La columna se puede descartar del marco de datos. Una manera de eliminar variables que no desea en un marco de datos es mediante la función 'del'.

In [58]: # Code cell 10

```
del SF['IncidentNum']
```

b) De manera similar, el atributo 'Location' no se incluirá en este análisis. Puede descartarse del marco de datos.

O bien, puede usar la función 'drop' en el marco de datos y especificar que el *eje* es el 1 (0 para las filas) y que el comando no requiere una asignación a otro valor para almacenar el resultado (*inplace = True*).

In [59]: # Code cell 11

```
SF.drop('Location', axis=1, inplace=True, errors='ignore' )
```

c) Verifique que se hayan eliminado las columnas.

In [60]: # Code cell 12

```
SF.columns
```

Out[60]: Index(['Category', 'Descript', 'DayOfWeek', 'Date', 'Time', 'PdDistrict', 'Resolution', 'Address', 'X', 'Y', 'Month', 'Day'],
dtype='object')

Parte 4: Analizar los datos

Ahora que el marco de datos se ha elaborado con datos, es momento de analizar los datos.

Paso 1: Resumir las variables para obtener información estadística.

a) Utilice la función 'value_counts' para resumir la cantidad de delitos cometidos por tipo; luego seleccione 'print' para visualizar los contenidos de la variable *CountCategory*.

In [54]: # Code cell 13

```
CountCategory = SF['Category'].value_counts()  
print(CountCategory)
```

```
Category
LARCENY/THEFT           8205
OTHER OFFENSES          4004
NON-CRIMINAL            3653
ASSAULT                  2518
VEHICLE THEFT            1885
...
SEX OFFENSES, NON FORCIBLE    5
BAD CHECKS                 3
GAMBLING                   1
PORNOGRAPHY/OBSCENE MAT      1
BRIBERY                     1
Name: count, Length: 36, dtype: int64
```

- b) De manera predeterminada, los conteos se ordenan de forma descendente. El valor del parámetro opcional *ascendente* se puede configurar en *True* para invertir este comportamiento.

```
In [61]: # Code cell 14
SF['Category'].value_counts(ascending=True)
```

```
Out[61]: Category
GAMBLING                  1
PORNOGRAPHY/OBSCENE MAT     1
BRIBERY                     1
BAD CHECKS                   3
SEX OFFENSES, NON FORCIBLE    5
...
VEHICLE THEFT                1885
ASSAULT                      2518
NON-CRIMINAL                  3653
OTHER OFFENSES                  4004
LARCENY/THEFT                  8205
Name: count, Length: 36, dtype: int64
```

¿Qué tipo de delitos se cometió más?

LARCENY/THEFT con 8205 casos.

- c) Al jerarquizar las dos funciones en un comando, puede lograr el mismo resultado con una línea de código.

```
In [62]: # Code cell 15
print(SF['Category'].value_counts(ascending=True))
```

```
Category
GAMBLING                  1
PORNOGRAPHY/OBSCENE MAT     1
BRIBERY                     1
BAD CHECKS                   3
SEX OFFENSES, NON FORCIBLE    5
...
VEHICLE THEFT                1885
ASSAULT                      2518
NON-CRIMINAL                  3653
OTHER OFFENSES                  4004
LARCENY/THEFT                  8205
Name: count, Length: 36, dtype: int64
```

Pregunta de desafío: ¿Qué PdDistrict presentaba la mayoría de los incidentes de delitos informados? Proporcione los comandos de Python utilizados para apoyar su respuesta.

```
In [63]: conteo_distritos = SF['PdDistrict'].value_counts()
print(conteo_distritos)
distrito_mas_incidentes = conteo_distritos.idxmax()
cantidad_mas_incidentes = conteo_distritos.max()
print(f"El distrito con la mayoría de incidentes es {distrito_mas_incidentes} con {cantidad_mas_incidentes} incidentes.")

PdDistrict
SOUTHERN      6185
MISSION        4011
CENTRAL         3867
NORTHERN        3205
BAYVIEW          2970
INGLESIDE        2613
TENDERLOIN        2449
TARAVAL           2038
PARK              1800
RICHMOND          1622
Name: count, dtype: int64
El distrito con la mayoría de incidentes es SOUTHERN con 6185 incidentes.
```

El distrito con la mayoría de incidentes es SOUTHERN con 6185 incidentes.

Paso 2: Crear subconjuntos de datos y organizarlos en marcos de datos más pequeños.

