covid-19 visualization & prediction

In [1]:
```
#Importing required packages
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
#Invoke data from CSV file
covid_dataset=pd.read_csv('country_wise_latest.csv')
print(covid_dataset)
```

```
          Country/Region  Confirmed  Deaths  Recovered  Active  New cases  \
0            Afghanistan      36263    1269      25198    9796        106
1                Albania       4880     144       2745    1991        117
2                Algeria      27973    1163      18837    7973        616
3                Andorra        907      52        803      52         10
4                 Angola        950      41        242     667         18
..                   ...        ...     ...        ...     ...        ...
182   West Bank and Gaza      10621      78       3752    6791        152
183       Western Sahara         10       1          8       1          0
184                Yemen       1691     483        833     375         10
185               Zambia       4552     140       2815    1597         71
186             Zimbabwe       2704      36        542    2126        192

     New deaths  New recovered  Deaths / 100 Cases  Recovered / 100 Cases  \
0            10             18                3.50                  69.49
1             6             63                2.95                  56.25
2             8            749                4.16                  67.34
3             0              0                5.73                  88.53
4             1              0                4.32                  25.47
..          ...            ...                 ...                    ...
182           2              0                0.73                  35.33
183           0              0               10.00                  80.00
184           4             36               28.56                  49.26
185           1            465                3.08                  61.84
186           2             24                1.33                  20.04

     Deaths / 100 Recovered  Confirmed last week  1 week change  \
0                      5.04                35526            737
1                      5.25                 4171            709
2                      6.17                23691           4282
3                      6.48                  884             23
4                     16.94                  749            201
..                      ...                  ...            ...
182                    2.08                 8916           1705
183                   12.50                   10              0
184                   57.98                 1619             72
185                    4.97                 3326           1226
186                    6.64                 1713            991

     1 week % increase             WHO Region
0                 2.07  Eastern Mediterranean
1                17.00                 Europe
2                18.07                 Africa
3                 2.60                 Europe
4                26.84                 Africa
..                 ...                    ...
182              19.12  Eastern Mediterranean
183               0.00                 Africa
184               4.45  Eastern Mediterranean
185              36.86                 Africa
186              57.85                 Africa

[187 rows x 15 columns]
```

In [3]:
```
# Printing the first 5 rows from data set
covid_dataset.head()
```

Out[3]:

	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered	Confirmed last week	1 week change	1 week % increase	WHO Region
0	Afghanistan	36263	1269	25198	9796	106	10	18	3.50	69.49	5.04	35526	737	2.07	Eastern Mediterranean
1	Albania	4880	144	2745	1991	117	6	63	2.95	56.25	5.25	4171	709	17.00	Europe
2	Algeria	27973	1163	18837	7973	616	8	749	4.16	67.34	6.17	23691	4282	18.07	Africa
3	Andorra	907	52	803	52	10	0	0	5.73	88.53	6.48	884	23	2.60	Europe
4	Angola	950	41	242	667	18	1	0	4.32	25.47	16.94	749	201	26.84	Africa

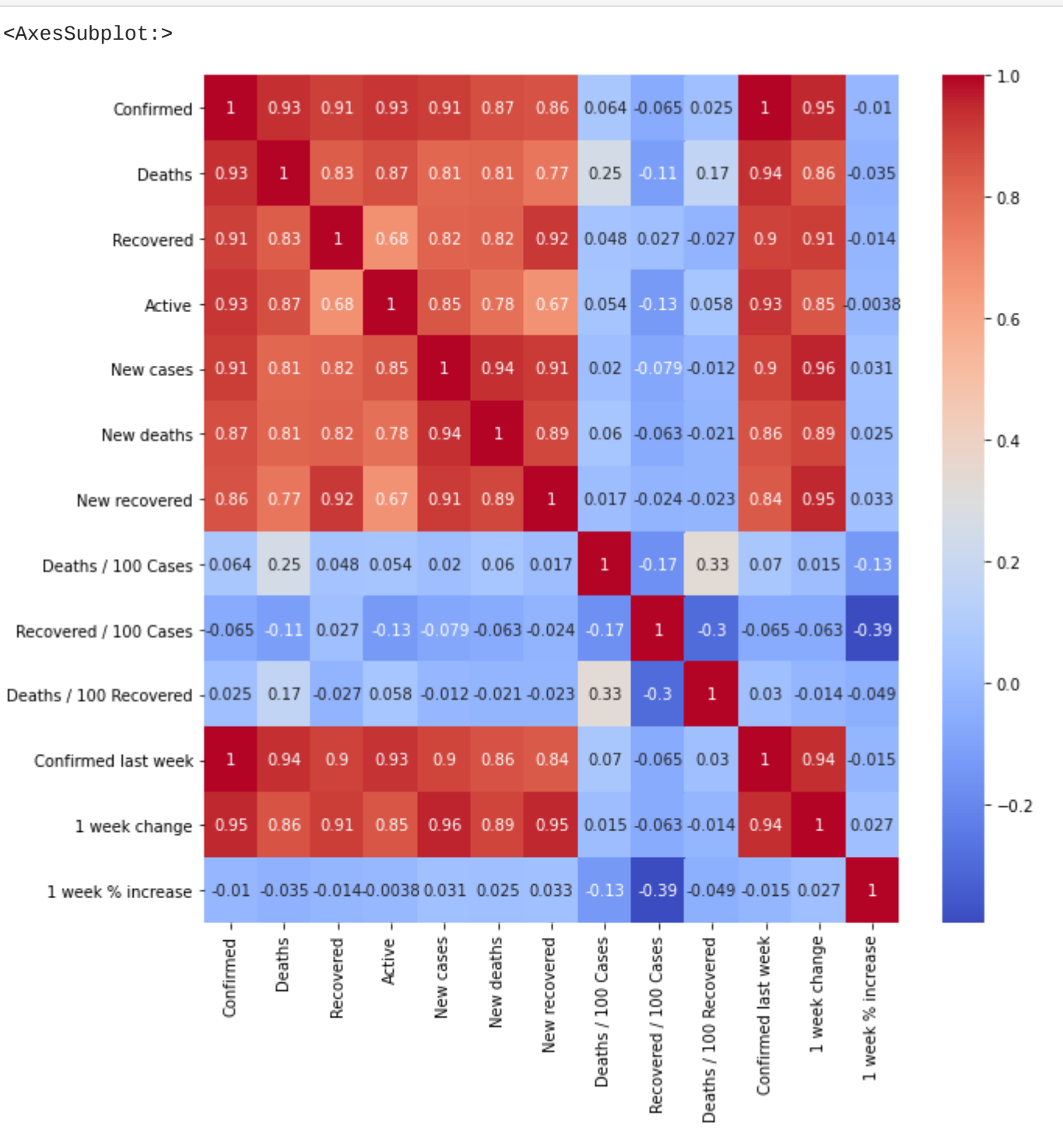
In [4]:
```
# checking the shape of data
covid_dataset.shape
```

Out[4]: (187, 15)

Data visualization

In [5]:
```
# Constructing a heatmap
correlation=covid_dataset.corr()
plt.figure(figsize=(10,10))
sns.heatmap(correlation,cbar=True,annot=True,cmap='coolwarm')
```

Out[5]: <AxesSubplot:>



preprocessing...

In [6]:
```
# Features selection
X=covid_dataset.iloc[:,1:5].values
y=covid_dataset.iloc[:,3].values
```

In [7]:
```
print(X)
```

```
[[  36263    1269   25198    9796]
 [   4880     144    2745    1991]
 [  27973    1163   18837    7973]
 [    907      52     803      52]
 [    950      41     242     667]
 [     86       3      65      18]
 [ 167416    3059   72575   91782]
 [  37390     711   26665   10014]
 [  15303     167    9311    5825]
 [  20558     713   18246    1599]
 [  30446     423   23242    6781]
 [    382      11      91     280]
 [  39482     141   36110    3231]
 [ 226225    2965  125683   97577]
 [    110       7      94       9]
 [  67251     538   60492    6221]
 [  66428    9822   17452   39154]
 [     48       2      26      20]
 [   1770      35    1036     699]
 [     99       0      86      13]
 [  71181    2647   21478   47056]
 [  10498     294    4930    5274]
 [    739       2      63     674]
 [2442375   87618 1846641  508116]
 [    141       3     138       0]
 [  10621     347    5585    4689]
 [   1100      53     926     121]
 [    350       6     292      52]
 [    378       1     301      76]
 [   2328      22    1550     756]
 [    226       0     147      79]
 [  17110     391   14539    2180]
 [ 116458    8944       0  107514]
 [   4599      59    1546    2994]
 [    922      75     810      37]
 [ 347923    9187  319954   18782]
 [  86783    4656   78869    3258]
 [ 257101    8777  131161  117163]
 [    354       7     328      19]
 [   3200      54     829    2317]
 [   8844     208    5700    2936]
 [  15841     115    3824   11902]
 [  15655      96   10361    5198]
 [   4881     139    3936     806]
 [   2532      87    2351      94]
 [   1060      19     852     189]
 [  15516     373   11428    3715]
 [  13761     613   12605     543]
 [   5059      58    4977      24]
 [     18       0      18       0]
 [  64156    1083   30204   32869]
 [  81161    5532   34896   40733]
 [  92482    4652   34838   52992]
 [  15035     408    7778    6849]
 [   3071      51     842    2178]
 [    265       0     191      74]
 [   2034      69    1923      42]
 [   2316      34    1025    1257]
 [  14547     228    6386    7933]
 [     27       0      18       9]
 [   7398     329    6920     149]
 [ 220352   30212   81212  108928]
 [   7189      49    4682    2458]
 [    326       8      66     252]
 [   1137      16     922     199]
 [ 207112    9125  190314    7673]
 [  33624     168   29801    3655]
 [   4227     202    1374    2651]
 [     14       0      13       1]
 [     23       0      23       0]
 [  45309    1761   32455   11093]
 [   7055      45    6257     753]
 [   1954      26     803    1125]
 [    389      20     181     188]
 [   7340     158    4365    2817]
 [     12       0      12       0]
 [  39741    1166    5039   33536]
 [   4448     596    3329     523]
 [   1854      10    1823      21]
 [1480073   33408  951166  495499]
 [ 100303    4838   58173   37292]
 [ 293606   15912  255144   22550]
 [ 112585    4458   77144   30983]
 [  25892    1764   23364     764]
 [  63985     474   27133   36378]
 [ 246286   35112  198593   12581]
 [    853      10     714     129]
 [  31142     998   21970    8174]
 [   1176      11    1041     124]
 [  84648     585   54404   29659]
 [  17975     285    7833    9857]
 [   7413     185    4027    3201]
 [  64379     438   55057    8884]
 [  33296    1301   21205   10790]
 [     20       0      19       1]
 [   1219      31    1045     143]
 [   3882      51    1709    2122]
 [    505      12     128     365]
 [   1167      72     646     449]
 [   2827      64     577    2186]
 [     86       1      81       4]
 [   2019      80    1620     319]
 [   6321     112    4825    1384]
 [   9690      91    6260    3339]
 [   3664      99    1645    1920]
 [   8904     124    8601     179]
 [   3369      15    2547     807]
 [   2513     124    1913     476]
 [    701       9     665      27]
 [   6208     156    4653    1399]
 [    344      10     332       2]
 [ 395489   44022  303810   47657]
 [  23154     748   16154    6252]
 [    116       4     104       8]
 [    289       0     222      67]
 [   2893      45     809    2039]
 [  20887     316   16553    4018]
 [   1701      11       0    1690]
 [   1843       8     101    1734]
 [  18752      48   13754    4950]
 [  53413    6160     189   47064]
 [   1557      22    1514      21]
 [   3439     108    2492     839]
 [   1132      69    1027      36]
 [  41180     860   18203   22117]
 [  10213     466    5564    4183]
 [   9132     255    8752     125]
 [  77058     393   57028   19637]
 [ 274289    5842  241026   27421]
 [  61442    1322   35086   25034]
 [     62       0      11      51]
 [   4548      43    2905    1600]
 [ 389717   18418  272547   98752]
 [  82040    1945   26446   53649]
 [  43402    1676   32856    8870]
 [  50299    1719   35375   13205]
 [ 109597     165  106328    3104]
 [  45902    2206   25794   17902]
 [ 816680   13334  602249  201097]
 [   1879       5     975     899]
 [     17       0      15       2]
 [     24       0      22       2]
 [     52       0      39      13]
 [    699      42     657       0]
 [    865      14     734     117]
 [ 268934    2760  222936   43238]
 [   9764     194    6477    3093]
 [  24141     543       0   23598]
 [    114       0      39      75]
 [   1783      66    1317     400]
 [  50838      27   45692    5119]
 [   2181      28    1616     537]
 [   2087     116    1733     238]
 [   3196      93    1543    1560]
 [ 452529    7067  274925  170537]
 [  14203     300   13007     896]
 [   2305      46    1175    1084]
 [ 272421   28432  150376   93613]
 [   2805      11    2121     673]
 [  11424     720    5939    4765]
 [   1483      24     925     534]
 [  79395    5700       0   73695]
 [  34477    1978   30900    1599]
 [    674      40       0     634]
 [    462       7     440      15]
 [   7235      60    6028    1147]
 [    509      21     183     305]
 [   3297      58    3111     128]
 [     24       0       0      24]
 [    874      18     607     249]
 [    148       8     128      12]
 [   1455      50    1157     248]
 [ 227019    5630  210469   10920]
 [4290259  148011 1325804 2816444]
 [   1128       2     986     140]
 [  67096    1636   37202   28258]
 [  59177     345   52510    6322]
 [ 301708   45844    1437  254427]
 [   1202      35     951     216]
 [  21209     121   11674    9414]
 [  15988     146    9959    5883]
 [    431       0     365      66]
 [  10621      78    3752    6791]
 [     10       1       8       1]
 [   1691     483     833     375]
 [   4552     140    2815    1597]
 [   2704      36     542    2126]]
```

In [8]:
```
print(y)
```

```
[  25198    2745   18837     803     242      65   72575   26665    9311
   18246   23242      91   36110  125683      94   60492   17452      26
    1036      86   21478    4930      63 1846641     138    5585     926
     292     301    1550     147   14539       0    1546     810  319954
   78869  131161     328     829    5700    3824   10361    3936    2351
     852   11428   12605    4977      18   30204   34896   34838    7778
     842     191    1923    1025    6386      18    6920   81212    4682
      66     922  190314   29801    1374      13      23   32455    6257
     803     181    4365      12    5039    3329    1823  951166   58173
  255144   77144   23364   27133  198593     714   21970    1041   54404
    7833    4027   55057   21205      19    1045    1709     128     646
     577      81    1620    4825    6260    1645    8601    2547    1913
     665    4653     332  303810   16154     104     222     809   16553
       0     101   13754     189    1514    2492    1027   18203    5564
    8752   57028  241026   35086      11    2905  272547   26446   32856
   35375  106328   25794  602249     975      15      22      39     657
     734  222936    6477       0      39    1317   45692    1616    1733
    1543  274925   13007    1175  150376    2121    5939     925       0
   30900       0     440    6028     183    3111       0     607     128
    1157  210469 1325804     986   37202   52510    1437     951   11674
    9959     365    3752       8     833    2815     542]
```

In [9]:
```
# Splitting the data into training data and testing data
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=10)
```

using Random forest Regressor

In [10]:
```
# Loading Algorithm module
from sklearn.ensemble import RandomForestRegressor
Reg=RandomForestRegressor(n_estimators=5)
```

In [11]:
```
# Fitting training data
Reg.fit(X_train,y_train)
```

Out[11]: RandomForestRegressor(n_estimators=5)

In [12]:
```
# Prediction on Test data
y_pre=Reg.predict(X_test)
```

Accuracy score for Random forest Regressor

In [13]:
```
from sklearn.metrics import r2_score
acc_score=r2_score(y_test,y_pre)
print (acc_score)
```

0.9997451779207741

Using KNN Regressor

In [14]:
```
# Loading Algorithm module
from sklearn.neighbors import KNeighborsRegressor
Reg=KNeighborsRegressor(n_neighbors=5)
```

In [15]:
```
# Fitting training data
Reg.fit(X_train,y_train)
```

Out[15]: KNeighborsRegressor()

In [16]:
```
#Prediction on test data
y_pre=Reg.predict(X_test)
```

Accuracy score for KNN Regressor

In [17]:
```
from sklearn.metrics import r2_score
acc_score=r2_score(y_test,y_pre)
print (acc_score)
```

0.994659048268545

Using Linear Regressor

In [18]:
```
# Loading Algoritm module
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
```

In [19]:
```
 Fit
reg.fit(X_train,y_train)
```

Out[19]: LinearRegression()

In [20]:
```
#Prediction on test data
y_pre=reg.predict(X_test)
```

Accuracy score for Linear Regressor

In [21]:
```
from sklearn.metrics import r2_score
acc_score=r2_score(y_test,y_pre)
print(acc_score)
```

1.0

CONCLUSION:

Random Forest Regressor = 99%

K Neighbors Regressor =99%

Linear Regression = 100%

In [ ]: