In [17]:

```
import pandas as pd
import numpy as np
import time
import datetime
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from datetime import datetime,timedelta
from scipy.optimize import curve_fit
from scipy.optimize import fsolve
from sklearn import linear_model
import matplotlib.pyplot as plt
%matplotlib inline
```

In [18]:

```
url = 'Casos covid por provincias.xlsx'
df = pd.read_excel(url)
df = df.fillna(0)
df
```

Out[18]:

	Provincia	16/3/2020	17/3/2020	18/3/2020	19/3/2020	20/3/2020	21/3/2020	22/3/2020	23/3/2020	24/3/2020	...	13/4/2020	14/4/2020
0	Azuay	1.0	5.0	5.0	14.0	18.0	19.0	19.0	23.0	28.0	...	182	192
1	Bolivar	0.0	0.0	2.0	2.0	4.0	5.0	8.0	9.0	9.0	...	33	33
2	Cañar	0.0	0.0	0.0	3.0	3.0	3.0	4.0	5.0	7.0	...	100	104
3	Carchi	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	25	26
4	Chimborazo	0.0	0.0	0.0	2.0	3.0	4.0	9.0	9.0	11.0	...	85	86
5	Cotopaxi	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	...	42	43
6	El Oro	0.0	1.0	1.0	1.0	2.0	2.0	6.0	9.0	14.0	...	160	166
7	Esmeraldas	0.0	0.0	0.0	0.0	0.0	2.0	2.0	3.0	3.0	...	38	41
8	Galápagos	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	...	11	12
9	Guayas	37.0	81.0	128.0	187.0	318.0	397.0	607.0	769.0	826.0	...	5395	5417
10	Imbabura	0.0	0.0	0.0	1.0	2.0	3.0	4.0	4.0	4.0	...	33	33
11	Loja	0.0	0.0	0.0	4.0	5.0	5.0	5.0	5.0	7.0	...	69	72
12	Los Ríos	10.0	10.0	10.0	16.0	19.0	22.0	25.0	28.0	37.0	...	239	240
13	Manabí	1.0	1.0	8.0	8.0	9.0	9.0	25.0	27.0	32.0	...	215	219
14	Morona\n Santiago	0.0	1.0	1.0	3.0	3.0	3.0	3.0	6.0	6.0	...	20	21
15	Napo	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	6	8
16	Orellana	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	6	6
17	Pastaza	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	13	14
18	Pichincha	8.0	8.0	12.0	16.0	35.0	50.0	60.0	65.0	72.0	...	634	646
19	SantaElena	0.0	3.0	0.0	0.0	1.0	1.0	2.0	4.0	6.0	...	78	78
20	Santo Domingo	0.0	0.0	0.0	1.0	1.0	4.0	4.0	7.0	7.0	...	65	66
21	Sucumbíos	1.0	1.0	1.0	2.0	3.0	3.0	6.0	6.0	6.0	...	33	33
22	Tungurahua	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	2.0	...	43	43
23	Zamora\n Chinchipe	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	4	4

24 rows × 39 columns

In [20]:

```
#df = df.loc[:,['Provincia']] #Seleccione las columnas de analasis
```

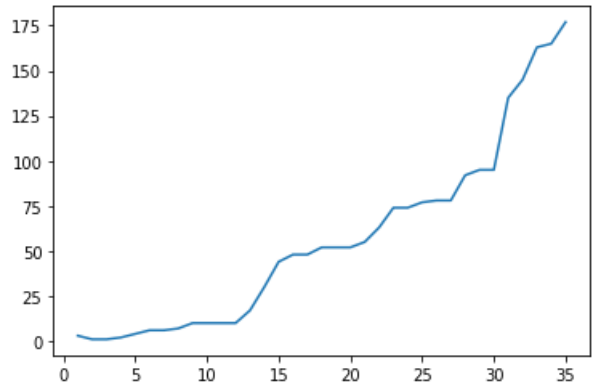
```
#df = df.loc[:,['Provincia']] #Selecciono las columnas de analasis
country = 'Santa Elena '
start_date = '17/3/2020'
country_df = df[df['Provincia'] == 'SantaElena']
country_df = country_df.drop('16/3/2020',1)
country_df = country_df.drop('18/3/2020',1)
country_df = country_df.drop('19/3/2020',1)
filtro = country_df.iloc[0].loc[start_date:]
```

In [21]:

```
x = np.array(range(1,len(filtro)+1))
y = filtro
plt.plot(x,y)
```

Out[21]:

```
[<matplotlib.lines.Line2D at 0x273e2f34cc8>]
```



Modelo Lineal

In [22]:

```
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:
 [4.78571429]
Independent term:
 -29.599999999999994
```

In [23]:

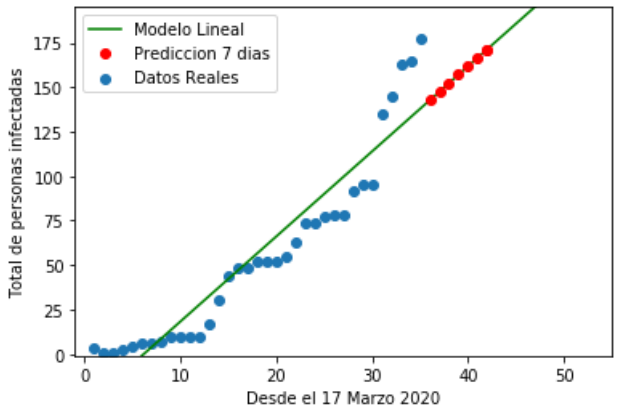
```
for i in range(2,9):
    y_prediccion = regr.predict([[len(filtro)+i]])
    print(int(y_prediccion))
```

```
147
152
157
161
166
171
176
```

In [24]:

```
plt.scatter(x, y,label="Datos Reales")
```

```
x_real = np.array(range(1, 50))
plt.plot(x_real, regr.predict(x_real.reshape(-1, 1)), color='green', label='Modelo Lineal')
for i in range(1,8):
    y_prediccion = regr.predict([[len(filtro)+i]])
    plt.plot([len(filtro)+i],y_prediccion, 'or')
plt.plot(42,y_prediccion, 'or', label='Prediccion 7 dias')
plt.legend()
plt.xlabel("Desde el 17 Marzo 2020")
plt.ylabel("Total de personas infectadas")
plt.ylim(-1,195) # Definir los limites de Y
plt.xlim(-1,55)
plt.show()
```



Modelo Logistico

In [25]:

```
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)
```

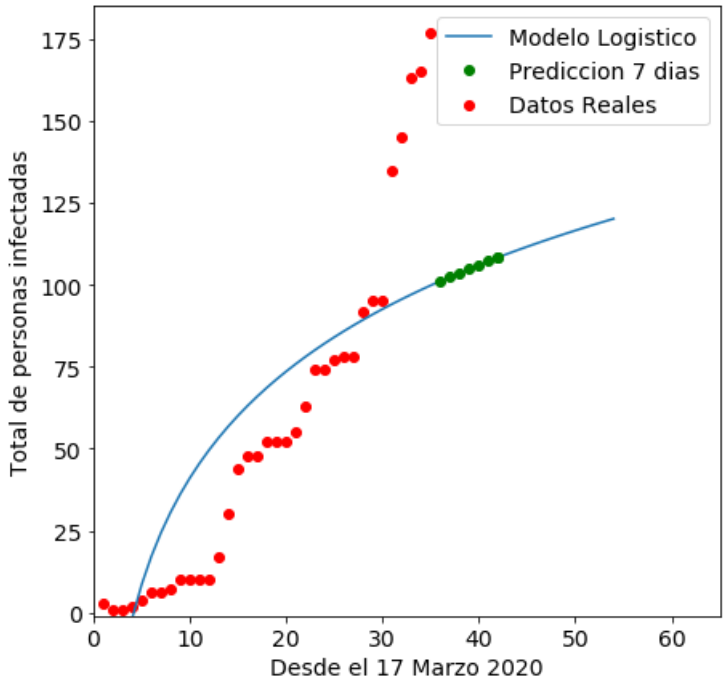
```
(array([-66.88325082,  46.88618505]), array([[ 327.3248515 , -112.58451822],
       [-112.58451822,   42.76776317]]))
```

