In [2]:

```
import pandas as pd
file1 = r'D:\Work\Projects\COVID-19\covid_19_data.csv'
df = pd.read_csv(file1)

df1 = df.loc[df['Country/Region'] == 'India', ['ObservationDate', 'Confirmed
print ("\nFirst 5 rows:\n",df1.head())
print ("\nLast 5 rows:\n",df1.tail())
```

```

First 5 rows:
     ObservationDate  Confirmed  Deaths
430      01/30/2020          1       0
491      01/31/2020          1       0
547      02-01-2020          1       0
607      02-02-2020          2       0
672      02-03-2020          3       0

Last 5 rows:
       ObservationDate  Confirmed  Deaths
75166      07/29/2020       4485      21
75180      07/29/2020          0       0
75189      07/29/2020      77334    1530
75190      07/29/2020       6866      72
75216      07/29/2020      65258    1490
```

In [27]:

```
print("\ndescribe:\n", df1.describe())
```

```

describe:
            Confirmed        Deaths
count    1934.000000   1934.000000
mean    22700.979317    609.530507
std     48620.522498   1669.455955
min         0.000000      0.000000
25%       727.000000      1.000000
50%      4745.000000     29.000000
75%     19909.250000    397.000000
max    400651.000000  14463.000000
```

In [3]:

```
df1['Date-parsed'] = pd.to_datetime(df1['ObservationDate'],format = "%d%m%y"
   infer_datetime_format=True)
week_of_the_year = df1['Date-parsed'].dt.week
new_week =week_of_the_year.astype(float)
```

In [7]:

```
new_confirmed = df1["Confirmed"].astype(float)
new_deaths = df1["Deaths"].astype(float)
```

In [8]:

```
import numpy as np
bins = np.linspace(min(df['Confirmed']), max(new_confirmed), 4)
group = ['Mild',"Severe","Critical"]

df1['Confirmed-binned'] = pd.cut(new_confirmed, bins,
                           labels=group, include_lowest = True)
v_counts = df1["Confirmed-binned"].value_counts()
v_counts.index.name = "Confirmed cases outbreak Range"
print("\n", v_counts)
```

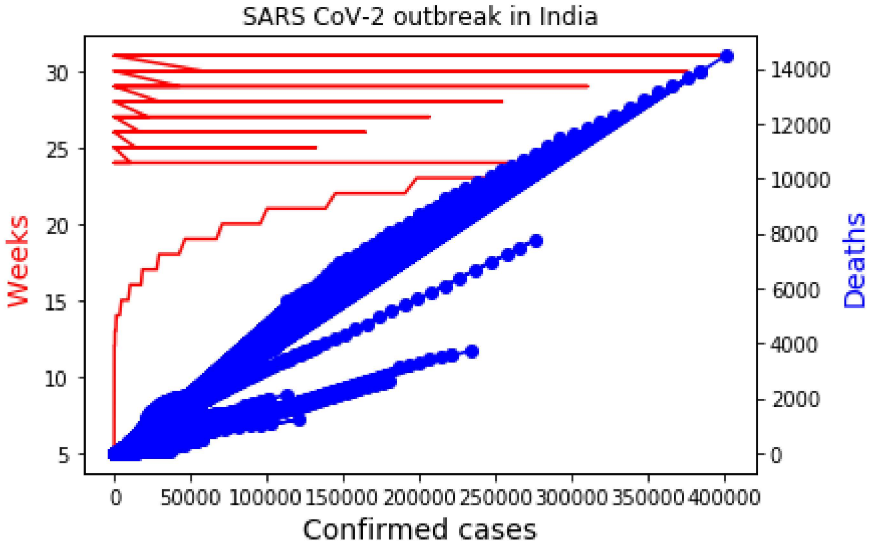
```
 Confirmed cases outbreak Range
Mild        1860
Severe        57
Critical      17
Name: Confirmed-binned, dtype: int64
```

In [11]:

```
#twinx method

import matplotlib.pyplot as plt

fig,ax = plt.subplots()
ax.plot(df1["Confirmed"],week_of_the_year , color="red")
ax.set_xlabel("Confirmed cases",fontsize=14)
ax.set_ylabel("Weeks ",color="red",fontsize=14)
ax2=ax.twinx()
ax2.plot(df1["Confirmed"],df1["Deaths"] ,color="blue",marker="o")
ax2.set_ylabel("Deaths",color="blue",fontsize=14)
plt.title("SARS CoV-2 outbreak in India")

```

Out[11]:

```
Text(0.5, 1.0, 'SARS CoV-2 outbreak in India')
```



In [12]:

```python
#SimpleLinearRegression


from sklearn.linear_model import LinearRegression

X = df1[["Confirmed"]]
Y = week_of_the_year
Z = df1[['Confirmed','Deaths']]
lm = LinearRegression()

lm.fit(X,Y)
#   wlm.fit(Z, week_of_the_year)
Yhat = lm.predict(X)
print("\nIntercept:\n",lm.intercept_)
print("\nCoefficient:\n",lm.coef_)
print("\nThis shows the rise in Confirmed cases has very less impact on aver
print("This implies cases are escalating in shorter period of time.\n")
```

```

Intercept:
 26.26334709505897

Coefficient:
 [9.37704053e-06]

This shows the rise in Confirmed cases has very less impact on average change i
n weeks.
This implies cases are escalating in shorter period of time.
```

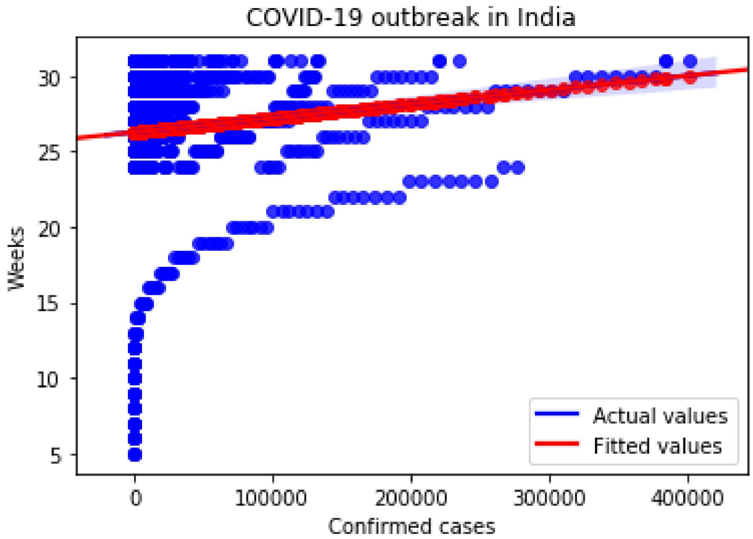
In [13]:

```
#Regression Plot

import seaborn as sns

print("\nRegression Plot:")
sns.regplot(X, Y, data=df1, color="blue", label="A")
sns.regplot(X, Yhat, data=df1, color="red", label="B")
plt.legend(labels=['Actual values', 'Fitted values'])
plt.xlabel('Confirmed cases')
plt.ylabel('Weeks')
plt.title("COVID-19 outbreak in India")
```

```
Regression Plot:
```

Out[13]:

```
Text(0.5, 1.0, 'COVID-19 outbreak in India')
```



In [14]:

```
#Mean Squared Error

from sklearn.metrics import mean_squared_error as mse

MSE = mse(X, Yhat)
print('\nMSE:\n',MSE)
```

```
MSE:
 2876821674.264937
```

In [15]:

```python
#R-squared error

from sklearn.metrics import r2_score as rsq

R_sq = rsq(X, Yhat)
print('\nR_squared:\n',R_sq)
```

```
R_squared:
 -0.2175822673686192
```

In [17]:

```python
#Training and testing

from sklearn.model_selection import train_test_split as tts
from sklearn.preprocessing import StandardScaler

x_train, x_test, y_train, y_test = tts(X, Y, test_size=0.2,
                                       random_state=0)
s = StandardScaler()
x_train = s.fit_transform(x_train)
x_test = s.transform(x_test)
```

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

class sklearn.svm.SVC(C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr',break_ties=False, random_state=None)

In [21]:

```python
from sklearn import svm

clf = svm.SVC()
clf.fit(x_train, y_train)
pred = clf.predict(x_test)
```

```
In [20]:  print(pred)
```

```
[26 26 27 26 30 26 26 28 26 26 26 26 30 30 26 26 28 26 28 26 30 26 26 28
 26 28 26 30 27 26 26 28 26 26 28 26 26 26 26 27 26 30 26 26 28 26 26 26
 26 27 26 26 28 28 26 27 26 26 26 26 26 26 26 26 28 26 26 26 26 26 30 27
 26 26 26 29 28 26 26 26 26 30 26 30 26 28 26 26 26 26 26 28 26 26 26 29
 26 26 26 30 30 30 26 28 26 30 26 26 29 26 26 30 29 26 26 26 26 26 27 26
 26 26 26 26 26 26 27 30 30 27 26 26 26 26 26 26 30 26 26 29 28 27 26 26
 30 30 28 29 26 26 26 29 30 28 30 28 28 26 30 26 26 30 26 26 26 26 27 30
 26 27 26 30 27 28 26 30 26 26 28 28 26 26 26 28 27 26 26 26 26 28 27 28
 26 26 26 26 26 26 29 26 26 26 27 26 26 26 26 26 26 26 26 29 26 28 26 26
 26 26 28 26 26 26 26 30 26 26 26 26 30 30 27 26 27 26 26 30 26 30 26 28
 26 27 26 26 27 26 30 30 26 26 27 26 30 26 26 26 26 28 26 26 26 26 30 26
 26 26 26 26 26 26 26 26 30 26 26 26 26 26 26 26 26 26 26 26 26 26 28 28
 26 26 30 26 28 26 26 30 26 26 26 26 26 28 26 28 26 26 26 26 26 26 26 28
 27 29 26 26 26 28 26 30 28 26 28 26 28 30 26 27 28 30 26 26 30 30 26 26
 26 30 26 28 26 26 26 26 26 26 26 26 26 30 29 26 26 26 26 28 26 26 26 29
 30 26 28 27 26 26 26 29 26 28 26 26 26 28 26 26 26 26 30 27 26 26 26 28
 26 26 30]
```

Classification Report syntax
classification_report(y_true, y_pred, labels=None, target_names=None, sample_weight=None, digits=2, output_dict=False, zero_division='warn')

In [24]:

```
#Classification report

from sklearn.metrics import classification_report

print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           5       0.00      0.00      0.00         1
           6       0.00      0.00      0.00         3
           7       0.00      0.00      0.00         2
           8       0.00      0.00      0.00         2
           9       0.00      0.00      0.00         2
          10       0.00      0.00      0.00         2
          11       0.00      0.00      0.00         1
          12       0.00      0.00      0.00         3
          13       0.00      0.00      0.00         4
          14       0.00      0.00      0.00         1
          16       0.00      0.00      0.00         3
          17       0.00      0.00      0.00         2
          18       0.00      0.00      0.00         1
          20       0.00      0.00      0.00         1
          21       0.00      0.00      0.00         1
          22       0.00      0.00      0.00         2
          23       0.00      0.00      0.00         1
          24       0.00      0.00      0.00        43
          25       0.00      0.00      0.00        49
          26       0.13      0.71      0.21        45
          27       0.24      0.15      0.18        41
          28       0.21      0.19      0.20        52
          29       0.23      0.06      0.09        54
          30       0.19      0.20      0.19        46
          31       0.00      0.00      0.00        25

    accuracy                           0.16       387
   macro avg       0.04      0.05      0.04       387
weighted avg       0.12      0.16      0.11       387
```

Accuracy Score

sklearn.metrics.accuracy_score(y_true, y_pred, normalize=True, sample_weight=None)

In [26]:

```
#Accuracy_score

from sklearn.metrics import accuracy_score

print("Accuracy Score:",accuracy_score(y_test, pred))
```

```
Accuracy Score: 0.15503875968992248
```