Available online at: http://www.ijmtst.com/vol6issue04.html

International Journal for Modern Trends in Science and Technology

ISSN: 2455-3778 :: Volume: 06, Issue No: 04, April 2020

Steps to predict the Coronavirus disease (COVID-19) – AI Machine Learning

Subash Kumar

Bachelor of Engineering Computer Science, Anna University, Chennai, India.

To Cite this Article

Subash Kumar, "Steps to predict the Coronavirus disease (COVID-19) – AI Machine Learning", *International Journal for Modern Trends in Science and Technology*, Vol. 06, Issue 04, April 2020, pp.:257-260.

Article Info

Received on 20-March-2020, Revised on 16-April-2020, Accepted on 20-April-2020, Published on 22-April-2020.

ABSTRACT

Coronavirus disease (COVID-19) is an infectious disease caused by a newly discovered coronavirus.Most people infected with the COVID-19 virus will experience mild to moderate respiratory illness and recover without requiring special treatment. Older people, and those with underlying medical problems like cardiovascular disease, diabetes, chronic respiratory disease, and cancer are more likely to develop serious illness.The best way to prevent and slow down transmission is be well informed about the COVID-19 virus, the disease it causes and how it spreads. In this paper let us discuss about the steps to predict the spread of coronavirus across globe

.

KEYWORDS: *Learning, Deep Learning, Artificial Intelligence, Medicine, COVID-19, Coronavirus, disease Data Science, Regression, Supervised Machine Learning*

I. INTRODUCTION

In this paper, we will use 3 attributes (Confirmed, Death, and Recovered) cases, the data is obtained from various sources across globe

II. METHODOLOGY

Importing the libraries from Python

Importing all the required libraries from python that includes

1. Numpy
2. Pandas
3. Matplotlib pyplot

```
# COVID PREDICTIONS COUNT
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Obtaining data from Novel Coronavirus (COVID-19) Cases, provided by Johns Hopkins University

```
confirmed_cases = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_se
confirmed_df = pd.read_csv(confirmed_cases,error_bad_lines=False)
death_cases = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/
death_df = pd.read_csv(death_cases,error_bad_lines=False)
recover_cases = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_serie
recover_df = pd.read_csv(recover_cases,error_bad_lines=False)
population_df = pd.read_csv('C:\\Users\\skumarlocal\\Desktop\\COVID\\population.csv',sep=',', encoding='latin1')
confirmed_population_df = pd.merge(confirmed_df,population_df,how='left',on=['Province/State','Country/Region'])
death_population_df = pd.merge(death_df,population_df,how='left',on=['Province/State','Country/Region'])
recover_population_df = pd.merge(recover_df,population_df,how='left',on=['Province/State','Country/Region'])
confirmed_population_df['region_merged'] = confirmed_population_df['Country/Region'].map(str)+'_'+confirmed_population_df['Pr
death_population_df['region_merged'] = death_population_df['Country/Region'].map(str)+'_'+death_population_df['Province/State
recover_population_df['region_merged'] = recover_population_df['Country/Region'].map(str)+'_'+recover_population_df['Province
```

Get the population data from Github for our analysis

Get Population

```
In [8]: # read population data for each province. China is divided into region whereas other regions of the world is nation
population=pd.read_csv('https://raw.githubusercontent.com/Rank23/COVID19/master/population.csv', sep=',', encoding='latin1')
population.head()
```

Out[5]:

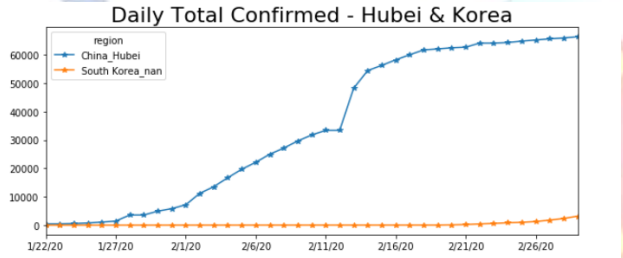
	Province/State	Country/Region	Population
0	Anhui	China	62,000,000
1	Beijing	China	21,710,000
2	Chongqing	China	7,990,000
3	Fujian	China	36,894,216
4	Gansu	China	25,575,254

Now merge the Population data with the confirmed/recovered/death data

```
confirmed=pd.merge(confirmed, population,how='left' ,on=['Province/State','Country/Region'])
death=pd.merge(death, population,how='left' ,on=['Province/State','Country/Region'])
recover=pd.merge(recover, population,how='left' ,on=['Province/State','Country/Region'])
confirmed.head()
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	2/1/20	2/2/20	2/
0	Anhui	China	31.8257	117.2264	1	9	15	39	60	70	106	152	200	237	297	340	
1	Beijing	China	40.1824	116.4142	14	22	36	41	68	80	91	111	114	139	168	191	
2	Chongqing	China	30.0572	107.8740	6	9	27	57	75	110	132	147	182	211	247	300	
3	Fujian	China	26.0789	117.9874	1	5	10	18	35	59	80	84	101	120	144	159	
4	Gansu	China	36.0611	103.8343	0	2	2	4	7	14	19	24	26	29	40	51	

Let us create the timeseries plot from the above datapoints



We can see a clear eruption trend in Hubei with exceptional day 13.02.20 of a high jump. On this day there was a change in the counting method and it will later affect the model. The above plot compare the data between China Hubei with South Korea region

Let us create a plot on the recovered cases in Hubei

```
# plot graph of hubei's recovered casses and other major areas
p_r=ts_rec.reindex(ts.mean().sort_values(ascending=False).index, axis=1)
p_r.iloc[:,:1].plot(marker='*',figsize=(10,4)).set_title('Daily Total Recoverd - Hubei',fontdict={'fontsize': 22})
p_r.iloc[:,1:10].plot(marker='*',figsize=(10,4)).set_title('Daily Total Recoverd - Major areas',fontdict={'fontsize': 22})
```

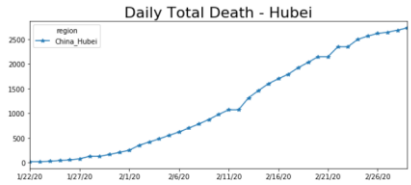
```
Text(0.5, 1.0, 'Daily Total Recoverd - Major areas')
```

Sharp trend of recovered cases in all areas, mostly in Hubei which can imply us for a positive future

Let us create a Plot for the death cases

```
# plot graph of hubei's deceased casses and other major areas
p_d=ts_d.reindex(ts.mean().sort_values(ascending=False).index, axis=1)
p_d.iloc[:,:1].plot(marker='*',figsize=(10,4)).set_title('Daily Total Death - Hubei',fontdict={'fontsize': 22})
p_d.iloc[:,1:10].plot(marker='*',figsize=(10,4)).set_title('Daily Total Death - Major areas',fontdict={'fontsize': 22})
```

```
Text(0.5, 1.0, 'Daily Total Death - Major areas')
```



Death cases are mostly in Hubei with more than 2,000. After that, Henan with 19. Death rate in Hubei is 2.1978% and 1.5055% in Henan. In the rest of the areas, the numbers are too small.

Applying the Kalman prediction in Python and see what is the prediction the model shows

We are importing the region data that we already pulled into the dataframe in the above step

```
region='South Korea_nan'

evaluation=pd.DataFrame(columns=['region','mse','rmse','mae'])
place=0
```

```
for i in range(1,len(t)):
  if(t.iloc[i,1] is not t.iloc[i-1,1]):
    ex=np.array(t.iloc[i-len(ts):i,10])
    # print(ex)
    pred=np.array(t.iloc[i-len(ts):i,2])
    # print(pred)
    evaluation=evaluation.append({'region': t.iloc[i-1,1], 'mse': np.power((ex - pred),2).mean(),'rmse':sqrt(mean_squared_er
    # print(evaluation)
```

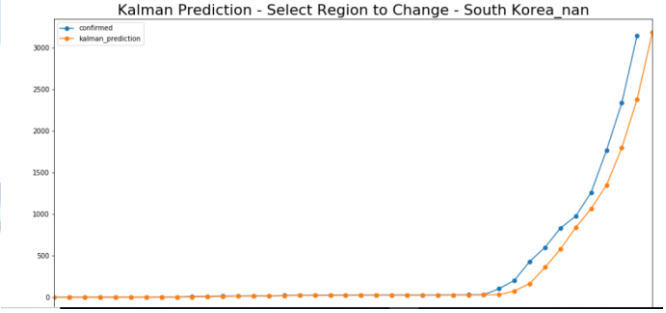
```
p=t[t['region']==region][['date','region','confirmed','kalman_prediction']]
p.tail(10)
```

	date	region	confirmed	kalman_prediction
3630	2020-02-21	South Korea_nan	204	77.0
3631	2020-02-22	South Korea_nan	433	166.0
3632	2020-02-23	South Korea_nan	602	361.0
3633	2020-02-24	South Korea_nan	833	579.0
3634	2020-02-25	South Korea_nan	977	841.0

After successfully fitting the data into the model, we are observing the following plot

```
p.iloc[:,1:].plot(marker='o',figsize=(16,8)).set_title('Kalman Prediction - Select Region to Change - {}'.format(p.iloc[0,0
print(evaluation[evaluation['region']==p.iloc[0,0]])
```

	region	mse	rmse	mae
90	South Korea_nan	294578.102564	542.750498	160.0

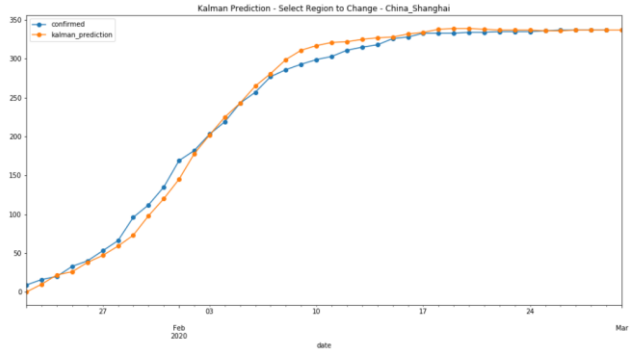


```
# Select region
region='China_Shanghai'

evaluation=pd.DataFrame(columns=['region','mse','rmse','mae'])
place=0
for i in range(1,len(t)):
  if(t.iloc[i,1] is not t.iloc[i-1,1]):
    ex=np.array(t.iloc[i-len(ts):i,10])
    pred=np.array(t.iloc[i-len(ts):i,2])
    evaluation=evaluation.append({'region': t.iloc[i-1,1], 'mse': np.power((ex - pred),2).mean(),'rmse':sqrt(mean_squared_er
p=t[t['region']==region][['date','region','confirmed','kalman_prediction']]
p.iloc[len(p)-1,2]=None
p=p.set_index(['date'])
p.iloc[:,1:].plot(marker='o',figsize=(16,8)).set_title('Kalman Prediction - Select Region to Change - {}'.format(p.iloc[0,0]
print(evaluation[evaluation['region']==p.iloc[0,0]])
```

The below plot is for the china

```
          region          mse        rmse        mae
41  China_Shanghai  2999.615385  54.768745  15.307692
```



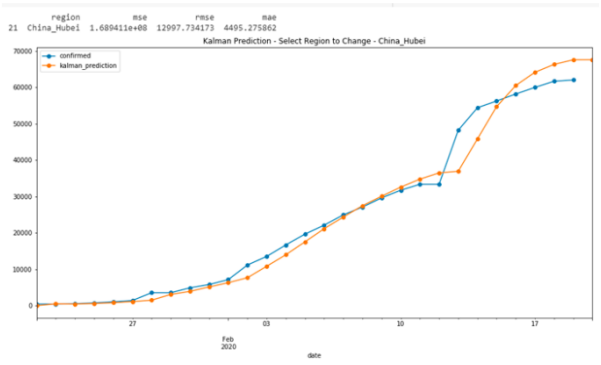
The table below shows how close are the predictions to the actual values. For example, on 19.02.20 Kalman predicts 394 cases which are 7 new confirmed cases while there were actual 6. For tomorrow (20.02.20) Kalman predicts 5 new confirmed cases in Beijing.

date	confirmed	kalman_prediction
2020-02-06	274.0	272.0
2020-02-07	297.0	295.0
2020-02-08	315.0	318.0
2020-02-09	326.0	338.0
2020-02-10	337.0	352.0
2020-02-11	342.0	363.0
2020-02-12	352.0	368.0
2020-02-13	366.0	372.0
2020-02-14	372.0	381.0
2020-02-15	375.0	387.0
2020-02-16	380.0	389.0
2020-02-17	381.0	392.0
2020-02-18	387.0	392.0
2020-02-19	393.0	394.0
2020-02-20	NaN	398.0

Daily confirmed prediction vs. actual — Beijing

B. **Hubei confirmed** cases — here we see almost perfect prediction until the 13.02.20, then since the jump in the data that was mentioned above the model had a larger error. But it is adapting fast and, in a few days, will gain better predictions.

```
          region          mse          rmse         mae
21  China_Hubei1  1.689411e+08  12997.734173  4495.275862
```



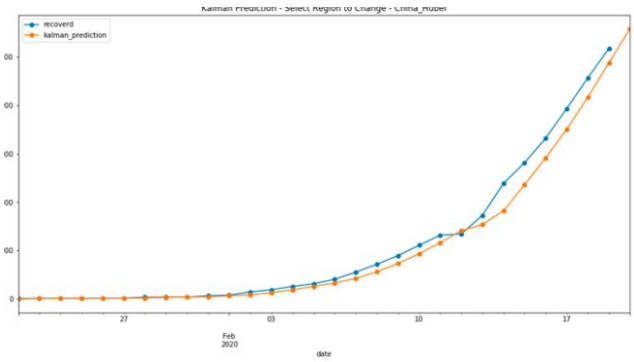
Kalman prediction vs actual confirmed cases in Hubei

The table below shows the predictions and the actual **death** in numbers, in **Hubei**. For example, on 18.02 Kalman predicted 140 more death where there were eventually 132. On 19.02 Kalman predict 129 more death where there were eventually 108. For tomorrow (20.02) Kalman predicts another 132 new death cases. The predictions are very close to the real values.

date	death	kalman_prediction
2020-02-06	618.0	597.0
2020-02-07	699.0	676.0
2020-02-08	780.0	759.0
2020-02-09	871.0	844.0
2020-02-10	974.0	936.0
2020-02-11	1068.0	1039.0
2020-02-12	1068.0	1142.0
2020-02-13	1310.0	1185.0
2020-02-14	1457.0	1341.0
2020-02-15	1596.0	1509.0
2020-02-16	1696.0	1677.0
2020-02-17	1789.0	1815.0
2020-02-18	1921.0	1929.0
2020-02-19	2029.0	2050.0
2020-02-20	NaN	2161.0

Daily death prediction vs. actual — Hubei

Recovered daily predictions: We can get some positive insights as to the sharp trend of recovered cases in **Hubei**. The Kalman prediction adapts this trend and predicts an increasing number of recovered cases recently



Kalman prediction vs actual recovered cases in Hubei

III. RESULTS

To evaluate results, I've added some basic error estimator parameters for each region: MSE — mean square error, RMSE — root mean square error, MAE — mean absolute error. I compared them with other methods to optimize the models and find the best one. The model can also be evaluated with plots and tables as shown below. The script allows us to select regions and get a suitable plot and predictions. It was run iteratively for each region

REFERENCES

[1] https://www.who.int/emergencies/diseases/novel-coronavirus-2019
[2] https://towardsdatascience.com/using-kalman-filter-to-predict-corona-virus-spread-72d91b74cc8
[3] https://www.nytimes.com/2020/03/24/world/coronavirus-updates-maps.html