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 [1]:

```
# 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 [2]:

```
# Actualizar los datos (URL)

url = 'http://cowid.netlify.com/data/full_data.csv'

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

Out[2]:

	date	location	new_cases	new_deaths	total_cases	total_deaths
0	2020-02-25	Afghanistan	NaN	NaN	1	NaN
1	2020-02-26	Afghanistan	0.0	NaN	1	NaN
2	2020-02-27	Afghanistan	0.0	NaN	1	NaN
3	2020-02-28	Afghanistan	0.0	NaN	1	NaN
4	2020-02-29	Afghanistan	0.0	NaN	1	NaN
...	...	...	...	...	...	...
2862	2020-03-13	World	7488.0	338.0	132758	4956.0
2863	2020-03-14	World	9761.0	433.0	142534	5392.0
2864	2020-03-15	World	10967.0	343.0	153517	5735.0
2865	2020-03-16	World	13971.0	855.0	167506	6606.0
2866	2020-03-17	World	11594.0	819.0	179112	7426.0

2867 rows × 6 columns

Imprimos los resultados y agregamos el numero del dia

In [3]:

```
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", FMT)).days)

df
```

Out[3]:

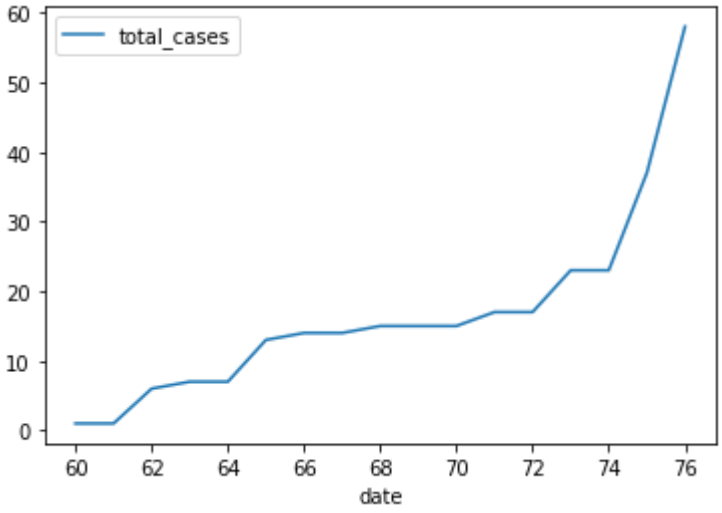
	date	total_cases
681	60	1
682	61	1
683	62	6
684	63	7
685	64	7
686	65	13
687	66	14
688	67	14
689	68	15
690	69	15
691	70	15
692	71	17
693	72	17
694	73	23
695	74	23
696	75	37
697	76	58

In [4]:

```
df.plot(x ='date', y='total_cases')
```

Out[4]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x14dabe3b0d0>
```



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.

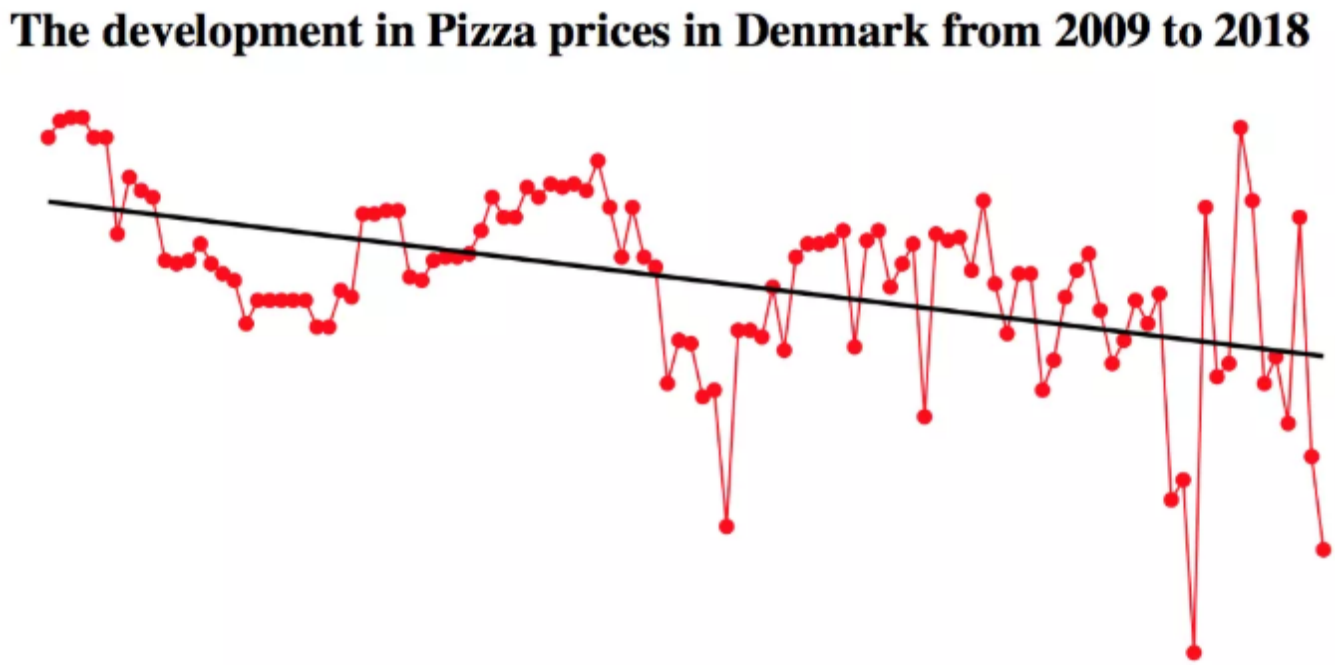
EL modelo lineal

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



Recordemos que los algoritmos de Machine Learning Supervisados, aprenden por sí mismos y -en este caso- a obtener automáticamente esa «recta» que buscamos con la tendencia de predicción. Para hacerlo se mide el error con respecto a los puntos de entrada y el valor «Y» de salida real.

In [8]:

```
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:
 [2.31617647]
Independent term:
 -140.85294117647058
```

De la ecuación de la recta y = mX + b nuestra pendiente «m» es el coeficiente y el término independiente «b»

In [9]:

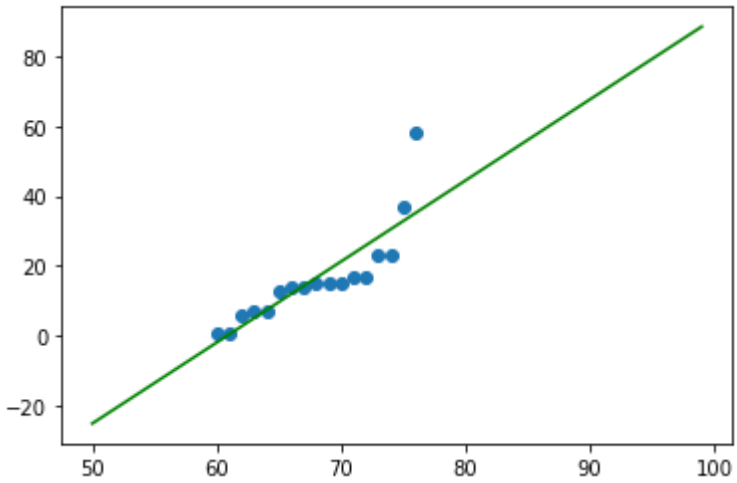
```
#Vamos a comprobar:
# Quiero predecir cuántos "Casos" voy a obtener por en el dia 100,
# según nuestro modelo, hacemos:
y_prediccion = regr.predict([[100]])
print(int(y_prediccion))
```

```
90
```

In [10]:

```
#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]
```



Analisis covid Ecuador

In [11]:

```
# Actualizar los datos (URL)
#datos tomados
url = 'ecuador-covid-data.csv'
df = pd.read_csv(url)
df= df.fillna(0)
df
```

Out[11]:

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	tot
0	ECU	South America	Ecuador	2019-12-31	0.0	0.0	0.000	
1	ECU	South America	Ecuador	2020-01-01	0.0	0.0	0.000	
2	ECU	South America	Ecuador	2020-01-02	0.0	0.0	0.000	
3	ECU	South America	Ecuador	2020-01-03	0.0	0.0	0.000	
4	ECU	South America	Ecuador	2020-01-04	0.0	0.0	0.000	
...	...	...	...	...	...	...	...	
303	ECU	South America	Ecuador	2020-10-29	164908.0	1716.0	1326.143	
304	ECU	South America	Ecuador	2020-10-30	166302.0	1394.0	1407.286	
305	ECU	South America	Ecuador	2020-10-31	167147.0	845.0	1268.143	
306	ECU	South America	Ecuador	2020-11-01	168192.0	1045.0	1225.429	
307	ECU	South America	Ecuador	2020-11-02	169194.0	1002.0	1079.857	

308 rows × 50 columns

In [60]:

```
#Vamos a comprobar:
# Quiero predecir cuántos "Casos" voy a obtener por en el dia 322,
# según nuestro modelo, hacemos:
y_prediccion = regr.predict([[322]])
print("Despues de 7 dias")
print(int(y_prediccion))
```

```
Despues de 7 dias
604
```

In [61]:

```
#Vamos a comprobar:
# Quiero predecir cuántos "Casos" voy a obtener por en el dia 352,
# según nuestro modelo, hacemos:
y_prediccion = regr.predict([[352]])
print("Despues de 30 dias")
print(int(y_prediccion))
```

```
Despues de 30 dias
674
```

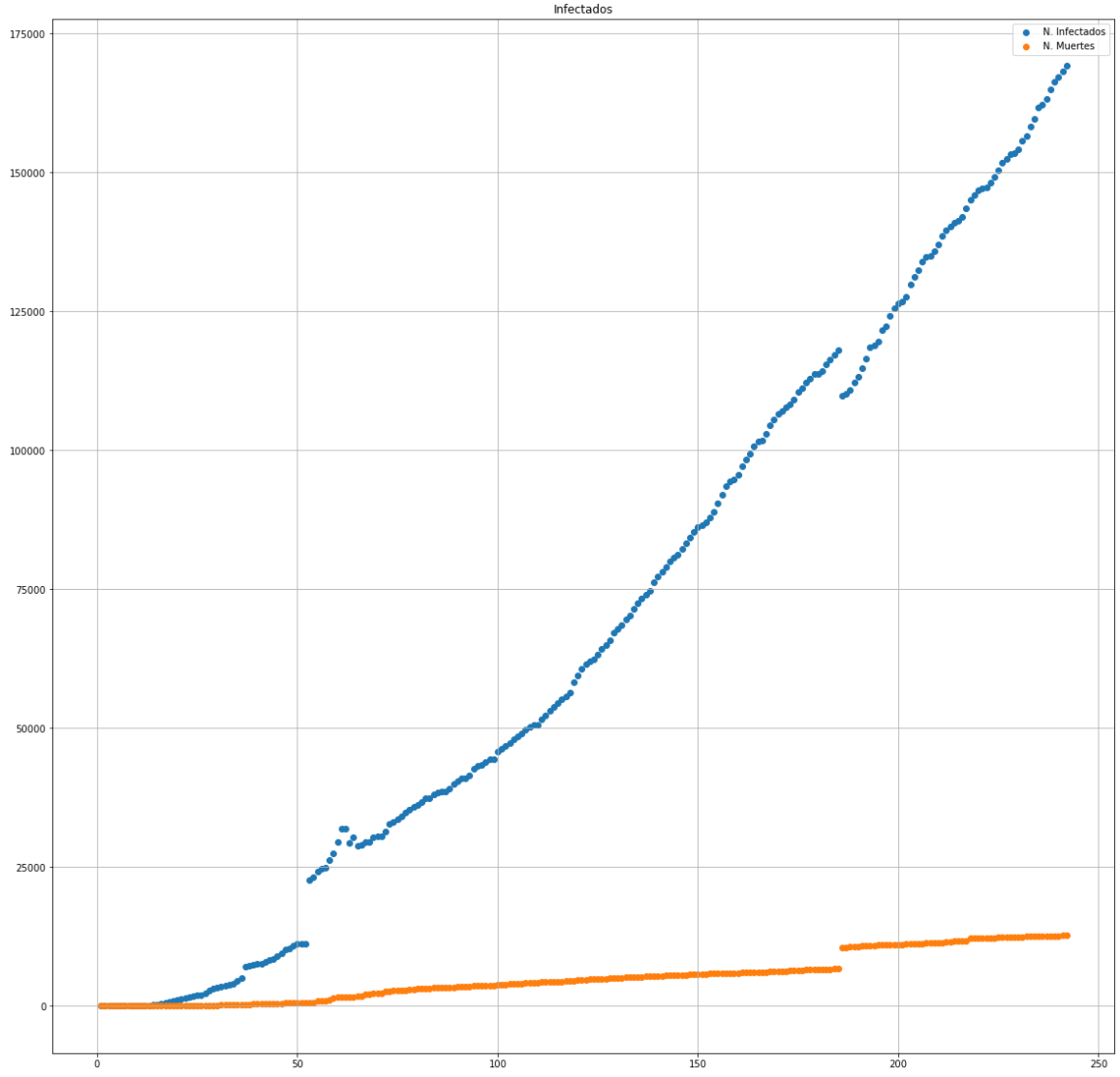
In [25]:

```
# Implementar Covid-19 Regresión Lineal
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression #Regresión Lineal con scikit-learn
import pandas as pd
from datetime import datetime,timedelta
import matplotlib.pyplot as plt
%matplotlib inline
```

In [31]:

```
def f(x):  # función f(x) = 0.1*x + 1.25 + 0.2*Ruido_Gaussiano
    np.random.seed(42) # para poder reproducirlo
    y = 0.1*x + 1.25 + 0.2*np.random.randn(x.shape[0])
    return y

df = pd.read_csv('ecuador-covid-data.csv').fillna(0) # poniendo datos nan a cero
ndf1=ndf[['date','total_cases','total_deaths']]
x=np.arange(1,len(ndf1)+1,1) # arreglo de x lo creo para simular el numero del dia y el
numero de casos
y=np.array(ndf1.values[:,1])
y1=np.array(ndf1.values[:,2])
# hacemos un gráfico de los datos que hemos generado
plt.figure(figsize=(20, 20))
plt.scatter(x,y,label='N. Infectados')
plt.scatter(x,y1,label='N. Muertes')
plt.grid(True)
plt.legend()
plt.title('Infectados');
```


Infectados
N. Infectados
N. Muertes
175000
150000
125000
100000
75000
50000
25000
0
0
50
100
150
200
250

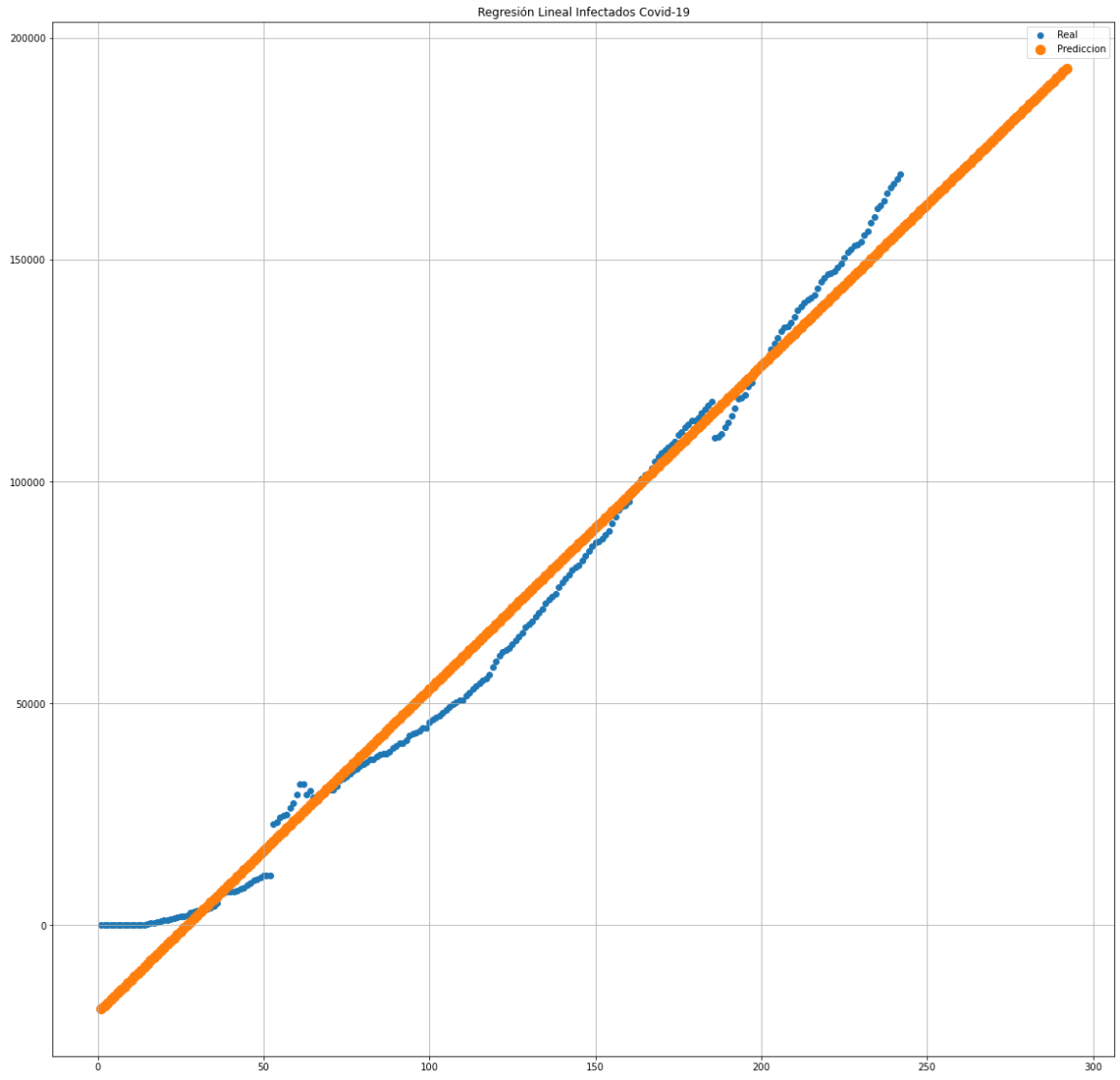
In [32]:

```
regresion_lineal = LinearRegression() # creamos una instancia de LinearRegression
# instruimos a la regresión lineal que aprenda de los datos (x,y)
regresion_lineal.fit(x.reshape(-1,1), y)
# vemos los parámetros que ha estimado la regresión lineal
print('w = ' + str(regresion_lineal.coef_[0]) + ', b = ' + str(regresion_lineal.interce
pt_))
# resultado: w = [0.09183522], b = 1.2858792525736682
if (regresion_lineal.intercept_ < 0):
    ecua='y = {}x {}'
else:
    ecua='y = {}x + {}'
print(ecua.format(regresion_lineal.coef_[0],regresion_lineal.intercept_))
```

```
w = 728.4140897638947, b = -19622.134220362786
y = 728.4140897638947x -19622.134220362786
```

In [36]:

```
fun= lambda num: regresion_lineal.coef_[0]*num+regresion_lineal.intercept_
plt.figure(figsize=(20, 20))
plt.scatter(x,y,label='Real')
plt.grid(True)
plt.title('Regresión Lineal Infectados Covid-19');
x1=np.arange(1,len(ndf1)+51,1)
plt.scatter(x1,fun(x1),linewidth=5.0,label='Prediccion')
plt.legend()
plt.show()
```


Regresión Lineal Infectados Covid-19
Real
Prediccion
200000
150000
100000
50000
0
0
50
100
150
200
250
300

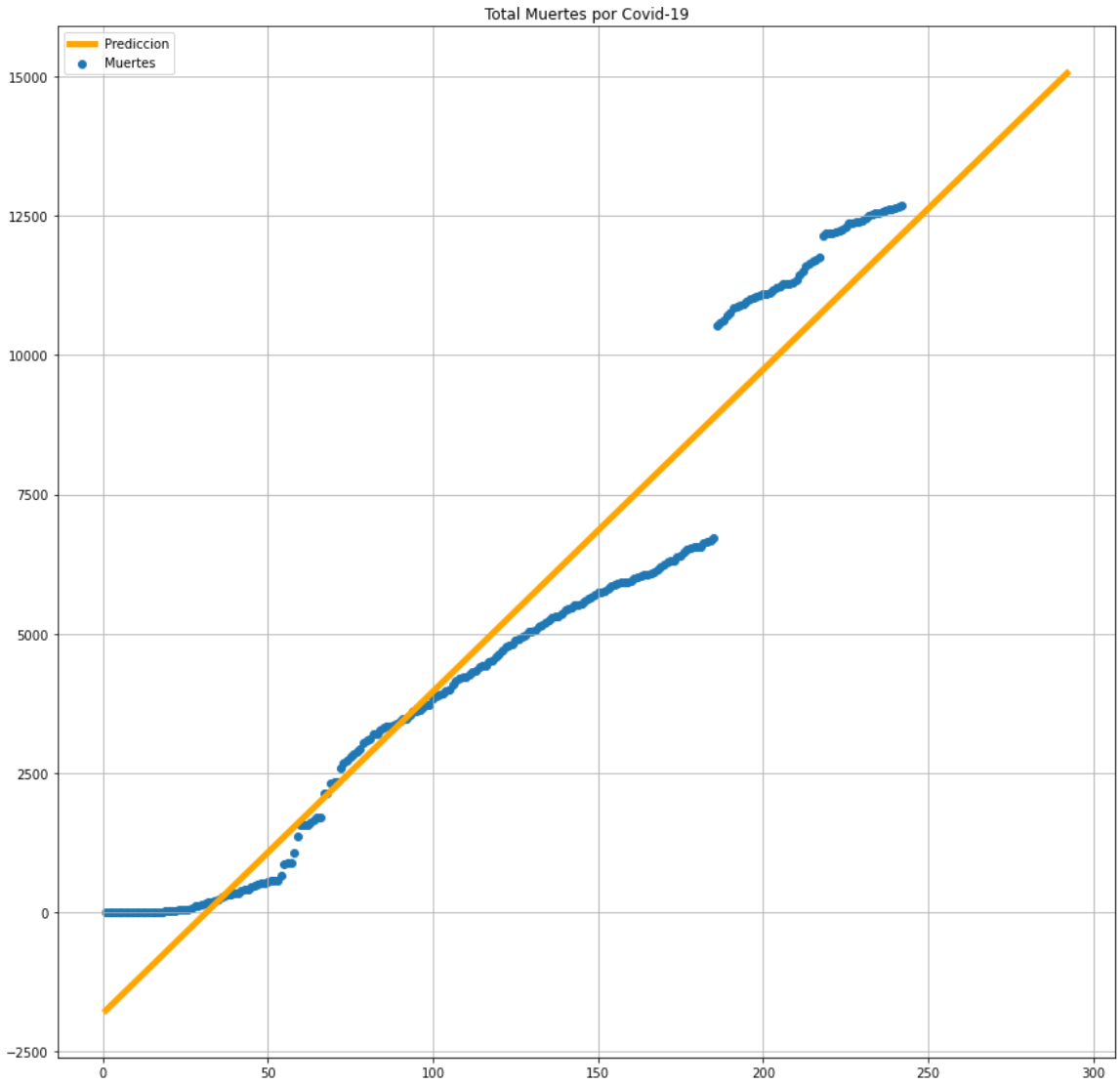
In [37]:

```
regresion_lineal = LinearRegression() # creamos una instancia de LinearRegression
# instruimos a la regresión lineal que aprenda de los datos (x,y)
regresion_lineal.fit(x.reshape(-1,1), y1)
# vemos los parámetros que ha estimado la regresión lineal
print('w = ' + str(regresion_lineal.coef_[0]) + ', b = ' + str(regresion_lineal.interce
pt_))
print(ecua.format(regresion_lineal.coef_[0],regresion_lineal.intercept_))
```

```
w = 57.77491796289733, b = -1815.1029457151653
y = 57.77491796289733x -1815.1029457151653
```

In [46]:

```
fun= lambda num: regresion_lineal.coef_[0]*num+regresion_lineal.intercept_
plt.figure(figsize=(15, 15))
plt.scatter(x,y1,label='Muertes')
plt.grid(True)
plt.title('Total Muertes por Covid-19');
x2 = np.arange(1,len(ndf1)+51,1)
plt.plot(x2,fun(x2),linewidth=5.0,label='Prediccion', color="orange")
plt.legend()
plt.show()
```



In [ ]:

```
```