Stage-4

April 26, 2023

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
import scipy.stats as stats
import seaborn as sns
from sklearn.metrics import mean_squared_error
```

[2]:
```
# Reading new cases data of USA.

dayWise = pd.read_csv("../../Member/Pulibandla-Venkatesh/dayWise.csv")
dayWise.drop(['Unnamed: 0'],axis = 1, inplace = True)
dayWise
```

[2]:

	countyFIPS	County Name	State	StateFIPS	Date	\
0	1001	Autauga County	AL	1	2022-05-30	
1	1003	Baldwin County	AL	1	2022-05-30	
2	1005	Barbour County	AL	1	2022-05-30	
3	1007	Bibb County	AL	1	2022-05-30	
4	1009	Blount County	AL	1	2022-05-30	
…	…	…	…	…	…	
681809	56037	Sweetwater County	WY	56	2023-01-01	
681810	56039	Teton County	WY	56	2023-01-01	
681811	56041	Uinta County	WY	56	2023-01-01	
681812	56043	Washakie County	WY	56	2023-01-01	
681813	56045	Weston County	WY	56	2023-01-01	

	Number of new cases	Number of new Deaths
0	9	0
1	55	1
2	1	0
3	9	0
4	6	0
…	…	…
681809	0	0
681810	0	0

```
681811                    0                    0
681812                    0                    0
681813                    0                    0

[681814 rows x 7 columns]
```

[3]:
```
# Choosing Virginia State.

virginiaState = dayWise[dayWise['State']=='VA']
virginiaState
```

[3]:
```
        countyFIPS             County Name State  StateFIPS        Date  \
2820         51001        Accomack County     VA         51  2022-05-30
2821         51003       Albemarle County     VA         51  2022-05-30
2822         51005       Alleghany County     VA         51  2022-05-30
2823         51007          Amelia County     VA         51  2022-05-30
2824         51009         Amherst County     VA         51  2022-05-30
…              …                      …   …        …          …
681620       51800         City of Suffolk     VA         51  2023-01-01
681621       51810  City of Virginia Beach     VA         51  2023-01-01
681622       51820      City of Waynesboro     VA         51  2023-01-01
681623       51830    City of Williamsburg     VA         51  2023-01-01
681624       51840      City of Winchester     VA         51  2023-01-01

        Number of new cases  Number of new Deaths
2820                      0                     0
2821                      0                     0
2822                      0                     0
2823                      0                     0
2824                      0                     0
…                       …                     …
681620                    0                     0
681621                    0                     0
681622                    0                     0
681623                    0                     0
681624                    0                     0

[28861 rows x 7 columns]
```

[4]:
```
# Group by date to count number of cases and deaths of virginia state for each␣
↪day.

virginiaState = virginiaState.groupby(['Date']).sum().reset_index()
```

[5]:
```
# Displaying virginia state.

virginiaState
```

[5]:
```
            Date  countyFIPS  StateFIPS  Number of new cases  \
0     2022-05-30     6818111       6783                    0
1     2022-05-31     6818111       6783                 9943
2     2022-06-01     6818111       6783                 2970
3     2022-06-02     6818111       6783                 2918
4     2022-06-03     6818111       6783                 4056
..           …         …        …                    …
212   2022-12-28     6818111       6783                 2435
213   2022-12-29     6818111       6783                    0
214   2022-12-30     6818111       6783                    0
215   2022-12-31     6818111       6783                    0
216   2023-01-01     6818111       6783                    0

     Number of new Deaths
0                       0
1                      13
2                      11
3                      10
4                      14
..                    …
212                     6
213                     0
214                     0
215                     0
216                     0

[217 rows x 5 columns]
```

[6]:
```
# calculating the length of virginia state dataframe to count number of days in␣
 ↪the last six months of 2022.

len(virginiaState)
```

[6]:
```
217
```

[7]:
```
# calculating the number of days.

days = []
for i in range(217):
    days.append(i)

days
```

[7]:
```
[0,
 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,
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,
100,
101,
102,
103,
104,
105,
106,
107,
108,
109,
110,
111,
112,
113,
114,
115,
116,
117,
118,
119,
120,
121,
122,
123,
124,
125,
126,
127,
128,
129,
130,
131,
132,
133,
134,
135,
136,
137,
138,
139,
140,
141,
142,
143,
144,
```

145,
146,
147,
148,
149,
150,
151,
152,
153,
154,
155,
156,
157,
158,
159,
160,
161,
162,
163,
164,
165,
166,
167,
168,
169,
170,
171,
172,
173,
174,
175,
176,
177,
178,
179,
180,
181,
182,
183,
184,
185,
186,
187,
188,
189,
190,
191,

```
 192,
 193,
 194,
 195,
 196,
 197,
 198,
 199,
 200,
 201,
 202,
 203,
 204,
 205,
 206,
 207,
 208,
 209,
 210,
 211,
 212,
 213,
 214,
 215,
 216]
```

[8]:
```
# converting these days into dataframe and reading new cases and new deaths to␣
 ↪new variables.

xAxis = pd.DataFrame({'days': days})

yCases = virginiaState['Number of new cases']
yDeaths = virginiaState['Number of new Deaths']
```

[9]:
```
# Modeling linear regression for cases.

linearModelCases = LinearRegression()
linearModelCases.fit(xAxis,yCases)
```

[9]:
```
LinearRegression()
```

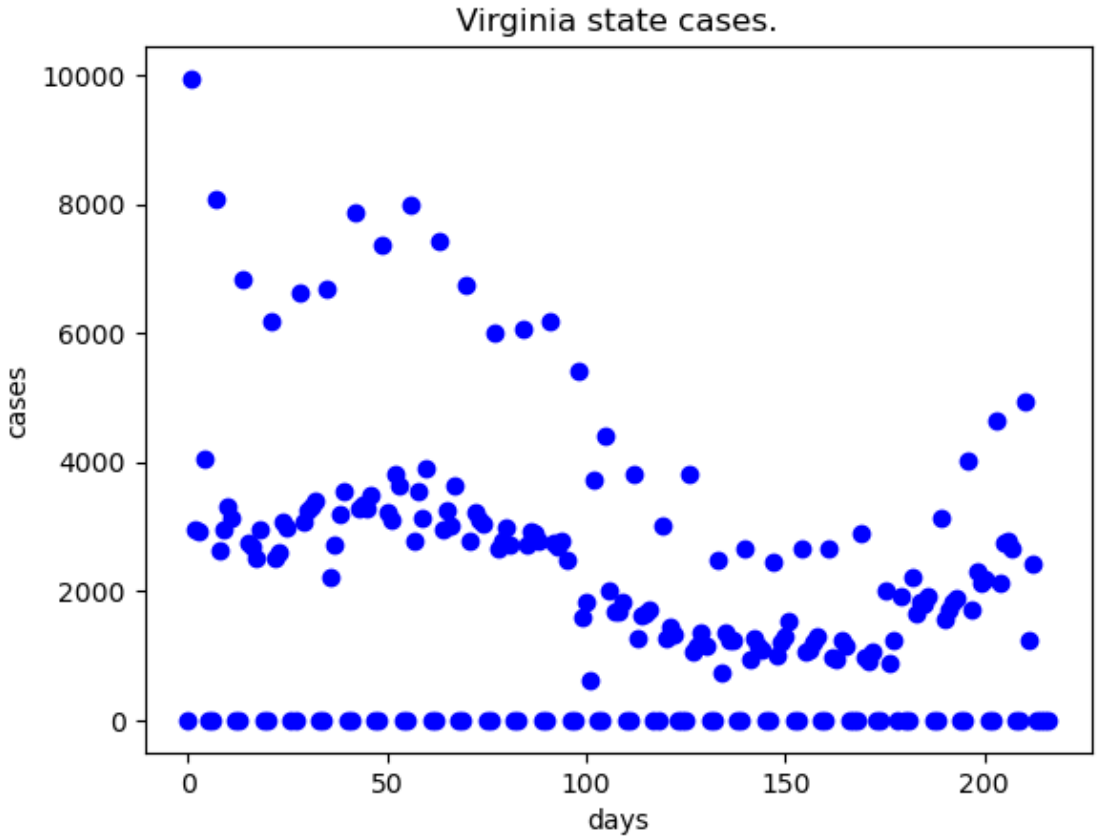
[10]:
```
# Finding intercept and slope for cases model

print(linearModelCases.intercept_)

print(linearModelCases.coef_)
```

```
3088.475499936583
```

```
[-10.61231368]
```

[11]:
```
# plotting cases in virginia sate using scatter plot.

plt.scatter(xAxis,yCases, color='blue')
plt.xlabel('days')
plt.ylabel('cases')
plt.title("Virginia state cases.")
```

[11]:
```
Text(0.5, 1.0, 'Virginia state cases.')
```



[12]:
```
# Generating linear trend data for cases in virginia state.

linearCasesPrediction = linearModelCases.predict(xAxis)
linearCasesPrediction
```

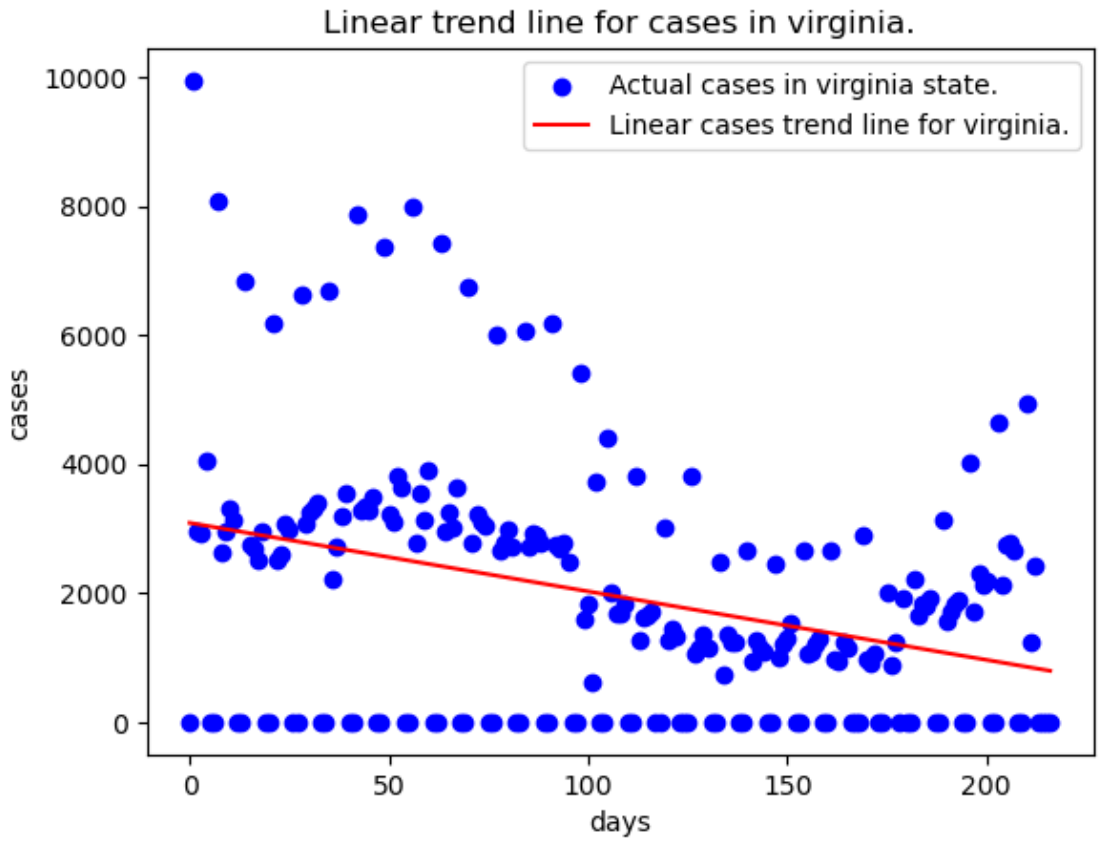
[12]:
```
array([3088.47549994, 3077.86318625, 3067.25087257, 3056.63855889,
       3046.0262452 , 3035.41393152, 3024.80161784, 3014.18930415,
       3003.57699047, 2992.96467679, 2982.3523631 , 2971.74004942,
       2961.12773573, 2950.51542205, 2939.90310837, 2929.29079468,
```

```
2918.678481  , 2908.06616732, 2897.45385363, 2886.84153995,
2876.22922627, 2865.61691258, 2855.0045989 , 2844.39228522,
2833.77997153, 2823.16765785, 2812.55534417, 2801.94303048,
2791.3307168 , 2780.71840312, 2770.10608943, 2759.49377575,
2748.88146206, 2738.26914838, 2727.6568347 , 2717.04452101,
2706.43220733, 2695.81989365, 2685.20757996, 2674.59526628,
2663.9829526 , 2653.37063891, 2642.75832523, 2632.14601155,
2621.53369786, 2610.92138418, 2600.3090705 , 2589.69675681,
2579.08444313, 2568.47212945, 2557.85981576, 2547.24750208,
2536.6351884 , 2526.02287471, 2515.41056103, 2504.79824734,
2494.18593366, 2483.57361998, 2472.96130629, 2462.34899261,
2451.73667893, 2441.12436524, 2430.51205156, 2419.89973788,
2409.28742419, 2398.67511051, 2388.06279683, 2377.45048314,
2366.83816946, 2356.22585578, 2345.61354209, 2335.00122841,
2324.38891473, 2313.77660104, 2303.16428736, 2292.55197367,
2281.93965999, 2271.32734631, 2260.71503262, 2250.10271894,
2239.49040526, 2228.87809157, 2218.26577789, 2207.65346421,
2197.04115052, 2186.42883684, 2175.81652316, 2165.20420947,
2154.59189579, 2143.97958211, 2133.36726842, 2122.75495474,
2112.14264106, 2101.53032737, 2090.91801369, 2080.30570001,
2069.69338632, 2059.08107264, 2048.46875895, 2037.85644527,
2027.24413159, 2016.6318179 , 2006.01950422, 1995.40719054,
1984.79487685, 1974.18256317, 1963.57024949, 1952.9579358 ,
1942.34562212, 1931.73330844, 1921.12099475, 1910.50868107,
1899.89636739, 1889.2840537 , 1878.67174002, 1868.05942634,
1857.44711265, 1846.83479897, 1836.22248528, 1825.6101716 ,
1814.99785792, 1804.38554423, 1793.77323055, 1783.16091687,
1772.54860318, 1761.9362895 , 1751.32397582, 1740.71166213,
1730.09934845, 1719.48703477, 1708.87472108, 1698.2624074 ,
1687.65009372, 1677.03778003, 1666.42546635, 1655.81315267,
1645.20083898, 1634.5885253 , 1623.97621162, 1613.36389793,
1602.75158425, 1592.13927056, 1581.52695688, 1570.9146432 ,
1560.30232951, 1549.69001583, 1539.07770215, 1528.46538846,
1517.85307478, 1507.2407611 , 1496.62844741, 1486.01613373,
1475.40382005, 1464.79150636, 1454.17919268, 1443.566879  ,
1432.95456531, 1422.34225163, 1411.72993795, 1401.11762426,
1390.50531058, 1379.89299689, 1369.28068321, 1358.66836953,
1348.05605584, 1337.44374216, 1326.83142848, 1316.21911479,
1305.60680111, 1294.99448743, 1284.38217374, 1273.76986006,
1263.15754638, 1252.54523269, 1241.93291901, 1231.32060533,
1220.70829164, 1210.09597796, 1199.48366428, 1188.87135059,
1178.25903691, 1167.64672323, 1157.03440954, 1146.42209586,
1135.80978217, 1125.19746849, 1114.58515481, 1103.97284112,
1093.36052744, 1082.74821376, 1072.13590007, 1061.52358639,
1050.91127271, 1040.29895902, 1029.68664534, 1019.07433166,
1008.46201797,  997.84970429,  987.23739061,  976.62507692,
 966.01276324,  955.40044956,  944.78813587,  934.17582219,
```

```
923.5635085 ,  912.95119482,  902.33888114,  891.72656745,
881.11425377,  870.50194009,  859.8896264 ,  849.27731272,
838.66499904,  828.05268535,  817.44037167,  806.82805799,
796.2157443 ])
```

[13]:
```
# plotting linear cases trend line.

plt.scatter(xAxis, yCases, color = 'blue', label="Actual cases in virginia␣
 ↪state.")
plt.plot(xAxis, linearCasesPrediction, color = 'red', label='Linear cases trend␣
 ↪line for virginia.')
plt.xlabel('days')
plt.ylabel('cases')
plt.title('Linear trend line for cases in virginia.')
plt.legend()
```

[13]: <matplotlib.legend.Legend at 0x1e9f1df9880>



[14]:
```
# RMSE error linear model cases and actual cases in virginia.
originalCases = yCases
```

```
predicCases = linearCasesPrediction
rmse = mean_squared_error(originalCases, predicCases, squared=False)
rmse
```

[14]:
```
1809.9253706218853
```

[15]:
```
# Modelling linear regression for deaths in virginia.

linearModelDeaths = LinearRegression()
linearModelDeaths.fit(xAxis,yDeaths)
```

[15]:
```
LinearRegression()
```

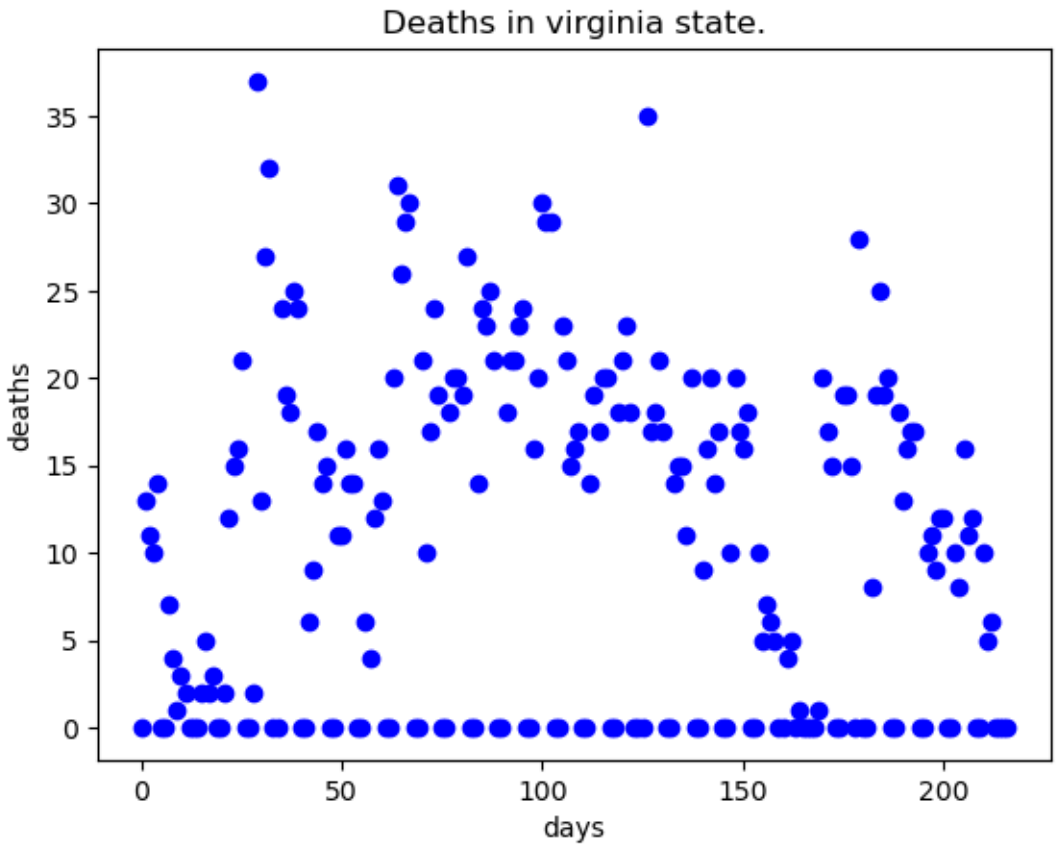
[16]:
```
# generating linear trend data for deaths in virginia.

linearDeathsPrediction = linearModelDeaths.predict(xAxis)
linearDeathsPrediction
```

[16]:
```
array([11.6089291 , 11.59932261, 11.58971613, 11.58010964, 11.57050315,
       11.56089667, 11.55129018, 11.5416837 , 11.53207721, 11.52247072,
       11.51286424, 11.50325775, 11.49365126, 11.48404478, 11.47443829,
       11.4648318 , 11.45522532, 11.44561883, 11.43601235, 11.42640586,
       11.41679937, 11.40719289, 11.3975864 , 11.38797991, 11.37837343,
       11.36876694, 11.35916045, 11.34955397, 11.33994748, 11.330341  ,
       11.32073451, 11.31112802, 11.30152154, 11.29191505, 11.28230856,
       11.27270208, 11.26309559, 11.2534891 , 11.24388262, 11.23427613,
       11.22466964, 11.21506316, 11.20545667, 11.19585019, 11.1862437 ,
       11.17663721, 11.16703073, 11.15742424, 11.14781775, 11.13821127,
       11.12860478, 11.11899829, 11.10939181, 11.09978532, 11.09017884,
       11.08057235, 11.07096586, 11.06135938, 11.05175289, 11.0421464 ,
       11.03253992, 11.02293343, 11.01332694, 11.00372046, 10.99411397,
       10.98450749, 10.974901  , 10.96529451, 10.95568803, 10.94608154,
       10.93647505, 10.92686857, 10.91726208, 10.90765559, 10.89804911,
       10.88844262, 10.87883614, 10.86922965, 10.85962316, 10.85001668,
       10.84041019, 10.8308037 , 10.82119722, 10.81159073, 10.80198424,
       10.79237776, 10.78277127, 10.77316479, 10.7635583 , 10.75395181,
       10.74434533, 10.73473884, 10.72513235, 10.71552587, 10.70591938,
       10.69631289, 10.68670641, 10.67709992, 10.66749344, 10.65788695,
       10.64828046, 10.63867398, 10.62906749, 10.619461  , 10.60985452,
       10.60024803, 10.59064154, 10.58103506, 10.57142857, 10.56182209,
       10.5522156 , 10.54260911, 10.53300263, 10.52339614, 10.51378965,
       10.50418317, 10.49457668, 10.48497019, 10.47536371, 10.46575722,
       10.45615073, 10.44654425, 10.43693776, 10.42733128, 10.41772479,
       10.4081183 , 10.39851182, 10.38890533, 10.37929884, 10.36969236,
       10.36008587, 10.35047938, 10.3408729 , 10.33126641, 10.32165993,
       10.31205344, 10.30244695, 10.29284047, 10.28323398, 10.27362749,
       10.26402101, 10.25441452, 10.24480803, 10.23520155, 10.22559506,
```

```
10.21598858, 10.20638209, 10.1967756 , 10.18716912, 10.17756263,
10.16795614, 10.15834966, 10.14874317, 10.13913668, 10.1295302 ,
10.11992371, 10.11031723, 10.10071074, 10.09110425, 10.08149777,
10.07189128, 10.06228479, 10.05267831, 10.04307182, 10.03346533,
10.02385885, 10.01425236, 10.00464588,  9.99503939,  9.9854329 ,
 9.97582642,  9.96621993,  9.95661344,  9.94700696,  9.93740047,
 9.92779398,  9.9181875 ,  9.90858101,  9.89897453,  9.88936804,
 9.87976155,  9.87015507,  9.86054858,  9.85094209,  9.84133561,
 9.83172912,  9.82212263,  9.81251615,  9.80290966,  9.79330318,
 9.78369669,  9.7740902 ,  9.76448372,  9.75487723,  9.74527074,
 9.73566426,  9.72605777,  9.71645128,  9.7068448 ,  9.69723831,
 9.68763182,  9.67802534,  9.66841885,  9.65881237,  9.64920588,
 9.63959939,  9.62999291,  9.62038642,  9.61077993,  9.60117345,
 9.59156696,  9.58196047,  9.57235399,  9.5627475 ,  9.55314102,
 9.54353453,  9.53392804])
```

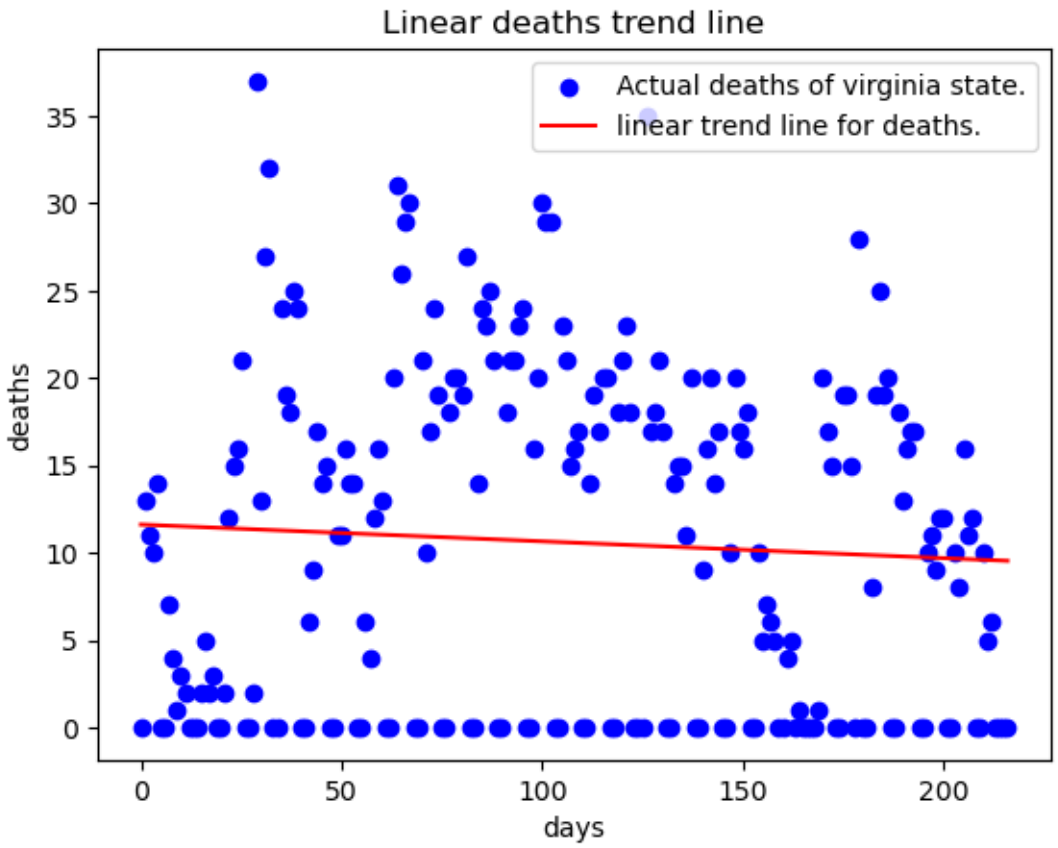
[17]:
```
# Plotting deaths in virginia state.

plt.scatter(xAxis, yDeaths, color='blue')
plt.title("Deaths in virginia state.")
plt.xlabel('days')
plt.ylabel("deaths")
plt.plot()
```

[17]: []



[18]:
```
# plotting the linear deaths trend line for virginia state.

plt.scatter(xAxis, yDeaths, color='blue', label='Actual deaths of virginia␣
 ↪state.')
plt.plot(xAxis,linearDeathsPrediction, color='red', label='linear trend line␣
 ↪for deaths.')
plt.title('Linear deaths trend line')
plt.xlabel('days')
plt.ylabel('deaths')
plt.legend()
```

[18]: <matplotlib.legend.Legend at 0x1e9f1aa0c40>



[19]:
```
#RMSE error for linear model deaths.
originalDeaths = yDeaths
predicDeaths = linearDeathsPrediction
rmse = mean_squared_error(originalDeaths, predicDeaths, squared=False)
rmse
```

[19]: 9.610996437836102

1 Non Linear regression

[20]:
```
# Modeling a non-linear regression of degree 3 for virginia data.
# I have tried using different degrees for optimal fit, degree 3 has the␣
 ↪optimal fit for this data.

nonLinear = PolynomialFeatures(degree=3)
days = np.array(days)
days = days.reshape(-1,1)
xAxisPoly = nonLinear.fit_transform(days)
days
```

```
[20]: array([[  0],
             [  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],
[ 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],
[100],
[101],
[102],
[103],
[104],
[105],
[106],
[107],
[108],
[109],
[110],
[111],
[112],
[113],
[114],
[115],
[116],
[117],
[118],
[119],
[120],
[121],
[122],
[123],
[124],
[125],
[126],
[127],
[128],
[129],
[130],
[131],
[132],
[133],
[134],
[135],
[136],
[137],
[138],
[139],
[140],
```

```
[141],
[142],
[143],
[144],
[145],
[146],
[147],
[148],
[149],
[150],
[151],
[152],
[153],
[154],
[155],
[156],
[157],
[158],
[159],
[160],
[161],
[162],
[163],
[164],
[165],
[166],
[167],
[168],
[169],
[170],
[171],
[172],
[173],
[174],
[175],
[176],
[177],
[178],
[179],
[180],
[181],
[182],
[183],
[184],
[185],
[186],
[187],
```

```
       [188],
       [189],
       [190],
       [191],
       [192],
       [193],
       [194],
       [195],
       [196],
       [197],
       [198],
       [199],
       [200],
       [201],
       [202],
       [203],
       [204],
       [205],
       [206],
       [207],
       [208],
       [209],
       [210],
       [211],
       [212],
       [213],
       [214],
       [215],
       [216]])
```

[21]:
```
# Applying linear model on the non-linear data to genrate non-linear data.

linearModelCases.fit(xAxisPoly, yCases)
```

[21]:
```
LinearRegression()
```

[22]:
```
# Generating non-linear trend data for cases in virginia.

polyCasesPrediction = linearModelCases.predict(xAxisPoly)
polyCasesPrediction
```

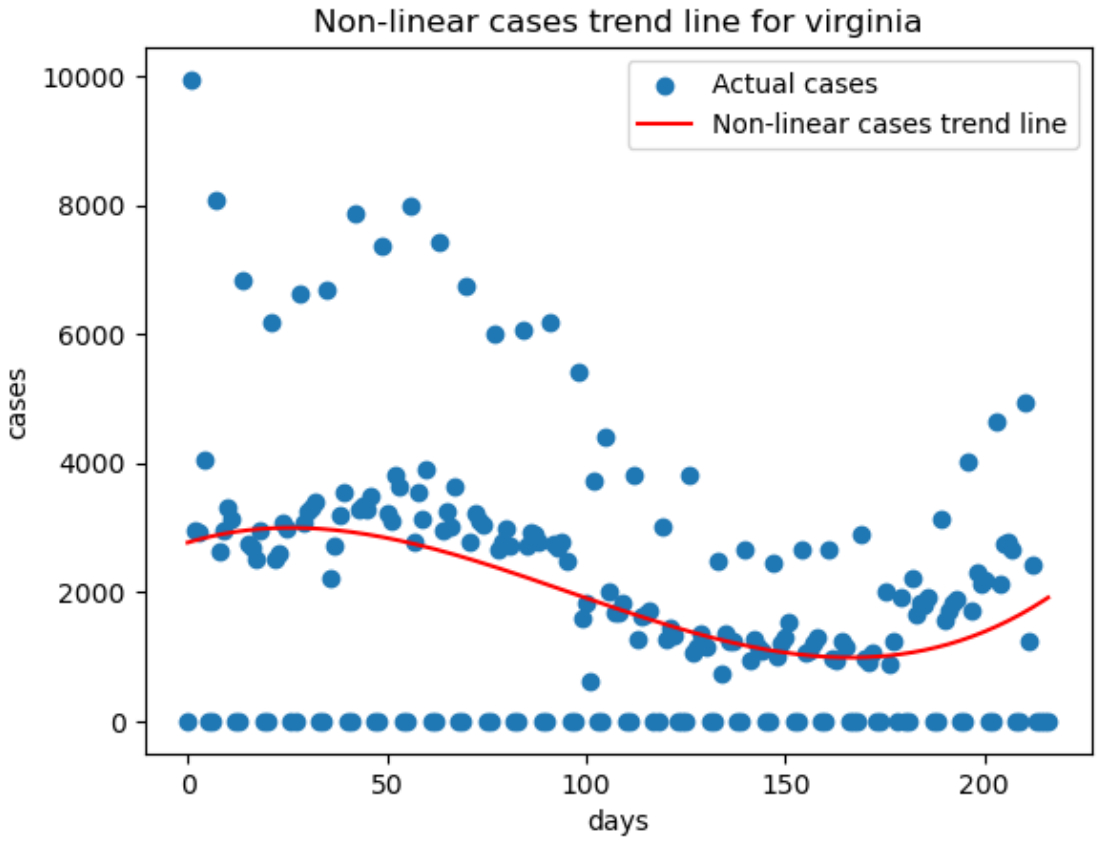
[22]:
```
array([2768.06643997, 2786.35886156, 2803.82413722, 2820.47096269,
       2836.30803366, 2851.34404584, 2865.58769495, 2879.0476767 ,
       2891.73268681, 2903.65142097, 2914.81257491, 2925.22484432,
       2934.89692494, 2943.83751246, 2952.0553026 , 2959.55899106,
       2966.35727357, 2972.45884583, 2977.87240355, 2982.60664244,
       2986.67025821, 2990.07194659, 2992.82040327, 2994.92432396,
```

2996.39240439, 2997.23334026, 2997.45582728, 2997.06856116,
2996.08023761, 2994.49955235, 2992.33520109, 2989.59587954,
2986.2902834 , 2982.42710839, 2978.01505023, 2973.06280461,
2967.57906727, 2961.57253389, 2955.0519002 , 2948.02586191,
2940.50311473, 2932.49235437, 2924.00227653, 2915.04157694,
2905.61895131, 2895.74309533, 2885.42270474, 2874.66647522,
2863.48310251, 2851.88128231, 2839.86971033, 2827.45708227,
2814.65209386, 2801.46344081, 2787.89981882, 2773.96992361,
2759.68245088, 2745.04609635, 2730.06955573, 2714.76152474,
2699.13069907, 2683.18577445, 2666.93544659, 2650.38841119,
2633.55336396, 2616.43900063, 2599.05401689, 2581.40710847,
2563.50697106, 2545.36230039, 2526.98179217, 2508.3741421 ,
2489.54804589, 2470.51219927, 2451.27529793, 2431.84603759,
2412.23311397, 2392.44522277, 2372.4910597 , 2352.37932047,
2332.11870081, 2311.7178964 , 2291.18560298, 2270.53051625,
2249.76133192, 2228.8867457 , 2207.9154533 , 2186.85615044,
2165.71753283, 2144.50829617, 2123.23713618, 2101.91274857,
2080.54382905, 2059.13907333, 2037.70717712, 2016.25683614,
1994.79674609, 1973.33560269, 1951.88210165, 1930.44493868,
1909.03280948, 1887.65440978, 1866.31843528, 1845.03358169,
1823.80854473, 1802.65202011, 1781.57270353, 1760.57929071,
1739.68047736, 1718.88495919, 1698.20143191, 1677.63859124,
1657.20513288, 1636.90975254, 1616.76114595, 1596.7680088 ,
1576.93903681, 1557.28292569, 1537.80837115, 1518.5240689 ,
1499.43871466, 1480.56100414, 1461.89963304, 1443.46329707,
1425.26069196, 1407.30051341, 1389.59145712, 1372.14221882,
1354.96149422, 1338.05797901, 1321.44036893, 1305.11735967,
1289.09764695, 1273.38992648, 1258.00289397, 1242.94524513,
1228.22567567, 1213.85288131, 1199.83555775, 1186.18240071,
1172.9021059 , 1160.00336903, 1147.49488581, 1135.38535195,
1123.68346316, 1112.39791516, 1101.53740365, 1091.11062434,
1081.12627296, 1071.5930452 , 1062.51963678, 1053.91474342,
1045.78706081, 1038.14528468, 1030.99811074, 1024.35423469,
1018.22235224, 1012.61115912, 1007.52935102, 1002.98562367,
 998.98867276, 995.54719402, 992.66988316, 990.36543588,
 988.64254789, 987.50991492, 986.97623266, 987.05019683,
 987.74050314, 989.05584731, 991.00492504, 993.59643204,
 996.83906403, 1000.74151672, 1005.31248582, 1010.56066703,
1016.49475608, 1023.12344867, 1030.45544051, 1038.49942732,
1047.2641048 , 1056.75816867, 1066.99031464, 1077.96923842,
1089.70363571, 1102.20220224, 1115.47363371, 1129.52662584,
1144.36987433, 1160.0120749 , 1176.46192325, 1193.7281151 ,
1211.81934617, 1230.74431215, 1250.51170877, 1271.13023173,
1292.60857674, 1314.95543952, 1338.17951578, 1362.28950123,
1387.29409158, 1413.20198253, 1440.02186981, 1467.76244912,
1496.43241618, 1526.04046669, 1556.59529637, 1588.10560093,
1620.58007607, 1654.02741752, 1688.45632098, 1723.87548216,

```
       1760.29359677, 1797.71936053, 1836.16146914, 1875.62861833,
       1916.12950379])
```

[23]:
```
# plotting non-linear cases trend line for virginia.

plt.title("Non-linear cases trend line for virginia")
plt.xlabel('days')
plt.ylabel('cases')
plt.scatter(xAxis, yCases, label='Actual cases')
plt.plot(xAxis, polyCasesPrediction,  color='red', label='Non-linear cases␣
 ↪trend line')
plt.legend()
```

[23]:
```
<matplotlib.legend.Legend at 0x1e9f1f702e0>
```



[24]:
```
#RMSE error for non-linear cases.
originalPolyCases = yCases
predicPolyCases = polyCasesPrediction
mse = mean_squared_error(originalPolyCases, predicPolyCases, squared=False)
mse
```

[24]: 1778.9520263005948

[25]:
```
# Applying linear model on non-linear deaths data.

linearModelDeaths.fit(xAxisPoly, yDeaths)
```

[25]: LinearRegression()

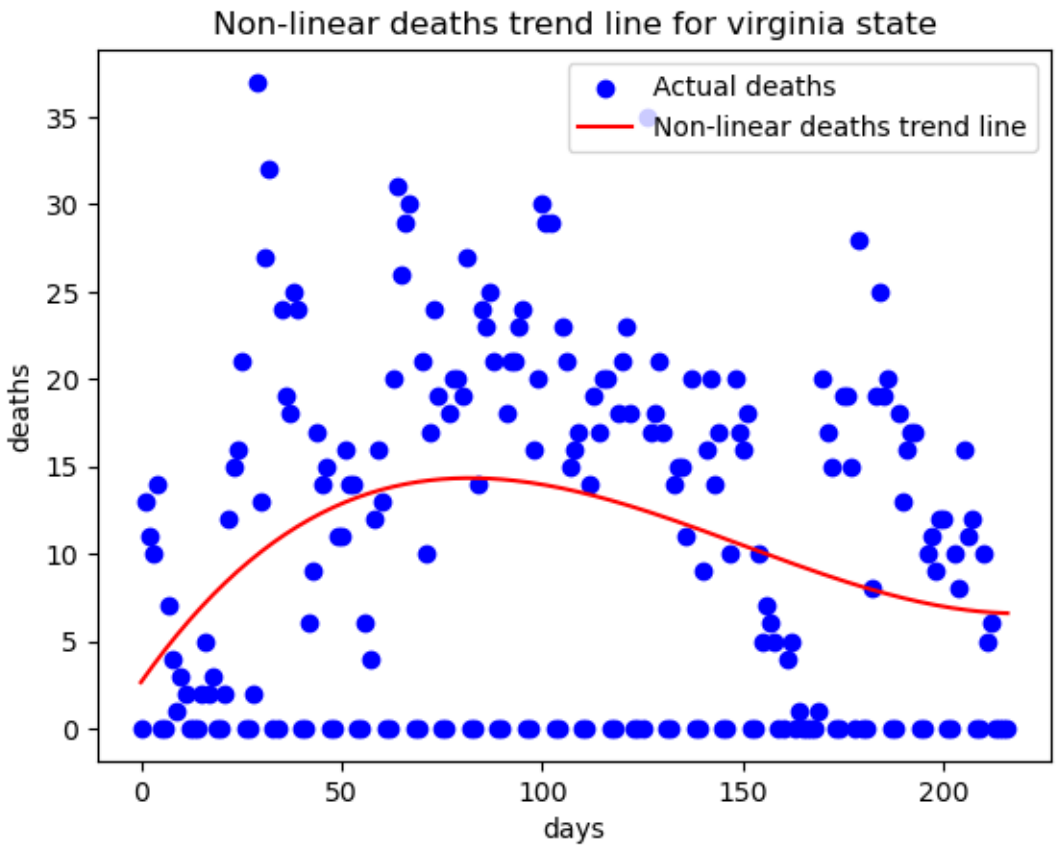
[26]:
```
# Generating non-linear trend data for deaths.

polyDeathsPrediction = linearModelDeaths.predict(xAxisPoly)
polyDeathsPrediction
```

[26]:
```
array([ 2.63298561,  2.95646437,  3.27450735,  3.58715097,  3.89443163,
        4.19638573,  4.49304968,  4.7844599 ,  5.07065278,  5.35166473,
        5.62753216,  5.89829148,  6.16397909,  6.4246314 ,  6.68028482,
        6.93097575,  7.17674059,  7.41761577,  7.65363767,  7.88484272,
        8.11126731,  8.33294785,  8.54992075,  8.76222242,  8.96988926,
        9.17295768,  9.37146409,  9.56544488,  9.75493648,  9.93997528,
       10.12059769, 10.29684013, 10.46873898, 10.63633067, 10.7996516 ,
       10.95873818, 11.1136268 , 11.26435389, 11.41095584, 11.55346906,
       11.69192996, 11.82637495, 11.95684042, 12.0833628 , 12.20597848,
       12.32472388, 12.43963539, 12.55074942, 12.65810239, 12.7617307 ,
       12.86167075, 12.95795896, 13.05063172, 13.13972544, 13.22527654,
       13.30732142, 13.38589648, 13.46103813, 13.53278278, 13.60116684,
       13.66622671, 13.72799879, 13.7865195 , 13.84182524, 13.89395242,
       13.94293745, 13.98881672, 14.03162666, 14.07140366, 14.10818412,
       14.14200447, 14.1729011 , 14.20091043, 14.22606885, 14.24841277,
       14.26797861, 14.28480276, 14.29892164, 14.31037165, 14.31918919,
       14.32541069, 14.32907253, 14.33021113, 14.32886289, 14.32506422,
       14.31885154, 14.31026123, 14.29932972, 14.2860934 , 14.27058869,
       14.25285199, 14.23291971, 14.21082825, 14.18661402, 14.16031343,
       14.13196288, 14.10159878, 14.06925754, 14.03497557, 13.99878926,
       13.96073503, 13.92084929, 13.87916843, 13.83572887, 13.79056702,
       13.74371928, 13.69522205, 13.64511175, 13.59342478, 13.54019754,
       13.48546645, 13.42926791, 13.37163832, 13.3126141 , 13.25223165,
       13.19052737, 13.12753768, 13.06329898, 12.99784767, 12.93122017,
       12.86345288, 12.79458221, 12.72464456, 12.65367633, 12.58171395,
       12.50879381, 12.43495231, 12.36022588, 12.28465091, 12.2082638 ,
       12.13110098, 12.05319883, 11.97459378, 11.89532222, 11.81542057,
       11.73492522, 11.65387259, 11.57229909, 11.49024111, 11.40773507,
       11.32481737, 11.24152443, 11.15789263, 11.07395841, 10.98975815,
       10.90532826, 10.82070516, 10.73592525, 10.65102493, 10.56604062,
       10.48100872, 10.39596563, 10.31094776, 10.22599153, 10.14113333,
       10.05640957,  9.97185666,  9.887511  ,  9.80340901,  9.71958709,
        9.63608165,  9.55292908,  9.47016581,  9.38782823,  9.30595275,
        9.22457578,  9.14373373,  9.063463  ,  8.98379999,  8.90478113,
```

```
8.8264428 ,  8.74882143,  8.67195341,  8.59587515,  8.52062306,
8.44623354,  8.37274301,  8.30018787,  8.22860452,  8.15802937,
8.08849883,  8.02004931,  7.95271721,  7.88653893,  7.8215509 ,
7.7577895 ,  7.69529115,  7.63409226,  7.57422923,  7.51573846,
7.45865637,  7.40301937,  7.34886385,  7.29622623,  7.2451429 ,
7.19565029,  7.14778479,  7.10158281,  7.05708076,  7.01431505,
6.97332207,  6.93413825,  6.89679997,  6.86134366,  6.82780572,
6.79622255,  6.76663057,  6.73906617,  6.71356577,  6.69016576,
6.66890257,  6.64981259,  6.63293223,  6.61829789,  6.605946  ,
6.59591294,  6.58823513])
```

[27]:
```
# plotting non-linear deaths trend line.

plt.title('Non-linear deaths trend line for virginia state')
plt.xlabel('days')
plt.ylabel('deaths')
plt.scatter(xAxis, yDeaths, color='blue', label='Actual deaths')
plt.plot(xAxis, polyDeathsPrediction, color='red', label='Non-linear deaths␣
 ↪trend line')
plt.legend()
```

[27]:
```
<matplotlib.legend.Legend at 0x1e9f1e7adf0>
```



[28]:
```
# RMSE error for non-linear deaths data.
originalPolyDeaths = yDeaths
predicPolyDeaths = polyDeathsPrediction
mse = mean_squared_error(originalPolyDeaths, predicPolyDeaths, squared=False)
mse
```

[28]:
```
9.148398406836286
```

2 Top counties

In this step, I am going to select top 5 counties for highest number of cases in virginia state.

[29]:
```
counties = dayWise[dayWise['State']=='VA']
counties.loc[counties['Number of new cases']<0, 'Number of new cases']=0
counties.loc[counties['Number of new Deaths']<0, 'Number of new Deaths']=0
```

[30]:
```
# Grouping data on county name to find total number of cases in each state.

countiesGroup = counties.groupby(['County Name'])['Number of new cases',␣
 ↪'Number of new Deaths'].sum().reset_index()
```

```
C:\Users\venka\AppData\Local\Temp\ipykernel_17608\2244776869.py:3:
FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of
keys) will be deprecated, use a list instead.
  countiesGroup = counties.groupby(['County Name'])['Number of new cases',
'Number of new Deaths'].sum().reset_index()
```

[31]:
```
countiesGroup
```

[31]:

	County Name	Number of new cases	Number of new Deaths
0	Accomack County	1563	8
1	Albemarle County	4813	39
2	Alleghany County	926	13
3	Amelia County	688	6
4	Amherst County	1633	18
..	…	…	…
128	Washington County	3695	33
129	Westmoreland County	810	6
130	Wise County	3360	32
131	Wythe County	1844	19
132	York County	2676	18

[133 rows x 3 columns]

[32]:
```
# choosing top 5 counties with highest number of cases.

counties5 = countiesGroup.sort_values(by = 'Number of new cases', ascending =␣
 ↪False)
counties5.head(5)
```

[32]:

	County Name	Number of new cases	Number of new Deaths
66	Fairfax County	55369	200
110	Prince William County	22203	63
90	Loudoun County	19152	45
20	Chesterfield County	18244	103
55	City of Virginia Beach	18078	100

[33]:
```
# Printing all the county names in virginia.

counties['County Name'].unique()
```

[33]:
```
array(['Accomack County ', 'Albemarle County ', 'Alleghany County ',
       'Amelia County ', 'Amherst County ', 'Appomattox County ',
       'Arlington County ', 'Augusta County ', 'Bath County ',
       'Bedford County ', 'Bland County ', 'Botetourt County ',
       'Brunswick County ', 'Buchanan County ', 'Buckingham County ',
       'Campbell County ', 'Caroline County ', 'Carroll County ',
       'Charles City County ', 'Charlotte County ',
       'Chesterfield County ', 'Clarke County ', 'Craig County ',
       'Culpeper County ', 'Cumberland County ', 'Dickenson County ',
       'Dinwiddie County ', 'Essex County ', 'Fairfax County ',
       'Fauquier County ', 'Floyd County ', 'Fluvanna County ',
       'Franklin County ', 'Frederick County ', 'Giles County ',
       'Gloucester County ', 'Goochland County ', 'Grayson County ',
       'Greene County ', 'Greensville County ', 'Halifax County ',
       'Hanover County ', 'Henrico County ', 'Henry County ',
       'Highland County ', 'Isle of Wight County ', 'James City County ',
       'King and Queen County ', 'King George County ',
       'King William County ', 'Lancaster County ', 'Lee County ',
       'Loudoun County ', 'Louisa County ', 'Lunenburg County ',
       'Madison County ', 'Mathews County ', 'Mecklenburg County ',
       'Middlesex County ', 'Montgomery County ', 'Nelson County ',
       'New Kent County ', 'Northampton County ',
       'Northumberland County ', 'Nottoway County ', 'Orange County ',
       'Page County ', 'Patrick County ', 'Pittsylvania County ',
       'Powhatan County ', 'Prince Edward County ',
       'Prince George County ', 'Prince William County ',
       'Pulaski County ', 'Rappahannock County ', 'Richmond County ',
       'Roanoke County ', 'Rockbridge County ', 'Rockingham County ',
       'Russell County ', 'Scott County ', 'Shenandoah County ',
       'Smyth County ', 'Southampton County ', 'Spotsylvania County ',
```

```
       'Stafford County ', 'Surry County ', 'Sussex County ',
       'Tazewell County ', 'Warren County ', 'Washington County ',
       'Westmoreland County ', 'Wise County ', 'Wythe County ',
       'York County ', 'City of Alexandria', 'City of Bristol',
       'City of Buena Vista', 'City of Charlottesville',
       'City of Chesapeake', 'City of Colonial Heights',
       'City of Covington', 'City of Danville', 'City of Emporia',
       'City of Fairfax', 'City of Falls Church', 'City of Franklin',
       'City of Fredericksburg', 'City of Galax', 'City of Hampton',
       'City of Harrisonburg', 'City of Hopewell', 'City of Lexington',
       'City of Lynchburg', 'City of Manassas', 'City of Manassas Park',
       'City of Martinsville', 'City of Newport News', 'City of Norfolk',
       'City of Norton', 'City of Petersburg', 'City of Poquoson',
       'City of Portsmouth', 'City of Radford', 'City of Richmond',
       'City of Roanoke', 'City of Salem', 'City of Staunton',
       'City of Suffolk', 'City of Virginia Beach', 'City of Waynesboro',
       'City of Williamsburg', 'City of Winchester'], dtype=object)
```

[34]:
```
# Choosing top 5 counties into separate dataframes.

county1 = counties[counties['County Name'].isin(['Fairfax County '])].
 ↪reset_index()
county2 = counties[counties['County Name'].isin(['Prince William County '])].
 ↪reset_index()
county3 = counties[counties['County Name'].isin(['Loudoun County '])].
 ↪reset_index()
county4 = counties[counties['County Name'].isin(['Chesterfield County '])].
 ↪reset_index()
county5 = counties[counties['County Name'].isin(['City of Virginia Beach'])].
 ↪reset_index()
```

[35]:
```
# Fairfax county data.
county1
```

[35]:

	index	countyFIPS	County Name	State	StateFIPS	Date
0	2848	51059	Fairfax County	VA	51	2022-05-30
1	5990	51059	Fairfax County	VA	51	2022-05-31
2	9132	51059	Fairfax County	VA	51	2022-06-01
3	12274	51059	Fairfax County	VA	51	2022-06-02
4	15416	51059	Fairfax County	VA	51	2022-06-03
..	…	…	… …	…	…	
212	668952	51059	Fairfax County	VA	51	2022-12-28
213	672094	51059	Fairfax County	VA	51	2022-12-29
214	675236	51059	Fairfax County	VA	51	2022-12-30
215	678378	51059	Fairfax County	VA	51	2022-12-31
216	681520	51059	Fairfax County	VA	51	2023-01-01

```
     Number of new cases  Number of new Deaths
0                      0                     0
1                   1685                     1
2                    516                     1
3                    360                     1
4                    704                     1
..                   …                     …
212                  245                     1
213                    0                     0
214                    0                     0
215                    0                     0
216                    0                     0

[217 rows x 8 columns]
```

[36]:
```
# Reading county1 cases and deaths.

yCasesCounty1 = county1['Number of new cases']
yDeathsCounty1 = county1['Number of new Deaths']
```

[37]:
```
yCasesCounty1
```

[37]:
```
0         0
1      1685
2       516
3       360
4       704
       …
212     245
213       0
214       0
215       0
216       0
Name: Number of new cases, Length: 217, dtype: int64
```

[38]:
```
linearModelCases.fit(xAxisPoly, yCasesCounty1)
```

[38]:
```
LinearRegression()
```

[39]:
```
# Generating non-linear cases trend data for county1.

casesPredictionCounty1 = linearModelCases.predict(xAxisPoly)
casesPredictionCounty1
```

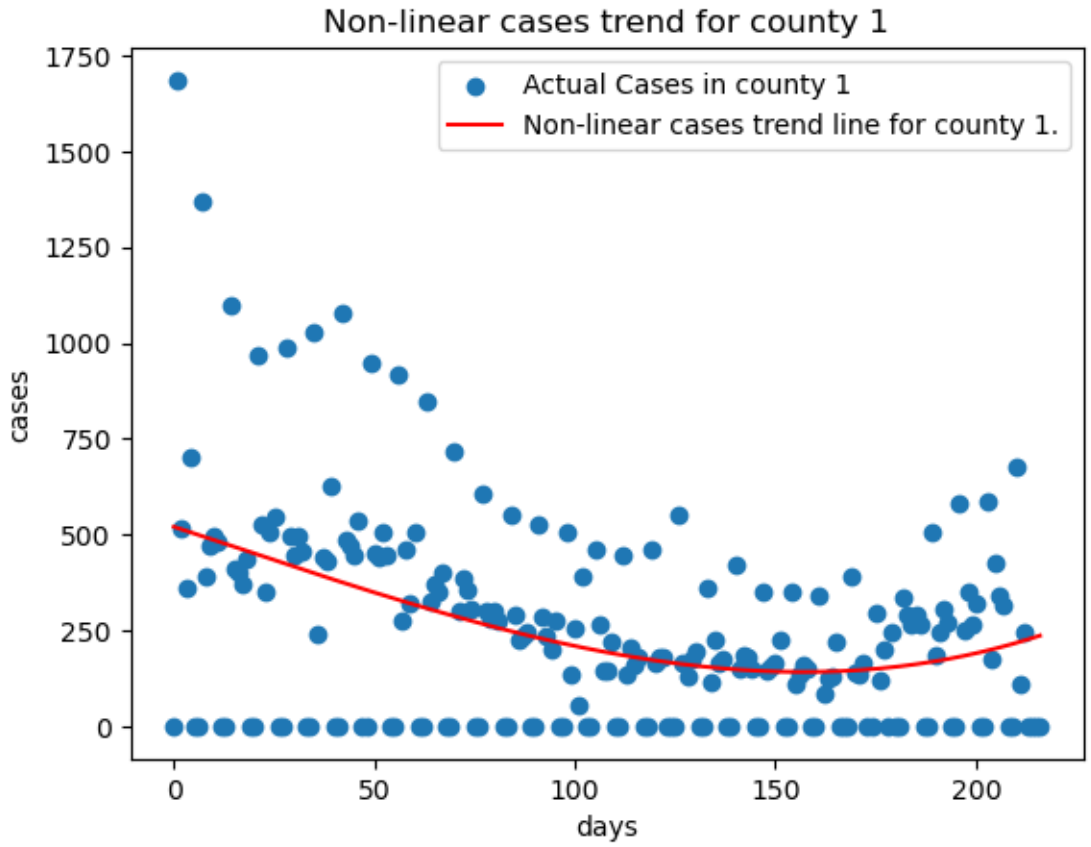
[39]:
```
array([520.20618939, 516.73478199, 513.25993807, 509.78198719,
       506.30125889, 502.81808272, 499.3327882 , 495.8457049 ,
       492.35716236, 488.86749012, 485.37701772, 481.88607471,
```

```
478.39499063, 474.90409503, 471.41371746, 467.92418745,
464.43583456, 460.94898832, 457.46397829, 453.981134  ,
450.50078499, 447.02326083, 443.54889104, 440.07800518,
436.61093278, 433.1480034 , 429.68954657, 426.23589185,
422.78736877, 419.34430689, 415.90703574, 412.47588486,
409.05118382, 405.63326214, 402.22244937, 398.81907507,
395.42346876, 392.03596   , 388.65687834, 385.2865533 ,
381.92531445, 378.57349133, 375.23141347, 371.89941043,
368.57781175, 365.26694697, 361.96714563, 358.67873729,
355.40205149, 352.13741777, 348.88516567, 345.64562474,
342.41912453, 339.20599457, 336.00656442, 332.82116362,
329.65012171, 326.49376824, 323.35243275, 320.22644478,
317.11613389, 314.02182961, 310.94386149, 307.88255908,
304.83825191, 301.81126954, 298.8019415 , 295.81059735,
292.83756663, 289.88317887, 286.94776363, 284.03165046,
281.13516888, 278.25864846, 275.40241873, 272.56680924,
269.75214953, 266.95876915, 264.18699764, 261.43716455,
258.70959942, 256.00463179, 253.32259122, 250.66380724,
248.0286094 , 245.41732724, 242.83029031, 240.26782815,
237.73027031, 235.21794633, 232.73118576, 230.27031814,
227.83567301, 225.42757992, 223.04636842, 220.69236805,
218.36590835, 216.06731886, 213.79692914, 211.55506873,
209.34206717, 207.158254  , 205.00395878, 202.87951103,
200.78524032, 198.72147618, 196.68854817, 194.68678581,
192.71651866, 190.77807626, 188.87178816, 186.9979839 ,
185.15699303, 183.34914508, 181.57476961, 179.83419616,
178.12775428, 176.4557735 , 174.81858337, 173.21651344,
171.64989326, 170.11905236, 168.62432028, 167.16602659,
165.74450081, 164.3600725 , 163.0130712 , 161.70382645,
160.4326678 , 159.19992479, 158.00592697, 156.85100387,
155.73548506, 154.65970006, 153.62397843, 152.62864971,
151.67404345, 150.76048918, 149.88831645, 149.05785481,
148.26943381, 147.52338298, 146.82003187, 146.15971002,
145.54274699, 144.96947231, 144.44021553, 143.9553062 ,
143.51507385, 143.11984803, 142.76995829, 142.46573418,
142.20750522, 141.99560098, 141.830351  , 141.71208481,
141.64113197, 141.61782202, 141.6424845 , 141.71544896,
141.83704494, 142.00760199, 142.22744965, 142.49691746,
142.81633498, 143.18603174, 143.60633728, 144.07758117,
144.60009293, 145.17420211, 145.80023826, 146.47853093,
147.20940965, 147.99320397, 148.83024343, 149.72085759,
150.66537598, 151.66412815, 152.71744364, 153.82565201,
154.98908278, 156.20806551, 157.48292974, 158.81400502,
160.20162089, 161.64610689, 163.14779258, 164.70700749,
166.32408116, 167.99934315, 169.733123  , 171.52575025,
173.37755445, 175.28886514, 177.26001186, 179.29132416,
181.38313159, 183.53576369, 185.74955   , 188.02482007,
```

```
       190.36190344, 192.76112967, 195.22282828, 197.74732883,
       200.33496086, 202.98605391, 205.70093754, 208.47994128,
       211.32339467, 214.23162728, 217.20496863, 220.24374827,
       223.34829575, 226.51894061, 229.7560124 , 233.05984065,
       236.43075493])
```

[40]:
```
# plotting non-linear cases trend line for county1.

plt.scatter(xAxis, yCasesCounty1, label='Actual Cases in county 1')
plt.plot(xAxis, casesPredictionCounty1,  color='red', label='Non-linear cases␣
 ↪trend line for county 1.')
plt.title('Non-linear cases trend for county 1')
plt.xlabel('days')
plt.ylabel('cases')
plt.legend()
```

[40]: <matplotlib.legend.Legend at 0x1e9f1ef0c70>



[41]:
```
# RMSE error for non-linear cases trend data.
originalPolyCasesC1 = yCasesCounty1
predicPolyCasesC1 = casesPredictionCounty1
rmse = mean_squared_error(originalPolyCasesC1, predicPolyCasesC1, squared=False)
rmse
```

[41]: 243.38815651122138

[42]:
```
linearModelDeaths.fit(xAxisPoly, yDeathsCounty1)
```

[42]: LinearRegression()

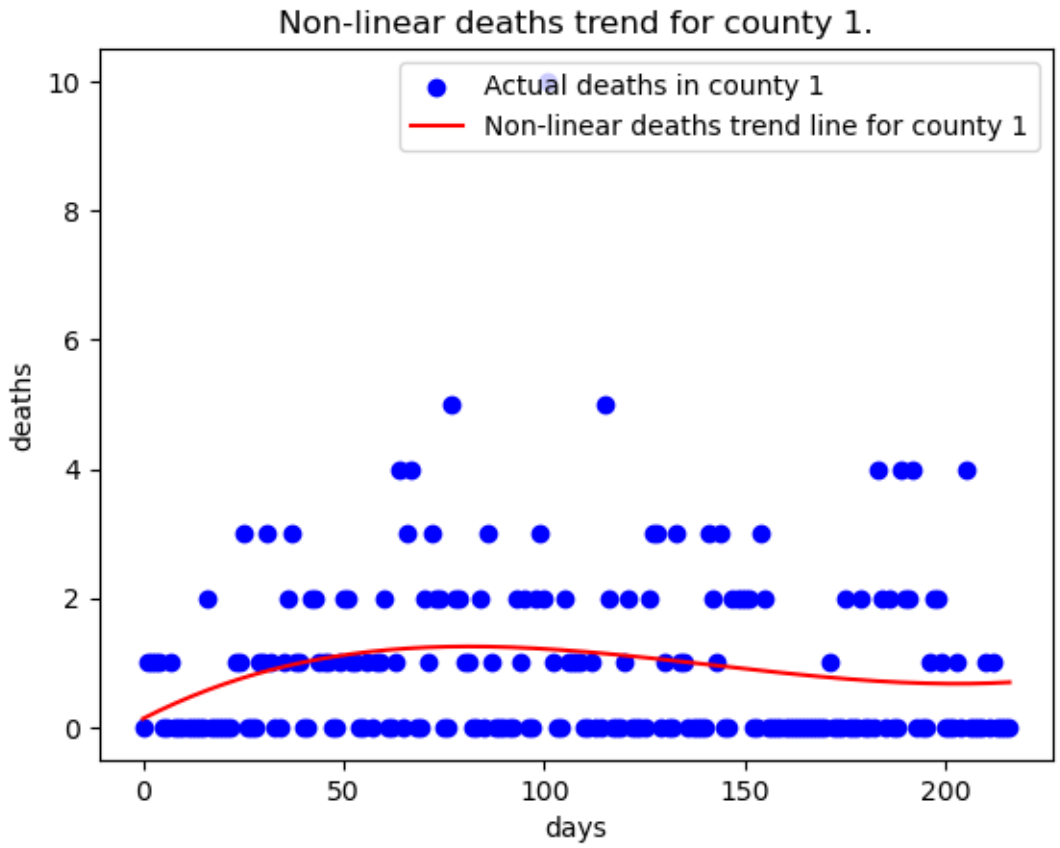
[43]:
```
# Generating non-linear deaths trend data for county 1.

deathsPredictionCounty1 = linearModelDeaths.predict(xAxisPoly)
deathsPredictionCounty1
```

[43]:
```
array([0.13244055, 0.16391844, 0.19485297, 0.22524799, 0.25510735,
       0.28443491, 0.31323452, 0.34151004, 0.36926531, 0.39650419,
       0.42323052, 0.44944817, 0.47516098, 0.50037281, 0.5250875 ,
       0.54930892, 0.57304091, 0.59628733, 0.61905202, 0.64133885,
       0.66315165, 0.68449429, 0.70537062, 0.72578449, 0.74573975,
       0.76524025, 0.78428986, 0.8028924 , 0.82105176, 0.83877176,
       0.85605627, 0.87290914, 0.88933421, 0.90533535, 0.92091641,
       0.93608123, 0.95083368, 0.96517759, 0.97911683, 0.99265525,
       1.00579669, 1.01854502, 1.03090407, 1.04287772, 1.0544698 ,
       1.06568418, 1.07652469, 1.08699521, 1.09709957, 1.10684162,
       1.11622524, 1.12525425, 1.13393252, 1.14226391, 1.15025225,
       1.15790141, 1.16521523, 1.17219757, 1.17885228, 1.18518322,
       1.19119423, 1.19688917, 1.20227188, 1.20734623, 1.21211607,
       1.21658524, 1.2207576 , 1.22463701, 1.2282273 , 1.23153235,
       1.23455599, 1.23730208, 1.23977447, 1.24197703, 1.24391358,
       1.245588  , 1.24700413, 1.24816583, 1.24907694, 1.24974133,
       1.25016283, 1.25034531, 1.25029261, 1.2500086 , 1.24949711,
       1.24876201, 1.24780714, 1.24663637, 1.24525353, 1.24366248,
       1.24186708, 1.23987118, 1.23767862, 1.23529327, 1.23271898,
       1.22995958, 1.22701895, 1.22390093, 1.22060937, 1.21714813,
       1.21352106, 1.20973201, 1.20578483, 1.20168337, 1.19743149,
       1.19303305, 1.18849188, 1.18381185, 1.1789968 , 1.1740506 ,
       1.16897708, 1.16378011, 1.15846354, 1.15303121, 1.14748699,
       1.14183471, 1.13607825, 1.13022144, 1.12426814, 1.1182222 ,
       1.11208748, 1.10586782, 1.09956708, 1.09318912, 1.08673778,
       1.08021691, 1.07363037, 1.06698202, 1.06027569, 1.05351526,
       1.04670456, 1.03984745, 1.03294778, 1.02600941, 1.01903618,
       1.01203195, 1.00500058, 0.99794591, 0.99087179, 0.98378209,
       0.97668064, 0.96957131, 0.96245795, 0.9553444 , 0.94823452,
       0.94113216, 0.93404118, 0.92696543, 0.91990876, 0.91287502,
```

```
0.90586806, 0.89889174, 0.89194991, 0.88504641, 0.87818512,
0.87136986, 0.86460451, 0.85789291, 0.85123891, 0.84464636,
0.83811912, 0.83166104, 0.82527598, 0.81896778, 0.81274029,
0.80659737, 0.80054288, 0.79458066, 0.78871456, 0.78294844,
0.77728616, 0.77173155, 0.76628848, 0.7609608 , 0.75575236,
0.75066701, 0.74570861, 0.740881  , 0.73618804, 0.73163358,
0.72722148, 0.72295558, 0.71883974, 0.71487781, 0.71107365,
0.7074311 , 0.70395401, 0.70064625, 0.69751166, 0.6945541 ,
0.69177741, 0.68918545, 0.68678207, 0.68457113, 0.68255647,
0.68074195, 0.67913142, 0.67772874, 0.67653775, 0.67556231,
0.67480627, 0.67427348, 0.6739678 , 0.67389307, 0.67405315,
0.6744519 , 0.67509316, 0.67598078, 0.67711863, 0.67851054,
0.68016038, 0.68207199, 0.68424924, 0.68669596, 0.68941601,
0.69241325, 0.69569153])
```

[44]:
```
# plotting non-linear deaths trend for county 1.

plt.scatter(xAxis, yDeathsCounty1, color='blue', label='Actual deaths in county␣
 ↪1')
plt.plot(xAxis, deathsPredictionCounty1, color='red', label='Non-linear deaths␣
 ↪trend line for county 1')
plt.title('Non-linear deaths trend for county 1.')
plt.xlabel('days')
plt.ylabel('deaths')
plt.legend()
```

[44]:
```
<matplotlib.legend.Legend at 0x1e9f221f0a0>
```



[45]:
```
# RMSE error fpr non-linear deaths data for county 1.
originalPolyDeathsC1 = yDeathsCounty1
predicPolyDeathsC1 = deathsPredictionCounty1
mse = mean_squared_error(originalPolyDeathsC1, predicPolyDeathsC1,␣
 ↪squared=False)
mse
```

[45]:
```
1.274954842958111
```

[46]:
```
# Reading cases and deaths data for county 2.

yCasesCounty2 = county2['Number of new cases']
yDeathsCounty2 = county2['Number of new Deaths']
```

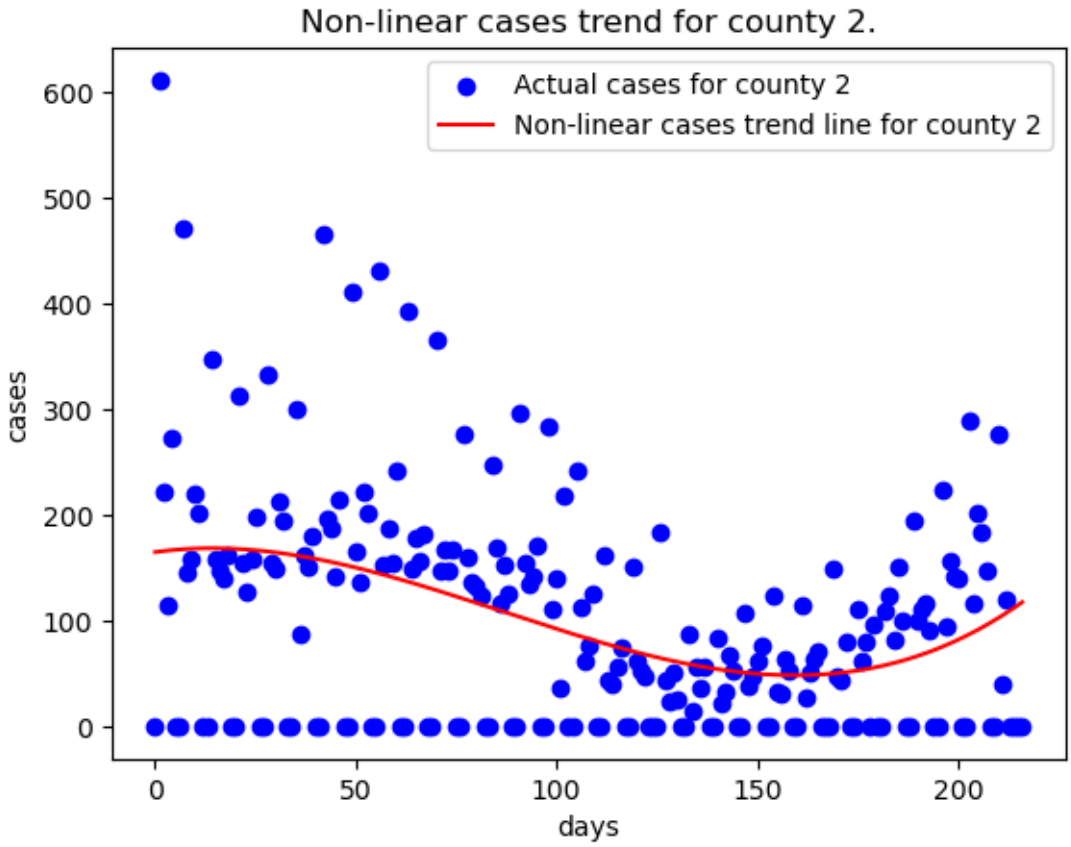
[47]:
```
linearModelCases.fit(xAxisPoly, yCasesCounty2)
linearModelDeaths.fit(xAxisPoly, yDeathsCounty2)
```

[47]:
```
LinearRegression()
```

[48]:
```
# Generating non-linear cases and deaths trend data for county 2.

casesPredictionCounty2 = linearModelCases.predict(xAxisPoly)
deathsPredictionCounty2 = linearModelDeaths.predict(xAxisPoly)
```

[49]:
```
# plotting non-linear cases trend line for county 2.

plt.scatter(xAxis, yCasesCounty2, color='blue', label='Actual cases for county␣
 ↪2')
plt.plot(xAxis, casesPredictionCounty2, color='red', label='Non-linear cases␣
 ↪trend line for county 2')
plt.title('Non-linear cases trend for county 2.')
plt.xlabel('days')
plt.ylabel('cases')
plt.legend()
```

[49]:
```
<matplotlib.legend.Legend at 0x1e9f1ff8460>
```

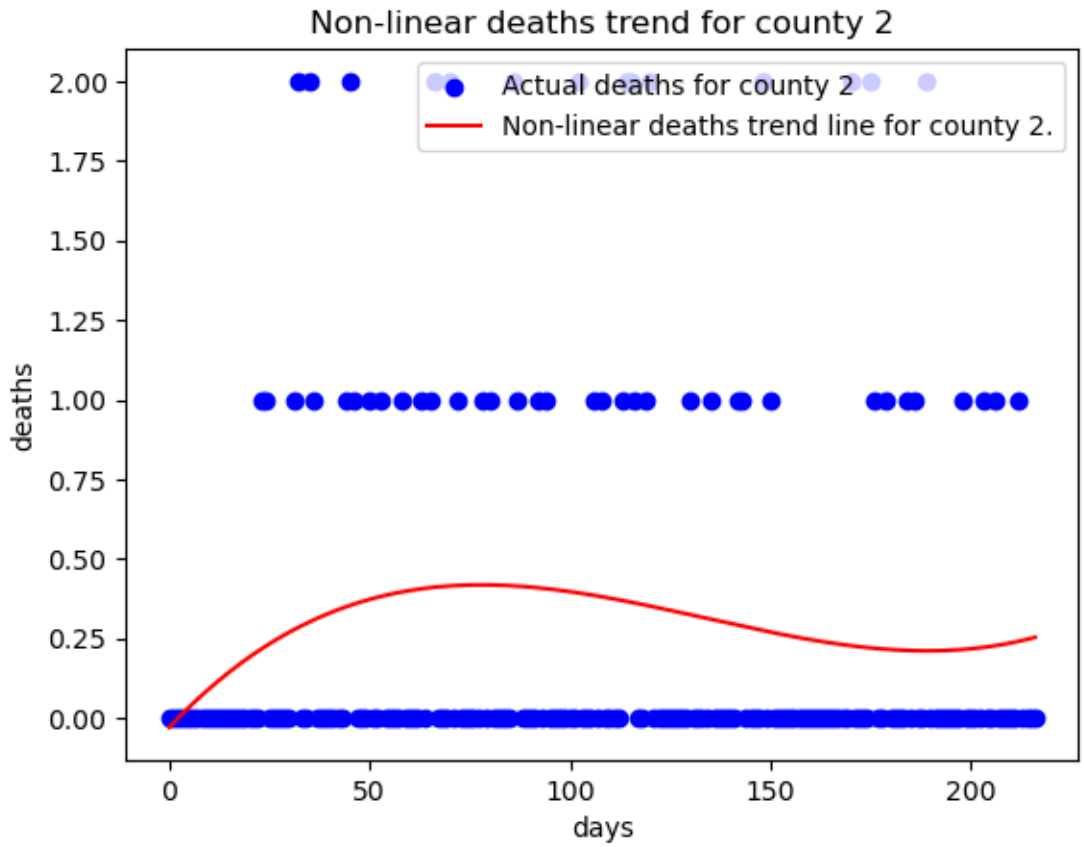


[50]:
```
# RMSE error for non-linear cases data for county 2.

originalPolyCasesC2 = yCasesCounty2
predicPolyCasesC2 = casesPredictionCounty2
rmse = mean_squared_error(originalPolyCasesC2, predicPolyCasesC2, squared=False)
rmse
```

[50]:
```
97.85964006409478
```

[51]:
```
# plotting non-linear deaths trend line for county 2.

plt.scatter(xAxis, yDeathsCounty2, color='blue', label='Actual deaths for␣
 ↪county 2')
plt.plot(xAxis, deathsPredictionCounty2, color='red', label='Non-linear deaths␣
 ↪trend line for county 2.')
plt.title('Non-linear deaths trend for county 2')
plt.xlabel('days')
plt.ylabel('deaths')
plt.legend()
```

[51]:
```
<matplotlib.legend.Legend at 0x1e9f1eccb50>
```



[52]:
```
# RMSE for non-linear deaths data for county 2.

originalPolyDeathsC2 = yDeathsCounty2
predicPolyDeathsC2 = deathsPredictionCounty2
rmse = mean_squared_error(originalPolyDeathsC2, predicPolyDeathsC2,␣
 ↪squared=False)
rmse
```

[52]:
```
0.5701946717169496
```

[53]:
```
# Reading county 3 cases and deaths.

yCasesCounty3 = county3['Number of new cases']
yDeathsCounty3 = county3['Number of new Deaths']
```

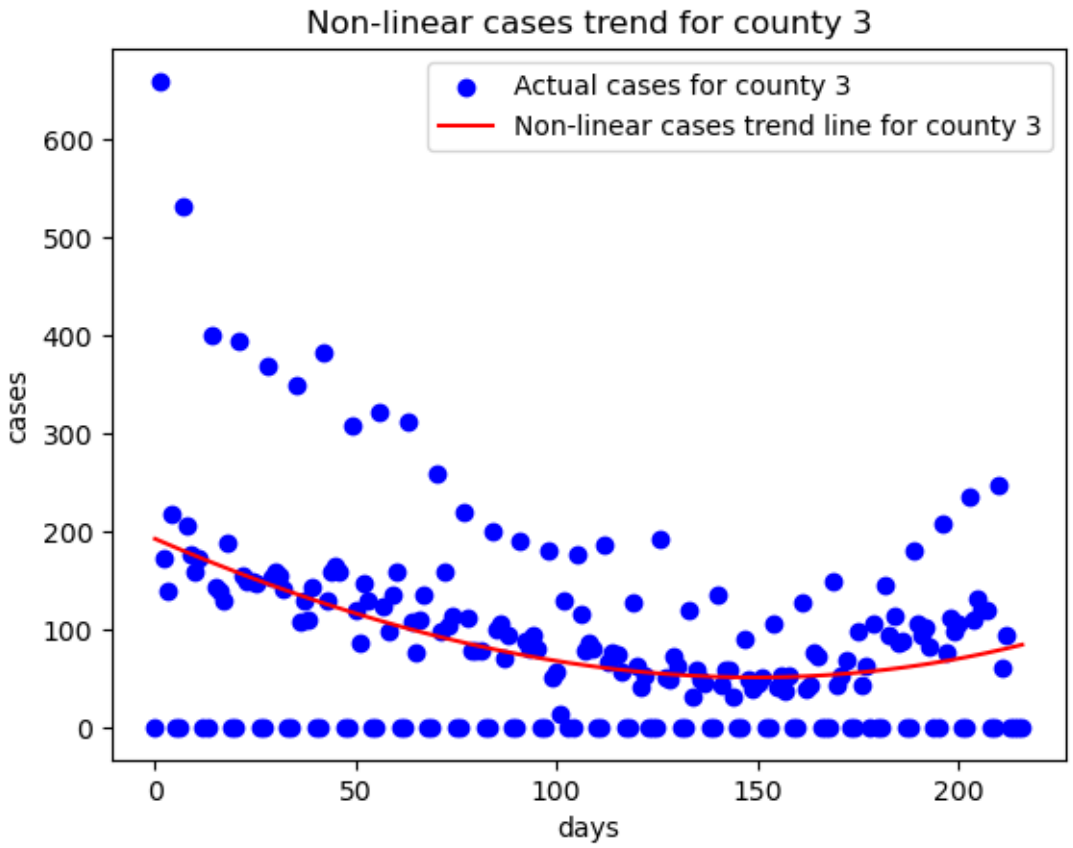
[54]:
```
# Fitting the polynomial data.

linearModelCases.fit(xAxisPoly, yCasesCounty3)
linearModelDeaths.fit(xAxisPoly, yDeathsCounty3)
```

[54]:
```
LinearRegression()
```

[55]:
```
# Generating non-linear cases and deaths trend for county 3.

casesPredictionCounty3 = linearModelCases.predict(xAxisPoly)
deathsPredictionCounty3 = linearModelDeaths.predict(xAxisPoly)
```

[56]:
```
# plotting non-linear cases trend line for county 3.

plt.scatter(xAxis, yCasesCounty3, color='blue', label='Actual cases for county␣
 ↪3')
plt.plot(xAxis, casesPredictionCounty3, color='red', label='Non-linear cases␣
 ↪trend line for county 3')
plt.title('Non-linear cases trend for county 3')
plt.xlabel('days')
plt.ylabel('cases')
plt.legend()
```

[56]:
```
<matplotlib.legend.Legend at 0x1e9f247a700>
```

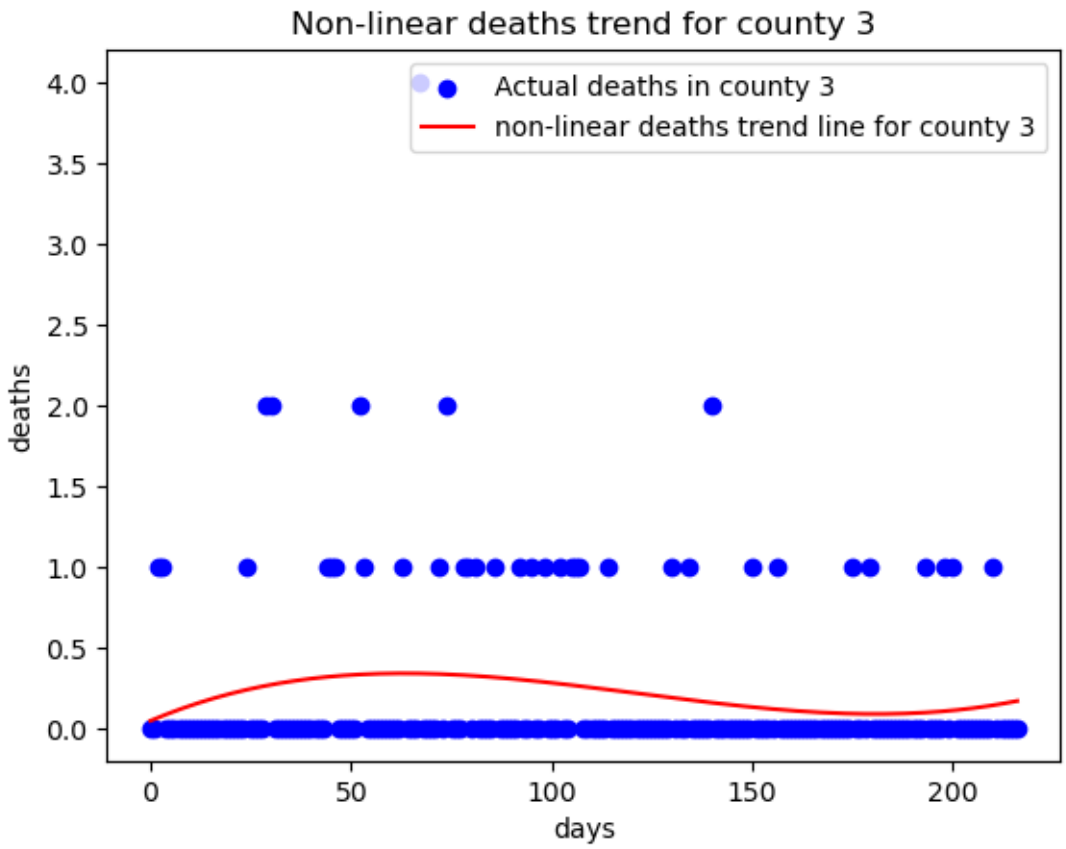


[57]:
```
# RMSE error for non-linear cases data for county 3

originalPolyCasesC3 = yCasesCounty3
predicPolyCasesC3 = casesPredictionCounty3
rmse = mean_squared_error(originalPolyCasesC3, predicPolyCasesC3, squared=False)
rmse
```

[57]: 88.54968441630119

[58]:
```
# plotting non-linear deaths trend line for county 3.

plt.scatter(xAxis, yDeathsCounty3, color='blue', label='Actual deaths in county␣
 ↪3')
plt.plot(xAxis, deathsPredictionCounty3, color='red', label='non-linear deaths␣
 ↪trend line for county 3')
plt.title('Non-linear deaths trend for county 3')
plt.xlabel('days')
plt.ylabel('deaths')
plt.legend()
```

[58]: <matplotlib.legend.Legend at 0x1e9eb5fe1c0>



[59]:
```
# RMSE eeror for non-linear deaths data.

originalPolyDeathsC3 = yDeathsCounty3
predicPolyDeathsC3 = deathsPredictionCounty3
rmse = mean_squared_error(originalPolyDeathsC3, predicPolyDeathsC3,␣
 ↪squared=False)
rmse
```

[59]: 0.5072846223995535

[60]:
```
# Reading cases and deaths data for county 4.

yCasesCounty4 = county4['Number of new cases']
yDeathsCounty4 = county4['Number of new Deaths']
```

[61]:
```
# fitting non-linear data.

linearModelCases.fit(xAxisPoly, yCasesCounty4)
```

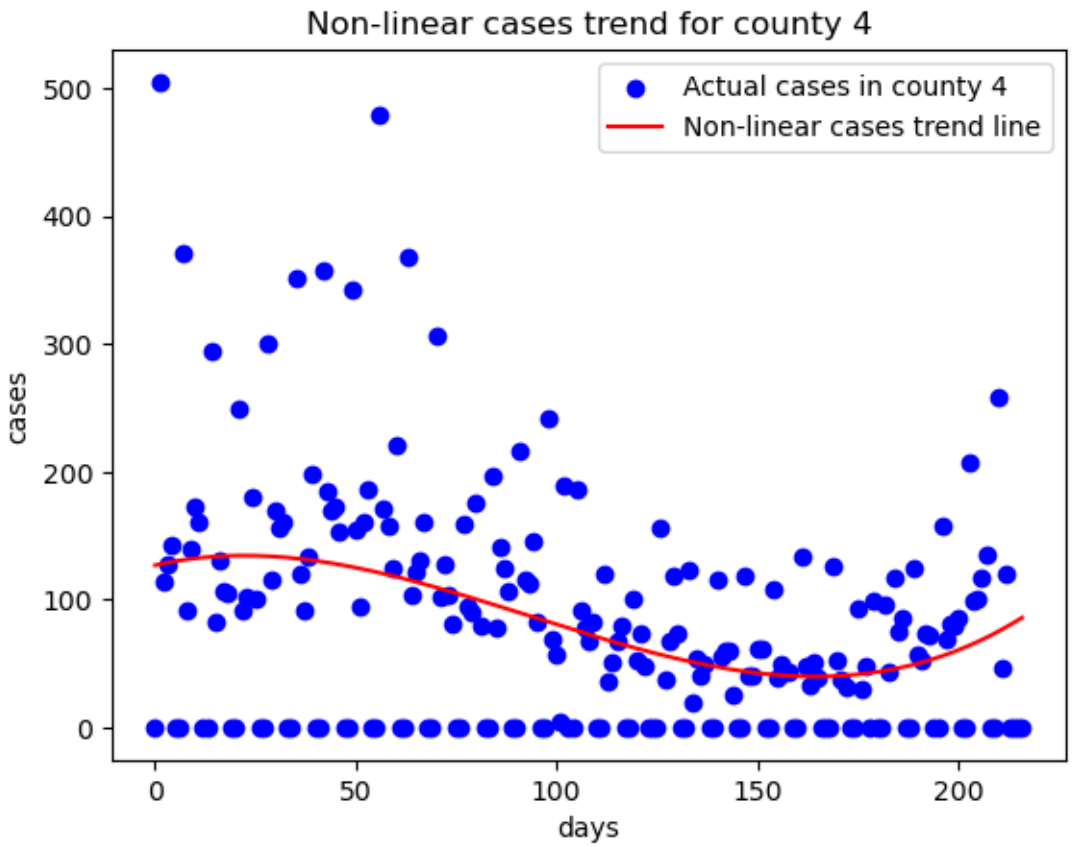
```
linearModelDeaths.fit(xAxisPoly, yDeathsCounty4)
```

[61]:
```
LinearRegression()
```

[62]:
```
# generating non-linear cases and deaths trend data.

casesPredictionCounty4 = linearModelCases.predict(xAxisPoly)
deathsPredictionCounty4 = linearModelDeaths.predict(xAxisPoly)
```

[63]:
```
# plotting non-linear cases trend line for county 4.

plt.scatter(xAxis, yCasesCounty4, color='blue', label='Actual cases in county␣
 ↪4')
plt.plot(xAxis, casesPredictionCounty4, color='red', label='Non-linear cases␣
 ↪trend line')
plt.title('Non-linear cases trend for county 4')
plt.xlabel('days')
plt.ylabel('cases')
plt.legend()
```

[63]:
```
<matplotlib.legend.Legend at 0x1e9f2497be0>
```

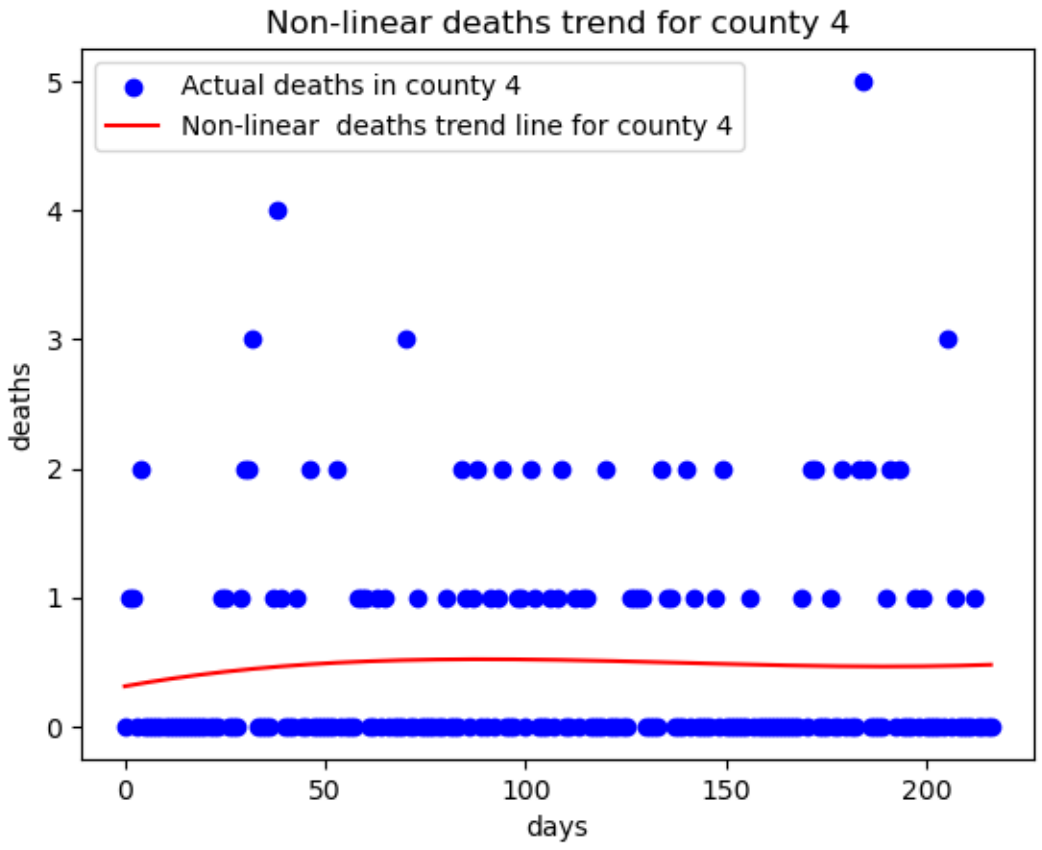


[64]:
```
# RMSE error for non-linear cases data of county 4

originalPolyCasesC4 = yCasesCounty4
predicPolyCasesC4 = casesPredictionCounty4
rmse = mean_squared_error(originalPolyCasesC4, predicPolyCasesC4, squared=False)
rmse
```

[64]:
```
83.31105867501222
```

[65]:
```
# plotting non-linear deaths trend line of county 4.

plt.scatter(xAxis, yDeathsCounty4, color='blue', label='Actual deaths in county␣
 ↪4')
plt.plot(xAxis, deathsPredictionCounty4, color='red', label='Non-linear  deaths␣
 ↪trend line for county 4')
plt.title('Non-linear deaths trend for county 4')
plt.xlabel('days')
plt.ylabel('deaths')
plt.legend()
```

[65]:
```
<matplotlib.legend.Legend at 0x1e9f2377a00>
```



[66]:
```
# RMSE error for non-linear deaths data.

originalPolyDeathsC4 = yDeathsCounty4
predicPolyDeathsC4 = deathsPredictionCounty4
rmse = mean_squared_error(originalPolyDeathsC4, predicPolyDeathsC4,␣
 ↪squared=False)
rmse
```

[66]:
```
0.8193120681433574
```

[67]:
```
# reading case and deaths data of county 5.

yCasesCounty5 = county5['Number of new cases']
yDeathsCounty5 = county5['Number of new Deaths']
```

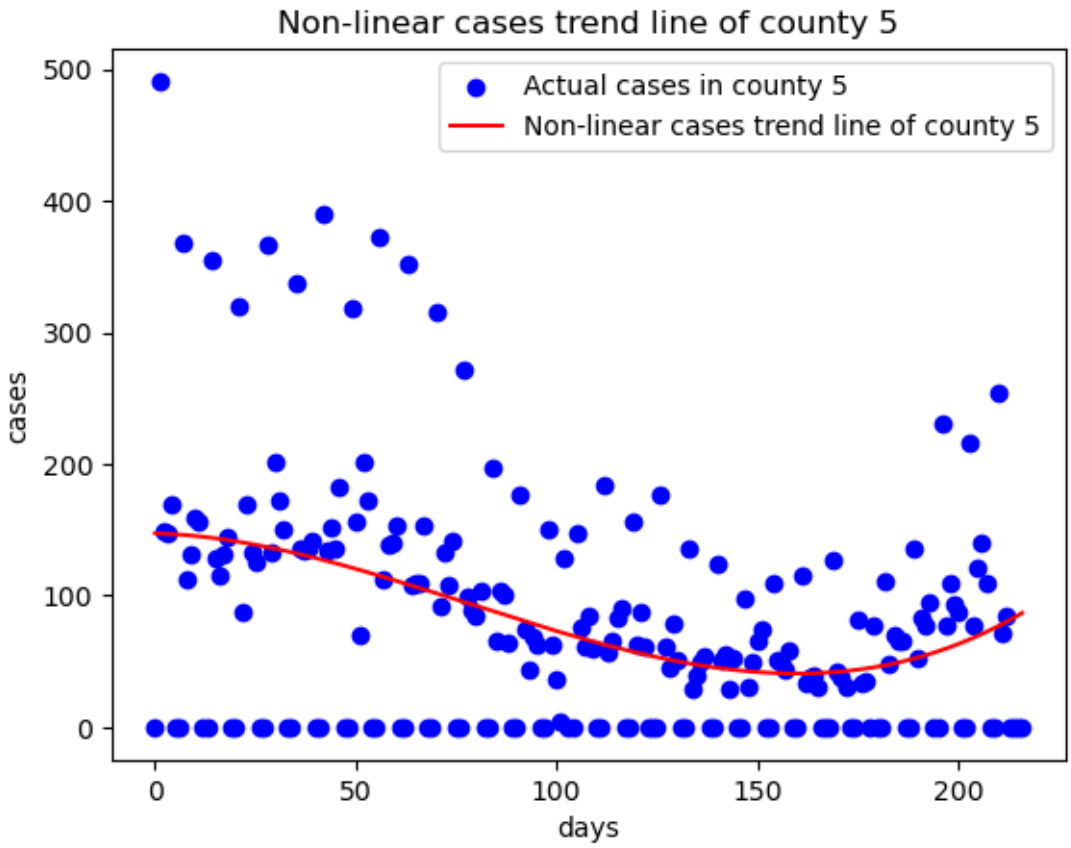
[68]:
```
# fitting non-linear data.

linearModelCases.fit(xAxisPoly, yCasesCounty5)
linearModelDeaths.fit(xAxisPoly, yDeathsCounty5)
```

[68]:
```
LinearRegression()
```

[69]:
```
# Generating non-linear trend data of cases and deaths in county 5

casesPredictionCounty5 = linearModelCases.predict(xAxisPoly)
deathsPredictionCounty5 = linearModelDeaths.predict(xAxisPoly)
```

[70]:
```
# plotting non-linear cases trend line of county 5

plt.scatter(xAxis, yCasesCounty5, color='blue', label='Actual cases in county␣
 ↪5')
plt.plot(xAxis, casesPredictionCounty5, color='red', label='Non-linear cases␣
 ↪trend line of county 5')
plt.title('Non-linear cases trend line of county 5')
plt.xlabel('days')
plt.ylabel('cases')
plt.legend()
```

[70]:
```
<matplotlib.legend.Legend at 0x1e9f244d5b0>
```

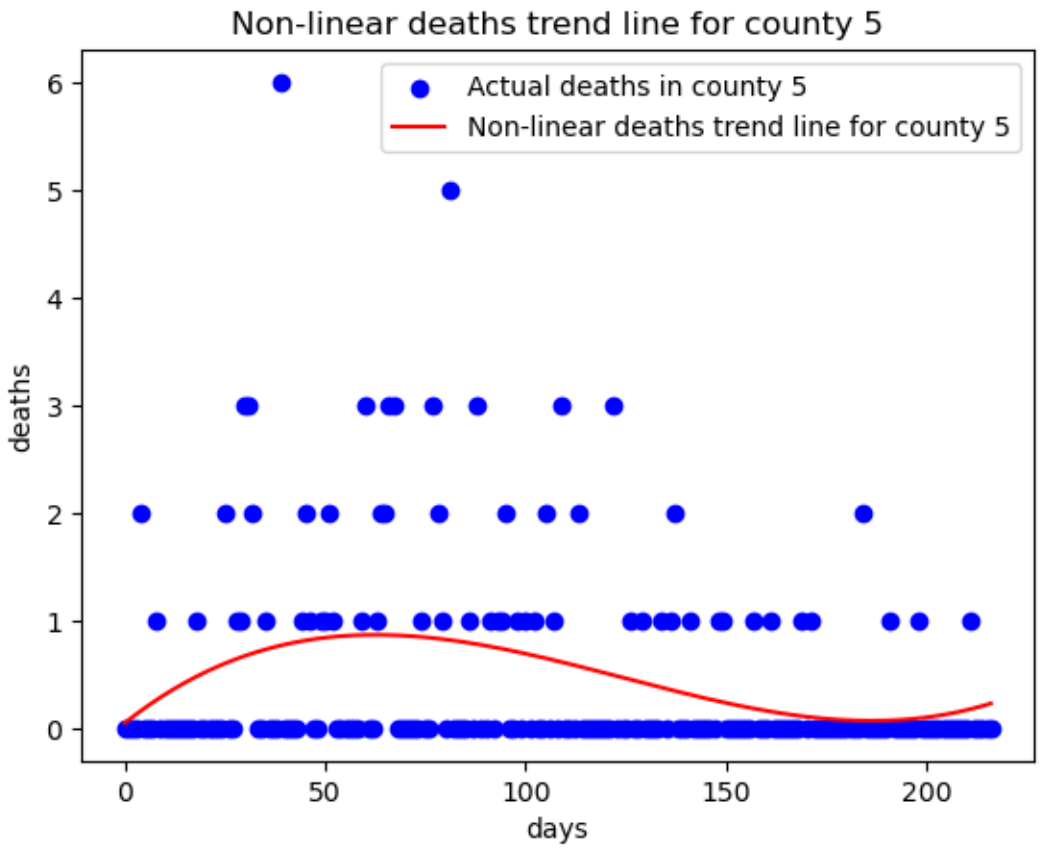


[71]:
```
# RMSE error for non-linear cases data of county 5.

originalPolyCasesC5 = yCasesCounty5
predicPolyCasesC5 = casesPredictionCounty5
rmse = mean_squared_error(originalPolyCasesC5, predicPolyCasesC5, squared=False)
rmse
```

[71]: 82.41510437035109

[72]:
```
# plotting non-linear deaths trend line of county 5.

plt.scatter(xAxis, yDeathsCounty5, color='blue', label='Actual deaths in county␣
 ↪5')
plt.plot(xAxis, deathsPredictionCounty5, color='red', label='Non-linear deaths␣
 ↪trend line for county 5')
plt.title('Non-linear deaths trend line for county 5')
plt.xlabel('days')
plt.ylabel('deaths')
plt.legend()
```

[72]: <matplotlib.legend.Legend at 0x1e9f244df40>



[73]:
```
# RMSE error for non-linear deaths data of county 5.

originalPolyDeathsC5 = yDeathsCounty5
predicPolyDeathsC5 = deathsPredictionCounty5
rmse = mean_squared_error(originalPolyDeathsC5, predicPolyDeathsC5,␣
 ↪squared=False)
rmse
```

[73]: 0.8738202065035134

3 Hypothesis Testing

[74]:
```
# Reading enrichment data for hypothesis testing.

enrichmentCombined = pd.read_csv("../../Member/Pulibandla-Venkatesh/
 ↪enrichmentCombined.csv")
enrichmentCombined.drop('Unnamed: 0', axis=1, inplace = True)
```

```
enrichmentCombined
```

```
C:\Users\venka\AppData\Local\Temp\ipykernel_17608\33891018.py:3: DtypeWarning:
Columns (23,24,25,26,27) have mixed types. Specify dtype option on import or set
low_memory=False.
  enrichmentCombined = pd.read_csv("../../Member/Pulibandla-
Venkatesh/enrichmentCombined.csv")
```

[74]:

	County Name	State	Date	Number of new cases	\
0	Baldwin County	AL	2022-05-30	55	
1	Baldwin County	AL	2022-05-31	183	
2	Baldwin County	AL	2022-06-01	68	
3	Baldwin County	AL	2022-06-02	68	
4	Baldwin County	AL	2022-06-03	0	
…	…	…	…	…	
180105	Natrona County	WY	2022-12-28	0	
180106	Natrona County	WY	2022-12-29	0	
180107	Natrona County	WY	2022-12-30	0	
180108	Natrona County	WY	2022-12-31	0	
180109	Natrona County	WY	2023-01-01	0	

	Total Population	Male population	Female Population	Sex Ratio	\
0	239294	115696	123598	93.6	
1	239294	115696	123598	93.6	
2	239294	115696	123598	93.6	
3	239294	115696	123598	93.6	
4	239294	115696	123598	93.6	
…	…	…	…	…	
180105	79555	41070	38485	106.7	
180106	79555	41070	38485	106.7	
180107	79555	41070	38485	106.7	
180108	79555	41070	38485	106.7	
180109	79555	41070	38485	106.7	

	Population under 5 Years	Population 5 to 9 Years	…	\
0	12360	12848	…	
1	12360	12848	…	
2	12360	12848	…	
3	12360	12848	…	
4	12360	12848	…	
…	…	…	…	
180105	4733	4844	…	
180106	4733	4844	…	
180107	4733	4844	…	
180108	4733	4844	…	
180109	4733	4844	…	

```
        Population of White Race  Population of Black Race  \
0                         198355                     21305
1                         198355                     21305
2                         198355                     21305
3                         198355                     21305
4                         198355                     21305
…                          …                        …
180105                     69303                        72
180106                     69303                        72
180107                     69303                        72
180108                     69303                        72
180109                     69303                        72

        Population of American Indian and Alaska Native  \
0                                                   884
1                                                   884
2                                                   884
3                                                   884
4                                                   884
…                                                  …
180105                                              598
180106                                              598
180107                                              598
180108                                              598
180109                                              598

        Population of Asian Race  \
0                           1956
1                           1956
2                           1956
3                           1956
4                           1956
…                          …
180105                       228
180106                       228
180107                       228
180108                       228
180109                       228

        Population of Native Hawaiian and Other Pacific Islander  \
0                                                               0
1                                                               0
2                                                               0
3                                                               0
4                                                               0
…                                                              …
180105                                                          0
```

```
180106                                                  0
180107                                                  0
180108                                                  0
180109                                                  0

        Total Housing Units  Population of Citizen Over 18  \
0                    128533                         185566
1                    128533                         185566
2                    128533                         185566
3                    128533                         185566
4                    128533                         185566
…                       …                              …
180105                37051                          59929
180106                37051                          59929
180107                37051                          59929
180108                37051                          59929
180109                37051                          59929

        Population of Male Citizen  Population of Female Citizen  population
0                            90007                         95559      223234
1                            90007                         95559      223234
2                            90007                         95559      223234
3                            90007                         95559      223234
4                            90007                         95559      223234
…                              …                             …           …
180105                       30603                         29326       79858
180106                       30603                         29326       79858
180107                       30603                         29326       79858
180108                       30603                         29326       79858
180109                       30603                         29326       79858

[180110 rows x 32 columns]
```

[75]:
```
# Cleaning the data and grouping by county name.

enrichmentCombined[enrichmentCombined['Number of new cases'] <0] =0
enrichmentCombined = enrichmentCombined.groupby(['County Name']).sum()
enrichmentCombined
```

[75]:
```
                    Number of new cases  Total Population  Male population  \
County Name
0                                     0                 0                0
Ada County                        20339         111089027         55841695
Adams County                      31065         135377799         68539826
Aiken County                          0          37058392         18042899
Alachua County                    15584          60594646         29320172
…                                     …                 …                …
```

Yellowstone County	6793	36270682	17912916
Yolo County	9810	46435004	22600326
York County	30608	223902358	110133772
Yuba County	3286	17852094	8934500
Yuma County	7065	44916830	23366994

County Name	Female Population	Sex Ratio	Population under 5 Years	\
0	0	0.0	0	
Ada County	55247332	21938.7	5833394	
Adams County	66837973	43817.6	8082110	
Aiken County	19015493	20593.3	2004646	
Alachua County	31274474	20354.6	2931236	
…	…	…	…	
Yellowstone County	18357766	21179.2	1904826	
Yolo County	23834678	20287.2	2427402	
York County	113768586	83751.7	11641145	
Yuba County	8917594	21442.8	1318240	
Yuma County	21549836	23522.8	3061870	

County Name	Population 5 to 9 Years	Population 10 to 14 Years	\
0	0	0	
Ada County	6635643	7934171	
Adams County	8671449	10176562	
Aiken County	2327759	2166528	
Alachua County	3434242	2820566	
…	…	…	
Yellowstone County	2326240	2759589	
Yolo County	2646966	2648678	
York County	14092712	13968445	
Yuba County	1228574	1623832	
Yuma County	3193155	3114601	

County Name	Population 15 to 19 Years	Population 20 to 24 Years	\
0	0	0	
Ada County	7344365	6512170	
Adams County	9436125	8755300	
Aiken County	2679733	1610357	
Alachua County	5176318	8734901	
…	…	…	
Yellowstone County	2274594	1995966	
Yolo County	4617692	6310004	
York County	14548725	12070958	
Yuba County	1190268	1214450	
Yuma County	3147802	3350480	

```
                    …  Population 60 to 64 Years  \
County Name         …
0                   …                         0
Ada County          …                   6273036
Adams County        …                   8206990
Aiken County        …                   2972466
Alachua County      …                   3719380
…                   …                         …
Yellowstone County  …                   2361394
Yolo County         …                   2135292
York County         …                  15774884
Yuba County         …                   1158596
Yuma County         …                   2083417

                    Population 65 to 74 Years  Population 75 to 84 Years  \
County Name
0                                           0                         0
Ada County                           10836112                   4875773
Adams County                         11081140                   4528566
Aiken County                          4797653                   2166962
Alachua County                        5519612                   2154376
…                                           …                         …
Yellowstone County                    3869327                   2063887
Yolo County                           3786088                   1577608
York County                          24729734                  11165263
Yuba County                           1451348                    723106
Yuma County                           4394901                   3551205

                    Population Above 85 Years  Meadin Age Population  \
County Name
0                                           0                    0.0
Ada County                            1523774                 8289.4
Adams County                          1767452                17043.1
Aiken County                           734111                 8962.1
Alachua County                        1424388                 7117.6
…                                           …                      …
Yellowstone County                     536207                 8246.0
Yolo County                            819192                 6997.8
York County                           4330682                35868.2
Yuba County                            193028                 7297.4
Yuma County                            770784                 7681.8

                    Total Housing Units  Population of Citizen Over 18  \
County Name
0                                     0                              0
Ada County                     44397332                       82639025
```

```
Adams County                    50727884                    91239687
Aiken County                    16998478                    28311556
Alachua County                  27237406                    47007625
…                                  …                           …
Yellowstone County              15942122                    27322253
Yolo County                     17387928                    32566734
York County                     97101447                   171415846
Yuba County                      6414222                    11949546
Yuma County                     20353949                    28473004

                    Population of Male Citizen  Population of Female Citizen  \
County Name
0                                            0                             0
Ada County                            41564831                      41074194
Adams County                          45747884                      45491803
Aiken County                          13428394                      14883162
Alachua County                        22658489                      24349136
…                                        …                             …
Yellowstone County                    13403222                      13919031
Yolo County                           15546244                      17020490
York County                           83410382                      88005464
Yuba County                            5961612                       5987934
Yuma County                           14854518                      13618486

                    population
County Name
0                            0
Ada County           104504379
Adams County         134115889
Aiken County          37079224
Alachua County        58382331
…                        …
Yellowstone County    35002100
Yolo County           47187000
York County          218085245
Yuba County           16834952
Yuma County           46391779

[662 rows x 24 columns]
```

4 Hypothesis question 1:

H0 : Population of age group from 60 to 64 has impact on increase of covid cases. H1 : population of age group from 60 to 64 has no impact on increase of covid cases.

Lets calculate p value whether to accept or reject the null hypothesis.

[76]:
```
stats.ttest_ind(a=enrichmentCombined['Population 60 to 64 Years'], b =␣
↪enrichmentCombined['Number of new cases'], equal_var=False)
```

[76]:
```
Ttest_indResult(statistic=16.14805442202087, pvalue=1.082495253753287e-49)
```

The p value value is less than 0.05, so we reject the null hypothesis. So, Population of age group 60 to 64 has no impact on increase of covid cases.

5 Hypothesis question 2:

H0 : Population of male citizen has impact on increase of covid cases. H1 : population of male citizen has no impact on increase of covid cases.

Lets calculate p value whether to accept or reject the null hypothesis.

[77]:
```
stats.ttest_ind(enrichmentCombined['Population of Male Citizen'],␣
↪enrichmentCombined['Number of new cases'])
```

[77]:
```
Ttest_indResult(statistic=16.380868346519275, pvalue=4.7473638230939255e-55)
```

The p value is less than 0.05, so we reject the null hypothesis. So, population of male citizen has no impact on increase of covid cases.

6 Hypothesis question 3:

H0 : Total housing units has impact on increase of covid cases. H1 : Total housing units has no impact on increase of covid cases.

Lets calculate p value whether to accept or reject the null hypothesis.

[78]:
```
stats.ttest_ind(enrichmentCombined['Total Housing Units'],␣
↪enrichmentCombined['Number of new cases'])
```

[78]:
```
Ttest_indResult(statistic=16.237228021311182, pvalue=3.36193155125312e-54)
```

The p value is less than 0.05 , so we reject the null hypothesis. So, Total housing units has no impact on increase ofcovid cases.

7 Confidence Intervals

[79]:
```
x= xAxis['days']
x
```

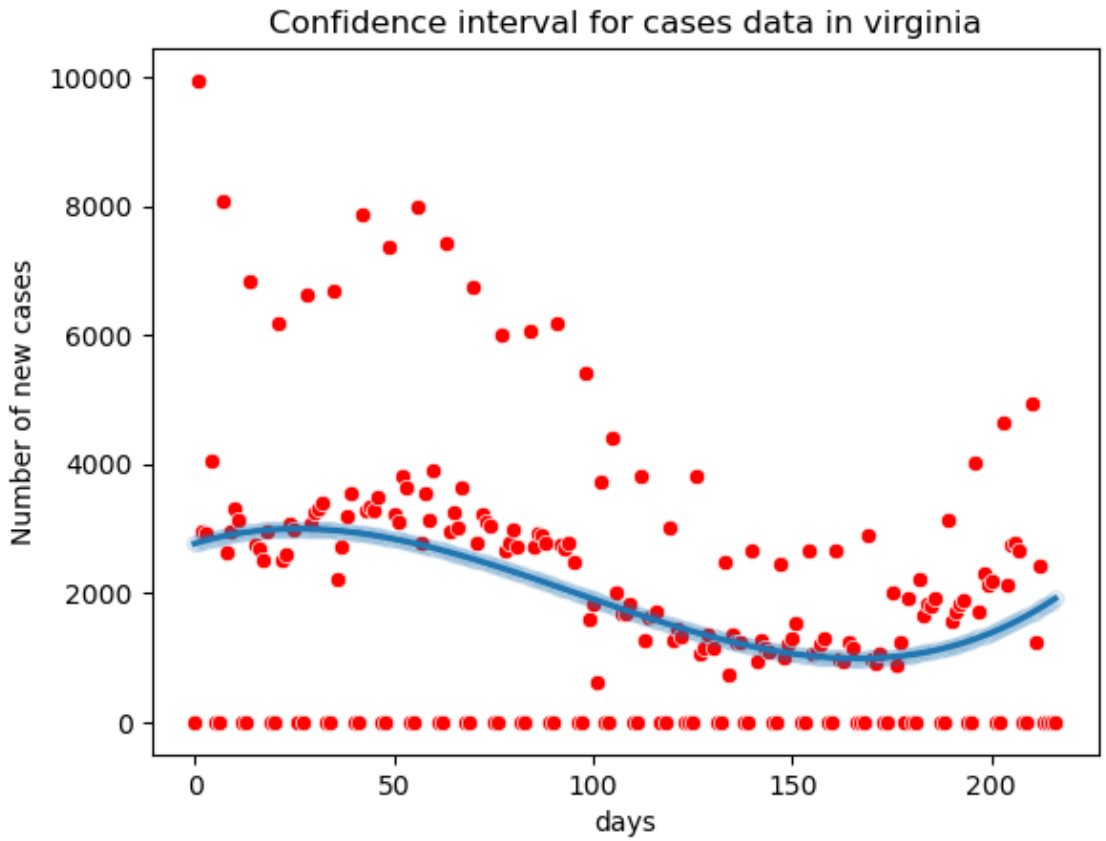
[79]:
```
0        0
1        1
2        2
3        3
4        4
      ...
```

```
212    212
213    213
214    214
215    215
216    216
Name: days, Length: 217, dtype: int64
```

[80]:
```python
# Plotting the confidence interval for cases data in virginia.

plt.title('Confidence interval for cases data in virginia')
sns.scatterplot(x, yCases, color='red')
sns.lineplot(x, polyCasesPrediction, color='red')
sns.regplot(xAxis['days'], polyCasesPrediction, scatter_kws={'alpha':0.1},␣
 ↪order=3)
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

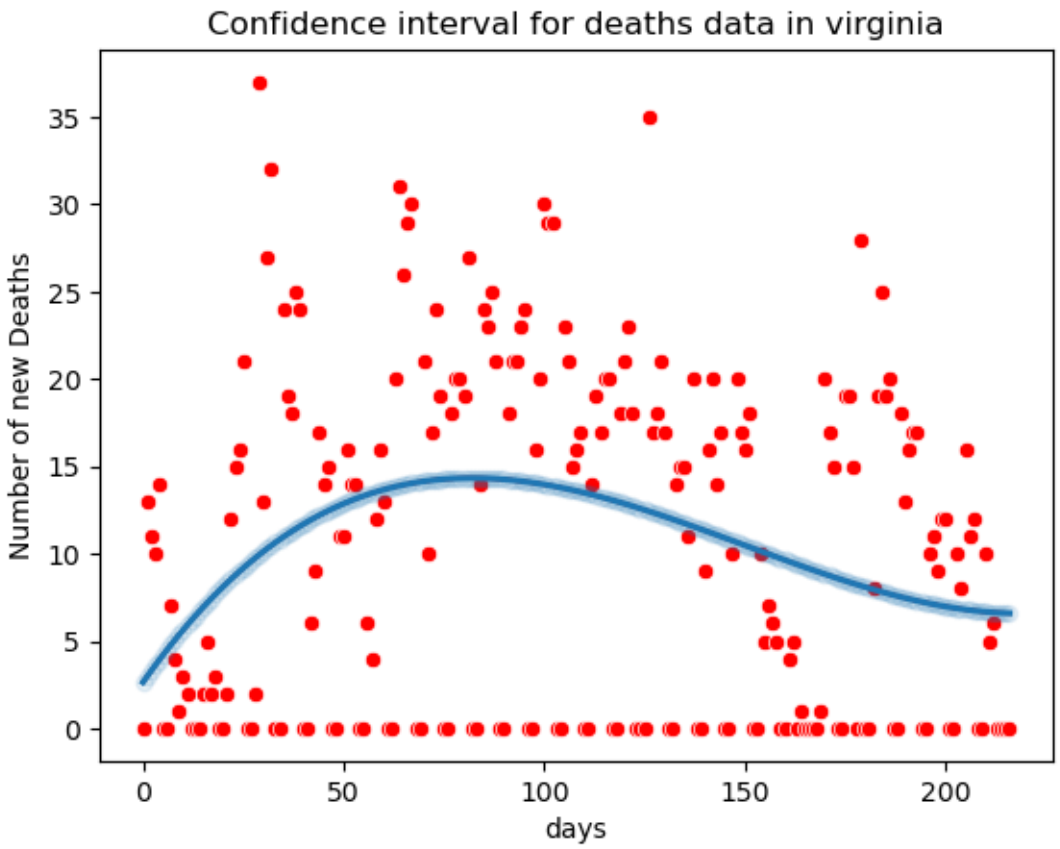
[80]:
```
<AxesSubplot:title={'center':'Confidence interval for cases data in virginia'},
xlabel='days', ylabel='Number of new cases'>
```



[81]:
```
# plotting confidence interval for deaths data in virginia.

plt.title('Confidence interval for deaths data in virginia')
sns.scatterplot(x, yDeaths, color='red')
sns.lineplot(x, polyDeathsPrediction, color='red')
sns.regplot(xAxis['days'], polyDeathsPrediction, scatter_kws={'alpha':0.1},␣
↪order=3)
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[81]: <AxesSubplot:title={'center':'Confidence interval for deaths data in virginia'}, xlabel='days', ylabel='Number of new Deaths'>

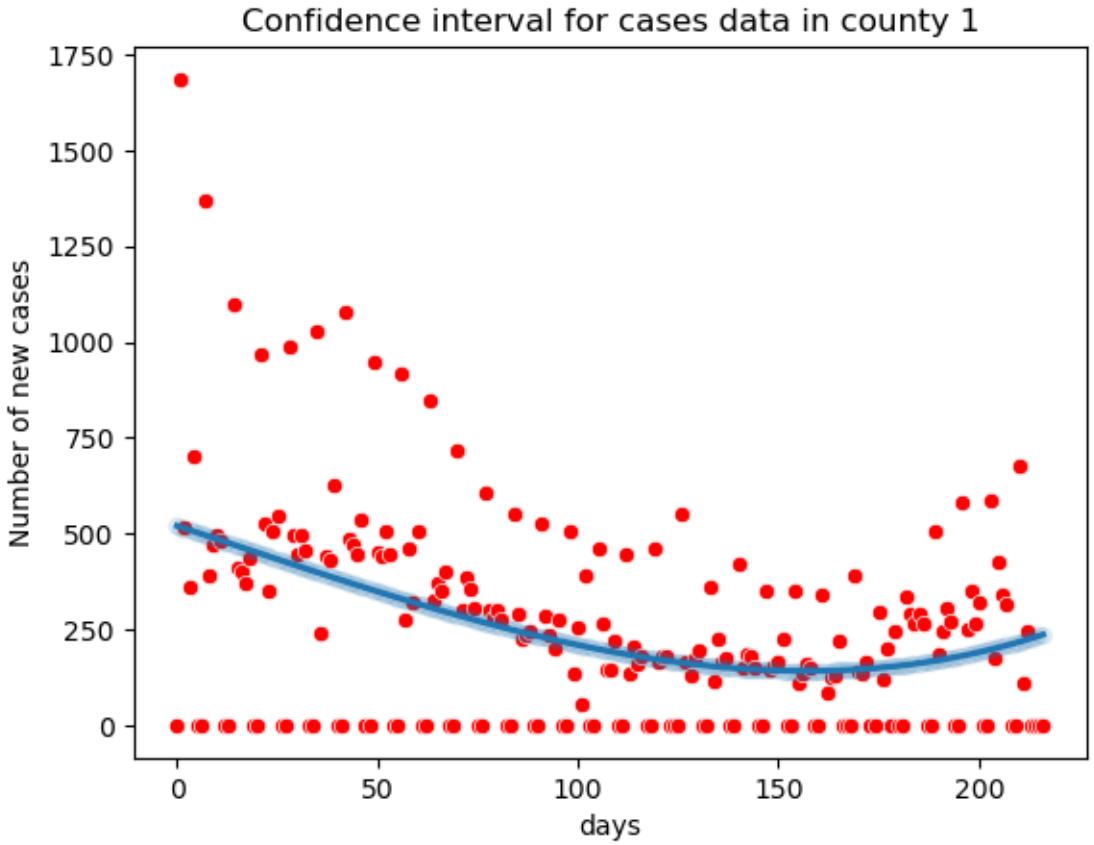


[82]:
```
# plotting confidence interval for cases data in county 1.

plt.title('Confidence interval for cases data in county 1')
sns.scatterplot(x, yCasesCounty1, color='red')
sns.lineplot(x, casesPredictionCounty1, color='red')
sns.regplot(xAxis['days'], casesPredictionCounty1, scatter_kws={'alpha':0.1},␣
 ↪order=3)
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
```

```
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[82]:
```
<AxesSubplot:title={'center':'Confidence interval for cases data in county 1'},
xlabel='days', ylabel='Number of new cases'>
```

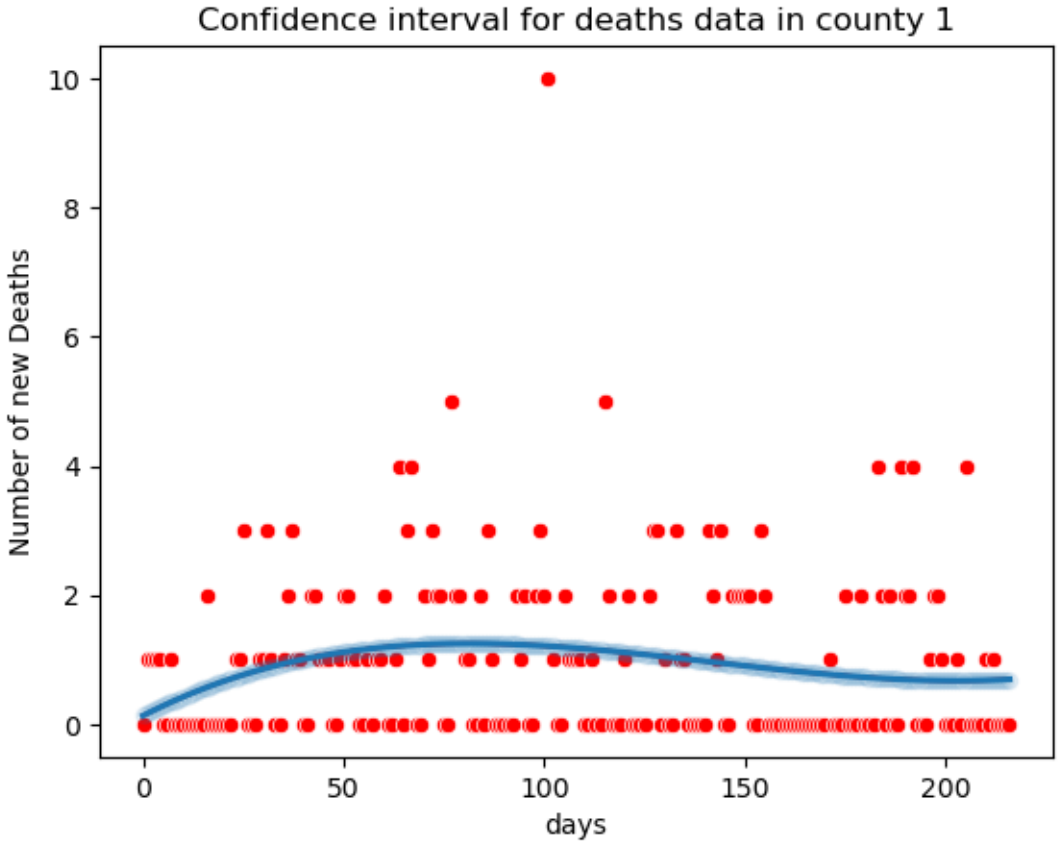


[83]:
```
# plotting confidence interval for deaths data in county 1.

plt.title('Confidence interval for deaths data in county 1')
sns.scatterplot(x, yDeathsCounty1, color='red')
sns.lineplot(x, deathsPredictionCounty1, color='red')
sns.regplot(xAxis['days'], deathsPredictionCounty1, scatter_kws={'alpha':0.1},␣
 ↪order=3)
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[83]:
```
<AxesSubplot:title={'center':'Confidence interval for deaths data in county 1'},
xlabel='days', ylabel='Number of new Deaths'>
```

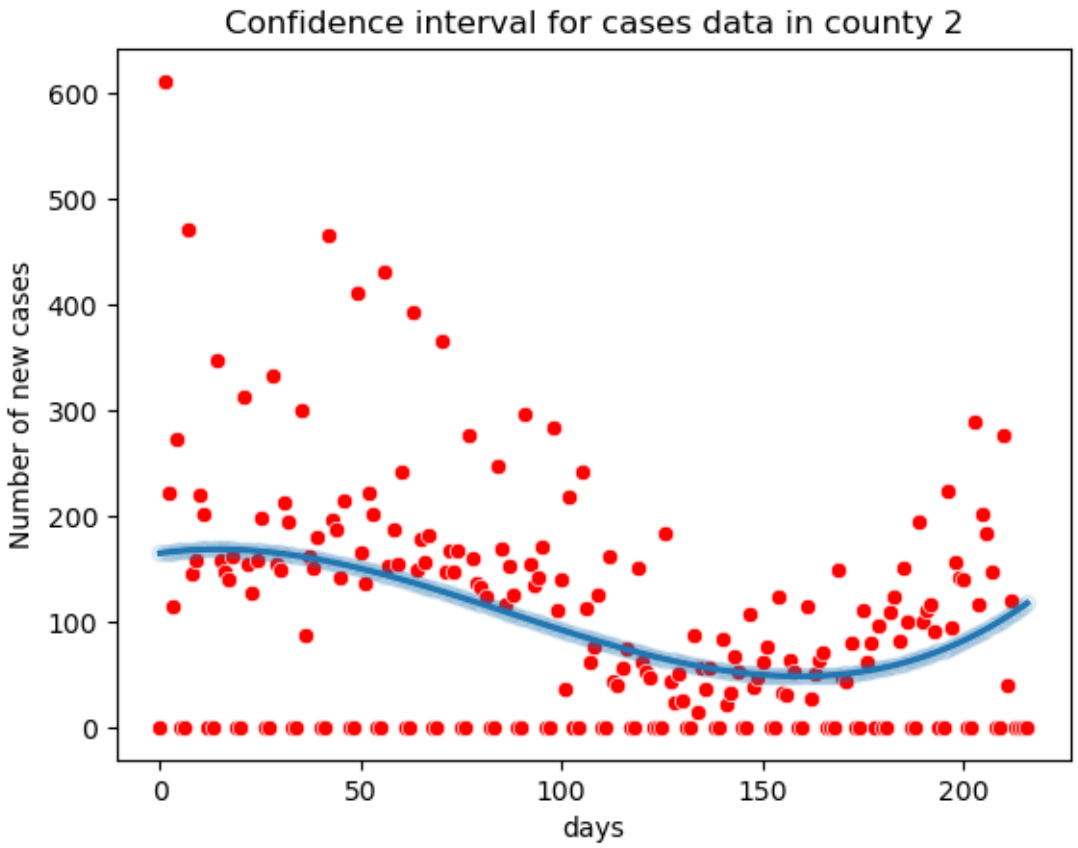


[84]:
```
# plotting confidence interval for cases data in county 2.

plt.title('Confidence interval for cases data in county 2')
sns.scatterplot(x, yCasesCounty2, color='red')
sns.lineplot(x, casesPredictionCounty2, color='red')
sns.regplot(xAxis['days'], casesPredictionCounty2, scatter_kws={'alpha':0.1},␣
 ↪order=3)
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
```

```
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

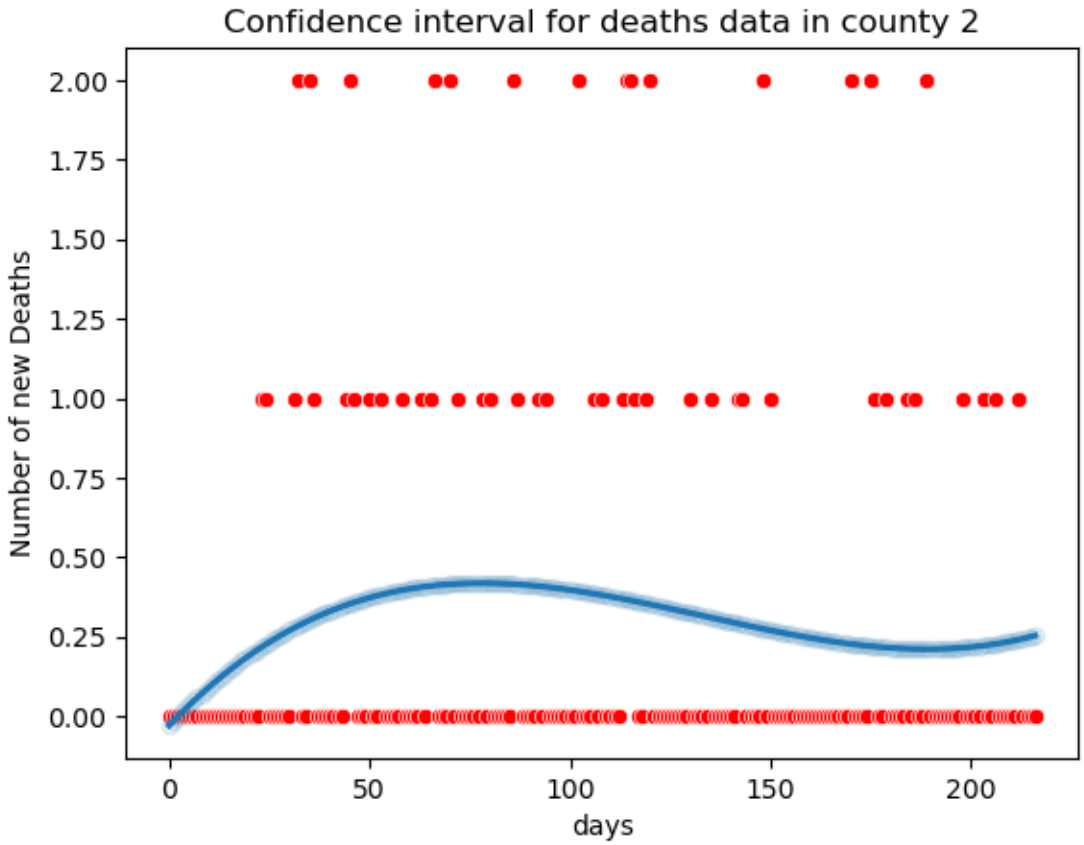
[84]:
```
<AxesSubplot:title={'center':'Confidence interval for cases data in county 2'},
xlabel='days', ylabel='Number of new cases'>
```



[85]:
```
# plotting confidence interval for deaths data in county 2.

plt.title('Confidence interval for deaths data in county 2')
sns.scatterplot(x, yDeathsCounty2, color='red')
sns.lineplot(x, deathsPredictionCounty2, color='red')
sns.regplot(xAxis['days'], deathsPredictionCounty2, scatter_kws={'alpha':0.1},␣
 ↪order=3)
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[85]:
```
<AxesSubplot:title={'center':'Confidence interval for deaths data in county 2'},
xlabel='days', ylabel='Number of new Deaths'>
```

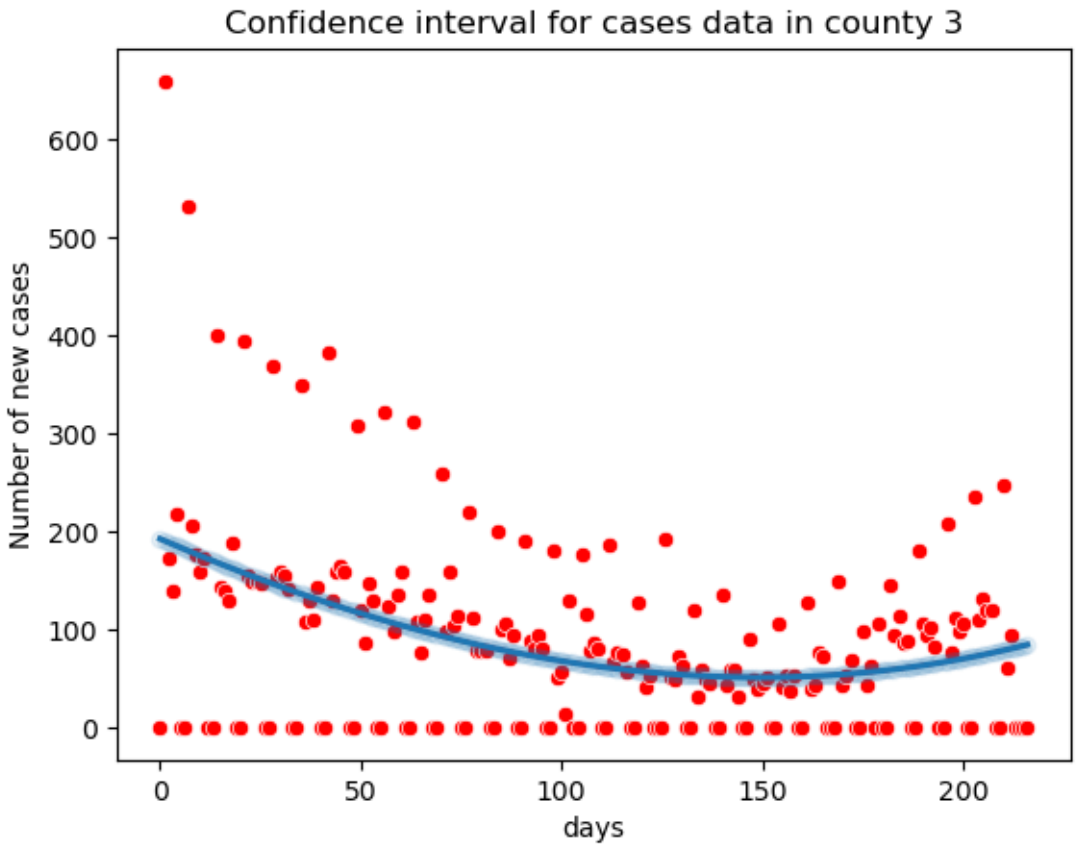


[86]:
```
# plotting confidence interval for cases data in county 3.

plt.title('Confidence interval for cases data in county 3')
sns.scatterplot(x, yCasesCounty3, color='red')
```

```
sns.lineplot(x, casesPredictionCounty3, color='red')
sns.regplot(xAxis['days'], casesPredictionCounty3, scatter_kws={'alpha':0.1},␣
 ↪order=3)
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[86]: <AxesSubplot:title={'center':'Confidence interval for cases data in county 3'}, xlabel='days', ylabel='Number of new cases'>



[87]:
```
# plotting confidence interval for deaths data in county 3.

plt.title('Confidence interval for deaths data in county 3')
sns.scatterplot(x, yDeathsCounty3, color='red')
sns.lineplot(x, deathsPredictionCounty3, color='red')
sns.regplot(xAxis['days'], deathsPredictionCounty3, scatter_kws={'alpha':0.1},␣
↪order=3)
```

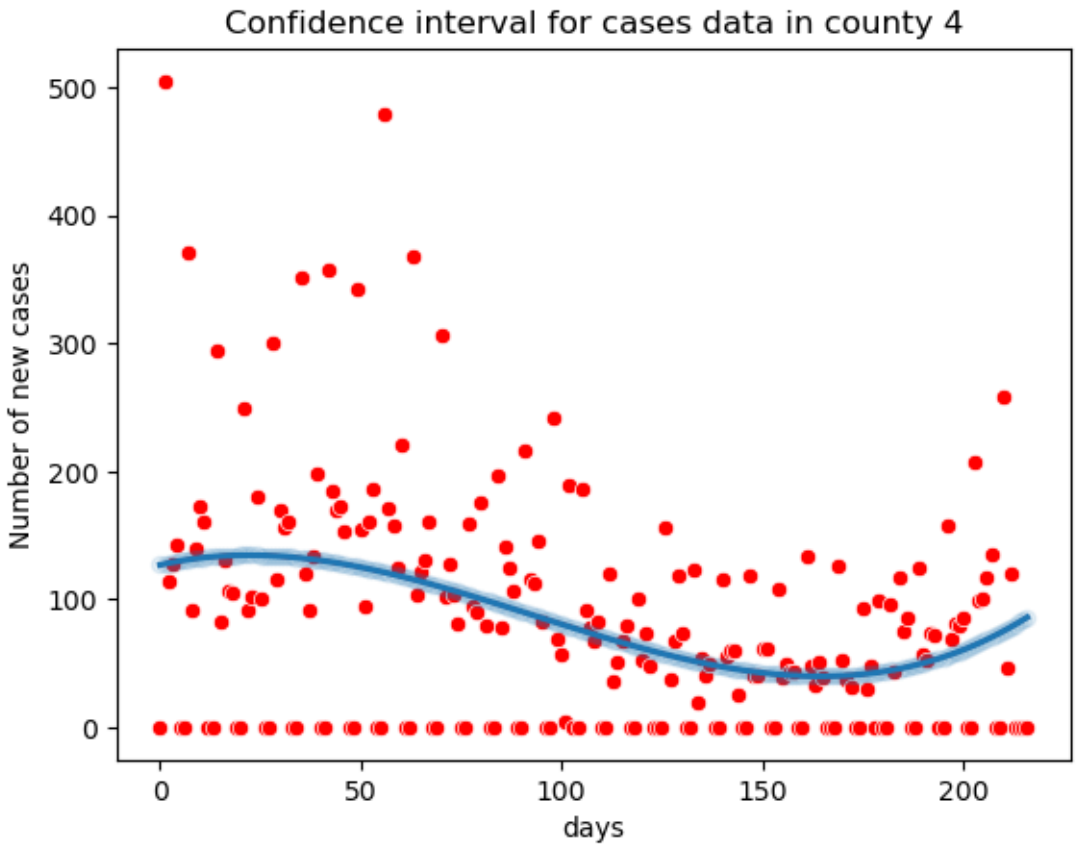
```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[87]:
```
<AxesSubplot:title={'center':'Confidence interval for deaths data in county 3'},
xlabel='days', ylabel='Number of new Deaths'>
```

[88]:
```
# plotting confidence interval for cases data in county 4.

plt.title('Confidence interval for cases data in county 4')
sns.scatterplot(x, yCasesCounty4, color='red')
sns.lineplot(x, casesPredictionCounty4, color='red')
sns.regplot(xAxis['days'], casesPredictionCounty4, scatter_kws={'alpha':0.1},␣
↪order=3)
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

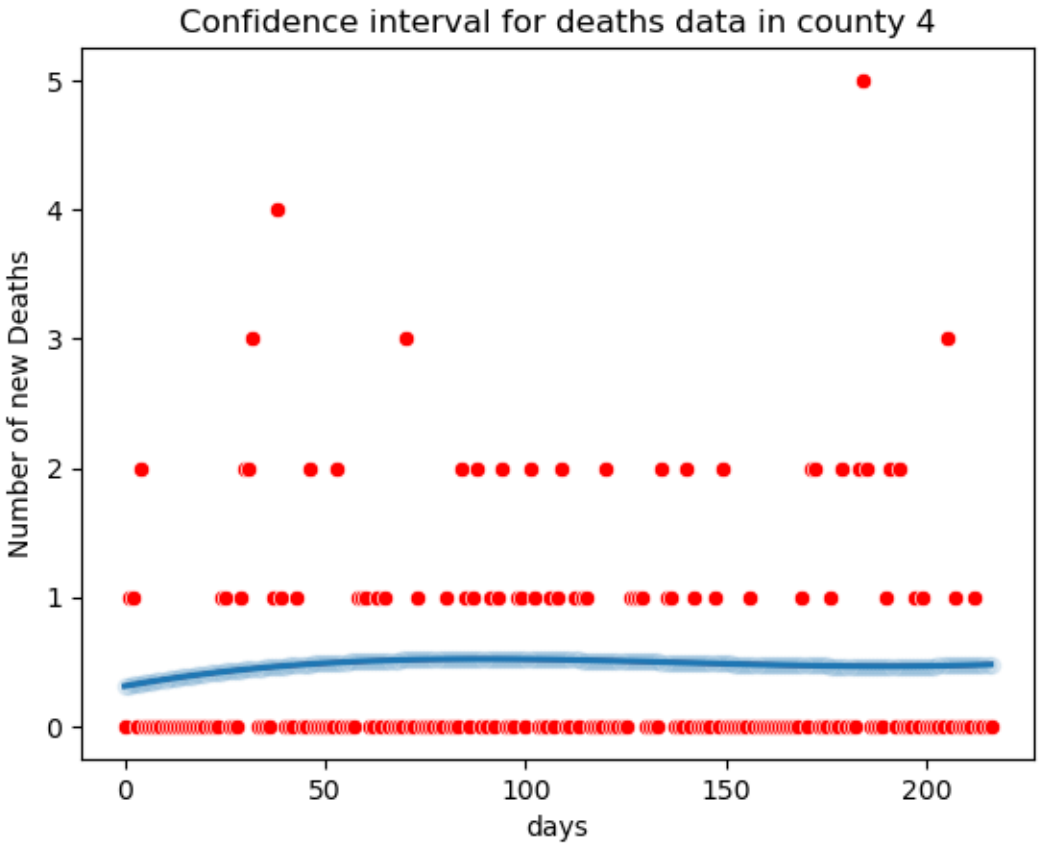
[88]:
```
<AxesSubplot:title={'center':'Confidence interval for cases data in county 4'},
xlabel='days', ylabel='Number of new cases'>
```



[89]:
```
# plotting confidence interval for deaths data in county 4.

plt.title('Confidence interval for deaths data in county 4')
sns.scatterplot(x, yDeathsCounty4, color='red')
sns.lineplot(x, deathsPredictionCounty4, color='red')
sns.regplot(xAxis['days'], deathsPredictionCounty4, scatter_kws={'alpha':0.1},␣
 ↪order=3)
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[89]: <AxesSubplot:title={'center':'Confidence interval for deaths data in county 4'}, xlabel='days', ylabel='Number of new Deaths'>

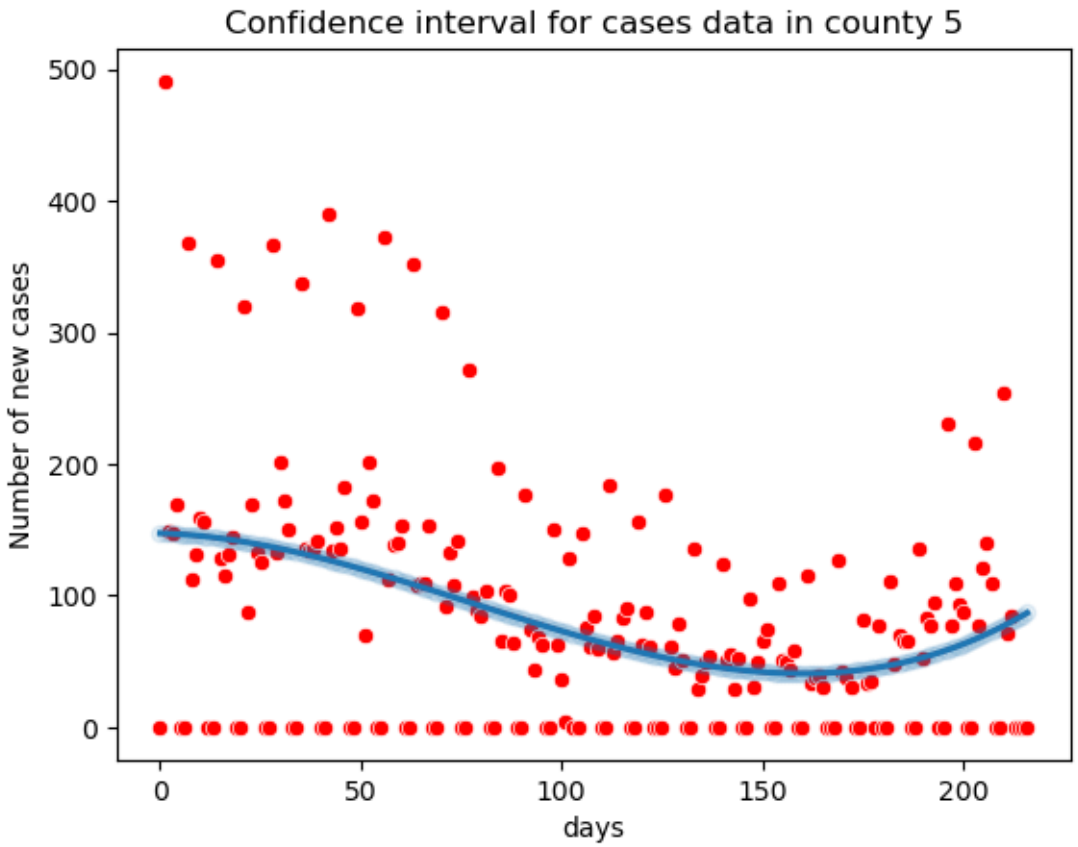


[90]:
```
# plotting confidence interval for cases data in county 5.

plt.title('Confidence interval for cases data in county 5')
sns.scatterplot(x, yCasesCounty5, color='red')
sns.lineplot(x, casesPredictionCounty5, color='red')
sns.regplot(xAxis['days'], casesPredictionCounty5, scatter_kws={'alpha':0.1},␣
 ↪order=3)
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
```

```
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[90]: `<AxesSubplot:title={'center':'Confidence interval for cases data in county 5'}, xlabel='days', ylabel='Number of new cases'>`

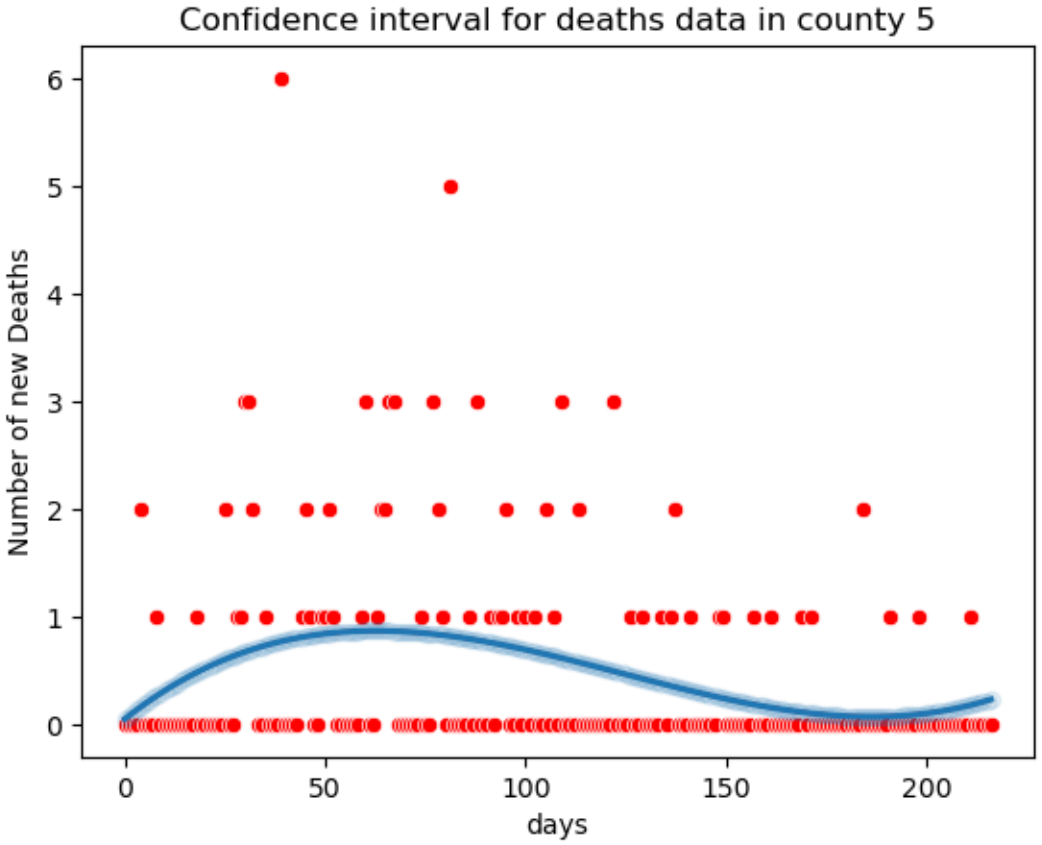


[91]:
```
# plotting confidence interval for deaths data in county 5.

plt.title('Confidence interval for deaths data in county 5')
sns.scatterplot(x, yDeathsCounty5, color='red')
sns.lineplot(x, deathsPredictionCounty5, color='red')
sns.regplot(xAxis['days'], deathsPredictionCounty5, scatter_kws={'alpha':0.1},␣
 ↪order=3)
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[91]:
```
<AxesSubplot:title={'center':'Confidence interval for deaths data in county 5'},
xlabel='days', ylabel='Number of new Deaths'>
```



8 Prediction Path

[92]:
```
x
```

[92]:
```
0      0
1      1
2      2
3      3
4      4
```

```
      …
212    212
213    213
214    214
215    215
216    216
Name: days, Length: 217, dtype: int64
```

[93]:
```
# Generating forecasting days for 1 week ahead.

predictDays = []
for i in range(len(x), len(x)+7):
    predictDays.append(i)
predictDays = pd.DataFrame({'Future days': predictDays})
predictDays
```

[93]:
```
   Future days
0          217
1          218
2          219
3          220
4          221
5          222
6          223
```

[94]:
```
# Prediction path for Virginia Cases and forecasting cases 1 week ahead.
linearModelCases = LinearRegression()
linearModelCases.fit(xAxis,yCases)
actualCases = linearModelCases.predict(xAxis)

predictCases = linearModelCases.predict(predictDays)
predictCases
```

```
C:\Users\venka\anaconda3\lib\site-packages\sklearn\base.py:493: FutureWarning:
The feature names should match those that were passed during fit. Starting
version 1.2, an error will be raised.
Feature names unseen at fit time:
- Future days
Feature names seen at fit time, yet now missing:
- days

  warnings.warn(message, FutureWarning)
```

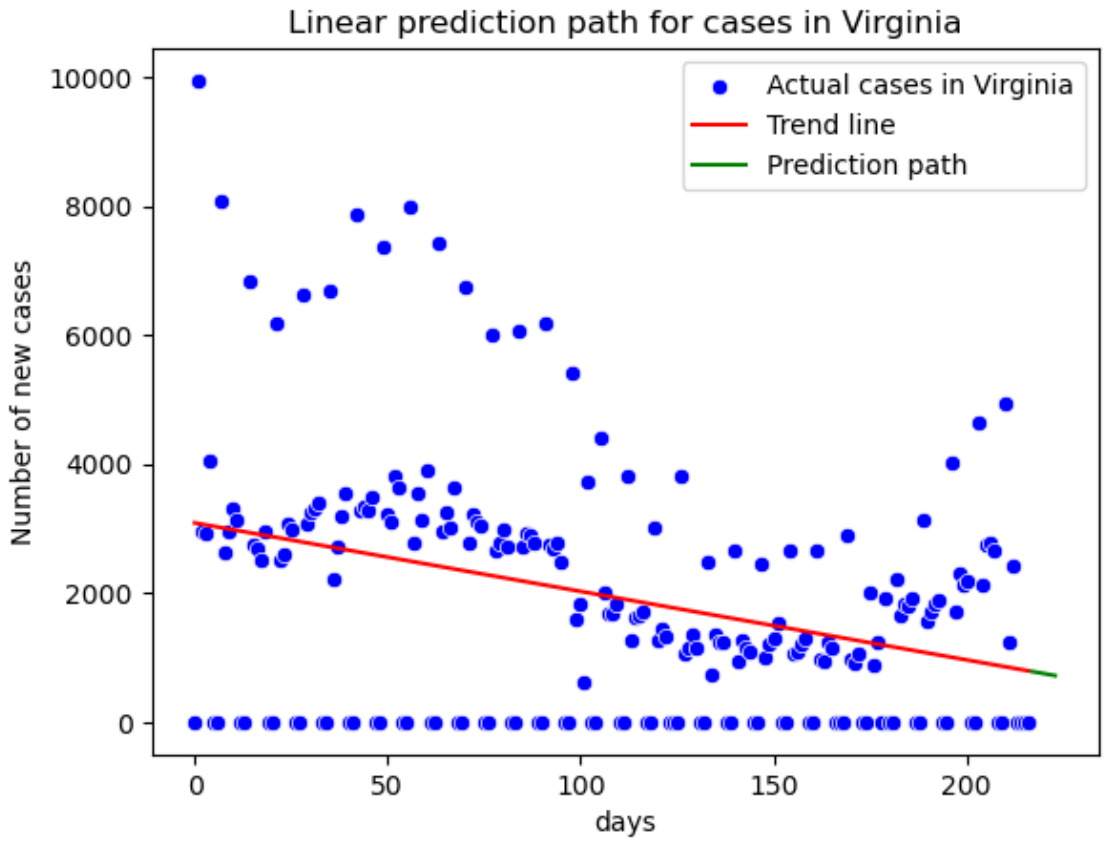
[94]:
```
array([785.60343062, 774.99111694, 764.37880325, 753.76648957,
       743.15417589, 732.5418622 , 721.92954852])
```

[95]:
```
# Prediction path for linear model cases.
```

```python
plt.title('Linear prediction path for cases in Virginia')
sns.scatterplot(x, yCases, color='blue', label='Actual cases in Virginia')
sns.lineplot(x, actualCases, color='red',  label='Trend line')
sns.lineplot(predictDays['Future days'], predictCases, color='green',␣
 ↪label='Prediction path')
plt.legend()
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[95]:
```
<matplotlib.legend.Legend at 0x1e9f3c6a5b0>
```



[96]:
```
# Prediction path for Virginia Deaths.
linearModelDeaths = LinearRegression()
linearModelDeaths.fit(xAxis,yDeaths)
actualDeaths = linearModelDeaths.predict(xAxis)

predictDeaths = linearModelDeaths.predict(predictDays)
predictDeaths
```

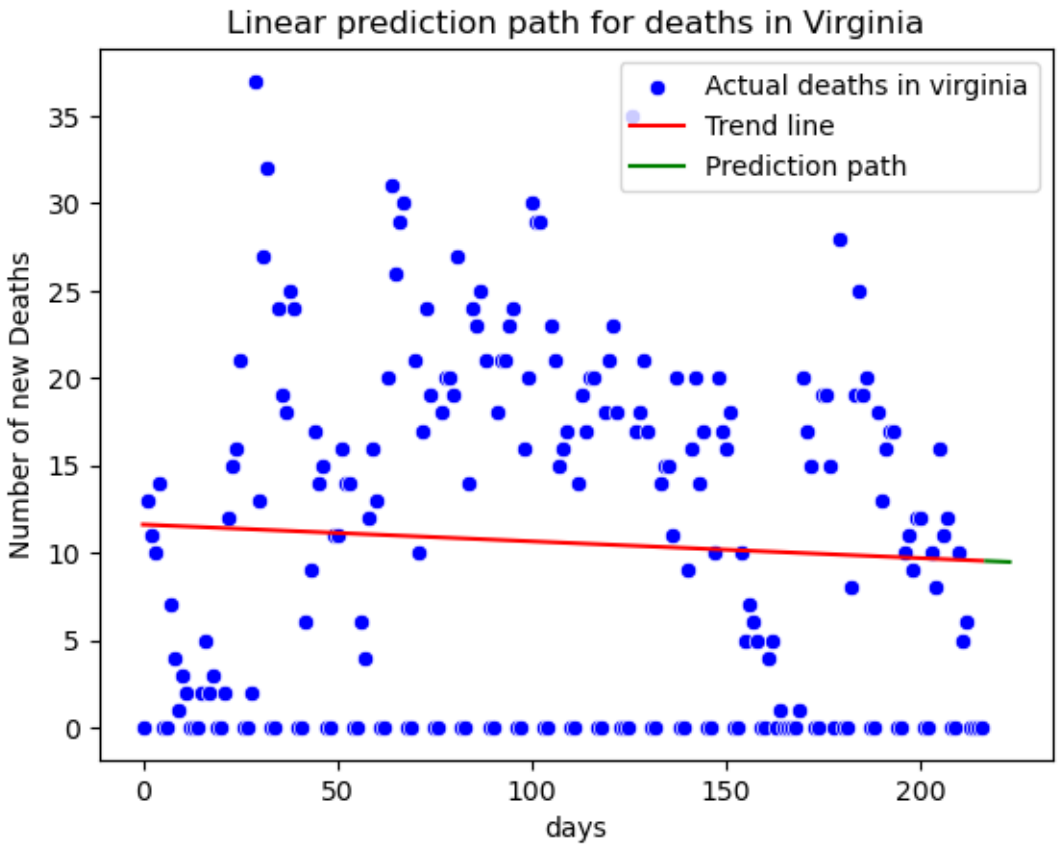
```
C:\Users\venka\anaconda3\lib\site-packages\sklearn\base.py:493: FutureWarning:
The feature names should match those that were passed during fit. Starting
version 1.2, an error will be raised.
Feature names unseen at fit time:
- Future days
Feature names seen at fit time, yet now missing:
- days

  warnings.warn(message, FutureWarning)
```

[96]:
```
array([9.52432156, 9.51471507, 9.50510858, 9.4955021 , 9.48589561,
       9.47628912, 9.46668264])
```

[97]:
```python
# Plotting prediction path for deaths in Virginia.

plt.title('Linear prediction path for deaths in Virginia')
sns.scatterplot(x, yDeaths, color='blue', label='Actual deaths in virginia')
sns.lineplot(x, actualDeaths, color='red', label='Trend line')
sns.lineplot(predictDays['Future days'], predictDeaths, color='green',␣
    ↪label='Prediction path')
plt.legend()
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[97]:
```
<matplotlib.legend.Legend at 0x1e9f3cb72b0>
```



[98]:
```
# prediction path for non-linear cases.

nonLinear = PolynomialFeatures(degree=3)
days = np.array(days)
days = days.reshape(-1,1)
xAxisPoly = nonLinear.fit_transform(days)
predictPoly = nonLinear.fit_transform(predictDays)
linearModelCases.fit(xAxisPoly, yCases)
```

[98]:
```
LinearRegression()
```

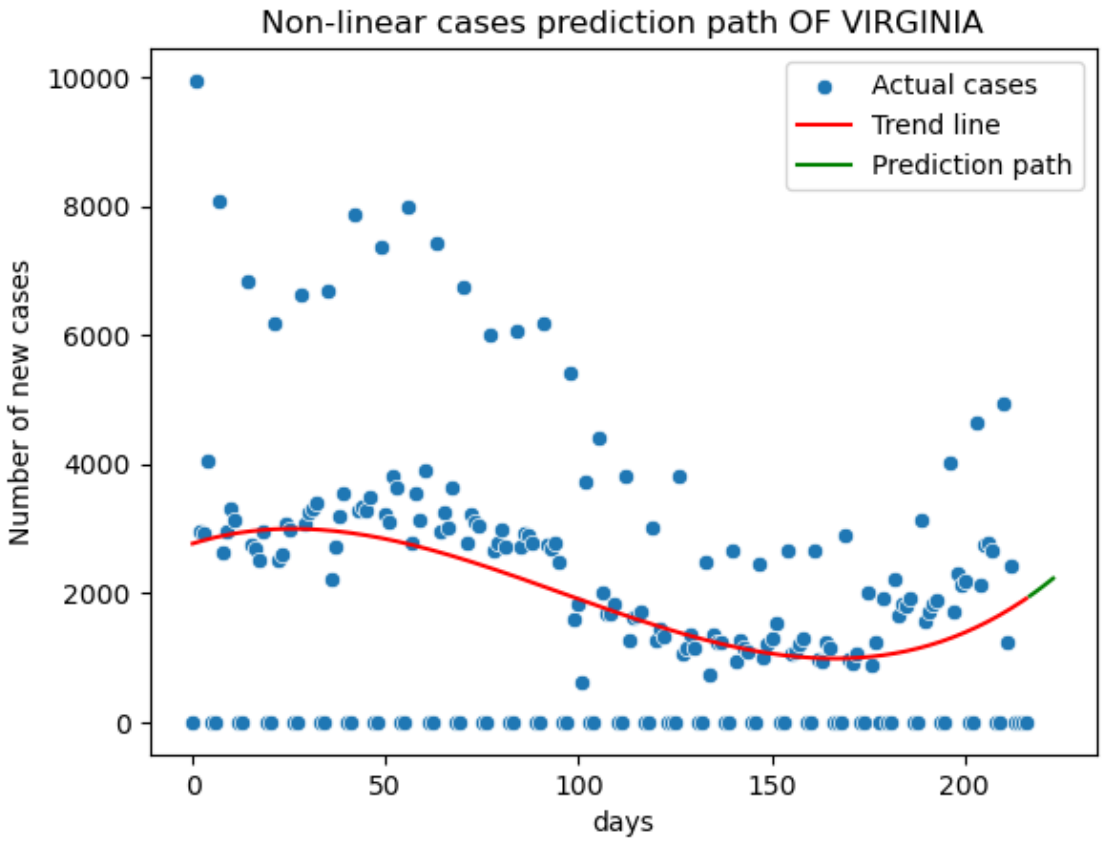
[99]:
```
# Non-linear cases forecasting.

actualPolyCases = linearModelCases.predict(xAxisPoly)

predictPolyCases = linearModelCases.predict(predictPoly)
predictPolyCases
```

[99]:
```
array([1957.67282124, 2000.26726639, 2043.92153495, 2088.64432264,
       2134.44432517, 2181.33023823, 2229.31075756])
```

[100]:
```python
# Plotting non-linear cases forecast.

plt.title('Non-linear cases prediction path OF VIRGINIA')
sns.scatterplot(x, yCases, label='Actual cases')
sns.lineplot(x, actualPolyCases,  color='red', label='Trend line')
sns.lineplot(predictDays['Future days'], predictPolyCases,  color='green',␣
 ↪label='Prediction path')
plt.legend()
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[100]: <matplotlib.legend.Legend at 0x1e9f3ce91c0>



[101]:
```
# Non-linear deaths forecasting.

linearModelDeaths.fit(xAxisPoly, yDeaths)
actualPolyDeaths = linearModelDeaths.predict(xAxisPoly)

predictPolyDeaths = linearModelDeaths.predict(predictPoly)
predictPolyDeaths
```

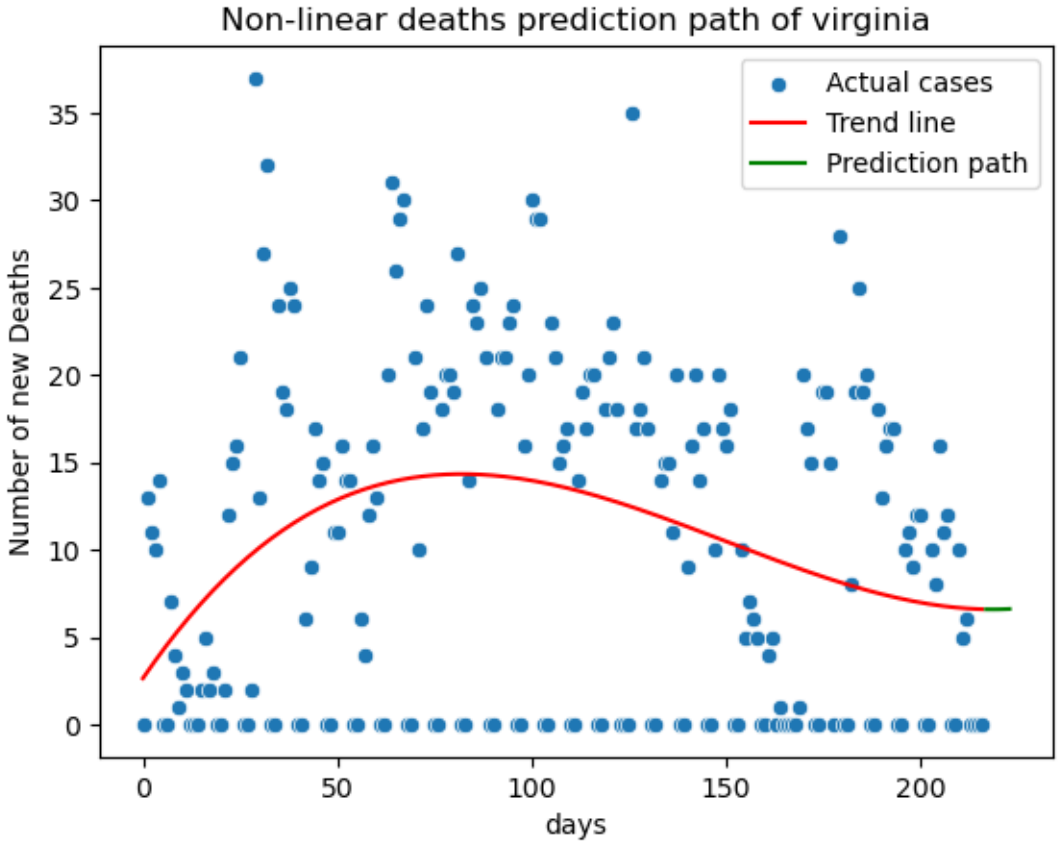
[101]:
```
array([6.58294897, 6.58009088, 6.57969725, 6.5818045 , 6.58644903,
       6.59366725, 6.60349556])
```

[102]:
```
# Plotting non-linear DEATHS forecast.

plt.title('Non-linear deaths prediction path of virginia')
sns.scatterplot(x, yDeaths, label='Actual cases')
sns.lineplot(x, actualPolyDeaths,  color='red', label='Trend line')
sns.lineplot(predictDays['Future days'], predictPolyDeaths,  color='green',␣
 ↪label='Prediction path')
plt.legend()
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
```

```
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[102]: `<matplotlib.legend.Legend at 0x1e9f3d5e460>`



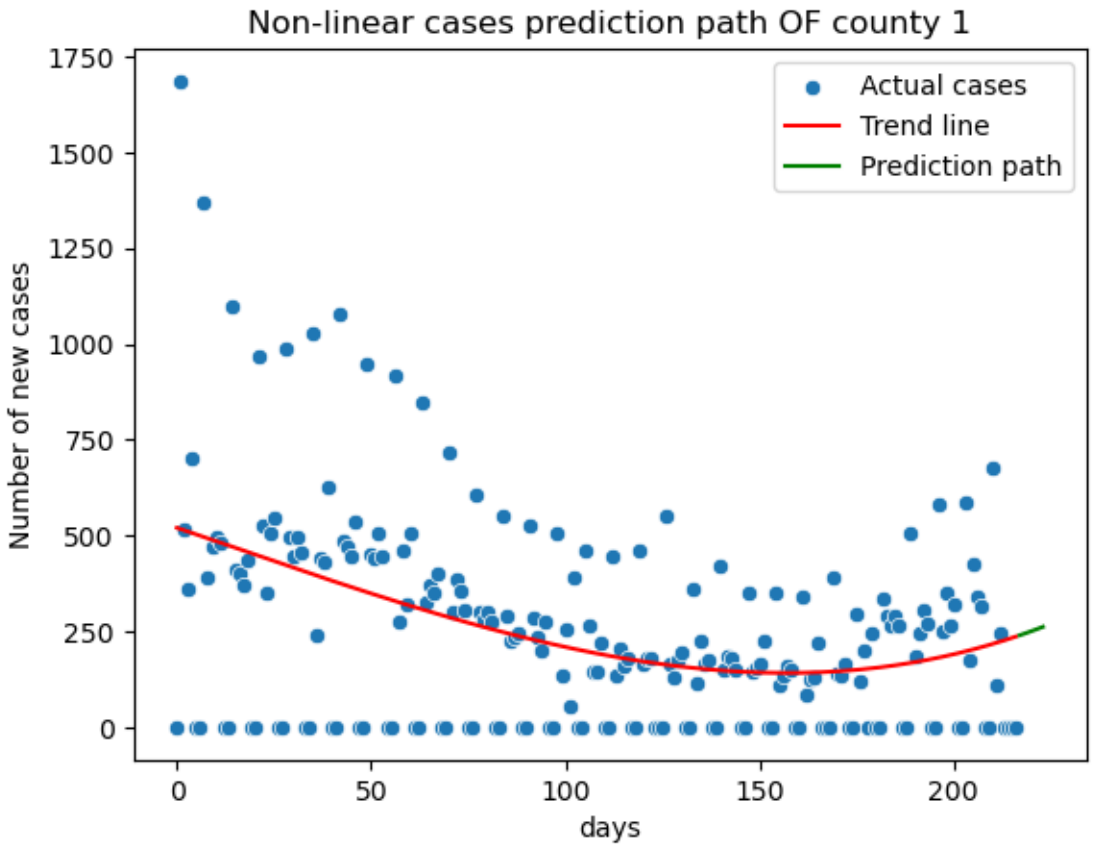
[103]:
```python
# Non-linear cases forecasting for county 1.

linearModelCases.fit(xAxisPoly, yCasesCounty1)
actualPolyCases = linearModelCases.predict(xAxisPoly)

predictPolyCases = linearModelCases.predict(predictPoly)
predictPolyCases
```

[103]:
```
array([239.86908476, 243.3751597 , 246.94930928, 250.59186306,
       254.30315058, 258.08350137, 261.933245  ])
```

[104]:
```python
# Plotting non-linear cases forecast for county 1.

plt.title('Non-linear cases prediction path OF county 1')
sns.scatterplot(x, yCasesCounty1, label='Actual cases')
sns.lineplot(x, actualPolyCases,  color='red', label='Trend line')
sns.lineplot(predictDays['Future days'], predictPolyCases,  color='green',␣
 ↪label='Prediction path')
plt.legend()
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[104]:
```
<matplotlib.legend.Legend at 0x1e9f3dd9970>
```



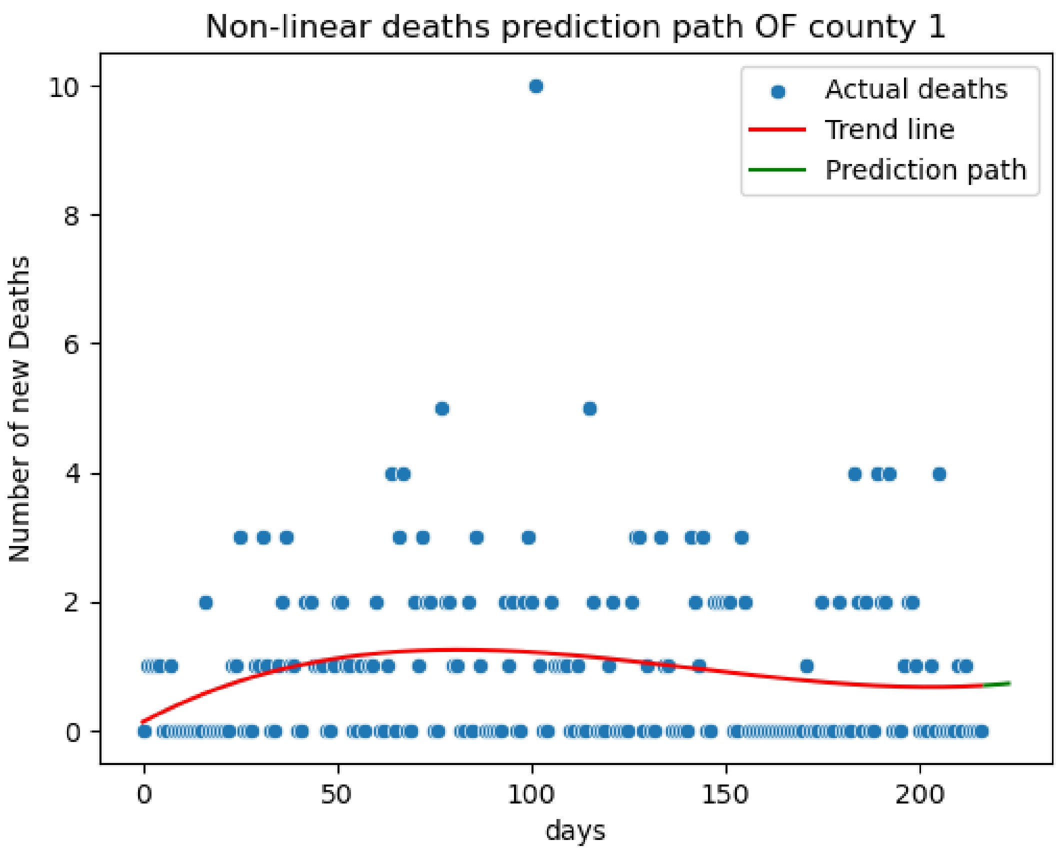
[105]:
```
# Non-linear deaths forecasting for county 1.

linearModelDeaths.fit(xAxisPoly, yDeathsCounty1)
actualPolyDeaths = linearModelDeaths.predict(xAxisPoly)

predictPolyDeaths = linearModelDeaths.predict(predictPoly)
predictPolyDeaths
```

[105]:
```
array([0.6992547 , 0.70310661, 0.70725111, 0.71169206, 0.71643331,
       0.72147871, 0.72683211])
```

[106]:
```
# Plotting non-linear deaths forecast for county 1.

plt.title('Non-linear deaths prediction path OF county 1')
sns.scatterplot(x, yDeathsCounty1, label='Actual deaths')
sns.lineplot(x, actualPolyDeaths,  color='red', label='Trend line')
sns.lineplot(predictDays['Future days'], predictPolyDeaths,  color='green',␣
 ↪label='Prediction path')
plt.legend()
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[106]: <matplotlib.legend.Legend at 0x1e9f3fa5b50>



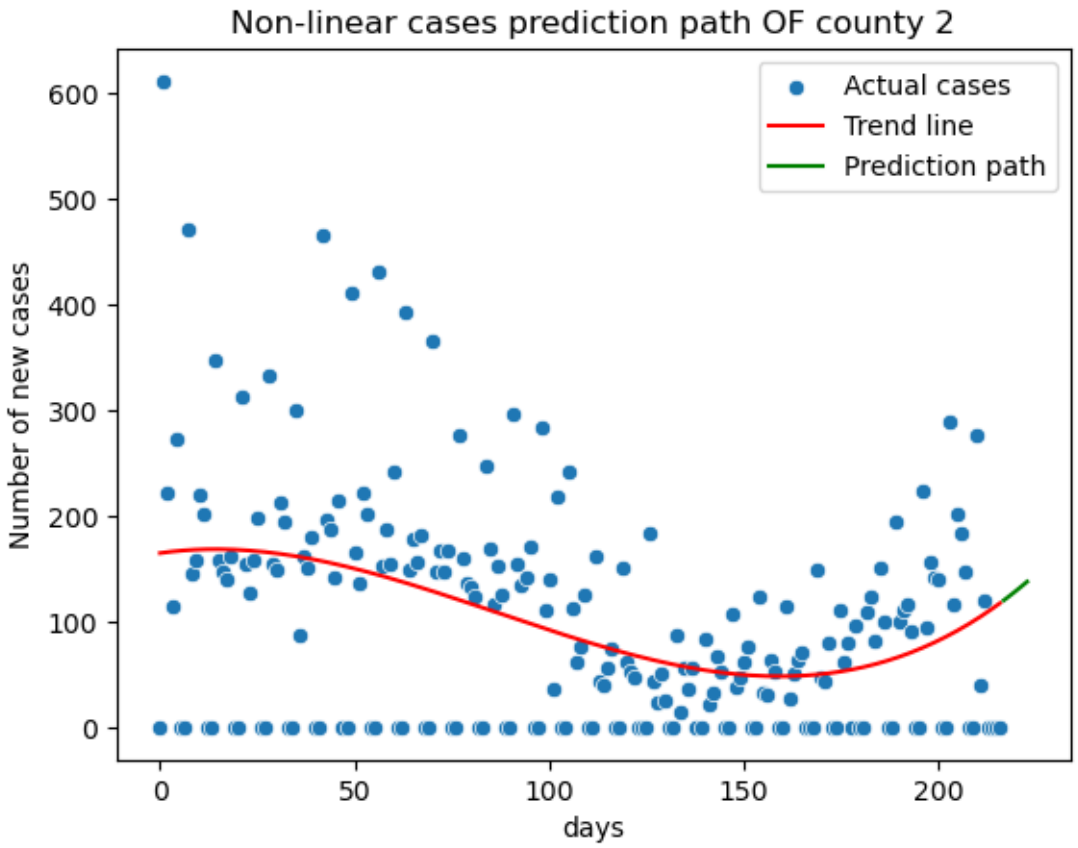
[107]:
```python
# Non-linear cases forecasting for county 2.

linearModelCases.fit(xAxisPoly, yCasesCounty2)
actualPolyCases = linearModelCases.predict(xAxisPoly)

predictPolyCases = linearModelCases.predict(predictPoly)
predictPolyCases
```

[107]:
```
array([120.05966367, 122.84587841, 125.69396622, 128.60439867,
       131.5776473 , 134.61418368, 137.71447935])
```

[108]:
```python
# Plotting non-linear cases forecast for county 2.

plt.title('Non-linear cases prediction path OF county 2')
sns.scatterplot(x, yCasesCounty2, label='Actual cases')
sns.lineplot(x, actualPolyCases,  color='red', label='Trend line')
sns.lineplot(predictDays['Future days'], predictPolyCases,  color='green',␣
 ↪label='Prediction path')
plt.legend()
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[108]:
```
<matplotlib.legend.Legend at 0x1e9f401be80>
```



