• **PASO 9.** Identificando valores nulos:

• **PASO 11.** Pivotaer columnas:

```
df1.pivot_table(values='Calificacion',index=['Nombre','Calificacion'],columns
='Nombre')
```

	Nombre
Nombre	Calificacion
Jaime	11
Jose	10
Juan	10
	11
Pedro	12

3.2 ANÁLISIS DEL BITCOIN

Para la implementación de nuestro primer análisis vamos a utilizar la siguiente fuente de datos:

https://www.kaggle.com/sudalairajkumar/cryptocurrencypricehistory?select=coin_Bitcoin.csv

Esta fuente de datos posee 29992 registros y la fecha, precio más alto, precio más bajo, precio de apertura, precio de cierre, volumen y mercado.

Análisis BTC.ipynb

Markdown

```
# ANALISIS DE DATOS–BITCOIN
![Image of
Yaktocat](http://c.files.bbci.co.uk/14656/production/_107524538_gettyima-
ges-924986096.jpg)
```

Para la implementación de nuestro análisis vamos a utilizar la siguiente fuente de datos:

https://www.datosabiertos.gob.pe/dataset/dataset-de-pruebas-moleculares-del-instituto-nacional-de-salud-para-covid-19-ins/resource-5

Esta fuente de datos posee 29992 registros y la fecha, precio más alto, precio más bajo, precio de apertura, precio de cierre, volumen y mercado.

Análisis CORONAVIRUS1.ipynb

• **PASO 1**. Importamos las librerías a usar:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
```

• **PASO 2**. Lectura de datos en memoria RAM

```
df1 = pd.read_csv('pm28Noviembre2021.csv', sep='|')
df1
```

df1

6339492	20211128	14688944.0	20211128.0	34.0	MASCULINO	PRIVADO	140114.0	LIMA	
6339493	20211128	30807784.0	20211128.0	23.0	MASCULINO	GOBIERNO REGIONAL	60107.0	CAJAMARCA	CAJAM.
6339494	20211128	23167028.0	20211128.0	31.0	MASCULINO	GOBIERNO REGIONAL	60101.0	CAJAMARCA	CAJAM.
6339495	20211128	34844655.0	20211128.0	68.0	MASCULINO	GOBIERNO REGIONAL	60105.0	CAJAMARCA	CAJAM.
6339496	(6339496 row(s) affected)	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

6339497 rows × 15 columns

• **PASO 3**. Analizar los datos:

El método info retorna básicamente el número de registros, los nombres de las columnas y los tipos de columnas:

```
df1.info()
```

```
df1.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6339497 entries, 0 to 6339496
Data columns (total 15 columns):
 #   Column                 Dtype
---  ------                 -----
 0   FECHA_CORTE            object
 1   UUID                   float64
 2   FECHA_MUESTRA          float64
 3   Edad                   float64
 4   Sexo                   object
 5   Institucion            object
 6   UBIGEO_PACIENTE        float64
 7   DEPARTAMENTO_PACIENTE  object
 8   PROVINCIA_PACIENTE     object
 9   DISTRITO_PACIENTE      object
 10  DEPARTAMENTO_MUESTRA   object
 11  PROVINCIA_MUESTRA      object
 12  DISTRITO_MUESTRA       object
 13  TIPO_MUESTRA           object
 14  RESULTADO              object
dtypes: float64(4), object(11)
memory usage: 725.5+ MB
```

El método de describe muestra información estadística básica sobre los valores numéricos del conjunto de datos como media, desviación estándar, percentil, mínimo y máximo.

```
df1.describe()
```

	UUID	FECHA_MUESTRA	Edad	UBIGEO_PACIENTE
count	6.332756e+06	6.339496e+06	6.339496e+06	6.339484e+06
mean	1.692884e+07	2.020789e+07	3.990018e+01	1.258833e+05
std	1.085747e+07	4.406280e+03	1.525426e+01	5.217007e+04
min	1.000000e+00	1.959101e+07	0.000000e+00	1.000000e+04
25%	7.349311e+06	2.020121e+07	2.900000e+01	1.101080e+05
50%	1.585661e+07	2.021033e+07	3.800000e+01	1.401100e+05
75%	2.499034e+07	2.021073e+07	4.900000e+01	1.401370e+05
max	3.833296e+07	2.021113e+07	1.049000e+03	8.500000e+05

El método isnull().sum() se usa básicamente para verificar si hay valores nulos en el conjunto de datos. Esto enumerará el número de valores nulos en cada columna.

```
df1.isnull().sum()
```

Análisis CORONAVIRUS2.ipynb

- **PASO 1.** Importamos las librerías a usar:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
import time
import seaborn as sns
from shapely.geometry import Point
import geopandas as gpd
from geopandas import GeoDataFrame
```

- **PASO 2.** Lectura de datos en memoria RAM

Casos confirmados

```
confirmados_df = pd.read_csv('time_series_covid19_confirmed_global.csv')
confirmados _df
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	12/5/21	12/6/21	12/7/21	12/8/21	12/9/
0	NaN	Afghanistan	33.939110	67.709953	0	0	0	0	0	0	...	157445	157499	157508	157542	1575
1	NaN	Albania	41.153300	20.168300	0	0	0	0	0	0	...	201730	201902	202295	202641	2028
2	NaN	Algeria	28.033900	1.659600	0	0	0	0	0	0	...	211469	211662	211859	212047	2122
3	NaN	Andorra	42.506300	1.521800	0	0	0	0	0	0	...	18010	18031	18815	18815	192
4	NaN	Angola	-11.202700	17.873900	0	0	0	0	0	0	...	65259	65259	65301	65332	653
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
275	NaN	Vietnam	14.058324	108.277199	0	2	2	2	2	2	...	1309092	1323683	1337523	1352122	13674
276	NaN	West Bank and Gaza	31.952200	35.233200	0	0	0	0	0	0	...	461467	462219	462821	462958	4632
277	NaN	Yemen	15.552727	48.516388	0	0	0	0	0	0	...	10025	10034	10043	10047	100
278	NaN	Zambia	-13.133897	27.849332	0	0	0	0	0	0	...	210312	210327	210374	210436	2105
279	NaN	Zimbabwe	-19.015438	29.154857	0	0	0	0	0	0	...	139046	139046	141601	150628	1558

280 rows × 697 columns

Casos fallecidos

```
fallecidos_df = pd.read_csv('time_series_covid19_deaths_global.csv')
fallecidos_df
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	12/5/21	12/6/21	12/7/21	12/8/21	12/9/21	1
0	NaN	Afghanistan	33.939110	67.709953	0	0	0	0	0	0	...	7312	7316	7317	7317	7321	
1	NaN	Albania	41.153300	20.168300	0	0	0	0	0	0	...	3110	3115	3122	3126	3128	
2	NaN	Algeria	28.033900	1.659600	0	0	0	0	0	0	...	6103	6111	6114	6122	6126	
3	NaN	Andorra	42.506300	1.521800	0	0	0	0	0	0	...	132	133	133	133	133	
4	NaN	Angola	-11.202700	17.873900	0	0	0	0	0	0	...	1735	1735	1735	1735	1736	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
275	NaN	Vietnam	14.058324	108.277199	0	0	0	0	0	0	...	26260	26483	26700	26930	27186	
276	NaN	West Bank and Gaza	31.952200	35.233200	0	0	0	0	0	0	...	4810	4817	4822	4823	4826	
277	NaN	Yemen	15.552727	48.516388	0	0	0	0	0	0	...	1954	1955	1955	1956	1957	
278	NaN	Zambia	-13.133897	27.849332	0	0	0	0	0	0	...	3667	3668	3668	3668	3668	
279	NaN	Zimbabwe	-19.015438	29.154857	0	0	0	0	0	0	...	4710	4710	4713	4720	4723	

280 rows × 697 columns

Casos recuperados

```
recuperados_df = pd.read_csv('time_series_covid19_recovered_global.csv')
recuperados_df
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	12/5/21	12/6/21	12/7/21	12/8/21	12/9/21	12
0	NaN	Afghanistan	33.939110	67.709953	0	0	0	0	0	0	...	0	0	0	0	0	
1	NaN	Albania	41.153300	20.168300	0	0	0	0	0	0	...	0	0	0	0	0	
2	NaN	Algeria	28.033900	1.659600	0	0	0	0	0	0	...	0	0	0	0	0	
3	NaN	Andorra	42.506300	1.521800	0	0	0	0	0	0	...	0	0	0	0	0	
4	NaN	Angola	-11.202700	17.873900	0	0	0	0	0	0	...	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
260	NaN	Vietnam	14.058324	108.277199	0	0	0	0	0	0	...	0	0	0	0	0	
261	NaN	West Bank and Gaza	31.952200	35.233200	0	0	0	0	0	0	...	0	0	0	0	0	
262	NaN	Yemen	15.552727	48.516388	0	0	0	0	0	0	...	0	0	0	0	0	
263	NaN	Zambia	-13.133897	27.849332	0	0	0	0	0	0	...	0	0	0	0	0	
264	NaN	Zimbabwe	-19.015438	29.154857	0	0	0	0	0	0	...	0	0	0	0	0	

265 rows × 697 columns

Casos diarios

```
diarios_df = pd.read_csv('diarios.csv')
diarios_df
```

	FIPS	Admin2	Province_State	Country_Region	Last_Update	Lat	Long_	Confirmed	Deaths	Recovered	Active	Combined_Key	Incidence_R
0	NaN	NaN	NaN	Afghanistan	2020-08-17 04:27:20	33.93911	67.709953	37682	1379	27166	9137	Afghanistan	96.798
1	NaN	NaN	NaN	Albania	2020-08-17 04:27:20	41.15330	20.168300	7380	228	3794	3358	Albania	256.445
2	NaN	NaN	NaN	Algeria	2020-08-17 04:27:20	28.03390	1.659600	38583	1370	27017	10196	Algeria	87.986
3	NaN	NaN	NaN	Andorra	2020-08-17 04:27:20	42.50630	1.521800	989	53	863	73	Andorra	1280.010
4	NaN	NaN	NaN	Angola	2020-08-17 04:27:20	-11.20270	17.873900	1906	88	628	1190	Angola	5.790
...	...	...	...	...	...	...	...	...	...	...	...	...	
3966	NaN	NaN	W.P. Kuala Lumpur	Malaysia	2020-08-17 04:27:20	3.13900	101.686900	2538	18	2461	59	W.P. Kuala Lumpur, Malaysia	142.712
3967	NaN	NaN	W.P. Labuan	Malaysia	2020-08-17 04:27:20	5.28310	115.230800	22	0	19	3	W.P. Labuan, Malaysia	22.132
3968	NaN	NaN	W.P. Putrajaya	Malaysia	2020-08-17 04:27:20	2.92640	101.696400	99	1	97	1	W.P. Putrajaya, Malaysia	93.927
3969	NaN	NaN	Unknown	Malaysia	2020-08-17 04:27:20	NaN	NaN	0	0	0	0	Unknown, Malaysia	N
3970	NaN	NaN	NaN	Tonga	2020-08-17 04:27:20	-21.17900	-175.198200	0	0	0	0	Tonga	0.000

3971 rows × 14 columns

• **PASO 3.** Analizar los datos:

El método info retorna básicamente el número de registros, los nombres de las columnas y los tipos de columnas:

El método de describe muestra información estadística básica sobre los valores numéricos del conjunto de datos como media, desviación estándar, percentil, mínimo y máximo.

```
confirmados_df.describe()
```

```
confirmados_df.describe()
```

	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	...	12/5/21	1
count	278.000000	278.000000	280.000000	280.000000	280.000000	280.000000	280.000000	280.000000	280.000000	280.000000	...	2.800000e+02	2.8000
mean	20.156042	21.788955	1.989286	2.339286	3.360714	5.121429	7.564286	10.453571	19.921429	22.025000	...	9.495658e+05	9.5168
std	25.283318	76.200169	26.590143	26.687678	33.225879	46.244243	64.627991	87.077220	213.666694	214.980193	...	4.030404e+06	4.0396
min	-51.796300	-178.116500	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000e+00	0.0000
25%	4.643279	-37.713675	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	3.087250e+03	3.0945
50%	21.517170	20.921188	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	4.414450e+04	4.4485
75%	40.393350	84.992575	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	4.398472e+05	4.4014
max	71.706900	178.065000	444.000000	444.000000	549.000000	761.000000	1058.000000	1423.000000	3554.000000	3554.000000	...	4.909994e+07	4.9282

8 rows × 695 columns

```
fallecidos_df.describe()
```

```
fallecidos_df.describe()
```

	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	...	12/5/21	12/6
count	278.000000	278.000000	280.000000	280.000000	280.000000	280.000000	280.000000	280.000000	280.000000	280.000000	...	280.000000	280.000
mean	20.156042	21.788955	0.060714	0.064286	0.092857	0.150000	0.200000	0.292857	0.467857	0.475000	...	18771.492857	18798.571
std	25.283318	76.200169	1.015944	1.017487	1.436326	2.391517	3.109011	4.542900	7.470302	7.470809	...	73905.890118	73986.728
min	-51.796300	-178.116500	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000
25%	4.643279	-37.713675	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	22.000000	22.000
50%	21.517170	20.921188	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	696.000000	701.000
75%	40.393350	84.992575	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	6963.250000	6975.500
max	71.706900	178.065000	17.000000	17.000000	24.000000	40.000000	52.000000	76.000000	125.000000	125.000000	...	786525.000000	786907.000

8 rows × 695 columns

```
recuperados_df.describe()
```

```
recuperados_df.describe()
```

	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	...	12/5/21	12/6/21	12/7/21	1
count	264.000000	264.000000	265.000000	265.000000	265.000000	265.000000	265.000000	265.000000	265.000000	265.000000	...	265.0	265.0	265.0	
mean	18.731013	27.087363	0.113208	0.120755	0.147170	0.158491	0.211321	0.245283	0.407547	0.479245	...	0.0	0.0	0.0	
std	24.724885	73.922864	1.723944	1.727620	1.916076	1.982254	2.611285	2.807631	4.948211	5.442585	...	0.0	0.0	0.0	
min	-51.796300	-178.116500	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	
25%	4.454096	-10.007650	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	
50%	19.254600	24.242250	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	
75%	38.889675	91.814200	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	
max	71.706900	178.065000	28.000000	28.000000	31.000000	32.000000	42.000000	45.000000	80.000000	88.000000	...	0.0	0.0	0.0	

8 rows × 695 columns

```
diarios_df.describe()
```

	FIPS	Lat	Long_	Confirmed	Deaths	Recovered	Active	Incidence_Rate	Case-Fatality_Ratio
count	3252.000000	3889.000000	3889.000000	3971.000000	3971.000000	3.971000e+03	3971.000000	3888.000000	3907.000000
mean	32387.881919	35.806103	-71.852138	5469.196424	205.321581	3.443999e+03	2372.532108	1079.279344	2.392146
std	17990.852405	13.148529	54.356885	28213.033839	1403.902967	3.595536e+04	10993.145033	1120.919757	3.311112
min	66.000000	-52.368000	-175.198200	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.000000
25%	19048.500000	33.202226	-96.577496	83.000000	1.000000	0.000000e+00	68.000000	359.416872	0.366749
50%	30086.000000	37.877361	-86.845176	337.000000	5.000000	0.000000e+00	272.000000	724.504460	1.470588
75%	47041.500000	42.134404	-77.492245	1493.500000	34.000000	0.000000e+00	1056.500000	1430.311825	3.125000
max	99999.000000	71.706900	178.065000	699493.000000	42072.000000	1.833067e+06	233128.000000	14046.437434	64.922481

El método isnull().sum() se usa básicamente para verificar si hay valores nulos en el conjunto de datos. Esto enumerará el número de valores nulos en cada columna.

```
df1.isnull().sum()
```

```
confirmados_df.isnull().sum()

Province/State    193
Country/Region      0
Lat                 2
Long                2
1/22/20             0
                 ...
12/10/21            0
12/11/21            0
12/12/21            0
12/13/21            0
12/14/21            0
Length: 697, dtype: int64
```

```
fallecidos_df.isnull().sum()

Province/State    193
Country/Region      0
Lat                 2
Long                2
1/22/20             0
                 ...
12/10/21            0
12/11/21            0
12/12/21            0
12/13/21            0
12/14/21            0
Length: 697, dtype: int64
```

```
recuperados_df.isnull().sum()

Province/State    194
Country/Region      0
Lat                 1
Long                1
1/22/20             0
                 ...
12/10/21            0
12/11/21            0
12/12/21            0
12/13/21            0
12/14/21            0
Length: 697, dtype: int64
```

Observamos los casos diarios, para ello generamos un Mapa con GeoPandas:

```
geometry = [Point(xy) for xy in zip(confirmados_df['Long'], confirmados_df['Lat'])]
gdf = GeoDataFrame(confirmados_df[['Lat','Long']], geometry=geometry)
mundo = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
gdf.plot(ax=mundo.plot(figsize=(15, 10)), marker='o', color='red', markersize=15);
```

A continuación, detallamos el código paso a paso:

Comprima las coordenadas en un objeto puntual y conviértalo en un GeoDataFrame:

```
geometry = [Point(xy) for xy in zip(confirmados_df[‘Long’], confirmados_df[‘Lat’])]
gdf = GeoDataFrame(confirmados_df[[‘Lat’,’Long’]], geometry=geometry)
```

```
geometry = [Point(xy) for xy in zip(confirmados_df['Long'], confirmados_df['Lat'])]
gdf = GeoDataFrame(confirmados_df[['Lat','Long']], geometry=geometry)
gdf
```

	Lat	Long	geometry
0	33.939110	67.709953	POINT (67.70995 33.93911)
1	41.153300	20.168300	POINT (20.16830 41.15330)
2	28.033900	1.659600	POINT (1.65960 28.03390)
3	42.506300	1.521800	POINT (1.52180 42.50630)
4	-11.202700	17.873900	POINT (17.87390 -11.20270)
...	...	...	...
275	14.058324	108.277199	POINT (108.27720 14.05832)
276	31.952200	35.233200	POINT (35.23320 31.95220)
277	15.552727	48.516388	POINT (48.51639 15.55273)
278	-13.133897	27.849332	POINT (27.84933 -13.13390)
279	-19.015438	29.154857	POINT (29.15486 -19.01544)

280 rows × 3 columns

Ahora trazamos las coordenadas sobre un mapa a nivel de país.

```
mundo = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
gdf.plot(ax=mundo.plot(figsize=(15, 10)), marker='o', color='red', markersi-
ze=15);
```

Observamos los 10 países principales:

```
confirmados_df['Country/Region'].value_counts()[:10]
```

```
confirmados_df['Country/Region'].value_counts()[:10]
```

```
China             34
Canada            16
United Kingdom    12
France            12
Australia          8
Netherlands        5
Denmark            3
New Zealand        2
Panama             1
Niger              1
Name: Country/Region, dtype: int64
```

Observamos los casos diarios por país:

```
diarios_df['Country_Region'].value_counts()
diarios_df
```

```
diarios_df['Country_Region'].value_counts()
diarios_df
```

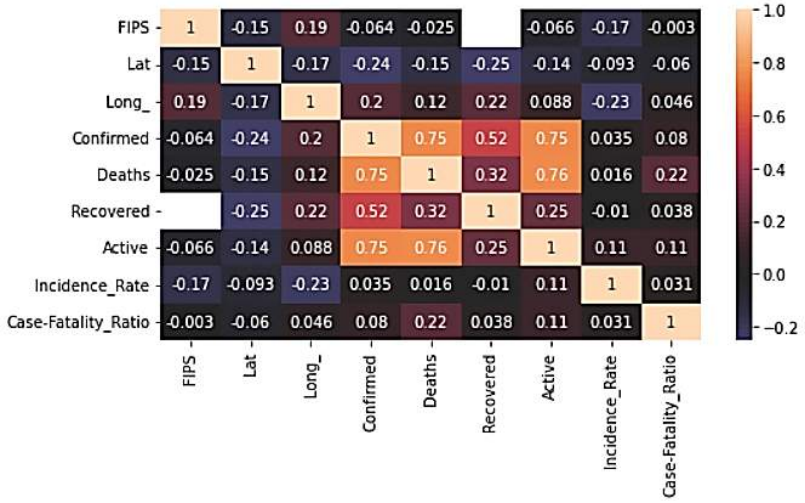
	FIPS	Admin2	Province_State	Country_Region	Last_Update	Lat	Long_	Confirmed	Deaths	Recovered	Active	Combined_Key	Incidence
0	NaN	NaN	NaN	Afghanistan	2020-08-17 04:27:20	33.93911	67.709953	37682	1379	27166	9137	Afghanistan	96.7
1	NaN	NaN	NaN	Albania	2020-08-17 04:27:20	41.15330	20.168300	7380	228	3794	3358	Albania	256.4
2	NaN	NaN	NaN	Algeria	2020-08-17 04:27:20	28.03390	1.659600	38583	1370	27017	10196	Algeria	87.9
3	NaN	NaN	NaN	Andorra	2020-08-17 04:27:20	42.50630	1.521800	989	53	863	73	Andorra	1280.0
4	NaN	NaN	NaN	Angola	2020-08-17 04:27:20	-11.20270	17.873900	1806	88	628	1190	Angola	5.7
...	...	...	...	...	...	...	...	...	...	...	...	...	...
3966	NaN	NaN	W.P. Kuala Lumpur	Malaysia	2020-08-17 04:27:20	3.13900	101.686900	2538	18	2461	59	W.P. Kuala Lumpur,	142.7

Mapa de calor:

El mapa de calor y el análisis de correlación son métodos de análisis comunes. La intensidad del calor se usa a menudo para expresar la densidad de distribución en un mapa, y también puede entenderse simplemente como un mapeo de valores a colores en coordenadas bidimensionales.

Ahora dibujamos un mapa de calor para determinar correlación de variables:

```
plt.figure(figsize=(8,4))
sns.heatmap(diarios_df.corr(), annot=True,center=0)
print("")
plt.show()
```



ⓘ NOTA

Confirmados, defunciones y activos tienen una correlación positiva alta, mientras que el resto tiene correlación negativa.

Esta fuente de datos posee 10,000 registros y proporciona el sexo, la altura y el peso de la persona. Tenemos que construir y entrenar un modelo en este conjunto de datos para que pueda predecir el peso de una persona dado su altura.

Para nuestro primer ejemplo vamos a utilizar la fuente de datos:

Ejemplo1.ipynb

• **PASO 1.** Importamos las librerías a usar:

```
import pandas as pd
import numpy as np
import matplotlib as plt
from sklearn.model_selection import train_test_split
```

• **PASO 2.** Lectura de datos en memoria RAM

```
df1 = pd.read_csv('weight-height.csv')
```

```
df1 = pd.read_csv('weight-height.csv')
df1
```

	Gender	Height	Weight
0	Male	73.847017	241.893563
1	Male	68.781904	162.310473
2	Male	74.110105	212.740856
3	Male	71.730978	220.042470
4	Male	69.881796	206.349801
...	...	...	...
9995	Female	66.172652	136.777454
9996	Female	67.067155	170.867906
9997	Female	63.867992	128.475319
9998	Female	69.034243	163.852461
9999	Female	61.944246	113.649103

10000 rows × 3 columns

• **PASO 3.** Analizar los datos

El método info retorna básicamente el número de registros, los nombres de las columnas y los tipos de columnas:

```
df1.info()
```

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Gender  10000 non-null  object
 1   Height  10000 non-null  float64
 2   Weight  10000 non-null  float64
dtypes: float64(2), object(1)
memory usage: 234.5+ KB
```

El método de describe muestra información estadística básica sobre los valores numéricos del conjunto de datos como media, desviación estándar, percentil, mínimo y máximo.

```
df1.describe()
```

```
df1.describe()
```

	Height	Weight
count	10000.000000	10000.000000
mean	66.367560	161.440357
std	3.847528	32.108439
min	54.263133	64.700127
25%	63.505620	135.818051
50%	66.318070	161.212928
75%	69.174262	187.169525
max	78.998742	269.989699

El método isnull().sum() se usa básicamente para verificar si hay valores nulos en el conjunto de datos. Esto enumerará el número de valores nulos en cada columna.

```
df1.isnull().sum()
```

```
df1.isnull().sum()
```

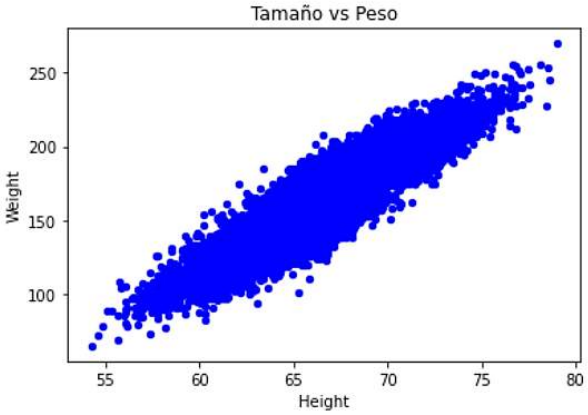
```
Gender    0
Height    0
Weight    0
dtype: int64
```

- **PASO 4.** Graficamos la nube de puntos:

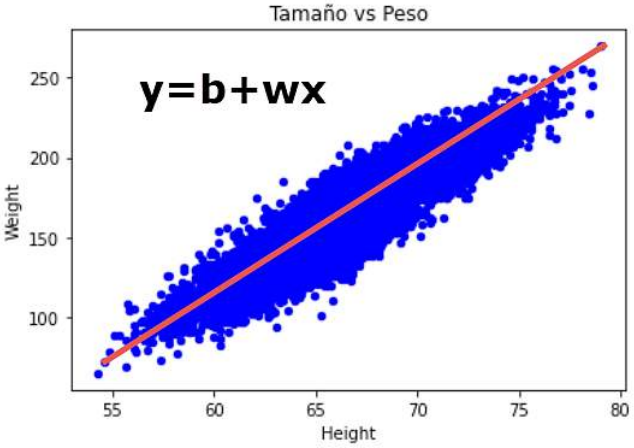
```
df1.plot(kind='scatter',x='Height',y='Weight',title='Tamaño vs
Peso',color='blue')
```

```
df1.plot(kind='scatter',x='Height',y='Weight',title='Tamaño vs Peso',color='blue')
```

```
<AxesSubplot:title={'center':'Tamaño vs Peso'}, xlabel='Height', ylabel='Weight'>
```



- **PASO 5.** Definimos la ecuación lineal que define la recta que pueda representar mejor la realidad:




```
def linea(x,w,b):
y=b+w*x
return y
```

"PRIMEROS PARAMETROS"

- **PASO 6.** Colocamos los valores de la altura dentro de una array y definimos un valor fijo de w y b:

```
altura=df1['Height'].values
w=1.5
b=0
```

- **PASO 7.** Probamos nuestra primera hipótesis:

```
altura
```

```
array([73.84701702, 68.78190405, 74.11010539, ..., 63.86799221,
       69.03424313, 61.94424588])
```

```
y
```

```
array([110.77052553, 103.17285607, 111.16515809, ...,  95.80198832,
       103.5513647 ,  92.91636882])
```

Por ejemplo, para un valor de altura de 73.84701702 pulgadas mide 110.77052553 libras.

```
df1
```

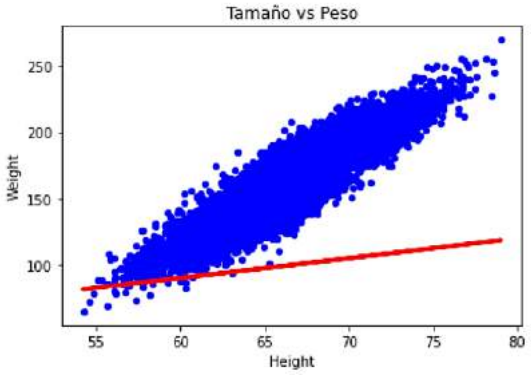
	Gender	Height	Weight
0	Male	73.847017	241.893563
1	Male	68.781904	162.310473
2	Male	74.110105	212.740856
3	Male	71.730978	220.042470
4	Male	69.881796	206.349801
...	...	...	...
9995	Female	66.172652	136.777454
9996	Female	67.067155	170.867906
9997	Female	63.867992	128.475319
9998	Female	69.034243	163.852461

- **PASO 8.** Ahora comprobaremos si la nueva recta representa la realidad:

```
import matplotlib.pyplot as plt
df1.plot(kind='scatter',x='Height',y='Weight',title='Tamaño vs
Peso',color='blue')
plt.plot(altura,y,color='red',linewidth=3)
```

```
import matplotlib.pyplot as plt
df1.plot(kind='scatter',x='Height',y='Weight',title='Tamaño vs Peso',color='blue')
plt.plot(altura,y,color='red',linewidth=3)
```

```
[<matplotlib.lines.Line2D at 0x1b614ef4fa0>]
```



"Calibramos otra vez el modelo, segunda prueba"

- **PASO 9.** Colocamos los valores de la altura dentro de una array y definimos un valor fijo de w y b:

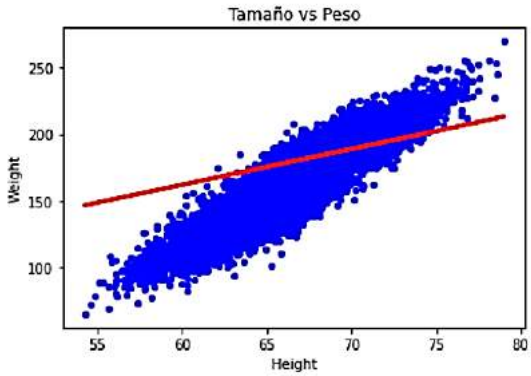
```
altura=df1['Height'].values
w=2.7
b=0
```

- **PASO 10.** Ahora comprobaremos si la nueva recta representa la realidad:

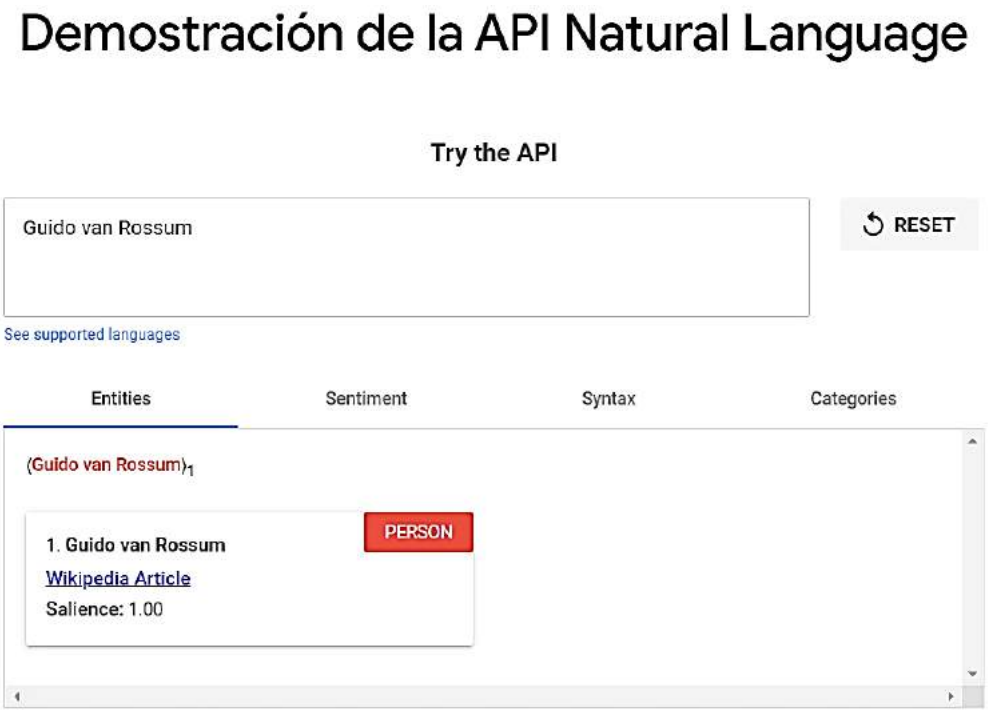
```
import matplotlib.pyplot as plt
df1.plot(kind='scatter',x='Height',y='Weight',title='Tamaño vs
Peso',color='blue')
plt.plot(altura,y,color='red',linewidth=3)
```

```
import matplotlib.pyplot as plt
df1.plot(kind='scatter',x='Height',y='Weight',title='Tamaño vs Peso',color='blue')
plt.plot(altura,y,color='red',linewidth=3)
```

```
[<matplotlib.lines.Line2D at 0x1b614d35d60>]
```



Ahora vamos a API de Google para comprobar si reconoce personas:



5.1.6 Instalando TextBlob y NLTK en anaconda

TextBlob, que es otra biblioteca para el Procesamiento del Lenguaje Natural muy eficiente para Python. TextBlob se basa en NLTK y proporciona una interfaz fácil de usar para la biblioteca NLTK. Veremos cómo se puede usar TextBlob para realizar una variedad de tareas del Procesamiento del Lenguaje Natural que van desde el etiquetado de partes del habla hasta el análisis de sentimientos, y la traducción de idiomas a la clasificación de textos.

Para la instalación de TextBlob en anaconda ir a la siguiente url: *https://anaconda.org/conda-forge/textblob*

```
conda install -c conda-forge textblob
conda install -c conda-forge/label/gcc7 textblob
conda install -c conda-forge/label/cf201901 textblob
conda install -c conda-forge/label/cf202003 textblob
```

Natural Language Toolkit–NLTK, es un conjunto de bibliotecas para el procesamiento del lenguaje natural simbólico y estadístico. Contiene bibliotecas de procesamiento de texto para tokenización, análisis, clasificación, derivación, etiquetado y razonamiento semántico.

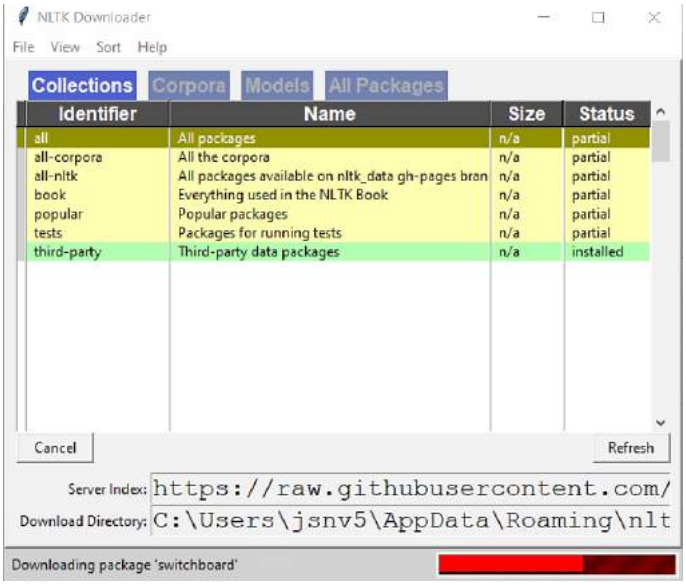
Para la instalación de NLTK en anaconda ir a la siguiente url:

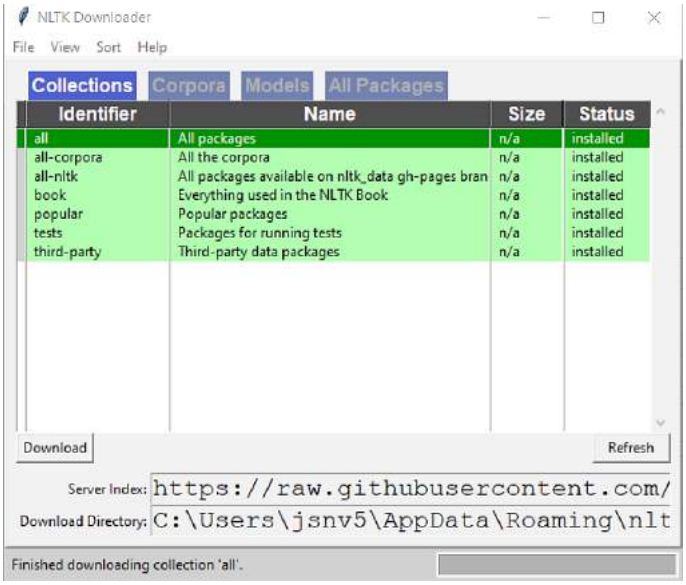
https://anaconda.org/anaconda/nltk

```
conda install -c anaconda nltk
```

Una vez que instalada la librería NLTK, debes instalar los paquetes NLTK ejecutando el siguiente código:

```
import nltk
nltk.download()
```





A continuación:

Análisis de Texto1.ipynb

- **PASO 1.** Importamos las librerías

```
import nltk
import string
nltk.download("punkt")
nltk.download("stopwords")
from nltk.tokenize import sent_tokenize,word_tokenize
from nltk.probability import FreqDist
from nltk.corpus import stopwords
```

```
import nltk
import string
nltk.download("punkt")
nltk.download("stopwords")
from nltk.tokenize import sent_tokenize,word_tokenize
from nltk.probability import FreqDist
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\jsnv5\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\jsnv5\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

- **PASO 2.** Lectura de la fuente de datos:

```
cadena=open("documento1.txt","r").read()
```

```
cadena=open("documento1.txt","r").read()
```

- **PASO 3.** Partir el Texto en Frases:

```
frases = sent_tokenize(cadena)
print(frases)
```

```
# partir el texto en frases
frases = sent_tokenize(cadena)
print(frases)
```

```
['Python es un lenguaje de programacian de proposito general creado por Guido Van Rosum en los 90 trabajo en Google y en la act
ualidad en Dropbox, su nombre proviene del comic Monty Python.', 'Cuenta con una sintaxis muy limpia y legible.', 'Posee tipado
dinamico esto quiere decir que una variable puede poseer datos de varios tipos, junto con su naturaleza interpretada, hacen de
un lenguaje para ser el primer en aprender.', 'Python es un lenguaje interpretado, lo que nos indica que no se necesita compila
r el codigo fuente para poder ejecutarlo, lo que ofrece ventajas.', 'Python esta escrito en el lenguaje C, por lo que se puede
extender a traves de su api en C o C++ y escribir nuevos tipos de datos, funciones, etc.', 'En la actualidad hay dos vertientes
la version 2.x y 3.x, al final llegara el momento que se integraran estas dos versiones, es recomendable utilizar la ultima ver
sion estable 3.x Algunas de las caractera\xadsticas mas importantes es que Python es multiparadigma: Programacion estructurada,
Programacion Orientada a Objetos y Programacion Funcional.']
```

- **PASO 4**. Partir el texto en palabras:

```
palabras = word_tokenize(cadena)
print(palabras)
```

```
# partir el texto en palabras
palabras = word_tokenize(cadena)
print(palabras)
```

```
['Python', 'es', 'un', 'lenguaje', 'de', 'programacian', 'de', 'proposito', 'general', 'creado', 'por', 'Guido', 'Van', 'Rosu
m', 'en', 'los', '90', 'trabajo', 'en', 'Google', 'y', 'en', 'la', 'actualidad', 'en', 'Dropbox', ',', 'su', 'nombre', 'provien
e', 'del', 'comic', 'Monty', 'Python', '.', 'Cuenta', 'con', 'una', 'sintaxis', 'muy', 'limpia', 'y', 'legible', '.', 'Posee',
'tipado', 'dinamico', 'esto', 'quiere', 'decir', 'que', 'una', 'variable', 'puede', 'poseer', 'datos', 'de', 'varios', 'tipos',
',', 'junto', 'con', 'su', 'naturaleza', 'interpretada', ',', 'hacen', 'de', 'un', 'lenguaje', 'para', 'ser', 'el', 'primer',
'en', 'aprender', '.', 'Python', 'es', 'un', 'lenguaje', 'interpretado', ',', 'lo', 'que', 'nos', 'indica', 'que', 'no', 'se',
'necesita', 'compilar', 'el', 'codigo', 'fuente', 'para', 'poder', 'ejecutarlo', ',', 'lo', 'que', 'ofrece', 'ventajas', '.',
'Python', 'esta', 'escrito', 'en', 'el', 'lenguaje', 'C', ',', 'por', 'lo', 'que', 'se', 'puede', 'extender', 'a', 'traves', 'd
e', 'su', 'api', 'en', 'C', 'o', 'C++', 'y', 'escribir', 'nuevos', 'tipos', 'de', 'datos', ',', 'funciones', ',', 'etc', '.',
'En', 'la', 'actualidad', 'hay', 'dos', 'vertientes', 'la', 'version', '2.x', 'y', '3.x', ',', 'al', 'final', 'llegara', 'el',
'momento', 'que', 'se', 'integraran', 'estas', 'dos', 'versiones', ',', 'es', 'recomendable', 'utilizar', 'la', 'ultima', 'vers
ion', 'estable', '3.x', 'Algunas', 'de', 'las', 'caractera\xadsticas', 'mas', 'importantes', 'es', 'que', 'Python', 'es', 'mult
iparadigma', ':', 'Programacion', 'estructurada', ',', 'Programacion', 'Orientada', 'a', 'Objetos', 'y', 'Programacion', 'Funci
onal', '.']
```

- **PASO 5**. Analizamos los caracteres más comunes:

```
fdist = FreqDist(palabras)
print(fdist.most_common(20))
```

```
# Ahora analizamos los caracteres mas comunes
fdist = FreqDist(palabras)
print(fdist.most_common(20))
```

```
[(',', 11), ('de', 7), ('en', 7), ('que', 7), ('.', 6), ('Python', 5), ('es', 5), ('y', 5), ('lenguaje', 4), ('la', 4), ('el',
4), ('un', 3), ('su', 3), ('lo', 3), ('se', 3), ('Programacion', 3), ('por', 2), ('actualidad', 2), ('con', 2), ('una', 2)]
```

```
for c in fdist.most_common(20):
    print(c)
```

```
(',', 11)
('de', 7)
('en', 7)
('que', 7)
('.', 6)
('Python', 5)
('es', 5)
('y', 5)
('lenguaje', 4)
('la', 4)
('el', 4)
('un', 3)
('su', 3)
('lo', 3)
('se', 3)
('Programacion', 3)
('por', 2)
('actualidad', 2)
('con', 2)
('una', 2)
```

- **PASO 6.** Identificamos los stopwords, que son las palabras conectoras que repetimos con frecuencia en un idioma, palabras más básicas e informativas:

```
# identificamos los stopwords
stopword=stopwords.words("spanish")
print(stopword)
```

```
# identificamos los stopwords
stopword=stopwords.words("spanish")
print(stopword)

['de', 'la', 'que', 'el', 'en', 'y', 'a', 'los', 'del', 'se', 'las', 'por', 'un', 'para', 'con', 'no', 'una', 'su', 'al', 'lo',
'como', 'más', 'pero', 'sus', 'le', 'ya', 'o', 'este', 'sí', 'porque', 'esta', 'entre', 'cuando', 'muy', 'sin', 'sobre', 'tambi
én', 'me', 'hasta', 'hay', 'donde', 'quien', 'desde', 'todo', 'nos', 'durante', 'todos', 'uno', 'les', 'ni', 'contra', 'otros',
'ese', 'eso', 'ante', 'ellos', 'e', 'esto', 'mi', 'antes', 'algunos', 'qué', 'unos', 'yo', 'otro', 'otras', 'otra', 'él', 'tant
o', 'esa', 'estos', 'mucho', 'quienes', 'nada', 'muchos', 'cual', 'poco', 'ella', 'estar', 'estas', 'algunas', 'algo', 'nosotro
s', 'mi', 'mis', 'tú', 'te', 'ti', 'tu', 'tus', 'ellas', 'nosotras', 'vosotros', 'vosotras', 'os', 'mío', 'mía', 'míos', 'mía
s', 'tuyo', 'tuya', 'tuyos', 'tuyas', 'suyo', 'suya', 'suyos', 'suyas', 'nuestro', 'nuestra', 'nuestros', 'nuestras', 'vuestr
o', 'vuestra', 'vuestros', 'vuestras', 'esos', 'esas', 'estoy', 'estás', 'está', 'estamos', 'estáis', 'están', 'esté', 'estés',
'estemos', 'estéis', 'estén', 'estaré', 'estarás', 'estará', 'estaremos', 'estaréis', 'estarán', 'estaría', 'estarías', 'estari
amos', 'estaríais', 'estarían', 'estaba', 'estabas', 'estábamos', 'estabais', 'estaban', 'estuve', 'estuviste', 'estuvo', 'estu
vimos', 'estuvisteis', 'estuvieron', 'estuviera', 'estuvieras', 'estuviéramos', 'estuvierais', 'estuvieran', 'estuviese', 'estu
vieses', 'estuviésemos', 'estuvieseis', 'estuviesen', 'estando', 'estado', 'estada', 'estados', 'estadas', 'estad', 'he', 'ha
s', 'ha', 'hemos', 'habéis', 'han', 'haya', 'hayas', 'hayamos', 'hayáis', 'hayan', 'habré', 'habrás', 'habrá', 'habremos', 'hab
réis', 'habrán', 'habría', 'habrías', 'habríamos', 'habríais', 'habrían', 'había', 'habías', 'habíamos', 'habíais', 'habían',
'hube', 'hubiste', 'hubo', 'hubimos', 'hubisteis', 'hubieron', 'hubiera', 'hubieras', 'hubiéramos', 'hubierais', 'hubieran', 'h
ubiese', 'hubieses', 'hubiésemos', 'hubieseis', 'hubiesen', 'habiendo', 'habido', 'habida', 'habidos', 'habidas', 'soy', 'ere
s', 'es', 'somos', 'sois', 'son', 'sea', 'seas', 'seamos', 'seáis', 'sean', 'seré', 'serás', 'será', 'seremos', 'seréis', 'será
n', 'sería', 'serías', 'seríamos', 'seríais', 'serían', 'era', 'eras', 'éramos', 'erais', 'eran', 'fui', 'fuiste', 'fue', 'fuim
os', 'fuisteis', 'fueron', 'fuera', 'fueras', 'fuéramos', 'fuerais', 'fueran', 'fuese', 'fueses', 'fuésemos', 'fueseis', 'fuese
n', 'sintiendo', 'sentido', 'sentida', 'sentidos', 'sentidas', 'siente', 'sentid', 'tengo', 'tienes', 'tiene', 'tenemos', 'tené
is', 'tienen', 'tenga', 'tengas', 'tengamos', 'tengáis', 'tengan', 'tendré', 'tendrás', 'tendrá', 'tendremos', 'tendréis', 'ten
drán', 'tendría', 'tendrías', 'tendríamos', 'tendríais', 'tendrían', 'tenía', 'tenías', 'teníamos', 'teníais', 'tenían', 'tuv
e', 'tuviste', 'tuvo', 'tuvimos', 'tuvisteis', 'tuvieron', 'tuviera', 'tuvieras', 'tuviéramos', 'tuvierais', 'tuvieran', 'tuvie
se', 'tuvieses', 'tuviésemos', 'tuvieseis', 'tuviesen', 'teniendo', 'tenido', 'tenida', 'tenidos', 'tenidas', 'tened']
```

- **PASO 7.** Extraemos los stopwords de nuestra lista de palabras:

```
palabras2 = [p for p in palabras if p not in stopword]
print(palabras2)
```

```
palabras2 = [p for p in palabras if p not in stopword]
print(palabras2)

['Python', 'lenguaje', 'programacion', 'proposito', 'general', 'creado', 'Guido', 'Van', 'Rosum', '90', 'trabajo', 'Google', 'a
ctualidad', 'Dropbox', ',', 'nombre', 'proviene', 'comic', 'Monty', 'Python', '.', 'Cuenta', 'sintaxis', 'limpia', 'legible',
'.', 'Posee', 'tipado', 'dinamico', 'quiere', 'decir', 'variable', 'puede', 'poseer', 'datos', 'varios', 'tipos', ',', 'junto',
'naturaleza', 'interpretada', ',', 'hacen', 'lenguaje', 'ser', 'primer', 'aprender', '.', 'Python', 'lenguaje', 'interpretado',
',', 'indica', 'necesita', 'compilar', 'codigo', 'fuente', 'poder', 'ejecutarlo', ',', 'ofrece', 'ventajas', '.', 'Python', 'es
crito', 'lenguaje', 'C', ',', 'puede', 'extender', 'traves', 'api', 'C', 'C++', 'escribir', 'nuevos', 'tipos', 'datos', ',', 'f
unciones', ',', 'etc', '.', 'En', 'actualidad', 'dos', 'vertientes', 'version', '2.x', '3.x', ',', 'final', 'llegara', 'moment
o', 'integraran', 'dos', 'versiones', ',', 'recomendable', 'utilizar', 'ultima', 'version', 'estable', '3.x', 'Algunas', 'carac
tera\xadsticas', 'mas', 'importantes', 'Python', 'multiparadigma', ':', 'Programacion', 'estructurada', ',', 'Programacion', 'O
rientada', 'Objetos', 'Programacion', 'Funcional', '.']
```

- **PASO 8.** Al observar vemos que no eliminamos las comas y algunas palabras que se encuentran en mayúscula:

```
['Python', 'lenguaje', 'programacion', 'proposito', 'general', 'creado', 'Guido', 'Van', 'Rosum', '90', 'trabajo', 'Google', 'a
ctualidad', 'Dropbox', ',', 'nombre', 'proviene', 'comic', 'Monty', 'Python', '.', 'Cuenta', 'sintaxis', 'limpia', 'legible',
'.', 'Posee', 'tipado', 'dinamico', 'quiere', 'decir', 'variable', 'puede', 'poseer', 'datos', 'varios', 'tipos', ',', 'junto',
'naturaleza', 'interpretada', ',', 'hacen', 'lenguaje', 'ser', 'primer', 'aprender', '.', 'Python', 'lenguaje', 'interpretado',
',', 'indica', 'necesita', 'compilar', 'codigo', 'fuente', 'poder', 'ejecutarlo', ',', 'ofrece', 'ventajas', '.', 'Python', 'es
crito', 'lenguaje', 'C', ',', 'puede', 'extender', 'traves', 'api', 'C', 'C++', 'escribir', 'nuevos', 'tipos', 'datos', ',', 'f
unciones', ',', 'etc', '.', 'En', 'actualidad', 'dos', 'vertientes', 'version', '2.x', '3.x', ',', 'final', 'llegara', 'moment
o', 'integraran', 'dos', 'versiones', ',', 'recomendable', 'utilizar', 'ultima', 'version', 'estable', '3.x', 'Algunas', 'carac
tera\xadsticas', 'mas', 'importantes', 'Python', 'multiparadigma', ':', 'Programacion', 'estructurada', ',', 'Programacion', 'O
rientada', 'Objetos', 'Programacion', 'Funcional', '.']
```

- **PASO 9.** Ajustamos el tipo de análisis, excluyendo los signos de puntuación ya con un significado concreto:

```
# partir el texto en palabras
palabras = word_tokenize(cadena.lower())
palabras2 = [p for p in palabras if p not in stopword + list(string.punctuation)]
print(palabras2)
```

```
# partir el texto en palabras -excluyendo los signos de puntuación
palabras = word_tokenize(cadena.lower())
palabras2 = [p for p in palabras if p not in stopword + list(string.punctuation)]
print(palabras2)
```

```
['python', 'lenguaje', 'programacion', 'proposito', 'general', 'creado', 'guido', 'van', 'rosum', '90', 'trabajo', 'google', 'a
ctualidad', 'dropbox', 'nombre', 'proviene', 'comic', 'monty', 'python', 'cuenta', 'sintaxis', 'limpia', 'legible', 'posee', 't
ipado', 'dinamico', 'quiere', 'decir', 'variable', 'puede', 'poseer', 'datos', 'varios', 'tipos', 'junto', 'naturaleza', 'inter
pretada', 'hacen', 'lenguaje', 'ser', 'primer', 'aprender', 'python', 'lenguaje', 'interpretado', 'indica', 'necesita', 'compil
ar', 'codigo', 'fuente', 'poder', 'ejecutarlo', 'ofrece', 'ventajas', 'python', 'escrito', 'lenguaje', 'c', 'puede', 'extende
r', 'traves', 'api', 'c', 'c++', 'escribir', 'nuevos', 'tipos', 'datos', 'funciones', 'etc', 'actualidad', 'dos', 'vertientes',
'version', '2.x', '3.x', 'final', 'llegara', 'momento', 'integraran', 'dos', 'versiones', 'recomendable', 'utilizar', 'ultima',
'version', 'estable', '3.x', 'caractera\xadsticas', 'mas', 'importantes', 'python', 'multiparadigma', 'programacion', 'estructu
rada', 'programacion', 'orientada', 'objetos', 'programacion', 'funcional']
```

- **PASO 10.** Volvemos a analizamos los caracteres más comunes:

```
fdist = FreqDist(palabras)
print(fdist.most_common(20))
```

```
fdist = FreqDist(palabras2)
print(fdist.most_common(20))
```

```
[('python', 5), ('lenguaje', 4), ('programacion', 4), ('actualidad', 2), ('puede', 2), ('datos', 2), ('tipos', 2), ('c', 2),
('dos', 2), ('version', 2), ('3.x', 2), ('proposito', 1), ('general', 1), ('creado', 1), ('guido', 1), ('van', 1), ('rosum',
1), ('90', 1), ('trabajo', 1), ('google', 1)]
```

- **PASO 11.** Una vez eliminado el ruido, ahora reduzcamos las palabras a su raíz, despojando las palabras de sus sufijos:

```
from nltk.stem import SnowballStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
stemmer=SnowballStemmer("spanish")
raices =[]
for palabra2 in palabras2:
  raices.append(stemmer.stem(palabra2))
raices
```

```
from nltk.stem import SnowballStemmer
from nltk.tokenize import sent_tokenize , word_tokenize
stemmer=SnowballStemmer("spanish")
raices =[]
for palabra2 in palabras2:
    raices.append(stemmer.stem(palabra2))
raices
```

```
['python',
 'lenguaj',
 'programacion',
 'proposit',
 'general',
 'cre',
 'guid',
 'van',
 'rosum',
 '90',
 'trabaj',
 'googl',
 'actual',
 'dropbox',
 'nombr',
 'provien',
 'comic',
 'monty',
 'python',
```

Esto nos permite que cuando realicemos recuento de frecuencia, nos afecte la conjugación o variante de la palabra.

5.1.13 Mi segundo ejemplo de traducción de texto

Googletrans

Googletrans es una biblioteca de Python gratuita e ilimitada que implementó la API de Google Translate. Esto utiliza la API Ajax de Google Translate para realizar llamadas a métodos como detectar y traducir.

A continuación, describimos algunas de sus características:

- Rápido y confiable: utiliza los mismos servidores que utiliza translate.google.com.
- Detección automática de idioma.
- Traducciones masivas.
- URL de servicio personalizable.
- Agrupación de conexiones (la ventaja de usar solicitudes. Sesión).
- Soporte HTTP / 2.

Instalación anaconda:

```
conda install -c conda-forge googletrans
conda install -c conda-forge/label/cf201901 googletrans
conda install -c conda-forge/label/cf202003 googletrans
```

Traducción .ipynb

- **PASO 1.** Importamos las librerías

```
from googletrans import Translator
```

- **PASO 2.** Creamos un objeto translator

```
translator=Translator()
```

- **PASO 3.** Detectamos el idioma

```
print(translator.detect('Idioma Español'))
```

- **PASO 1.** Traducimos al inglés

```
translation = translator.translate("Curso de análisis de Datos", dest='en')
print(translation.text)
```

A continuación, el código completo:

```
from googletrans import Translator
```

```
translator=Translator()
```

```
#detectando el idioma
print(translator.detect('Idioma Español'))
```

```
Detected(lang=es, confidence=None)
Detected(lang=es, confidence=None)
```

```
#Traduciendo
translation = translator.translate("Curso de Analisis de Datos", dest='en')
print(translation.text)
```

```
Data analysis course
Data analysis course
```

A continuación, traducimos el archivo: documento1.txt al alemán:

```
file = open('documento1.txt', 'r')
contenido = file.read()
ftranslator=Translator()
result = translator.translate(contenido, dest='de')
print(result.text)
```

```
file = open('documento1.txt', 'r')
contenido = file.read()
ftranslator=Translator()
result = translator.translate(contenido, dest='de')
print(result.text)
```

```
Python ist eine allgemeine Prognose-Programmiersprache, die von Guido van Rosum in den 90 Jobs auf Google erstellt wurde und de
rzeit in Dropbox, der Name kommt von Comic Monty Python. Es hat eine sehr saubere und lesbare Syntax. Es hat dynamisch eingetip
pt, das bedeutet, dass eine Variable Daten verschiedener Typen mit seiner interpretierten Natur besitzen kann, eine Sprache, um
die erste zu lernen. Python ist eine interpretierte Sprache, die uns sagt, dass Sie den Quellcode nicht kompilieren müssen, um
es ausführen zu können, die Vorteile bietet.

Python wird in der C-Sprache geschrieben, sodass es in C oder C ++ über Ihre API erweitert werden kann und neue Arten von Date
n, Funktionen usw. schreibt. Derzeit gibt es zwei Aspekte Version 2.x und 3.x, am Ende wird die Zeit integriert, es ist ratsam,
die neueste stabile Version 3.x einigen der wichtigsten Funktionen zu verwenden ist, dass Python Multiparadigma ist: Programmie
rt strukturiert , Programmierung auf Objekte und funktionale Programmierung ausgerichtet.
Python ist eine allgemeine Prognose-Programmiersprache, die von Guido van Rosum in den 90 Jobs auf Google erstellt wurde und de
rzeit in Dropbox, der Name kommt von Comic Monty Python. Es hat eine sehr saubere und lesbare Syntax. Es hat dynamisch eingetip
pt, das bedeutet, dass eine Variable Daten verschiedener Typen mit seiner interpretierten Natur besitzen kann, eine Sprache, um
die erste zu lernen. Python ist eine interpretierte Sprache, die uns sagt, dass Sie den Quellcode nicht kompilieren müssen, um
es ausführen zu können, die Vorteile bietet.

Python wird in der C-Sprache geschrieben, sodass es in C oder C ++ über Ihre API erweitert werden kann und neue Arten von Date
n, Funktionen usw. schreibt. Derzeit gibt es zwei Aspekte Version 2.x und 3.x, am Ende wird die Zeit integriert, es ist ratsam,
die neueste stabile Version 3.x einigen der wichtigsten Funktionen zu verwenden ist, dass Python Multiparadigma ist: Programmie
rt strukturiert , Programmierung auf Objekte und funktionale Programmierung ausgerichtet.
```

5.1.14 Mi tercer segundo ejemplo análisis de sentimiento

Análisis de Sentimiento.ipynb

- **PASO 1.** Importamos las librerías

```
from sklearn.feature_extraction.text import CountVectorizer
from nltk.tokenize import RegexpTokenizer
import pandas as pd
```

- **PASO 2.** Lista de etiquetas de los comentarios si es una opinión negativa o positiva

```
List1=[“Bueno”,”Malo”,”Malo”,”Malo”,”Bueno”,”Bueno”,”Malo”,”Bueno”,”Bueno”,
”Malo”]
```

- **PASO 3.** Lista que define las palabras utilizadas en las opiniones

```
List2=[“Lo recomendaria”,
 “Es un Mal Producto”,
 “No lo compraria”,
 “No lo Recomendaria a nadie”,
 “Satisface mi necesidades”,
 “Voy a comentar a mis amigos de este buen producto”,
 “Muy Baja calidad”,
 “Muy Buen Precio”,
 “Si lo hubiera adquirido antes”,
 “Ojala salga del Mercado”
 ]
```

- **PASO 4.** Convertimos las dos listas en un dataframe

```
df = pd.DataFrame({“Sentimiento”:List1,”Valoracion”:List2})
```

- **PASO 5.** Observamos el contenido del dataframe

```
df
```

	Sentimiento	Valoracion
0	Bueno	Lo recomendaria
1	Malo	Es un Mal Producto
2	Malo	No lo compraria
3	Malo	No lo Recomendaria a nadie
4	Bueno	Satisface mi necesidades
5	Bueno	Voy a comentar a mis amigos de este buen producto
6	Malo	Muy Baja calidad
7	Bueno	Muy Buen Precio
8	Bueno	Si lo hubiera adquirido antes
9	Malo	Ojala salga del Mercado

- **PASO 6.** Tokenizer para quitar todo aquello que no son letras y numeros

```
token = RegexpTokenizer(r'[a-zA-Z0-9]+')
```

- **PASO 7.** Ahora utilizaremos CountVectorizer para convertir una colección de documentos de texto en una matriz de recuentos de tokens

```
cv =
CountVectorizer(lowercase=True,ngram_range=(1,2),tokenizer=token.tokenize)
```

  - **lowercase=True.** Pone en minúsculas su texto
  - **ngram_range=(1,2).** Unidades de significado o Palabras, en nuestro caso parejas de dos palabras
  - **tokenizer=token.tokenize.** Utilizar la expresión regular para quitar todo aquello que no son letras y numeros

- **PASO 8.** Ahora le diremos que entrene y transforme a la columna de valoración

```
text_counts = cv.fit_transform(df['Valoracion'])
```

- **PASO 9.** Muestra la cuanta vez aparece una pareja de palabras como nuestra muestra es pequeña solo aparecerá una vez

```
print(text_counts)
```

```
print(text_counts)
```

```
  (0, 28)       1
  (0, 50)       1
  (0, 31)       1
  (1, 22)       1
  (1, 58)       1
  (1, 32)       1
  (1, 49)       1
  (1, 23)       1
  (1, 59)       1
  (1, 33)       1
  (2, 28)       1
  (2, 44)       1
  (2, 17)       1
  (2, 45)       1
  (2, 29)       1
  (3, 28)       1
  (3, 50)       1
  (3, 31)       1
  (3, 44)       1
```

- **PASO 10.** Ahora vamos a crear una muestra de entrenamiento y prueba, utilizando el paquete sklearn utilizaremos como variables regresoras el recuento de palabras: text_counts, y utilizaremos como respuesta el sentimiento: List1, el entrenamiento con un 50% y la prueba con el otro 50% y fijamos al final con una semilla de 0

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(text_
counts,df['Sentimiento'],test_size=0.5,random_state=0)
```

- **PASO 11.** Recomiendo un modelo multinomial

```
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
```

ⓘ NOTA

El modelo de clasificador multinomial Naive Bayes(MultinomialNB) es adecuado para la clasificación con características discretas (por ejemplo, recuento de palabras para la clasificación de texto). La distribución multinomial normalmente requiere recuentos de características enteras. Sin embargo, en la práctica, los recuentos fraccionarios como tf-idf también pueden funcionar.

- **PASO 12.** Ajustamos el modelo usando fit, vamos a utilizar train y vamos a generar las predicciones con el test y para evaluar el porcentaje de acierto de nuestro modelo vamos a utilizar metrics.accuracy_score

```
c = MultinomialNB().fit(X_train,y_train)
prediccion= c.predict(X_test)
print("Precision del Modelo multinomiaINB:",metrics.accuracy_score(y_
test,prediccion))
```

```
c = MultinomialNB().fit(X_train,y_train)
prediccion= c.predict(X_test)
print("Precision del Modelo multinomiaINB:",metrics.accuracy_score(y_test,prediccion))
```

```
Precision del Modelo multinomiaINB: 0.6
```

- **PASO 13.** Ahora mostramos las predicciones:

```
print(y_test)
```

```
print(y_test)
```

```
2     Malo
8    Bueno
4    Bueno
9     Malo
1     Malo
Name: Sentimiento, dtype: object
```

5.1.15 Stanza

Stanza es una colección de herramientas precisas y eficientes para el análisis lingüístico de muchos lenguajes humanos desarrollado por la universidad de stanford. Desde el texto en bruto hasta el análisis sintáctico y el reconocimiento de entidades, Stanza lleva modelos de PNL de última generación a los idiomas que elijas.

Stanza1 .ipynb

- **PASO 1.** Instalamos Stanza

```
!pip install stanza
```

```
#Instalar stanza
!pip install stanza
```

```
Requirement already satisfied: stanza in c:\users\jsnv5\anaconda3\lib\site-packages (1.3.0)
Requirement already satisfied: requests in c:\users\jsnv5\anaconda3\lib\site-packages (from stanza) (2.26.0)
Requirement already satisfied: six in c:\users\jsnv5\anaconda3\lib\site-packages (from stanza) (1.16.0)
Requirement already satisfied: protobuf in c:\users\jsnv5\anaconda3\lib\site-packages (from stanza) (3.19.1)
Requirement already satisfied: tqdm in c:\users\jsnv5\anaconda3\lib\site-packages (from stanza) (4.62.3)
Requirement already satisfied: numpy in c:\users\jsnv5\anaconda3\lib\site-packages (from stanza) (1.20.3)
Requirement already satisfied: torch>=1.3.0 in c:\users\jsnv5\anaconda3\lib\site-packages (from stanza) (1.10.0)
Requirement already satisfied: emoji in c:\users\jsnv5\anaconda3\lib\site-packages (from stanza) (1.6.1)
Requirement already satisfied: typing-extensions in c:\users\jsnv5\anaconda3\lib\site-packages (from torch>=1.3.0->stanza) (3.1
0.0.2)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\jsnv5\anaconda3\lib\site-packages (from requests->stanza)
(2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\jsnv5\anaconda3\lib\site-packages (from requests->stanza) (2021.1
0.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\jsnv5\anaconda3\lib\site-packages (from requests->stanza) (1.2
6.7)
Requirement already satisfied: idna<4,>=2.5 in c:\users\jsnv5\anaconda3\lib\site-packages (from requests->stanza) (3.2)
Requirement already satisfied: colorama in c:\users\jsnv5\anaconda3\lib\site-packages (from tqdm->stanza) (0.4.4)
```

- **PASO 2.** Importamos las librerías

  `import stanza`

- **PASO 3.** Trabajamos con el paquete en español (descargamos)

  `stanza.download('es')`

```
import stanza
stanza.download('es')
```

```
Downloading https://raw.githubusercontent.com/stanfordnlp/stanza-resources/main/resources_1.3.0.json: 142k/? [00:00<00:00, 5.58MB/s]
2021-11-24 20:40:26 INFO: Downloading default packages for language: es (Spanish)...
2021-11-24 20:40:28 INFO: File exists: C:\Users\jsnv5\stanza_resources\es\default.zip.
2021-11-24 20:40:32 INFO: Finished downloading models and saved to C:\Users\jsnv5\stanza_resources.
```

- **PASO 4.** Stanza trabaja con Pipeline, trabajar diferentes tareas de lenguaje de procesamiento natural ahora especificamos los argumentos:
  - El lenguaje con el cual se va a trabajar: es (español).
  - El processors. Puede especificar los procesadores para descargar o cargar, enumerando los nombres de los procesadores en una cadena separada por comas. En nuestro ejemplo solo descargamos y cargamos los procesadores predeterminados tokenize (TokenizeProcessor) para que realice un tokenizado y pos (POSProcessor) para asignar etiquetas para el idioma español.

```
nlp=stanza.Pipeline('es',processors='tokenize,pos')
```

```
nlp=stanza.Pipeline('es',processors='tokenize,pos')
```

```
2021-11-24 20:50:48 WARNING: Language es package default expects mwt, which has been adde
2021-11-24 20:50:48 INFO: Loading these models for language: es (Spanish):
=======================
| Processor | Package |
-----------------------
| tokenize  | ancora  |
| mwt       | ancora  |
| pos       | ancora  |
=======================

2021-11-24 20:50:48 INFO: Use device: cpu
2021-11-24 20:50:48 INFO: Loading: tokenize
2021-11-24 20:50:49 INFO: Loading: mwt
2021-11-24 20:50:49 INFO: Loading: pos
2021-11-24 20:50:49 INFO: Done loading processors!
```

ⓘ NOTA

AnCora es un corpus del catalán (AnCora-CA) y del español (AnCora-ES)

- **PASO 5.** Ahora creado un string con el texto que deseo analizar, pasándolo por una instancia nlp:

```
documento=nlp("python es el mejor lenguaje de programacion")
```

- **PASO 6.** A continuación, mostramos los tokens (texto), lema y sus etiquetas Pos de cada palabra en cada oración de un documento anotado:

```
for sentence in documento.sentences:
 for word in sentence.words:
  print(word.text, word.lemma, word.pos)
```

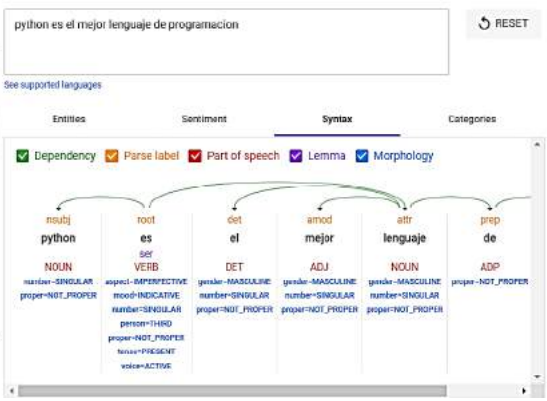
```
for sentence in documento.sentences:
    for word in sentence.words:
        print(word.text, word.lemma, word.pos)
```

```
python None NOUN
es None AUX
el None DET
mejor None ADJ
lenguaje None NOUN
de None ADP
programacion None NOUN
```

• **PASO 7.** Ahora lo comparamos con el api de Google en el análisis sintáctico:

```
for sentence in documento.sentences:
    for word in sentence.words:
        print(word.text, word.lemma, word.pos)
```

```
python None NOUN
es None AUX
el None DET
mejor None ADJ
lenguaje None NOUN
de None ADP
programacion None NOUN
```



5.1.16 Modelos ocultos de Markov para el etiquetado de texto

El etiquetado de texto es un problema cuyos requisitos se adaptan a la forma natural de los modelos de markov. Por su descripción resulta evidente que los eventos observables van a ser las palabras y los eventos ocultos, su categoría semántica. A continuación, mostramos un ejemplo de un modelo Markoviano latente:

`Modelo Markoviano Latente (HMM)1.ipynb`

• **PASO 1.** Deberá descargar el dataset específico del siguiente repositorio y colocarlo en carpeta de trabajo:

https://github.com/UniversalDependencies/UD_Spanish-AnCora

ⓘ NOTA

Corpus en español llamado ancora.

• **PASO 2.** Ahora deberá instalar conllu:

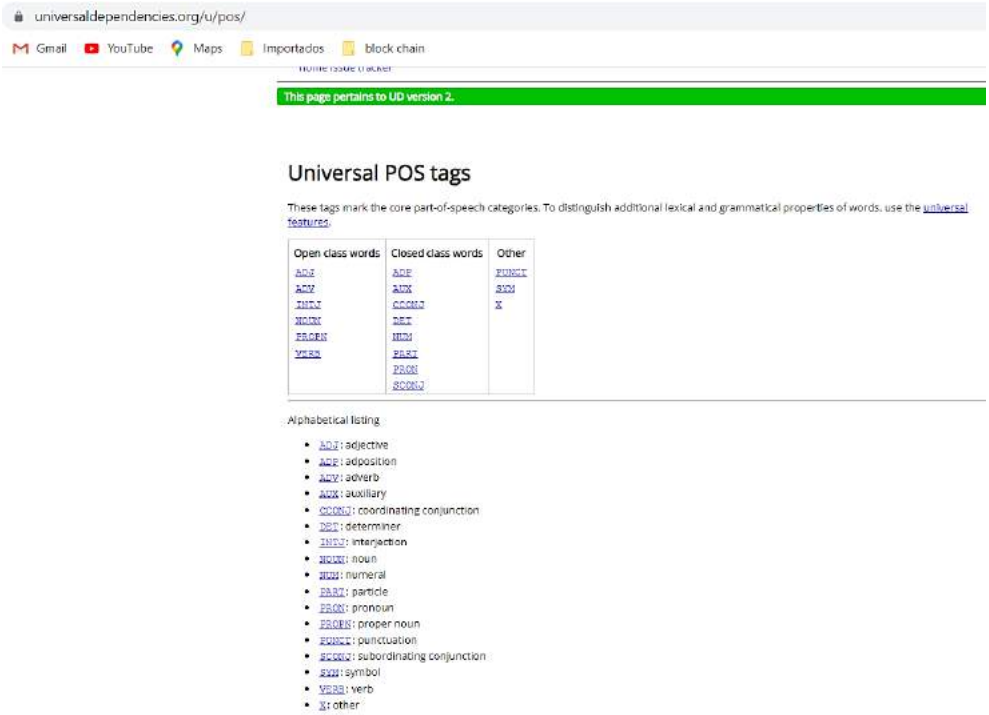
```
!pip install conllu
```

ⓘ NOTA

Conllu es una librería muy utilizada en el procesamiento de lenguaje natural.

- **PASO 3.** Ahora observe las categorías gramaticales que utilizaremos:

  https://universaldependencies.org/u/pos/



- **PASO 4.** Importamos la librería necesaria para lectura de archivos conllu

```
from conllu import parse_incr
```

- **PASO 5.** Ahora creamos una lista vacía:

```
Lista = []
```

- **PASO 6.** Importamos la librería necesaria para lectura de archivos conllu

```
from conllu import parse_incr
```

- **PASO 7.** Ahora vamos a abrir el archivo es_ancora-ud-dev.conllu ,de solo lectura utilizando uf-8 :

```
fichero = open("UD_Spanish-AnCora-master/es_ancora-ud-dev.conllu", "r",
encoding="utf-8")
```

- **PASO 8.** Mostramos una lista de token con sus categorías gramáticas y otros metadatos:

```
for tokenlist in parse_incr(fichero):
 print(tokenlist.serialize())
```

```
for tokenlist in parse_incr(fichero):
    print(tokenlist.serialize())
```

```
# newdoc id = 3LB-CAST-111_C-2
# sent_id = 3LB-CAST-111_C-2-s1
# text = El gobernante, con ganada fama desde que llegó hace 16 meses al poder de explotar al máximo su oratoria y acusado po
r sus detractores de incontinencia verbal, enmudeció desde el momento en el que el Tribunal Supremo de Justicia (TSJ) decidió
suspender temporalmente los comicios múltiples ante la imposibilidad "técnica" de celebrarlos el 28 de mayo.
1       El      el      DET     da0ms0  Definite=Def|Gender=Masc|Number=Sing|PronType=Art       2       det     2:det   _
2       gobernante      gobernante      NOUN    ncms000 Gender=Masc|Number=Sing 32      nsubj   32:nsubj        SpaceAfter=No
3       ,       ,       PUNCT   fc      PunctType=Comm  6       punct   6:punct _
4       con     con     ADP     sps00   _       6       case    6:case  _
5       ganada  ganado  ADJ     aq0fsp  Gender=Fem|Number=Sing|VerbForm=Part    6       amod    6:amod  _
6       fama    fama    NOUN    ncfs000 Gender=Fem|Number=Sing  2       nmod    2:nmod  _
7       desde   desde   ADP     sps00   _       9       mark    9:mark  _
8       que     que     SCONJ   cs      _       9       mark    9:mark  _
9       llegó   llegar  VERB    vmis3s0 Mood=Ind|Number=Sing|Person=3|Tense=Past|VerbForm=Fin   6       acl     6:acl   _
10      hace    hacer   VERB    vmip3s0 Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin   9       advcl   9:advcl _
11      16      16      NUM     _       NumForm=Digit|NumType=Card      12      nummod  12:nummod       _
```

Análisis sintáctico

```
# text = El gobernante, con ganada fama desde que llegó hace 16 meses al poder de explotar al máximo su oratoria y acusado po
r sus detractores de incontinencia verbal, enmudeció desde el momento en el que el Tribunal Supremo de Justicia (TSJ) decidió
suspender temporalmente los comicios múltiples ante la imposibilidad "técnica" de celebrarlos el 28 de mayo.
1       El      el      DET     da0ms0  Definite=Def|Gender=Masc|Number=Sing|PronType=Art       2       det     2:det   _
2       gobernante      gobernante      NOUN    ncms000 Gender=Masc|Number=Sing 32      nsubj   32:nsubj        SpaceAfter=No
3       ,       ,       PUNCT   fc      PunctType=Comm  6       punct   6:punct _
4       con     con     ADP     sps00   _       6       case    6:case  _
5       ganada  ganado  ADJ     aq0fsp  Gender=Fem|Number=Sing|VerbForm=Part    6       amod    6:amod  _
6       fama    fama    NOUN    ncfs000 Gender=Fem|Number=Sing  2       nmod    2:nmod  _
7       desde   desde   ADP     sps00   _       9       mark    9:mark  _
8       que     que     SCONJ   cs      _       9       mark    9:mark  _
9       llegó   llegar  VERB    vmis3s0 Mood=Ind|Number=Sing|Person=3|Tense=Past|VerbForm=Fin   6       acl     6:acl   _
10      hace    hacer   VERB    vmip3s0 Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin   9       advcl   9:advcl _
11      16      16      NUM     _       NumForm=Digit|NumType=Card      12      nummod  12:nummod       _
12      meses   mes     NOUN    ncmp000 Gender=Masc|Number=Plur 10      obj     10:obj  _
13-14   al      _       _       _       _       _       _       _       _
13      a       a       ADP     spcms   15      case    15:case
```

- **PASO 9.** Podemos observar los token de manera separada:

  `tokenlist[1]`

```
tokenlist[1]
```

```
{'id': 2,
 'form': 'El',
 'lemma': 'el',
 'upos': 'DET',
 'xpos': 'da0ms0',
 'feats': {'Definite': 'Def',
  'Gender': 'Masc',
  'Number': 'Sing',
  'PronType': 'Art'},
 'head': 5,
 'deprel': 'det',
 'deps': [('det', 5)],
 'misc': None}
```

- **PASO 10.** Podemos observar el token con su categoría de manera separada:

  `tokenlist[1]['form']+"-"+tokenlist[1]['upos']`

```
tokenlist[1]['form']+"-"+tokenlist[1]['upos']
```

```
'El-DET'
```

5.1.17 Modelos Markoviano de máxima entropía

El modelo de máxima entropía es un modelo determinado en base a la entropía El modelo de máxima entropía cree que el modelo de máxima entropía es el mejor entre todos los modelos de probabilidad posibles. Es decir, solo captamos parte de la información de la distribución desconocida, y hay más de una que se ajusta a la distribución conocida, y luego la distribución está determinada por la entropía máxima.

Modelo Markoviano de Máxima Entropia1

- **PASO 1.** Importamos las librerías a usar:

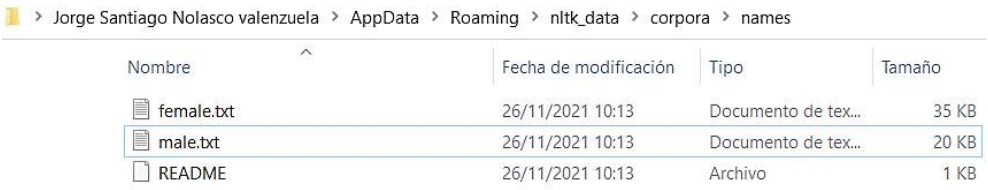
```
import nltk, random
```

- **PASO 2.** Ahora descargamos el dataset:names (nombres en ingles tanto masculinos y femeninos)

```
nltk.download('names')
```

```
nltk.download('names')
```

```
[nltk_data] Downloading package names to
[nltk_data]     C:\Users\jsnv5\AppData\Roaming\nltk_data...
[nltk_data]   Package names is already up-to-date!
```



- **PASO 3.** Importamos el dataset:

```
from nltk.corpus import names
```

- **PASO 4.** Creo dos listas uno para masculino y otro femenino:

```
lista_masculino = [(name, 'male') for name in names.words('male.txt')]
lista_femenino = [(name, 'female') for name in names.words('female.txt')]
```

- **PASO 5.** Ahora unimos las dos listas de manera aleatoria:

```
lista=lista_masculino+lista_femenino
random.shuffle(lista)
```

- **PASO 6.** Observamos su contenido:

```
print(lista)
```

```
print(lista)
```

```
[('Carmelle', 'female'), ('Fan', 'female'), ('Jackelyn', 'female'), ('Orsa', 'female'), ('Hector', 'male'), ('Darbie', 'femal
e'), ('Maurice', 'male'), ('Luise', 'female'), ('Secunda', 'female'), ('Kimmi', 'female'), ('Elane', 'female'), ('Clement',
'male'), ('Jeromy', 'male'), ('Hirsch', 'male'), ('Sophronia', 'female'), ('Reed', 'male'), ('Bailie', 'male'), ('Murray', 'm
ale'), ('Kaari', 'female'), ('Trev', 'male'), ('Connie', 'male'), ('Mela', 'female'), ('Ned', 'male'), ('Egbert', 'male'),
('Sondra', 'female'), ('Ketti', 'female'), ('Claudelle', 'female'), ('Bernie', 'male'), ('Consuela', 'female'), ('Reggie', 'm
ale'), ('Alta', 'female'), ('Gardner', 'male'), ('Marcia', 'female'), ('Claretta', 'female'), ('Reeva', 'female'), ('Phip',
'male'), ('Meridel', 'female'), ('Aubrette', 'female'), ('Lizbeth', 'female'), ('Breanne', 'female'), ('Tirrell', 'male'),
('Crysta', 'female'), ('Idelle', 'female'), ('Carlo', 'male'), ('Raoul', 'male'), ('Alyse', 'female'), ('Grove', 'male'), ('T
essi', 'female'), ('Adolpho', 'male'), ('Jesselyn', 'female'), ('Felice', 'female'), ('Moore', 'male'), ('Voltaire', 'male'),
('Nancie', 'female'), ('Lauryn', 'female'), ('Orson', 'male'), ('Nadeen', 'female'), ('Shel', 'female'), ('Querida', 'femal
e'), ('Ofelia', 'female'), ('Moyra', 'female'), ('Josefina', 'female'), ('Masha', 'female'), ('Cyb', 'female'), ('Zsazsa', 'f
emale'), ('Rubin', 'male'), ('Tricia', 'female'), ('Lilith', 'female'), ('Bay', 'male'), ('Marietta', 'male'), ('Cherye', 'fe
male'), ('Briana', 'female'), ('Mireille', 'female'), ('Piet', 'male'), ('Steve', 'male'), ('Sharity', 'female'), ('Sandy',
'female'), ('Marv', 'male'), ('Tara', 'female'), ('Linus', 'male'), ('Opaline', 'female'), ('Anita', 'female'), ('Dimitri',
'male'), ('Gavrielle', 'female'), ('Dirk', 'male'), ('Ninnetta', 'female'), ('Ulrick', 'male'), ('Nicky', 'male'), ('Francin
e', 'female'), ('Erin', 'female'), ('Randee', 'female'), ('Stephannie', 'female'), ('Algernon', 'male'), ('Allys', 'female'),
('Tildie', 'female'), ('Jemima', 'female'), ('Cora', 'female'), ('Bartholemy', 'male'), ('Claire', 'female'), ('Arnoldo', 'ma
le'), ('Renaud', 'male'), ('Tabbie', 'female'), ('Abbey', 'female'), ('Idalina', 'female'), ('Rowland', 'male'), ('Melany',
'female'), ('Willow', 'female'), ('Tuck', 'male'), ('Mignonne', 'female'), ('Ardis', 'female'), ('Carli', 'female'), ('Donova
```

- **PASO 7.** Ahora creamos una función para extraer la última letra de los nombres:

```
def ultima(nombre):
return {'ultima':nombre[-1]}
```

- **PASO 8.** Ahora creamos una nueva lista a través de la lista: lista, añadiéndole la última letra del nombre:

```
datos = [(ultima(nombre),genero) for (nombre,genero) in lista]
```

- **PASO 9.** Efectuó la división entre Entrenamiento y Prueba:

```
train, test = datos[500:],datos[:500]
```

- **PASO 10.** Ahora creamos el clasificador (entrenamos el modelo)

```
classifier=nltk.NaiveBayesClassifier.train(train)
```

- **PASO 11.** Verificamos la predicción

```
classifier.classify(ultima('ana'))
```

```
classifier.classify(ultima('ana'))
```

```
'female'
```

- **PASO 12.** Calculando la métrica del modelo

```
nltk.classify.accuracy(classifier, test)
```

```
nltk.classify.accuracy(classifier, test)
```

```
0.74
```

ⓘ NOTA

De cada 100 personas, acertamos su sexo en 74.

http://www.inkadroid.com/cms1/

Para ello es necesario instalar los siguientes paquetes

```
$ pip install beautifulsoup4
$ pip install requests
```

URL

Ahora analizaremos la página para indicar qué porción extraeremos:

1. Ingrese a la URL:
   http://www.inkadroid.com/cms1/

Consola

2. Presione f12–Elements:
   Queremos extraer la información:
   Bienvenido a la Página Oficial de Inkadroid
   El contenido se encuentra en la siguiente etiqueta:

```
<h1 class="page-header text-center"> == $0
  "
                    Bienvenido a la Página Oficial de Inkadroid
                "
</h1>
```

Escribiendo el código:

Scraping1.py

```
#liberrias necesarias
from bs4 import BeautifulSoup
import requests
URL = "http://www.inkadroid.com/cms1/"
#realizamos la peticion WEB
req = requests.get(URL)
#obtenemos el Status Code
codigo_estatus = req.status_code
# Comprobamos que la petición nos devuelve un Status Code = 200
if codigo_estatus == 200:
 # el contenido de la pagina web lo almacenamos a un objeto BeautifulSoup()
 html = BeautifulSoup(req.text, "html.parser")
 # Obtenemos el div y lo almacenamos en obtenidos
 obtenidos = html.find_all('div', {'class': 'col-lg-12'})
 # Recorremos obtenidos para extraer los datos
 for obtenido in obtenidos:
 try:
 mostrar=obtenido.find('h1', {'class': 'page-header text-center'}).getText()
 # Imprimo los datos extraidos
 print("%s" % (mostrar))
 except:
 pass
else:
 #imprimir codigo de error
 print("ERROR",codigo_estatus)
```

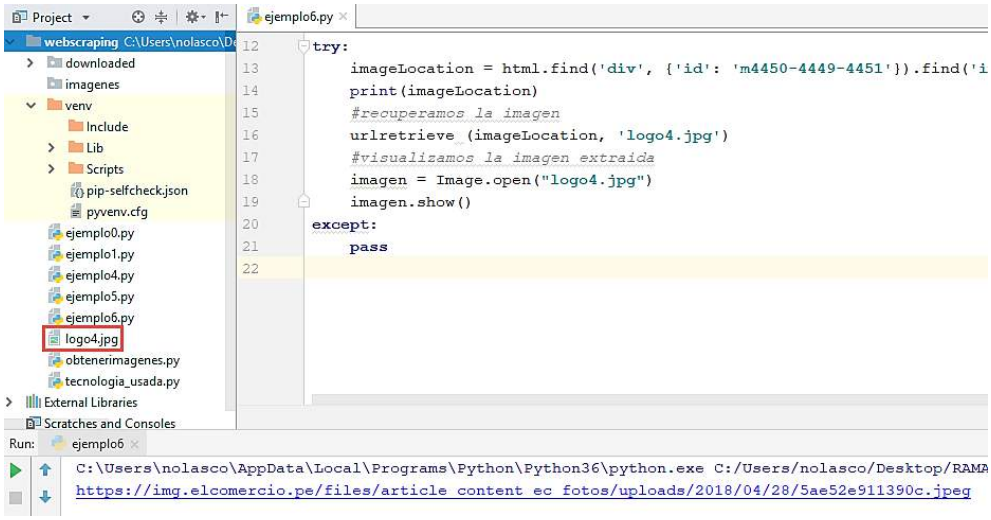
Escribiendo el código:

Scraping2.py

```
from PIL import Image
from urllib.request import urlretrieve
from bs4 import BeautifulSoup
import requests
URL = “https://elcomercio.pe/opinion/editorial/editorial-pobres-teoria-frente-
```

```
amplio-gregorio-santos-pobreza-noticia-515833”
#realizamos la peticion WEB
req = requests.get(URL)
#obtenemos el Status Code
codigo_estatus = req.status_code
# el contenido de la pagina web lo almacenamos a un objeto BeautifulSoup()
html = BeautifulSoup(req.text, “html.parser”)
try:
 imageLocation = html.find(‘div’, {‘id’: ‘m4450-4449-4451’}).find(‘img’)[‘src’]
 print(imageLocation)
 #recuperamos la imagen
 urlretrieve (imageLocation, ‘logo4.jpg’)
 #visualizamos la imagen extraida
 imagen = Image.open(“logo4.jpg”)
 imagen.show()
except:
 pass
```

Ejecutándose:



IDE: Pycharm

- **cv2.IMREAD_GRAYSCALE:** carga la imagen en modo de escala de grises.
- **cv2.IMREAD_UNCHANGED:** carga la imagen como sin alteraciones incluyendo el canal alfa.

7.2.2 imshow

Utiliza la función cv2.imshow () para mostrar una imagen en una ventana. La ventana se ajusta automáticamente al tamaño de la imagen.

El primer argumento es un nombre de ventana el cual es una cadena (tipo de dato string). El segundo argumento es nuestra imagen. Puedes crear tantas ventanas como desees, pero con nombres diferentes de ventana.

```
cv2.imshow ('image',img)
cv2.waitKey (0)
cv2.destroyAllWindows ()
```

cv2.waitKey () es una función de enlace con el teclado. Su argumento es tiempo en milisegundos. La función espera durante milisegundos especificados que suceda cualquier evento de teclado. Si presionas cualquier tecla en ese momento, el programa continúa. Si se pasa el valor 0, la espera del evento es indefinida hasta que se presione una tecla. También se puede configurar para detectar pulsaciones de teclas específicas, por ejemplo, si se presiona tecla a tecla, etc., lo cual veremos más adelante.

cv2.destroyAllWindows () esta función destruye todas las ventanas que hemos creado. Si deseas destruir una ventana específica, utilice la función de cv2. destroyWindow () donde se pasa el nombre de la ventana a eliminar como argumento.

Imread

Utiliza la función cv2.imread () para leer una imagen. La imagen debe estar en el directorio de trabajo o se ha de señalar una ruta absoluta a la imagen.

El segundo argumento es un indicador (o bandera) que especifica la forma en que se debe leer la imagen.

- cv2.IMREAD_COLOR: carga una imagen de color. Cualquier transparencia de la imagen se ignorará. Es el indicador (o bandera) predeterminado.
- cv2.IMREAD_GRAYSCALE: carga la imagen en modo de escala de grises.
- cv2.IMREAD_UNCHANGED: carga la imagen como sin alteraciones incluyendo el canal alfa.

imwrite

Utiliza la función cv2.imwrite () para guardar una imagen.

El primer argumento es el nombre del archivo y el segundo argumento es la imagen que deseas guardar.

```
cv2.imwrite (‘deepgris.png’,img)
```

7.3 LEER IMÁGENES

OpenCV proporciona la función imread() para la lectura de imágenes, puede admitir diferentes formatos de imágenes :PNG, JPEG y TIFF:

```
import cv2
img=cv2.imread (‘imagen.jpg’)
cv2.imshow (‘leyendo’,img)
cv2.waitKey ()
```

Notebook: Jupyter

7.4 ESCRIBIR IMÁGENES

OpenCV proporciona la función imwrite () para la escritura de imágenes, puede admitir diferentes formatos de imágenes: PNG, JPEG y TIFF:

```
import cv2
img=cv2.imread (‘imagen.jpg’)
cv2.imshow (‘nueva imagen’,img)
cv2.imwrite (‘output.jpg’,img)
cv2.waitKey ()
```

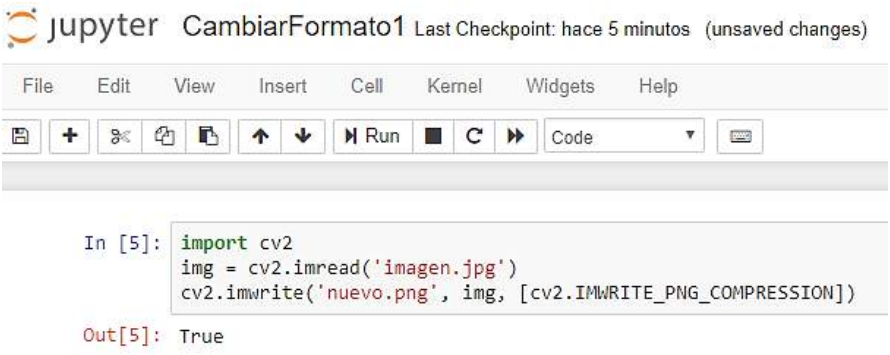
7.5 CAMBIANDO EL FORMATO DE UNA IMAGEN

También podemos guardar esta imagen como un archivo y cambiar el formato de imagen original a PNG:

```
import cv2
img = cv2.imread (‘imagen.jpg’)
cv2.imwrite (‘nuevo.png’, img, [cv2.IMWRITE_PNG_COMPRESSION])
```

El método imwrite () guardará la imagen en escala de grises como un archivo de salida llamado

- **output.png.** Esto se hace usando compresión PNG con la ayuda de ImwriteFlag y
- **cv2.IMWRITE_PNG_COMPRESSION.** El ImwriteFlag permite que la imagen de salida cambie el formato, o incluso la calidad de imagen.



7.6 MODELO DE COLOR YUV

YUV es un espacio de color típicamente usado como parte de un sistema de procesamiento de imagen en color. Una imagen o vídeo en color se codifica en este espacio de color teniendo en cuenta la percepción humana, lo que permite un ancho de banda reducido para los componentes de diferencia de color o crominancia, de esta forma, hace que los errores de transmisión o las imperfecciones de compresión se oculten más eficientemente a la percepción humana que usando una representación RGB directa. Otros espacios de color tienen propiedades similares y la principal razón para implementar o investigar las propiedades de YUV o de su similar, YUV se encuentran tanto las de interactuar con televisión analógica o digital o con equipo fotográfico que sea conforme a ciertos estándares de este espacio.

A continuación, se muestran las relaciones entre Y y R, entre G y B, entre U, R y luminancia, y finalmente entre V, B y luminancia:

Y = 0.299R + 0.587 G + 0.114 B
U = -0.147R–0.289 G + 0.436B = 0.492(B–Y)
V = 0.615R -0.515G -0.100B = 0.877(R-Y)

Por lo tanto, U a veces se escribe como Cr y V a veces se escribe como Cb, de ahí la notación YCrCb.

7.7 MODELO DE COLOR YUV – DIVISIÓN DE COLORES

YUV es un espacio de color típicamente usado como parte de un sistema de procesamiento de imagen en color. Una imagen o vídeo en color se codifica en este espacio de color teniendo en cuenta la percepción humana, lo que permite un ancho, ejemplo:

```
import cv2
img = cv2.imread ('imagen.jpg', cv2.IMREAD_COLOR)
gris_img = cv2.cvtColor (img, cv2.COLOR_RGB2GRAY)
yuv_img = cv2.cvtColor (img, cv2.COLOR_BGR2YUV)
y,u,v = cv2.split (yuv_img)
cv2.imshow ('Imagen en Escala Gris', gris_img)
# Y luminancia Y=0.2992+0.587G+0.114B
cv2.imshow ('Y', y)
# U crominancia U=B-Y
cv2.imshow ('U', u)
# V crominancia U=R-Y
cv2.imshow ('V', v)
cv2.waitKey ()
```

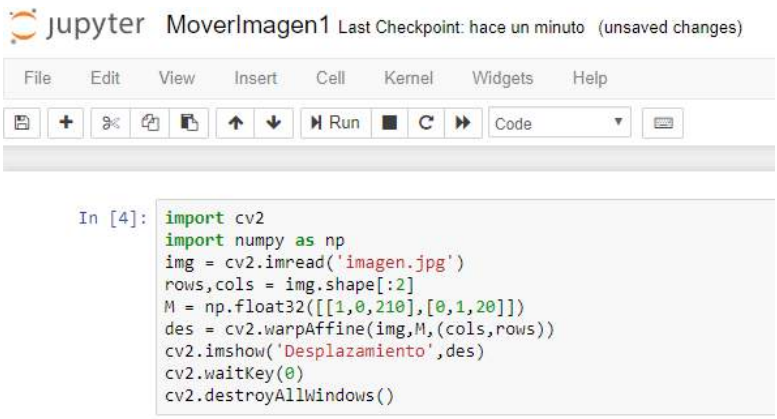
7.9 TRASLACIÓN DE IMÁGENES

Una traslación es el desplazamiento de la posición de un objeto. Si se conoce la magnitud del desplazamiento (t_x,t_y) en las direcciones x e y, respectivamente, se puede escribir la matriz de transformación M como:

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

A continuación, el siguiente ejemplo:

```
import cv2
import numpy as np
img = cv2.imread (‘imagen.jpg’)
rows,cols = img.shape[:2]
M = np.float32 ([[1,0,210],[0,1,20]])
des = cv2.warpAffine(img,M,(cols,rows))
cv2.imshow (‘Desplazamiento’,des)
cv2.waitKey (0)
cv2.destroyAllWindows ()
```



Notebook: Jupyter

A continuación mostramos un ejemplo:

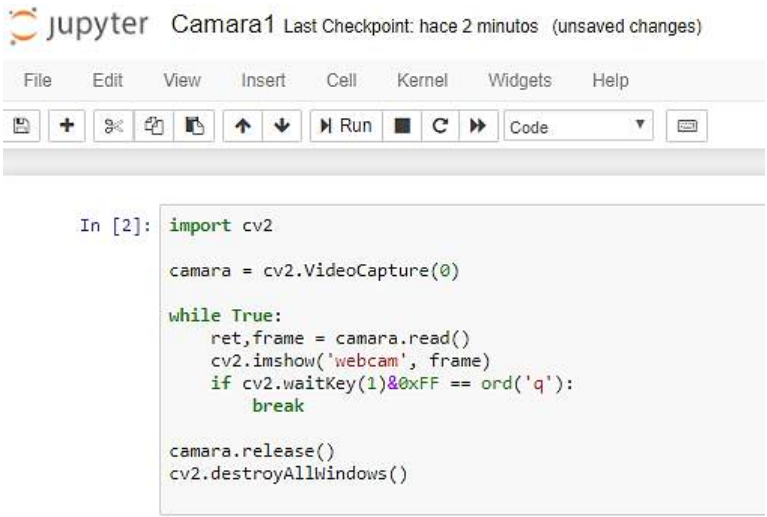
```
import cv2
import numpy as np
img = cv2.imread('imagen.jpg')
rows,cols = img.shape[:2]
M = cv2.getRotationMatrix2D((cols,rows),45,1)
dst = cv2.warpAffine(img,M,(cols,rows))
cv2.imshow('Rotacion', img_rotation)
cv2.waitKey()
```

7.11 UTILIZANDO LA CÁMARA

Usando la biblioteca OpenCV podemos acceder a la cámara web o cualquier otro dispositivo de captura que tengamos instalado en nuestro sistema, cada una de las imágenes capturadas podrán almacenarse para su análisis o procesamiento en tiempo real si así lo deseamos, tenemos disponible una clase que nos servirá para guardar los videos previamente capturados y procesados, el formato de almacenamiento depende de las características habilitadas, pero puede ser MP4, AVI, WMV, etc., y otros si tenemos los códec.

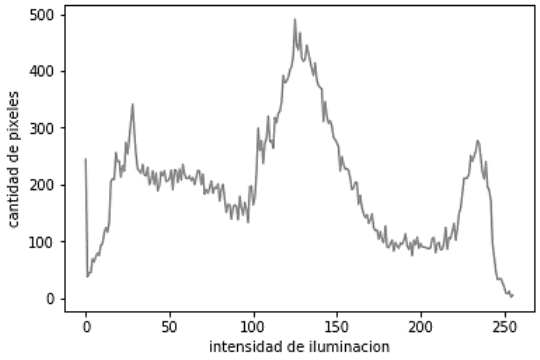
Camara1.ipynb

```
import cv2

camara = cv2.VideoCapture(0)
while True:
 ret,frame = camara.read()
 cv2.imshow('webcam', frame)
 if cv2.waitKey(1)&0xFF == ord('q'):
  break

camara.release()
cv2.destroyAllWindows()
```



MayorIntesidaddecolor1gris .ipynb

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
#cv2.IMREAD_GRAYSCALE: carga la imagen en modo de escala de grises
img = cv2.imread('imagen2.jpg', cv2.IMREAD_GRAYSCALE)
#cv2.imshow() para mostrar una imagen en una ventana
cv2.imshow('imagen2.jpg', img)
""""

cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]]
1. images: imagen de estrada, puede ser a escala de grises o colores.
2- channels: índice de canal para el cual deseamos calcular el histograma, en
una imagen a escala de grises [0],
si la imagen es a colores podemos indicar [0], [1], [2] para los canales B, G, R
respectivamente.
3. mask: mascara que define la región sobre la que deseamos calcular el histogra-
ma, es opcional.
4. histSize: intensidad máxima, para nosotros [256].
5. ranges: nuestro rango de valores, usaremos [0, 256]
""""

hist = cv2.calcHist([img], [0], None, [256], [0, 256])
#color gris
plt.plot(hist, color='gray' )
#etiqueta del eje X
plt.xlabel('intensidad de iluminacion')
#etiqueta del eje X
plt.ylabel('cantidad de pixeles')
#muestra el histograma
plt.show()
#destruye las ventanas
cv2.destroyAllWindows()
```



MayorIntesidaddecolor1color.ipynb

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
#cv2.IMREAD_GRAYSCALE: carga la imagen
img = cv2.imread('imagen.jpg')
#cv2.imshow() para mostrar una imagen en una ventana
cv2.imshow('imagen.jpg', img)
#canal: b=azul, g=verde, r=azul
color = ('b','g','r')
#recorrido por cada canal
for i, c in enumerate(color):
 """

 cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]]
 1. images: imagen de estrada, puede ser a escala de grises o colores.
 2- channels: índice de canal para el cual deseamos calcular el histograma, en
una imagen a escala de grises [0],
 si la imagen es a colores podemos indicar [0], [1], [2] para los canales B, G,
R respectivamente.
 3. mask: mascara que define la región sobre la que deseamos calcular el histo-
grama, es opcional.
 4. histSize: intensidad máxima, para nosotros [256].
 5. ranges: nuestro rango de valores, usaremos [0, 256]
 """

 hist = cv2.calcHist([img], [i], None, [256], [0, 256])
 #color = c
 plt.plot(hist, color = c)
 #establece límites x de los ejes actuales
 plt.xlim([0,256])
#muestra el histograma
plt.show()
#destruye las ventanas
cv2.destroyAllWindows()
```

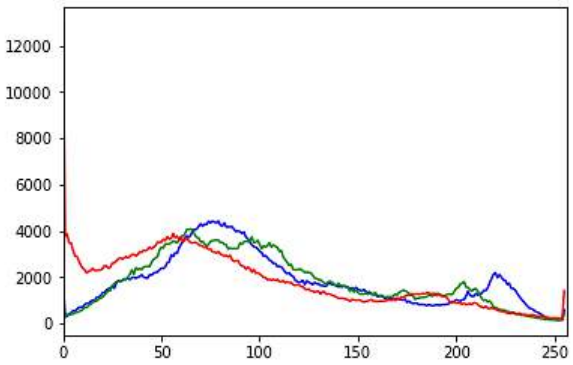
jupyter MayorIntesidaddecolor1color Last Checkpoint: hace 7 minutos (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [2]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
#cv2.IMREAD_GRAYSCALE: carga la imagen
img = cv2.imread('imagen.jpg')
#cv2.imshow() para mostrar una imagen en una ventana
cv2.imshow('imagen.jpg', img)
#canal; b=azul , g=verde , r=azul
color = ('b','g','r')
#recorrido por cada canal
for i, c in enumerate(color):
    """
    cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]]
    1.-images: imagen de estrada, puede ser a escala de grises o colores.
    2- channels: indice de canal para el cual deseamos calcular el histograma, en una imagen a escala de grises [0],
    si la imagen es a colores podemos indicar [0], [1], [2] para los canales B, G, R respectivamente.
    3.-mask: mascara que define la región sobre la que deseamos calcular el histograma, es opcional.
    4.-histSize: intensidad máxima, para nosotros [256].
    5.-ranges: nuestro rango de valores, usaremos [0, 256]
    """
    hist = cv2.calcHist([img], [i], None, [256], [0, 256])
    #color = c
    plt.plot(hist, color = c)
    #establece límites x de los ejes actuales
    plt.xlim([0,256])
#muestra el histograma
plt.show()
#destruye las ventanas
cv2.destroyAllWindows()
```

Notebook: Jupyter



7.13 ECUALIZACIÓN DE HISTOGRAMAS

Un histograma es una representación gráfica de la distribución de los niveles de grises en una imagen, utilizando el método de Ecualización de Histogramas podemos obtener una distribución uniforme de los distintos niveles de intensidad, se utiliza esta técnica para mejorar el contraste de una imagen.

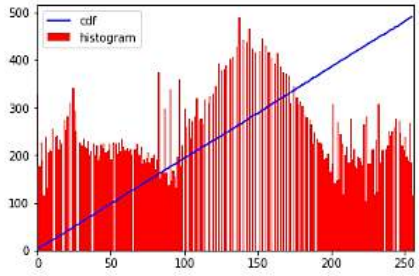
Para aplicar la ecualización de histograma OpenCV nos provee la función:

```
cv2.equalizeHist(src)
```

ecualizacion1.ipynb

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('imagen2.jpg',0)
img = cv2.equalizeHist(img)
cv2.imshow('Histogramas', img)
cv2.waitKey()
hist,bins = np.histogram(img.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * hist.max()/ cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(img.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('imagen2.jpg',0)

img = cv2.equalizeHist(img)
cv2.imshow('Histogramas', img)


hist,bins = np.histogram(img.flatten(),256,[0,256])

cdf = hist.cumsum()
cdf_normalized = cdf * hist.max()/ cdf.max()

plt.plot(cdf_normalized, color = 'b')
plt.hist(img.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()

cv2.waitKey()
```

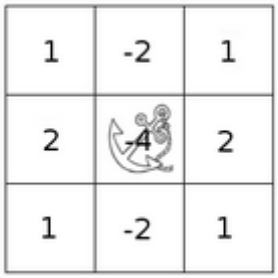


Notebook: Jupyter

7.14 CONVOLUCIÓN DE IMÁGENES

Un filtro de imagen es un procedimiento que se aplica a una imagen para resaltar o mejorar algunas características de la misma, para lograr esto se modifica la matriz que compone la imagen aplicándole un determinado procedimiento, en este tutorial estudiaremos el procedimiento llamado convolución de matrices.

OpenCV cuenta con gran variedad de función que aplican los distintos filtros más comunes, estos.



A continuación, algunos ejemplos:

```
import cv2
import numpy as np

img = cv2.imread('imagen2.jpg')
rows, cols = img.shape[:2]

kernel_identity = np.array([[0,0,0], [0,1,0], [0,0,0]])
kernel_3x3 = np.ones((3,3), np.float32) / 9.0
kernel_5x5 = np.ones((5,5), np.float32) / 25.0

cv2.imshow('Original', img)

output = cv2.filter2D(img, -1, kernel_identity)
cv2.imshow('Filtro de Identidad', output)

output = cv2.filter2D(img, -1, kernel_3x3)
cv2.imshow('Filtro de 3x3 ', output)

output = cv2.filter2D(img, -1, kernel_5x5)
cv2.imshow('Filtro de 5x5', output)
cv2.waitKey(0)
```

Primero tenemos que cargar los clasificadores XML requeridos. Luego cargue nuestra imagen de entrada (o video) en el modo de escala de grises.

```
import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

img = cv2.imread('sachin.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Ahora encontramos las caras en la imagen. Si se encuentran caras, devuelve las posiciones de las caras detectadas como Rect (x, y, w, h). Una vez que tenemos estos lugares, podemos crear un ROI para la cara y aplicar la detección de ojos en este ROI (¡¡¡ya que los ojos siempre están en la cara !!!).

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
roi_gray = gray[y:y+h, x:x+w]
roi_color = img[y:y+h, x:x+w]
eyes = eye_cascade.detectMultiScale(roi_gray)
for (ex,ey,ew,eh) in eyes:
 cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

cv2.imshow('img',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Ahora ejecutamos el código anterior:

Deteccion1.ipynb

```
import numpy as np
import cv2

#cargamos los clasificadores requeridos
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')
#utilizamos la camara 1
cap = cv2.VideoCapture(0)
while(True):
 #lee el objeto de la camara
 ret, img = cap.read()
 #convertimos la imagen a blanco y negro
 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
 #buscamos las coordenadas de los rostros
 faces = face_cascade.detectMultiScale(gray, 1.3, 5)
 #Dibujamos un rectangulo en las coordenadas de cada rostro
 for (x,y,w,h) in faces:
  cv2.rectangle(img,(x,y),(x+w,y+h),(125,255,0),2)
 #Mostramos la imagen
 cv2.imshow('img',img)
 #con la tecla 'q' salimos del programa
 if cv2.waitKey(1) & 0xFF == ord('q'):
  break
```

Este sencillo ejemplo demuestra el cifrado de Julio César:

Julio Cesar.ipynb

```
letras="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
def cifrar(msg,key):
    mensaje=msg
    key=key
    cifrado = ""
    for caracter in mensaje:
        if caracter.upper() in letras:
            num = letras.find(caracter.upper())
            num = num + key

            if num >= len(letras):
                num -= len(letras)
            elif num < 0:
                num += len(letras)

            cifrado += letras[num]
        else:
            cifrado += caracter
    return cifrado

msg=input("Ingrese Mensaje a Cifrar:")
key=int(input("Ingrese Key:"))
print(cifrar(msg,key))
```

```
Ingrese Mensaje a Cifrar:jorge nolasco valenzuela
Ingrese Key:4
NSVKI RSPEWGS ZEPIRDYIPE
```

8.4 ALGORITMOS DISPONIBLES

Python contiene un módulo incorporado llamado hashlib, este módulo puede definir un Api que puede realizar un hash criptográfico unidireccional.

8.4.3 SHA512

Este sencillo ejemplo demuestra lo fácil que es acceder y utilizar la biblioteca estándar de Python y cómo se ejecuta el mismo código en diferentes plataformas y genera el valor de hash SHA512:

SHA512.Ipynb

```
#programa : SHA512.Ipynb
#autor : jorge nolasco valenzuela
#fecha : 23-12-2021
"""
descripcion : este programa muestra
el uso de import hashlib - Algoritmo SHA512
"""
import hashlib
#ingresamos la cadena
cadena=input("Ingrese Cadena a Codificar :")
# Crear un objeto llamado hash1 que es del tipo SHA512
hash1=hashlib.sha512()
# Utilizar el método update para generar el SHA512 de la cadena
hash1.update(cadena.encode("UTF-8"))
# Obtener los valores hexadecimales generados del SHA512 ,Mediante la utilización del método hex-digest
hex1=hash1.hexdigest()
# muestra el resultado
print("SHA512: " + hex1.upper())
```

```
Ingrese Cadena a Codificar :Jorge Nolasco Valenzuela
SHA512: 5B2091E7F21A965518025F9E543CA67DB5C8B4C57F9DB99E312D2A021D225841C22C2D3B07E2D560762B43E86CC6383AF578EEB3C85AB11439450A8
D9C467F83
```

8.4.4 Diferentes

Este sencillo ejemplo demuestra el uso de diferentes algoritmos.

diferentes.py

```
#programa: diferentes.Ipynb
#autor: jorge nolasco valenzuela
#fecha: 23-12-2021
“””
descripcion: este programa muestra
el uso de diferentes algoritmos hash
“””
import hashlib
import os
def tamano(the_path):
 path_size = 0
 for path, dirs, files in os.walk(the_path):
  for fil in files:
   filename = os.path.join(path, fil)
   path_size += os.path.getsize(filename)

 listado=os.listdir(“imagenes”)
 size=path_size/1024
 return listado,path_size
def md5(ruta):
```

```
 listado2=os.listdir(ruta)
 for fichero in listado2:
  fp = open(ruta+"/"+fichero, "rb")
  buffer = fp.read()
  # md5
  hashObj = hashlib.md5()
  hashObj.update(buffer)
  lastHash = hashObj.hexdigest().upper()
  md5 = lastHash
  fp.close()
  return fichero,buffer,lastHash
def sha1(ruta):
 listado2=os.listdir(ruta)
 for fichero in listado2:
  fp = open(ruta+"/"+fichero, "rb")
  buffer = fp.read()
  # sha1
  hashObj = hashlib.sha1()
  hashObj.update(buffer)
  lastHash = hashObj.hexdigest().upper()
  sha1 = lastHash
  fp.close()
  return fichero,buffer,lastHash

def sha224(ruta):
 listado2=os.listdir(ruta)
 for fichero in listado2:
  fp = open(ruta+"/"+fichero, "rb")
  buffer = fp.read()
  # sha224
  hashObj = hashlib.sha224()
  hashObj.update(buffer)
  lastHash = hashObj.hexdigest().upper()
  sha224 = lastHash
  fp.close()
  return fichero,buffer,lastHash
def sha256(ruta):
 listado2=os.listdir(ruta)
 for fichero in listado2:
  fp = open(ruta+"/"+fichero, "rb")
  buffer = fp.read()
  # sha256
  hashObj = hashlib.sha256()
  hashObj.update(buffer)
 lastHash = hashObj.hexdigest().upper()
  sha256 = lastHash
  fp.close()
  return fichero,buffer,lastHash
def sha384(ruta):
 listado2=os.listdir(ruta)
 for fichero in listado2:
  fp = open(ruta+"/"+fichero, "rb")
  buffer = fp.read()
  # sha384
  hashObj = hashlib.sha384()
  hashObj.update(buffer)
  lastHash = hashObj.hexdigest().upper()
```

```
    sha384 = lastHash
    fp.close()
    return fichero,buffer,lastHash
  def sha512(ruta):
   listado2=os.listdir(ruta)
   for fichero in listado2:
    fp = open(ruta+"/"+fichero, "rb")
    buffer = fp.read()
    # sha512
    hashObj = hashlib.sha512()
    hashObj.update(buffer)
    lastHash = hashObj.hexdigest().upper()
    sha512 = lastHash
    fp.close()
    return fichero,buffer,lastHash
  while True:
   print("MENU:")
   print("1: TAMAÑO FICHERO")
   print("2: MD5")
   print("3: SHA1")
   print("4: SHA224")
   print("5: SHA256")
   print("6: SHA384")
   print("7: SHA512")
   print("8: SALIR")
   opc=int(input("SELECCIONE OPCION :"))
   if opc==1:
    listado1,size1=tamano("imagenes")
    print("=" * 40)
    print("Listado de Ficheros:",listado1)
    print("Tamaño:{0:8.0f} kb".format(size1/ 1024))
    print("=" * 40)
   elif opc==2:
    print("=" * 40)
    fichero, buffer, lastHash= md5("imagenes")
    print("FICHERO :"+fichero)
    print("MD5 :"+lastHash)
    print("=" * 40)
   elif opc==3:
    print("=" * 40)
    fichero, buffer, lastHash = sha1("imagenes")
    print("FICHERO :" + fichero)
    print("SHA1 :" + lastHash)
    print("=" * 40)
   elif opc == 4:
    print("=" * 40)
    fichero, buffer, lastHash = sha224("imagenes")
    print("FICHERO :" + fichero)
    print("SHA224 :" + lastHash)
    print("=" * 40)
   elif opc == 5:
    print("=" * 40)
    fichero, buffer, lastHash = sha256("imagenes")
    print("FICHERO :" + fichero)
    print("SHA256 :" + lastHash)
    print("=" * 40)
   elif opc == 6:
```

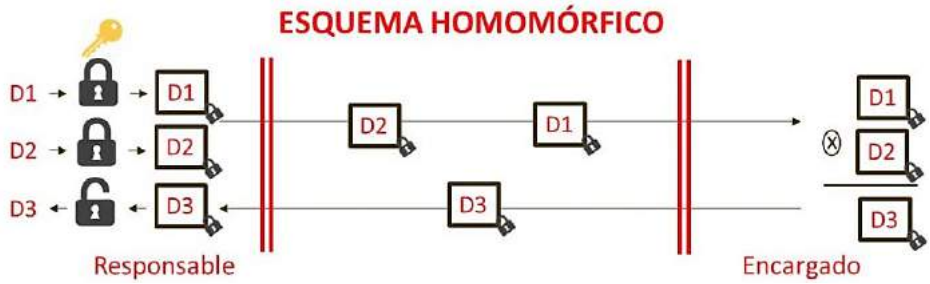
```
  print("=" * 40)
  fichero, buffer, lastHash = sha384("imagenes")
  print("FICHERO :" + fichero)
  print("SHA384 :" + lastHash)
  print("=" * 40)

 elif opc == 7:
  print("=" * 40)
  fichero, buffer, lastHash = sha512("imagenes")
  print("FICHERO :" + fichero)
  print("SHA512 :" + lastHash)
  print("=" * 40)

 else:
  break
```

MENU:
1: TAMAÑO FICHERO
2: MD5
3: SHA1
4: SHA224
5: SHA256
6: SHA384
7: SHA512
8: SALIR
SELECCIONE OPCIÓN :1
==
Listado de Ficheros: ['imagen1.png']
Tamaño: 668 kb
==
MENU:
1: TAMAÑO FICHERO
2: MD5
3: SHA1
4: SHA224
5: SHA256
6: SHA384
7: SHA512
8: SALIR
SELECCIONE OPCIÓN :2
==
FICHERO :imagen1.png
MD5 :3F9AF6B0FD1AEC2C9DE13B75C05DB081
==
MENU:
1: TAMAÑO FICHERO
2: MD5
3: SHA1
4: SHA224
5: SHA256
6: SHA384

El cifrado homomórfico es una técnica que permite realizar operaciones sobre los datos cifrados y obtener resultados, también cifrados, equivalentes a las operaciones realizadas directamente sobre los datos fuentes. Aunque a primera vista puede parecer mágico, lo cierto es que a nuestro alrededor abundan algoritmos criptográficos de uso cotidiano que soportan parcialmente el cifrado homomórfico.

Si se parte del esquema tradicional descrito, lo ideal para minimizar los riesgos sería que el encargado no tuviera la posibilidad de descifrar la información, y que todo su tratamiento pudiera llevarse a cabo sobre los datos cifrados por el responsable. De esta forma, se evitaría que un encargado desleal o un tercero suyo pudieran acceder a los mismos y usarlos para finalidades diferentes. Una forma de conseguir esta protección es mediante el llamado cifrado homomórfico.

El cifrado homomórfico es una técnica que permite realizar operaciones sobre los datos cifrados y obtener resultados, también cifrados, equivalentes a las operaciones realizadas directamente sobre la información original. En la figura 2 se muestra uno de los posibles casos de uso de esta técnica.



8.4.6 Lista archivos

El método listdir () devuelve una lista que contiene los nombres de los archivos en el directorio actual:

Listado.py

```
#programa: Listado.py
#autor: jorge nolasco valenzuela
#fecha: 23-12-2021
“””
descripcion: este programa muestra
el uso de la funcion listdir
“””
import os
mycwd = os.getcwd()
directorios = os.listdir(mycwd)
print(directorios)
```

```
['.idea', 'algoritmos.py', 'Listado.py', 'MD5.py', 'SHA1.py', 'SHA512.py']
```

8.4.7 Plataforma (platform)

Algunas veces es necesario saber qué tipo de sistema está ejecutando un programa, para ello utilizamos el módulo plataforma:

Plataforma.py

```
#programa: Plataforma.py
#autor: jorge nolasco valenzuela
#fecha: 23-12-2021
“””
descripcion: este programa muestra
el uso del módulo plataform
“””
import sys
import platform
print(“Platform: “+sys.platform)
print(“Machine: “+platform.machine())
print(“Version: “+platform.version())
```

```
Platform: win32
Machine: AMD64
Version: 6.3.9600
```

8.4.8 Socket

Los sockets son un canal de comunicación de datos bidireccional dentro de un proceso. Entre procesos en la misma máquina, o entre procesos en diferentes máquinas puede usar protocolos como tcp/ip o udp.

Los tipos de socket son los siguientes:

- **SOCK_STREAM:** este protocolo nos da una comunicación fiable de dos direcciones en un flujo de datos (protocolo: TCP).
- **SOCK_DGRAM:** este protocolo nos da una conexión no fiable (protocolo: UDP).
- **SOCK_RAW:** este protocolo es para acceder a los campos e interfaces internos de la red.
- **SOCK_RDM:** este protocolo garantiza la llegada de paquetes, pero no garantiza el orden de llegada.
- **SOCK_SEQPACKET:** datagramas fiables y secundarios, de longitud fija, basado en la conexión.
- **SOCK_PACKET:** coloca el socket en modo promiscuo en la que recibe todos los paquetes de la red.

Ahora listaremos los métodos más importantes:

- **socket.socket:** crea un canal bidireccional con el que generalmente se establece una conexión de red.
- **socket.bind:** define un puerto y un nombre para un socket.
- **socket.listen:** convierte el socket en un socket en escucha.
- **socket.accept:** espera que llegue una conexión. Al llegar la descripción devuelve un socket nuevo para dicha conexión específica.
- **socket.connect:** conecta un socket con otro que lo esté esperando en un puerto y dirección específica.
- **socket.send:** es por el método que enviaremos los mensajes.
- **socket.recv:** es por el método que recibiremos los mensajes.
- **socket.close:** cierra el socket.

A continuación, un ejemplo de Socket:

Servidor-Clientes

SocketServidor.py

```
#programa: SocketServidor.py
#autor: jorge nolasco valenzuela
#fecha: 23-12-2021
“””
descripcion: este programa muestra
el uso de socket
Servidor
“””
#libreria para trabajar socket
import socket
import sys

#crear socket
s1=socket.socket()
#hostname para recibir conexion externas
HOSTNAME=””
#puerto de escucha de nuestra servidor
PORT=8765

try:
 “””
 el metodo bind
  conecta el socket en una tupla que especifique
 una direccion y puerto
```

```
 “””
 s1.bind((HOSTNAME,PORT))
except socket.error as message:
 print(“bind fallo”)
 sys.exit()
#comenzamos la escucha
s1.listen()
print(“escuchando socket en puerto :”,PORT)

while True:
 #a la espera de una conexion
 connection,address = s1.accept()
 print(“conexion con :”,address[0],address[1])
  connection.send(“HOLA DESDE EL SERVIDOR”)
 #cerrrar el socket
 connection.close()
```

SocketCliente.py

```
#programa: SocketCliente.py
#autor: jorge nolasco valenzuela
#fecha: 23-12-2021
“””
descripcion: este programa muestra
el uso de socket
Clientes
“””
#libreria para trabajar socket
import socket
import sys

#crear socket
s1=socket.socket()
#obtenemos el nombre de host
HOST=socket.gethostname()
#el puerto a utilizar
PORT=8765
#Conecta el socket del cliente con el servidor en un puerto
s1.connect((HOST,PORT))
#recibiendo mensajes
print(s1.recv(1024))
#cerrar el socket
s1.close()
```

8.4.9 Obtener la dirección IP

Python ha ganado mucha fuerza en el desarrollo de programas para la red, ya que, gracias al manejo de sockets, podemos utilizar la estructura de red, dicha estructura puede ser local o a través de Internet, a continuación, mostraremos algunos ejemplos:

Socket1.py

```
#programa: Socket1.py
#autor: jorge nolasco valenzuela
#fecha: 23-12-2021
“””
descripcion: este programa muestra
el uso de socket
“””
import socket
#obtener el nombre del equipo
nombre_equipo = socket.gethostname()
#obtener la direccion de IP del equipo
direccion_equipo = socket.gethostbyname(nombre_equipo)
print(“el nombre del equipo es:”,nombre_equipo)
print(“La IP es:”,direccion_equipo)
```

el nombre del equipo es: jnolasco
La IP es: 192.168.56.1

8.4.10 Listar direcciones IPs

Socket2.py

```
#programa: Socket2.py
#autor: jorge nolasco valenzuela
#fecha: 23-12-2021
“””
descripcion: este programa muestra
el uso de socket
“””
# Especificamos la IP de la Red
ipBase=’192.168.0.’
# lista de IPs
ipList=[]
# IPs de 1-255
for ip in range(0,256):
 ipList.append(ipBase+str(ip))
 print(ipList.pop())
```

192.168.0.0
192.168.0.1
192.168.0.2
192.168.0.3
192.168.0.4
192.168.0.5
……
192.168.0.248
192.168.0.249

192.168.0.250
192.168.0.251
192.168.0.252
192.168.0.253
192.168.0.254
192.168.0.255

A continuación, realizamos un Ping a un host específico:

ping1.py

```
#programa: Ping1.py
#autor: jorge nolasco valenzuela
#fecha: 23-12-2021
“””
descripcion: este programa muestra
el uso de ping
“””
import os
respuesta = os.popen(‘ping -n 1 192.168.1.1’)
for line in respuesta.readlines():
 print(line)
```

Estadísticas de ping para 192.168.1.1:
Paquetes: enviados = 1, recibidos = 1, perdidos = 0
(0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
Mínimo = 1ms, M ximo = 1ms, Media = 1ms

A continuación, muestra las direcciones IP activas

Scaneo1.py

```
#programa: Scaneo1.py
#autor: jorge nolasco valenzuela
#fecha: 23-12-2021
“””
descripcion: este programa muestra
el uso de ping
“””
import os
from datetime import datetime
red = input(“IP de la Red :”)
red1= red.split(“.”)
net2 = red1[0]+”.”+red1[1]+”.”+red1[2]+”.”
st1 = int(input(“Primer Numero de la Red :”))
en1 = int(input(“Ultimo Numero de la Red :”))
en1=en1+1
ping1 = “ping -n 1 “
inicio= datetime.now()
```

```
print("Scaneo....")
for ip in range(st1,en1):
 addr = net2+str(ip)
 comm = ping1+addr
 response = os.popen(comm)
 for line in response.readlines():
  if(line.count("TTL")):
   break
  if (line.count("TTL")):
   print(addr, " :Activos")
fin= datetime.now()
total =fin-inicio
print("Finalizacion del Scaneo en ", total)
```

IP de la Red :192.168.1.0
Primer Número de la Red :2
Último Número de la Red :50
Scaneo.
192.168.1.33 :Activos
192.168.1.34 :Activos
192.168.1.40 :Activos
192.168.1.42 :Activos
192.168.1.45 :Activos
Finalización de Scaneo en 0:01:14.004394

8.4.11 Búsqueda e Indexación

Archivo.txt
Adán
AGUSTIN
Alberto
Alejandro
Alfonso
Alfredo
Andres
Antonio
Armando
Arturo
Benito
Benjamín
Bernardo
Carlos
Cesar
Claudio
Clemente
Cristian
Cristóbal

Daniel
David
Diego
Eduardo
Emilio
Enrique
Ernesto
Esteban
Federico
Felipe
Fernando
Francisco
Gabriel
Gerardo
German
Gilberto
Gonzalo
Gregorio
Guillermo
Gustavo
Hernan
Homero
Horacio
Hugo

Primero comenzaremos creando un programa que realice la búsqueda de una palabra en un archivo específico:

Busqueda1.py

```
#programa: Busqueda1.py
#autor: jorge nolasco valenzuela
#fecha: 23-12-2021
“””
descripcion: este programa muestra
la busqueda de Nombres en una Archivo
“””
import sys
#creamos un conjunto vacio llamado Nombres
Nombres=set()
try:
 #Abrir el archivo: archivo.txt
 Fichero=open(‘archivo.txt’)
 #adicionar linea a linea el contenido del archivo al conjunto Nombres
 for line in Fichero:
 Nombres.add(line.strip())
except:
 print(“Error Fichero”)
 sys.exit()
#ingresar Nombre a Buscar
```

```
palabra=input("Ingrese Nombre a Buscar:")
#Buscar Nombre
if (palabra in Nombres):
 print("Nombre Encontrado")
else:
 print("Nombre Encontrado")
```

Ingrese Nombre a Buscar: Luis
Nombre Encontrado

Ahora un ejemplo de lectura de un archivo binario:

Busqueda2.py

```
#programa: Busqueda2.py
#autor: jorge nolasco valenzuela
#fecha: 23-12-2021
"""
descripcion: este programa muestra
la lectura de archivos binarios
"""
import sys
#creamos un conjunto vacio llamado Nombres
Nombres=set()
try:
 #Abrir el archivo: archivo.bin
 Fichero=open('archivo.bin','rb')
 #cargar el archivo binario en un bytearray
 binario=bytearray(Fichero.read())
except:
 print("Error Fichero")
 sys.exit()
print("Archivo:",binario)
```

Archivo:
bytearray(b'Ad\xe1n\r\nAgust\xedn\r\nAlberto \r\nAlejandro\r\nAlfonso\r\nAlfredo\r\nAndr\xe9s\r\nAntonio \r\nArmando\r\nArturo \r\nBenito\r\nBenjam\xedn\r\nBernardo\r\nCarlos \r\nC\xe9sar\r\nClaudio\r\nClemente \r\nCristian\r\nCristobal\r\nDaniel \r\nDavid\r\nDiego\r\nEduardo \r\nEmilio\r\nEnrique \r\nErnesto\r\nEsteban\r\nFederico\r\nFelipe\r\nFernando \r\nFrancisco \r\nGabriel \r\nGerardo \r\nGerm\xe1n\r\nGilberto\r\nGonzalo\r\nGregorio\r\nGuillermo \r\nGustavo\r\nHern\xe1n\r\nHomero\r\nHoracio\r\nHugo\r\nIgnacio\r\nJacobo\r\nJaime\r\nJavier\r\nJer\xf3nimo\r\nJes\xfas\r\nJoaqu\xedn\r\nJorge\r\nJorge Luis\r\nJos\xe9 \r\nJos\xe9 Eduardo\r\nJos\xe9 Emilio\r\nJos\xe9 Luis\r\nJos\xe9 Mar\xeda\r\nJuan \r\nJuan Carlos\r\nJulio\r\nJulio C\xe9sar\r\nLorenzo\r\nLucas\r\nLuis \r\nLuis Miguel\r\nManuel \r\nMarco Antonio\r\nMarcos\r\nMariano\r\nMario\r\nMart\xedn\r\nMateo\r\nMiguel\r\nMiguel \xc1ngel\r\nNicol\xe1s (Nico)\r\nOctavio\r\n\xd3scar\r\nPablo\r\nPatricio\r\nPedro\r\nRafael (Rafa)\r\nRamiro\r\nRam\xf3n\r\nRa\xfal\r\nRicardo \r\nRoberto \r\nRodrigo \r\nRub\xe9n\r\nSalvador\r\nSamuel\r\nSancho\r\nSantiago \r\nSergio\r\nTeodoro\r\nTimoteo \r\nTom\xe1s \r\nVicente\r\nV\xedctor\r\nAdela\r\nAdriana\r\nAlejandra\r\nAlicia\r\nAmalia\r\nAna\r\nAna Luisa\r\nAna Mar\xeda\r\nAndrea\r\nAnita\r\n\xc1ngela\r\nAn-

```
tonia (Toni)\r\nBarbara\r\nBeatriz\r\nBerta\r\nBlanca\r\nCaridad\r\nCarla\r\nCarlota\r\
nCarmen\r\nCarolina (Caro)\r\nCatalina (Cata)\r\nCecilia (Ceci)\r\nClara\r\nClaudia\r\nCon-
cepci\xf3n\r\n(Concha, Conchita)\r\nCristina (Cris, Tina)\r\nDaniela\r\nD\xe9bora\r\nDiana\r\
nDolores (Lola)\r\nDorotea (Dora)\r\nElena\r\nElisa\r\nEloisa\r\nElsa\r\nElvira\r\nEmilia
(Emi)\r\nEsperanza\r\nEstela\r\nEster\r\nEva\r\nFlorencia\r\nFrancisca (Paca,\r\nPaquita)\r\
nGabriela (Gabi)\r\nGloria\r\nGraciela (Chela)\r\nGuadalupe (Lupe)\r\nGuillermina\r\nIn\
xe9s\r\nIrene\r\nIsabel (Chabela,\r\nChavela, Isa)\r\nIsabela\r\nJosefina (Pepita)\r\nJuana\r\
nJulia\r\nLaura\r\nLeonor\r\nLeticia (Leti)\r\nLilia\r\nLorena\r\nLourdes\r\nLucia\r\nLuisa\r\
nLuz\r\nMagdalena\r\nManuela\r\nMarcela (Chela)\r\nMargarita (Rita)\r\nMar\xeda\r\nMar\
xeda del Carmen\r\nMar\xeda Cristina\r\nMar\xeda Elena\r\nMar\xeda Eugenia\r\nMar\xeda
Jos\xe9 (Marij\xf3)\r\nMar\xeda Luisa\r\nMar\xeda Soledad\r\nMar\xeda Teresa (Maite,\r\
nMarite)\r\nMariana\r\nMaricarmen\r\nMarilu\r\nMarisol\r\nMarta\r\nMercedes (Meche)\r\
nMicaela\r\nM\xf3nica (Moni)\r\nNatalia (Nati)\r\nNorma\r\nOlivia\r\nPatricia (Pati)\r\nPilar
(Pili)\r\nRamona\r\nRaquel\r\nRebeca\r\nReina\r\nRocio\r\nRosa (Rosi, Rosita)\r\nRosalia\r\
nRosario\r\nSara (Saruca)\r\nSilvia\r\nSofia\r\nSoledad (Sole)\r\nSonia\r\nSusana (Susa,\r\
nSusanita)\r\nTeresa (Tere)\r\nVer\xf3nica (Vero)\r\nVictoria (Vicki)\r\nVirginia\r\nYolanda
(Yoli)')
```

8.4.12 Recolección de información

Analizando dominios–Módulo Whois

Módulo de Python para recuperar información de dominios.

La información que suministra whois de un dominio proporciona diversos detalles como registrador, propietario, fecha de registro, fecha de caducidad, etc.

Si queremos instalar el módulo específico:

```
pip install python-whois
```

A continuación, un ejemplo para obtener información de un dominio específico:

whois1.py

```
#programa: whois1.py
#autor: jorge nolasco valenzuela
#fecha: 23-12-2021
“””
descripcion: este programa muestra
el uso de whois para analizar dominios
“””
import whois
#analizaremos el dominio gloria
dominio = whois.whois(‘www.gloria.com.pe’)
print(dominio)
```

```
{
“domain_name”: “gloria.com.pe”,
“registrar”: “NIC .PE”,
“whois_server”: “NIC .PE”,
“referral_url”: null,
“updated_date”: null,
“creation_date”: null,
“expiration_date”: null,
“name_servers”: [
“ns1.grupogloria.com”,
“ns2.grupogloria.com”
],
“status”: “ok”,
“emails”: “gtalavera@gloria.com.pe”,
“dnssec”: “unsigned”,
“name”: “gloria s.a.”,
“org”: null,
“address”: null,
“city”: null,
“state”: null,
“zipcode”: null,
“country”: null
}
```


Analizando dominios–Módulo DNS

Dnspython es una un conjunto de herramientas de DNS para Python. Soporta casi todos los tipos de registros. Se puede utilizar para consultas, transferencias de zona y actualizaciones dinámicas. Soporta mensajes TSIG autenticados y EDNS0. Dnspython proporciona acceso de alto y bajo nivel al DNS. Las clases de alto nivel realizan consultas para datos de un nombre, tipo y clase, y devuelven un conjunto de respuestas. Las clases de bajo nivel permiten la manipulación directa de zonas DNS, mensajes, nombres y registros.

A continuación, un ejemplo para obtener información de un dominio específico:

dns1.py

```
#programa: dns1.py
#autor: jorge nolasco valenzuela
#fecha: 23-12-2021
“””
descripcion: este programa muestra
el uso de dns para analizar dominios
“””
import dns.query
qname = dns.name.from_text(‘facebook.com’)
consulta = dns.message.make_query(qname, dns.rdatatype.NS)
```

```
print("="*30)
print("Resultado de la Consulta:")
print(consulta)
print("="*30)
```

```
==============================
Resultado de la Consulta:
id 13208
opcode QUERY
rcode NOERROR
flags RD
;QUESTION
facebook.com. IN NS
;ANSWER
;AUTHORITY
;ADDITIONAL
==============================
```

Geolocalización–Geocoder

Módulo de geolocalización escrita en Python.

Aquí está un ejemplo típico de recuperar latitud y longitud de Google usando Python:

geocoder1.py

```
#programa: geocoder1.py
#autor: jorge nolasco valenzuela
#fecha: 23-12-2021
"""
descripcion: este programa muestra
el uso de requests
para geolocalizacion
"""
import requests
url = 'https://maps.googleapis.com/maps/api/geocode/json'
parametros = {'sensor': 'false', 'address': 'Mountain View, CA'}
r = requests.get(url, params=parametros)
resultado = r.json()['results']
location = resultado[0]['geometry']['location']
location['lat'], location['lng']
print("="*45)
print(location)
print("="*45)
```

```
=============================================
{'lat': 37.3860517, 'lng': -122.0838511}
=============================================
```

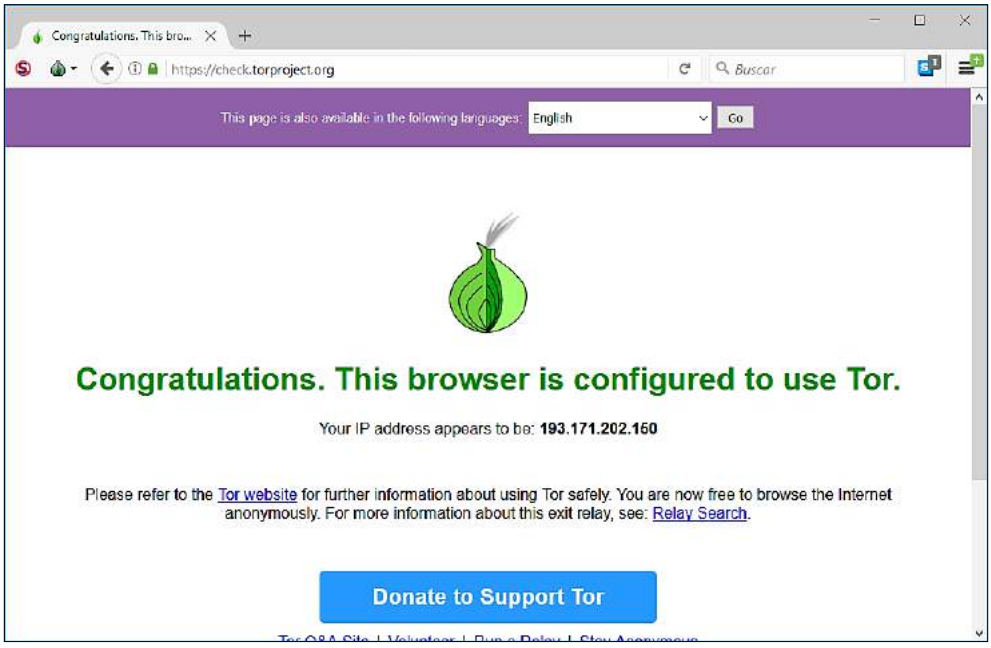
A continuación, para poder realizar múltiples procesos tenemos que hacer uso de hilos, hacer varias tareas a la vez, la librería usada se le denomina thread, es utilizado para procesos pesados que pueden interrumpir el flujo de la aplicación.

Hilos1.py

```
#programa: Hilos1.py
#autor: Jorge Nolasco valenzuela
#fecha: 23-12-2021
“””
descripcion: este programa muestra
el uso de hilos
“””
import _thread
import time
def miHilo(mensaje):
 input(mensaje)
 print(“presionastes enter”)
_thread.start_new_thread(miHilo,(“presione enter”,))
print(“El input ya no interrumpe la ejecucion”)
time.sleep(10)
```

```
El input ya no interrumpe la ejecucion
presione enter
```

Otra manera es ingresar a la siguiente URL:



9.9 CONTROLANDO UNA INSTANCIA LOCAL TOR

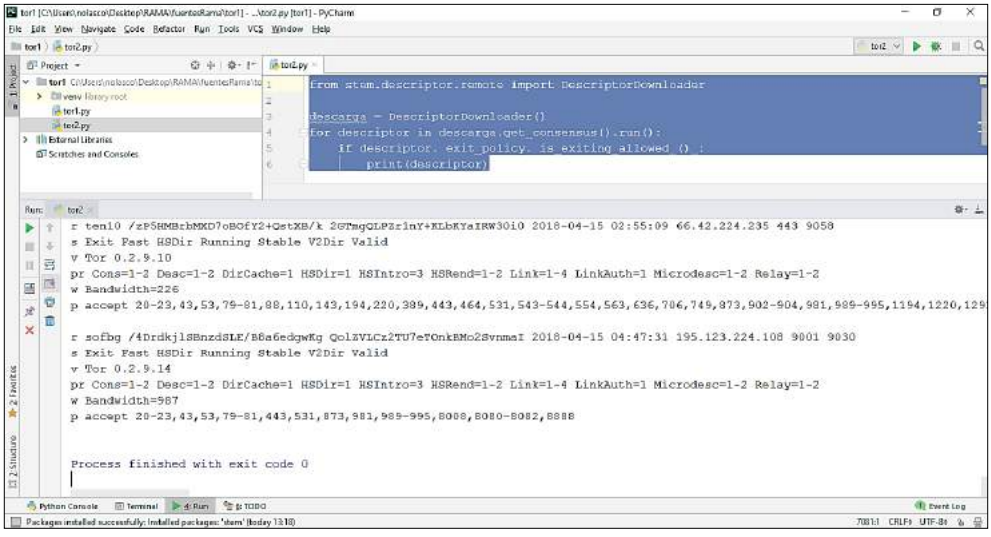
Antes de iniciar una instancia local de **TOR** (cliente) es necesario verificar el valor de la propiedad "**ControlPorf**' ya que dicho valor, indica el puerto utilizado para que rutinas externas puedan controlar la instancia local de **TOR.**

```
from stem.control import Controller
import getpass
#solicita una contraseña
clave=getpass.getpass('Controller password pass:')
#Ahora habilita ControlPort for Tor para escuchar los comandos.
controller=Controller.from_port(port=9151)
#autentificacion en una instancia de TOR
controller.authenticate(clave)
print(controller.get_info("address")) #consulta claves disponibles
```

9.10 INFORMACIÓN DE REPETIDORES DISPONIBLES

Stem puede ser útil para controlar una instancia de **TOR** desde una rutina de código externo ,pero también cuenta con algunas utilidades que le permiten a un atacante consultar los descriptores. Dichos documentos contienen mucha información sobre los repetidores que conforman la red de **TOR** y dicha información puede utilizarse para realizar ataques dirigidos contra cualquier repetidor en la red. La información que puede recuperarse de los descriptores emitidos, contiene datos sobre la versión del sistema operativo del repetidor, Ancho de banda aportado, **Nickname, Fingerprint,** dirección **IP,** entre otros datos que pueden resultar bastantes informativos y reveladores para un atacante. Uno de los mecanismos más sencillos para obtener los últimos descriptores generados es utilizando la clase **stem.descriptor.remote. DescriptionDownloader.**

```
from stem.descriptor.remote import DescriptorDownloader

descarga = DescriptorDownloader()
for descriptor in descarga.get_consensus().run():
 if descriptor. exit_policy. is_exiting_allowed () :
  print(descriptor)
```

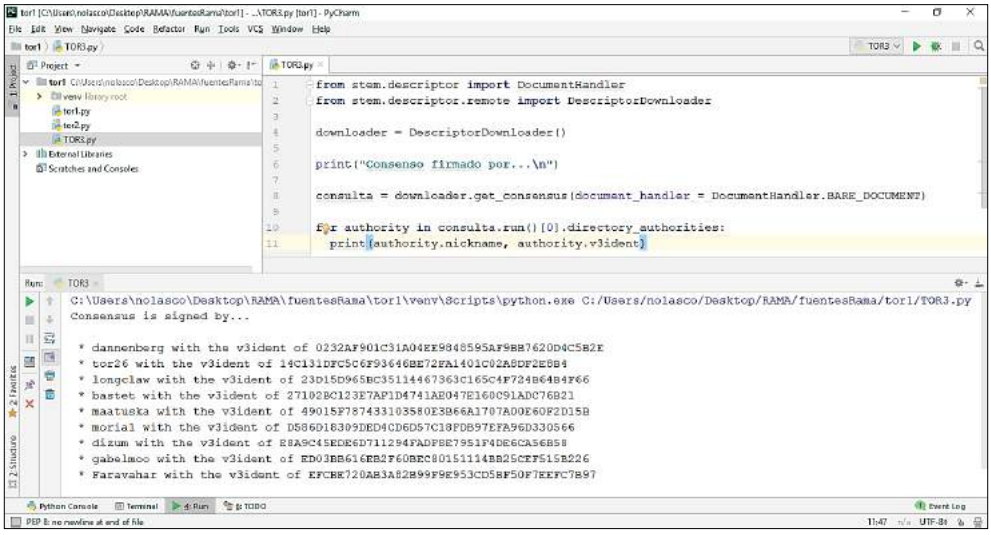


IDE: Pycharm

9.11 INFORMACIÓN DE AUTORIDADES DE DIRECTORIO

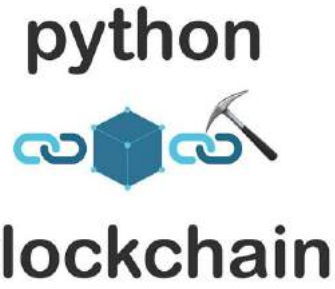
Cuando una aplicación necesita conectarse a la red Tor, primero busca un retransmisor de salida con las políticas adecuadas a la conexión que desea realizar. Esto lo hace mediante las Autoridades de Directorio, que guardan dicha y otra información referida a los Retransmisores.

```
from stem.descriptor import DocumentHandler
from stem.descriptor.remote import DescriptorDownloader

downloader = DescriptorDownloader()

print("Consenso firmado por...\n")

consulta = downloader.get_consensus(document_handler = DocumentHandler.BARE_DO-
CUMENT)

for authority in consulta.run()[0].directory_authorities:
 print(authority.nickname, authority.v3ident)
```



IDE: Pycharm

11.13 CONSTRUCCIÓN DE UNA CADENA DE BLOQUES –PYTHON



Es importante saber cómo funciona la cadena de bloqueo Hashing. La tecnología Blockchain es uno de los descubrimientos más innovadores y definidores de la época del siglo pasado. Al ver la influencia que ha tenido en los últimos años y el impacto que tendrá en el futuro, seguramente no es una exageración decir eso. Para entender cómo funcionan varias criptomonedas como Ethereum y Bitcoin.

En términos simples, Hashing significa tomar una cadena de entrada de cualquier longitud y producir una salida de una longitud fija. En el contexto de las criptomonedas como Bitcoin, las transacciones se toman como entrada y se ejecutan a través de un algoritmo hash (Bitcoin usa SHA-256) que da una salida de una longitud fija.

Veamos cómo funciona el proceso de hash. Vamos a poner en ciertas entradas. Para este ejercicio, vamos a usar el SHA-256 (Secure Hashing Algorithm 256).

Para este ejemplo, estoy usando la distribución Anaconda Python 3. Como la mayoría de las cosas en Python, crear un hash es tan simple como importar una biblioteca que alguien ya ha creado para nosotros. En este caso, esa biblioteca es: hashlib. Entonces, nuestro primer paso es importar hashlib.

```
Importar hashlib
```

Ahora tomemos un momento para aprender la sintaxis requerida para crear un hash criptográfico con hashlib. En este ejemplo, estoy usando el algoritmo de hash SHA 256. Estoy usando esto porque es el mismo algoritmo utilizado por Bitcoin.

Aquí está la sintaxis utilizada.

```
hashlib.sha256 (string.encode()).hexdigest()
```

Entonces, en el siguiente ejemplo, puedes ver que asigné la variable nombre a la cadena 'pedro'. Luego pasé nombre a nuestra función hash. El resultado se puede ver a continuación.

```
import hashlib
nombre=”jorge”
print(hashlib.sha256(nombre.encode()).hexdigest())
```

```
ee5cd7d5d96c8874117891b2c92a036f96918e66c102bc
698ae77542c186f981
```

A continuación, colocaremos el código en una función.

```
import hashlib
def hash(cadena):
 print(hashlib.sha256(cadena.encode()).hexdigest())

if __name__ == '__main__':
 hash("Jorge")
```

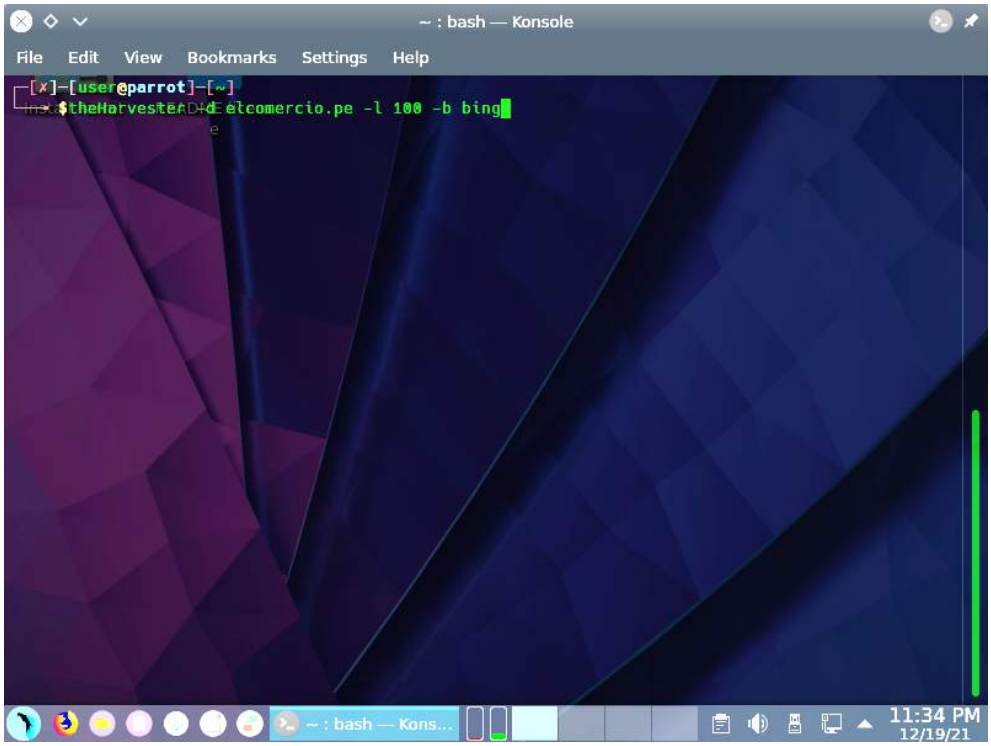
```
6411c2711069573d0fc030e80b91f72b40c95cd9fc8c247c
177dbfaa63daf1cd
```

Ahora el código para la creación de bloques:

```
import hashlib
import time
class Bloque:
 def __init__(self):
 self.nombre_bloque = ""
 self.id = 0
 self.hora = time.strftime("%H:%M:%S")
 self.datos = ""
 self.hash_anterior=0
 self.siguiente=""
 self.hash = ""
 def hash_bloque(self):
 sha=(str(self.id) + str(self.hora) + str(self.datos) + str(self.hash_anterior))
 return (hashlib.sha256(sha.encode()).hexdigest())
class Intercambio:
 bloque = Bloque()
 hora = time.strftime("%H:%M:%S")
 data=input("Ingrese Datos :")
 bloque.nombre_bloque ="Inkadroid"
 bloque.id =bloque.id+1
 bloque.hora = hora
 bloque.datos = data
 bloque.hash_anterior = bloque.hash_anterior
 bloque.siguiente = bloque.hash_bloque()
 bloque.hash = bloque.siguiente
 print("Bloque :",bloque.nombre_bloque)
 print("Id Bloque :",bloque.id)
 print("Transaccion :",bloque.hora)
 print("Data :",bloque.datos)
 print("hash anterior :",bloque.hash_anterior)
 print("bloque.siguiente",bloque.siguiente)
 print("hash",bloque.hash)

if __name__ == '__main__':
 intercambio = Intercambio()
```

Uso: TheHarvester

```
-d: Dominio para buscar o nombre de la empresa
 -b: fuente de datos: baidu, bing, bingapi, dogpile, google, googleCSE,
     googleplus, google-profiles, linkedin, pgp, twitter,
     vhost, virustotal, amenazacrowd, crtsh, netcraft, yahoo, all
 -s: comienza en el número de resultado X (predeterminado: 0)
 -v: verifica el host nombre mediante resolución dns y búsqueda de hosts virtua-
les
 -f: guarde los resultados en un archivo HTML y XML (ambos)
 -n: realice una consulta inversa DNS en todos los rangos descubiertos
 -c: realiza una fuerza bruta de DNS para el nombre de dominio
 -t: realiza un descubrimiento de expansión de TLD de DNS
 -e: usa este servidor DNS
 -p: escanea el puerto de los hosts detectados y comprueba las adquisiciones
(80,443,22,21,8080)
 - l: limite la cantidad de resultados para trabajar (bing va de 50 a 50 resul-
tados,
     google 100 a 100 y pgp no usa esta opción)
 -h: usa la base de datos SHODAN para consultar hosts descubiertos
Ejemplos:
 theharvester -d microsoft. com -l 500 -b google -h myresults.html
 theharvester -d microsoft.com -b
 pgp theharvester -d microsoft -l 200 -b linkedin
 theharvester -d apple.com -b googleCSE -l 500 -s 300
```



Instalación

```
sudo apt-get install nmap
```

Identificación del Host

```
nmap -h
nmap -sn [Dirección_IP]
nmap -n -sn 192.168.0.0/24
```

La opción "-sn" le indica a nmap a no realizar un escaneo de puertos después del descubrimiento del host, y solo imprimir los hosts disponibles que respondieron al escaneo.

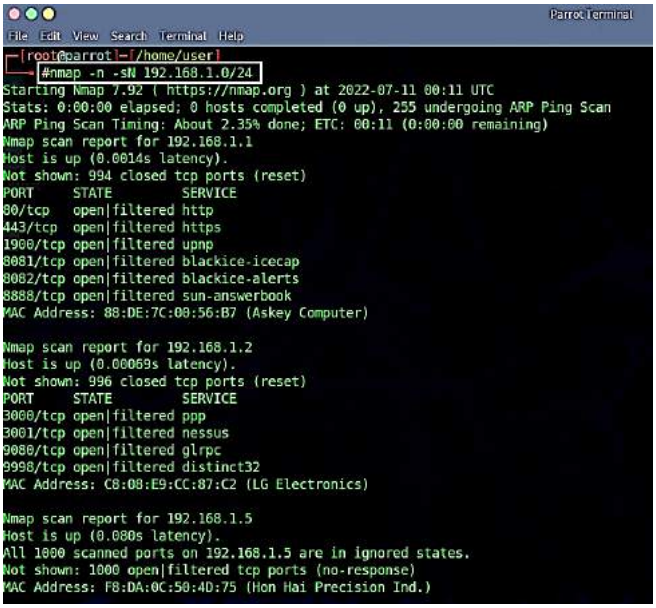
La opción "-n" le indica a nmap a no realizar una resolución inversa al DNS sobre las direcciones IP activas que encuentre.

> **NOTA**
>
> Cuando un usuario privilegiado intenta escanear objetivos sobre una red ethernet local, se utilizan peticiones ARP a menos que sea especificada la opción "—send-ip", la cual indica a nmap a enviar paquetes mediante sockets IP en bruto en lugar de tramas ethernet de bajo nivel.

La opción "p" puerto o rangos específicos.

Reconocimiento de Host Disponibles. No hacer DNS o resolución local (-n), Envía TCP con todos los flags a 0(-sN)

```
nmap -n -sN 192.168.0.0/24
```

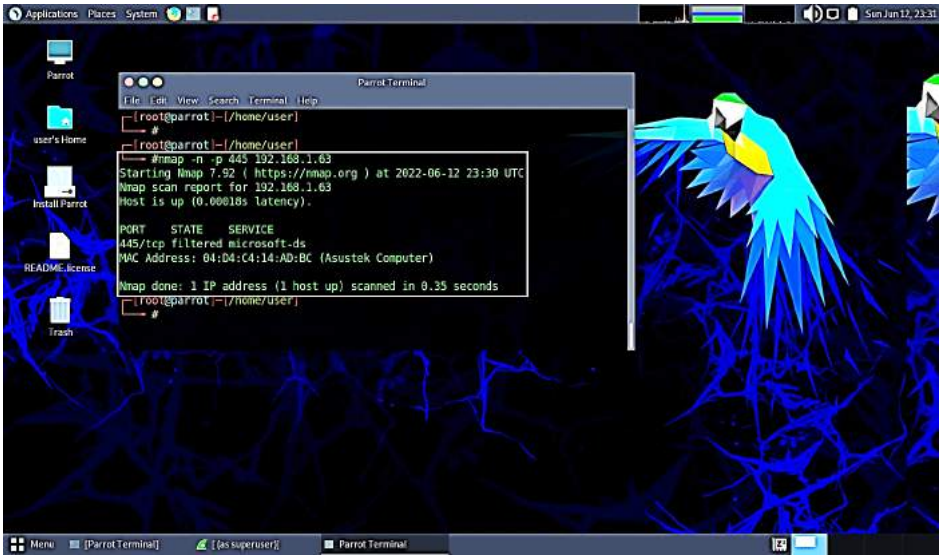


Barrido de todas las IPs de la red 192.168.1.0 y todos sus puertos

Reconocimiento de un Puerto Específico. No hacer DNS o resolución local (-n), Escanear solo puertos específicos (-p)

Observando el estado del puerto 445

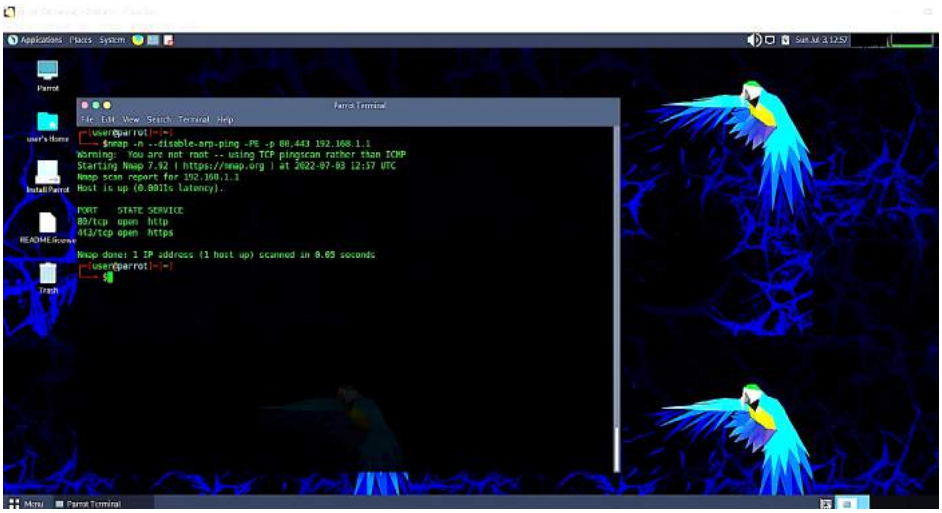
```
nmap -n -p 445 192.168.1.63
```



Barrido de todas las IPs de la red 192.168.1.0 y todos sus puerto

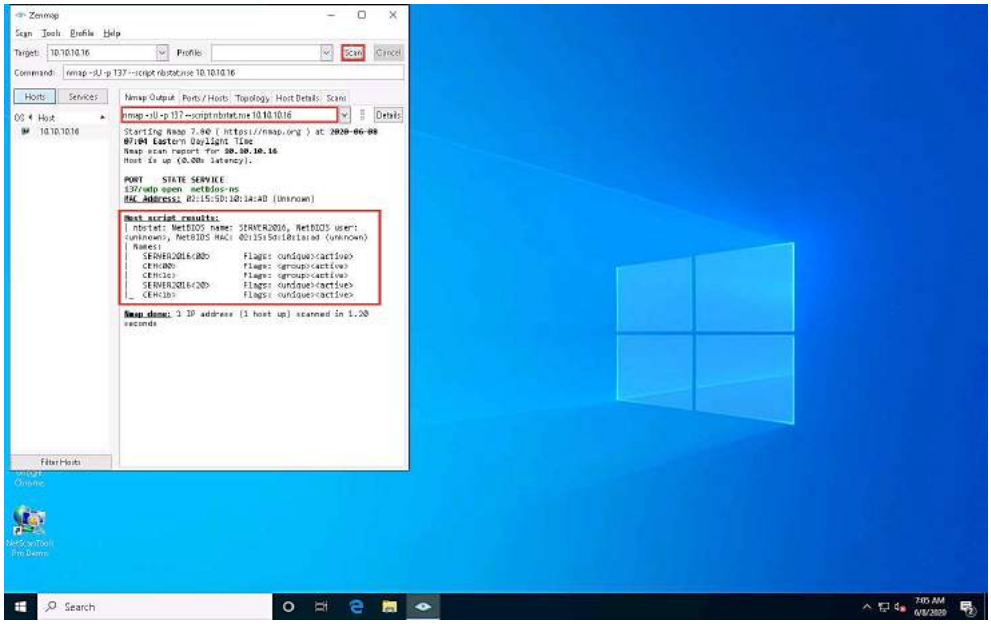
Reconocimiento de un Puerto Específico. No hacer DNS o resolución local (-n), Recomendamos el reconocimiento de puerto sin el protocolo ARP (para evitar no generar señal no deseada), los puertos: 80,443

```
Nmap –n -disable-arp-ping -PE 80,443 192.168.1.1
```



nmap -sU -p 137 -script nbstat.nse 10.10.10.16

- -sU realiza un análisis UDP,
- -p especifica el puerto que se analizará y
- —script nbtstat.nse realiza la enumeración NetBIOS.



También se pueden utilizar otras herramientas para realizar la enumeración NetBIOS en la red de destino, como Global Network Inventory (http://www.magnetosoft.com), Advanced IP Scanner (http://www.advanced-ip-scanner.com), Hyena (https://www.systemtools.com) y Nsauditor Network Security Auditor (https://www.nsauditor.co

En Parrot o Kali

Escaneo SNMP

```
nmap -sU -p 161 10.10.10.16
```

- -sU realiza un escaneo UDP y
- -p especifica el puerto a escanear.

El comando snmp-check enumera la máquina de destino, enumerando información confidencial como información del sistema y cuentas de usuario.