In [26]:

```
pred_x = list(range(min(x),max(x)+20)) # Predecir 20 dias mas
print(pred_x)
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="Modelo L
ogistico" )
for i in range(1,8):
    print("Total infectados del dia ",len(filtro)+i,modelo_logistico(len(filtro)+i,exp_fit[0][0],ex
p_fit[0][1]))
    plt.plot([len(filtro)+i],modelo_logistico(len(filtro)+i,exp_fit[0][0],exp_fit[0][1]),'go')
plt.plot(42,modelo_logistico(42,exp_fit[0][0],exp_fit[0][1]),'go', label="Prediccion 7 dias")
plt.legend()
plt.xlabel("Desde el 17 Marzo 2020")
plt.ylabel("Total de personas infectadas")
plt.ylim(-1,185) # Definir los limites de Y
plt.xlim(0,65)
plt.show()
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27
, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
52, 53, 54]
Total infectados del dia  36 101.13428124128006
Total infectados del dia  37 102.41891461511776
Total infectados del dia  38 103.66928698264806
```

```
Total infectados del dia  39 104.88717844479524
Total infectados del dia  40 106.07423387489024
Total infectados del dia  41 107.23197627805644
Total infectados del dia  42 108.361818540533
```



Modelo Exponencial

In [27]:

```
# Implementar
def modelo_exponencial(x, a, b,c):
    #return (a*np.exp(b*x-50)
    return a * np.exp(b*(x - c))

popt, pcov = curve_fit(modelo_exponencial, x, y)
print(popt)
```

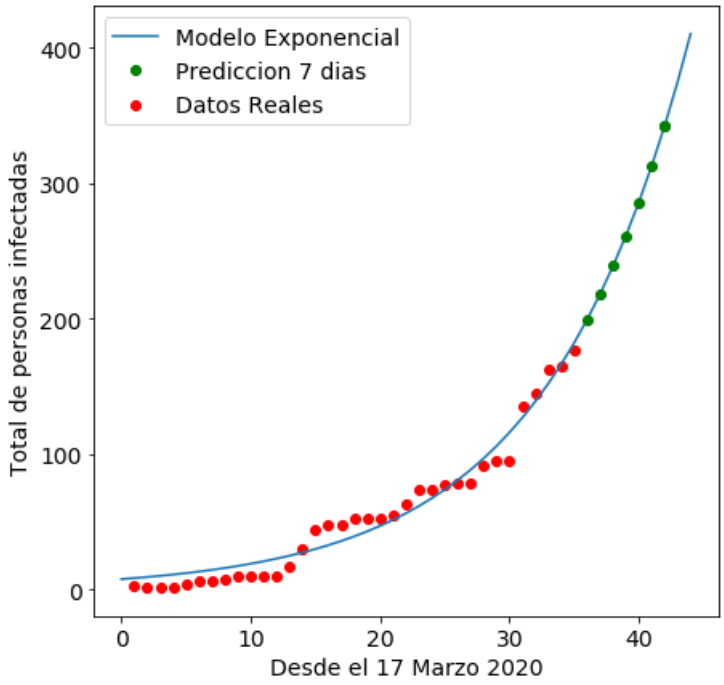
```
[12.7960552   0.0902575   5.58437858]
```

In [28]:

```
pred_x = np.array(range(0,45)) # Predecir
plt.rcParams['figure.figsize'] = [7, 7]
plt.rc('font', size=14)
plt.scatter(x,y,label="Datos Reales",color="red")
plt.plot(pred_x, [modelo_exponencial(i,popt[0],popt[1],popt[2]) for i in pred_x], label="Modelo Exp
onencial" )
print("Modelo de prediccion EXPONENCIAL")
for j in range(1,8):
    print("Total infectados del dia :",35+j,round(modelo_exponencial(35+j,popt[0],popt[1],popt[2]))
)
    plt.plot((35+j), round(modelo_exponencial(35+j,popt[0],popt[1],popt[2])), 'go')
plt.plot(42, round(modelo_exponencial(42,popt[0],popt[1],popt[2])), 'go', label='Prediccion 7
dias')
plt.legend()
plt.xlabel("Desde el 17 Marzo 2020")
plt.ylabel("Total de personas infectadas")
plt.show()
```

```
Modelo de prediccion EXPONENCIAL
Total infectados del dia : 36 199.0
Total infectados del dia : 37 218.0
Total infectados del dia : 38 239.0
Total infectados del dia : 39 261.0
Total infectados del dia : 40 286.0
```

```
Total infectados del dia : 41 313.0
Total infectados del dia : 42 342.0
```

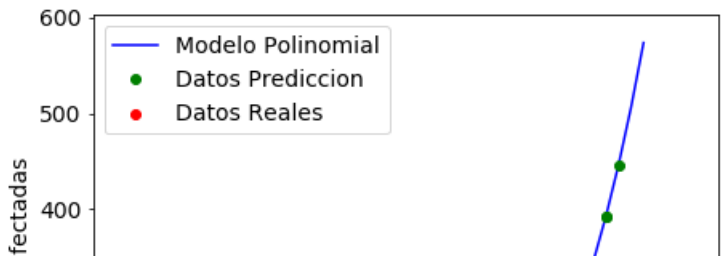


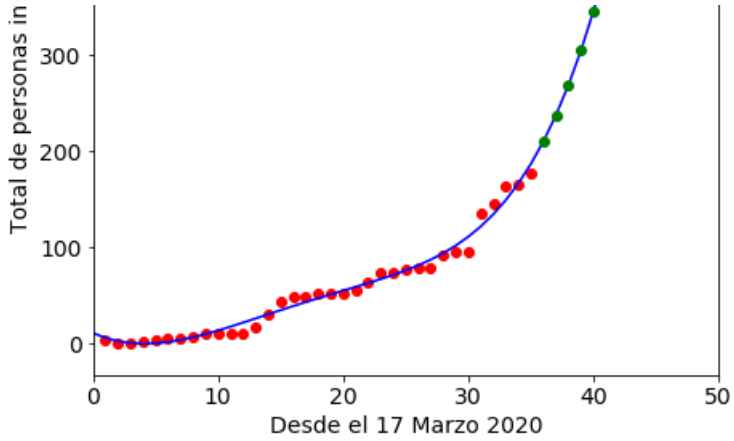
Polinomial

In [30]:

```
pf = PolynomialFeatures(degree = 5)     # usaremos polinomios de grado 3
X = pf.fit_transform(np.array(x).reshape(-1, 1))

regresion_lineal = LinearRegression()

regresion_lineal.fit(X, y)
pred_x = np.array(range(0,45))

prediccion = pf.fit_transform(np.array(pred_x).reshape(-1, 1))
prediccion_entrenamiento = regresion_lineal.predict(prediccion)
plt.plot(pred_x, prediccion_entrenamiento, color='blue', label='Modelo Polinomial')
for j in range(1,8):
    plt.plot((35+j), round(prediccion_entrenamiento[len(filtro)+j]), 'go')
    print("La predicción dentro de",35+j, "días es: ",round(prediccion_entrenamiento[35+j]))
plt.scatter(x,y,label="Datos Reales",color="red")
plt.plot(x[len(x)-1]+6,prediccion_entrenamiento[x[len(x)-1]+6], 'og', label="Datos Prediccion")
plt.xlim(0,50)
plt.legend()
plt.xlabel("Desde el 17 Marzo 2020")
plt.ylabel("Total de personas infectadas")
plt.show()
```

```
La predicción dentro de 36 días es:  210.0
La predicción dentro de 37 días es:  238.0
La predicción dentro de 38 días es:  269.0
La predicción dentro de 39 días es:  305.0
La predicción dentro de 40 días es:  346.0
La predicción dentro de 41 días es:  393.0
La predicción dentro de 42 días es:  446.0
```





Modelo SIR

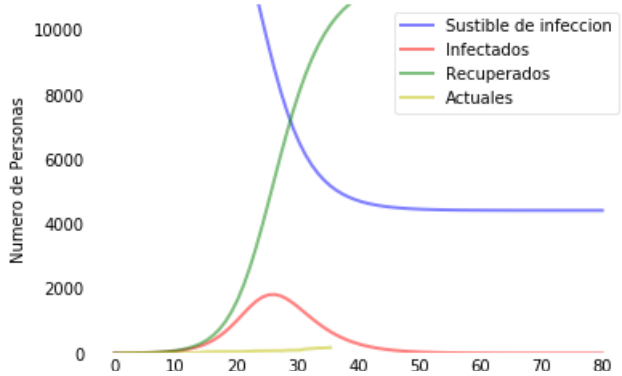
In [31]:

```
import numpy as np
import pandas as pd
from csv import reader
from csv import writer
from scipy.integrate import solve_ivp
from scipy.optimize import minimize
import matplotlib.pyplot as plt
from datetime import timedelta, datetime
import argparse
import sys
import json
import ssl
import urllib.request
%matplotlib inline

class Confirmados(object):
    def __init__(self, country, loss, start_date,s_0, i_0, r_0):
        self.country = country
        self.loss = loss
        self.start_date = start_date
        self.s_0 = s_0
        self.i_0 = i_0
        self.r_0 = r_0

    def train(self):
        data = y
        optimal = minimize(
            loss,
            [0.001, 0.001],
            args=(y, self.s_0, self.i_0, self.r_0),
            method='L-BFGS-B',
            bounds=[(0.00000001, 0.4), (0.00000001, 0.4)]
        )

        beta, gamma = optimal.x
        print(f"country={self.country}, beta={beta:.8f}, gamma={gamma:.8f}, r_0:{(beta/gamma):.8f}'
)

def loss(point, y, s_0, i_0, r_0):
    size = len(y)
    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))

confirmados = Confirmados('SantaElena', loss, '17/3/2020', 16000, 3, 0)
confirmados.train()
```

```
country=SantaElena, beta=0.40000000, gamma=0.40000000, r_0:1.00000000
```

In [34]:

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

# Total de la poblacion
N = 9003
# Numero Inicial de Infectados
I0 = 3
# Numero de Recuperados
R0 = 0
# Todos los demás, S0, son susceptibles a la infección inicialmente.
S0 = 16000
# Tasa de contacto, beta (nivel de repoductividad del virus)
# La tasa de recuperación media, gamma,(1/días) Una persona se recupera en 15 dias.
beta, gamma = 0.4, 0.4
# Una cuadrícula de puntos de tiempo (en días)
t = np.linspace(0, 80, 80)
A = np.concatenate((y, [None] * (len(filtro) - len(y))))
xa=t[1:36]

# Las ecuaciones diferenciales del modelo SIR..
def deriv(y, t, N, beta, gamma):
    S, I, R = y
    dSdt = -beta * S * I / N
    dIdt = beta * S * I / N - gamma * I
    dRdt = gamma * I
    return dSdt, dIdt, dRdt

# Vector de condiciones iniciales
y0 = S0, I0, R0
# Integre las ecuaciones SIR en la cuadrícula de tiempo, t. A traves de la funcion odeint()
ret = odeint(deriv, y0, t, args=(N, beta, gamma))
S, I, R = ret.T # Obtenicion de resultados


#  Trace los datos en tres curvas separadas para S (t), I (t) y R (t)
fig = plt.figure(facecolor='w')
ax = fig.add_subplot(111,  axisbelow=True)
ax.plot(t, S, 'b', alpha=0.5, lw=2, label='Sustible de infeccion')
ax.plot(t, I, 'r', alpha=0.5, lw=2, label='Infectados')
ax.plot(t, R, 'g', alpha=0.5, lw=2, label='Recuperados')
ax.plot(xa, y, 'y', alpha=0.5, lw=2, label='Actuales')
ax.set_xlabel('Tiempo en dias')
ax.set_ylabel('Numero de Personas')
ax.set_ylim(0,N*1.2)
ax.yaxis.set_tick_params(length=0)
ax.xaxis.set_tick_params(length=0)
ax.grid(b=True, which='major', c='w', lw=2, ls='-')
legend = ax.legend()
legend.get_frame().set_alpha(0.5)
for spine in ('top', 'right', 'bottom', 'left'):
    ax.spines[spine].set_visible(False)
plt.show()

Ro = beta/gamma
print('R0=',Ro)
```




```
R0= 1.0
```

In [35]:

```
Re = Ro*2500
print(Re)
Re = Re/1000
print(Re)
```

```
2500.0
2.5
```

Simulacion

In [2]:

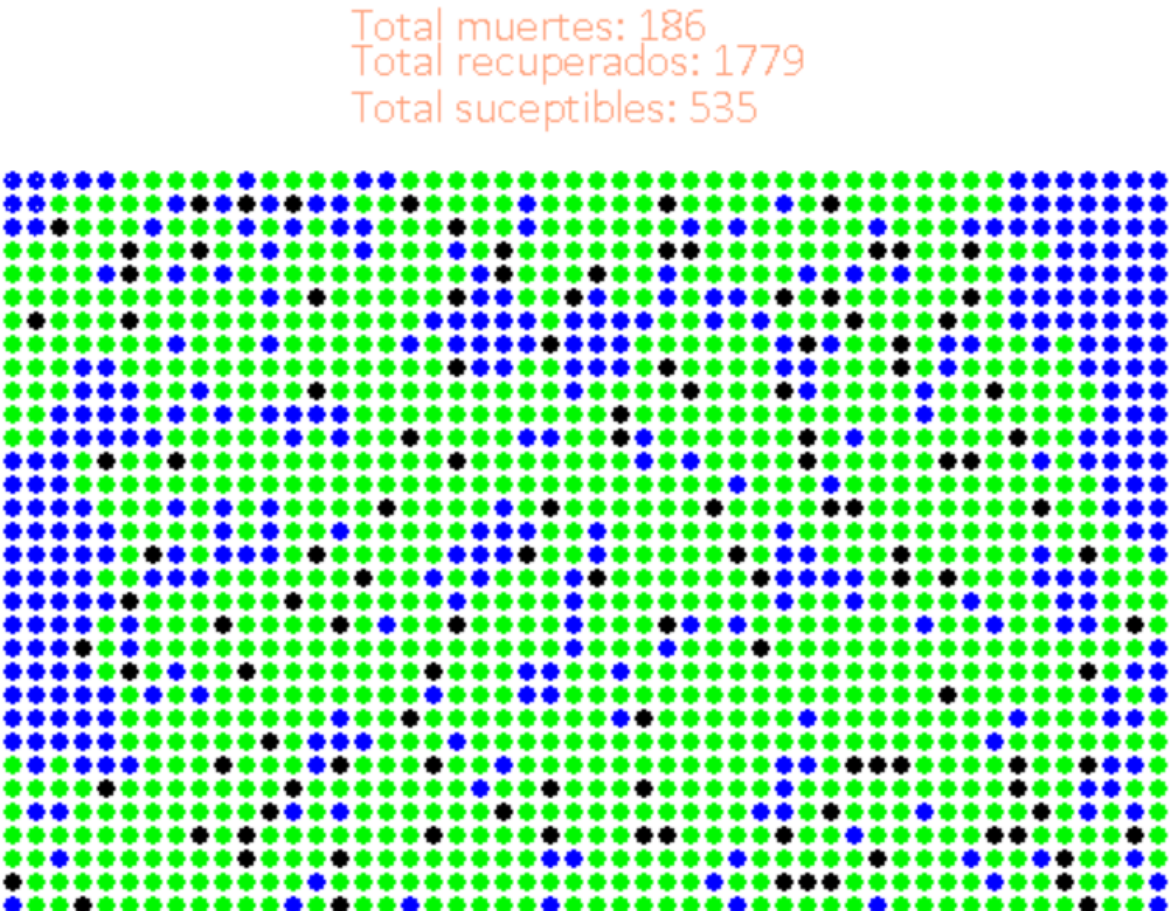
```
from random import randrange # Obtener un numero randomico
import pygame

#Parametros de inicio
PROBA_MUERTE = 8.4  # Probabilidad de que la gente muera COVID
CONTAGION_RATE = 2.5  # Factor R0 para la simulacion COVID probabilidad
PROBA_INFECT = CONTAGION_RATE * 10
PROBA_VACU = 0 # Probabilidad de que exista una vacuna, COVID = 0
SIMULACION_SPEED = 50 # Tiempo de un dia en milisegundos (Cada 25 es un dia)
nb_rows = 50#205 #Numero de filas
nb_cols = 50#100 #Numero de columnas

global display, myfont, states, states_temp #Declaracion de variables globales

#Declaro colores en formato RGB
WHITE = (255, 255, 255)
BLUE = (0, 0, 255)
GREEN = (0, 247, 0)
BLACK = (0, 0, 0)

#Obtiene los vecinos dado un punto x,y
def get_vecinos(x, y):
    incx = randrange(3)
    incy = randrange(3)
    incx = (incx * 1) - 1
    incy = (incy * 1) - 1
    x2 = x + incx
    y2 = y + incy
    #Validar limites
    if x2 < 0:
        x2 = 0
    if x2 >= nb_cols:
        x2 = nb_cols - 1
    if y2 < 0:
        y2 = 0
    if y2 >= nb_rows:
        y2 = nb_rows - 1
    return [x2, y2] # Nuevos contagiados

#Genero las personas que cuentan con inmunidad o vacuna
def vacunar():
    for x in range(nb_cols):
        for y in range(nb_rows):
            if randrange(99) < PROBA_VACU:
                states[x][y] = 1

#Funcion que permite contar el numero de muertosde la matriz states == -1
def contar_muertes():
    contador = 0
    for x in range(nb_cols):
        for y in range(nb_rows):
            if states[x][y] == -1:
                contador +=  1
    return contador
```

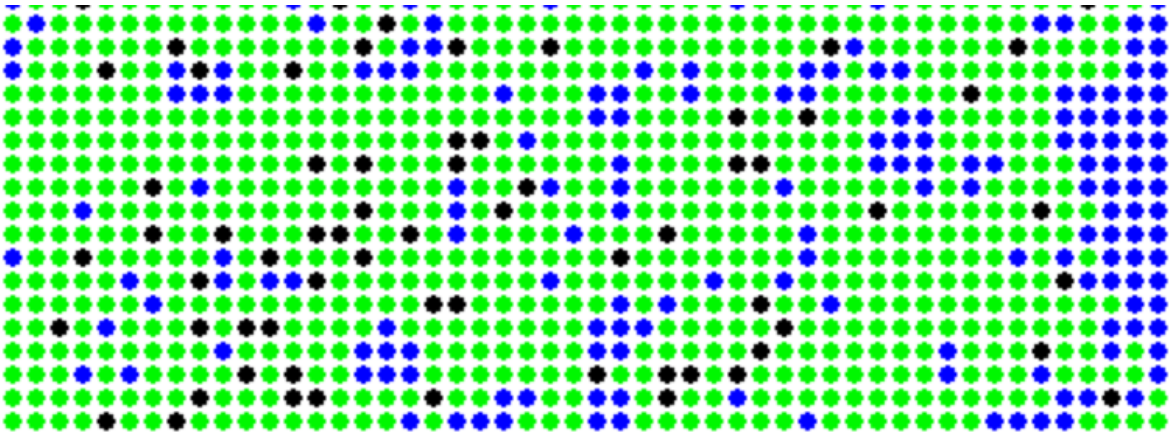
```
def contar_suceptible():
    contador = 0
    for x in range(nb_cols):
        for y in range(nb_rows):
            if states[x][y] <= 20:
                contador +=  1
    return contador


def contar_recuperados():
    contador = 0
    for x in range(nb_cols):
        for y in range(nb_rows):
            if states[x][y] == 1:
                contador +=  1
    return contador
#Definimos datos de inicio
states = [[0] * nb_cols for i1 in range(nb_rows)]
states_temp = states.copy()
states[randrange(50)][randrange(50)] = 10 # Estado inicial de la simulacion Posicion del Infectado
it = 0 # Variable para contar las Iteraciones
total_muerte = 0 # Contabiliza el numero de muertos
vacunar() #Llamar a la funcion vacunar

pygame.init() #Incializo el motor de juegos pygame
pygame.font.init() #Inicializo el tipo de letra
display=pygame.display.set_mode((750,650),0,32) #Tamanio de la ventana
pygame.display.set_caption("Simulacion de Epidemia Covid-19 Ecuador")# Titulo
font=pygame.font.SysFont('Calibri', 20) # Tipo de letra
display.fill(WHITE) # Color de fondo

while True:
    pygame.time.delay(SIMULACION_SPEED) # Sleep o pausa
    it = it + 1
    if it <= 10000 and it >= 2:
        states_temp = states.copy() #Copia de la matriz
        #Recorrera la matriz
        for x in range(nb_cols):
            for y in range(nb_rows):
                state = states[x][y]
                if state == -1:
                    pass
                if state >= 10: # Numero de dias de contagio
                    states_temp[x][y] = state + 1
                if state >= 20:
                    if randrange(99) < PROBA_MUERTE: # Genero un randomico para verificar si
fallece o se recupera
                        states_temp[x][y] = -1 # Muere
                    else:
                        states_temp[x][y] = 1 # Cura o recupera
                if state >= 10 and state <= 20: # Rango de infectado
                    if randrange(99) < PROBA_INFECT: # Infecto a las personas cercanas entre  10 y
20
                        neighbour = get_vecinos(x, y) #Obtenemos los vecinos a contagiar
                        x2 = neighbour[0]
                        y2 = neighbour[1]
                        neigh_state = states[x2][y2]
                        if neigh_state == 0: #Verifico que este sano
                            states_temp[x2][y2] = 10 # Contagia
        states = states_temp.copy()
        total_muerte = contar_muertes() # contar el numero de muertos

    pygame.draw.rect(display, WHITE, (250, 30, 260, 50)) # Grafico el fondo
    textsurface = font.render("Total muertes: "+ str(total_muerte), False, (255,160,122))
    textsurface1 = font.render("Total recuperados: "+ str(contar_recuperados()), False, (255,160,12
2))
    textsurface2 = font.render("Total suceptibles: "+ str(contar_suceptible()-contar_muertes()-cont
ar_recuperados()), False, (255,160,122))#El numero de muertos
    display.blit(textsurface, (250, 30)) # Graficar el texto de muertes
    display.blit(textsurface1, (250, 45))
    display.blit(textsurface2, (250, 65))
    #Graficar el estado del paciente matriz
    for x in range(nb_cols):
        for y in range(nb_rows):
            if states[x][y] == 0:
                color = BLUE # No infectado
```

```
            if states[x][y] == 1:
                color = GREEN # Recupero
            if states[x][y] >= 10:
                color = (states[x][y] * 12, 50, 50) # Injectado - Rojo
            if states[x][y] == -1:
                color = BLACK # Muerto
            pygame.draw.circle(display, color, (100 + x * 10 + 5, 100 + y * 10 + 5), 4)
            pygame.draw.rect(display, WHITE, (100 + x * 12 + 3, 100 + y * 12 + 4, 1, 1))
    #Escuachar los eventos del teclado
    for event in pygame.event.get():
        if event.type == pygame.KEYDOWN and event.key == pygame.K_ESCAPE: #Presiona y Escape
            pygame.quit() #Termino simulacion
            print('Total muertes',contar_muertes())
            print('Recuperados', contar_recuperados())
            print('Suseptible', contar_suceptible()-contar_muertes()-contar_recuperados())
        if event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE: #Presiona y espacio
            #Reiniciamos valores
            states = [[0] * nb_cols for i1 in range(nb_rows)]
            states_temp = states.copy()
            states[5][5] = 10
            it = 0
            total_muerte = 0
            vacunar() #Llamar a la funcion vacunar

    pygame.display.update()# Mandar actualizar la ventana
```

```
Total muertes 186
Recuperados 1779
Suseptible 535
```

```
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
<ipython-input-2-d091fb7e4412> in <module>
    150             vacunar() #Llamar a la funcion vacunar
    151
--> 152     pygame.display.update()# Mandar actualizar la ventana

error: video system not initialized
```





Cual tiene una mejor prediccion

Desde mi punto de vista el que mejor prediccion tiene es el modelo polinomial, ya que como se aprecian en las graficas el q mejor se ajusta es este precisamente ya que el lineal y logistico estan muy aleados de la realidad, mientras que el exponencial y polinomial son los que mas se acercan a la realidad, se debe tener en cuenta tambien que el para el modelo polinomial se debe tener en cuenta el grado.

Ventajas y Desventajas del modelos

Ventajas

- Fácil de entender y explicar, lo que puede ser muy valioso para las decisiones de negocios.
- Es rápido de modelar y es particularmente útil cuando la relación a modelar no es extremadamente compleja y no tiene mucha información.
- Es menos propenso al sobreajuste. ### Desventajas
- No se puede modelar relaciones complejas.
- No se pueden capturar relaciones no lineales sin transformar la entrada, por lo que tienes que trabajar duro para que se ajuste a funciones no lineales.
- Puede sufrir con valores atípicos.