Actualización de datos de Covid-19 en la infeccion hasta el 2021.

Covid-19 infección en Ecuador. Modelos matemáticos y predicciones

Una comparación de modelos, lineal, polilnomico,logísticos y exponenciales aplicados a la infección por el virus Covid-19

Se realiza un análisis matemático simple del crecimiento de la infección en Python y dos modelos para comprender mejor la evolución de la infección.

Se crea modelos de series temporales del número total de personas infectadas hasta la fecha (es decir, las personas realmente infectadas más las personas que han sido infectadas). Estos modelos tienen parámetros , que se estimarán por ajuste de curva.

In [10]:
```
# Importar las librerias para el analasis
import pandas as pd
import numpy as np
from datetime import datetime,timedelta
from sklearn.metrics import mean_squared_error
from scipy.optimize import curve_fit
from scipy.optimize import fsolve
from sklearn import linear_model
import matplotlib.pyplot as plt
%matplotlib inline
```

In [11]:
```
url = 'http://cowid.netlify.com/data/owid-covid-data.csv'

df = pd.read_csv(url)
df
```

Out[11]:

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_
0	AFG	Asia	Afghanistan	2020-02-24	1.0	1.0	NaN	NaN	
1	AFG	Asia	Afghanistan	2020-02-25	1.0	0.0	NaN	NaN	
2	AFG	Asia	Afghanistan	2020-02-26	1.0	0.0	NaN	NaN	
3	AFG	Asia	Afghanistan	2020-02-27	1.0	0.0	NaN	NaN	
4	AFG	Asia	Afghanistan	2020-02-28	1.0	0.0	NaN	NaN	
...	...	...	...	...	...	...	...	...	
93527	ZWE	Africa	Zimbabwe	2021-05-31	38961.0	17.0	37.857	1594.0	
93528	ZWE	Africa	Zimbabwe	2021-06-01	38998.0	37.0	41.714	1599.0	
93529	ZWE	Africa	Zimbabwe	2021-06-02	39031.0	33.0	30.286	1599.0	
93530	ZWE	Africa	Zimbabwe	2021-06-03	39092.0	61.0	34.000	1604.0	

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_
93531	ZWE	Africa	Zimbabwe	2021-06-04	39144.0	52.0	32.286	1605.0	

93532 rows × 59 columns

In [12]:
```
df = df[df['location'].isin(['Ecuador'])] #Filtro la Informacion solo para Ecuador
df = df.loc[:,['date','total_cases']] #Selecciono las columnas de analasis
# Expresar las fechas en numero de dias desde el 01 Enero
FMT = '%Y-%m-%d'
date = df['date']
df['date'] = date.map(lambda x : (datetime.strptime(x, FMT) - datetime.strptime("2020-01-01", F

df
```

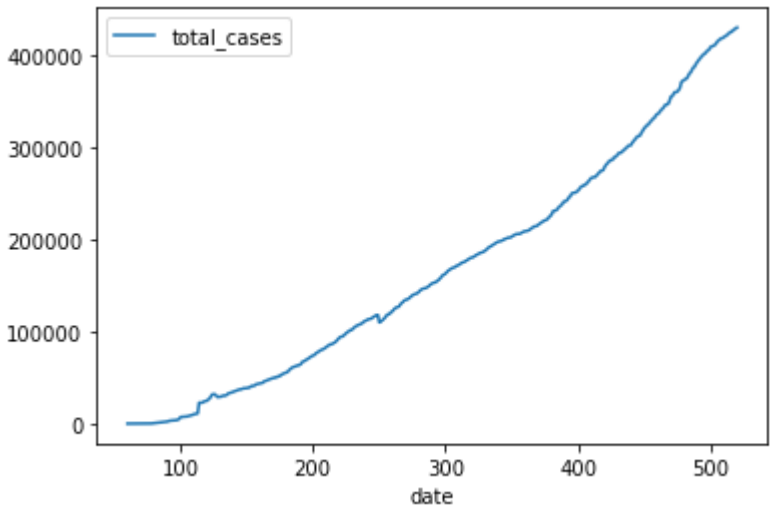
Out[12]:

	date	total_cases
24480	60	6.0
24481	61	6.0
24482	62	7.0
24483	63	10.0
24484	64	13.0
...	...	...
24936	516	426037.0
24937	517	427690.0
24938	518	428865.0
24939	519	429817.0
24940	520	430739.0

461 rows × 2 columns

In [13]:
```
df.plot(x ='date', y='total_cases')
```

Out[13]:
```
<AxesSubplot:xlabel='date'>
```



Ahora podemos analizar los cuatro modelos que tomaré en el examen, que son la función lineal, polinomica,logística y la función exponencial . Cada modelo tiene tres parámetros , que se estimarán mediante un cálculo de ajuste de curva en los datos históricos.

EL modelo lineal

La regresión lineal es un algoritmo de aprendizaje supervisado que se utiliza en Machine Learning y en estadística. En su versión más sencilla, lo que haremos es «dibujar una recta» que nos indicará la tendencia de un conjunto de datos continuos.

Recordemos rápidamente la fórmula de la recta:

Y = mX + b

Donde Y es el resultado, X es la variable, m la pendiente (o coeficiente) de la recta y b la constante o también conocida como el «punto de corte con el eje Y» en la gráfica (cuando X=0) Ejemplo

In [14]:
```
x = list(df.iloc [:, 0]) # Fecha
y = list(df.iloc [:, 1]) # Numero de casos
# Creamos el objeto de Regresión Lineal
regr = linear_model.LinearRegression()

# Entrenamos nuestro modelo
regr.fit(np.array(x).reshape(-1, 1) ,y)

# Veamos los coeficienetes obtenidos, En nuestro caso, serán la Tangente
print('Coefficients: \n', regr.coef_)
# Este es el valor donde corta el eje Y (en X=0)
print('Independent term: \n', regr.intercept_)
# Error Cuadrado Medio
```

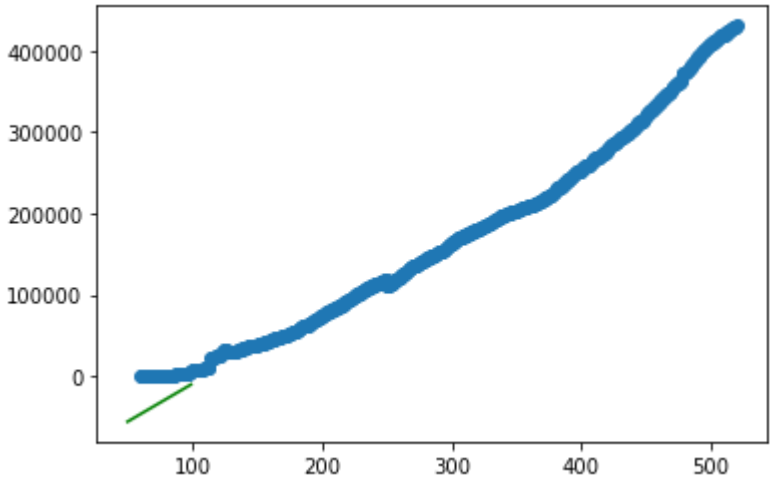
```
Coefficients:
 [932.24128457]
Independent term:
 -102541.89226401248
```

In [15]:
```
#Vamos a comprobar:
# Quiero predecir cuántos "Casos" voy a obtener por en el dia 100,
# según nuestro modelo, hacemos:
y_prediccion = regr.predict([[100]])
print(int(y_prediccion))
```

```
-9317
```

In [16]:
```
#Graficar
plt.scatter(x, y)
x_real = np.array(range(50, 100))
print(x_real)
plt.plot(x_real, regr.predict(x_real.reshape(-1, 1)), color='green')
plt.show()
```

```
[50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73
 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97
 98 99]
```



