Proyecto 0 "Covid-19 en México"

Tuesday, May 5, 2020 4:24 PM

En este proyecto se busca el obtener un modelo de machine learning que pronostique la sobrevivencia de un paciente con enfermedades crónicas degenerativas

Para comenzar se deben descargar los archivos de la página: https://www.kaggle.com/lalish99/covid19-mx, estos archivos cambian conforme se actualiza la información

En total se encuentran 1 elementos cuyos nombres son:

- casos_confirmados.csv
- covid_mx.csv
- covid-19_general_MX.csv
- ENTIDADES.csv
- NACIONALIDAD.csv
- ORIGEN.csv
- RESULTADO.csv
- SECTOR.csv
- SEXO.csv
- SI_NO.csv
- TIPO_PACIENTE.csv

El primer archivo ENTIDADES.csv contiene 3 columnas, la primera es la clave de entidad, la segunda la entidad y por último la abreviatura

	A	B	C	D	E	F	G	H	I
1	CLAVE_ENTID	ENTIDAD_FEDERATIVA	ABREVIATURA						
2	1	AGUASCALIENTES	AS	16	MICHOACÃN DE OCAMPO	MN	31	YUCATÃN	YN
3	2	BAJA CALIFORNIA	BC	17	MORELOS	MS	32	ZACATECAS	ZS
4	3	BAJA CALIFORNIA SUR	BS	18	NAYARIT	NT	36	ESTADOS UNIDOS MEXICANOS	EUM
5	4	CAMPECHE	CC	19	NUEVO LEÃ"N	NL	97	NO APLICA	NA
6	5	COAHUILA DE ZARAGOZA	CL	20	OAXACA	OC	98	SE IGNORA	SI
7	6	COLIMA	CM	21	PUEBLA	PL	99	NO ESPECIFICADO	NE
8	7	CHIAPAS	CS	22	QUERÃ‰TARO	QT			
9	8	CHIHUAHUA	CH	23	QUINTANA ROO	QR			
10	9	CIUDAD DE MÃ‰XICO	DF	24	SAN LUIS POTOSÃ	SP			
11	10	DURANGO	DG	25	SINALOA	SL			
12	11	GUANAJUATO	GT	26	SONORA	SR			
13	12	GUERRERO	GR	27	TABASCO	TC			
14	13	HIDALGO	HG	28	TAMAULIPAS	TS			
15	14	JALISCO	JC	29	TLAXCALA	TL			
16	15	MÃ‰XICO	MC	30	VERACRUZ DE IGNACIO DE LA LLAVE	VZ			

El segundo archivo NACIONALIDAD.csv contiene 2 columnas, la primera respecto a la clave y la segunda la descripción de la misma todo sobre su nacionalidad

	A	B
1	CLAVE	DESCRIPCIÃ"N
2	1	MEXICANA
3	2	EXTRANJERA
4	99	NO ESPECIFICADO

El tercer archivo ORIGEN.csv contiene 2 columnas, la primera respecto a la clave y la segunda la descripción de la misma acerca de las unidades monitoras de enfermedad respiratoria viral

	A	B
1	CLAVE	DESCRIPCIÃ"N
2	1	USMER
3	2	FUERA DE USMER
4	99	NO ESPECIFICADO

El cuarto archivo RESULTADO.csv contiene 2 columnas, la primera respecto a la clave y la segunda la descripción de la misma, da referencia a si una persona tiene o no la enfermedad

	A	B
1	CLAVE	DESCRIPCIÃ"N
2	1	Positivo SARS-CoV-2
3	2	No positivo SARS-CoV-2
4	3	Resultado pendiente

El quinto archivo SECTOR.csv contiene 2 columnas, la primera respecto a la clave y la segunda la descripción de la misma, indica en qué tipo de hospital se encuentra

	A	B
1	CLAVE	DESCRIPCIÃ“N
2	1	CRUZ ROJA
3	2	DIF
4	3	ESTATAL
5	4	IMSS
6	5	IMSS-BIENESTAR
7	6	ISSSTE
8	7	MUNICIPAL
9	8	PEMEX
10	9	PRIVADA
11	10	SEDENA
12	11	SEMAR
13	12	SSA
14	13	UNIVERSITARIO
15	99	NO ESPECIFICADO

El sexto archivo SEXO.csv contiene 2 columnas, la primera respecto a la clave y la segunda la descripción de la misma, indica el sexo de la persona

	A	B
1	CLAVE	DESCRIPCIÃ“N
2	1	MUJER
3	2	HOMBRE
4	99	NO ESPECIFICADO

El séptimo archivo SI_NO.csv contiene 2 columnas, la primera respecto al _NO. y la segunda la descripción de la misma, se desconoce que indica

	A	B
1	_NO.	DESCRIPCIÃ“N
2	1	SI
3	2	NO
4	97	NO APLICA
5	98	SE IGNORA
6	99	NO ESPECIFICADO

El octavo archivo TIPO_PACIENTE.csv contiene 2 columnas, la primera respecto a la clave y la segunda la descripción de la misma, indica el estado del paciente

	A	B
1	CLAVE	DESCRIPCIÃ“N
2	1	AMBULATORIO
3	2	HOSPITALIZADO
4	99	NO ESPECIFICADO

El noveno archivo covid_mx.csv contiene 3 columnas, casos confirmados, fallecidos y la fecha

	A	B	C
1	Confirmed Cases	Deceased	Dates
2	3	0	28-Feb-20
3	4	0	29-Feb-20
4	5	0	1-Mar-20
5	5	0	2-Mar-20
6	5	0	3-Mar-20
7	5	0	4-Mar-20
8	5	0	5-Mar-20
9	6	0	6-Mar-20
10	6	0	7-Mar-20
11	7	0	8-Mar-20
12	7	0	9-Mar-20
13	7	0	10-Mar-20
14	11	0	11-Mar-20
15	15	0	12-Mar-20

El décimo archivo casos_confirmados.csv contiene 5 columnas, Entidad, Sexo, Edad, Dia, Confirmación

	A	B	C	D	E	F
1		State	Sex	Age	Date	Confirmed
2	0	sinaloa	MASCULINO	42	4/6/2020	1
3	1	mÃ©xico	FEMENINO	61	4/10/2020	1
4	2	tabasco	FEMENINO	32	4/13/2020	1
5	3	baja californ	FEMENINO	33	4/13/2020	1
6	4	tabasco	MASCULINO	63	4/13/2020	1
7	5	sinaloa	MASCULINO	45	4/14/2020	1
8	6	sinaloa	FEMENINO	50	4/14/2020	1
9	7	tabasco	FEMENINO	48	4/17/2020	1
10	8	tabasco	FEMENINO	46	4/22/2020	1
11	9	sinaloa	MASCULINO	47	4/23/2020	1
12	10	chihuahua	FEMENINO	55	4/23/2020	1
13	11	sinaloa	MASCULINO	32	4/27/2020	1
14	12	baja californ	MASCULINO	34	4/27/2020	1
15	13	baja californ	FEMENINO	33	4/23/2020	1

Por último el archivo covid-19_general_MX.csv contiene las siguientes columnas

- SECTOR
- ENTIDAD_UM
- SEXO
- ENTIDAD_RES
- TIPO_PACIENTE
- FECHA_INGRESO
- FECHA_SINTOMAS
- FECHA_DEF
- INTUBADO
- NEUMONIA
- EDAD
- NACIONALIDAD
- DIABETES
- EPOC
- ASMA
- INMUSPR
- HIPERTENSION
- OTRA_CON
- CARDIOVASCULAR
- OBESIDAD
- RENAL_CRONICA
- TABAQUISMO
- OTRO_CASO
- RESULTADO
- UCI

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		SECTOR	ENTIDAD_UM	SEXO	ENTIDAD_	TIPO_PAC	FECHA_INGF	FECHA_SINTOMAS	FECHA_DEF	INTUBADO	NEUMONI	EDAD	NACIONA
2	0	3	25	2	25	2	4/6/2020	3/31/2020	4/10/2020	1	1	42	1
3	1	3	15	1	15	1	4/10/2020	4/5/2020	9999-99-99	97	2	61	1
4	2	3	27	1	27	1	4/13/2020	4/11/2020	9999-99-99	97	2	32	1
5	3	3	2	1	2	1	4/13/2020	4/9/2020	9999-99-99	97	2	33	1
6	4	3	27	2	27	2	4/13/2020	4/5/2020	4/14/2020	2	1	63	1
7	5	3	25	2	25	2	4/14/2020	4/4/2020	4/17/2020	1	1	45	2
8	6	3	25	1	25	2	4/14/2020	4/6/2020	9999-99-99	2	1	50	1
9	7	3	27	1	27	1	4/17/2020	4/7/2020	9999-99-99	97	2	48	1
10	8	3	27	1	27	1	4/22/2020	4/17/2020	9999-99-99	97	1	46	1
11	9	3	25	2	25	1	4/23/2020	4/20/2020	9999-99-99	97	1	47	1
12	10	3	8	1	8	1	4/23/2020	4/19/2020	9999-99-99	97	2	55	1
13	11	3	25	2	25	2	4/27/2020	4/25/2020	9999-99-99	2	1	32	1
14	12	3	2	2	2	1	4/27/2020	4/25/2020	9999-99-99	97	2	34	1
15	13	3	2	1	2	1	4/23/2020	4/22/2020	9999-99-99	97	2	33	1

	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	DIABETES	EPOC	ASMA	INMUSUPI	HIPERTEN	OTRA_CC	CARDIOV.	OBESIDAI	RENAL_C	TABAQUI	OTRO_CA	RESULTAI	UCI
	1	2	2	2	1	2	2	1	2	2	2	1	2
	2	2	2	2	1	2	2	2	2	2	1	1	97
	2	2	2	2	2	2	2	2	2	2	2	1	97
	2	2	2	2	2	2	2	2	2	2	1	1	97
	2	2	2	2	1	2	2	1	1	2	2	1	2
	2	2	2	2	2	2	2	2	2	2	2	1	2
	2	2	2	2	2	2	2	1	2	2	2	1	2
	2	2	2	2	2	2	2	2	2	2	1	1	97
	2	2	1	2	2	2	2	2	2	2	1	1	97
	2	2	2	2	2	2	2	2	2	2	99	1	97
	2	2	2	2	2	2	2	2	2	2	1	1	97
	2	2	2	2	2	2	2	2	2	2	99	1	2
	2	2	2	2	2	2	2	1	2	2	99	1	97
	2	2	2	2	2	2	2	1	2	2	1	1	97
	2	2	2	2	2	2	2	2	2	2	1	1	97
	2	2	2	2	2	2	2	2	2	2	1	1	97

Importar Librerías

Para iniciar el proyecto en Python con el IDE Sublime Text 3, debemos importar las siguientes librerías, que se separan en 3 categorías

- Datascience
- Gráficos
- Machine Learning

```
#-----------------------------------
#Importamos las librerias a utilizar
#-----------------------------------

#Librerias de datascience
import numpy as np
import pandas as pd
#Librerias para gráficar
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib
import requests
import plotly.express as px
#Librerias de Machine Learning
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
```

Preparar la Data

Continuamos con la preparación de la Data, primero se declaran 3 variables, 2 son de colores y una de explode, son para la estética de los gráficos

```
#-----------------
#Preparamos la Data
#-----------------
#Estos son los colores a utilizar
colors = ['#b00c00', '#edad5f', '#d69e04', '#b5d902', '#63ba00', '#05b08e', '#128ba6',
    '#5f0da6', '#b30bb0', '#c41484', '#a1183d', '#3859eb', '#4da1bf', '#6bcfb6']
colors2=colors[::-1]
#Para las graficas circulares se observen mucho mejor
explode=(0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1)
```

Proseguimos con la lectura de los siguientes archivos "CSV" y se imprimirá las primeras 5 filas, el primer archivo es de los casos confirmados

```
#Se lee el primer archivo que es el de casos confirmados
casos_confirmados=pd.read_csv('casos_confirmados.csv')
print(casos_confirmados.head())
```

```
   Unnamed: 0           State        Sex  Age        Date  Confirmed
0           0  baja california   FEMENINO   49  2020-03-26          1
1           1         tabasco  MASCULINO   27  2020-04-09          1
2           2  baja california   FEMENINO   28  2020-04-13          1
3           3          méxico   FEMENINO   51  2020-04-14          1
4           4         tabasco  MASCULINO   39  2020-04-14          1
```

El segundo archivo es el de los casos acumulados

```
#Se lee el segundo archivo que es el de casos acumulados
casos_acumulados=pd.read_csv('covid_mx.csv')
print(casos_acumulados.head())
```

```
   Confirmed Cases  Deceased        Dates
0                3         0  28-Feb-2020
1                4         0  29-Feb-2020
2                5         0  01-Mar-2020
3                5         0  02-Mar-2020
4                5         0  03-Mar-2020
```

El tercer archivo es el que contiene la información general de los pacientes del Sars-Cov-2 en México y el total de pruebas realizadas, este es nuestro archivo madre

```
#Se lee el tercer archivo que coniene la descripción de todos los pacientes
covid_MX=pd.read_csv('covid-19_general_MX.csv')
print(covid_MX.head())
```

```
   Unnamed: 0  SECTOR  ENTIDAD_UM  SEXO  ...  TABAQUISMO  OTRO_CASO  RESULTADO  UCI
0           0       3           2     1  ...           2          2          1   97
1           1       3          27     2  ...           2          1          1   97
2           2       3           2     1  ...           2          1          1   97
3           3       3          15     1  ...           2          2          1   97
4           4       3          27     2  ...           2          1          1    1
```

El cuarto archivo se contempla la información de los sectores de salud

```
#Se lee el cuarto archivo que coniene la descripción de todos los sectores
df_sector=pd.read_csv('SECTOR.csv')
print(df_sector.head())
```

```
   CLAVE     DESCRIPCIÓN
0      1       CRUZ ROJA
1      2             DIF
2      3         ESTATAL
3      4            IMSS
4      5  IMSS-BIENESTAR
```

Por último el quinto archivo contiene la descripción de las Entidades Federativas

```
#Se lee el quinto archivo que coniene la descripción de las entidades
df_entidades=pd.read_csv('ENTIDADES.csv')
print(df_entidades.head())
```

```
   CLAVE_ENTIDAD    ENTIDAD_FEDERATIVA ABREVIATURA
0              1        AGUASCALIENTES          AS
1              2       BAJA CALIFORNIA          BC
2              3   BAJA CALIFORNIA SUR          BS
3              4              CAMPECHE          CC
4              5  COAHUILA DE ZARAGOZA          CL
```

Vamos a analizar la información de nuestro archivo madre

```
#Analizamos los datos que tenemos disponibles
print("\nInformación del tipo de dato en el Dataset:")
print(covid_MX.info())
```

```
Data columns (total 26 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Unnamed: 0      148497 non-null  int64
 1   SECTOR          148497 non-null  int64
 2   ENTIDAD_UM      148497 non-null  int64
 3   SEXO            148497 non-null  int64
 4   ENTIDAD_RES     148497 non-null  int64
 5   TIPO_PACIENTE   148497 non-null  int64
 6   FECHA_INGRESO   148497 non-null  object
 7   FECHA_SINTOMAS  148497 non-null  object
 8   FECHA_DEF       148497 non-null  object
 9   INTUBADO        148497 non-null  int64
 10  NEUMONIA        148497 non-null  int64
 11  EDAD            148497 non-null  int64
 12  NACIONALIDAD    148497 non-null  int64
 13  DIABETES        148497 non-null  int64
 14  EPOC            148497 non-null  int64
 15  ASMA            148497 non-null  int64
 16  INMUSUPR        148497 non-null  int64
 17  HIPERTENSION    148497 non-null  int64
 18  OTRA_CON        148497 non-null  int64
 19  CARDIOVASCULAR  148497 non-null  int64
 20  OBESIDAD        148497 non-null  int64
 21  RENAL_CRONICA   148497 non-null  int64
 22  TABAQUISMO      148497 non-null  int64
 23  OTRO_CASO       148497 non-null  int64
 24  RESULTADO       148497 non-null  int64
 25  UCI             148497 non-null  int64
dtypes: int64(23), object(3)
memory usage: 27.8+ MB
None
```

El archivo solo cuenta con 3 columnas de tipo objecto y el resto son de tipo entero, lo cual indica solo

modificación en los de tipo objeto.

Ahora vamos a ver la descripción de las características estadísticas de nuestro archivo

```
#Describimos la información del Dataset
print("\nDescripción de las estadisticas del Dataset:")
print(covid_MX.describe())
```

```
Descripción de las estadisticas del Dataset:
         Unnamed: 0         SECTOR  ...      RESULTADO            UCI
count  148497.000000  148497.000000  ...  148497.000000  148497.000000
mean    74248.000000       9.619851  ...       1.896766      72.526832
std     42867.535799       8.025083  ...       0.653718      41.570480
min         0.000000       1.000000  ...       1.000000       1.000000
25%     37124.000000       4.000000  ...       1.000000       2.000000
50%     74248.000000      12.000000  ...       2.000000      97.000000
75%    111372.000000      12.000000  ...       2.000000      97.000000
max    148496.000000      99.000000  ...       3.000000      99.000000

[8 rows x 23 columns]
```

Ahora verificaremos si contamos con algún dato faltante

```
#Verificamos si hay un dato faltante
print("\nDatos faltantes:")
print(pd.isnull(covid_MX).sum())
```

```
Datos faltantes:
Unnamed: 0        0
SECTOR            0
ENTIDAD_UM        0
SEXO              0
ENTIDAD_RES       0
TIPO_PACIENTE     0
FECHA_INGRESO     0
FECHA_SINTOMAS    0
FECHA_DEF         0
INTUBADO          0
NEUMONIA          0
EDAD              0
NACIONALIDAD      0
DIABETES          0
EPOC              0
ASMA              0
INMUSUPR          0
HIPERTENSION      0
OTRA_CON          0
CARDIOVASCULAR    0
OBESIDAD          0
RENAL_CRONICA     0
TABAQUISMO        0
OTRO_CASO         0
RESULTADO         0
UCI               0
dtype: int64
```

Procesamiento de la Data

Nuestra Data no cuenta con archivos faltantes lo cual es bueno, damos paso al procesamiento de la información para que nuestro análisis sea lo más correcto posible, las instrucciones de las modificaciones se explican en las imágenes del codigo

```
#----------------------
#Procesamiento de datos
#----------------------

#Se modificaran los valores numericos para un mejor manejo de la información

#Se cambiara el sexo de los pasajeros 0 para mujeres y 1 para hombres
covid_MX['SEXO'].replace([1,2],[1,0],inplace=True)

#Se cambiara el tipo de paciente de 0 para ambulatorio y 1 para hospitalizado
covid_MX['TIPO_PACIENTE'].replace([1,2],[0,1],inplace=True)

#Se modificara el estado de intubado 0 es si, 1 es no, 3 no aplica y 5 no especificado
covid_MX['INTUBADO'].replace([1,2,97,99],[0,1,3,5],inplace=True)

#Se modificara el estado de si el paciente cuenta con neumonia 0 es no, 1 es si y 5 no especificado
covid_MX['NEUMONIA'].replace([1,2,99],[0,1,5],inplace=True)
```

Modificamos la información de la edad para manejarla por grupos

```
#Se crean varios grupos segun la edad
#Bandas 0-10,11-20,21-30,31-40,41-50,51-60,61-70,71-80,81-90,91-100 y 101-115
bins=[-1,10,20,30,40,50,60,70,80,90,100,115]
names1=['0 a 10','11 a 20','21 a 30','31 a 40','41 a 50','51 a 60','61 a 70',
'71 a 80','81 a 90','91 a 100','101 a 115']
names=['1','2','3','4','5','6','7','8','9','10','11']
covid_MX['EDAD']=pd.cut(covid_MX['EDAD'],bins,labels=names)
```

```
#Se modificara si cuenta con Diabetes 0 es si, 1 es no y 4 se ignora
covid_MX['DIABETES'].replace([1,2,98],[0,1,4],inplace=True)

#Se modificara si cuenta con enfermedad pulmonar obstructiva crónica 0 es si, 1 es no y 4 se ignora
covid_MX['EPOC'].replace([1,2,98],[0,1,4],inplace=True)

#Se modificara si cuenta con asma 0 es si, 1 es no y 4 se ignora
covid_MX['ASMA'].replace([1,2,98],[0,1,4],inplace=True)

#Se modificara si cuenta con inmusupr 0 es si, 1 es no y 4 se ignora
covid_MX['INMUSUPR'].replace([1,2,98],[0,1,4],inplace=True)

#Se modificara si cuenta con hipertensión 0 es si, 1 es no y 4 se ignora
covid_MX['HIPERTENSION'].replace([1,2,98],[0,1,4],inplace=True)

#Se modificara si cuenta con otra con 0 es si, 1 es no y 4 se ignora
covid_MX['OTRA_CON'].replace([1,2,98],[0,1,4],inplace=True)

#Se modificara si cuenta con enfermedad cardiovascular 0 es si, 1 es no y 4 se ignora
covid_MX['CARDIOVASCULAR'].replace([1,2,98],[0,1,4],inplace=True)

#Se modificara si cuenta con obesidad 0 es si, 1 es no y 4 se ignora
covid_MX['OBESIDAD'].replace([1,2,98],[0,1,4],inplace=True)
```

```
#Se modificara si cuenta con enfermedad renal crónica 0 es si, 1 es no y 4 se ignora
covid_MX['RENAL_CRONICA'].replace([1,2,98],[0,1,4],inplace=True)

#Se modificara si cuenta con tabaquismo 0 es si, 1 es no y 4 se ignora
covid_MX['TABAQUISMO'].replace([1,2,98],[0,1,4],inplace=True)

#Se modificara si cuenta con otro caso 0 es si, 1 es no y 4 se ignora
covid_MX['OTRO_CASO'].replace([1,2,98],[0,1,4],inplace=True)

#Se modificara si cuenta con unidad de cuidados intensivos 0 es si, 1 es no, 3 no aplica y 5 no especificado
covid_MX['UCI'].replace([1,2,97,99],[0,1,3,5],inplace=True)

#Se modificara si cuenta con otro caso 0 es positivo, 1 es negativo y 2 pendiente
covid_MX['RESULTADO'].replace([1,2,3],[0,1,2],inplace=True)
```

Las columnas de tipo objecto son fechas, por lo cual se cambiaran a tipo fecha

```
#Se transforma de objeto fecha de ingreso a fecha
covid_MX["FECHA_INGRESO"] = pd.to_datetime(covid_MX["FECHA_INGRESO"])

#Se transforma de objeto fecha de sintomas a fecha
covid_MX["FECHA_SINTOMAS"] = pd.to_datetime(covid_MX["FECHA_SINTOMAS"])

#Se modifican los valores nulos por una fecha lejana
covid_MX["FECHA_DEF"]=covid_MX["FECHA_DEF"].replace('9999-99-99','31/12/2020')

#Se transforma de objeto fecha de defuncion a fecha
covid_MX["FECHA_DEF"] = pd.to_datetime(covid_MX["FECHA_DEF"])
```

Se va a corregir la información de las Entidades Federativas para el mapa de México

```
#Corrigiendo la información de casos confirmados para el mapa de México
estados_or=['Ciudad De México','Veracruz De Ignacio De La Llave','Michoacán De Ocampo','Coahuila De Zaragoza']
estados_or2=['Ciudad de México','Veracruz','Michoacán','Coahuila']
estados=np.array(list(df_entidades['ENTIDAD_FEDERATIVA']))
df_entidades['ENTIDAD_FEDERATIVA']=[i.title() for i in estados]
df_entidades['ENTIDAD_FEDERATIVA'].replace(estados_or,estados_or2,inplace=True)
```

Se elimina la primera columna de nuestro archivo madre

```
#Ahora eliminaremos la primer columa que no tiene nombre
covid_MX=covid_MX.drop(covid_MX.columns[0],axis=1)
```

Verificamos que la información ya se encuentre de manera correcta

Revisamos las primeras 5 filas de nuestro archivo

```
print(covid_MX.head())
```

```
   SECTOR  ENTIDAD_UM  SEXO  ENTIDAD_RES  ...  TABAQUISMO OTRO_CASO RESULTADO UCI
0       3           2     1            2  ...           1         1         0   2
1       3          27     0           27  ...           1         0         0   2
2       3           2     1            2  ...           1         0         0   2
3       3          15     1           15  ...           1         1         0   2
4       3          27     0           27  ...           1         0         0   0

[5 rows x 25 columns]
```

Que forma tiene nuestro archivo

```
print(covid_MX.shape)
```

```
(148497, 25)
```

Contiene 148497 registros en filas y 25 columnas, ahora veremos qué tipo de dato contiene cada columna

```
print(covid_MX.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148497 entries, 0 to 148496
Data columns (total 25 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   SECTOR          148497 non-null  int64
 1   ENTIDAD_UM      148497 non-null  int64
 2   SEXO            148497 non-null  int64
 3   ENTIDAD_RES     148497 non-null  int64
 4   TIPO_PACIENTE   148497 non-null  int64
 5   FECHA_INGRESO   148497 non-null  datetime64[ns]
 6   FECHA_SINTOMAS  148497 non-null  datetime64[ns]
 7   FECHA_DEF       148497 non-null  datetime64[ns]
 8   INTUBADO        148497 non-null  int64
 9   NEUMONIA        148497 non-null  int64
 10  EDAD            148497 non-null  category
 11  NACIONALIDAD    148497 non-null  int64
 12  DIABETES        148497 non-null  int64
 13  EPOC            148497 non-null  int64
 14  ASMA            148497 non-null  int64
 15  INMUSUPR        148497 non-null  int64
 16  HIPERTENSION    148497 non-null  int64
 17  OTRA_CON        148497 non-null  int64
 18  CARDIOVASCULAR  148497 non-null  int64
 19  OBESIDAD        148497 non-null  int64
 20  RENAL_CRONICA   148497 non-null  int64
 21  TABAQUISMO      148497 non-null  int64
 22  OTRO_CASO       148497 non-null  int64
 23  RESULTADO       148497 non-null  int64
 24  UCI             148497 non-null  int64
dtypes: category(1), datetime64[ns](3), int64(21)
memory usage: 27.3 MB
None
```

Ya no se cuenta con ninguna columna de tipo objeto, por último revisamos si no se cuenta con datos nulos

```
print(pd.isnull(covid_MX).sum())
```

```
SECTOR            0
ENTIDAD_UM        0
SEXO              0
ENTIDAD_RES       0
TIPO_PACIENTE     0
FECHA_INGRESO     0
FECHA_SINTOMAS    0
FECHA_DEF         0
INTUBADO          0
NEUMONIA          0
EDAD              0
NACIONALIDAD      0
DIABETES          0
EPOC              0
ASMA              0
INMUSUPR          0
HIPERTENSION      0
OTRA_CON          0
CARDIOVASCULAR    0
OBESIDAD          0
RENAL_CRONICA     0
TABAQUISMO        0
OTRO_CASO         0
RESULTADO         0
UCI               0
dtype: int64
```

En este momento se inicia el análisis de la información a partir de gráficos, siendo el primero de estos un mapa de la república mexicana

Gráficos

Creamos una nueva columna llamada "CASOS" en el csv de entidades y esta va a tener

```
#Número de infectados en México por estado
#-----------------------------------------

#Se contabiliza el numero de fallecidos por cada institución de salud
df_entidades['CASOS']=[len(covid_MX.loc[(covid_MX['ENTIDAD_UM']== n) &
        (covid_MX['RESULTADO']==0)]) for n in list(df_entidades['CLAVE_ENTIDAD'])]
```

Se realiza un llamado a un archivo geoJson en una página web, está realizará un mapa dinamico de la

república mexicana

```
repo_url = 'https://raw.githubusercontent.com/angelnmara/geojson/master/mexicoHigh.json' #Archivo GeoJSON
mx_regions_geo = requests.get(repo_url).json()
```

Creamos el mapa con choropleth y las siguientes caracteristicas

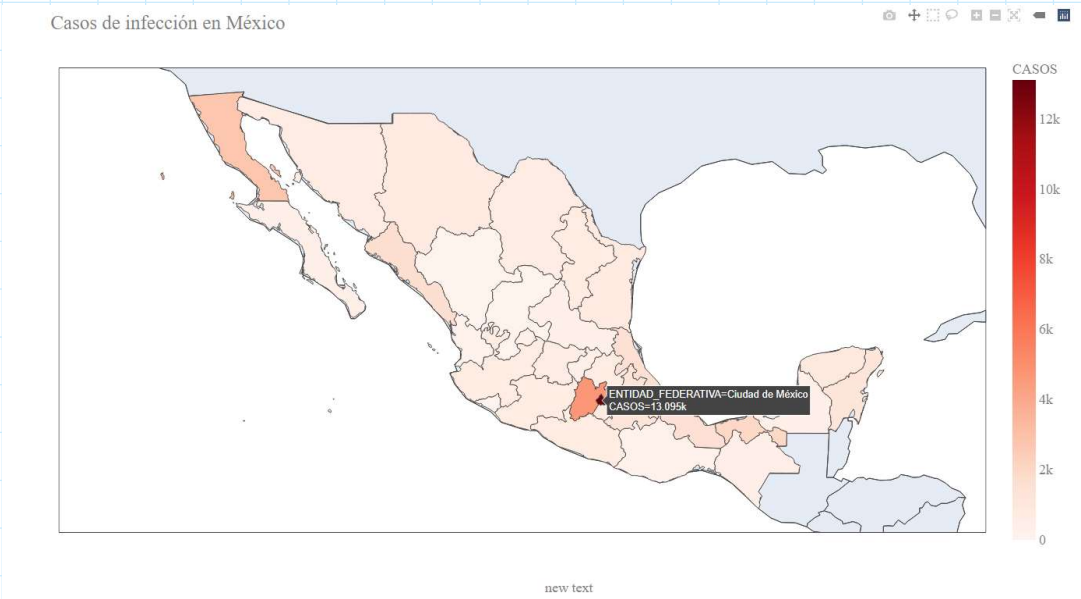
```
fig = px.choropleth(data_frame=df_entidades,
                    geojson=mx_regions_geo, #Se hace referencia a la linea mx_regions_geo
                    locations='ENTIDAD_FEDERATIVA', # nombre de la columna del Dataframe
                    featureidkey='properties.name',  # ruta al campo del archivo GeoJSON con el que se hará la relación (nombre de los estados)
                    color='CASOS', #El color depende de las cantidades
                    color_continuous_scale="Reds",
                   )
```

Modificar las características del gráfico y mostrarlo en nuestro navegador web ya que es un mapa dinámico

```
fig.update_geos(showcountries=True, showcoastlines=True, showland=True, fitbounds="locations")

fig.update_layout(
    title_text = 'Casos de infección en México',
    font=dict(
        family="Ubuntu",
        size=18,
        color="#7f7f7f"
    ),
    annotations = [dict(
        x=0.55,
        y=-0.1,
        xref='paper',
        yref='paper',
        showarrow = False
    )]
)
fig.show()
plt.show()
```

Aquí se ve el mapa



Como segundo gráfico se creará con los datos de personas que han dado positivo a Covid-19 en 4 clasificaciones:

Personas con resultado positivo

```
#Personas cuyo resultado fue positivo
Covid_positivo=covid_MX.loc[covid_MX['RESULTADO']==0]
```

Personas fallecidas con resultado positivo

```
#Personas fallecidas con resultado positivo
Covid_muerto_positivo=covid_MX.loc[(covid_MX['RESULTADO']==0) &
    (covid_MX['FECHA_DEF']!='31/12/2020') & (covid_MX['FECHA_DEF'].notnull())]
```

Personas con resultado positivo pero en situación de estar intubadas

```
#Personas vivas con resultado positivo pero intubada
Vivo_intubado=covid_MX.loc[(covid_MX['RESULTADO']==0) & ((covid_MX['FECHA_DEF']=='31/12/2020') |
    (covid_MX['FECHA_DEF'].notnull()))&(covid_MX['INTUBADO']==0)]
```

Personas con resultado positivo pero en estado crítico

```
#Personas vivas con resultado positivo en estado crítico
Vivo_ICU=covid_MX.loc[(covid_MX['RESULTADO']==0) & ((covid_MX['FECHA_DEF']=='31/12/2020') |
    (covid_MX['FECHA_DEF'].notnull())) & (covid_MX['UCI']==0)]
```

Hay que obtener la longitud de cada variable

```
#Se obtiene el numero total del estado de las personas
cptotal=len(Covid_positivo)
cmptotal=len(Covid_muerto_positivo)
Vitotal=len(Vivo_intubado)
Vicutotal=len(Vivo_ICU)
```

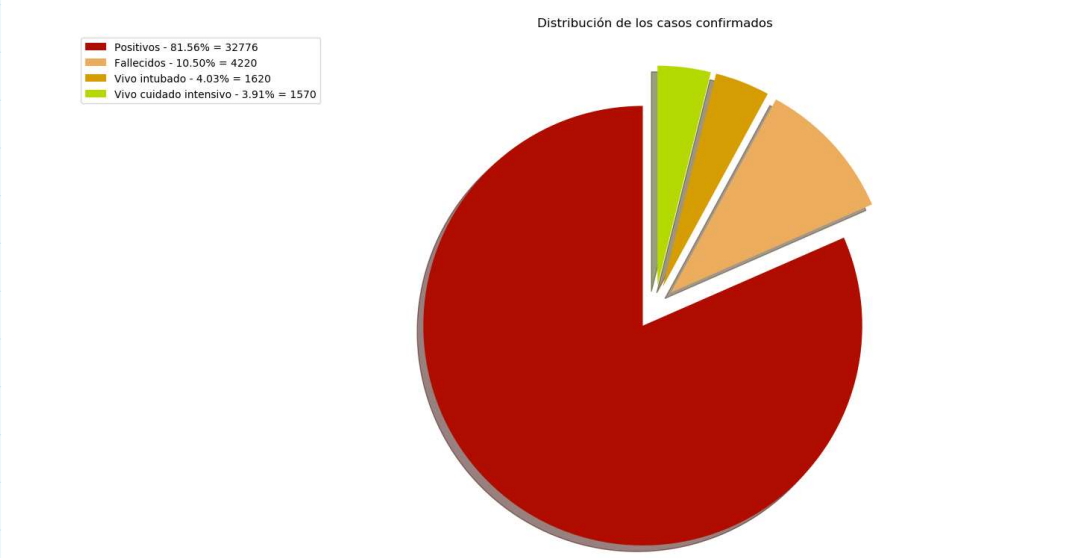
Realizamos la información que llevará el gráfico, el tamaño de la información en un arreglo, los títulos de cada la categoría, el porcentaje de personas en cada categoría y la leyenda del gráfico

```
#Se realiza las características para crear el gráfico
sizes=np.array([cptotal-cmptotal-Vitotal-Vicutotal,cmptotal,Vitotal,Vicutotal])
titulos=['Positivos','Fallecidos','Vivo intubado','Vivo cuidado intensivo']
porcentaje=100.*sizes/sizes.sum()
leyenda=['{0} - {1:0.2f}% = {2:0.0f}'.format(titulos[i],porcentaje[i],sizes[i]) for i in range(len(titulos))]
```

Por ultimo solo creamos un gráfico de pastel con las siguientes características

```
#Se crea el gráfico
fig, ax1=plt.subplots(figsize=(10,6))
ax1.pie(sizes,startangle=90,shadow=True,explode=(0.1,0.1,0.1,0.1),colors=colors)
ax1.set_title('Distribución de los casos confirmados')
ax1.legend(leyenda,loc='best',fontsize=10,bbox_to_anchor=(-0.1, 1.))
fig.tight_layout()
plt.show()
```

El gráfico que hemos creado es el siguiente



El tercer gráfico muestra el número de personas fallecidas según la institución de salud

Creamos una columna en el data de Sector con el total de personas fallecidas, según la institución de salud

que corresponde y los acomodamos de forma descendiente

```
#Se obtiene el total de casos positivos fallecidos
df_sector['TOTAL_POSITIVOS_DEF']=[len(covid_MX.loc[(covid_MX['SECTOR']==n) &
    (covid_MX['FECHA_DEF']!='31/12/2020') & (covid_MX['FECHA_DEF'].notnull()) &
    (covid_MX['RESULTADO']==0)])for n in list(df_sector['CLAVE'])]
#Se acomodan los valores de manera descendente
df_sector=df_sector.sort_values('TOTAL_POSITIVOS_DEF',ascending=False)
```

Creamos las características del gráfico que son:

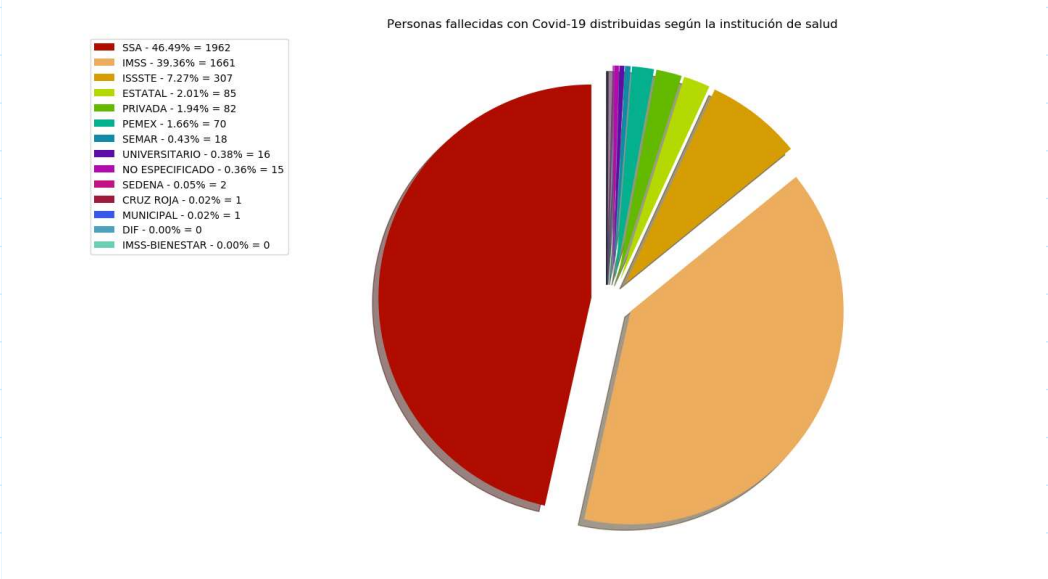
- Total de fallecidos en cada institución
- Los títulos de la institución
- El porcentaje de personas fallecidas en cada institución
- Leyenda para el gráfico

```
#Se realiza las características para crear el gráfico
tamaño=np.array(list(df_sector['TOTAL_POSITIVOS_DEF']))
titulos=[i for i in df_sector['DESCRIPCIÓN']]
porcentaje=100.0*tamaño/tamaño.sum()
leyenda=['{0} - {1:0.2f}% = {2:0.0f}'.format(titulos[i],porcentaje[i],tamaño[i]) for i in range(len(titulos))]
```

Creamos el gráfico de pastel con lo siguiente

```
#Se crea el gráfico
fig, ax1=plt.subplots(figsize=(10,6))
ax1.pie(tamaño,startangle=90,shadow=True,colors=colors,explode=explode)
ax1.set_title('Personas fallecidas con Covid-19 distribuidas según la institución de salud')
ax1.legend(leyenda,loc='best',fontsize=10,bbox_to_anchor=(-0.1, 1.))
fig.tight_layout()
plt.show()
```

Se visualiza el siguiente gráfico



El cuarto gráfico mostrará el número total de personas con resultado positivo según la institución de salud y el número de fallecidos.

Se contabiliza el número total de personas fallecidas según la institución

```
#Se contabiliza el numero de fallecidos por cada institución de salud
df_sector['TOTAL_DEF']=[len(covid_MX.loc[(covid_MX['SECTOR']== n) & (covid_MX['FECHA_DEF']!='31/12/2020')
    & (covid_MX.FECHA_DEF.notnull())]) for n in list(df_sector['CLAVE'])]
```

Se contabiliza el número de personas con resultado positivo

```
#Se contabiliza el numero de personas con covid
df_sector['TOTAL_POS']=[len(covid_MX.loc[(covid_MX['SECTOR']== n) & (covid_MX['RESULTADO']==0)])
    for n in list(df_sector['CLAVE'])]
```

Acomodamos lo valores de manera descendente

```
#Se acomodan los valores de manera descendente
df_sector=df_sector.sort_values('TOTAL_POS',ascending=False)
```

Realizamos las características del gráfico:

- Total de positivos
- Total de personas fallecidas con resultado positivo
- Porcentaje de fallecidos respecto al de positivos que es la tasa de letalidad

```
#Se realiza las características para crear el gráfico
total_positivos=np.array(list(df_sector['TOTAL_POS'])) #Numero de casos positivos
total_positivos_muertos=np.array(list(df_sector['TOTAL_POSITIVOS_DEF'])) #Numero de muertos positivos
porcentaje1=100.0*total_positivos_muertos/total_positivos.sum() #La tasa de Mortalidad
```

Para crear este gráfico realizamos los siguientes pasos:

- Creamos labels para modificar los labels en el eje X
- Creamos un vector x que va de 0 a la longitud de la columna "DESCRIPCIÓN"
- Un ancho que es w que será del gráfico

```
#Se crea el gráfico
fig, ax1=plt.subplots(figsize=(10,6))
loc,labels=plt.xticks() #Modificar los labels en x
x=np.arange(len(df_sector['DESCRIPCIÓN'])) #Los titulos en el eje X
w=0.6 #Ancho de la gráfica
```

Creamos 2 gráficas la primera con los datos positivos y la segunda con los positivos fallecidos

```
graf1=ax1.bar(x,total_positivos,width=w/2,align='center') #Grafica de casos positivos
graf2=ax1.bar(x+w,total_positivos_muertos,width=w/2,align='center') #Grafica de casos positivos muertos
```

Modificaremos los colores da cada gráfico con enumerate y le pondremos el color de nuestra variable colors

```
#Se modifican los colores para cada barra
for i, bar in enumerate(graf1):
    bar.set_color(colors[i])
for i, bar in enumerate(graf2):
    bar.set_color(colors2[i])
```

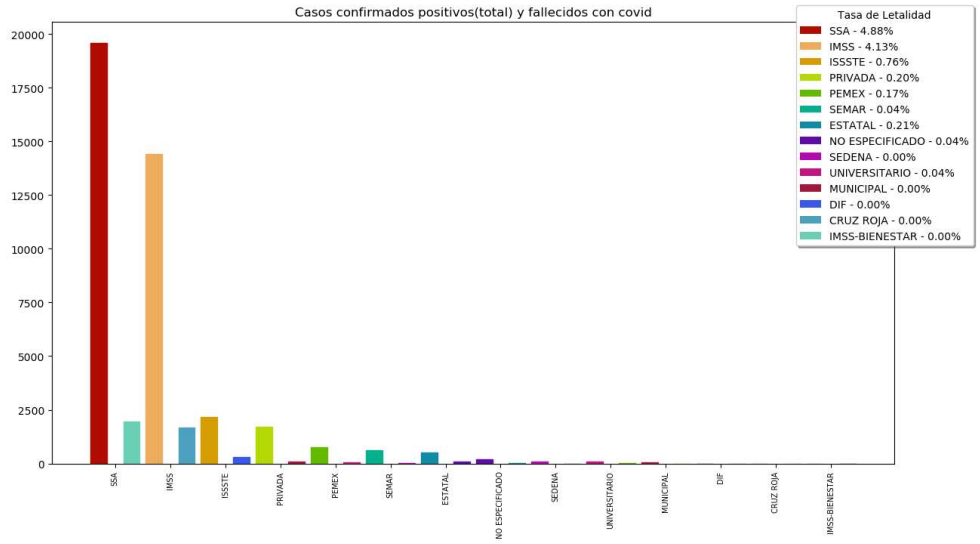
Creamos un gráfico con xticks que tendrá la siguiente información

```
#Se modifican los labels en X
plt.xticks(x + w /2, df_sector['DESCRIPCIÓN'], rotation='vertical',fontsize=7)
```

Creamos las leyendas del gráfico respecto al color que se le ha brindado, el titulo y se muestra el gráfico

```
#Se prepara la leyenda según el color de cada grafico
ax1.legend(handles=[matplotlib.patches.Patch(facecolor=colors[x],
    label='{0} - {1:0.2f}%'.format(list(df_sector['DESCRIPCIÓN'])[x],porcentaje1[x]))
    for x in range(len(df_sector['DESCRIPCIÓN']))],
    loc='best',bbox_to_anchor=(1.1, 1.05), fancybox=True, shadow=True, title="Tasa de Letalidad")
ax1.set_title('Casos confirmados positivos(total) y fallecidos con covid')
plt.show()
```

El gráfico visualizado es el siguiente



El quinto gráfico es el número de infectados acumulados en México

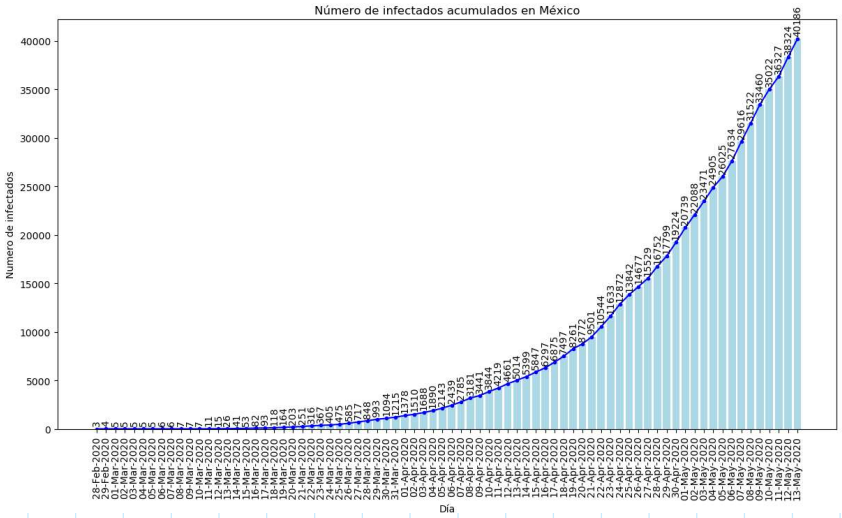
Se aplica prácticamente lo mismo que en el gráfico, solo que la información es de los casos confirmados de la data de casos acumulados

```
fig, ax1=plt.subplots(figsize=(10,6))
loc,labels=plt.xticks() #Modificar los labels en x
labels=casos_acumulados['Dates']
x=np.arange(len(labels))
datos1=ax1.bar(x,casos_acumulados['Confirmed Cases'],color='lightblue')
ax1.plot(x,casos_acumulados['Confirmed Cases'],'.-b')
ax1.set_title('Número de infectados acumulados en México')
ax1.set_xlabel('Día')
ax1.set_ylabel('Numero de infectados')
ax1.set_xticks(x)
ax1.set_xticklabels(labels,rotation=90)
```

Creamos una función para poner el valor de la barra arriba de está y mostramos el gráfico

```
def autolabel(datos):
    for dato in datos:
        height=dato.get_height()
        ax1.annotate('{}'.format(height),
                     xy=(dato.get_x() + dato.get_width() / 2, height),
                     xytext=(0, 6),  # 3 points vertical offset
                     textcoords="offset points",
                     ha='center', va='bottom',rotation=90)
autolabel(datos1)
plt.show()
```

Nuestro gráfico se ve de la siguiente manera



El sexto gráfico es el número de infectados por día, creamos una columna en el data de casos acumulados con nombre "Cases per Day"

```
#Número de infectados por día en México
#----------------------------------------

fig, ax1=plt.subplots(figsize=(10,6))
loc,labels=plt.xticks() #Modificar los labels en x
casos_acumulados['Cases per Day']=[len(covid_MX.loc[(covid_MX['FECHA_INGRESO']== n) &
    (covid_MX['RESULTADO']==0)]) for n in list(casos_acumulados['Dates'])]
labels=casos_acumulados['Dates']
```

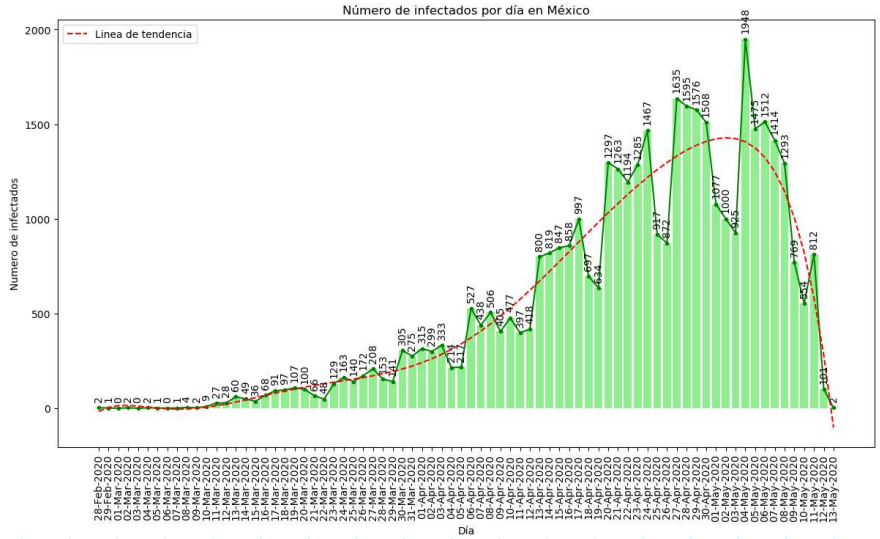
Creamos un polinomio que me muestre la tendencia del gráfico usando los siguientes comandos

```
x=np.arange(len(labels))
y1=np.array(casos_acumulados['Cases per Day'])
z=np.polyfit(x,y1,8)
p=np.poly1d(z)
```

Generamos el gráfico, contiene las barras, un gráfico de línea y el polinomio de tendencia

```
#Aquí se genera el gráfico
datos1=ax1.bar(x,casos_acumulados['Cases per Day'],color='lightgreen')
ax1.plot(x,casos_acumulados['Cases per Day'],'.-g')
ax1.plot(x,p(x),'--r',label='Linea de tendencia')
ax1.set_title('Número de infectados por día en México')
ax1.set_xlabel('Día')
ax1.set_ylabel('Numero de infectados')
ax1.legend()
ax1.set_xticks(x)
ax1.set_xticklabels(labels,rotation=90)
autolabel(datos1)
plt.show()
```

Nuestro gráfico se observa de la siguiente forma



Machine Learning

Crearemos una columna en donde se mencione si una persona sobrevivió o no a la enfermedad y creamos las variables X e y para los modelos de machine learning

```
#Creamos la columa de sobrevivio ante la enfermedad 0 es si y 1 es no
#Condición uno si ha muerto y el resultado es positivo se le pone 1
#Condición 2 si ha muerto y el resultado es negativo se le pone 1
covid_MX['SOBREVIVIO']=np.where(((covid_MX['FECHA_DEF']=='31/12/2020') & (covid_MX['RESULTADO']==0) |
    ((covid_MX['FECHA_DEF']=='31/12/2020') & (covid_MX['RESULTADO']==1))), 1, 0)
No_necesario=['SECTOR','ENTIDAD_UM','ENTIDAD_RES','FECHA_INGRESO','FECHA_SINTOMAS',
    'FECHA_DEF','NACIONALIDAD','SOBREVIVIO']

y=np.array(covid_MX['SOBREVIVIO'])
X=np.array(covid_MX.drop(No_necesario,1))
```

Se hacen los vectores de prueba y test con el siguiente comando

```
#Creamos los arrays de prueba y test
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

Aplicando la regresión logística

```
#-------------------
#Regresión Logística
#-------------------
logreg=LogisticRegression(max_iter=100)
logreg.fit(X_train,y_train)
Y_pred=logreg.predict(X_test)
```

Aplicando Maquinas de Vectores de Soporte

```
#-------------------------------
#Maquinas de Vectores de Soporte
#-------------------------------
MSV=SVC(max_iter=100)
MSV.fit(X_train,y_train)
Y_pred_MSV=MSV.predict(X_test)
```

Aplicando el Árbol de decisiones

```
#-------------------
#Arbol de Decisiones
#-------------------
AD=DecisionTreeClassifier()
AD.fit(X_train,y_train)
Y_pred_AD=AD.predict(X_test)
```

Observemos la precisión de cada modelo

```
#Presición de los modelos
print("Presición de la regresión logistica: ",logreg.score(X_train,y_train))
print('Presición del algoritmo Maquinas de Vectores de Soporte es: {}'.format(MSV.score(X_train,y_train)))
print('Presición del algoritmo Árbol de Decisiones es: {}'.format(AD.score(X_train,y_train)))
```

La precisión de cada modelo es:

```
Presición de la regresión logistica:  0.9492663956160509
Presición del algoritmo Maquinas de Vectores de Soporte es: 0.42172782140963155
Presición del algoritmo Árbol de Decisiones es: 0.974502723132739
```

El mejor modelo que pronostica un mayor resultado es el de Árbol de Decisiones

Advertencia: Se debe hacer una limpieza más grande en la información