- a) La indexación lógica se puede utilizar para seleccionar únicamente las filas en las cuales se cumple una condición específica. Por ejemplo, el código siguiente recupera sólo los delitos cometidos en agosto y guarda el resultado en un nuevo marco de datos.

```
In [64]: # Code cell 16
AugustCrimes = SF[SF['Month'] == 8]
AugustCrimes
```

	Category	Descript	DayOfWeek	Date	Time	PdDistrict	R
0	LARCENY/THEFT	GRAND THEFT FROM UNLOCKED AUTO	Sunday	08/31/2014	07:00:00 AM +0000	20:30	CENTRAL
1	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Sunday	08/31/2014	07:00:00 AM +0000	14:30	CENTRAL
2	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Sunday	08/31/2014	07:00:00 AM +0000	11:30	CENTRAL
3	DRUG/NARCOTIC	POSSESSION OF METH-AMPHETAMINE	Sunday	08/31/2014	07:00:00 AM +0000	17:49	MISSION
4	DRUG/NARCOTIC	POSSESSION OF COCAINE	Sunday	08/31/2014	07:00:00 AM +0000	18:05	NORTHERN
...
9715	NON-CRIMINAL	AIDED CASE, MENTAL DISTURBED	Friday	08/01/2014	07:00:00 AM +0000	19:55	MISSION
9716	OTHER OFFENSES	MISCELLANEOUS INVESTIGATION	Friday	08/01/2014	07:00:00 AM +0000	22:47	RICHMOND
9717	ASSAULT	THREATS AGAINST LIFE	Friday	08/01/2014	07:00:00 AM +0000	23:55	BAYVIEW
9718	DRIVING UNDER THE INFLUENCE	DRIVING WHILE UNDER THE INFLUENCE OF ALCOHOL	Friday	08/01/2014	07:00:00 AM +0000	23:38	NORTHERN
9719	SEX OFFENSES, FORCIBLE	ASSAULT TO RAPE WITH BODILY FORCE	Friday	08/01/2014	07:00:00 AM +0000	00:01	MISSION

9720 rows × 12 columns

In [68]:

```
agosto_df = SF[SF['Month'] == 8]
num_incidentes_agosto = agosto_df.shape[0]
print(f"En agosto hubo {num_incidentes_agosto} incidentes de delitos.")
```

En agosto hubo 9720 incidentes de delitos.

¿Cuántos incidentes de delitos hubo en agosto?

9720 incidentes

```
In [69]: robos_agosto = SF[(SF['Month'] == 8) & (SF['Category'] == 'LARCENY/THEFT')]
num_robos_agosto = robos_agosto.shape[0]
print(f"En agosto se informaron {num_robos_agosto} robos (LARCENY/THEFT).")
```

En agosto se informaron 2697 robos (LARCENY/THEFT).

¿Cuántos robos se informaron en agosto?

2697 robos

```
In [70]: # code cell 17
# Possible code for the question: How many burglaries were reported in the month
AugustCrimes = SF[SF['Month'] == 8]
AugustCrimesB = SF[SF['Category'] == 'BURGLARY']
len(AugustCrimesB)
```

Out[70]: 1257

b) Para crear un subconjunto del marco de datos de SF para un día específico, use la función 'query' para comparar el mes y el día al mismo tiempo.

```
In [71]: # Code cell 18
Crime0704 = SF.query('Month == 7 and Day == 4')
Crime0704
```

Out[71]:		Category	Descript	DayOfWeek	Date	Time	PdDistrict
	19087	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Friday	07/04/2014 07:00:00 AM +0000	22:30	SOUTHERN
	19088	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Friday	07/04/2014 07:00:00 AM +0000	18:15	SOUTHERN
	19089	BURGLARY	BURGLARY,RESIDENCE UNDER CONSTRT, FORCIBLE ENTRY	Friday	07/04/2014 07:00:00 AM +0000	00:50	TARAVAWA
	19090	NON-CRIMINAL	LOST PROPERTY	Friday	07/04/2014 07:00:00 AM +0000	19:00	PAR
	19091	ASSAULT	BATTERY	Friday	07/04/2014 07:00:00 AM +0000	21:00	NORTHERN

	19423	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Friday	07/04/2014 07:00:00 AM +0000	19:25	SOUTHERN
	19424	OTHER OFFENSES	LOST/STOLEN LICENSE PLATE	Friday	07/04/2014 07:00:00 AM +0000	11:00	INGLESIDE
	19425	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Friday	07/04/2014 07:00:00 AM +0000	20:30	SOUTHERN
	19426	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Friday	07/04/2014 07:00:00 AM +0000	08:00	SOUTHERN
	19427	LARCENY/THEFT	PETTY THEFT OF PROPERTY	Friday	07/04/2014 07:00:00 AM +0000	15:30	RICHMOND

341 rows × 12 columns



In [72]: # Code cell 19
SF.columns

Out[72]: Index(['Category', 'Descript', 'DayOfWeek', 'Date', 'Time', 'PdDistrict',
'Resolution', 'Address', 'X', 'Y', 'Month', 'Day'],
dtype='object')

Parte 5: Presentar los datos

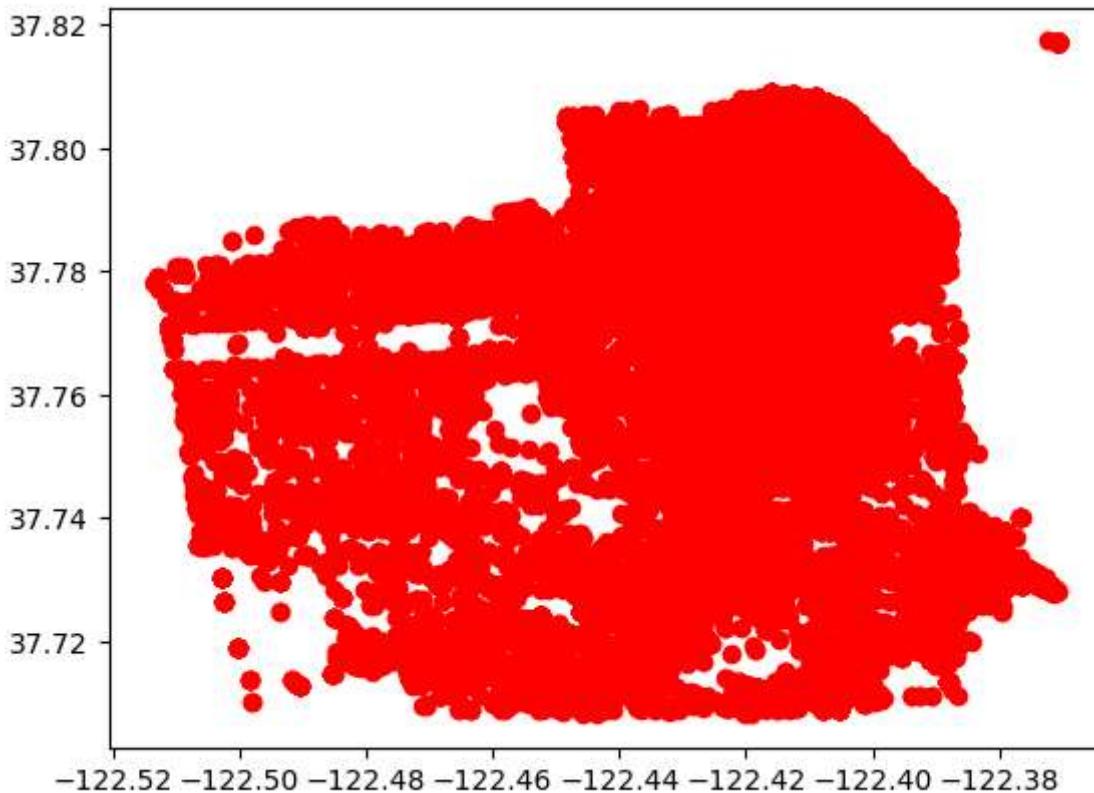
La visualización y presentación de datos proporciona una descripción general inmediata que puede no ser evidente simplemente observando los datos sin procesar. El marco de

datos de SF contiene las coordenadas de longitud y latitud que se pueden utilizar para graficar los datos.

Paso 1: Graficar el marco de datos de SF con las variables X e Y.

a) Utilice la función 'plot()' para graficar el marco de datos de SF. Utilice el parámetro opcional para crear el gráfico en rojo y configurar la forma del marcador en un círculo utilizando *ro*.

```
In [74]: # Code cell 20
plt.plot(SF['X'], SF['Y'], 'ro')
plt.show()
```



b) Identifique la cantidad de distritos de departamentos policiales y luego cree el diccionario *pd_districts* para asociar la secuencia a un valor entero.

```
In [75]: # Code cell 21
pd_districts = np.unique(SF['PdDistrict'])
pd_districts_levels = dict(zip(pd_districts, range(len(pd_districts))))
pd_districts_levels
```

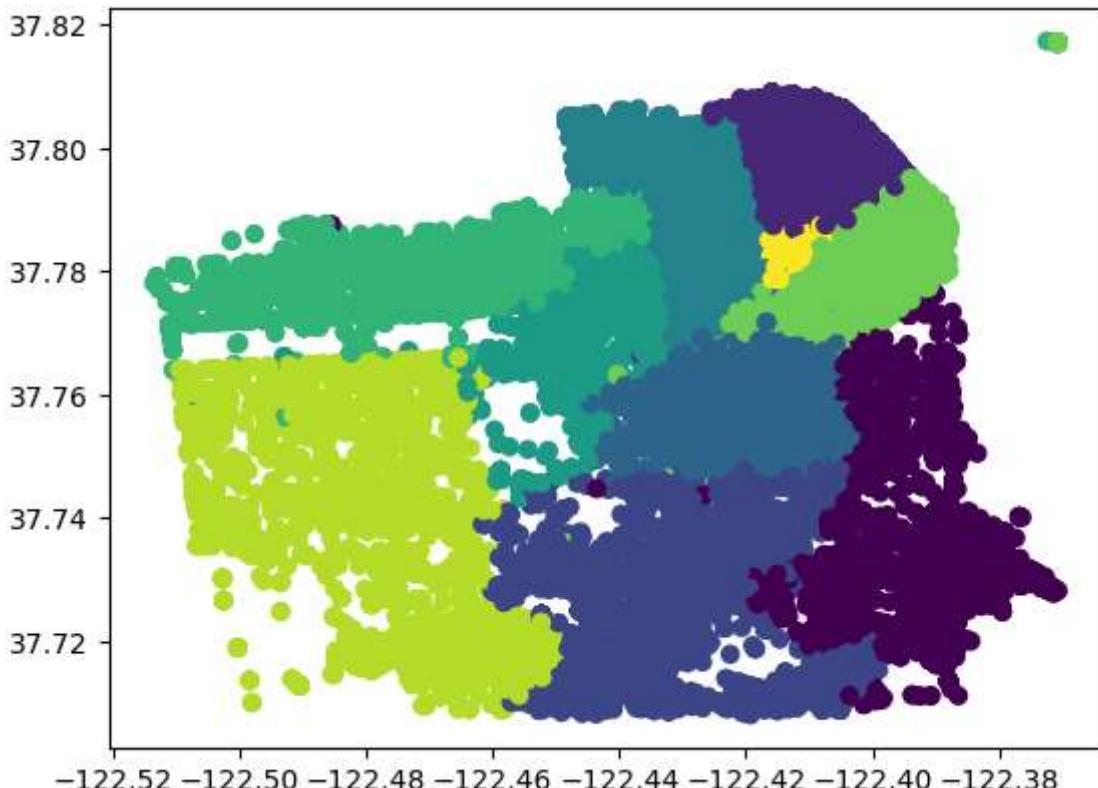
```
Out[75]: {'BAYVIEW': 0,
'CENTRAL': 1,
'INGLESIDE': 2,
'MISSION': 3,
'NORTHERN': 4,
'PARK': 5,
'RICHMOND': 6,
'SOUTHERN': 7,
'TARAVAL': 8,
'TENDERLOIN': 9}
```

c) Utilice 'apply' y 'lambda' para agregar la ID del valor entero del departamento policial a una nueva columna para el marco de datos.

```
In [81]: # Code cell 22
SF['PdDistrictCode'] = SF['PdDistrict'].apply(lambda row: pd_districts_levels[re
```

d) Utilice el *PdDistrictCode* creado recientemente para cambiar automáticamente el color.

```
In [84]: # Code cell 23
plt.scatter(SF['X'], SF['Y'], c=SF['PdDistrictCode'])
plt.show()
```



Paso 2: Agregar los paquetes de mapas para mejorar el gráfico.

En el paso 1, creó un gráfico simple que muestra dónde ocurrieron los incidentes de delitos en el condado de SF. Este gráfico es útil, pero 'folium' ofrece funciones adicionales que permiten superponer este gráfico en un mapa de OpenStreet.

a) 'Folium' requiere que se especifique el color del marcador mediante un valor hexadecimal. Por este motivo, utilizamos el paquete *colores* y seleccionamos los colores necesarios.

```
In [86]: # Code cell 24
from matplotlib import colors
districts = np.unique(SF['PdDistrict'])
color_list = list(colors.cnames.values())
selected_colors = color_list[:len(districts)]
pd_districts_colors = {district: color for district, color in zip(districts, sel
print(pd_districts_colors)
```

```
{'BAYVIEW': '#F0F8FF', 'CENTRAL': '#FAEBD7', 'INGLESIDE': '#00FFFF', 'MISSION': '#7FFFD4', 'NORTHERN': '#FFFFFF', 'PARK': '#F5F5DC', 'RICHMOND': '#FFE4C4', 'SOUTHERN': '#000000', 'TARAVAL': '#FFEBBC', 'TENDERLOIN': '#0000FF'}
```

b) Cree un diccionario de colores para cada distrito del departamento policial.

```
In [88]: # Code cell 25
districts = np.unique(SF['PdDistrict'])
color_values = list(colors.cnames.values())
n = len(districts)
indices = np.linspace(0, len(color_values) - 1, n, dtype=int)
selected_colors = [color_values[i] for i in indices]
color_dict = dict(zip(districts, selected_colors))
print(color_dict)
```

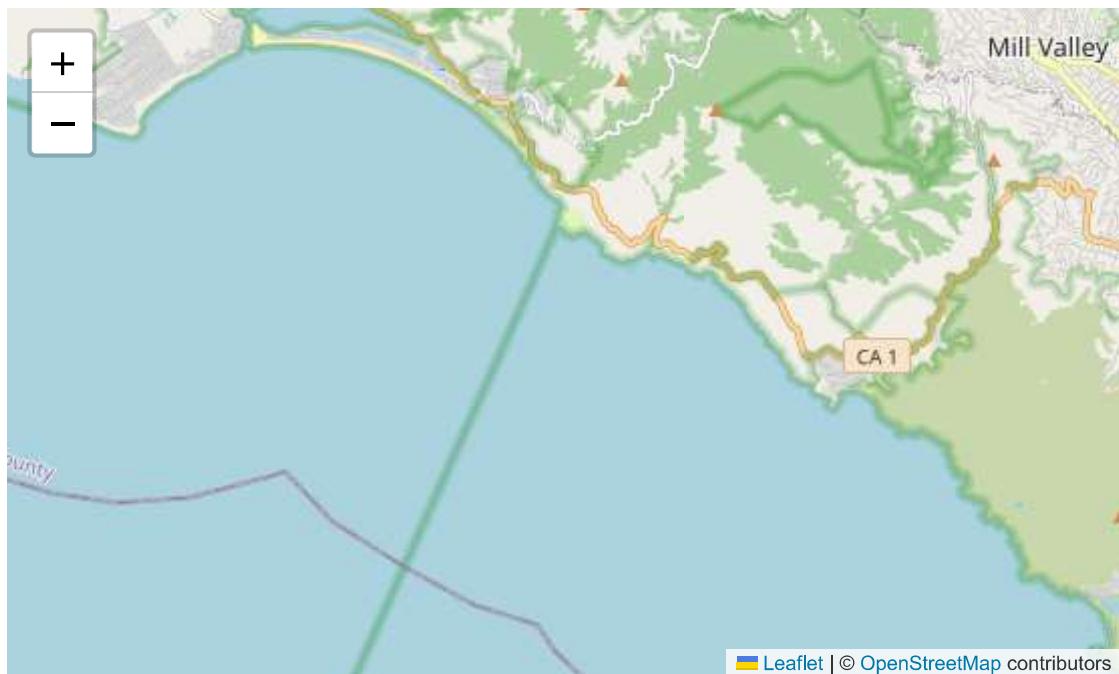
```
{'BAYVIEW': '#F0F8FF', 'CENTRAL': '#FF7F50', 'INGLESIDE': '#8B0000', 'MISSION': '#DCDCDC', 'NORTHERN': '#7CFC00', 'PARK': '#FFFFE0', 'RICHMOND': '#FFE4E1', 'SOUTHERN': '#CD853F', 'TARAVAL': '#87CEEB', 'TENDERLOIN': '#9ACD32'}
```

c) Cree el mapa con las coordenadas centrales de los datos de SF para centrar el mapa (utilice 'mean'). Para reducir el tiempo de cómputo, se utiliza *plotEvery* para limitar la cantidad de datos graficados. Establezca este valor a 1 para graficar todas las filas (puede llevar mucho tiempo visualizar el mapa).

```
In [89]: # Code cell 26
#Create map
map_osm = folium.Map(location=[SF['Y'].mean(), SF['X'].mean()], zoom_start=12)
plotEvery = 50
obs = list(zip(SF['Y'], SF['X'], SF['PdDistrict']))
for el in obs[0:-1:plotEvery]:
    folium.CircleMarker(
        location=[el[0], el[1]],
        color=color_dict[el[2]],
        fill=True,
        fill_color=color_dict[el[2]],
        radius=5,
        fill_opacity=0.7
    ).add_to(map_osm)
```

```
In [90]: # Code cell 27
map_osm
```

Out[90]:



© 2017 Cisco y/o sus filiales. Todos los derechos reservados. Este documento es información pública de Cisco.