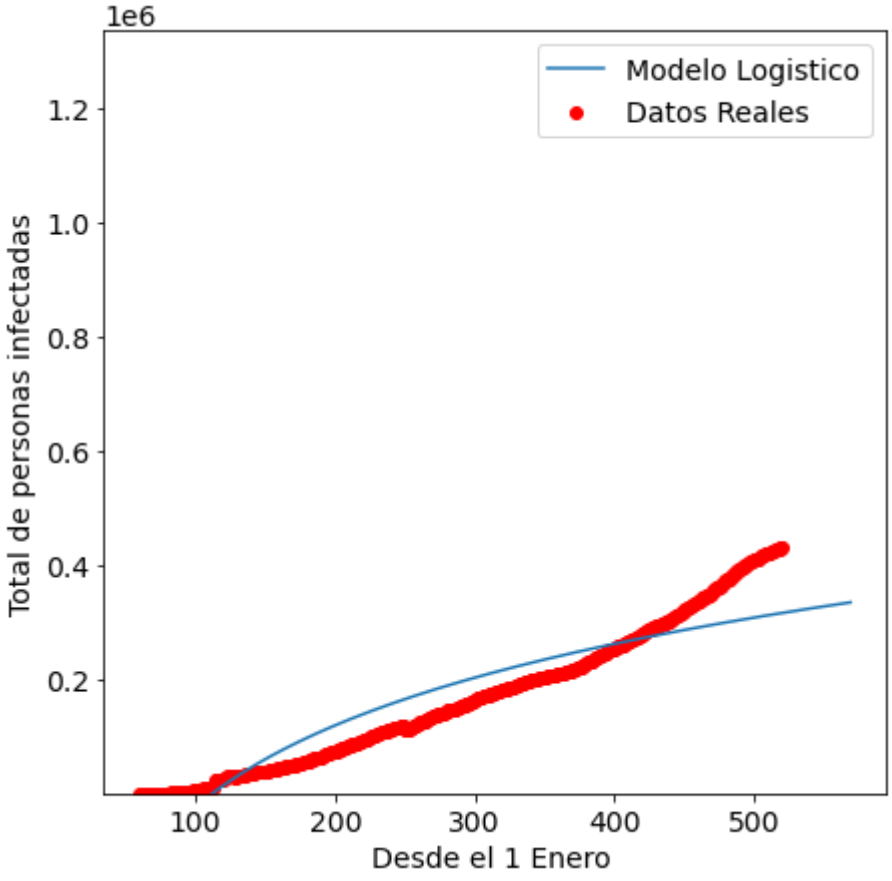
El modelo logístico se ha utilizado ampliamente para describir el crecimiento de una población. Una infección puede describirse como el crecimiento de la población de un agente patógeno, por lo que un modelo logístico parece razonable . La expresión más genérica de una función logística es:

In [17]:
```
def modelo_logistico(x,a,b):
    return a+b*np.log(x)

exp_fit = curve_fit(modelo_logistico,x,y) #Extraemos los valores de los paramatros
print(exp_fit)
```

```
(array([-971984.43006665,  205935.21274674]), array([[ 5.29235740e+08, -9.46493007e+07],
       [-9.46493007e+07,  1.71010282e+07]]))
```

In [19]:
```
pred_x = list(range(min(x),max(x)+50)) # Predecir 50 dias mas
plt.rcParams['figure.figsize'] = [7, 7]
plt.rc('font', size=14)
# Real data
plt.scatter(x,y,label="Datos Reales",color="red")
# Predicted exponential curve
plt.plot(pred_x, [modelo_logistico(i,exp_fit[0][0],exp_fit[0][1]) for i in pred_x], label="Mode
plt.legend()
plt.xlabel("Desde el 1 Enero ")
plt.ylabel("Total de personas infectadas")
plt.ylim((min(y)*0.9,max(y)*3.1)) # Definir los limites de Y
plt.show()
```



In [ ]: