Programming Assignment 2: Linear regression

As we know, the COVID-19 pandemic has been disrupting people's life around the world. To get it under control, a crucial aspect is to be able to accurate forecast the spread of the disease, which can be helpful as a planning tool for policymakers, clinicians, and public health officers to deal with this crisis. In this notebook,we will try to do some forecasting on the covid-19 epidemic progression using machine learning. We will use a dataset based on the COVID-19 Data Repository at John Hopkins university.

In [99]:
```
import pandas as pd
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
from matplotlib.dates import DateFormatter
import matplotlib.dates as mdates
import numpy as np
from math import sqrt
import sys
```

In [100]:
```
print (sys.version)
```

```
3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD6
4)]
```

First, let us load the data.

In [101]:
```
data_orig = pd.read_csv("us_covid_data.csv")
data = data_orig.copy()
print (data_orig.columns)
data_orig
```

```
Index(['date', 'Country', 'hospitalized_with_symptom', 'Intensive_care',
       'Total_hospitalized', 'Home_Isolation', 'Total_positive',
       'Daily_change_in_positive_cases', 'New_positive_cases', 'Recovered',
       'Deaths', 'Total_cases', 'People_tested'],
      dtype='object')
```

Out[101]:

	date	Country	hospitalized_with_symptom	Intensive_care	Total_hospitalized	Home_I
0	2020-02-24T18:00:00	ITA	101	26	127	
1	2020-02-25T18:00:00	ITA	114	35	150	
2	2020-02-26T18:00:00	ITA	128	36	164	
3	2020-02-27T18:00:00	ITA	248	56	304	
4	2020-02-28T18:00:00	ITA	345	64	409	
...	...	...	...	...	...	
201	2020-09-12T17:00:00	ITA	1951	182	2133	
202	2020-09-13T17:00:00	ITA	2042	187	2229	
203	2020-09-14T17:00:00	ITA	2122	197	2319	
204	2020-09-15T17:00:00	ITA	2222	201	2423	
205	2020-09-16T17:00:00	ITA	2285	207	2492	

206 rows × 13 columns

In [102]:
```
# change the date format
dates = data['date']
date_format = [pd.to_datetime(d) for d in dates]
```

Data Visualization

Task P1: complete the following **three** visualization graphs that show the trend of the epidemic progression. Copy them to the solution file.

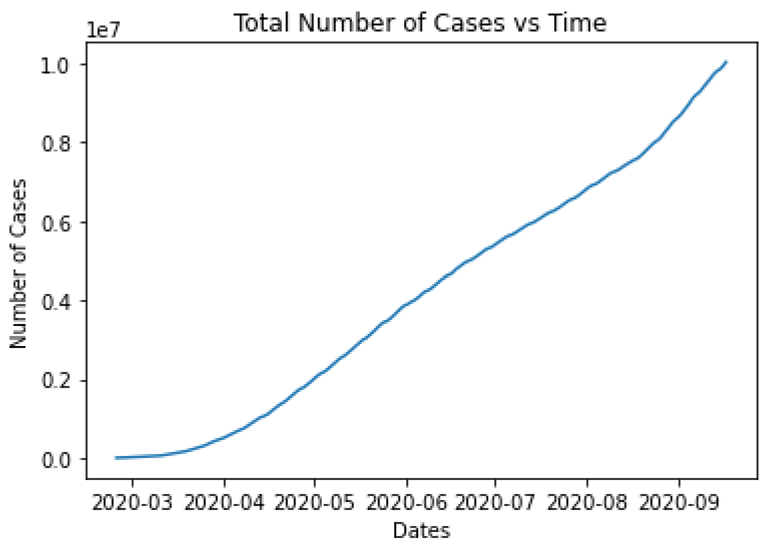
Graph 1: plot the total number of people tested for the entire period of the dataset. Your X axis will be the dates ("Dates") and Y-axis will be the total number of cases ("People_tested") over the period of time.

In [103]:
```
# Add code to plot the trend of the total number of people being tested as day
s progressed.
# X axis -> dates('Dates')
# Y axis -> number of people tested.('People_tested')

### STUDENT: Start of Code ###
plt.title("Total Number of Cases vs Time")
plt.xlabel("Dates")
plt.ylabel("Number of Cases")
People_tested = data['People_tested']

plt.plot(date_format, People_tested)

plt.show()
### End of code ####
```



Graph 2: plot the total number of deaths for the entire period. Your X axis will be the dates ("Dates") and Y-axis will be the total number of death cases("Deaths") over the period of time.

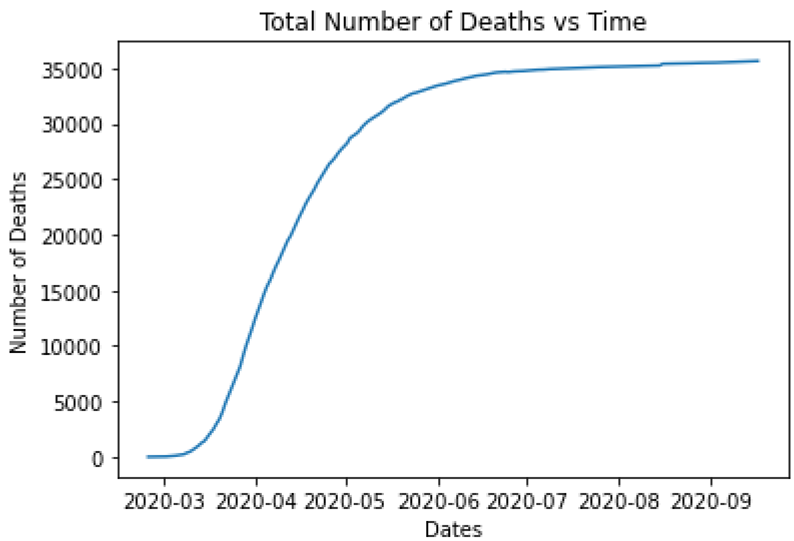
In [104]:
```
# Add code to plot the trend of total deaths as days progressed.
# X axis -> dates ('Dates')
# Y axis -> number of deaths ('Deaths')


### STUDENT: Start of Code ###
plt.title("Total Number of Deaths vs Time")
plt.xlabel("Dates")
plt.ylabel("Number of Deaths")
Deaths = data['Deaths']

plt.plot(date_format, Deaths)

plt.show()

### End of code ####
```



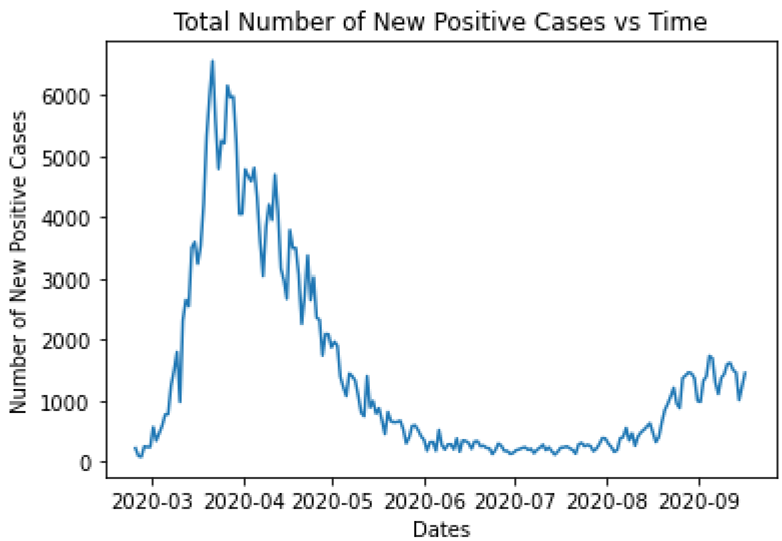
Graph 3: plot the total number of infected cases for the entire period. Your X axis will be the dates ("Dates") and Y-axis will be the total number of infected cases ('New_positive_cases') over the period of time.

In [105]:
```
### STUDENT: Start of Code ###
plt.title("Total Number of New Positive Cases vs Time")
plt.xlabel("Dates")
plt.ylabel("Number of New Positive Cases")
New_positive_cases = data['New_positive_cases']

plt.plot(date_format, New_positive_cases)

plt.show()


### End of code ####
```



As we can see that our data has different ranges of values for every feature and this can cause problems in our model, so here we will normalize our data (ignoring the categorical variables) so that our data is scaled between 0 and 1. The downside, however, is that the numbers are no longer interpretable. To interpret it, you need to multiply back by the scaling factor.

IMPORTANT: From now on, we will work with the normalized features to build the regression model. However, in **Task P8**, you need to convert the number back to the actual units.

In [106]:
```
data_list = data_orig.columns.values.tolist()

for i in data_list[-11:]:
    data[[i]]=(data_orig[i]-data_orig[i].min())/(data_orig[i].max()-data_orig[
i].min())

data
```

Out[106]:

	date	Country	hospitalized_with_symptom	Intensive_care	Total_hospitalized	Home_I
0	2020-02-24T18:00:00	ITA	0.000000	0.000000	0.000000	(
1	2020-02-25T18:00:00	ITA	0.000450	0.002227	0.000700	(
2	2020-02-26T18:00:00	ITA	0.000934	0.002474	0.001125	(
3	2020-02-27T18:00:00	ITA	0.005085	0.007422	0.005384	(
4	2020-02-28T18:00:00	ITA	0.008440	0.009401	0.008577	(
...	...	...	...	...	...	
201	2020-09-12T17:00:00	ITA	0.063994	0.038595	0.061015	(
202	2020-09-13T17:00:00	ITA	0.067142	0.039832	0.063935	(
203	2020-09-14T17:00:00	ITA	0.069909	0.042306	0.066673	(
204	2020-09-15T17:00:00	ITA	0.073368	0.043295	0.069836	(
205	2020-09-16T17:00:00	ITA	0.075547	0.044780	0.071935	(

206 rows × 13 columns

Calculate the feature matrix

The following is a function that accepts a list of feature names (e.g. ['Total_Hospitalized', 'People_tested']) and an target feature e.g. ('Deaths') and returns two things:

1. A numpy matrix whose columns are the desired features plus a column with a constant value 1, which is also known as the 'intercept'.
2. A numpy array that contains the values of the target output.

In [107]:
```
def get_numpy_data(data_frame, features, output):
    # Steps
    # select the columns of data_Frame given by the features list into the var
iable features_sframe which will include the constant
    # Convert the features_frame into a numpy matrix
    # assign the column of data_frame associated with the output to the array
 output_array
    # convert the array into a numpy array by first converting it to a list
    # return feature_matrix,output_array

    data_frame['constant'] = 1 # here we are adding a constant column
    # add the column 'constant' to the front of the features list.
    features = ['constant'] + features

    # select the columns of data_Frame given by the features list into the var
iable features_sframe which will include the constant)
    features_frame = data_frame[features]

    # Convert the features_frame into a numpy matrix
    feature_matrix = features_frame.to_numpy()
    # print ("feature_matrix:", feature_matrix)

    # assign the column of data_frame associated with the output to the array
 output_array
    output_array = data_frame[output]

    # convert the array into a numpy array by first converting it to a list
    output_array = output_array.to_numpy()

    return(feature_matrix, output_array)
```

For dates, we need to convert them into a sequence of numbers. We now add a new column to our dataframe corresponding to the number of days since the start of the dataset.

In [108]:
```
X = date_format
day_numbers = []
for i in range(1, len(X) + 1):
    day_numbers.append([i])

data['Days'] = pd.DataFrame(day_numbers,columns = ['Days'])
data["Days"] = data["Days"].astype(int)
```

Test the above function for a particular input and output feature.

In [109]:
```
(example_features, example_output) = get_numpy_data(data, ['Days'], 'New_posit
ive_cases')
print (example_features[0,:])
print (example_output[0])
```

```
[1 1]
0.022071307300509338
```

Predict the outputs with given regression weights

Suppose we had the weights $[1, 1]$ corresponding to the features $[1, 100]$, to compute the predicted output, we can simply take the dot product between them, so the output is $1 * 1 + 1 * 100 = 101$. Now, let's create the data with

In [110]:
```
(test_features, output) = get_numpy_data(data, ['Days'], 'People_tested')
```

Task P2: Complete the following function 'predict_output'. Copy the the outputs of the code to the solution file.

In [111]:
```
def predict_output(feature_matrix, weights):
    # Inputs:
    # feature_matrix: a numpy matrix containing the features as columns (inclu
ding the intercept),
    #                 and each row corresponds to a data point
    # weights: a numpy array for the corresponding regression weights (includi
ng the intercept)
    # Output:
    # a numpy array that contains the predicted outputs (according to the prov
ided weights)
    # for all the data points in the feature_matrix

    # STUDENT: Start of code ####
    features = np.array(feature_matrix)
    regress_weights = np.array(weights)

    return np.dot(features, regress_weights)
    ## end of code
```

In [112]:
```
# Copy the outputs of this code to the solution file
my_weights = np.array([1., 1.])
test_predictions = predict_output(example_features, my_weights)
print ("(normalized) prediction at day 5: ", test_predictions[5])
print ("(normalized) prediction at day 20 ", test_predictions[20])
```

```
(normalized) prediction at day 5:  7.0
(normalized) prediction at day 20  22.0
```

Compute the derivative

We will now compute the derivative of the regression cost function:

$$L_D(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - w \cdot x_i)^2,$$

where $x_i \in \mathrm{R}^d$ is the input feature of dimension d, $y_i \in \mathrm{R}$ is the output response, and $w \in \mathrm{R}^d$ is the regression weights.

Task P3: Complete the function 'weight_derivative' to calculate the derivative of the cost function with respect to regression weights w, i.e., $\frac{\partial}{\partial w} L_D(w)$. Note that this should be a d dimensional vector. Also copy the output of the code for the test example to the solution file.

In [113]:
```
def weight_derivative(weights, feature_matrix, labels):
    # Input:
    # weights: weight vector w, a numpy vector of dimension d
    # feature_matrix: numpy array of size n by d, where n is the number of dat
a points, and d is the feature dimension
    # labels: true labels y, a numpy vector of dimension d
    # Output:
    # Derivative of the regression cost function with respect to the weight w,
a numpy array of dimension d

    ## STUDENT: Start of code ###

    error = np.subtract( labels, np.dot(feature_matrix, weights ) )
    derivative = -2/len(labels)*np.dot(np.transpose(feature_matrix), error)

#     print("Size of Weights:", weights.shape )
#     print("Size of features:", feature_matrix.shape )
#     print("Size of Labels:", labels.shape )

    return derivative
    # End of code ###
```

In [114]:
```
# NOTE: copy the output to the solution file.

(example_features, example_output) = get_numpy_data(data, ['Days'], 'People_te
sted')
my_weights = np.array([0., 0.]) # this makes all the predictions 0
derivative = weight_derivative(my_weights, example_features,example_output)

print (derivative)
```

```
[  -0.82103242 -120.60087518]
```

Gradient descent algorithm

Here, we will write a function to perform gradient descent algorithm on the lineare regression cost. Given an initial point, we will update the current weights by moving in the negative gradient direction to minimize the cost function. Thus, in each iteration we obtain the updated weight w_{t+1} from the current iterate w_t as follows:

$$w_{t+1} = w_t - h\frac{\partial}{\partial w}L_D(w_t),$$

where h is the 'step_size' that is the amount by which we move in the negative gradient direction.

We stop when we are sufficiently close to the optimum (where gradient is the zero vector) by checking the condition with respect to the magnitude (length) of the gradient vector:

$$\|\frac{\partial}{\partial w}L_D(w_t)\|_2 \leq \epsilon,$$

where ϵ is the 'tolerance' parameter.

Task P4: Complete the code section to perform the gradient decent in the function `regression_gradient_descent`. Copy the code to the solution file.

In [115]:

```
def regression_gradient_descent(feature_matrix, labels, initial_weights, step_
size, tolerance):
    # Gradient descent algorithm for linear regression problem

    # Input:
    # feature_matrix: numpy array of size n by d, where n is the number of dat
a points, and d is the feature dimension
    # labels: true labels y, a numpy vector of dimension d
    # initial_weights: initial weight vector to start with, a numpy vector of
 dimension d
    # step_size: step size of update
    # tolerance: tolerace epsilon for stopping condition
    # Output:
    # Weights obtained after convergence

    converged = False
    weights = np.array(initial_weights) # current iterate
    i = 0
    while not converged:
        i += 1
        # STUDENT: Start of code: your impelementation of what the gradient de
scent algorithm does in every iteration
        # Refer back to the update rule listed above: update the weight
        derivative = weight_derivative(weights, feature_matrix, labels);
        weights = weights - step_size*derivative

        # Compute the gradient magnitude:

        gradient_magnitude = np.linalg.norm( weight_derivative(weights, featur
e_matrix, labels) )

        # Check the stopping condition to decide whether you want to stop the
 iterations
        if  gradient_magnitude <= tolerance:   # STUDENT: check the stopping c
ondition here
            converged = True

        # End of code

        print ("Iteration: ",i,"gradient_magnitude: ", gradient_magnitude) # f
or us to check about convergence

    return(weights)
```

Use gradient descent for linear regression

Let's test the gradient descent algorithm for linear regression with a single feature ('Day'). Here we are using first 180 days' data as our training data.

In [116]:

```
#train_data
train_data = data[:180]
```

Task P5: Specify the initial_weights, step_size, and tolerance for the function `regression_gradient_descent` . Copy the outputs of the code to the solution file.

In [117]:

```
simple_features = ['Days']
my_output = 'People_tested'

# Use get_numpy_data method to calculate the feature matrix and output.
(simple_feature_matrix, output) = get_numpy_data(train_data, simple_features,
my_output)

#Initialize the weights, step size and tolerance
# Start of code
#STUDENT: Specify the initial_weights, step_size, and tolerance
initial_weights = [1.0, 1.0]
step_size = 7e-8
tolerance = 1.2e4
# end of code

# Use the regression_gradient_descent function to calculate the gradient decen
t and store it in the variable 'final_weights'
final_weights = regression_gradient_descent(simple_feature_matrix, output, ini
tial_weights, step_size, tolerance)

# end of code
print ("Here are the final weights after convergence:")
print (final_weights)
```

```
<ipython-input-107-8b2916f5b454>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_frame['constant'] = 1 # here we are adding a constant column
```

```
Iteration:  1 gradient_magnitude:  21840.81707866705
Iteration:  2 gradient_magnitude:  21807.51575965956
Iteration:  3 gradient_magnitude:  21774.265216126485
Iteration:  4 gradient_magnitude:  21741.06537064902
Iteration:  5 gradient_magnitude:  21707.91614592638
Iteration:  6 gradient_magnitude:  21674.81746477566
Iteration:  7 gradient_magnitude:  21641.769250131623
Iteration:  8 gradient_magnitude:  21608.77142504654
Iteration:  9 gradient_magnitude:  21575.82391269002
Iteration:  10 gradient_magnitude:  21542.92663634879
Iteration:  11 gradient_magnitude:  21510.079519426563
Iteration:  12 gradient_magnitude:  21477.282485443844
Iteration:  13 gradient_magnitude:  21444.53545803773
Iteration:  14 gradient_magnitude:  21411.838360961778
Iteration:  15 gradient_magnitude:  21379.191118085775
Iteration:  16 gradient_magnitude:  21346.593653395597
Iteration:  17 gradient_magnitude:  21314.045890993017
Iteration:  18 gradient_magnitude:  21281.547755095533
Iteration:  19 gradient_magnitude:  21249.099170036196
Iteration:  20 gradient_magnitude:  21216.70006026343
Iteration:  21 gradient_magnitude:  21184.350350340832
Iteration:  22 gradient_magnitude:  21152.04996494705
Iteration:  23 gradient_magnitude:  21119.79882887556
Iteration:  24 gradient_magnitude:  21087.59686703451
Iteration:  25 gradient_magnitude:  21055.444004446545
Iteration:  26 gradient_magnitude:  21023.340166248632
Iteration:  27 gradient_magnitude:  20991.28527769187
Iteration:  28 gradient_magnitude:  20959.279264141343
Iteration:  29 gradient_magnitude:  20927.32205107593
Iteration:  30 gradient_magnitude:  20895.41356408814
Iteration:  31 gradient_magnitude:  20863.55372888393
Iteration:  32 gradient_magnitude:  20831.74247128252
Iteration:  33 gradient_magnitude:  20799.979717216258
Iteration:  34 gradient_magnitude:  20768.26539273041
Iteration:  35 gradient_magnitude:  20736.599423983014
Iteration:  36 gradient_magnitude:  20704.98173724469
Iteration:  37 gradient_magnitude:  20673.412258898476
Iteration:  38 gradient_magnitude:  20641.89091543965
Iteration:  39 gradient_magnitude:  20610.417633475583
Iteration:  40 gradient_magnitude:  20578.99233972552
Iteration:  41 gradient_magnitude:  20547.614961020485
Iteration:  42 gradient_magnitude:  20516.28542430301
Iteration:  43 gradient_magnitude:  20485.003656627076
Iteration:  44 gradient_magnitude:  20453.769585157825
Iteration:  45 gradient_magnitude:  20422.583137171507
Iteration:  46 gradient_magnitude:  20391.44424005523
Iteration:  47 gradient_magnitude:  20360.352821306817
Iteration:  48 gradient_magnitude:  20329.308808534643
Iteration:  49 gradient_magnitude:  20298.312129457467
Iteration:  50 gradient_magnitude:  20267.36271190423
Iteration:  51 gradient_magnitude:  20236.46048381396
Iteration:  52 gradient_magnitude:  20205.605373235514
Iteration:  53 gradient_magnitude:  20174.797308327477
Iteration:  54 gradient_magnitude:  20144.036217357967
Iteration:  55 gradient_magnitude:  20113.322028704482
Iteration:  56 gradient_magnitude:  20082.654670853706
Iteration:  57 gradient_magnitude:  20052.03407240138
```

```
Iteration:  58 gradient_magnitude:  20021.46016205212
Iteration:  59 gradient_magnitude:  19990.932868619224
Iteration:  60 gradient_magnitude:  19960.45212102455
Iteration:  61 gradient_magnitude:  19930.017848298332
Iteration:  62 gradient_magnitude:  19899.629979579015
Iteration:  63 gradient_magnitude:  19869.28844411305
Iteration:  64 gradient_magnitude:  19838.993171254828
Iteration:  65 gradient_magnitude:  19808.744090466414
Iteration:  66 gradient_magnitude:  19778.54113131744
Iteration:  67 gradient_magnitude:  19748.38422348492
Iteration:  68 gradient_magnitude:  19718.273296753105
Iteration:  69 gradient_magnitude:  19688.208281013285
Iteration:  70 gradient_magnitude:  19658.189106263653
Iteration:  71 gradient_magnitude:  19628.215702609137
Iteration:  72 gradient_magnitude:  19598.28800026125
Iteration:  73 gradient_magnitude:  19568.405929537894
Iteration:  74 gradient_magnitude:  19538.56942086323
Iteration:  75 gradient_magnitude:  19508.7784047675
Iteration:  76 gradient_magnitude:  19479.03281188685
Iteration:  77 gradient_magnitude:  19449.33257296321
Iteration:  78 gradient_magnitude:  19419.677618844125
Iteration:  79 gradient_magnitude:  19390.06788048253
Iteration:  80 gradient_magnitude:  19360.503288936685
Iteration:  81 gradient_magnitude:  19330.98377536993
Iteration:  82 gradient_magnitude:  19301.50927105059
Iteration:  83 gradient_magnitude:  19272.07970735178
Iteration:  84 gradient_magnitude:  19242.69501575124
Iteration:  85 gradient_magnitude:  19213.355127831197
Iteration:  86 gradient_magnitude:  19184.059975278215
Iteration:  87 gradient_magnitude:  19154.809489882977
Iteration:  88 gradient_magnitude:  19125.603603540203
Iteration:  89 gradient_magnitude:  19096.442248248433
Iteration:  90 gradient_magnitude:  19067.325356109905
Iteration:  91 gradient_magnitude:  19038.25285933037
Iteration:  92 gradient_magnitude:  19009.224690218958
Iteration:  93 gradient_magnitude:  18980.240781187997
Iteration:  94 gradient_magnitude:  18951.301064752897
Iteration:  95 gradient_magnitude:  18922.405473531922
Iteration:  96 gradient_magnitude:  18893.55394024609
Iteration:  97 gradient_magnitude:  18864.74639771902
Iteration:  98 gradient_magnitude:  18835.982778876747
Iteration:  99 gradient_magnitude:  18807.26301674756
Iteration:  100 gradient_magnitude:  18778.58704446187
Iteration:  101 gradient_magnitude:  18749.954795252062
Iteration:  102 gradient_magnitude:  18721.366202452296
Iteration:  103 gradient_magnitude:  18692.821199498412
Iteration:  104 gradient_magnitude:  18664.319719927713
Iteration:  105 gradient_magnitude:  18635.861697378845
Iteration:  106 gradient_magnitude:  18607.447065591656
Iteration:  107 gradient_magnitude:  18579.075758407
Iteration:  108 gradient_magnitude:  18550.747709766623
Iteration:  109 gradient_magnitude:  18522.462853712976
Iteration:  110 gradient_magnitude:  18494.221124389092
Iteration:  111 gradient_magnitude:  18466.022456038405
Iteration:  112 gradient_magnitude:  18437.866783004625
Iteration:  113 gradient_magnitude:  18409.754039731557
Iteration:  114 gradient_magnitude:  18381.684160762976
```

```
Iteration:  115 gradient_magnitude:  18353.657080742418
Iteration:  116 gradient_magnitude:  18325.672734413143
Iteration:  117 gradient_magnitude:  18297.731056617842
Iteration:  118 gradient_magnitude:  18269.83198229859
Iteration:  119 gradient_magnitude:  18241.975446496646
Iteration:  120 gradient_magnitude:  18214.161384352312
Iteration:  121 gradient_magnitude:  18186.389731104784
Iteration:  122 gradient_magnitude:  18158.660422092016
Iteration:  123 gradient_magnitude:  18130.97339275053
Iteration:  124 gradient_magnitude:  18103.328578615303
Iteration:  125 gradient_magnitude:  18075.7259153196
Iteration:  126 gradient_magnitude:  18048.165338594838
Iteration:  127 gradient_magnitude:  18020.646784270404
Iteration:  128 gradient_magnitude:  17993.17018827355
Iteration:  129 gradient_magnitude:  17965.735486629208
Iteration:  130 gradient_magnitude:  17938.34261545987
Iteration:  131 gradient_magnitude:  17910.991510985405
Iteration:  132 gradient_magnitude:  17883.68210952294
Iteration:  133 gradient_magnitude:  17856.414347486694
Iteration:  134 gradient_magnitude:  17829.18816138785
Iteration:  135 gradient_magnitude:  17802.00348783438
Iteration:  136 gradient_magnitude:  17774.86026353092
Iteration:  137 gradient_magnitude:  17747.75842527861
Iteration:  138 gradient_magnitude:  17720.69790997496
Iteration:  139 gradient_magnitude:  17693.678654613675
Iteration:  140 gradient_magnitude:  17666.700596284554
Iteration:  141 gradient_magnitude:  17639.763672173285
Iteration:  142 gradient_magnitude:  17612.867819561372
Iteration:  143 gradient_magnitude:  17586.012975825906
Iteration:  144 gradient_magnitude:  17559.199078439487
Iteration:  145 gradient_magnitude:  17532.426064970034
Iteration:  146 gradient_magnitude:  17505.693873080683
Iteration:  147 gradient_magnitude:  17479.0024405296
Iteration:  148 gradient_magnitude:  17452.351705169844
Iteration:  149 gradient_magnitude:  17425.741604949257
Iteration:  150 gradient_magnitude:  17399.17207791026
Iteration:  151 gradient_magnitude:  17372.643062189785
Iteration:  152 gradient_magnitude:  17346.15449601905
Iteration:  153 gradient_magnitude:  17319.706317723474
Iteration:  154 gradient_magnitude:  17293.298465722502
Iteration:  155 gradient_magnitude:  17266.93087852948
Iteration:  156 gradient_magnitude:  17240.60349475151
Iteration:  157 gradient_magnitude:  17214.31625308929
Iteration:  158 gradient_magnitude:  17188.06909233698
Iteration:  159 gradient_magnitude:  17161.86195138207
Iteration:  160 gradient_magnitude:  17135.694769205253
Iteration:  161 gradient_magnitude:  17109.567484880205
Iteration:  162 gradient_magnitude:  17083.480037573543
Iteration:  163 gradient_magnitude:  17057.432366544625
Iteration:  164 gradient_magnitude:  17031.42441114542
Iteration:  165 gradient_magnitude:  17005.45611082037
Iteration:  166 gradient_magnitude:  16979.527405106255
Iteration:  167 gradient_magnitude:  16953.63823363203
Iteration:  168 gradient_magnitude:  16927.78853611871
Iteration:  169 gradient_magnitude:  16901.978252379216
Iteration:  170 gradient_magnitude:  16876.207322318245
Iteration:  171 gradient_magnitude:  16850.47568593211
```

```
Iteration:  172 gradient_magnitude:  16824.78328330862
Iteration:  173 gradient_magnitude:  16799.130054626938
Iteration:  174 gradient_magnitude:  16773.51594015744
Iteration:  175 gradient_magnitude:  16747.940880261543
Iteration:  176 gradient_magnitude:  16722.404815391645
Iteration:  177 gradient_magnitude:  16696.9076860909
Iteration:  178 gradient_magnitude:  16671.449432993144
Iteration:  179 gradient_magnitude:  16646.02999682271
Iteration:  180 gradient_magnitude:  16620.649318394317
Iteration:  181 gradient_magnitude:  16595.30733861292
Iteration:  182 gradient_magnitude:  16570.00399847359
Iteration:  183 gradient_magnitude:  16544.73923906137
Iteration:  184 gradient_magnitude:  16519.513001551102
Iteration:  185 gradient_magnitude:  16494.32522720736
Iteration:  186 gradient_magnitude:  16469.175857384234
Iteration:  187 gradient_magnitude:  16444.064833525255
Iteration:  188 gradient_magnitude:  16418.992097163245
Iteration:  189 gradient_magnitude:  16393.957589920148
Iteration:  190 gradient_magnitude:  16368.961253506948
Iteration:  191 gradient_magnitude:  16344.003029723479
Iteration:  192 gradient_magnitude:  16319.082860458315
Iteration:  193 gradient_magnitude:  16294.200687688655
Iteration:  194 gradient_magnitude:  16269.356453480146
Iteration:  195 gradient_magnitude:  16244.550099986782
Iteration:  196 gradient_magnitude:  16219.781569450759
Iteration:  197 gradient_magnitude:  16195.050804202318
Iteration:  198 gradient_magnitude:  16170.35774665966
Iteration:  199 gradient_magnitude:  16145.702339328747
Iteration:  200 gradient_magnitude:  16121.08452480324
Iteration:  201 gradient_magnitude:  16096.504245764301
Iteration:  202 gradient_magnitude:  16071.961444980509
Iteration:  203 gradient_magnitude:  16047.45606530769
Iteration:  204 gradient_magnitude:  16022.988049688804
Iteration:  205 gradient_magnitude:  15998.557341153812
Iteration:  206 gradient_magnitude:  15974.163882819536
Iteration:  207 gradient_magnitude:  15949.807617889523
Iteration:  208 gradient_magnitude:  15925.48848965393
Iteration:  209 gradient_magnitude:  15901.206441489368
Iteration:  210 gradient_magnitude:  15876.961416858809
Iteration:  211 gradient_magnitude:  15852.753359311393
Iteration:  212 gradient_magnitude:  15828.582212482359
Iteration:  213 gradient_magnitude:  15804.447920092867
Iteration:  214 gradient_magnitude:  15780.35042594992
Iteration:  215 gradient_magnitude:  15756.289673946161
Iteration:  216 gradient_magnitude:  15732.2656080598
Iteration:  217 gradient_magnitude:  15708.278172354467
Iteration:  218 gradient_magnitude:  15684.32731097907
Iteration:  219 gradient_magnitude:  15660.412968167697
Iteration:  220 gradient_magnitude:  15636.53508823943
Iteration:  221 gradient_magnitude:  15612.693615598278
Iteration:  222 gradient_magnitude:  15588.88849473301
Iteration:  223 gradient_magnitude:  15565.119670217022
Iteration:  224 gradient_magnitude:  15541.387086708244
Iteration:  225 gradient_magnitude:  15517.69068894896
Iteration:  226 gradient_magnitude:  15494.030421765736
Iteration:  227 gradient_magnitude:  15470.406230069242
Iteration:  228 gradient_magnitude:  15446.818058854153
```

```
Iteration:  229 gradient_magnitude:  15423.265853199007
Iteration:  230 gradient_magnitude:  15399.749558266089
Iteration:  231 gradient_magnitude:  15376.269119301289
Iteration:  232 gradient_magnitude:  15352.824481633985
Iteration:  233 gradient_magnitude:  15329.415590676917
Iteration:  234 gradient_magnitude:  15306.042391926057
Iteration:  235 gradient_magnitude:  15282.704830960472
Iteration:  236 gradient_magnitude:  15259.402853442207
Iteration:  237 gradient_magnitude:  15236.136405116169
Iteration:  238 gradient_magnitude:  15212.905431809966
Iteration:  239 gradient_magnitude:  15189.709879433845
Iteration:  240 gradient_magnitude:  15166.549693980482
Iteration:  241 gradient_magnitude:  15143.424821524923
Iteration:  242 gradient_magnitude:  15120.335208224422
Iteration:  243 gradient_magnitude:  15097.280800318347
Iteration:  244 gradient_magnitude:  15074.261544128023
Iteration:  245 gradient_magnitude:  15051.277386056612
Iteration:  246 gradient_magnitude:  15028.328272589026
Iteration:  247 gradient_magnitude:  15005.41415029174
Iteration:  248 gradient_magnitude:  14982.534965812718
Iteration:  249 gradient_magnitude:  14959.690665881262
Iteration:  250 gradient_magnitude:  14936.881197307921
Iteration:  251 gradient_magnitude:  14914.106506984313
Iteration:  252 gradient_magnitude:  14891.366541883053
Iteration:  253 gradient_magnitude:  14868.66124905759
Iteration:  254 gradient_magnitude:  14845.99057564212
Iteration:  255 gradient_magnitude:  14823.354468851443
Iteration:  256 gradient_magnitude:  14800.752875980828
Iteration:  257 gradient_magnitude:  14778.185744405911
Iteration:  258 gradient_magnitude:  14755.653021582573
Iteration:  259 gradient_magnitude:  14733.154655046808
Iteration:  260 gradient_magnitude:  14710.690592414594
Iteration:  261 gradient_magnitude:  14688.260781381796
Iteration:  262 gradient_magnitude:  14665.865169724017
Iteration:  263 gradient_magnitude:  14643.503705296482
Iteration:  264 gradient_magnitude:  14621.17633603394
Iteration:  265 gradient_magnitude:  14598.883009950518
Iteration:  266 gradient_magnitude:  14576.623675139603
Iteration:  267 gradient_magnitude:  14554.39827977373
Iteration:  268 gradient_magnitude:  14532.206772104455
Iteration:  269 gradient_magnitude:  14510.049100462236
Iteration:  270 gradient_magnitude:  14487.925213256318
Iteration:  271 gradient_magnitude:  14465.835058974593
Iteration:  272 gradient_magnitude:  14443.778586183527
Iteration:  273 gradient_magnitude:  14421.75574352797
Iteration:  274 gradient_magnitude:  14399.766479731088
Iteration:  275 gradient_magnitude:  14377.810743594251
Iteration:  276 gradient_magnitude:  14355.888483996861
Iteration:  277 gradient_magnitude:  14333.999649896292
Iteration:  278 gradient_magnitude:  14312.144190327723
Iteration:  279 gradient_magnitude:  14290.32205440405
Iteration:  280 gradient_magnitude:  14268.533191315772
Iteration:  281 gradient_magnitude:  14246.777550330822
Iteration:  282 gradient_magnitude:  14225.055080794524
Iteration:  283 gradient_magnitude:  14203.365732129414
Iteration:  284 gradient_magnitude:  14181.709453835161
Iteration:  285 gradient_magnitude:  14160.086195488404
```

```
Iteration:  286 gradient_magnitude:  14138.495906742706
Iteration:  287 gradient_magnitude:  14116.938537328359
Iteration:  288 gradient_magnitude:  14095.414037052322
Iteration:  289 gradient_magnitude:  14073.922355798077
Iteration:  290 gradient_magnitude:  14052.46344352552
Iteration:  291 gradient_magnitude:  14031.03725027085
Iteration:  292 gradient_magnitude:  14009.643726146443
Iteration:  293 gradient_magnitude:  13988.28282134074
Iteration:  294 gradient_magnitude:  13966.954486118135
Iteration:  295 gradient_magnitude:  13945.658670818848
Iteration:  296 gradient_magnitude:  13924.39532585882
Iteration:  297 gradient_magnitude:  13903.164401729602
Iteration:  298 gradient_magnitude:  13881.965848998223
Iteration:  299 gradient_magnitude:  13860.79961830708
Iteration:  300 gradient_magnitude:  13839.66566037384
Iteration:  301 gradient_magnitude:  13818.563925991308
Iteration:  302 gradient_magnitude:  13797.494366027302
Iteration:  303 gradient_magnitude:  13776.456931424578
Iteration:  304 gradient_magnitude:  13755.451573200671
Iteration:  305 gradient_magnitude:  13734.478242447809
Iteration:  306 gradient_magnitude:  13713.536890332789
Iteration:  307 gradient_magnitude:  13692.627468096869
Iteration:  308 gradient_magnitude:  13671.74992705565
Iteration:  309 gradient_magnitude:  13650.904218598953
Iteration:  310 gradient_magnitude:  13630.090294190737
Iteration:  311 gradient_magnitude:  13609.30810536894
Iteration:  312 gradient_magnitude:  13588.557603745418
Iteration:  313 gradient_magnitude:  13567.838741005793
Iteration:  314 gradient_magnitude:  13547.151468909335
Iteration:  315 gradient_magnitude:  13526.495739288908
Iteration:  316 gradient_magnitude:  13505.871504050783
Iteration:  317 gradient_magnitude:  13485.278715174582
Iteration:  318 gradient_magnitude:  13464.71732471313
Iteration:  319 gradient_magnitude:  13444.187284792366
Iteration:  320 gradient_magnitude:  13423.688547611224
Iteration:  321 gradient_magnitude:  13403.221065441525
Iteration:  322 gradient_magnitude:  13382.784790627848
Iteration:  323 gradient_magnitude:  13362.379675587461
Iteration:  324 gradient_magnitude:  13342.005672810148
Iteration:  325 gradient_magnitude:  13321.66273485816
Iteration:  326 gradient_magnitude:  13301.35081436607
Iteration:  327 gradient_magnitude:  13281.069864040664
Iteration:  328 gradient_magnitude:  13260.819836660852
Iteration:  329 gradient_magnitude:  13240.600685077521
Iteration:  330 gradient_magnitude:  13220.412362213463
Iteration:  331 gradient_magnitude:  13200.254821063256
Iteration:  332 gradient_magnitude:  13180.128014693135
Iteration:  333 gradient_magnitude:  13160.031896240891
Iteration:  334 gradient_magnitude:  13139.966418915794
Iteration:  335 gradient_magnitude:  13119.931535998428
Iteration:  336 gradient_magnitude:  13099.927200840631
Iteration:  337 gradient_magnitude:  13079.953366865357
Iteration:  338 gradient_magnitude:  13060.009987566575
Iteration:  339 gradient_magnitude:  13040.09701650918
Iteration:  340 gradient_magnitude:  13020.21440732885
Iteration:  341 gradient_magnitude:  13000.362113731955
Iteration:  342 gradient_magnitude:  12980.540089495467
```

```
Iteration:  343 gradient_magnitude:  12960.748288466819
Iteration:  344 gradient_magnitude:  12940.98666456382
Iteration:  345 gradient_magnitude:  12921.255171774554
Iteration:  346 gradient_magnitude:  12901.553764157237
Iteration:  347 gradient_magnitude:  12881.882395840155
Iteration:  348 gradient_magnitude:  12862.24102102152
Iteration:  349 gradient_magnitude:  12842.629593969388
Iteration:  350 gradient_magnitude:  12823.04806902155
Iteration:  351 gradient_magnitude:  12803.496400585398
Iteration:  352 gradient_magnitude:  12783.974543137867
Iteration:  353 gradient_magnitude:  12764.48245122528
Iteration:  354 gradient_magnitude:  12745.020079463275
Iteration:  355 gradient_magnitude:  12725.587382536696
Iteration:  356 gradient_magnitude:  12706.184315199462
Iteration:  357 gradient_magnitude:  12686.810832274497
Iteration:  358 gradient_magnitude:  12667.466888653595
Iteration:  359 gradient_magnitude:  12648.152439297344
Iteration:  360 gradient_magnitude:  12628.867439234982
Iteration:  361 gradient_magnitude:  12609.611843564328
Iteration:  362 gradient_magnitude:  12590.385607451675
Iteration:  363 gradient_magnitude:  12571.188686131654
Iteration:  364 gradient_magnitude:  12552.02103490717
Iteration:  365 gradient_magnitude:  12532.882609149267
Iteration:  366 gradient_magnitude:  12513.773364297036
Iteration:  367 gradient_magnitude:  12494.69325585752
Iteration:  368 gradient_magnitude:  12475.642239405594
Iteration:  369 gradient_magnitude:  12456.620270583873
Iteration:  370 gradient_magnitude:  12437.627305102602
Iteration:  371 gradient_magnitude:  12418.663298739562
Iteration:  372 gradient_magnitude:  12399.728207339955
Iteration:  373 gradient_magnitude:  12380.821986816301
Iteration:  374 gradient_magnitude:  12361.944593148355
Iteration:  375 gradient_magnitude:  12343.095982382998
Iteration:  376 gradient_magnitude:  12324.276110634088
Iteration:  377 gradient_magnitude:  12305.484934082435
Iteration:  378 gradient_magnitude:  12286.72240897566
Iteration:  379 gradient_magnitude:  12267.988491628072
Iteration:  380 gradient_magnitude:  12249.283138420615
Iteration:  381 gradient_magnitude:  12230.606305800718
Iteration:  382 gradient_magnitude:  12211.957950282225
Iteration:  383 gradient_magnitude:  12193.338028445283
Iteration:  384 gradient_magnitude:  12174.746496936252
Iteration:  385 gradient_magnitude:  12156.183312467574
Iteration:  386 gradient_magnitude:  12137.648431817715
Iteration:  387 gradient_magnitude:  12119.141811831036
Iteration:  388 gradient_magnitude:  12100.663409417679
Iteration:  389 gradient_magnitude:  12082.21318155351
Iteration:  390 gradient_magnitude:  12063.791085279998
Iteration:  391 gradient_magnitude:  12045.397077704087
Iteration:  392 gradient_magnitude:  12027.031115998143
Iteration:  393 gradient_magnitude:  12008.693157399823
Iteration:  394 gradient_magnitude:  11990.383159211986
Here are the final weights after convergence:
[0.99621425 0.54625306]
```

Task P6: Use the learned weights to predict 'People*tested' in the last three weeks in the dataset. Copy the predictions to the solution file, and calculate the test error $$\frac{1}{n*{\mathrm{tst}}}\sum*{i=1}^{n*{\mathrm{tst}}}(y_i^{\mathrm{tst}}-\hat{y}*i^{\mathrm{tst}})^2,$$ where $n*{\mathrm{tst}} $is the number of test data,$ y_i^{\mathrm{tst}} $is the true label,$ \hat{y}_i^{\mathrm{tst}}$ is the predicted label.

In [118]:
```
# Create the test data
test_data = data.iloc[-21:]
(test_simple_feature_matrix, test_output) = get_numpy_data(test_data, simple_features, my_output)
test_predictions = predict_output(test_simple_feature_matrix, final_weights)
print (test_predictions)
```

```
[102.59928352 103.14553658 103.69178964 104.2380427  104.78429576
 105.33054882 105.87680188 106.42305494 106.969308   107.51556106
 108.06181412 108.60806718 109.15432024 109.7005733  110.24682637
 110.79307943 111.33933249 111.88558555 112.43183861 112.97809167
 113.52434473]
```

```
<ipython-input-107-8b2916f5b454>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_frame['constant'] = 1 # here we are adding a constant column
```

In [121]:
```
# Calculate the test error
# STUDENT: Start of code

test_error = (1/len(test_predictions))* np.sum(np.square( np.subtract(test_data.People_tested, test_predictions)))
print(test_error)

#end of code
```

```
11490.99138344532
```

Linear regression using multiple features

Here, we will be considering multiple input features (`Intensive_care` , `New_positive_cases` , `Days`) to predict the `People_tested` in the future.

Task P7: Specify the initial_weights, step_size, and tolerance for the function `regression_gradient_descent` . Print the outputs of the code.

In [127]:
```
model_features = ['Intensive_care','New_positive_cases','Days']
my_output = 'People_tested'

#call the get_nupy_data method to calculate the feature matrix and output. Sto
re them in the variables "multi_feature_matrix" & "output"

(multi_feature_matrix, output) = get_numpy_data(data, model_features, my_outpu
t)

# Initialize the weights, step size and tolerance
# STUDENT: Start of code
# STUDENT: Specify the initial_weights, step_size, and tolerance
initial_weights = [ 1.0, 1.0, 1.0, 1.0 ]
step_size = 7e-8
tolerance = 1.2e4
# end of code
weight_2 = regression_gradient_descent(multi_feature_matrix, output, initial_w
eights, step_size, tolerance)
print ("Here are the final weights after convergence:")
print (weight_2)
```

```
Iteration:  1 gradient_magnitude:  28577.826184864996
Iteration:  2 gradient_magnitude:  28520.816323869836
Iteration:  3 gradient_magnitude:  28463.920191758247
Iteration:  4 gradient_magnitude:  28407.13756165268
Iteration:  5 gradient_magnitude:  28350.468207128142
Iteration:  6 gradient_magnitude:  28293.911902211352
Iteration:  7 gradient_magnitude:  28237.46842137985
Iteration:  8 gradient_magnitude:  28181.137539561012
Iteration:  9 gradient_magnitude:  28124.91903213125
Iteration:  10 gradient_magnitude:  28068.812674915087
Iteration:  11 gradient_magnitude:  28012.818244184222
Iteration:  12 gradient_magnitude:  27956.93551665667
Iteration:  13 gradient_magnitude:  27901.164269495923
Iteration:  14 gradient_magnitude:  27845.50428030994
Iteration:  15 gradient_magnitude:  27789.955327150372
Iteration:  16 gradient_magnitude:  27734.517188511614
Iteration:  17 gradient_magnitude:  27679.18964332996
Iteration:  18 gradient_magnitude:  27623.972470982688
Iteration:  19 gradient_magnitude:  27568.8654512872
Iteration:  20 gradient_magnitude:  27513.86836450015
Iteration:  21 gradient_magnitude:  27458.98099131656
Iteration:  22 gradient_magnitude:  27404.203112868905
Iteration:  23 gradient_magnitude:  27349.53451072633
Iteration:  24 gradient_magnitude:  27294.974966893693
Iteration:  25 gradient_magnitude:  27240.52426381075
Iteration:  26 gradient_magnitude:  27186.182184351255
Iteration:  27 gradient_magnitude:  27131.948511822106
Iteration:  28 gradient_magnitude:  27077.82302996248
Iteration:  29 gradient_magnitude:  27023.805522943003
Iteration:  30 gradient_magnitude:  26969.895775364814
Iteration:  31 gradient_magnitude:  26916.09357225877
Iteration:  32 gradient_magnitude:  26862.398699084588
Iteration:  33 gradient_magnitude:  26808.810941729942
Iteration:  34 gradient_magnitude:  26755.330086509646
Iteration:  35 gradient_magnitude:  26701.955920164815
Iteration:  36 gradient_magnitude:  26648.688229861964
Iteration:  37 gradient_magnitude:  26595.526803192206
Iteration:  38 gradient_magnitude:  26542.47142817038
Iteration:  39 gradient_magnitude:  26489.521893234232
Iteration:  40 gradient_magnitude:  26436.677987243518
Iteration:  41 gradient_magnitude:  26383.93949947924
Iteration:  42 gradient_magnitude:  26331.306219642735
Iteration:  43 gradient_magnitude:  26278.77793785487
Iteration:  44 gradient_magnitude:  26226.3544446552
Iteration:  45 gradient_magnitude:  26174.03553100114
Iteration:  46 gradient_magnitude:  26121.82098826711
Iteration:  47 gradient_magnitude:  26069.710608243724
Iteration:  48 gradient_magnitude:  26017.704183136964
Iteration:  49 gradient_magnitude:  25965.801505567313
Iteration:  50 gradient_magnitude:  25914.00236856898
Iteration:  51 gradient_magnitude:  25862.306565589042
Iteration:  52 gradient_magnitude:  25810.71389048662
Iteration:  53 gradient_magnitude:  25759.22413753208
Iteration:  54 gradient_magnitude:  25707.83710140618
Iteration:  55 gradient_magnitude:  25656.552577199294
Iteration:  56 gradient_magnitude:  25605.370360410525
Iteration:  57 gradient_magnitude:  25554.290246946995
```

```
Iteration:  58 gradient_magnitude:  25503.312033122922
Iteration:  59 gradient_magnitude:  25452.435515658886
Iteration:  60 gradient_magnitude:  25401.660491680952
Iteration:  61 gradient_magnitude:  25350.98675871996
Iteration:  62 gradient_magnitude:  25300.41411471058
Iteration:  63 gradient_magnitude:  25249.94235799065
Iteration:  64 gradient_magnitude:  25199.571287300263
Iteration:  65 gradient_magnitude:  25149.300701781005
Iteration:  66 gradient_magnitude:  25099.130400975177
Iteration:  67 gradient_magnitude:  25049.06018482496
Iteration:  68 gradient_magnitude:  24999.089853671623
Iteration:  69 gradient_magnitude:  24949.21920825475
Iteration:  70 gradient_magnitude:  24899.44804971141
Iteration:  71 gradient_magnitude:  24849.776179575398
Iteration:  72 gradient_magnitude:  24800.203399776434
Iteration:  73 gradient_magnitude:  24750.729512639347
Iteration:  74 gradient_magnitude:  24701.35432088332
Iteration:  75 gradient_magnitude:  24652.077627621096
Iteration:  76 gradient_magnitude:  24602.899236358197
Iteration:  77 gradient_magnitude:  24553.81895099208
Iteration:  78 gradient_magnitude:  24504.83657581148
Iteration:  79 gradient_magnitude:  24455.95191549551
Iteration:  80 gradient_magnitude:  24407.16477511294
Iteration:  81 gradient_magnitude:  24358.47496012139
Iteration:  82 gradient_magnitude:  24309.88227636661
Iteration:  83 gradient_magnitude:  24261.38653008164
Iteration:  84 gradient_magnitude:  24212.987527886078
Iteration:  85 gradient_magnitude:  24164.685076785267
Iteration:  86 gradient_magnitude:  24116.47898416962
Iteration:  87 gradient_magnitude:  24068.369057813707
Iteration:  88 gradient_magnitude:  24020.355105875624
Iteration:  89 gradient_magnitude:  23972.436936896145
Iteration:  90 gradient_magnitude:  23924.614359798004
Iteration:  91 gradient_magnitude:  23876.8871838851
Iteration:  92 gradient_magnitude:  23829.25521884177
Iteration:  93 gradient_magnitude:  23781.718274731982
Iteration:  94 gradient_magnitude:  23734.276161998634
Iteration:  95 gradient_magnitude:  23686.928691462752
Iteration:  96 gradient_magnitude:  23639.675674322774
Iteration:  97 gradient_magnitude:  23592.516922153765
Iteration:  98 gradient_magnitude:  23545.452246906676
Iteration:  99 gradient_magnitude:  23498.4814609076
Iteration:  100 gradient_magnitude:  23451.604376857034
Iteration:  101 gradient_magnitude:  23404.8208078291
Iteration:  102 gradient_magnitude:  23358.130567270826
Iteration:  103 gradient_magnitude:  23311.533469001388
Iteration:  104 gradient_magnitude:  23265.02932721139
Iteration:  105 gradient_magnitude:  23218.617956462087
Iteration:  106 gradient_magnitude:  23172.29917168467
Iteration:  107 gradient_magnitude:  23126.072788179543
Iteration:  108 gradient_magnitude:  23079.93862161555
Iteration:  109 gradient_magnitude:  23033.896488029244
Iteration:  110 gradient_magnitude:  22987.94620382419
Iteration:  111 gradient_magnitude:  22942.08758577021
Iteration:  112 gradient_magnitude:  22896.32045100263
Iteration:  113 gradient_magnitude:  22850.644617021586
Iteration:  114 gradient_magnitude:  22805.059901691275
```

```
Iteration:  115 gradient_magnitude:  22759.566123239263
Iteration:  116 gradient_magnitude:  22714.163100255697
Iteration:  117 gradient_magnitude:  22668.850651692635
Iteration:  118 gradient_magnitude:  22623.62859686332
Iteration:  119 gradient_magnitude:  22578.496755441407
Iteration:  120 gradient_magnitude:  22533.45494746034
Iteration:  121 gradient_magnitude:  22488.502993312522
Iteration:  122 gradient_magnitude:  22443.640713748704
Iteration:  123 gradient_magnitude:  22398.867929877186
Iteration:  124 gradient_magnitude:  22354.18446316315
Iteration:  125 gradient_magnitude:  22309.590135427945
Iteration:  126 gradient_magnitude:  22265.08476884836
Iteration:  127 gradient_magnitude:  22220.668185955918
Iteration:  128 gradient_magnitude:  22176.340209636175
Iteration:  129 gradient_magnitude:  22132.100663128036
Iteration:  130 gradient_magnitude:  22087.949370022987
Iteration:  131 gradient_magnitude:  22043.88615426447
Iteration:  132 gradient_magnitude:  21999.910840147117
Iteration:  133 gradient_magnitude:  21956.023252316074
Iteration:  134 gradient_magnitude:  21912.223215766324
Iteration:  135 gradient_magnitude:  21868.510555841935
Iteration:  136 gradient_magnitude:  21824.885098235423
Iteration:  137 gradient_magnitude:  21781.34666898702
Iteration:  138 gradient_magnitude:  21737.895094483985
Iteration:  139 gradient_magnitude:  21694.53020145993
Iteration:  140 gradient_magnitude:  21651.251816994103
Iteration:  141 gradient_magnitude:  21608.059768510713
Iteration:  142 gradient_magnitude:  21564.953883778257
Iteration:  143 gradient_magnitude:  21521.933990908787
Iteration:  144 gradient_magnitude:  21478.999918357273
Iteration:  145 gradient_magnitude:  21436.151494920898
Iteration:  146 gradient_magnitude:  21393.388549738378
Iteration:  147 gradient_magnitude:  21350.710912289276
Iteration:  148 gradient_magnitude:  21308.118412393327
Iteration:  149 gradient_magnitude:  21265.61088020977
Iteration:  150 gradient_magnitude:  21223.188146236633
Iteration:  151 gradient_magnitude:  21180.8500413101
Iteration:  152 gradient_magnitude:  21138.59639660382
Iteration:  153 gradient_magnitude:  21096.42704362824
Iteration:  154 gradient_magnitude:  21054.341814229905
Iteration:  155 gradient_magnitude:  21012.340540590827
Iteration:  156 gradient_magnitude:  20970.423055227795
Iteration:  157 gradient_magnitude:  20928.5891909917
Iteration:  158 gradient_magnitude:  20886.8387810669
Iteration:  159 gradient_magnitude:  20845.171658970514
Iteration:  160 gradient_magnitude:  20803.58765855178
Iteration:  161 gradient_magnitude:  20762.086613991396
Iteration:  162 gradient_magnitude:  20720.668359800842
Iteration:  163 gradient_magnitude:  20679.33273082175
Iteration:  164 gradient_magnitude:  20638.0795622252
Iteration:  165 gradient_magnitude:  20596.90868951112
Iteration:  166 gradient_magnitude:  20555.81994850757
Iteration:  167 gradient_magnitude:  20514.813175370142
Iteration:  168 gradient_magnitude:  20473.88820658125
Iteration:  169 gradient_magnitude:  20433.044878949542
Iteration:  170 gradient_magnitude:  20392.28302960919
Iteration:  171 gradient_magnitude:  20351.60249601929
```

```
Iteration:  172 gradient_magnitude:  20311.003115963165
Iteration:  173 gradient_magnitude:  20270.48472754777
Iteration:  174 gradient_magnitude:  20230.047169202993
Iteration:  175 gradient_magnitude:  20189.69027968107
Iteration:  176 gradient_magnitude:  20149.413898055875
Iteration:  177 gradient_magnitude:  20109.217863722337
Iteration:  178 gradient_magnitude:  20069.102016395773
Iteration:  179 gradient_magnitude:  20029.066196111242
Iteration:  180 gradient_magnitude:  19989.110243222924
Iteration:  181 gradient_magnitude:  19949.233998403484
Iteration:  182 gradient_magnitude:  19909.43730264341
Iteration:  183 gradient_magnitude:  19869.719997250413
Iteration:  184 gradient_magnitude:  19830.081923848764
Iteration:  185 gradient_magnitude:  19790.522924378693
Iteration:  186 gradient_magnitude:  19751.042841095747
Iteration:  187 gradient_magnitude:  19711.641516570133
Iteration:  188 gradient_magnitude:  19672.318793686132
Iteration:  189 gradient_magnitude:  19633.074515641456
Iteration:  190 gradient_magnitude:  19593.908525946623
Iteration:  191 gradient_magnitude:  19554.82066842431
Iteration:  192 gradient_magnitude:  19515.810787208786
Iteration:  193 gradient_magnitude:  19476.87872674522
Iteration:  194 gradient_magnitude:  19438.02433178914
Iteration:  195 gradient_magnitude:  19399.247447405724
Iteration:  196 gradient_magnitude:  19360.547918969252
Iteration:  197 gradient_magnitude:  19321.92559216248
Iteration:  198 gradient_magnitude:  19283.38031297599
Iteration:  199 gradient_magnitude:  19244.911927707606
Iteration:  200 gradient_magnitude:  19206.52028296177
Iteration:  201 gradient_magnitude:  19168.20522564893
Iteration:  202 gradient_magnitude:  19129.966602984943
Iteration:  203 gradient_magnitude:  19091.80426249045
Iteration:  204 gradient_magnitude:  19053.718051990258
Iteration:  205 gradient_magnitude:  19015.707819612762
Iteration:  206 gradient_magnitude:  18977.77341378933
Iteration:  207 gradient_magnitude:  18939.91468325369
Iteration:  208 gradient_magnitude:  18902.131477041323
Iteration:  209 gradient_magnitude:  18864.42364448887
Iteration:  210 gradient_magnitude:  18826.791035233553
Iteration:  211 gradient_magnitude:  18789.233499212514
Iteration:  212 gradient_magnitude:  18751.75088666228
Iteration:  213 gradient_magnitude:  18714.343048118146
Iteration:  214 gradient_magnitude:  18677.009834413548
Iteration:  215 gradient_magnitude:  18639.751096679523
Iteration:  216 gradient_magnitude:  18602.56668634407
Iteration:  217 gradient_magnitude:  18565.45645513159
Iteration:  218 gradient_magnitude:  18528.42025506224
Iteration:  219 gradient_magnitude:  18491.45793845142
Iteration:  220 gradient_magnitude:  18454.569357909146
Iteration:  221 gradient_magnitude:  18417.75436633944
Iteration:  222 gradient_magnitude:  18381.01281693977
Iteration:  223 gradient_magnitude:  18344.344563200477
Iteration:  224 gradient_magnitude:  18307.749458904156
Iteration:  225 gradient_magnitude:  18271.227358125103
Iteration:  226 gradient_magnitude:  18234.7781152287
Iteration:  227 gradient_magnitude:  18198.40158487089
Iteration:  228 gradient_magnitude:  18162.097621997542
```

```
Iteration:  229 gradient_magnitude:  18125.86608184388
Iteration:  230 gradient_magnitude:  18089.706819933945
Iteration:  231 gradient_magnitude:  18053.619692079978
Iteration:  232 gradient_magnitude:  18017.60455438186
Iteration:  233 gradient_magnitude:  17981.661263226557
Iteration:  234 gradient_magnitude:  17945.789675287495
Iteration:  235 gradient_magnitude:  17909.989647524046
Iteration:  236 gradient_magnitude:  17874.26103718093
Iteration:  237 gradient_magnitude:  17838.603701787648
Iteration:  238 gradient_magnitude:  17803.01749915791
Iteration:  239 gradient_magnitude:  17767.50228738908
Iteration:  240 gradient_magnitude:  17732.057924861598
Iteration:  241 gradient_magnitude:  17696.684270238424
Iteration:  242 gradient_magnitude:  17661.38118246447
Iteration:  243 gradient_magnitude:  17626.148520766048
Iteration:  244 gradient_magnitude:  17590.98614465028
Iteration:  245 gradient_magnitude:  17555.893913904558
Iteration:  246 gradient_magnitude:  17520.87168859601
Iteration:  247 gradient_magnitude:  17485.91932907089
Iteration:  248 gradient_magnitude:  17451.036695954062
Iteration:  249 gradient_magnitude:  17416.223650148426
Iteration:  250 gradient_magnitude:  17381.480052834348
Iteration:  251 gradient_magnitude:  17346.80576546915
Iteration:  252 gradient_magnitude:  17312.200649786537
Iteration:  253 gradient_magnitude:  17277.664567796004
Iteration:  254 gradient_magnitude:  17243.197381782364
Iteration:  255 gradient_magnitude:  17208.798954305123
Iteration:  256 gradient_magnitude:  17174.469148198008
Iteration:  257 gradient_magnitude:  17140.207826568334
Iteration:  258 gradient_magnitude:  17106.014852796532
Iteration:  259 gradient_magnitude:  17071.89009053557
Iteration:  260 gradient_magnitude:  17037.83340371041
Iteration:  261 gradient_magnitude:  17003.84465651747
Iteration:  262 gradient_magnitude:  16969.923713424083
Iteration:  263 gradient_magnitude:  16936.070439167965
Iteration:  264 gradient_magnitude:  16902.284698756645
Iteration:  265 gradient_magnitude:  16868.566357466974
Iteration:  266 gradient_magnitude:  16834.915280844536
Iteration:  267 gradient_magnitude:  16801.33133470317
Iteration:  268 gradient_magnitude:  16767.814385124373
Iteration:  269 gradient_magnitude:  16734.36429845681
Iteration:  270 gradient_magnitude:  16700.980941315756
Iteration:  271 gradient_magnitude:  16667.664180582597
Iteration:  272 gradient_magnitude:  16634.413883404257
Iteration:  273 gradient_magnitude:  16601.229917192686
Iteration:  274 gradient_magnitude:  16568.112149624354
Iteration:  275 gradient_magnitude:  16535.06044863969
Iteration:  276 gradient_magnitude:  16502.074682442566
Iteration:  277 gradient_magnitude:  16469.15471949978
Iteration:  278 gradient_magnitude:  16436.300428540522
Iteration:  279 gradient_magnitude:  16403.51167855585
Iteration:  280 gradient_magnitude:  16370.788338798186
Iteration:  281 gradient_magnitude:  16338.130278780762
Iteration:  282 gradient_magnitude:  16305.53736827713
Iteration:  283 gradient_magnitude:  16273.009477320624
Iteration:  284 gradient_magnitude:  16240.546476203855
Iteration:  285 gradient_magnitude:  16208.14823547819
```

```
Iteration:  286 gradient_magnitude:  16175.81462595321
Iteration:  287 gradient_magnitude:  16143.545518696244
Iteration:  288 gradient_magnitude:  16111.340785031816
Iteration:  289 gradient_magnitude:  16079.200296541152
Iteration:  290 gradient_magnitude:  16047.123925061655
Iteration:  291 gradient_magnitude:  16015.11154268639
Iteration:  292 gradient_magnitude:  15983.163021763603
Iteration:  293 gradient_magnitude:  15951.278234896186
Iteration:  294 gradient_magnitude:  15919.457054941156
Iteration:  295 gradient_magnitude:  15887.699355009194
Iteration:  296 gradient_magnitude:  15856.005008464104
Iteration:  297 gradient_magnitude:  15824.373888922306
Iteration:  298 gradient_magnitude:  15792.805870252356
Iteration:  299 gradient_magnitude:  15761.300826574421
Iteration:  300 gradient_magnitude:  15729.85863225979
Iteration:  301 gradient_magnitude:  15698.479161930365
Iteration:  302 gradient_magnitude:  15667.162290458165
Iteration:  303 gradient_magnitude:  15635.907892964831
Iteration:  304 gradient_magnitude:  15604.715844821121
Iteration:  305 gradient_magnitude:  15573.586021646408
Iteration:  306 gradient_magnitude:  15542.51829930821
Iteration:  307 gradient_magnitude:  15511.512553921655
Iteration:  308 gradient_magnitude:  15480.568661849027
Iteration:  309 gradient_magnitude:  15449.686499699244
Iteration:  310 gradient_magnitude:  15418.865944327374
Iteration:  311 gradient_magnitude:  15388.106872834163
Iteration:  312 gradient_magnitude:  15357.409162565516
Iteration:  313 gradient_magnitude:  15326.772691112012
Iteration:  314 gradient_magnitude:  15296.197336308449
Iteration:  315 gradient_magnitude:  15265.682976233307
Iteration:  316 gradient_magnitude:  15235.229489208303
Iteration:  317 gradient_magnitude:  15204.836753797883
Iteration:  318 gradient_magnitude:  15174.504648808746
Iteration:  319 gradient_magnitude:  15144.233053289365
Iteration:  320 gradient_magnitude:  15114.021846529487
Iteration:  321 gradient_magnitude:  15083.870908059673
Iteration:  322 gradient_magnitude:  15053.780117650804
Iteration:  323 gradient_magnitude:  15023.749355313605
Iteration:  324 gradient_magnitude:  14993.778501298179
Iteration:  325 gradient_magnitude:  14963.867436093502
Iteration:  326 gradient_magnitude:  14934.016040426975
Iteration:  327 gradient_magnitude:  14904.224195263918
Iteration:  328 gradient_magnitude:  14874.49178180714
Iteration:  329 gradient_magnitude:  14844.818681496416
Iteration:  330 gradient_magnitude:  14815.204776008044
Iteration:  331 gradient_magnitude:  14785.64994725437
Iteration:  332 gradient_magnitude:  14756.154077383298
Iteration:  333 gradient_magnitude:  14726.71704877786
Iteration:  334 gradient_magnitude:  14697.338744055693
Iteration:  335 gradient_magnitude:  14668.01904606863
Iteration:  336 gradient_magnitude:  14638.757837902172
Iteration:  337 gradient_magnitude:  14609.55500287508
Iteration:  338 gradient_magnitude:  14580.410424538863
Iteration:  339 gradient_magnitude:  14551.323986677338
Iteration:  340 gradient_magnitude:  14522.295573306172
Iteration:  341 gradient_magnitude:  14493.325068672395
Iteration:  342 gradient_magnitude:  14464.412357253954
```

```
Iteration:  343 gradient_magnitude:  14435.55732375926
Iteration:  344 gradient_magnitude:  14406.759853126712
Iteration:  345 gradient_magnitude:  14378.019830524245
Iteration:  346 gradient_magnitude:  14349.33714134888
Iteration:  347 gradient_magnitude:  14320.711671226243
Iteration:  348 gradient_magnitude:  14292.143306010146
Iteration:  349 gradient_magnitude:  14263.631931782098
Iteration:  350 gradient_magnitude:  14235.177434850855
Iteration:  351 gradient_magnitude:  14206.779701752002
Iteration:  352 gradient_magnitude:  14178.438619247454
Iteration:  353 gradient_magnitude:  14150.15407432503
Iteration:  354 gradient_magnitude:  14121.925954197994
Iteration:  355 gradient_magnitude:  14093.754146304615
Iteration:  356 gradient_magnitude:  14065.6385383077
Iteration:  357 gradient_magnitude:  14037.579018094166
Iteration:  358 gradient_magnitude:  14009.57547377458
Iteration:  359 gradient_magnitude:  13981.62779368273
Iteration:  360 gradient_magnitude:  13953.735866375147
Iteration:  361 gradient_magnitude:  13925.899580630687
Iteration:  362 gradient_magnitude:  13898.118825450083
Iteration:  363 gradient_magnitude:  13870.393490055514
Iteration:  364 gradient_magnitude:  13842.723463890115
Iteration:  365 gradient_magnitude:  13815.108636617611
Iteration:  366 gradient_magnitude:  13787.548898121806
Iteration:  367 gradient_magnitude:  13760.04413850619
Iteration:  368 gradient_magnitude:  13732.59424809348
Iteration:  369 gradient_magnitude:  13705.199117425193
Iteration:  370 gradient_magnitude:  13677.858637261194
Iteration:  371 gradient_magnitude:  13650.57269857928
Iteration:  372 gradient_magnitude:  13623.341192574737
Iteration:  373 gradient_magnitude:  13596.164010659906
Iteration:  374 gradient_magnitude:  13569.041044463738
Iteration:  375 gradient_magnitude:  13541.972185831382
Iteration:  376 gradient_magnitude:  13514.957326823753
Iteration:  377 gradient_magnitude:  13487.996359717074
Iteration:  378 gradient_magnitude:  13461.089177002483
Iteration:  379 gradient_magnitude:  13434.235671385573
Iteration:  380 gradient_magnitude:  13407.435735785997
Iteration:  381 gradient_magnitude:  13380.689263337006
Iteration:  382 gradient_magnitude:  13353.996147385045
Iteration:  383 gradient_magnitude:  13327.356281489321
Iteration:  384 gradient_magnitude:  13300.769559421387
Iteration:  385 gradient_magnitude:  13274.235875164695
Iteration:  386 gradient_magnitude:  13247.755122914203
Iteration:  387 gradient_magnitude:  13221.327197075942
Iteration:  388 gradient_magnitude:  13194.951992266573
Iteration:  389 gradient_magnitude:  13168.629403313013
Iteration:  390 gradient_magnitude:  13142.359325251959
Iteration:  391 gradient_magnitude:  13116.141653329525
Iteration:  392 gradient_magnitude:  13089.976283000782
Iteration:  393 gradient_magnitude:  13063.863109929367
Iteration:  394 gradient_magnitude:  13037.802029987046
Iteration:  395 gradient_magnitude:  13011.792939253324
Iteration:  396 gradient_magnitude:  12985.835734015016
Iteration:  397 gradient_magnitude:  12959.930310765816
Iteration:  398 gradient_magnitude:  12934.076566205924
Iteration:  399 gradient_magnitude:  12908.274397241605
```

```
Iteration:  400 gradient_magnitude:  12882.52370098478
Iteration:  401 gradient_magnitude:  12856.824374752629
Iteration:  402 gradient_magnitude:  12831.176316067173
Iteration:  403 gradient_magnitude:  12805.579422654859
Iteration:  404 gradient_magnitude:  12780.03359244616
Iteration:  405 gradient_magnitude:  12754.538723575179
Iteration:  406 gradient_magnitude:  12729.094714379225
Iteration:  407 gradient_magnitude:  12703.701463398409
Iteration:  408 gradient_magnitude:  12678.358869375248
Iteration:  409 gradient_magnitude:  12653.06683125426
Iteration:  410 gradient_magnitude:  12627.82524818155
Iteration:  411 gradient_magnitude:  12602.634019504425
Iteration:  412 gradient_magnitude:  12577.49304477098
Iteration:  413 gradient_magnitude:  12552.4022237297
Iteration:  414 gradient_magnitude:  12527.361456329065
Iteration:  415 gradient_magnitude:  12502.370642717155
Iteration:  416 gradient_magnitude:  12477.429683241218
Iteration:  417 gradient_magnitude:  12452.53847844733
Iteration:  418 gradient_magnitude:  12427.696929079948
Iteration:  419 gradient_magnitude:  12402.904936081546
Iteration:  420 gradient_magnitude:  12378.1624005922
Iteration:  421 gradient_magnitude:  12353.469223949198
Iteration:  422 gradient_magnitude:  12328.82530768666
Iteration:  423 gradient_magnitude:  12304.23055353513
Iteration:  424 gradient_magnitude:  12279.684863421187
Iteration:  425 gradient_magnitude:  12255.188139467067
Iteration:  426 gradient_magnitude:  12230.740283990246
Iteration:  427 gradient_magnitude:  12206.341199503086
Iteration:  428 gradient_magnitude:  12181.990788712408
Iteration:  429 gradient_magnitude:  12157.688954519137
Iteration:  430 gradient_magnitude:  12133.435600017889
Iteration:  431 gradient_magnitude:  12109.230628496609
Iteration:  432 gradient_magnitude:  12085.073943436166
Iteration:  433 gradient_magnitude:  12060.965448509976
Iteration:  434 gradient_magnitude:  12036.905047583612
Iteration:  435 gradient_magnitude:  12012.892644714433
Iteration:  436 gradient_magnitude:  11988.928144151192
Here are the final weights after convergence:
[0.99572553 0.99950436 0.99941219 0.41591547]
```

Task P8: Use the learned weights to predict 'People_tested' in the last three weeks in the dataset.Find the value of the model predictions on the 10th day of the forecasting period.Also print the actual number of people tested on that particular day. Copy the predictions to the solution file, and calculate the test error. Note: here we are asking you to report the number before normalization. So you need to convert the prediction back to the unit of people.

In [128]:
```
(test_feature_matrix, test_output) = get_numpy_data(test_data, model_features, my_output)

test_predictions_2 = predict_output(test_feature_matrix, weight_2)

#Prediction for the 10th day of the forecasting period.
print (test_predictions_2[10])

#Convert the normalized data back to original figures using the same min-max normalization
prediction_10th_day = test_predictions_2[10] * (data_orig['People_tested'].max() - data_orig['People_tested'].min()) + data_orig['People_tested'].min()

print ("Model prediction of the 10th day:",int(prediction_10th_day))

# Get the actual number of people tested from our test data on 10 th day of forecasting period.
actual_people_tested = data_orig["People_tested"].iloc[190]

print ("Actual number of people tested on the 10th day:",actual_people_tested)
```

```
82.72965202021162
Model prediction of the 10th day: 830628809
Actual number of people tested on the 10th day: 8725909

<ipython-input-107-8b2916f5b454>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_frame['constant'] = 1 # here we are adding a constant column
```

In [129]:
```
# Calculate the test error
# STUDENT: Start of code
test_error = (1/len(test_predictions_2))* np.sum(np.square( np.subtract(test_data.People_tested, test_predictions_2)))
print(test_error)

# end of code
```

```
6701.844086335183
```

Explore on your own

Now that you have tried two models for predictions, in this section, you can explore on your own an aspect of the problem that interests you. Here are some examples:

- What features or what combination of features are most predictive for 'People_tested'?
- How does tolerance for convergence affect prediction errors?
- How does step size affect prediction errors?
- How can we use validation to select the set of features to improve prediction?

Report your question of investigation, as well as your results/interpretation in the solution file.

In [138]:
```
def regression_gradient_descent_2(feature_matrix, labels, initial_weights, step_size, tolerance):
    # Gradient descent algorithm for linear regression problem

    # Input:
    # feature_matrix: numpy array of size n by d, where n is the number of data points, and d is the feature dimension
    # labels: true labels y, a numpy vector of dimension d
    # initial_weights: initial weight vector to start with, a numpy vector of dimension d
    # step_size: step size of update
    # tolerance: tolerace epsilon for stopping condition
    # Output:
    # Weights obtained after convergence

    converged = False
    weights = np.array(initial_weights) # current iterate
    i = 0
    while not converged:
        i += 1
        # STUDENT: Start of code: your impelementation of what the gradient descent algorithm does in every iteration
        # Refer back to the update rule listed above: update the weight
        derivative = weight_derivative(weights, feature_matrix, labels);
        weights = weights - step_size*derivative

        # Compute the gradient magnitude:

        gradient_magnitude = np.linalg.norm( weight_derivative(weights, feature_matrix, labels) )

        # Check the stopping condition to decide whether you want to stop the iterations
        if  gradient_magnitude <= tolerance:   # STUDENT: check the stopping condition here
            converged = True

        # End of code

        #print ("Iteration: ",i,"gradient_magnitude: ", gradient_magnitude) # for us to check about convergence

    return(weights)
```

In [141]:
```
# Explore an aspect of the model that interests you
### STUDENT: Start of code

simple_features = ['Days']
my_output = 'People_tested'

# Use get_numpy_data method to calculate the feature matrix and output.
(simple_feature_matrix, output) = get_numpy_data(train_data, simple_features,
my_output)

#Initialize the weights, step size and tolerance
# Start of code
#STUDENT: Specify the initial_weights, step_size, and tolerance
initial_weights = [1.0, 1.0]
step_size = 7e-8

tolerance_1 = 0.8e4
tolerance_2 = 1.0e4
tolerance_3 = 1.2e4
tolerance_4 = 1.4e4
# end of code

# Use the regression_gradient_descent function to calculate the gradient decen
t and store it in the variable 'final_weights'
final_weights_1 = regression_gradient_descent_2(simple_feature_matrix, output,
initial_weights, step_size, tolerance_1)
final_weights_2 = regression_gradient_descent_2(simple_feature_matrix, output,
initial_weights, step_size, tolerance_2)
final_weights_3 = regression_gradient_descent_2(simple_feature_matrix, output,
initial_weights, step_size, tolerance_3)
final_weights_4 = regression_gradient_descent_2(simple_feature_matrix, output,
initial_weights, step_size, tolerance_4)
# end of code

#print ("Here are the final weights after convergence with tolerance of 0.8e
4:")
#print (final_weights_1)
#print ("Here are the final weights after convergence with tolerance of 1.0e
4:")
#print (final_weights_2)
#print ("Here are the final weights after convergence with tolerance of 1.2e
4:")
#print (final_weights_3)
#print ("Here are the final weights after convergence with tolerance of 1.4e
4:")
#print (final_weights_4)

test_data = data.iloc[-21:]
(test_simple_feature_matrix, test_output) = get_numpy_data(test_data, simple_f
eatures, my_output)
test_predictions_1 = predict_output(test_simple_feature_matrix, final_weights_
1)
test_predictions_2 = predict_output(test_simple_feature_matrix, final_weights_
2)
test_predictions_3 = predict_output(test_simple_feature_matrix, final_weights_
3)
```

```
test_predictions_4 = predict_output(test_simple_feature_matrix, final_weights_
4)

test_error = (1/len(test_predictions_1))* np.sum(np.square( np.subtract(test_d
ata.People_tested, test_predictions_1)))
print("Error for tolerance of 0.8e4:", test_error)
test_error = (1/len(test_predictions_2))* np.sum(np.square( np.subtract(test_d
ata.People_tested, test_predictions_2)))
print("Error for tolerance of 1.0e4:", test_error)
test_error = (1/len(test_predictions_3))* np.sum(np.square( np.subtract(test_d
ata.People_tested, test_predictions_3)))
print( "Error for tolerance of 1.2e4:", test_error)
test_error = (1/len(test_predictions_4))* np.sum(np.square( np.subtract(test_d
ata.People_tested, test_predictions_4)))
print("Error for tolerance of 1.4e4:", test_error)
#end of code

### End of code
```

```
Error for tolerance of 0.8e4: 5067.1321292306275
Error for tolerance of 1.0e4: 7969.441808885362
Error for tolerance of 1.2e4: 11490.99138344532
Error for tolerance of 1.4e4: 15670.65863747879

<ipython-input-107-8b2916f5b454>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_frame['constant'] = 1 # here we are adding a constant column
```

My question of investigation was how does tolerance for convergence affect the test error. I used 4 different tolerance values ranging from 0.8e4 to 1.4e4. As I increased the tolerance value, the error also increased as well. It seems like the smaller the tolerance I give it, the less room for error there is for the model to make.