Covid-19 infección en Ecuador. Modelos probabilisticos

Implementacion de un modelo probabilistico de infección por el virus Covid-19

Se realiza un análisis probabilistico simple del crecimiento de la infección en Python y el 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 probabilidad.

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 = 'https://covid.ourworldindata.org/data/ecdc/new_cases.csv'
df = pd.read_csv(url)
df = df.fillna(0)
df
```

Out[2]:

	date	World	Afghanistan	Albania	Algeria	Andorra	Angola	Anguilla	Antigua and Barbuda	Argentina	...	United States	United States Virgin Islands	Uruguay	Uzbek
0	2019-12-31	27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0.0	0.0	
1	2020-01-01	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0.0	0.0	
2	2020-01-02	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0.0	0.0	
3	2020-01-03	17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0.0	0.0	
4	2020-01-04	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
111	2020-04-20	74142	88.0	14.0	94.0	9.0	0.0	0.0	0.0	102.0	...	24601	0.0	11.0	
112	2020-04-21	77274	35.0	22.0	89.0	4.0	0.0	0.0	0.0	90.0	...	28065	1.0	7.0	
113	2020-04-22	87387	61.0	25.0	93.0	0.0	0.0	0.0	1.0	112.0	...	37289	0.0	8.0	
114	2020-04-23	67629	84.0	25.0	99.0	6.0	0.0	0.0	0.0	144.0	...	17588	0.0	6.0	
115	2020-04-24	80071	105.0	29.0	97.0	1.0	1.0	0.0	0.0	147.0	...	26543	0.0	8.0	

116 rows × 208 columns

Imprimos los resultados y agregamos el numero del dia

In [3]:

```
df = df.loc[:,['date','Ecuador']] #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("2019-12-31", FMT))
.days)
df['Ecuador'].astype(int)
df= df[df["Ecuador"] > 0.0]
df
```

Out[3]:

	date	Ecuador
61	61	1.0
62	62	5.0
63	63	1.0
65	65	3.0
66	66	3.0
69	69	1.0
70	70	1.0
71	71	2.0
74	74	6.0
75	75	5.0
76	76	9.0
77	77	21.0
78	78	53.0
79	79	57.0
80	80	31.0
81	81	227.0
82	82	106.0
83	83	257.0
84	84	192.0
85	85	101.0
86	86	129.0
87	87	192.0
88	88	224.0
89	89	208.0
90	90	55.0
91	91	76.0
92	92	336.0
93	93	456.0
94	94	405.0
95	95	205.0
96	96	97.0
97	97	181.0
98	98	101.0
99	99	248.0
100	100	455.0
101	101	515.0
102	102	2196.0
103	103	96.0
104	104	209.0
105	105	63.0

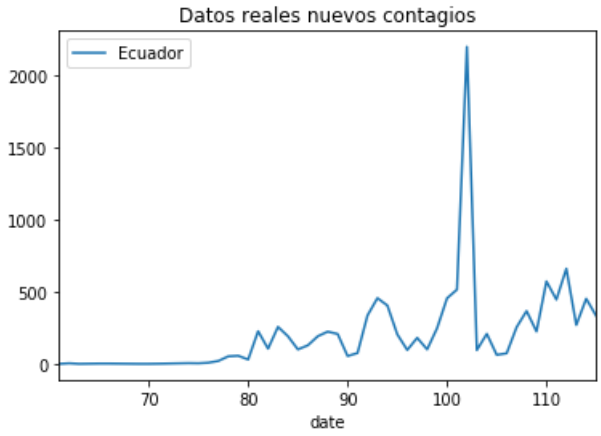
	date	Ecuador
106	106	74.0
107	107	255.0
108	108	367.0
109	109	225.0
110	110	572.0
111	111	446.0
112	112	660.0
113	113	270.0
114	114	452.0
115	115	333.0

In [4]:

```
df.plot(x ='date', y='Ecuador')
plt.title("Datos reales nuevos contagios")
```

Out[4]:

```
Text(0.5, 1.0, 'Datos reales nuevos contagios')
```



Ahora podemos analizar un modelo probabilisto para el examen.

El modelo basado en probabilidad

Para realizar un estimacion del factor de crecimiento de los casos de Covid 19 en Ecuador calculamos la mediana, con esto obtenemo el valor medio de crecimiento de un conjunto de datos, con esto podemos obtener un factor de crecimiento o taza de crecimiento de los nuevos casos.

In [5]:

```
filtro = df["Ecuador"] # Filtro los datos que se empezo a tener casos
#Obtenemos la mediana
media = filtro.mean()
mediana = filtro.median()
print(mediana)
print(media)
```

```
155.0
223.66
```

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

In [6]:

```
#Vamos a comprobar:
# según la media y la mediana podemos obtener la taza de crecieminto y predicir su comportamiento.
# Cargamos los datos de total de casos
```

```
# Cargamos los datos de total de casos
url = 'https://covid.ourworldindata.org/data/ecdc/total_cases.csv'
df_t = pd.read_csv(url)
FMT = '%Y-%m-%d'
date = df_t['date']
df_t['date'] = date.map(lambda x : (datetime.strptime(x, FMT) - datetime.strptime("2019-12-31", FMT
)).days)
df_t = df_t.loc[:,['date','Ecuador']] #Selecciono las columnas de analasis
df_t = df_t.fillna(0)
df_t['Ecuador'].astype(int)
df_t= df_t[df_t["Ecuador"] > 0.0]
y = list(df_t.iloc [:, 1]) # Total casos
x = list(df_t.iloc [:, 0]) # Dias
#Realizamos un ejemplo de prediccion
prediccion_siguiente = int(y[-1] + mediana)
print(prediccion_siguiente)
```

```
11338
```

Practica

1. Comparar el modelo de predicion matematico vs probabilidad.
2. Generar el SIR en base al modelo de probabilidad y obtener beta y gamma con una semana de prediccion.
3. Retroceder un semana y comparar el modelo matematico vs probabilidad vs reales. Solo cargan los datos para generar los modelos menos 7 dias.

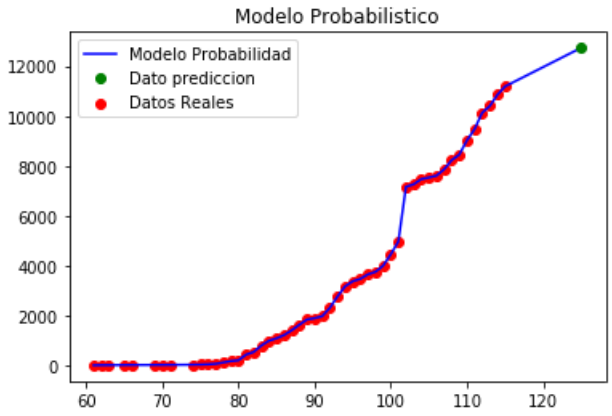
Puntos extras: Investigas sobre la correlacion de variables y aplicar el calculo en base a los datos del Ecuador.

Comparar el modelo de prediccion matemativo vs probabilidad

In [7]:

```
# Quiero predecir cuántos "Casos" voy a obtener de aqui a 10 dias.
plt.scatter(x, y,label="Datos Reales", color='red')
plt.plot()
for i in range(x[-1]+1, x[-1]+11):
    x.append(i)
    y.append(int(y[-1] + mediana))
plt.plot(x,y,label="Modelo Probabilidad", color="blue")
print('En 10 dias el numero de casos es', y[-1])
plt.plot(x[-1],y[-1],'go', label='Dato prediccion')
plt.legend()
plt.title("Modelo Probabilistico")
plt.show()
```

```
En 10 dias el numero de casos es 12733
```

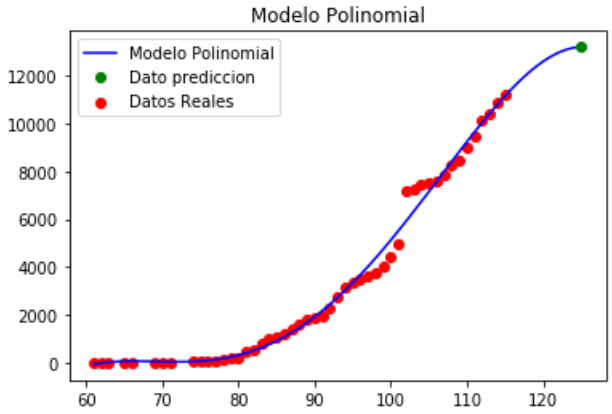


In [8]:

```
#Implementar
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

y_poli = list(df_t.iloc [:, 1]) # Total casos
```

```
y_poli = list(df_t.iloc [:, 1]) # Total casos
x_poli = list(df_t.iloc [:, 0]) # Dias

pf = PolynomialFeatures(degree = 4)     # usaremos polinomios de grado 4
X = pf.fit_transform(np.array(x_poli).reshape(-1, 1))
regresion_lineal = LinearRegression()
regresion_lineal.fit(X, y_poli)
pred_x = list(range(x_poli[0],x_poli[-1]+11))
prediccion = pf.fit_transform(np.array(pred_x).reshape(-1, 1))
prediccion_entrenamiento = regresion_lineal.predict(prediccion)
plt.scatter(x_poli,y_poli,label="Datos Reales",color="red")
plt.plot(pred_x, prediccion_entrenamiento, label='Modelo Polinomial', color='blue')
print('En 10 dias el numero de casos es', round(prediccion_entrenamiento[-1]))
plt.plot(x_poli[-1]+10,round(prediccion_entrenamiento[-1]), 'go', label='Dato prediccion')
plt.legend()
plt.title("Modelo Polinomial")
plt.show()
```

```
En 10 dias el numero de casos es 13204.0
```



Comparacion

Como se puede observar en estos dos graficos tanto del modelo probabilistico como el modelo polinomial el que mas se ajusta es el modelo probabilistico, así que se podría decir que este nos ayuda mas con las predicciones hasta ahora, es el que mas se apega a la realidad.

Generar el SIR en base al modelo de probabilidad y obtener beta y gamma con una semana de prediccion.

In [9]:

```
# Cargamos los datos de total de casos
url = 'https://covid.ourworldindata.org/data/ecdc/total_cases.csv'
df_s = pd.read_csv(url)
FMT = '%Y-%m-%d'
date = df_s['date']
df_s['date'] = date.map(lambda x : (datetime.strptime(x, FMT) - datetime.strptime("2019-12-31", FMT
)).days)
df_s = df_s.loc[:,['date','Ecuador']] #Selecciono las columnas de analasis
df_s = df_s.fillna(0)
df_s['Ecuador'].astype(int)
df_s= df_s[df_s["Ecuador"] > 0.0]
```

In [10]:

```
##obtenemos x y y
y = list(df_s.iloc [:, 1]) # Total casos
x = list(df_s.iloc [:, 0]) # Dias
#Obtenemos la prediccion a una semana
for i in range(x[-1]+1, x[-1]+8):
    x.append(i)
    y.append(int(y[-1] + mediana))
```

In [11]:

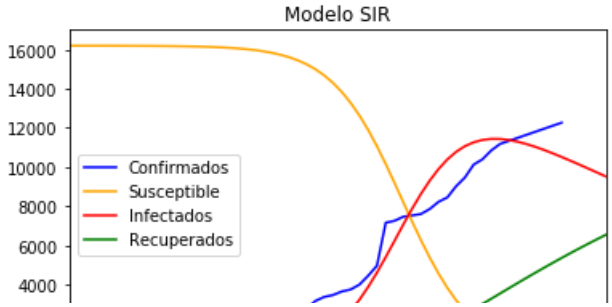
In [11]:

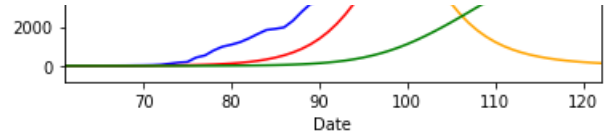
```
##Sir con una semana de prediccion
from scipy.integrate import solve_ivp
from scipy.optimize import minimize
from scipy.integrate import odeint
def loss(point, data, s_0, i_0, r_0):
    size = len(data)
    beta, gamma = point
    def SIR(t, y):
        S = y[0]
        I = y[1]
        R = y[2]
        return [-beta*S*I, beta*S*I-gamma*I, gamma*I]
    solution = solve_ivp(SIR, [0, size], [s_0, i_0, r_0], t_eval=np.arange(0, size, 1), vectorized=
True)
    return np.sqrt(np.mean((solution.y[1] - y)**2))

s_0 = 16200
i_0 = 1
r_0 = 0
data = y
optimal = minimize(
    loss,
    [0.001, 0.001],
    args=(y, s_0, i_0, r_0),
    method='L-BFGS-B',
    bounds=[(0.00000001, 0.4), (0.00000001, 0.4)]
)
beta, gamma = optimal.x
print(f"beta={beta:.8f}, gamma={gamma:.8f}, r_0:{(beta/gamma):.8f}")

def predict(beta, gamma, data, s_0, i_0, r_0):
    new_index = np.array(range(x[0],x[-1]+1))
    size = len(new_index)
    def SIR(t, y):
        S = y[0]
        I = y[1]
        R = y[2]
        return [-beta*S*I, beta*S*I-gamma*I, gamma*I]
    extended_actual = np.concatenate((data, [None] * (size - len(data))))
    return new_index, extended_actual, solve_ivp(SIR, [0, size], [s_0,i_0,r_0], t_eval=np.arange(0,
size, 1))
new_index, extended_actual,prediction = predict(beta, gamma, data,s_0,i_0,r_0)

df_n = pd.DataFrame({
            'Confirmados': extended_actual,
            'Susceptible': prediction.y[0],
            'Infectados': prediction.y[1],
            'Recuperados': prediction.y[2],
            'Date': new_index
        })
ax = plt.gca()
df_n.plot(kind='line',x='Date',y='Confirmados',color='blue', ax=ax)
df_n.plot(kind='line',x='Date',y='Susceptible', color='orange', ax=ax)
df_n.plot(kind='line',x='Date',y='Infectados', color='red', ax=ax)
df_n.plot(kind='line',x='Date',y='Recuperados', color='green', ax=ax)
plt.title("Modelo SIR")
```

```
beta=0.00001707, gamma=0.02344078, r_0:0.00072806
```

Out[11]:

```
Text(0.5, 1.0, 'Modelo SIR')
```





Retroceder un semana y comparar el modelo matematico vs probabilidad vs reales. Solo cargan los datos para generar los modelos menos 7 dias.

Obtenemos los nuevos contagios

In [12]:

```
url = 'https://covid.ourworldindata.org/data/ecdc/new_cases.csv'
df_menos = pd.read_csv(url)
df_menos = df_menos.fillna(0)
df_menos = df_menos.loc[:,['date','Ecuador']] #Selecciono las columnas de analasis
# Expresar las fechas en numero de dias desde el 01 Enero
FMT = '%Y-%m-%d'
date_menos = df_menos['date']
df_menos['date'] = date_menos.map(lambda x : (datetime.strptime(x, FMT) - datetime.strptime("2019-12-31", FMT)).days)
df_menos['Ecuador'].astype(int)
df_menos= df_menos[df_menos["Ecuador"] > 0.0]
##Quitamos 7 dias del regristro
df_menos = df_menos.iloc[0:-7]
```

In [13]:

```
df_menos.plot(x ='date', y='Ecuador')
filtro_menos = df_menos["Ecuador"] # Selecciono solo los casos
#Obtenemos la mediana
media_menos = filtro_menos.mean()
mediana_menos = filtro_menos.median()
plt.title('Datos Reales de nuevos casos')
print(mediana_menos)
print(media_menos)
```

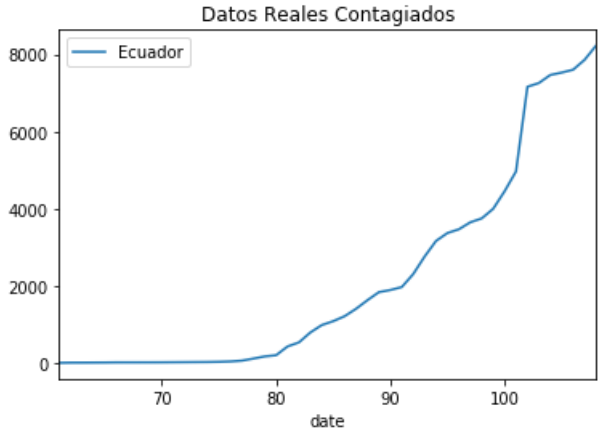
```
101.0
191.27906976744185
```

In [14]:

```
## Leo los contagiados y hago la prediccion
url = 'https://covid.ourworldindata.org/data/ecdc/total_cases.csv'
df_total = pd.read_csv(url)
FMT = '%Y-%m-%d'
date = df_total['date']
df_total['date'] = date.map(lambda x : (datetime.strptime(x, FMT) - datetime.strptime("2019-12-31", FMT)).days)
df_total = df_total.loc[:,['date','Ecuador']] #Selecciono las columnas de analasis
df_total = df_total.fillna(0)
df_total['Ecuador'].astype(int)
```

```
df_total= df_total[df_total["Ecuador"] > 0.0]
yR = list(df_total.iloc [:, 1]) # Total casos
xR = list(df_total.iloc [:, 0]) # Dias
# Resto 7 dias
df_total = df_total.iloc[0:-7]
#################################################################################
y_menos = list(df_total.iloc [:, 1]) # Total casos
x_menos = list(df_total.iloc [:, 0]) # Dias
#Realizamos un ejemplo de prediccion
prediccion = int(y_menos[-1]+ mediana_menos)
print("Prediccion:",prediccion)
df_total.plot(x ='date', y='Ecuador')
plt.title('Datos Reales Contagiados')
print(mediana_menos)
print(media_menos)
```

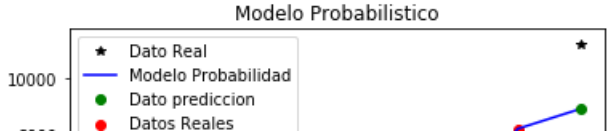
```
Prediccion: 8326
101.0
191.27906976744185
```

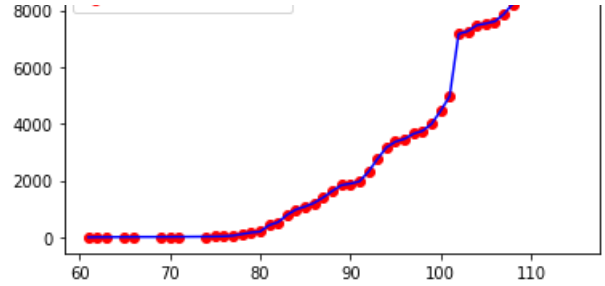


In [15]:

```
# Quiero predecir cuántos "Casos" voy a obtener de aqui a 7 dias.
plt.scatter(x_menos, y_menos,label="Datos Reales", color='red')
plt.plot(xR[-1],yR[-1],'k*', label='Dato Real')
plt.plot()
for i in range(x_menos[-1]+1, x_menos[-1]+8):
    x_menos.append(i)
    y_menos.append(int(y_menos[-1] + mediana_menos))
plt.plot(x_menos,y_menos,label="Modelo Probabilidad", color="blue")
lista = np.array(range(2,9))
cont = 1
for i in reversed(lista):
    print('Prediccion de casos en',xR[-1]+cont,'dia(s)', (y_menos[-i])+ mediana_menos)
    cont+=1
df_real = df_s['Ecuador']
print('En',xR[-1]+7,'dias el numero de real de casos fue',df_real.iloc[-1])
plt.plot(x_menos[-1],y_menos[-1],'go', label='Dato prediccion')
plt.legend()
plt.title('Modelo Probabilistico')
plt.show()
```

```
Prediccion de casos en 116 dia(s) 8326.0
Prediccion de casos en 117 dia(s) 8427.0
Prediccion de casos en 118 dia(s) 8528.0
Prediccion de casos en 119 dia(s) 8629.0
Prediccion de casos en 120 dia(s) 8730.0
Prediccion de casos en 121 dia(s) 8831.0
Prediccion de casos en 122 dia(s) 8932.0
En 122 dias el numero de real de casos fue 11183.0
```

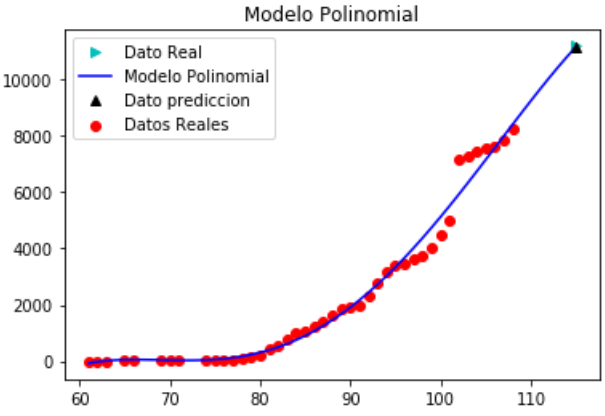




In [16]:

```
#Implementar
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

y_polim = list(df_total.iloc [:, 1]) # Total casos
x_polim = list(df_total.iloc [:, 0]) # Dias
polinomio = PolynomialFeatures(degree = 4)     # usaremos polinomios de grado 4
XM = polinomio.fit_transform(np.array(x_polim).reshape(-1, 1))
regresion_linealM = LinearRegression()
regresion_linealM.fit(XM, y_polim)
pred_xm = list(range(x_polim[0],x_polim[-1]+8))
prediccionM = polinomio.fit_transform(np.array(pred_xm).reshape(-1, 1))
prediccion_entrenamientoM = regresion_lineal.predict(prediccionM)
plt.scatter(x_polim,y_polim,label="Datos Reales",color="red")
plt.plot(xR[-1],yR[-1], 'c>', label='Dato Real')
plt.plot(pred_xm, prediccion_entrenamientoM, label='Modelo Polinomial', color='blue')
plt.plot(x_polim[-1]+7,round(prediccion_entrenamientoM[-1]), '^k', label='Dato prediccion')
listaP = np.array(range(1,8))
contP = 1
for j in reversed(listaP):
    print('Prediccion de casos en',xR[-1]+contP,'dia(s) el numero de posibles casos es', round(pred
iccion_entrenamientoM[-j]))
    contP+=1
df_real = df_s['Ecuador']
print('En',xR[-1]+7,'dias el numero de real de casos fue',df_real.iloc[-1])
plt.legend()
plt.title('Modelo Polinomial')
plt.show()
```

```
Prediccion de casos en 116 dia(s) el numero de posibles casos es 8820.0
Prediccion de casos en 117 dia(s) el numero de posibles casos es 9233.0
Prediccion de casos en 118 dia(s) el numero de posibles casos es 9638.0
Prediccion de casos en 119 dia(s) el numero de posibles casos es 10035.0
Prediccion de casos en 120 dia(s) el numero de posibles casos es 10421.0
Prediccion de casos en 121 dia(s) el numero de posibles casos es 10792.0
Prediccion de casos en 122 dia(s) el numero de posibles casos es 11148.0
En 122 dias el numero de real de casos fue 11183.0
```



Correlacion de variables

Para cuantificar las relaciones entre variables, podemos utilizar el Coeficiente de Correlación de Pearson. Esta es una medida de la fuerza y dirección de una relación lineal entre dos variables. Una puntuación de +1 es una relación positiva perfectamente lineal y

una puntuación de -1 es una relación lineal perfectamente negativa.

Correlacion de variable con el CSV de nuevos casos

In [17]:

```
url = 'https://covid.ourworldindata.org/data/ecdc/new_cases.csv'
#url = 'https://covid.ourworldindata.org/data/ecdc/total_cases.csv'
df_correlacion = pd.read_csv(url)
df_correlacion = df_correlacion.fillna(0)
```

In [18]:

```
df_correlacion = df_correlacion.loc[:,['Argentina','Ecuador','United States']] #Selecciono las
columnas de analasis
#FMT = '%Y-%m-%d'
#date = df_correlacion['date']
#df_correlacion['date'] = date.map(lambda x : (datetime.strptime(x, FMT) -
datetime.strptime("2019-12-31", FMT)).days)
df_correlacion['Ecuador'].astype(int)
df_correlacion = df_correlacion[df_correlacion["Ecuador"] > 0.0]
```

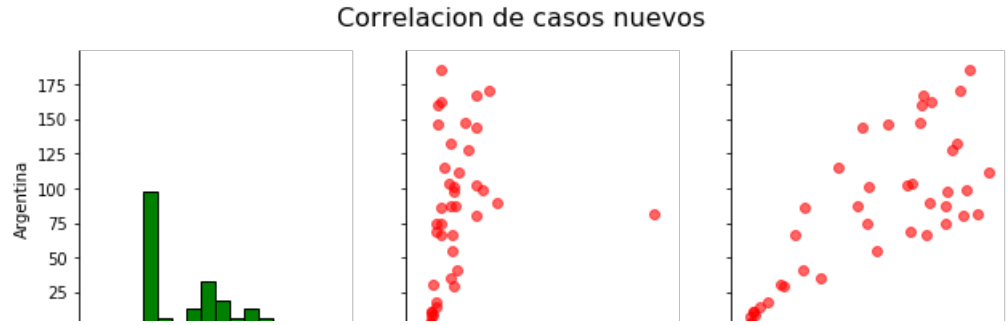
In [19]:

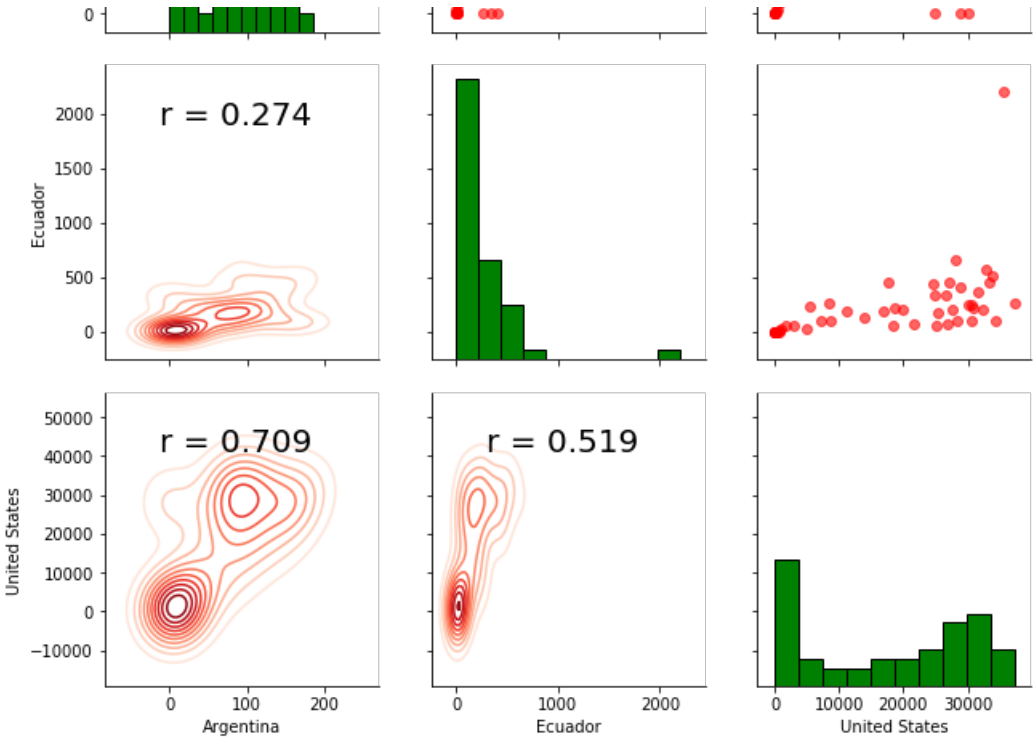
```
correlacion = df_correlacion.corr(method='pearson', min_periods=1)
print(correlacion)
#plt.matshow(df_correlacion.corr(method='pearson', min_periods=1))
```

```
               Argentina   Ecuador  United States
Argentina       1.000000  0.274364       0.708709
Ecuador         0.274364  1.000000       0.518862
United States   0.708709  0.518862       1.000000
```

In [25]:

```
import seaborn as sns
# Función para calcular el coeficiente de correlación entre dos columnas
def corr_func(x, y, **kwargs):
    r = np.corrcoef(x, y)[0][1]
    ax = plt.gca()
    ax.annotate("r = {:.3f}".format(r),
                xy=(.2, .8), xycoords=ax.transAxes,
                size = 20)
# Crear el objeto pairgrid
grid = sns.PairGrid(data = df_correlacion, size = 3)

# Upper es un scatter plot (es un diagrama de dispersión)
grid.map_upper(plt.scatter, color = 'red', alpha = 0.6)

# Diagonal es un histograma
grid.map_diag(plt.hist, color = 'green', edgecolor = 'black')

# La parte inferior es un gráfico de correlación y densidad
grid.map_lower(corr_func);
grid.map_lower(sns.kdeplot, cmap = plt.cm.Reds)

# Titulo para todo el gráfico
plt.suptitle('Correlacion de casos nuevos', size = 16, y = 1.02);
```





Correlacion de variable con el CSV de total de casos confirmados

In [26]:

```
url = 'https://covid.ourworldindata.org/data/ecdc/total_cases.csv'
df_correlacionT = pd.read_csv(url)
df_correlacionT = df_correlacionT.fillna(0)
```

In [27]:

```
df_correlacionT = df_correlacionT.loc[:,['Argentina','Ecuador','United States']] #Selecciono las c
olumnas de analasis
#FMT = '%Y-%m-%d'
#dateT = df_correlacionT['date']
#df_correlacionT['date'] = date.map(lambda x : (datetime.strptime(x, FMT) -
datetime.strptime("2019-12-31", FMT)).days)
df_correlacionT['Ecuador'].astype(int)
df_correlacionT = df_correlacionT[df_correlacionT["Ecuador"] > 0.0]
```

In [28]:

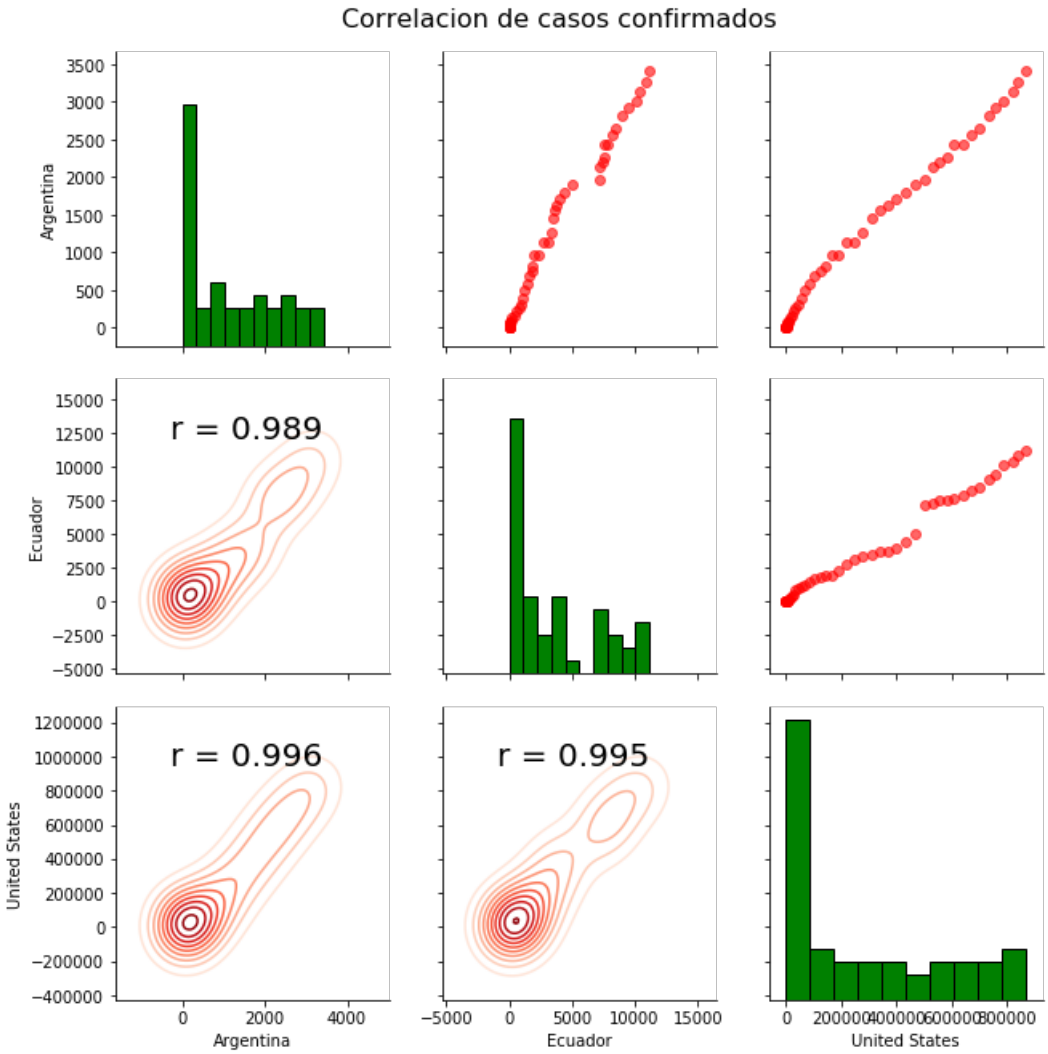
```
correlacionT = df_correlacionT.corr(method='pearson', min_periods=1)
print(correlacionT)
#plt.matshow(df_correlacionT.corr(method='pearson', min_periods=1))
```

```
               Argentina   Ecuador  United States
Argentina       1.000000  0.989455       0.995646
Ecuador         0.989455  1.000000       0.994826
United States   0.995646  0.994826       1.000000
```

In [29]:

```
import seaborn as sns
# Función para calcular el coeficiente de correlación entre 2 paises
def corr_func(x, y, **kwargs):
    r = np.corrcoef(x, y)[0][1]
    ax = plt.gca()
    ax.annotate("r = {:.3f}".format(r),
                xy=(.2, .8), xycoords=ax.transAxes,
                size = 20)

# Crear el objeto pairgrid
grid = sns.PairGrid(data = df_correlacionT, size = 3)
```

```
# Upper es un scatter plot (es un diagrama de dispersión)
grid.map_upper(plt.scatter, color = 'red', alpha = 0.6)

# Diagonal es un histograma
grid.map_diag(plt.hist, color = 'green', edgecolor = 'black')

# La parte inferior es un gráfico de correlación y densidad
grid.map_lower(corr_func);
grid.map_lower(sns.kdeplot, cmap = plt.cm.Reds)

# Titulo para todo el gráfico
plt.suptitle('Correlacion de casos confirmados', size = 16, y = 1.02);
```



Analisis

- La librería pandas en muy útil ya que tiene muchas funcionalidades como son las de filtrar datos de una forma sencilla, para tan solo obtener los datos que son necesarios para el analisis, asi mismo como el graficar mediante un plot, con un DataFrame es mucho mas sencillo y corto el proceso.
- Un vez que obtuvimos todas las gráficas se puede hacer un análisis de cada uno de los resultados, cabe mencionar que el modelo polinomial y el exponencial son los que más se acercan a la realidad del Ecuador, pero la ventaja del modelo polinomial es que por el grado que se use, permite acercarse mas a la realidad.
- El método probabilistico, a más de ser sencillo de calcular, se ajusta muy bien a la realidad del país, pero con la desventaja de que con crecimientos drásticos en los datos reales, la prediccion no es tan certera.

Conclusiones

- A pesar de que el modelo probabilistico, inicialmente es el que mas se ajusta a los datos reales, cuando existe un crecimiento exabrupto por así decirlo, este modelo no es capaz de precisarlo, y el modelo polinomial si, se acerca más en cuestión de predicciones y va a depender mucho el grado del polinomio con el que se trabaje.
- En cuestion de la correlación se necesita saber cuales son los datos que se desean comparar y sacar una relación entre ellos,

para asi saber que datos se deben obtener y analizar.

Criterio personal (politico, economico y social de la situacion)

- Al ver las gráficas y tambien la correlación de los datos, se puede apreciar que hay una fuerte realación entre Ecuador y Estados Unidos por lo que viendo la situación actual del país norteamericano, Ecuador a pesar de las medidas que ha tomado no ha logrado mantener o controlar la propagación del virus, la economía en algunos sectores sigue bajando, mientras que en otros se podría decir que se mantiene o a su vez a mejorado, me refiero a los sectores facmacéuticos y de comida ya que ellos siguen laborando y generando ingresos

Referencias

- https://www.researchgate.net/publication/340092755_Infeccion_del_Covid-19_en_Colombia_Una_comparacion_de_modelos_logisticos_y_exponenciales_aplicados_a_la_infeccion_por_el_virus_en_Colom
- https://www.aprendemachinelearning.com/regresion-lineal-en-espanol-con-python/

Correlacion

- https://blog.gcoding.academy/un-proyecto-completo-de-machine-learning-en-python-primera-parte/