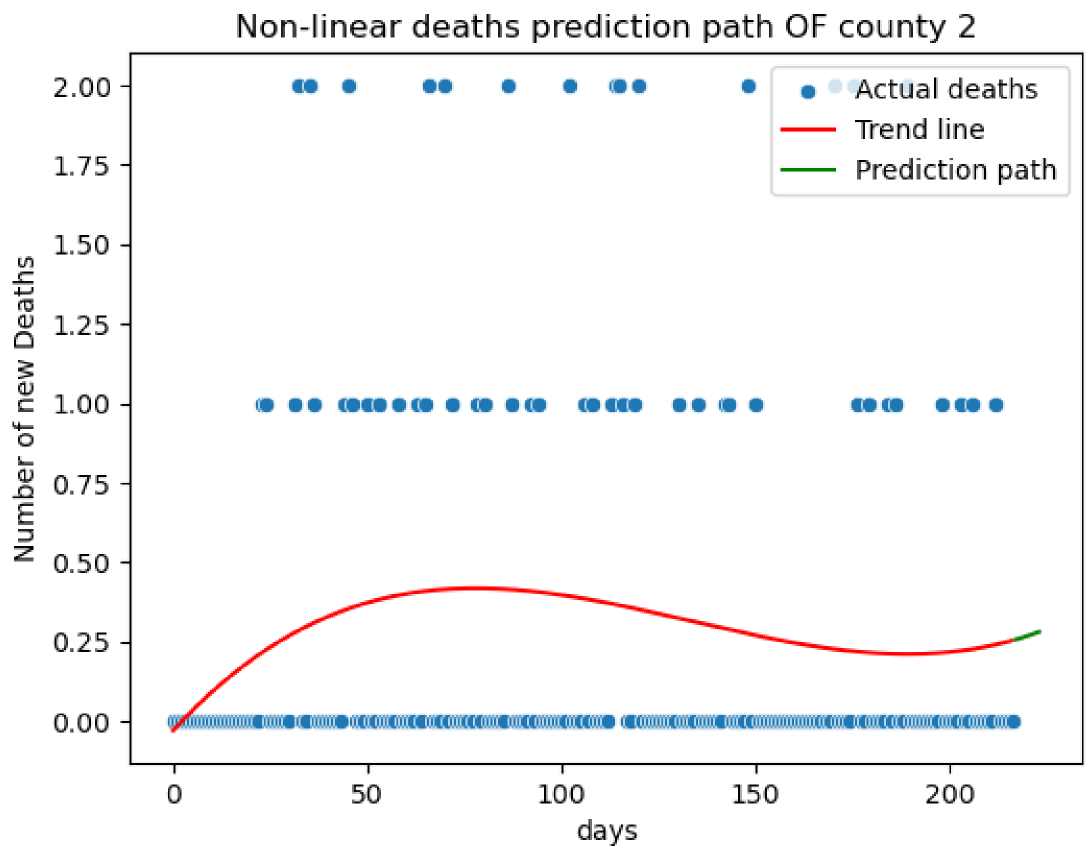
[109]:
```
# Non-linear deaths forecasting for county 2.

linearModelDeaths.fit(xAxisPoly, yDeathsCounty2)
actualPolyDeaths = linearModelDeaths.predict(xAxisPoly)

predictPolyDeaths = linearModelDeaths.predict(predictPoly)
predictPolyDeaths
```

[109]:
```
array([0.25673102, 0.26032427, 0.26407046, 0.26797141, 0.27202892,
       0.2762448 , 0.28062086])
```

[110]:
```
# Plotting non-linear deaths forecast for county 2.

plt.title('Non-linear deaths prediction path OF county 2')
sns.scatterplot(x, yDeathsCounty2, label='Actual deaths')
sns.lineplot(x, actualPolyDeaths,  color='red', label='Trend line')
sns.lineplot(predictDays['Future days'], predictPolyDeaths,  color='green',␣
 ↪label='Prediction path')
plt.legend()
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[110]: <matplotlib.legend.Legend at 0x1e9f400d910>



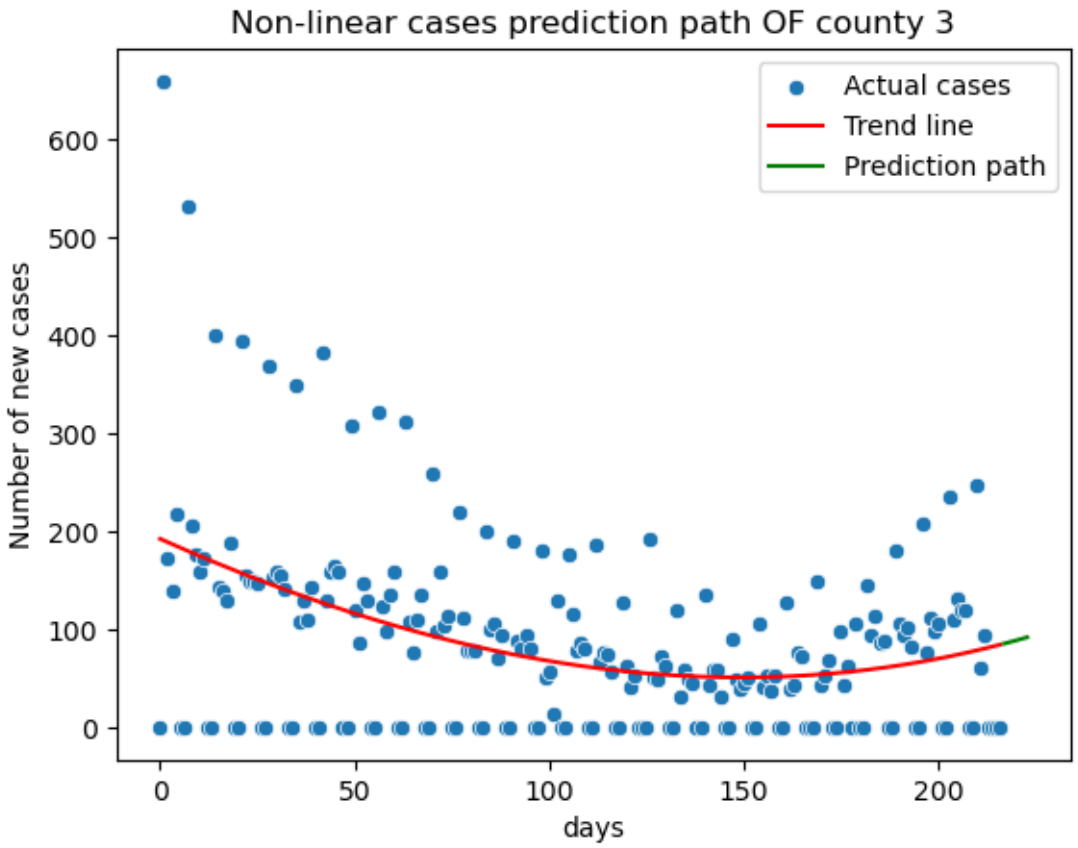
[111]:
```
# Non-linear cases forecasting for county 3.

linearModelCases.fit(xAxisPoly, yCasesCounty3)
actualPolyCases = linearModelCases.predict(xAxisPoly)

predictPolyCases = linearModelCases.predict(predictPoly)
predictPolyCases
```

[111]:
```
array([85.35910519, 86.40911859, 87.47566752, 88.55878448, 89.65850195,
       90.77485244, 91.90786846])
```

[112]:
```
# Plotting non-linear cases forecast for county 3.

plt.title('Non-linear cases prediction path OF county 3')
sns.scatterplot(x, yCasesCounty3, label='Actual cases')
sns.lineplot(x, actualPolyCases,  color='red', label='Trend line')
sns.lineplot(predictDays['Future days'], predictPolyCases,  color='green',␣
 ↪label='Prediction path')
plt.legend()
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[112]:
```
<matplotlib.legend.Legend at 0x1e9f41c28e0>
```



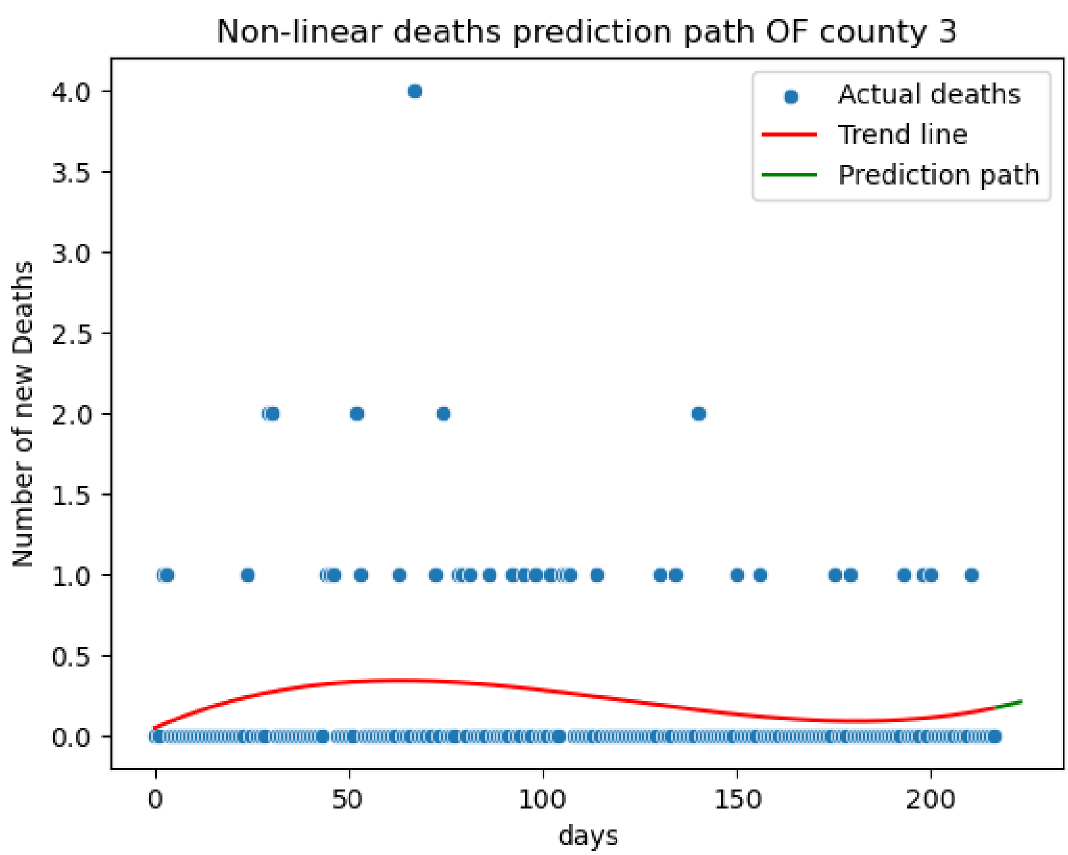
[113]:
```
# Non-linear deaths forecasting for county 3.

linearModelDeaths.fit(xAxisPoly, yDeathsCounty3)
actualPolyDeaths = linearModelDeaths.predict(xAxisPoly)

predictPolyDeaths = linearModelDeaths.predict(predictPoly)
predictPolyDeaths
```

[113]:
```
array([0.17418323, 0.17942137, 0.18483717, 0.19043247, 0.19620913,
       0.20216899, 0.20831391])
```

[114]:
```
# Plotting non-linear deaths forecast for county 3.

plt.title('Non-linear deaths prediction path OF county 3')
sns.scatterplot(x, yDeathsCounty3, label='Actual deaths')
sns.lineplot(x, actualPolyDeaths,  color='red', label='Trend line')
sns.lineplot(predictDays['Future days'], predictPolyDeaths,  color='green',␣
 ↪label='Prediction path')
plt.legend()
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[114]: <matplotlib.legend.Legend at 0x1e9f4441280>



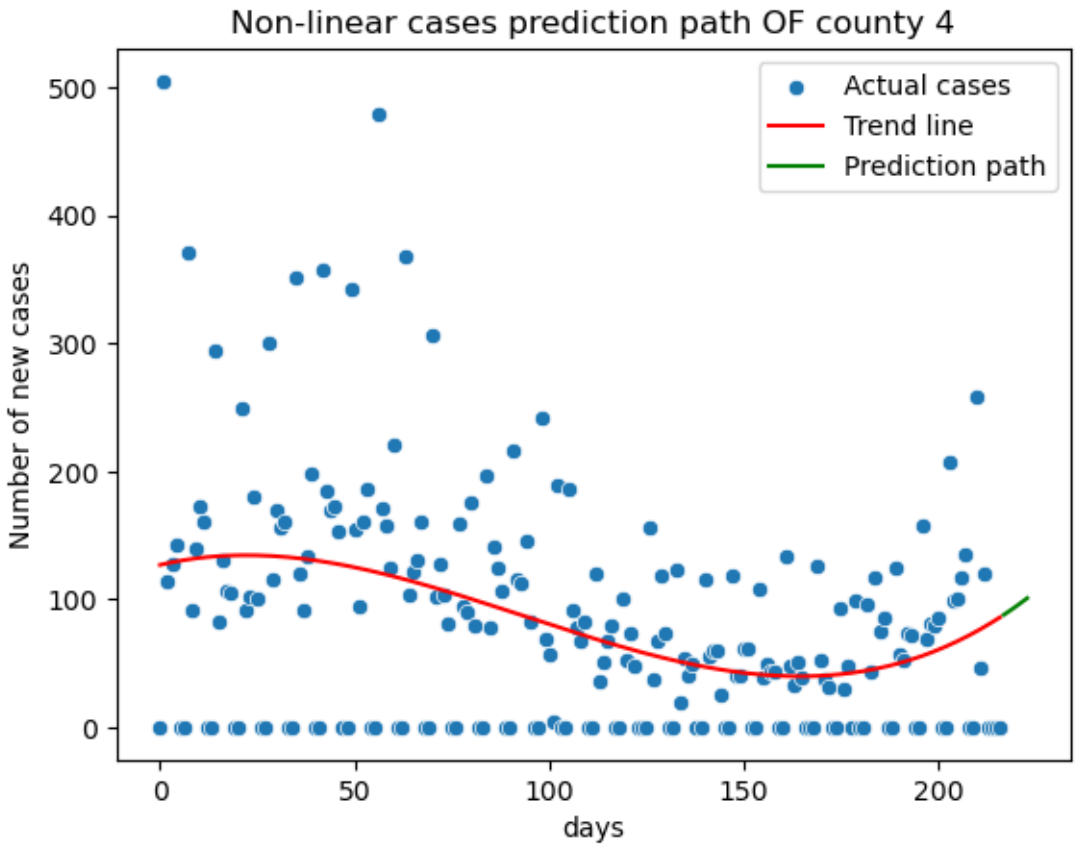
[115]:
```python
# Non-linear cases forecasting for county 4.

linearModelCases.fit(xAxisPoly, yCasesCounty4)
actualPolyCases = linearModelCases.predict(xAxisPoly)

predictPolyCases = linearModelCases.predict(predictPoly)
predictPolyCases
```

[115]:
```
array([ 87.91558164,  89.95260584,  92.0387005 ,  94.17425926,
        96.35967571,  98.59534348, 100.88165619])
```

[116]:
```python
# Plotting non-linear cases forecast for county 4.

plt.title('Non-linear cases prediction path OF county 4')
sns.scatterplot(x, yCasesCounty4, label='Actual cases')
sns.lineplot(x, actualPolyCases,  color='red', label='Trend line')
sns.lineplot(predictDays['Future days'], predictPolyCases,  color='green',␣
 ↪label='Prediction path')
plt.legend()
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[116]:
```
<matplotlib.legend.Legend at 0x1e9f43f05e0>
```



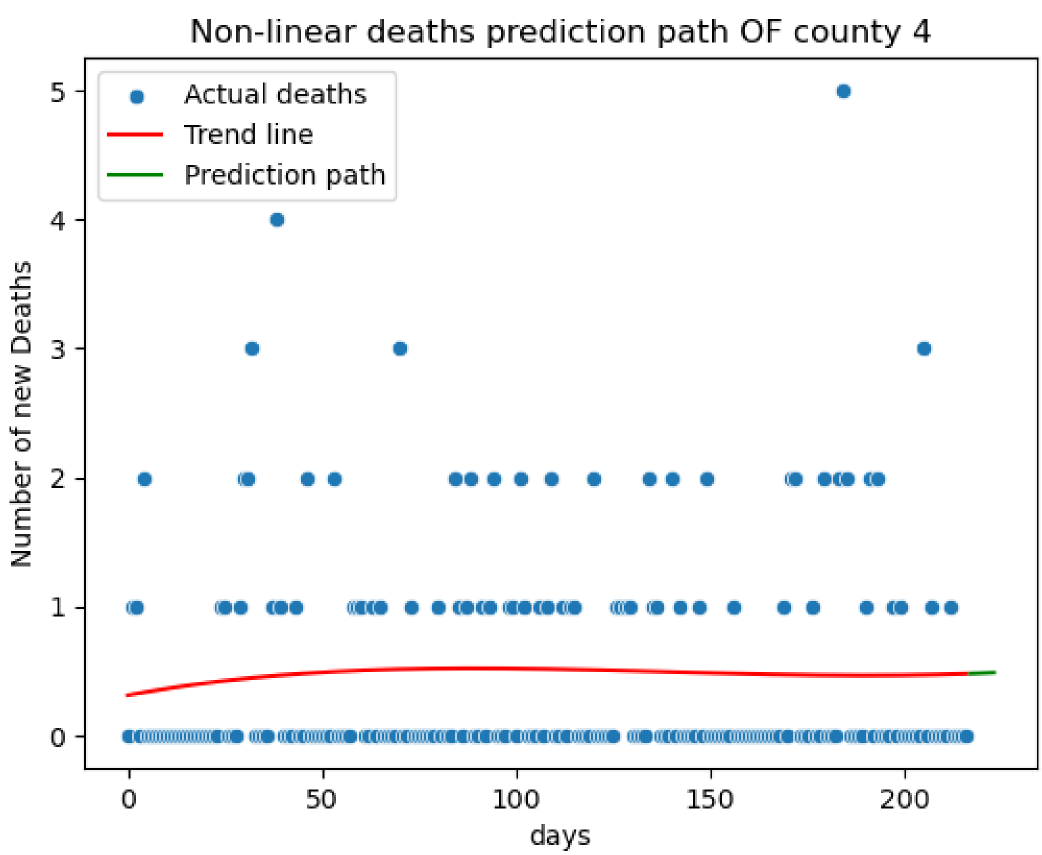
[117]:
```
# Non-linear deaths forecasting for county 4.

linearModelDeaths.fit(xAxisPoly, yDeathsCounty4)
actualPolyDeaths = linearModelDeaths.predict(xAxisPoly)

predictPolyDeaths = linearModelDeaths.predict(predictPoly)
predictPolyDeaths
```

[117]:
```
array([0.47903943, 0.48021475, 0.48144187, 0.48272144, 0.48405414,
       0.48544061, 0.48688152])
```

[118]:
```
# Plotting non-linear deaths forecast for county 4.

plt.title('Non-linear deaths prediction path OF county 4')
sns.scatterplot(x, yDeathsCounty4, label='Actual deaths')
sns.lineplot(x, actualPolyDeaths,  color='red', label='Trend line')
sns.lineplot(predictDays['Future days'], predictPolyDeaths,  color='green',␣
 ↪label='Prediction path')
plt.legend()
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[118]: <matplotlib.legend.Legend at 0x1e9f45dc700>



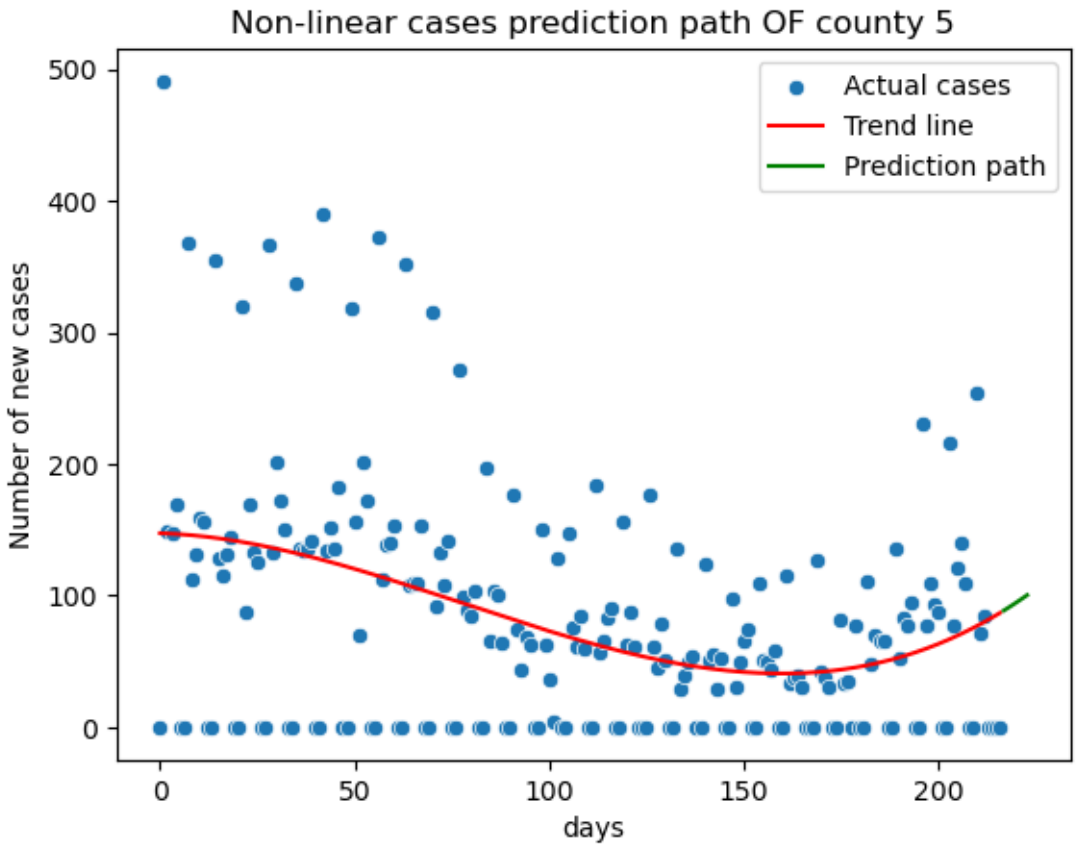
[119]:
```python
# Non-linear cases forecasting for county 5.

linearModelCases.fit(xAxisPoly, yCasesCounty5)
actualPolyCases = linearModelCases.predict(xAxisPoly)

predictPolyCases = linearModelCases.predict(predictPoly)
predictPolyCases
```

[119]:
```
array([ 88.70373095,  90.55699125,  92.45091655,  94.38579694,
        96.36192255,  98.37958346, 100.43906979])
```

[120]:
```python
# Plotting non-linear cases forecast for county 5.

plt.title('Non-linear cases prediction path OF county 5')
sns.scatterplot(x, yCasesCounty5, label='Actual cases')
sns.lineplot(x, actualPolyCases,  color='red', label='Trend line')
sns.lineplot(predictDays['Future days'], predictPolyCases,  color='green',␣
 ↪label='Prediction path')
plt.legend()
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[120]:
```
<matplotlib.legend.Legend at 0x1e9f4614e20>
```



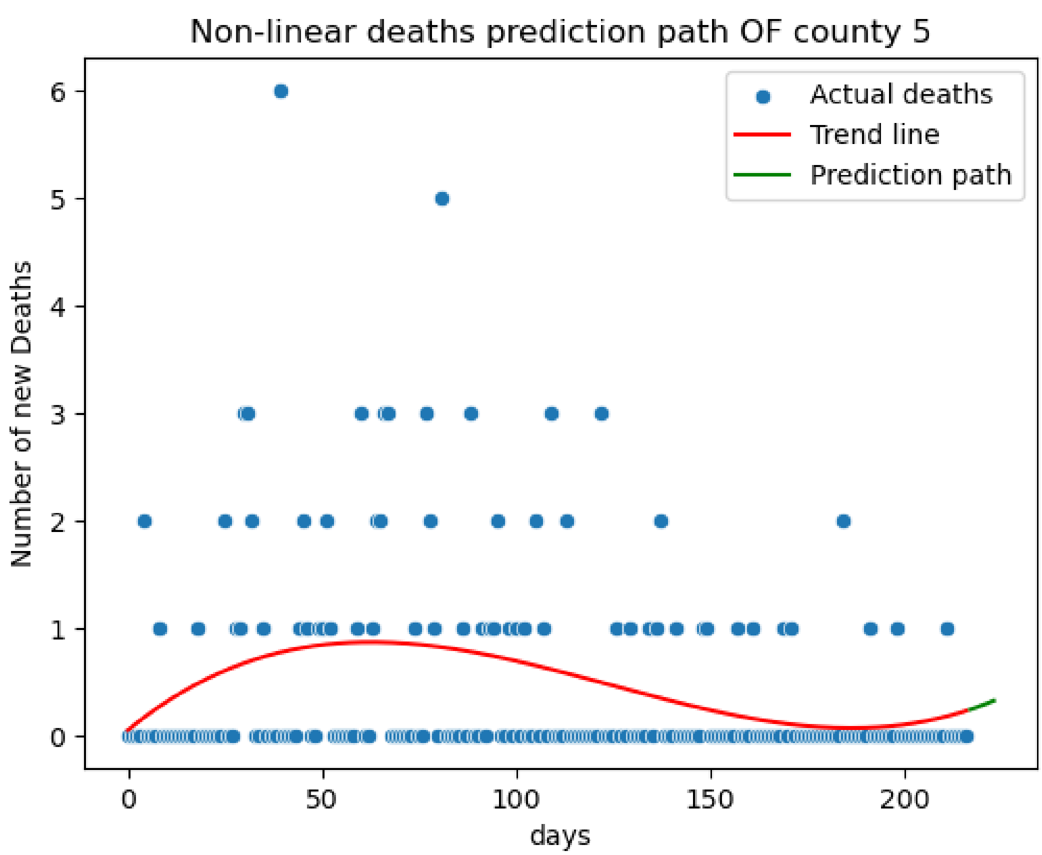
[121]:
```
# Non-linear deaths forecasting for county 5.

linearModelDeaths.fit(xAxisPoly, yDeathsCounty5)
actualPolyDeaths = linearModelDeaths.predict(xAxisPoly)

predictPolyDeaths = linearModelDeaths.predict(predictPoly)
predictPolyDeaths
```

[121]:
```
array([0.24194867, 0.25419066, 0.26690606, 0.28009993, 0.29377732,
       0.30794329, 0.32260291])
```

[122]:
```
# Plotting non-linear deaths forecast for county 5.

plt.title('Non-linear deaths prediction path OF county 5')
sns.scatterplot(x, yDeathsCounty5, label='Actual deaths')
sns.lineplot(x, actualPolyDeaths,  color='red', label='Trend line')
sns.lineplot(predictDays['Future days'], predictPolyDeaths,  color='green',␣
 ↪label='Prediction path')
plt.legend()
```

```
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\venka\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[122]: <matplotlib.legend.Legend at 0x1e9f1a3eca0>



9 References

1) https://www.tutorialspoint.com/how-to-visualize-95-confidence-interval-in-matplotlib

2) https://stackoverflow.com/questions/41328922/python-pandas-simple-example-of-calculating-rmse-from-data-frame

3) https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

4)https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_

5) https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html

6)https://scikitlearn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html#sklearn.prep

[ ]: