Programming Assignment I : Gaussian Process Regression

Amatya Sharma
17CS30042

Dependencies and Variables :

- We are predicting for Xtest, if user wants to predict for custom values, modify Xtest and Ytest correspndingly in cell 3.
- Xtrain := Trianing Data with only one feature i.e. Day Number
- Ytrain := Training Values Corresponding to Xtrain
- Ytrain_India := Ytrain for India
- Ytrain_World := Ytrain for World
- Xtest := Test Data with only one feature i.e. Day Number
- Ytest := Actual Test Values Corresponding to Xtest will be used for checking accuracy of the models
- Ytest_India := Ytest for India
- Ytest_World := Ytest for World
- K = Kernel(Xtrain, Xtrain)
- K_s = kernel(Xtrain, Xtest)
- K_ss = kernel(Xtest, Xtest)
- sigma_f := 1st hyperparam for the squared exponential kernel function
- l := 1st hyperparam for the squared exponential kernel function
- sigma_y := given gaussian noise error variance
- mu_s := PPD predicted mean vector corresp to PPD Gaussian Distribution
- sigma_s := PPD predicted mean vector corresp to PPD Gaussian Distribution

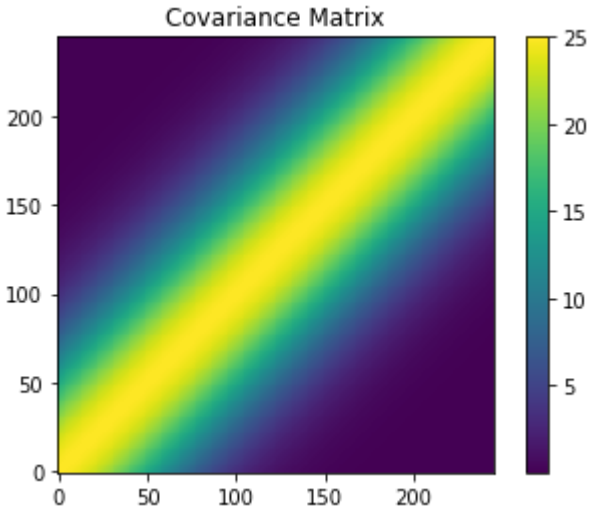
In [21]:
```
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import cholesky, det, lstsq
import pandas
# from scipy.optimize import minimize/
```

In [22]:
```
# read data
df = pandas.read_csv('new_casesWorld-India.csv')
data = df.values

# partition into training and test sets
# partiton into data from world and india
Xtrain = data[0:245,0]
Ytrain_World = data[0:245,2]
Ytrain_India = data[0:245,3]

Xtest = data[246:, 0]
Ytest_World = data[246:, 2]
Ytest_India = data[246:, 3]

# convert all arrays to numpy arrays to enable them to perform numpy array/mat
rix operations
Xtrain = np.array(Xtrain)
Ytrain_World = np.array(Ytrain_World)
Ytrain_India = np.array(Ytrain_India)
Xtest = np.array(Xtest)
Ytest_World = np.array(Ytest_World)
Ytest_India = np.array(Ytest_India)
```

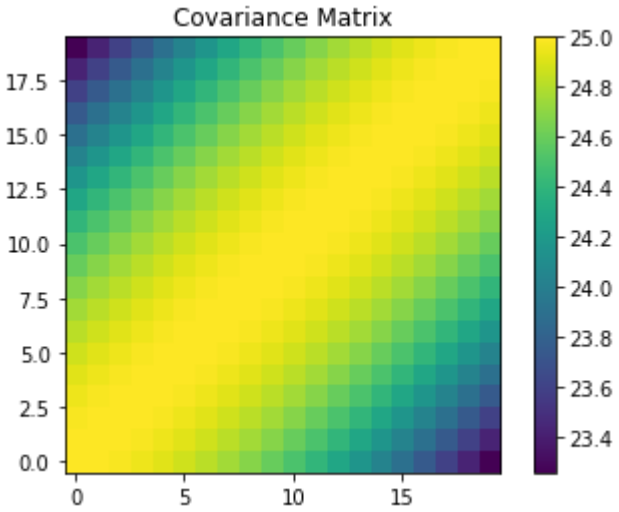
In [23]:
```
# kernel function for guasiian processes
# kernel function is only for vectors not for matrices as our inputs are 1-D
# we use squared exponential kernel
# with hyperparameters sigma_f and l


# funtion to caluculate kernel of 2 scalars

def rbf_kernel_util(sigma_f, x1, x2):
    return np.exp(-(x1-x2)**2/(2* sigma_f**2))

def sqexp_kernel_util(x1, x2, l, sigma_f):
    return sigma_f*sigma_f * np.exp(-0.5/(l*l) * (x1-x2)*(x1-x2))

# funtion to calculate SE Kernel of two input vectors
def kernel(X1, X2, l, sigma_f):
    l1 = len(X1)
    l2 = len(X2)
    K = np.zeros((l1, l2))

    #iterate over all indices of the covariance matrix to fill it with k(i,j)
    for i in range(l1):
        for j in range(l2):
            K[i][j] = sqexp_kernel_util(X1[i], X2[j], l, sigma_f)
    return K

def kernel_visualize(X_visualize, X_visualize2, l, sigma_f):
    K_visualize = kernel(X_visualize, X_visualize2, l, sigma_f)

    figre, axs = plt.subplots()
    plt.imshow(K_visualize, interpolation='none')
    plt.title("Covariance Matrix")
    axs.set_ylim(axs.get_ylim()[::-1])
    plt.colorbar()
    plt.show()
#to-do write some features of hyper parameters
```

In [24]:
```
# visualizing the kernel matrix for first 20 values
# the concentration can be seen along diagonal values due to the properties of
Squared Exponential Kernel
sigma_f = 5
l = 50

print("Histogram of K(Xtrain, Xtrain):")
kernel_visualize(Xtrain, Xtrain, l, sigma_f)

print("Histogram of K(Xtest, Xtest):")
kernel_visualize(Xtest, Xtest, l, sigma_f)

print("Histogram of K(Xtrain, Xtest):")
kernel_visualize(Xtrain, Xtest, l, sigma_f)
```

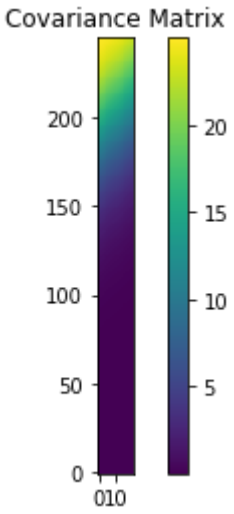
Histogram of K(Xtrain, Xtrain):



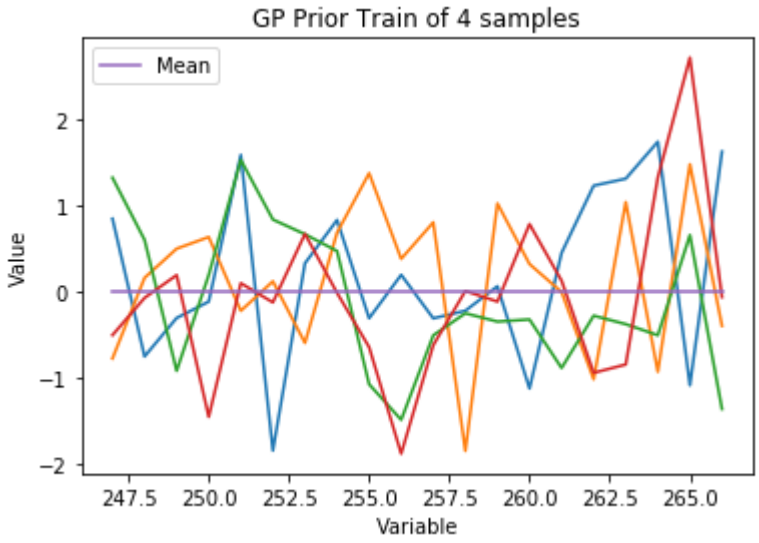
Histogram of K(Xtest, Xtest):



Histogram of K(Xtrain, Xtest):


Covariance Matrix
200
150
100
50
0
010
20
15
10
5

In [25]:
```
#visualize GP of  Finite number of points

Xplot = Xtest
numplot = len(Xplot)
Kplot = kernel(Xplot, Xplot, .1, 1)

rand_draw_prior = np.dot(Kplot, np.random.normal(size=(numplot, 4)))

figre, axs = plt.subplots()
variances = np.diag(Kplot) # diagonal elements are individual vairance
std_dev = np.sqrt(variances)

# plt.fill_between(Xplot.reshape(-1), 1.96 * std_dev,  - 1.96 * std_dev, alpha
=0.1)

plt.plot(Xplot, rand_draw_prior)
plt.plot(Xplot, np.zeros(len(Xplot)), label='Mean')
plt.title("GP Prior Train of 4 samples")
plt.xlabel("Variable")
plt.ylabel("Value")
plt.legend()
plt.show()
```

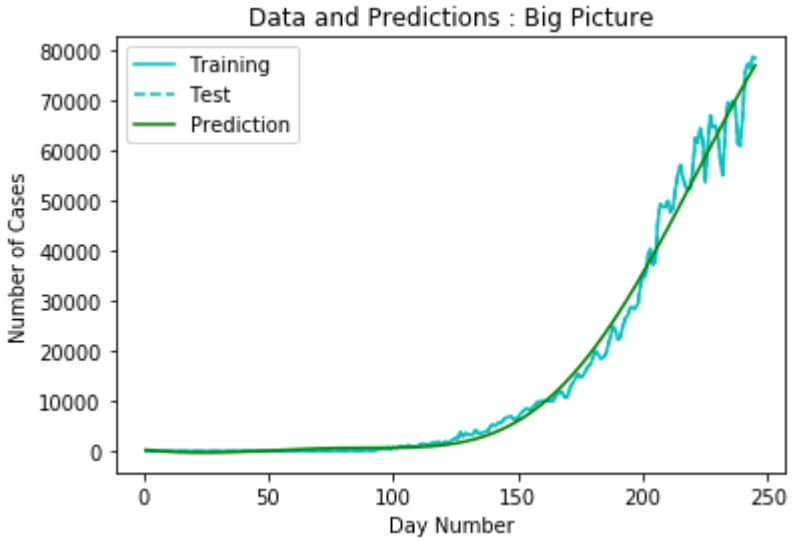


In [26]:
```
def Cinv(K, sigma_y):
    K_inv = np.linalg.inv(K + (sigma_y*sigma_y)*np.eye(len(Xtrain)))
    return K_inv

def posterior_predictive(Xtrain, Ytrain, l, sigma_y, sigma_f, X):
    cov_s = 0.0
    K_ss = kernel(X, X, l, sigma_f)
    K_s = kernel(Xtrain, X, l, sigma_f)

    mu_s = 0.0
    K = kernel(Xtrain, Xtrain, l, sigma_f)
    K_inv = Cinv(K, sigma_y)

    sigma_s = K_ss - K_s.T.dot(K_inv).dot(K_s)
    mu_s = K_s.T.dot(K_inv).dot(Ytrain)

    return mu_s, sigma_s
```

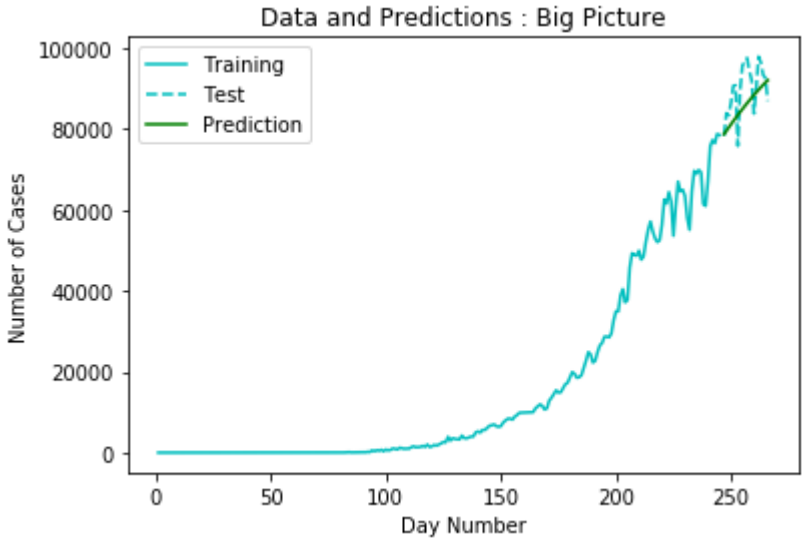
In [27]:
```
def predict_and_plot(Xtrain, Ytrain, l, sigma_f, sigma_y, Xtest, Ytest):
    mu, sigma = posterior_predictive(Xtrain, Ytrain, l, sigma_y, sigma_f, Xtes
t)

    figr, axs = plt.subplots()

    plt.plot(Xtrain, Ytrain, 'c')
    plt.plot(Xtest, Ytest, 'c--')
    plt.plot(Xtest, mu, 'g')
    plt.xlabel("Day Number")
    plt.ylabel("Number of Cases")
    plt.legend(["Training", "Test","Prediction"])
    plt.title("Data and Predictions : Big Picture")

    plt.show()

    if Xtrain.all() == Xtest.all():
        return mu, sigma

    plt.plot(Xtest, Ytest)
    plt.plot(Xtest, mu)
#     pred = np.array(mu)
#     pred = pred.T
#     sgma = np.array(sigma)
#     plt.errorbar(Xtest, pred, yerr = sgma, capsize = 0)
    plt.xlabel("Day Number")
    plt.ylabel("Number of Cases")
    plt.legend(["Test", "Prediction"])
    plt.title("Zoomed in View on Test set")
    plt.show()

    plt.imshow(sigma, interpolation='none')
    plt.title("Covariance Matrix PPD")
    axs.set_ylim(axs.get_ylim()[::-1])
    plt.colorbar()
    plt.show()

    return mu, sigma
```

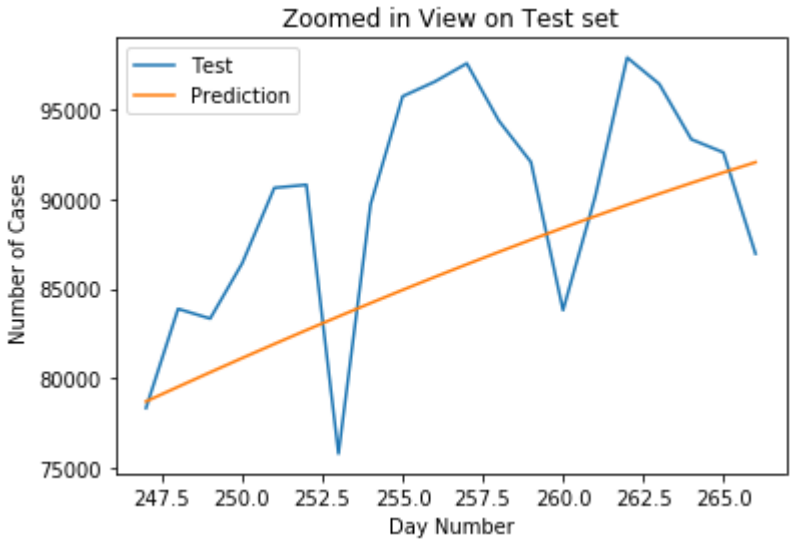
In [28]:
```
# 100 1
l= 95
sigma_f= .6
sigma_y=.1

print("Fitting Train Data - India on Self-fed HyperParameters:")
mu_india, sigma_india = predict_and_plot(Xtrain, Ytrain_India, l, sigma_f, sig
ma_y, Xtrain, Ytrain_India)


print("Predictions - India on Self-fed HyperParameters:")
mu_india, sigma_india = predict_and_plot(Xtrain, Ytrain_India, l, sigma_f, sig
ma_y, Xtest, Ytest_India)

# kernel_visualize()

print("(India)Mean values of predictions with guessed hyperparams day 246 onwa
rds:\n", mu_india, "\n\nSigma:\n", sigma_india)
```

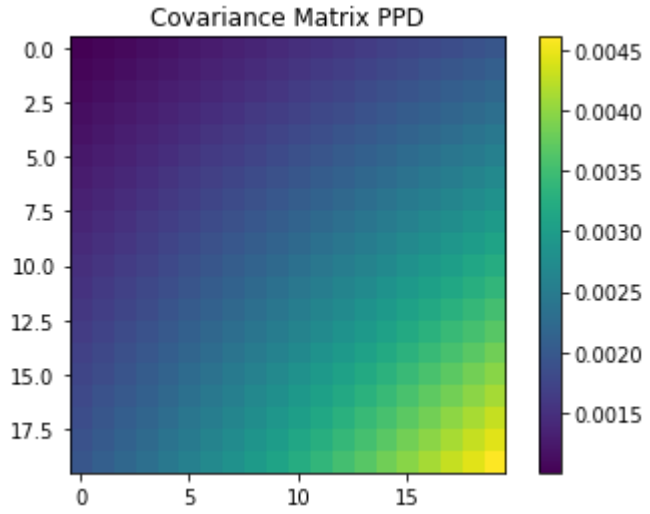
Fitting Train Data - India on Self-fed HyperParameters:



Predictions - India on Self-fed HyperParameters:




Zoomed in View on Test set
Test
Prediction
Number of Cases
95000
90000
85000
80000
75000
247.5
250.0
252.5
255.0
257.5
260.0
262.5
265.0
Day Number

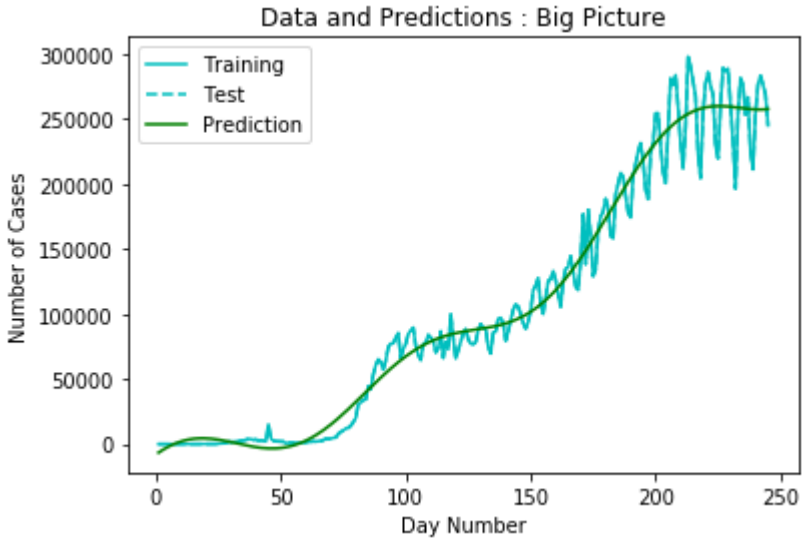

Covariance Matrix PPD
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
5
10
15
0.0045
0.0040
0.0035
0.0030
0.0025
0.0020
0.0015

```
(India)Mean values of predictions with guessed hyperparams day 246 onwards:
 [78718.54608308552 79536.89037626066 80344.30785235722 81140.42041637245
 81924.85588665327 82697.2482684448 83457.23802125627 84204.47231995096
 84938.60530914133 85659.29835076463 86366.2202646425 87059.04756167212
 87737.4646696469 88401.16415133387 89049.84691470122 89683.22241516887
 90301.00884962236 90902.9333420857 91488.73212094184 92058.1506875036]

Sigma:
 [[0.00100225 0.00104659 0.00109176 0.00113777 0.00118459 0.00123222
  0.00128064 0.00132984 0.0013798  0.00143052 0.00148196 0.00153412
  0.00158698 0.00164052 0.00169473 0.00174958 0.00180505 0.00186113
  0.00191779 0.00197501]
 [0.00104659 0.00109387 0.00114208 0.0011912  0.00124121 0.00129211
  0.00134388 0.0013965  0.00144997 0.00150426 0.00155936 0.00161525
  0.00167192 0.00172934 0.0017875  0.00184638 0.00190595 0.00196619
  0.00202708 0.00208861]
 [0.00109176 0.00114208 0.0011934  0.00124571 0.00129901 0.00135326
  0.00140848 0.00146462 0.0015217  0.00157967 0.00163854 0.00169828
  0.00175887 0.00182029 0.00188253 0.00194555 0.00200934 0.00207389
  0.00213915 0.00220511]
 [0.00113777 0.0011912  0.00124571 0.00130131 0.00135797 0.00141568
  0.00147443 0.0015342  0.00159498 0.00165675 0.00171949 0.00178319
  0.00184781 0.00191335 0.00197979 0.00204709 0.00211523 0.0021842
  0.00225397 0.00232451]
 [0.00118459 0.00124121 0.00129901 0.00135797 0.00141809 0.00147934
  0.00154173 0.00160522 0.00166981 0.00173548 0.0018022  0.00186996
  0.00193874 0.00200852 0.00207927 0.00215097 0.0022236  0.00229713
  0.00237154 0.0024468 ]
 [0.00123222 0.00129211 0.00135326 0.00141568 0.00147934 0.00154424
  0.00161036 0.00167767 0.00174617 0.00181584 0.00188665 0.0019586
  0.00203164 0.00210578 0.00218097 0.0022572  0.00233444 0.00241266
  0.00249184 0.00257195]
 [0.00128064 0.00134388 0.00140848 0.00147443 0.00154173 0.00161036
  0.00168029 0.00175153 0.00182404 0.00189782 0.00197283 0.00204907
  0.0021265  0.0022051  0.00228486 0.00236574 0.00244772 0.00253076
  0.00261486 0.00269996]
 [0.00132984 0.0013965  0.00146462 0.0015342  0.00160522 0.00167767
  0.00175153 0.00182678 0.00190341 0.0019814  0.00206072 0.00214135
  0.00222328 0.00230648 0.00239092 0.00247658 0.00256342 0.00265143
  0.00274057 0.0028308 ]
 [0.0013798  0.00144997 0.0015217  0.00159498 0.00166981 0.00174617
  0.00182404 0.00190341 0.00198425 0.00206656 0.00215029 0.00223544
  0.00232198 0.00240989 0.00249914 0.0025897  0.00268154 0.00277463
  0.00286895 0.00296446]
 [0.00143052 0.00150426 0.00157967 0.00165675 0.00173548 0.00181584
  0.00189782 0.0019814  0.00206656 0.00215328 0.00224153 0.00233131
  0.00242258 0.00251531 0.00260949 0.00270507 0.00280203 0.00290035
  0.00299998 0.0031009 ]
 [0.00148196 0.00155936 0.00163854 0.00171949 0.0018022  0.00188665
  0.00197283 0.00206072 0.00215029 0.00224153 0.00233442 0.00242893
  0.00252504 0.00262272 0.00272194 0.00282267 0.00292489 0.00302856
  0.00313364 0.00324011]
 [0.00153412 0.00161525 0.00169828 0.00178319 0.00186996 0.0019586
  0.00204907 0.00214135 0.00223544 0.00233131 0.00242893 0.00252829
  0.00262935 0.00273209 0.00283648 0.00294248 0.00305008 0.00315923
  0.0032699  0.00338205]
```

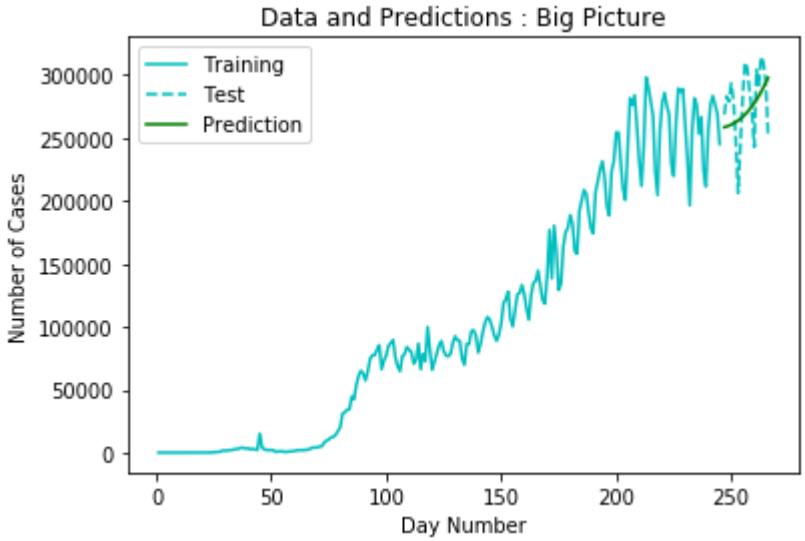
```
[0.00158698 0.00167192 0.00175887 0.00184781 0.00193874 0.00203164
 0.0021265  0.00222328 0.00232198 0.00242258 0.00252504 0.00262935
 0.00273548 0.00284339 0.00295307 0.00306447 0.00317757 0.00329233
 0.00340872 0.00352669]
[0.00164052 0.00172934 0.00182029 0.00191335 0.00200852 0.00210578
 0.0022051  0.00230648 0.00240989 0.00251531 0.00262272 0.00273209
 0.00284339 0.0029566  0.00307168 0.0031886  0.00330733 0.00342783
 0.00355007 0.003674  ]
[0.00169473 0.0017875  0.00188253 0.00197979 0.00207927 0.00218097
 0.00228486 0.00239092 0.00249914 0.00260949 0.00272194 0.00283648
 0.00295307 0.00307168 0.00319228 0.00331484 0.00343933 0.0035657
 0.00369392 0.00382395]
[0.00174958 0.00184638 0.00194555 0.00204709 0.00215097 0.0022572
 0.00236574 0.00247658 0.0025897  0.00270507 0.00282267 0.00294248
 0.00306447 0.0031886  0.00331484 0.00344317 0.00357353 0.0037059
 0.00384023 0.00397649]
[0.00180505 0.00190595 0.00200934 0.00211523 0.0022236  0.00233444
 0.00244772 0.00256342 0.00268154 0.00280203 0.00292489 0.00305008
 0.00317757 0.00330733 0.00343933 0.00357353 0.0037099  0.00384839
 0.00398897 0.00413159]
[0.00186113 0.00196619 0.00207389 0.0021842  0.00229713 0.00241266
 0.00253076 0.00265143 0.00277463 0.00290035 0.00302856 0.00315923
 0.00329233 0.00342783 0.0035657  0.0037059  0.00384839 0.00399313
 0.00414009 0.0042892 ]
[0.00191779 0.00202708 0.00213915 0.00225397 0.00237154 0.00249184
 0.00261486 0.00274057 0.00286895 0.00299998 0.00313364 0.0032699
 0.00340872 0.00355007 0.00369392 0.00384023 0.00398897 0.00414009
 0.00429354 0.00444929]
[0.00197501 0.00208861 0.00220511 0.00232451 0.0024468  0.00257195
 0.00269996 0.0028308  0.00296446 0.0031009  0.00324011 0.00338205
 0.00352669 0.003674   0.00382395 0.00397649 0.00413159 0.0042892
 0.00444929 0.00461179]]
```

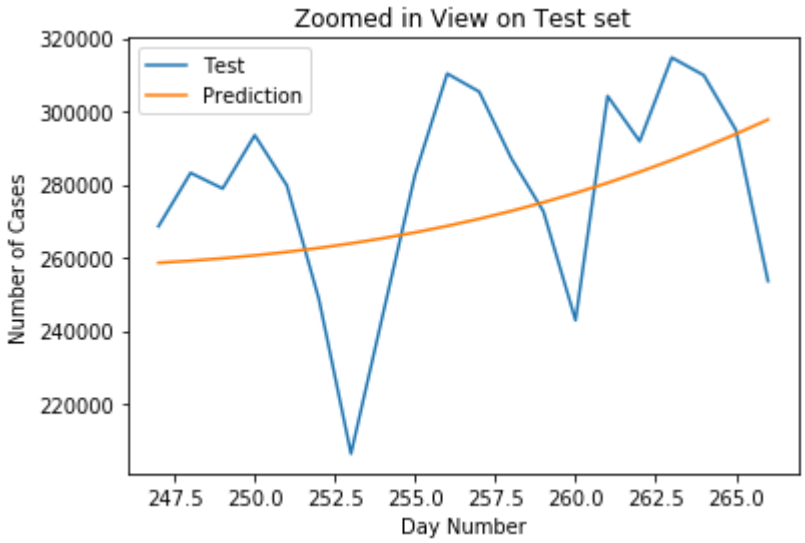
In [29]:
```
# 75 15
l= 75
sigma_f= 15
sigma_y=.1

print("Fiited Train Data - World on Self-fed HyperParameters:")
mu_world, sigma_world = predict_and_plot(Xtrain, Ytrain_World, l, sigma_f, sig
ma_y, Xtrain, Ytrain_World)

print("Predictions - World on Self-fed HyperParameters:")
mu_world, sigma_world = predict_and_plot(Xtrain, Ytrain_World, l, sigma_f, sig
ma_y, Xtest, Ytest_World)

print("(World)Mean values of predictions with guessed hyperparams day 246 onwa
rds:\n", mu_world, "\n\nSigma:\n", sigma_world)
```

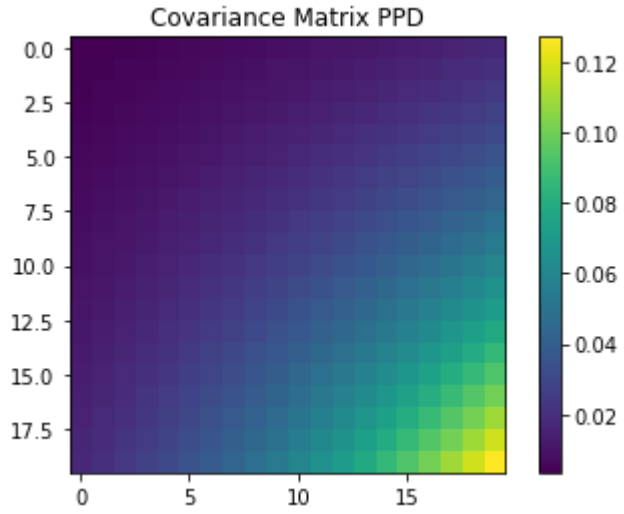
Fiited Train Data - World on Self-fed HyperParameters:



Predictions - World on Self-fed HyperParameters:




Zoomed in View on Test set
Test
Prediction
Number of Cases
320000
300000
280000
260000
240000
220000
247.5
250.0
252.5
255.0
257.5
260.0
262.5
265.0
Day Number


Covariance Matrix PPD
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
5
10
15
0.12
0.10
0.08
0.06
0.04
0.02

```
(World)Mean values of predictions with guessed hyperparams day 246 onwards:
 [258569.30143434135 259116.8456795354 259787.4325565353 260589.72106457135
 261532.25391827198 262623.43641178776 263871.51564264146
 265284.5598896381 266870.4384475909 268636.8018615223 270591.0625609751
 272740.37594253756 275091.622155956 277651.3882882107 280425.9512230514
 283421.26130390796 286642.9264358747 290096.1972081048 293785.95254269836
 297716.68630747043]

Sigma:
 [[0.00346758 0.00389563 0.00435381 0.00484314 0.00536464 0.00591931
  0.00650815 0.00713214 0.00779221 0.0084893  0.00922431 0.00999812
  0.01081156 0.01166545 0.01256056 0.01349764 0.01447738 0.01550044
  0.01656743 0.01767892]
 [0.00389564 0.00439276 0.00492567 0.00549563 0.0061039  0.0067517
  0.00744027 0.00817079 0.00894444 0.00976237 0.0106257  0.0115355
  0.01249284 0.01349873 0.01455414 0.01566    0.01681719 0.01802654
  0.01928885 0.02060483]
 [0.00435381 0.00492566 0.0055395  0.00619686 0.00689922 0.0076481
  0.00844496 0.00929124 0.01018838 0.01113776 0.01214074 0.01319865
  0.01431277 0.01548434 0.01671456 0.01800456 0.01935545 0.02076825
  0.02224393 0.02378341]
 [0.00484314 0.00549562 0.00619685 0.00694863 0.00775275 0.00861098
  0.00952506 0.01049674 0.0115277  0.01261961 0.01377411 0.01499277
  0.01627714 0.01762872 0.01904894 0.02053918 0.02210078 0.02373498
  0.02544297 0.02722587]
 [0.00536463 0.00610389 0.00689922 0.00775275 0.00866657 0.00964276
  0.01068339 0.01179049 0.01296606 0.01421207 0.01553045 0.01692306
  0.01839175 0.01993829 0.02156439 0.02327169 0.02506179 0.02693618
  0.02889629 0.03094347]
 [0.00591931 0.0067517  0.0076481  0.00861098 0.00964277 0.01074589
  0.01192275 0.01317571 0.01450712 0.01591926 0.0174144  0.01899473
  0.0206624  0.02241949 0.02426802 0.02620993 0.02824709 0.03038129
  0.03261421 0.03494745]
 [0.00650815 0.00744026 0.00844496 0.00952506 0.0106834  0.01192275
  0.01324589 0.01465555 0.01615445 0.01774525 0.01943054 0.0212129
  0.02309481 0.02507869 0.0271669  0.0293617  0.03166528 0.03407972
  0.03660701 0.03924904]
 [0.00713213 0.00817078 0.00929123 0.01049673 0.01179049 0.0131757
  0.01465555 0.01623317 0.01791166 0.01969409 0.02158346 0.02358271
  0.0256947  0.02792225 0.03026806 0.03273477 0.03532489 0.03804086
  0.04088498 0.04385946]
 [0.0077922  0.00894443 0.01018837 0.0115277  0.01296606 0.01450711
  0.01615445 0.01791166 0.01978229 0.02176982 0.02387768 0.02610923
  0.02846777 0.03095649 0.03357852 0.03633687 0.03923445 0.04227405
  0.04545835 0.04878987]
 [0.0084893  0.00976236 0.01113775 0.01261961 0.01421208 0.01591926
  0.01774525 0.0196941  0.02176983 0.02397638 0.02631766 0.02879748
  0.0314196  0.03418765 0.03710518 0.04017565 0.04340237 0.04678854
  0.05033722 0.05405132]
 [0.00922431 0.01062569 0.01214073 0.0137741  0.01553045 0.01741439
  0.01943054 0.02158346 0.02387768 0.02631766 0.02890779 0.03165239
  0.0345557  0.03762185 0.04085486 0.04425864 0.04783697 0.05159347
  0.05553163 0.05965479]
 [0.00999812 0.0115355  0.01319865 0.01499277 0.01692307 0.01899473
  0.0212129  0.02358272 0.02610924 0.02879749 0.0316524  0.03467883
  0.03788154 0.04126519 0.04483431 0.04859332 0.05254647 0.05669789
```

```
 0.06105151 0.06561111]
[0.01081157 0.01249285 0.01431277 0.01627715 0.01839176 0.0206624
 0.02309481 0.02569472 0.02846778 0.0314196  0.03455571 0.03788154
 0.04140243 0.0451236  0.04905013 0.05318698 0.05753896 0.06211067
 0.06690658 0.07193095]
[0.01166545 0.01349872 0.01548434 0.01762871 0.01993829 0.02241948
 0.02507868 0.02792225 0.0309565  0.03418764 0.03762185 0.04126518
 0.04512359 0.04920289 0.05350879 0.05804682 0.06282234 0.06784056
 0.07310645 0.07862481]
[0.01256057 0.01455414 0.01671456 0.01904894 0.02156439 0.02426802
 0.0271669  0.03026807 0.03357853 0.03710519 0.04085487 0.04483431
 0.04905013 0.0535088  0.05821666 0.06317987 0.06840443 0.07389612
 0.07966053 0.08570301]
[0.01349764 0.01565999 0.01800456 0.02053918 0.02327169 0.02620992
 0.02936169 0.03273477 0.03633687 0.04017565 0.04425864 0.04859331
 0.05318697 0.05804682 0.06317986 0.06859296 0.07429275 0.08028569
 0.08657797 0.09317558]
[0.01447738 0.01681719 0.01935545 0.02210078 0.02506179 0.02824709
 0.03166528 0.0353249  0.03923446 0.04340238 0.04783697 0.05254647
 0.05753895 0.06282235 0.06840443 0.07429276 0.08049471 0.08701741
 0.09386775 0.10105236]
[0.01550044 0.01802654 0.02076825 0.02373498 0.02693618 0.03038129
 0.03407972 0.03804087 0.04227406 0.04678854 0.05159347 0.05669788
 0.06211067 0.06784056 0.07389612 0.08028569 0.08701741 0.09409914
 0.10153852 0.10934287]
[0.01656743 0.01928884 0.02224393 0.02544297 0.02889629 0.0326142
 0.03660701 0.04088499 0.04545835 0.05033722 0.05553164 0.0610515
 0.06690658 0.07310645 0.07966053 0.08657798 0.09386775 0.10153852
 0.10959869 0.11805634]
[0.01767892 0.02060482 0.0237834  0.02722587 0.03094347 0.03494744
 0.03924904 0.04385946 0.04878987 0.05405132 0.05965479 0.0656111
 0.07193094 0.07862481 0.085703   0.09317558 0.10105236 0.10934287
 0.11805634 0.12720166]]
```

In [30]:
```
# hyper parameter optimization util function
def hyperparameter_opt(sigma_f, l, Xtrain, Ytrain, Xtest, Ytest):
    print("------------LOG OF OPTIMIZATION ALGORITHM - BEGIN------------")
    min_err = 9999999999.0
    sig_opt = 9999.0
    l_opt = 9999.0
    sig = max(sigma_f - 5.0, .1)
    sigma_lwr = sig
    ll = max(l - 50.0, 1)
    ll_lwr = ll
    while sig <= sigma_lwr + 10.0:
        ll = ll_lwr
        while ll <= ll_lwr + 100.0:
            mu_pred, sigma_pred = posterior_predictive(Xtrain, Ytrain, ll, sig
ma_y, sig, Xtest)
            err = abs(Ytest - mu_pred)
            err = err.sum()
            print(sig, ll, err)

            if err < min_err:
                min_err = err
                sig_opt = sig
                l_opt = ll
                print("new minima::", err, sig_opt, l_opt)

            ll+=10.0
        sig+=.5


    print("------------LOG OF OPTIMIZATION ALGORITHM - END------------")
    return sig_opt, l_opt
```

In [31]:
```
# hyperparameter optimization India
sigma_f = .6
l = 96
sig_opt, l_opt = hyperparameter_opt(sigma_f, l, Xtrain, Ytrain_India, Xtest, Y
test_India)
```

```
------------LOG OF OPTIMIZATION ALGORITHM - BEGIN------------
0.1 46.0 507331.55033745035
new minima:: 507331.55033745035 0.1 46.0
0.1 56.0 436727.30290450406
new minima:: 436727.30290450406 0.1 56.0
0.1 66.0 395413.4473520413
new minima:: 395413.4473520413 0.1 66.0
0.1 76.0 374966.4694038131
new minima:: 374966.4694038131 0.1 76.0
0.1 86.0 368522.28900331166
new minima:: 368522.28900331166 0.1 86.0
0.1 96.0 371767.78697517957
0.1 106.0 382895.4158718179
0.1 116.0 399579.85282489704
0.1 126.0 419016.4139206412
0.1 136.0 439266.6960433946
0.1 146.0 459628.65568101784
0.6 46.0 354276.2135445648
new minima:: 354276.2135445648 0.6 46.0
0.6 56.0 308883.41179451917
new minima:: 308883.41179451917 0.6 56.0
0.6 66.0 239991.31906947907
new minima:: 239991.31906947907 0.6 66.0
0.6 76.0 177347.439881915
new minima:: 177347.439881915 0.6 76.0
0.6 86.0 137749.87460714544
new minima:: 137749.87460714544 0.6 86.0
0.6 96.0 114128.37362284884
new minima:: 114128.37362284884 0.6 96.0
0.6 106.0 102944.84026764937
new minima:: 102944.84026764937 0.6 106.0
0.6 116.0 101356.39682240576
new minima:: 101356.39682240576 0.6 116.0
0.6 126.0 101117.30156093063
new minima:: 101117.30156093063 0.6 126.0
0.6 136.0 101591.4851939279
0.6 146.0 102528.69290579746
1.1 46.0 295609.9199985954
1.1 56.0 320607.5500214271
1.1 66.0 273026.6807848157
1.1 76.0 210236.74366253597
1.1 86.0 152171.79695279928
1.1 96.0 120038.40247537116
1.1 106.0 100763.70372466155
new minima:: 100763.70372466155 1.1 106.0
1.1 116.0 96057.12809262448
new minima:: 96057.12809262448 1.1 116.0
1.1 126.0 94910.17674441001
new minima:: 94910.17674441001 1.1 126.0
1.1 136.0 97197.74934769534
1.1 146.0 98875.56752886248
1.6 46.0 233035.61537911528
1.6 56.0 315470.7829552739
1.6 66.0 296787.63633724255
1.6 76.0 240693.99111166722
1.6 86.0 179889.05408696545
```

```
1.6 96.0 134015.37767191284
1.6 106.0 106573.08065890818
1.6 116.0 97152.11105066788
1.6 126.0 93642.9157565114
new minima:: 93642.9157565114 1.6 126.0
1.6 136.0 96373.84920454123
1.6 146.0 99444.91297737409
2.1 46.0 175583.53463372763
2.1 56.0 300731.42256452097
2.1 66.0 311208.8801623306
2.1 76.0 264376.4741965146
2.1 86.0 206142.54474530992
2.1 96.0 149853.46290859277
2.1 106.0 117919.99023215573
2.1 116.0 99306.46818257595
2.1 126.0 94064.23180966613
2.1 136.0 95116.16958266002
2.1 146.0 98911.65917150718
2.6 46.0 140465.5403956194
2.6 56.0 280681.0632639205
2.6 66.0 318119.782161277
2.6 76.0 282888.1003111275
2.6 86.0 227810.0830553195
2.6 96.0 169249.9283933072
2.6 106.0 128926.81894318551
2.6 116.0 103653.43089542429
2.6 126.0 95957.97664654687
2.6 136.0 93720.1201235463
2.6 146.0 97536.14209135562
3.1 46.0 130887.86840968963
3.1 56.0 258490.02279601552
3.1 66.0 320060.38405104855
3.1 76.0 297523.32803938753
3.1 86.0 245711.3643511632
3.1 96.0 188683.26661489336
3.1 106.0 139623.15404607932
3.1 116.0 111493.17110078355
3.1 126.0 97853.38949792173
3.1 136.0 93232.81716274413
new minima:: 93232.81716274413 3.1 136.0
3.1 146.0 96087.92889793408
3.6 46.0 137638.77391886743
3.6 56.0 235481.153139434
3.6 66.0 318666.2388274641
3.6 76.0 309492.94100488664
3.6 86.0 260787.87903977995
3.6 96.0 205783.28047552187
3.6 106.0 151213.29412965974
3.6 116.0 119650.6673331471
3.6 126.0 100006.6970496701
3.6 136.0 94325.21456580513
3.6 146.0 94578.7334784839
4.1 46.0 155885.8169121585
4.1 56.0 211919.5714848052
4.1 66.0 314951.3222320248
4.1 76.0 318529.4854121415
4.1 86.0 273727.9622757884
```

```
4.1 96.0 220722.63489914965
4.1 106.0 164049.80322789896
4.1 116.0 127358.49008144476
4.1 126.0 103422.61965921971
4.1 136.0 95888.70233291875
4.1 146.0 93430.88096497496
4.6 46.0 177411.93463979693
4.6 56.0 188744.93504486384
4.6 66.0 309555.5206981065
4.6 76.0 325175.8990914815
4.6 86.0 284983.6508315705
4.6 96.0 233797.92919666888
4.6 106.0 177778.17761447857
4.6 116.0 134703.9563962551
4.6 126.0 108926.60853046471
4.6 136.0 97347.28405813637
4.6 146.0 93030.18877443911
new minima:: 93030.18877443911 4.6 146.0
5.1 46.0 199203.20366722692
5.1 56.0 171623.2195521575
5.1 66.0 302899.0468928838
5.1 76.0 329885.91489458975
5.1 86.0 294848.9137162358
5.1 96.0 245319.91702139407
5.1 106.0 190480.58320729953
5.1 116.0 141982.8781915115
5.1 126.0 114858.10149220322
5.1 136.0 98689.8422937963
5.1 146.0 93606.38620019548
5.6 46.0 223979.31927502566
5.6 56.0 157065.8982563991
5.6 66.0 295271.53788974823
5.6 76.0 333031.04436531645
5.6 86.0 303856.25700242317
5.6 96.0 255565.80241777265
5.6 106.0 202147.40783788226
5.6 116.0 150083.52278489064
5.6 126.0 120464.2468101966
5.6 136.0 100923.6714163128
5.6 146.0 94843.27352284675
6.1 46.0 251398.03776590215
6.1 56.0 145211.08322461633
6.1 66.0 286882.78602153907
6.1 76.0 334910.6322397542
6.1 86.0 312050.15852069104
6.1 96.0 264764.82419175183
6.1 106.0 212832.69508946582
6.1 116.0 158928.77028404793
6.1 126.0 125811.32203755461
6.1 136.0 103357.60460073865
6.1 146.0 96006.3465195979
6.6 46.0 279665.43054316676
6.6 56.0 138844.6624203369
6.6 66.0 277891.86406891275
6.6 76.0 335763.5831506222
6.6 86.0 319242.45241123124
6.6 96.0 273099.1323322463
```

```
6.6 106.0 222610.52194036206
6.6 116.0 167865.08365845875
6.6 126.0 130946.7670778192
6.6 136.0 107260.78420521565
6.6 146.0 97092.74915778024
7.1 46.0 308162.86119567824
7.1 56.0 137539.29572207935
7.1 66.0 268641.0561350645
7.1 76.0 335779.7496567705
7.1 86.0 325535.24910843204
7.1 96.0 280710.42236946773
7.1 106.0 231564.2758606294
7.1 116.0 177135.38618122513
7.1 126.0 135902.8635412496
7.1 136.0 111590.01499390793
7.1 146.0 98104.6475651945
7.6 46.0 335939.0333709341
7.6 56.0 137521.4956811614
7.6 66.0 259333.60196767666
7.6 76.0 335109.9062980062
7.6 86.0 331017.44798128295
7.6 96.0 287707.8356804639
7.6 106.0 239779.01285985333
7.6 116.0 185942.68827027
7.6 126.0 140831.68899169218
7.6 136.0 115704.4908001721
7.6 146.0 99267.92806592205
8.1 46.0 363222.29872652167
8.1 56.0 140173.34700900712
8.1 66.0 249752.39205444057
8.1 76.0 333873.99666522315
8.1 86.0 335768.39331336133
8.1 96.0 294175.2624271391
8.1 106.0 247336.56365112704
8.1 116.0 194291.84622663594
8.1 126.0 145965.2896478692
8.1 136.0 119640.06282869616
8.1 146.0 101151.34317507292
8.6 46.0 390614.67665001267
8.6 56.0 145777.1362110747
8.6 66.0 239963.9808501622
8.6 76.0 332167.7383661878
8.6 86.0 339859.74083668436
8.6 96.0 300177.4376050789
8.6 106.0 254312.7404827007
8.6 116.0 202192.75159674394
8.6 126.0 152173.60214788743
8.6 136.0 123425.75651248143
8.6 146.0 102925.10466552922
9.1 46.0 416690.44674090913
9.1 56.0 154177.64325097448
9.1 66.0 230024.75150859007
9.1 76.0 330067.82405843894
9.1 86.0 343356.4627326562
9.1 96.0 305991.562065605
9.1 106.0 260776.00842923048
9.1 116.0 209659.70231811533
```

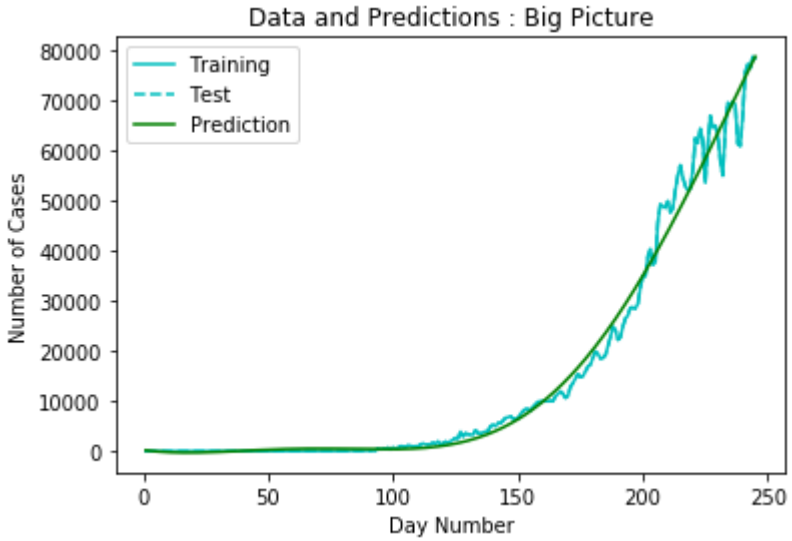
```
9.1 126.0 158200.2419392714
9.1 136.0 127084.60323660127
9.1 146.0 105376.78164226806
9.6 46.0 441550.7594719719
9.6 56.0 162427.10090939677
9.6 66.0 219982.58534486362
9.6 76.0 327635.9843591622
9.6 86.0 346317.46838640666
9.6 96.0 311581.6134621593
9.6 106.0 266787.10803741857
9.6 116.0 216710.4708728182
9.6 126.0 164047.70642606504
9.6 136.0 130634.58337678299
9.6 146.0 108602.92919770688
10.1 46.0 465742.1442830615
10.1 56.0 170523.5176695251
10.1 66.0 209878.09404623014
10.1 76.0 324922.1534826794
10.1 86.0 348796.0719431857
10.1 96.0 316803.24251379725
10.1 106.0 272399.25476009754
10.1 116.0 223365.31527639012
10.1 126.0 170574.96561133538
10.1 136.0 134089.5324536739
10.1 146.0 111687.75997153163
------------LOG OF OPTIMIZATION ALGORITHM - END------------
```

In [32]:
```
# 1.6 21
# sig_opt = 1.6
# l_opt = 21.0
print("Optimized Hyperparams India (sigma_f, l):", sig_opt, l_opt)
```

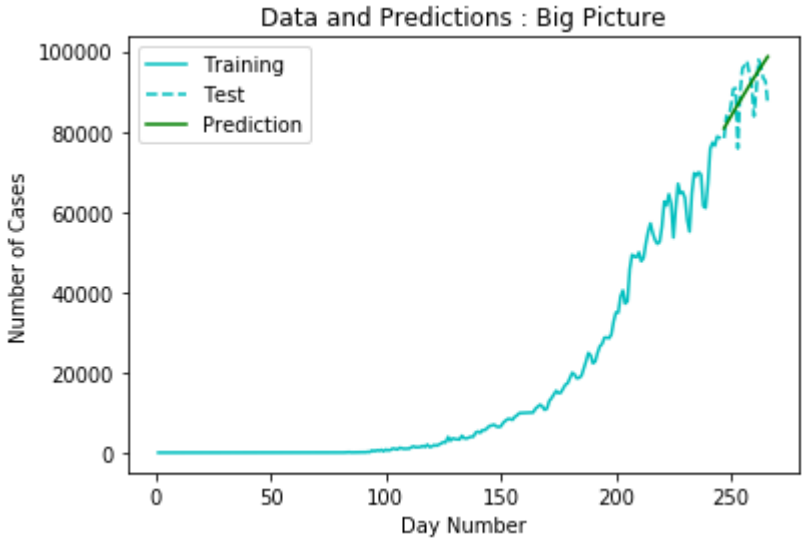
```
Optimized Hyperparams India (sigma_f, l): 4.6 146.0
```

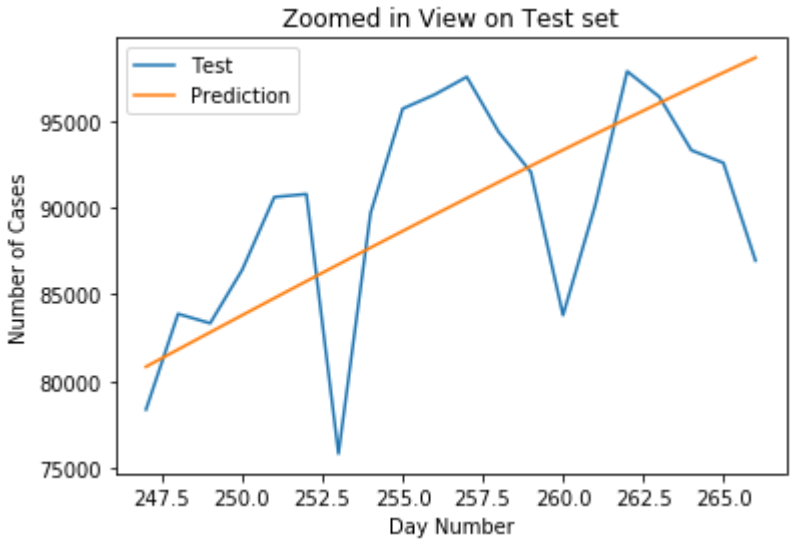
In [33]:
```
print("Fitted Train Data - India on Optimized HyperParameters:")
mu_world, sigma_world = predict_and_plot(Xtrain, Ytrain_India, l_opt, sig_opt,
sigma_y, Xtrain, Ytrain_India)

print("Predictions - India on Optimized HyperParameters:")
mu_india, sigma_india = predict_and_plot(Xtrain, Ytrain_India, l_opt, sig_opt,
sigma_y, Xtest, Ytest_India)

print("(INdia) Means of Predictions from optimized hyperparams day 246 onward
s:\n",mu_india, "\n\nSigma :\n", sigma_india)
```

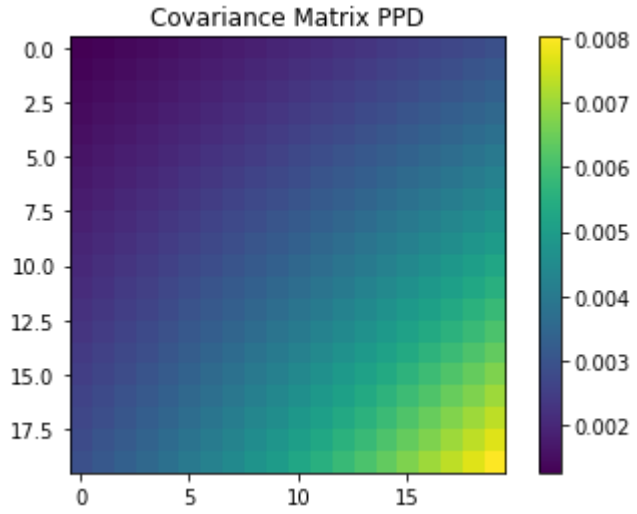
Fitted Train Data - India on Optimized HyperParameters:



Predictions - India on Optimized HyperParameters:




Zoomed in View on Test set
Test
Prediction
Number of Cases
95000
90000
85000
80000
75000
247.5
250.0
252.5
255.0
257.5
260.0
262.5
265.0
Day Number


Covariance Matrix PPD
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
5
10
15
0.008
0.007
0.006
0.005
0.004
0.003
0.002

```
(INdia) Means of Predictions from optimized hyperparams day 246 onwards:
 [80830.33713926667 81827.94891686203 82821.16098395456 83809.61562207522
 84792.95458519823 85770.81926483153 86742.85085822233 87708.69053508854
 88667.97960550014 89620.35968843459 90565.47287895514 91502.96191802621
 92432.47035762174 93353.6427324545 94266.12472519593 95169.56333349645
 96063.60703915302 96947.90597234876 97822.11207864305 98685.87928226178]

Sigma :
 [[0.00125785 0.00132604 0.00139626 0.00146855 0.00154293 0.00161941
  0.00169802 0.00177878 0.00186172 0.00194685 0.00203418 0.00212375
  0.00221557 0.00230965 0.00240602 0.00250469 0.00260568 0.00270899
  0.00281465 0.00292267]
 [0.00132604 0.00139934 0.00147487 0.00155265 0.0016327  0.00171506
  0.00179975 0.00188678 0.00197619 0.002068   0.00216222 0.00225888
  0.00235801 0.00245962 0.00256373 0.00267035 0.00277952 0.00289124
  0.00300552 0.0031224 ]
 [0.00139626 0.00147487 0.00155589 0.00163936 0.00172531 0.00181376
  0.00190475 0.00199829 0.00209442 0.00219315 0.00229453 0.00239856
  0.00250527 0.0026147  0.00272684 0.00284174 0.0029594  0.00307985
  0.00320311 0.0033292 ]
 [0.00146855 0.00155265 0.00163936 0.00172873 0.00182078 0.00191555
  0.00201307 0.00211336 0.00221645 0.00232238 0.00243117 0.00254284
  0.00265743 0.00277496 0.00289545 0.00301893 0.00314542 0.00327494
  0.00340752 0.00354316]
 [0.00154293 0.0016327  0.00172531 0.00182078 0.00191916 0.00202047
  0.00212474 0.00223202 0.00234234 0.00245572 0.00257219 0.00269179
  0.00281454 0.00294048 0.00306963 0.00320201 0.00333766 0.00347659
  0.00361883 0.0037644 ]
 [0.00161941 0.00171506 0.00181376 0.00191555 0.00202047 0.00212855
  0.00223982 0.00235434 0.00247213 0.00259322 0.00271766 0.00284547
  0.00297668 0.00311134 0.00324945 0.00339107 0.0035362  0.00368489
  0.00383716 0.00399303]
 [0.00169802 0.00179975 0.00190475 0.00201307 0.00212474 0.00223982
  0.00235835 0.00248035 0.00260587 0.00273495 0.00286763 0.00300394
  0.00314391 0.00328759 0.003435   0.00358617 0.00374114 0.00389994
  0.00406259 0.00422913]
 [0.00177878 0.00188678 0.00199829 0.00211335 0.00223202 0.00235434
  0.00248035 0.00261009 0.00274362 0.00288096 0.00302216 0.00316726
  0.0033163  0.00346931 0.00362633 0.0037874  0.00395256 0.00412182
  0.00429523 0.00447282]
 [0.00186172 0.00197619 0.00209442 0.00221645 0.00234234 0.00247213
  0.00260587 0.00274362 0.0028854  0.00303128 0.0031813  0.00333549
  0.0034939  0.00365657 0.00382354 0.00399484 0.00417053 0.00435062
  0.00453517 0.00472419]
 [0.00194685 0.002068   0.00219315 0.00232238 0.00245572 0.00259322
  0.00273495 0.00288096 0.00303128 0.00318598 0.0033451  0.00350868
  0.00367677 0.00384942 0.00402667 0.00420857 0.00439514 0.00458644
  0.00478249 0.00498335]
 [0.00203418 0.00216222 0.00229453 0.00243117 0.00257219 0.00271766
  0.00286763 0.00302216 0.0031813  0.0033451  0.00351361 0.00368689
  0.00386499 0.00404795 0.00423582 0.00442865 0.00462647 0.00482934
  0.0050373  0.00525038]
 [0.00212375 0.00225888 0.00239856 0.00254284 0.00269179 0.00284547
  0.00300394 0.00316726 0.00333549 0.00350868 0.00368689 0.00387018
  0.0040586  0.0042522  0.00445104 0.00465515 0.0048646  0.00507942
  0.00529967 0.00552538]
```

```
[0.00221557 0.00235801 0.00250527 0.00265743 0.00281455 0.00297668
 0.00314391 0.0033163  0.0034939  0.00367677 0.00386499 0.0040586
 0.00425767 0.00446225 0.0046724  0.00488816 0.0051096  0.00533676
 0.0055697  0.00580845]
[0.00230965 0.00245962 0.00261469 0.00277496 0.00294048 0.00311134
 0.00328759 0.00346931 0.00365657 0.00384942 0.00404795 0.0042522
 0.00446225 0.00467815 0.00489996 0.00512775 0.00536155 0.00560144
 0.00584746 0.00609967]
[0.00240602 0.00256373 0.00272684 0.00289545 0.00306963 0.00324945
 0.003435   0.00362633 0.00382353 0.00402667 0.00423582 0.00445104
 0.0046724  0.00489996 0.0051338  0.00537397 0.00562053 0.00587354
 0.00613306 0.00639914]
[0.00250469 0.00267035 0.00284174 0.00301893 0.00320201 0.00339107
 0.00358617 0.0037874  0.00399484 0.00420857 0.00442865 0.00465515
 0.00488816 0.00512775 0.00537397 0.0056269  0.00588659 0.00615312
 0.00642655 0.00670693]
[0.00260567 0.00277952 0.0029594  0.00314542 0.00333766 0.0035362
 0.00374114 0.00395256 0.00417053 0.00439514 0.00462647 0.0048646
 0.0051096  0.00536155 0.00562053 0.00588659 0.00615982 0.00644028
 0.00672803 0.00702314]
[0.00270899 0.00289124 0.00307985 0.00327494 0.00347659 0.00368489
 0.00389994 0.00412182 0.00435062 0.00458644 0.00482934 0.00507942
 0.00533676 0.00560144 0.00587354 0.00615312 0.00644028 0.00673507
 0.00703757 0.00734785]
[0.00281465 0.00300552 0.00320311 0.00340752 0.00361883 0.00383716
 0.00406259 0.00429523 0.00453517 0.00478249 0.0050373  0.00529967
 0.0055697  0.00584746 0.00613306 0.00642655 0.00672803 0.00703757
 0.00735525 0.00768114]
[0.00292267 0.0031224  0.0033292  0.00354316 0.0037644  0.00399303
 0.00422913 0.00447282 0.00472419 0.00498335 0.00525038 0.00552538
 0.00580845 0.00609967 0.00639914 0.00670693 0.00702314 0.00734785
 0.00768114 0.00802308]]
```

In [34]:
```
# hyper paramter optimization worlld
sigma_f = 15.0
l = 75.0
sig_opt, l_opt = hyperparameter_opt(sigma_f, l, Xtrain, Ytrain_World, Xtest, Y
test_World)
```

```
------------LOG OF OPTIMIZATION ALGORITHM - BEGIN------------
10.0 25.0 433083.46064743894
new minima:: 433083.46064743894 10.0 25.0
10.0 35.0 1109748.3423302132
10.0 45.0 964655.5113760114
10.0 55.0 526371.1994394772
10.0 65.0 564782.6017034996
10.0 75.0 535480.6888447056
10.0 85.0 777588.9995223928
10.0 95.0 1363262.9048331233
10.0 105.0 1764427.163628351
10.0 115.0 1813899.010951848
10.0 125.0 1656140.0479736696
10.5 25.0 433990.8253892086
10.5 35.0 1128175.137308037
10.5 45.0 969732.5621854138
10.5 55.0 567961.3873032294
10.5 65.0 562539.9082013303
10.5 75.0 524152.00156320864
10.5 85.0 741447.6828703347
10.5 95.0 1325937.9774779256
10.5 105.0 1754948.993197461
10.5 115.0 1832127.0161212576
10.5 125.0 1686329.8289438784
11.0 25.0 434029.95672661584
11.0 35.0 1143092.8889100952
11.0 45.0 972674.4620409025
11.0 55.0 612516.6423141551
11.0 65.0 560135.544992432
11.0 75.0 513863.6669142595
11.0 85.0 712779.5430133186
11.0 95.0 1288868.5653412533
11.0 105.0 1743138.5839874437
11.0 115.0 1847622.948334382
11.0 125.0 1714610.031032951
11.5 25.0 433248.50687805965
11.5 35.0 1154819.9194937767
11.5 45.0 973891.9512752712
11.5 55.0 656612.0966319846
11.5 65.0 557507.6001258946
11.5 75.0 504530.3708595925
11.5 85.0 687433.759798723
11.5 95.0 1252256.3215780354
11.5 105.0 1729318.8977978313
11.5 115.0 1860591.8228304556
11.5 125.0 1741077.848674765
12.0 25.0 431702.11515034334
new minima:: 431702.11515034334 12.0 25.0
12.0 35.0 1163651.3482094104
12.0 45.0 973739.1372915043
12.0 55.0 699425.8371466185
12.0 65.0 554608.7416263991
12.0 75.0 496075.0570313707
12.0 85.0 669783.3290309409
12.0 95.0 1216254.0007046885
12.0 105.0 1713774.4380739322
```

```
12.0 115.0 1871225.3953399062
12.0 125.0 1765819.1300378756
12.5 25.0 429451.2149388978
new minima:: 429451.2149388978 12.5 25.0
12.5 35.0 1169860.2284690926
12.5 45.0 972520.7903411147
12.5 55.0 740919.7276672418
12.5 65.0 551403.6969841081
12.5 75.0 488427.60925530375
12.5 85.0 653759.3395256626
12.5 95.0 1180974.592643592
12.5 105.0 1696756.1617020643
12.5 115.0 1879702.5462487969
12.5 125.0 1788911.8288130085
13.0 25.0 426558.6460114247
new minima:: 426558.6460114247 13.0 25.0
13.0 35.0 1173698.7967461734
13.0 45.0 970498.6338711138
13.0 55.0 781373.1779201091
13.0 65.0 547867.1543260051
13.0 75.0 481523.86796613707
13.0 85.0 638486.065956674
13.0 95.0 1146498.868262622
13.0 105.0 1678485.7121759611
13.0 115.0 1886189.733131309
13.0 125.0 1810428.3095797228
13.5 25.0 423150.1197710543
new minima:: 423150.1197710543 13.5 25.0
13.5 35.0 1175399.7455638605
13.5 45.0 967896.8023522377
13.5 55.0 825404.410749827
13.5 65.0 543982.0098253079
13.5 75.0 476692.87741706776
13.5 85.0 623906.3302504298
13.5 95.0 1112881.6178648518
13.5 105.0 1659159.0575188054
13.5 115.0 1890841.524537323
13.5 125.0 1830436.8558752781
14.0 25.0 419799.50751137926
new minima:: 419799.50751137926 14.0 25.0
14.0 35.0 1175177.4752721682
14.0 45.0 964906.5970646802
14.0 55.0 867925.5151329825
14.0 65.0 539737.9019837242
14.0 75.0 472442.2506720423
14.0 85.0 609969.1102536382
14.0 95.0 1080156.8078999417
14.0 105.0 1638949.6126885742
14.0 115.0 1893801.20484644
14.0 125.0 1849002.631625227
14.5 25.0 416088.10435493616
new minima:: 416088.10435493616 14.5 25.0
14.5 35.0 1173229.3015345857
14.5 45.0 961690.6451880048
14.5 55.0 908958.6756384172
14.5 65.0 535129.9860318522
14.5 75.0 468632.62156334455
```

```
14.5 85.0 596629.0065036335
14.5 95.0 1048341.8376322458
14.5 105.0 1618010.9243818163
14.5 115.0 1895201.4269991405
14.5 125.0 1866188.2700497308
15.0 25.0 412064.44870716013
new minima:: 412064.44870716013 15.0 25.0
15.0 35.0 1169736.608177876
15.0 45.0 958386.5435841626
15.0 55.0 948534.104286869
15.0 65.0 530157.9074893608
15.0 75.0 465228.686516186
15.0 85.0 583845.6763644501
15.0 95.0 1017441.046733573
15.0 105.0 1596478.9832020458
15.0 115.0 1895164.8926417646
15.0 125.0 1882054.2201122914
15.5 25.0 407774.8963430753
new minima:: 407774.8963430753 15.5 25.0
15.5 35.0 1164865.9412850987
15.5 45.0 955110.0570305501
15.5 55.0 988562.353945967
15.5 65.0 524824.9429063743
15.5 75.0 462197.7515514483
15.5 85.0 571583.2755709339
15.5 95.0 987448.5965465308
15.5 105.0 1574474.2232838427
15.5 115.0 1893805.0374564826
15.5 125.0 1896658.9316797783
16.0 25.0 404204.2977628131
new minima:: 404204.2977628131 16.0 25.0
16.0 35.0 1158770.0437841264
16.0 45.0 951957.9280372293
16.0 55.0 1030408.341545
16.0 65.0 519137.2822729136
16.0 75.0 460137.9135300922
16.0 85.0 559809.9272005263
16.0 95.0 958350.8315629661
16.0 105.0 1552103.257862432
16.0 115.0 1891226.707996955
16.0 125.0 1910058.9442080809
16.5 25.0 401166.14209111716
new minima:: 401166.14209111716 16.5 25.0
16.5 35.0 1151588.8328942559
16.5 45.0 949010.345403063
16.5 55.0 1071327.3822426992
16.5 65.0 513103.42854908376
16.5 75.0 458906.5357469081
16.5 85.0 548497.2311522918
16.5 95.0 930128.204789299
16.5 105.0 1529460.3915145432
16.5 115.0 1887526.8144268335
16.5 125.0 1922308.9185850616
17.0 25.0 398049.6983200423
new minima:: 398049.6983200423 17.0 25.0
17.0 35.0 1143450.3204132016
17.0 45.0 946333.1157063948
```

```
17.0 55.0 1110749.082378577
17.0 65.0 506733.6967818432
17.0 75.0 457816.6544959249
17.0 85.0 537619.8206266593
17.0 95.0 902756.8435203143
17.0 105.0 1506628.9457796467
17.0 115.0 1882794.9529837717
17.0 125.0 1933461.6378723779
17.5 25.0 394878.4344470545
new minima:: 394878.4344470545 17.5 25.0
17.5 35.0 1134471.479681963
17.5 45.0 943979.569959308
17.5 55.0 1148725.226766632
17.5 65.0 500039.7969161713
17.5 75.0 456856.6050859632
17.5 85.0 527154.9640896121
17.5 95.0 876209.8138794649
17.5 105.0 1483682.4243962762
17.5 115.0 1877113.9927805173
17.5 125.0 1943567.9980871787
18.0 25.0 392457.6564438937
new minima:: 392457.6564438937 18.0 25.0
18.0 35.0 1124759.060948629
18.0 45.0 941992.2402000193
18.0 55.0 1185307.4578459994
18.0 65.0 493034.4863616694
18.0 75.0 456015.5849103664
18.0 85.0 517082.2151345118
18.0 95.0 850458.1378510938
18.0 105.0 1460685.5439724869
18.0 115.0 1870560.6218007677
18.0 125.0 1952676.995615264
18.5 25.0 392188.29634941375
new minima:: 392188.29634941375 18.5 25.0
18.5 35.0 1114410.3556830832
18.5 45.0 940404.3301074187
18.5 55.0 1221009.1946531723
18.5 65.0 485731.28161700955
18.5 75.0 455283.5859322523
18.5 85.0 507383.104668779
18.5 95.0 825471.6044824758
18.5 105.0 1437695.1469858498
18.5 115.0 1863205.853773288
18.5 125.0 1961300.8232859408
19.0 25.0 391933.1509909872
new minima:: 391933.1509909872 19.0 25.0
19.0 35.0 1103513.9147192999
19.0 45.0 939241.0069592879
19.0 55.0 1255607.710412879
19.0 65.0 478544.68697753496
19.0 75.0 454651.3320443611
19.0 85.0 498040.87274899735
19.0 95.0 801219.4130446438
19.0 105.0 1414761.0164193134
19.0 115.0 1855115.4938627533
19.0 125.0 1969168.5842423267
19.5 25.0 391693.90700621455
```

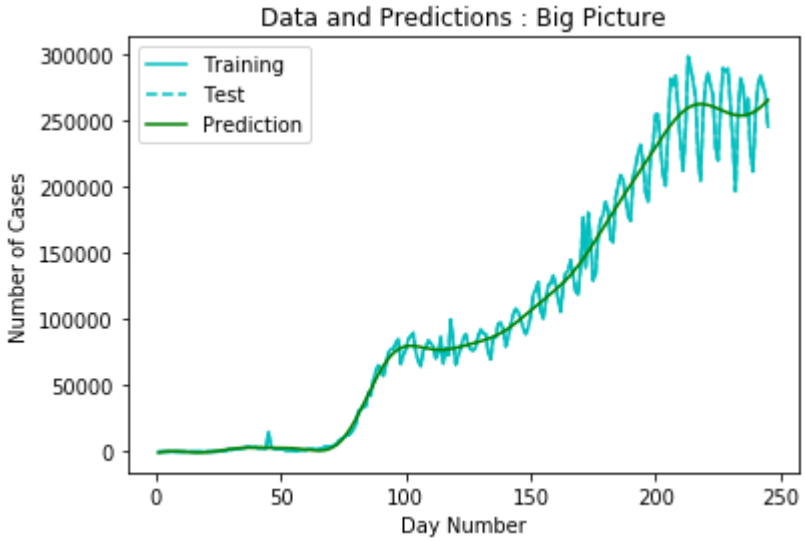
```
new minima:: 391693.90700621455 19.5 25.0
19.5 35.0 1092150.2209287141
19.5 45.0 938520.5336281416
19.5 55.0 1288936.953542548
19.5 65.0 472569.2888960656
19.5 75.0 454110.2226445989
19.5 85.0 489040.23647252517
19.5 95.0 777670.6757985865
19.5 105.0 1391926.6020876411
19.5 115.0 1846350.5661315133
19.5 125.0 1976111.529414915
20.0 25.0 394267.9488773451
20.0 35.0 1080392.3170999198
20.0 45.0 938255.2599563386
20.0 55.0 1321044.4581038544
20.0 65.0 467751.3622443956
20.0 75.0 453652.2805505034
20.0 85.0 480490.95618756633
20.0 95.0 754794.8090580024
20.0 105.0 1369229.6724354967
20.0 115.0 1836967.7041314554
20.0 125.0 1982174.2028239924
------------LOG OF OPTIMIZATION ALGORITHM - END------------
```

In [35]:
```
# 5 201
# sig_opt = 5.0
# l_opt = 201.0
print("Optimized HyperParams World : (sigma_f, l)",sig_opt, l_opt)
```

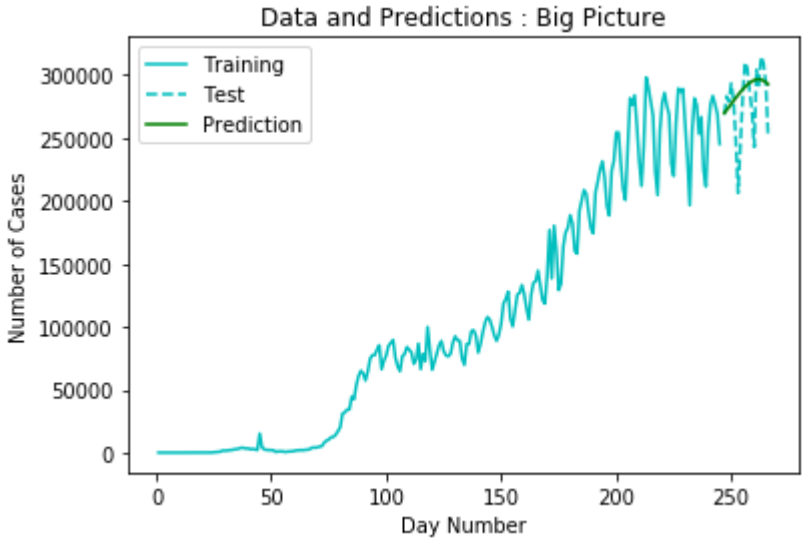
```
Optimized HyperParams World : (sigma_f, l) 19.5 25.0
```

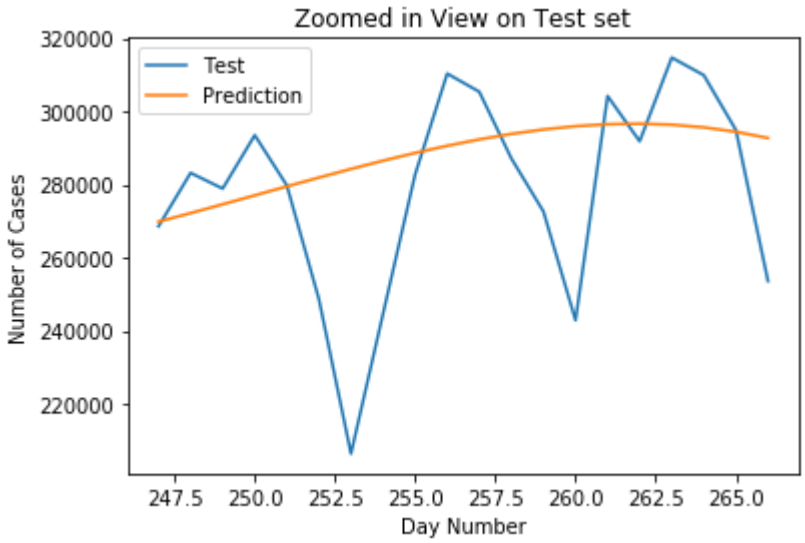
In [36]:
```
print("Fitted Train Data - World on Optimized HyperParameters:")
mu_world, sigma_world = predict_and_plot(Xtrain, Ytrain_World, l_opt, sig_opt,
sigma_y, Xtrain, Ytrain_World)

print("Predictions - World on Optimized HyperParameters:")
mu_world, sigma_world = predict_and_plot(Xtrain, Ytrain_World, l_opt, sig_opt,
sigma_y, Xtest, Ytest_World)

print("(World)Mean values of predictions day 246 onwards:\n", mu_world, "\n\n
 Sigma:\n", sigma_world)
```

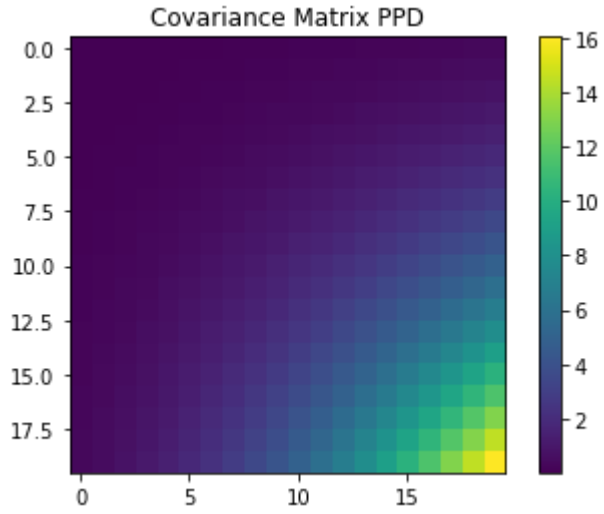
Fitted Train Data - World on Optimized HyperParameters:



Predictions - World on Optimized HyperParameters:




Zoomed in View on Test set
Test
Prediction
Number of Cases
320000
300000
280000
260000
240000
220000
247.5
250.0
252.5
255.0
257.5
260.0
262.5
265.0
Day Number


Covariance Matrix PPD
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
5
10
15
16
14
12
10
8
6
4
2

```
(World)Mean values of predictions day 246 onwards:
 [269814.68984703394 272157.00728556444 274563.1171950136
 277004.40352468356 279450.25102973514 281868.2374701417
 284224.34850549913 286483.2178838195 288608.39544450946 290562.6441696258
 292308.2670903988 293807.46336078923 295022.7117658451 295917.1786163619
 296455.14569650264 296602.4527695584 296326.94790132437
 295598.93810615595 294391.63194160163 292681.56528326217]

 Sigma:
 [[ 0.01874162  0.02527023  0.03302099  0.04208612  0.05254866  0.06448041
   0.0779399   0.09297057  0.10959913  0.12783414  0.14766482  0.16906025
   0.19196883  0.21631805  0.2420147   0.26894535  0.29697722  0.32595939
   0.3557243   0.38608957]
 [ 0.02527023  0.03453878  0.04561322  0.05863938  0.07375049  0.09106397
   0.11067827  0.13266998  0.15709119  0.18396714  0.21329428  0.24503882
   0.2791356   0.31548767  0.35396621  0.39441116  0.43663223  0.48041054
   0.52550073  0.57163351]
 [ 0.03302098  0.04561321  0.06073571  0.07860361  0.09941557  0.12334907
   0.15055585  0.18115762  0.2152421   0.2528595   0.2940195   0.33868886
   0.38678972  0.43819852  0.4927458   0.55021665  0.61035202  0.67285076
   0.73737237  0.8035405 ]
 [ 0.04208611  0.05863938  0.07860361  0.10228127  0.12995401  0.16187619
   0.19826855  0.23931211  0.28514259  0.33584534  0.39145097  0.45193177
   0.51719902  0.5871013   0.66142375  0.73988847  0.82215593  0.90782743
   0.99644869  1.08751423]
 [ 0.05254866  0.0737505   0.09941557  0.12995401  0.16575047  0.20715547
   0.25447682  0.30797147  0.36783774  0.43420843  0.50714467  0.58663085
   0.67257068  0.76478454  0.86300814  0.9668926   1.07600597  1.18983614
   1.30779513  1.42922474]
 [ 0.06448041  0.09106398  0.12334908  0.1618762   0.20715547  0.25965531
   0.31979126  0.38791517  0.46430503  0.54915561  0.64257026  0.74455387
   0.85500742  0.97372405  1.10038685  1.23456852  1.37573281  1.52323784
   1.67634124  1.83420703]
 [ 0.0779399   0.11067827  0.15055585  0.19826855  0.25447682  0.31979126
   0.39475827  0.4798462   0.57543218  0.6817899   0.79907882  0.92733484
   1.0664628   1.21623106  1.37626807  1.54606141  1.72495902  1.91217285
   2.10678484  2.30775515]
 [ 0.09297057  0.13266999  0.18115763  0.23931211  0.30797147  0.38791517
   0.4798462   0.58437355  0.70199548  0.83308405  0.9778711   1.1364362
   1.30869671  1.49440036  1.69312041  1.90425368  2.12702148  2.36047346
   2.60349437  2.85481375]
 [ 0.10959913  0.15709119  0.21524211  0.28514259  0.36783774  0.46430502
   0.57543218  0.70199547  0.84463896  1.00385498  1.17996655  1.3731119
   1.58323175  1.81005935  2.05311389  2.31169724  2.58489428  2.87157684
   3.17041122  3.47986931]
 [ 0.12783414  0.18396714  0.25285951  0.33584534  0.43420843  0.54915561
   0.6817899   0.83308404  1.00385499  1.19473989  1.40617422  1.63837242
   1.8913117   2.16471922  2.4580632   2.77054805  3.10111383  3.44844011
   3.8109541   4.18684327]
 [ 0.14766482  0.21329429  0.2940195   0.39145097  0.50714468  0.64257025
   0.79907882  0.9778711   1.17996655  1.40617422  1.6570659   1.93295232
   2.23386284  2.55952928  2.90937424  3.2825042   3.67770782  4.09345937
   4.52792751  4.97898917]
 [ 0.16906026  0.24503882  0.33868887  0.45193177  0.58663086  0.74455387
   0.92733484  1.1364362   1.37311191  1.63837243  1.93295232  2.25728124
   2.61145904  2.9952355   3.4079954   3.84874917  4.31612959  4.80839457
```

```
  5.32343629  5.85879637]
[ 0.19196883  0.27913561  0.38678973  0.51719903  0.67257069  0.85500742
  1.06646281  1.30869671  1.58323175  1.89131171  2.23386284  2.61145904
  3.02429158  3.47214425  3.95437453  4.46990128  5.01719943  5.59430178
  6.19880822  6.8279021 ]
[ 0.21631805  0.31548767  0.43819852  0.5871013   0.76478454  0.97372404
  1.21623105  1.49440035  1.81005935  2.16471922  2.55952928  2.99523549
  3.47214424  3.99009216  4.54842281  5.14597084  5.78105421  6.45147462
  7.15452646  7.88701424]
[ 0.2420147   0.35396621  0.4927458   0.66142375  0.86300813  1.10038685
  1.37626807  1.6931204   2.05311389  2.4580632   2.90937424  3.40799539
  3.95437452  4.5484228   5.18948627  5.87632587  6.60710661  7.37939621
  8.19017347  9.03584644]
[ 0.26894534  0.39441116  0.55021665  0.73988847  0.9668926   1.23456852
  1.54606141  1.90425368  2.31169724  2.77054804  3.2825042   3.84874917
  4.46990128  5.14597084  5.87632587  6.65966738  7.49401489  8.3767028
  9.30438777 10.27306728]
[ 0.29697721  0.43663222  0.61035202  0.82215592  1.07600597  1.3757328
  1.72495901  2.12702148  2.58489428  3.10111383  3.67770782  4.31612958
  5.01719942  5.78105421  6.60710661  7.49401489  8.43966423  9.44116023
 10.49483482 11.59626488]
[ 0.32595938  0.48041053  0.67285076  0.90782743  1.18983614  1.52323783
  1.91217284  2.36047345  2.87157683  3.4484401   4.09345937  4.80839456
  5.59430177  6.45147462  7.37939621  8.3767028   9.44116023 10.56965395
 11.75819293 13.00192793]
[ 0.3557243   0.52550073  0.73737237  0.99644869  1.30779513  1.67634124
  2.10678484  2.60349437  3.17041122  3.8109541   4.52792751  5.32343629
  6.19880822  7.15452646  8.19017347  9.30438777 10.49483483 11.75819294
 13.09015463 14.48544399]
[ 0.38608956  0.57163351  0.80354049  1.08751422  1.42922473  1.83420703
  2.30775515  2.85481374  3.47986931  4.18684326  4.97898917  5.85879636
  6.8279021   7.88701424  9.03584644 10.27306728 11.59626488 13.00192793
 14.48544398 16.04111527]]
```

In [37]:
```
# def nll(params,X,Y):
#     length_scale = params[0] ## Account for length scale here
#     sig_n = params[1] ## Noise std. deviation
#     K = kernel(X, X, length_scale, sig_n) + np.diag(sig_n**2*np.ones(len
(X))) ## Note the change here
#     c = np.linalg.inv(np.linalg.cholesky(K))#+eps*(np.eye(len(X)))))
#     Ki = np.dot(c.T,c)
#     (sign, logdetK) = np.linalg.slogdet(K)
#     ll = -numObs/2 * np.log(np.dot(Y.T,(np.dot(Ki,Y)))) - 1/2 * logdetK
#     return -ll

# def gnll(params,X,Y):
#     length_scale = params[0] ## Account for length scale here
#     sig_n = params[1]
#     K = kernel(X, X, length_scale, sig_n) + np.diag(sig_n**2*np.ones(len
(X))) ## Note the change here
#     c = np.linalg.inv(np.linalg.cholesky(K))#+eps*(np.eye(numObs))))
#     Ki = np.dot(c.T,c)
#     KiY = np.dot(Ki,Y)
#     dotK = np.dot(K,DistMat)/(length_scale**2)

#     # Compute derivatives for both components separately
#     dll_ls = (numObs)/2 * np.dot(KiY.T,np.dot(dotK,KiY))/(np.dot(Y.T,KiY))\
#             - (1/2)*(np.sum(np.diag(np.dot(Ki,dotK)))) # for length scale
#     dll_n = (numObs)/2 * np.dot(KiY.T,KiY)/(np.dot(Y.T,KiY)) - 1/2*(np.sum(n
p.diag(Ki))) # for noise
#     return(np.concatenate((dll_ls, dll_n), axis=0))


# initial_guess = [(0.1,0.1*np.var(YScaled_noisy))]
# HP_bounds = ((eps, 10),(eps, np.var(YScaled_noisy)))


# DistMat = Sq_Euclid_DistMat(Xtrain,Xtrain)
# res = minimize(nll, initial_guess,args=(DistMat,YScaled_noisy), method="L-BF
GS-B",\
#                jac=gnll,bounds = HP_bounds,options={'maxiter':1000, 'gtol':
 1e-6, 'disp': True})

# print("Optimal length scale: ",res.x[0])
# print("Optimal Noise Variance: ",res.x[1])
# print("Optimal Negative Log-Likelihood: ",res.fun[0][0]) # log marginal like
lihood.
# print("Convergence Status: ",res.message)
# print("No. of Evaluations: ", res.nfev)
```