Covid-19 Forecasting using an RNN

1. Introduction

The goal of this notebook is very simple: Generate additional features from the Covid19-global-forecasting dataset and feed it into an RNN. The RNN will take as inputs:

- number of cases for 20 days
- number of fatalities for 20 days
- restrictions applied for the area in the past 20 days
- quarantine applied for the area in the past 20 days
- school opened or closed for the area in the past 20 days
- additional information related to the area (population, density, number of hospital beds, lung measurement, number of centenarian people)

as outputs:

- number of cases for the 21th day
- number of fatalities for the 21th day

Update notes

- **V5**: Submission pipeline fixed - *score: 3.67449*
- **V6**: New RNN architecture with two separate branches for each output - *score: 3.26619*
- **V8**: Add a post-processing step checking if the model's output is equal or greater the previous value - *score: 2.73282*
- **V9**: Change the MSE losses to RMSLE - *score: 1.76123*
- **V11**: Change the outputs' activation fucntions from linear to ReLU - *score: 1.60605*
- **V12**: Use a 2-week period for predictions instead of 1. Replace the SimpleRNN layers with LSTM layers - *score: 1.26466*
- **V13**: Fix bug in Cell 4 (flagged by @jeremyoudin) which made the dataset much larger due to duplicates and also created a leakage between the training and validation sets - *score: 1.46697*
- **V15**: Start introducing new inputs from the enriched dataset by @optimo and fix a bug which prevented the generation of 25% of the training data - *score: 1.03853*
- **V16**: Add a new input to the model based on the coutries' total population. It is added as a separate input as it is not temporal like all previous inputs - *score: 1.02947*
- **V19**: Use additional inputs using the enriched dataset. It makes the model more complex and harder to train but might be worth it! *score: 1.05385*
- **V20**: Allow a 50% overlap between sequences when generating the dataset to double its size. It creates a minor leakage between the training and validation set so it will eventually needs to be fixed. I was curious to see how it would affect performances. Beside that, it appears that the model's training phase is not perfectly stable, with the "fatality" branch not always converging. *score: 1.08343*
- **V21**: Add new figures across the notebook and a function at the end of the notebook to visualise the predictions over the period of time defined for the Public Leaderboard. - *score: 1.02837*
- **V26**: Update the notebook to compare with the latest data, fix minor errors, new charts to visualise long term predictions, and add a conclusion at the end.
- **V27**: Now running on data from COVID19 Global Forecasting (Week 3) and all dates for predictions have been updated accordingly.
- **V31**: Fixed a bug in the way inputs are fed to the model (flagged by @SebastianPine). The model's ability to learn sequential information from the inputs should be dramatically increased.
- **V32**: Add data about the number of centenarian people among the countries/regions' populations in the inputs to the model.
- **V36**: Move to Week 4 to benefit from the additional data available! *score: 0.62611*
- **V38**: Attempt to use information from the 3 previous weeks for predictions instead of 2 weeks. It is motivated by the fact that we have more data available than in Week 2 and 3 already.

PS: Please note that the scores may have change slightly. I assume that the evaluation process is updated as new data is coming in for the period from 19/03/2020 to 01/04/2020. Moreover, most of the scores were on Week 2 of this competition.

In [1]:

```
import numpy as np
import pandas as pd
import geopandas as gpd
from shapely.geometry import Point
import os
import tensorflow as tf
from tqdm import tqdm
from sklearn.utils import shuffle
from sklearn.metrics import mean_squared_log_error

from datetime import datetime
from datetime import timedelta

from tensorflow.keras import layers
from tensorflow.keras import Input
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping

import matplotlib.pyplot as plt
import seaborn as sns
sns.set()


for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/covid19-global-forecasting-week-4/train.csv
/kaggle/input/covid19-global-forecasting-week-4/submission.csv
/kaggle/input/covid19-global-forecasting-week-4/test.csv
/kaggle/input/enriched-covid-19-week2/enriched_covid_19_week_2.csv
```

2. Preparing the training data

In [2]:

```
train_df = gpd.read_file("/kaggle/input/covid19-global-forecasting-week-4/train.csv")
train_df["ConfirmedCases"] = train_df["ConfirmedCases"].astype("float")
train_df["Fatalities"] = train_df["Fatalities"].astype("float")
#The country_region got modified in the enriched dataset by @optimo,
# so we have to apply the same change to this Dataframe to facilitate the merge.
train_df["Country_Region"] = [ row.Country_Region.replace("'","").strip(" ") if row.Province_State==""
else str(row.Country_Region+"_"+row.Province_State).replace("'","").strip(" ") for idx,row in train_df.
iterrows()]
```

In [3]:

```
#Still using the enriched data from week 2 as there is everything required for the model's training
extra_data_df = gpd.read_file("/kaggle/input/enriched-covid-19-week2/enriched_covid_19_week_2.csv")
extra_data_df["Country_Region"] = [country_name.replace("'","") for country_name in extra_data_df["Coun
try_Region"]]
extra_data_df["restrictions"] = extra_data_df["restrictions"].astype("int")
extra_data_df["quarantine"] = extra_data_df["quarantine"].astype("int")
extra_data_df["schools"] = extra_data_df["schools"].astype("int")
extra_data_df["total_pop"] = extra_data_df["total_pop"].astype("float")
extra_data_df["density"] = extra_data_df["density"].astype("float")
extra_data_df["hospibed"] = extra_data_df["hospibed"].astype("float")
extra_data_df["lung"] = extra_data_df["lung"].astype("float")
extra_data_df["total_pop"] = extra_data_df["total_pop"]/max(extra_data_df["total_pop"])
extra_data_df["density"] = extra_data_df["density"]/max(extra_data_df["density"])
extra_data_df["hospibed"] = extra_data_df["hospibed"]/max(extra_data_df["hospibed"])
extra_data_df["lung"] = extra_data_df["lung"]/max(extra_data_df["lung"])
extra_data_df["age_100+"] = extra_data_df["age_100+"].astype("float")
extra_data_df["age_100+"] = extra_data_df["age_100+"]/max(extra_data_df["age_100+"])

extra_data_df = extra_data_df[["Country_Region","Date","restrictions","quarantine","schools","hospibed"
,"lung","total_pop","density","age_100+"]]
extra_data_df.head()
```

Out[3]:

	Country_Region	Date	restrictions	quarantine	schools	hospibed	lung	total_pop	density	age_100+
0	Afghanistan	2020-01-22	0	0	0	0.036232	0.329191	0.027046	0.002278	0.001411
1	Afghanistan	2020-01-23	0	0	0	0.036232	0.329191	0.027046	0.002278	0.001411
2	Afghanistan	2020-01-24	0	0	0	0.036232	0.329191	0.027046	0.002278	0.001411
3	Afghanistan	2020-01-25	0	0	0	0.036232	0.329191	0.027046	0.002278	0.001411
4	Afghanistan	2020-01-26	0	0	0	0.036232	0.329191	0.027046	0.002278	0.001411

In [4]:

```
train_df = train_df.merge(extra_data_df, how="left", on=['Country_Region','Date']).drop_duplicates()
train_df.head()
```

Out[4]:

	Id	Province_State	Country_Region	Date	ConfirmedCases	Fatalities	geometry	restrictions	quarantine	schools	hospibed	lung
0	1		Afghanistan	2020-01-22	0.0	0.0	None	0.0	0.0	0.0	0.036232	0.329191
1	2		Afghanistan	2020-01-23	0.0	0.0	None	0.0	0.0	0.0	0.036232	0.329191
2	3		Afghanistan	2020-01-24	0.0	0.0	None	0.0	0.0	0.0	0.036232	0.329191
3	4		Afghanistan	2020-01-25	0.0	0.0	None	0.0	0.0	0.0	0.036232	0.329191
4	5		Afghanistan	2020-01-26	0.0	0.0	None	0.0	0.0	0.0	0.036232	0.329191

Still need to complete part of the data for dates past the 25th of March as the enriched dataset didn't go that far.

In [5]:

```
for country_region in train_df.Country_Region.unique():
    query_df = train_df.query("Country_Region=='"+country_region+"' and Date=='2020-03-25'")
    train_df.loc[(train_df["Country_Region"]==country_region) & (train_df["Date"]>"2020-03-25"),"total_
pop"] = query_df.total_pop.values[0]
    train_df.loc[(train_df["Country_Region"]==country_region) & (train_df["Date"]>"2020-03-25"),"hospib
ed"] = query_df.hospibed.values[0]
    train_df.loc[(train_df["Country_Region"]==country_region) & (train_df["Date"]>"2020-03-25"),"densit
y"] = query_df.density.values[0]
    train_df.loc[(train_df["Country_Region"]==country_region) & (train_df["Date"]>"2020-03-25"),"lung"]
 = query_df.lung.values[0]
    train_df.loc[(train_df["Country_Region"]==country_region) & (train_df["Date"]>"2020-03-25"),"age_10
0+"] = query_df["age_100+"].values[0]
    train_df.loc[(train_df["Country_Region"]==country_region) & (train_df["Date"]>"2020-03-25"),"restri
ctions"] = query_df.restrictions.values[0]
    train_df.loc[(train_df["Country_Region"]==country_region) & (train_df["Date"]>"2020-03-25"),"quaran
tine"] = query_df.quarantine.values[0]
    train_df.loc[(train_df["Country_Region"]==country_region) & (train_df["Date"]>"2020-03-25"),"school
s"] = query_df.schools.values[0]
```

As I was using an enriched dataset during the Week 2 competition, I have to add the new countries to my dataframe and fill the missing data with median values.

In [6]:

```
median_pop = np.median(extra_data_df.total_pop)
median_hospibed = np.median(extra_data_df.hospibed)
median_density = np.median(extra_data_df.density)
median_lung = np.median(extra_data_df.lung)
median_centenarian_pop = np.median(extra_data_df["age_100+"])
#need to replace that with a joint using Pandas
print("The missing countries/region are:")
for country_region in train_df.Country_Region.unique():
    if extra_data_df.query("Country_Region=='"+country_region+"'").empty:
        print(country_region)

        train_df.loc[train_df["Country_Region"]==country_region,"total_pop"] = median_pop
        train_df.loc[train_df["Country_Region"]==country_region,"hospibed"] = median_hospibed
        train_df.loc[train_df["Country_Region"]==country_region,"density"] = median_density
        train_df.loc[train_df["Country_Region"]==country_region,"lung"] = median_lung
        train_df.loc[train_df["Country_Region"]==country_region,"age_100+"] = median_centenarian_pop
        train_df.loc[train_df["Country_Region"]==country_region,"restrictions"] = 0
        train_df.loc[train_df["Country_Region"]==country_region,"quarantine"] = 0
        train_df.loc[train_df["Country_Region"]==country_region,"schools"] = 0
```

```
The missing countries/region are:
Botswana
Burma
Burundi
Canada_Northwest Territories
Canada_Yukon
France_Saint Pierre and Miquelon
Kosovo
MS Zaandam
Malawi
Netherlands_Bonaire, Sint Eustatius and Saba
Sao Tome and Principe
Sierra Leone
South Sudan
United Kingdom_Anguilla
United Kingdom_British Virgin Islands
United Kingdom_Falkland Islands (Malvinas)
United Kingdom_Turks and Caicos Islands
West Bank and Gaza
Western Sahara
```

I create a new dataframe where I will only store 13-day trends for each location with the resulting numbers on the 14th day. The time periods extracted do not overlap on purpose.

In [7]:

```
trend_df = pd.DataFrame(columns={"infection_trend","fatality_trend","quarantine_trend","school_trend",
"total_population","expected_cases","expected_fatalities"})
```

In [8]:

```
#Just getting rid of the first days to have a multiple of 7
#Makes it easier to generate the sequences
train_df = train_df.query("Date>'2020-01-22'and Date<'2020-04-01'")
days_in_sequence = 21

trend_list = []

with tqdm(total=len(list(train_df.Country_Region.unique()))) as pbar:
    for country in train_df.Country_Region.unique():
        for province in train_df.query(f"Country_Region=='{country}'").Province_State.unique():
            province_df = train_df.query(f"Country_Region=='{country}' and Province_State=='{province}
'")

            #I added a quick hack to double the number of sequences
            #Warning: This will later create a minor leakage from the
            # training set into the validation set.
            for i in range(0,len(province_df),int(days_in_sequence/3)):
                if i+days_in_sequence<=len(province_df):
                    #prepare all the temporal inputs
                    infection_trend = [float(x) for x in province_df[i:i+days_in_sequence-1].ConfirmedC
ases.values]
                    fatality_trend = [float(x) for x in province_df[i:i+days_in_sequence-1].Fatalities.
values]
                    restriction_trend = [float(x) for x in province_df[i:i+days_in_sequence-1].restrict
ions.values]
                    quarantine_trend = [float(x) for x in province_df[i:i+days_in_sequence-1].quarantin
e.values]
                    school_trend = [float(x) for x in province_df[i:i+days_in_sequence-1].schools.value
s]

                    #preparing all the demographic inputs
                    total_population = float(province_df.iloc[i].total_pop)
                    density = float(province_df.iloc[i].density)
                    hospibed = float(province_df.iloc[i].hospibed)
                    lung = float(province_df.iloc[i].lung)
                    centenarian_pop = float(province_df.iloc[i]["age_100+"])

                    expected_cases = float(province_df.iloc[i+days_in_sequence-1].ConfirmedCases)
                    expected_fatalities = float(province_df.iloc[i+days_in_sequence-1].Fatalities)

                    trend_list.append({"infection_trend":infection_trend,
                                     "fatality_trend":fatality_trend,
                                     "restriction_trend":restriction_trend,
                                     "quarantine_trend":quarantine_trend,
                                     "school_trend":school_trend,
                                     "demographic_inputs":[total_population,density,hospibed,lung,cente
narian_pop],
                                     "expected_cases":expected_cases,
                                     "expected_fatalities":expected_fatalities})
        pbar.update(1)
trend_df = pd.DataFrame(trend_list)
```

```
100%|██████████| 313/313 [00:16<00:00, 18.77it/s]
```

Preparing the inputs and shuffling the dataframe to make sure we have a bit of everything in our training and validation set.

In [9]:

```
trend_df["temporal_inputs"] = [np.asarray([trends["infection_trend"],trends["fatality_trend"],trends["r
estriction_trend"],trends["quarantine_trend"],trends["school_trend"]]) for idx,trends in trend_df.iterr
ows()]

trend_df = shuffle(trend_df)
```

In [10]:

```
trend_df.head()
```

Out[10]:

	infection_trend	fatality_trend	restriction_trend	quarantine_trend	school_trend	demographic_inputs	expected_cases	expected_f
2099	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.025933259892085968, 0.003607092683297262, 0...	0.0	
1400	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.026294712616898665, 0.004708205186619584, 0...	0.0	
1451	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.00012758005065787237, 0.011428788396552378,...	0.0	
1900	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.2322267255206195, 0.003607092683297262, 0.1...	0.0	
534	[18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18....	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[1.0, 0.003607092683297262, 0.1956521739130435...	18.0	

Only keeping 25 sequences where the number of cases stays at 0, as there were way too many of these samples in our dataset.

In [11]:

```
i=0
temp_df = pd.DataFrame()
for idx,row in trend_df.iterrows():
    if sum(row.infection_trend)>0:
        temp_df = temp_df.append(row)
    else:
        if i<25:
            temp_df = temp_df.append(row)
            i+=1
trend_df = temp_df
```

In [12]:

```
trend_df.head()
```

Out[12]:

	demographic_inputs	expected_cases	expected_fatalities	fatality_trend	infection_trend	quarantine_trend	restriction_trend	scho
2099	[0.025933259892085968, 0.003607092683297262, 0...	0.0	0.0	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0,
1400	[0.026294712616898665, 0.004708205186619584, 0...	0.0	0.0	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0,
1451	[0.00012758005065787237, 0.011428788396552378,...	0.0	0.0	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0,
1900	[0.2322267255206195, 0.003607092683297262, 0.1...	0.0	0.0	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0,
534	[1.0, 0.003607092683297262, 0.1956521739130435...	18.0	0.0	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18....	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0,

Splitting my dataset with 90% for training and 10% for validation

In [13]:

```
sequence_length = 20
training_percentage = 0.9
```

In [14]:

```
training_item_count = int(len(trend_df)*training_percentage)
validation_item_count = len(trend_df)-int(len(trend_df)*training_percentage)
training_df = trend_df[:training_item_count]
validation_df = trend_df[training_item_count:]
```

In [15]:

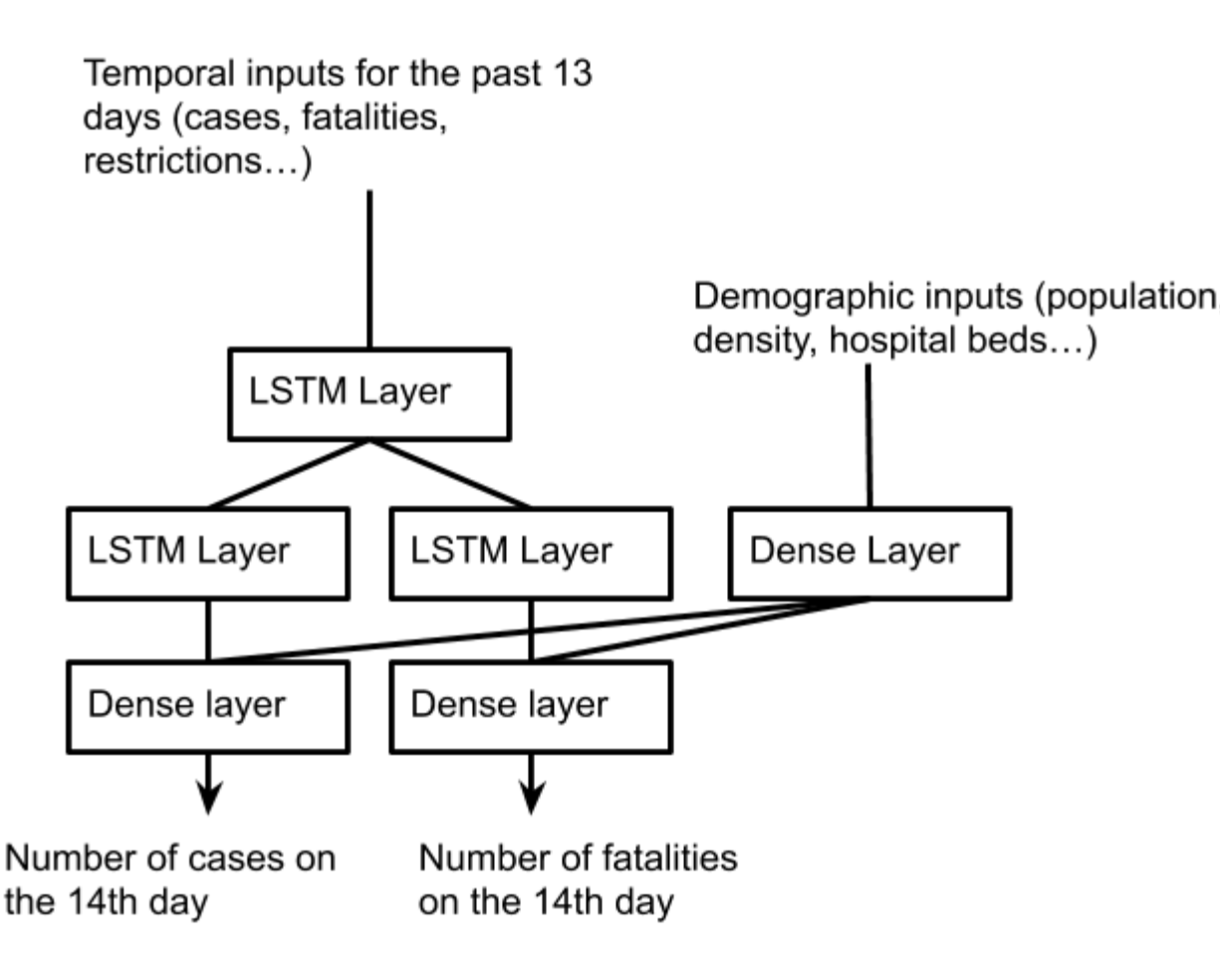
```
X_temporal_train = np.asarray(np.transpose(np.reshape(np.asarray([np.asarray(x) for x in training_df["t
emporal_inputs"].values]),(training_item_count,5,sequence_length)),(0,2,1) )).astype(np.float32)
X_demographic_train = np.asarray([np.asarray(x) for x in training_df["demographic_inputs"]]).astype(np.
float32)
Y_cases_train = np.asarray([np.asarray(x) for x in training_df["expected_cases"]]).astype(np.float32)
Y_fatalities_train = np.asarray([np.asarray(x) for x in training_df["expected_fatalities"]]).astype(np.
float32)
```

In [16]:

```
X_temporal_test = np.asarray(np.transpose(np.reshape(np.asarray([np.asarray(x) for x in validation_df[
"temporal_inputs"]]),(validation_item_count,5,sequence_length)),(0,2,1)) ).astype(np.float32)
X_demographic_test = np.asarray([np.asarray(x) for x in validation_df["demographic_inputs"]]).astype(np
.float32)
Y_cases_test = np.asarray([np.asarray(x) for x in validation_df["expected_cases"]]).astype(np.float32)
Y_fatalities_test = np.asarray([np.asarray(x) for x in validation_df["expected_fatalities"]]).astype(np
.float32)
```

3. Build the model

The model is very simple in terms of architecture. The only difference from what could traditionally be seen is that it has two outputs so we can have two different losses (one for the expected number of cases and for the expected number of fatalities).



In [17]:

```
#temporal input branch
temporal_input_layer = Input(shape=(sequence_length,5))
main_rnn_layer = layers.LSTM(64, return_sequences=True, recurrent_dropout=0.2)(temporal_input_layer)

#demographic input branch
demographic_input_layer = Input(shape=(5))
demographic_dense = layers.Dense(16)(demographic_input_layer)
demographic_dropout = layers.Dropout(0.2)(demographic_dense)

#cases output branch
rnn_c = layers.LSTM(32)(main_rnn_layer)
merge_c = layers.Concatenate(axis=-1)([rnn_c,demographic_dropout])
dense_c = layers.Dense(128)(merge_c)
dropout_c = layers.Dropout(0.3)(dense_c)
cases = layers.Dense(1, activation=layers.LeakyReLU(alpha=0.1),name="cases")(dropout_c)

#fatality output branch
rnn_f = layers.LSTM(32)(main_rnn_layer)
merge_f = layers.Concatenate(axis=-1)([rnn_f,demographic_dropout])
dense_f = layers.Dense(128)(merge_f)
dropout_f = layers.Dropout(0.3)(dense_f)
fatalities = layers.Dense(1, activation=layers.LeakyReLU(alpha=0.1), name="fatalities")(dropout_f)


model = Model([temporal_input_layer,demographic_input_layer], [cases,fatalities])

model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 20, 5)]      0
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, 5)]          0
__________________________________________________________________________________________________
lstm (LSTM)                     (None, 20, 64)       17920       input_1[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 16)           96          input_2[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 32)           12416       lstm[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 16)           0           dense[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   (None, 32)           12416       lstm[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 48)           0           lstm_1[0][0]
                                                                 dropout[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 48)           0           lstm_2[0][0]
                                                                 dropout[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 128)          6272        concatenate[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 128)          6272        concatenate_1[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 128)          0           dense_1[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 128)          0           dense_2[0][0]
__________________________________________________________________________________________________
cases (Dense)                   (None, 1)            129         dropout_1[0][0]
__________________________________________________________________________________________________
fatalities (Dense)              (None, 1)            129         dropout_2[0][0]
==================================================================================================
Total params: 55,650
Trainable params: 55,650
Non-trainable params: 0
__________________________________________________________________________________________________
```

In [18]:

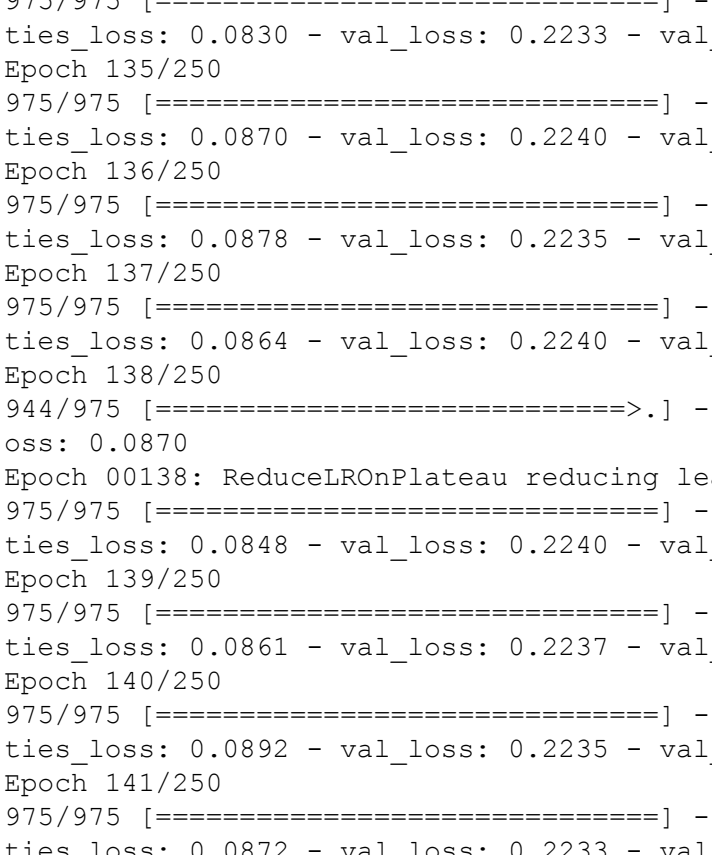
```
callbacks = [ReduceLROnPlateau(monitor='val_loss', patience=4, verbose=1, factor=0.6),
             EarlyStopping(monitor='val_loss', patience=20),
             ModelCheckpoint(filepath='best_model.h5', monitor='val_loss', save_best_only=True)]
model.compile(loss=[tf.keras.losses.MeanSquaredLogarithmicError(),tf.keras.losses.MeanSquaredLogarithmi
cError()], optimizer="adam")
```

```
In [19]: history = model.fit([X_temporal_train,X_demographic_train], [Y_cases_train, Y_fatalities_train],
                  epochs = 250,
                  batch_size = 16,
                  validation_data=([X_temporal_test,X_demographic_test],  [Y_cases_test, Y_fatalities_test]),
                  callbacks=callbacks)
```

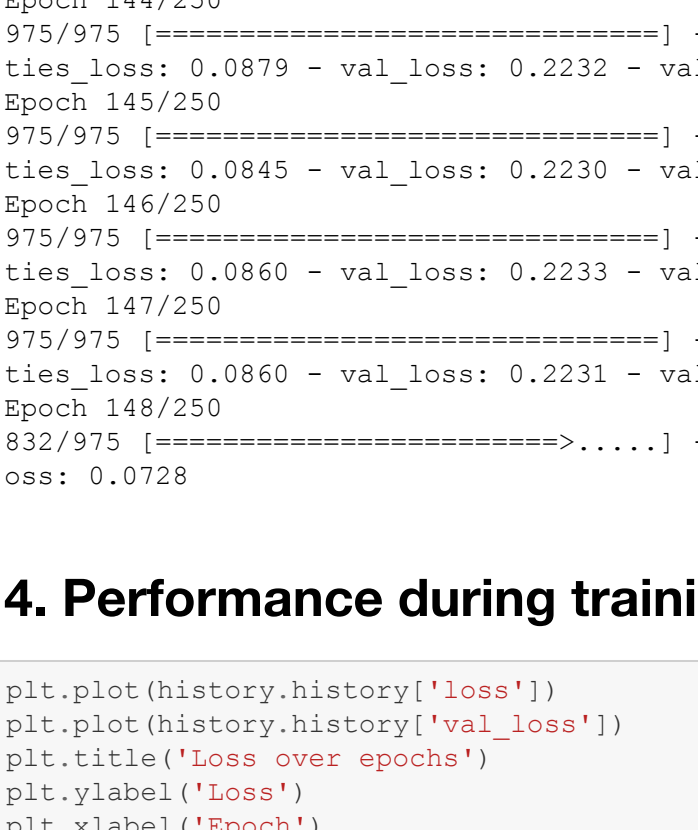
```
Train on 975 samples, validate on 109 samples
Epoch 1/250
975/975 [==============================] - 9s 9ms/sample - loss: 10.6014 - cases_loss: 9.1545 - fatal
ities_loss: 1.4393 - val_loss: 4.8438 - val_cases_loss: 3.9771 - val_fatalities_loss: 0.8372
Epoch 2/250
975/975 [==============================] - 2s 2ms/sample - loss: 5.0408 - cases_loss: 3.9627 - fatali
ties_loss: 1.0756 - val_loss: 3.1688 - val_cases_loss: 2.4836 - val_fatalities_loss: 0.6586
Epoch 3/250
975/975 [==============================] - 2s 2ms/sample - loss: 3.5641 - cases_loss: 2.8420 - fatali
ties_loss: 0.7206 - val_loss: 2.2247 - val_cases_loss: 1.8413 - val_fatalities_loss: 0.3620
Epoch 4/250
975/975 [==============================] - 2s 2ms/sample - loss: 2.8321 - cases_loss: 2.3003 - fatali
ties_loss: 0.5304 - val_loss: 1.7760 - val_cases_loss: 1.4858 - val_fatalities_loss: 0.2718
Epoch 5/250
975/975 [==============================] - 2s 2ms/sample - loss: 2.4104 - cases_loss: 1.9618 - fatali
ties_loss: 0.4468 - val_loss: 1.5880 - val_cases_loss: 1.2601 - val_fatalities_loss: 0.3095
Epoch 6/250
975/975 [==============================] - 2s 2ms/sample - loss: 2.1510 - cases_loss: 1.7179 - fatali
ties_loss: 0.4327 - val_loss: 1.4177 - val_cases_loss: 1.1267 - val_fatalities_loss: 0.2735
Epoch 7/250
975/975 [==============================] - 2s 2ms/sample - loss: 1.9387 - cases_loss: 1.5574 - fatali
ties_loss: 0.3800 - val_loss: 1.2350 - val_cases_loss: 0.9660 - val_fatalities_loss: 0.2503
Epoch 8/250
975/975 [==============================] - 2s 2ms/sample - loss: 1.8229 - cases_loss: 1.4262 - fatali
ties_loss: 0.3983 - val_loss: 1.1337 - val_cases_loss: 0.9396 - val_fatalities_loss: 0.1783
Epoch 9/250
975/975 [==============================] - 2s 2ms/sample - loss: 1.6446 - cases_loss: 1.3078 - fatali
ties_loss: 0.3357 - val_loss: 1.0216 - val_cases_loss: 0.7979 - val_fatalities_loss: 0.2074
Epoch 10/250
975/975 [==============================] - 2s 2ms/sample - loss: 1.5746 - cases_loss: 1.2295 - fatali
ties_loss: 0.3461 - val_loss: 0.9406 - val_cases_loss: 0.7237 - val_fatalities_loss: 0.2015
Epoch 11/250
975/975 [==============================] - 2s 2ms/sample - loss: 1.4594 - cases_loss: 1.1439 - fatali
ties_loss: 0.3190 - val_loss: 0.9880 - val_cases_loss: 0.6705 - val_fatalities_loss: 0.3000
Epoch 12/250
975/975 [==============================] - 2s 2ms/sample - loss: 1.5458 - cases_loss: 1.0632 - fatali
ties_loss: 0.4815 - val_loss: 0.8387 - val_cases_loss: 0.6017 - val_fatalities_loss: 0.2212
Epoch 13/250
975/975 [==============================] - 2s 2ms/sample - loss: 1.3365 - cases_loss: 0.9951 - fatali
ties_loss: 0.3426 - val_loss: 0.8547 - val_cases_loss: 0.5811 - val_fatalities_loss: 0.2616
Epoch 14/250
975/975 [==============================] - 2s 2ms/sample - loss: 1.2216 - cases_loss: 0.9317 - fatali
ties_loss: 0.2899 - val_loss: 0.6883 - val_cases_loss: 0.5099 - val_fatalities_loss: 0.1657
Epoch 15/250
975/975 [==============================] - 2s 2ms/sample - loss: 1.2157 - cases_loss: 0.9021 - fatali
ties_loss: 0.3128 - val_loss: 0.6989 - val_cases_loss: 0.4945 - val_fatalities_loss: 0.1933
Epoch 16/250
975/975 [==============================] - 2s 2ms/sample - loss: 1.1145 - cases_loss: 0.8539 - fatali
ties_loss: 0.2600 - val_loss: 0.6300 - val_cases_loss: 0.4845 - val_fatalities_loss: 0.1335
Epoch 17/250
975/975 [==============================] - 2s 2ms/sample - loss: 1.0910 - cases_loss: 0.8129 - fatali
ties_loss: 0.2777 - val_loss: 0.5894 - val_cases_loss: 0.4386 - val_fatalities_loss: 0.1390
Epoch 18/250
975/975 [==============================] - 2s 2ms/sample - loss: 1.0698 - cases_loss: 0.7762 - fatali
ties_loss: 0.2931 - val_loss: 0.6873 - val_cases_loss: 0.4557 - val_fatalities_loss: 0.2171
Epoch 19/250
975/975 [==============================] - 2s 2ms/sample - loss: 1.0295 - cases_loss: 0.7687 - fatali
ties_loss: 0.2602 - val_loss: 0.5136 - val_cases_loss: 0.3882 - val_fatalities_loss: 0.1143
Epoch 20/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.9330 - cases_loss: 0.7147 - fatali
ties_loss: 0.2176 - val_loss: 0.4985 - val_cases_loss: 0.3816 - val_fatalities_loss: 0.1067
Epoch 21/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.9563 - cases_loss: 0.6895 - fatali
ties_loss: 0.2661 - val_loss: 0.4992 - val_cases_loss: 0.3498 - val_fatalities_loss: 0.1382
Epoch 22/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.9607 - cases_loss: 0.6878 - fatali
ties_loss: 0.2723 - val_loss: 0.5109 - val_cases_loss: 0.3395 - val_fatalities_loss: 0.1617
Epoch 23/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.8750 - cases_loss: 0.6586 - fatali
ties_loss: 0.2166 - val_loss: 0.4637 - val_cases_loss: 0.3347 - val_fatalities_loss: 0.1206
Epoch 24/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.9150 - cases_loss: 0.6327 - fatali
ties_loss: 0.2818 - val_loss: 0.5910 - val_cases_loss: 0.3530 - val_fatalities_loss: 0.2277
Epoch 25/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.8870 - cases_loss: 0.6193 - fatali
ties_loss: 0.2677 - val_loss: 0.5088 - val_cases_loss: 0.3261 - val_fatalities_loss: 0.1724
Epoch 26/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.9198 - cases_loss: 0.5951 - fatali
ties_loss: 0.3239 - val_loss: 0.5893 - val_cases_loss: 0.3088 - val_fatalities_loss: 0.2694
Epoch 27/250
944/975 [============================>.] - ETA: 0s - loss: 0.8369 - cases_loss: 0.5710 - fatalities_l
oss: 0.2659
Epoch 00027: ReduceLROnPlateau reducing learning rate to 0.0006000000284984708.
975/975 [==============================] - 2s 2ms/sample - loss: 0.8728 - cases_loss: 0.5937 - fatali
ties_loss: 0.2806 - val_loss: 0.6223 - val_cases_loss: 0.4051 - val_fatalities_loss: 0.2047
Epoch 28/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.8387 - cases_loss: 0.5916 - fatali
ties_loss: 0.2473 - val_loss: 0.4468 - val_cases_loss: 0.2821 - val_fatalities_loss: 0.1563
Epoch 29/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.7581 - cases_loss: 0.5697 - fatali
ties_loss: 0.1877 - val_loss: 0.4214 - val_cases_loss: 0.2787 - val_fatalities_loss: 0.1338
Epoch 30/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.7263 - cases_loss: 0.5446 - fatali
ties_loss: 0.1814 - val_loss: 0.3915 - val_cases_loss: 0.2627 - val_fatalities_loss: 0.1210
Epoch 31/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.7109 - cases_loss: 0.5359 - fatali
ties_loss: 0.1751 - val_loss: 0.3687 - val_cases_loss: 0.2637 - val_fatalities_loss: 0.0973
Epoch 32/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.7413 - cases_loss: 0.5319 - fatali
ties_loss: 0.2102 - val_loss: 0.3616 - val_cases_loss: 0.2545 - val_fatalities_loss: 0.0993
Epoch 33/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.7274 - cases_loss: 0.5123 - fatali
ties_loss: 0.2145 - val_loss: 0.3563 - val_cases_loss: 0.2477 - val_fatalities_loss: 0.1004
Epoch 34/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.6968 - cases_loss: 0.5020 - fatali
ties_loss: 0.1943 - val_loss: 0.3671 - val_cases_loss: 0.2430 - val_fatalities_loss: 0.1168
Epoch 35/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.7149 - cases_loss: 0.5235 - fatali
ties_loss: 0.1913 - val_loss: 0.4718 - val_cases_loss: 0.3176 - val_fatalities_loss: 0.1442
Epoch 36/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.7067 - cases_loss: 0.5117 - fatali
ties_loss: 0.1947 - val_loss: 0.3705 - val_cases_loss: 0.2455 - val_fatalities_loss: 0.1168
Epoch 37/250
944/975 [============================>.] - ETA: 0s - loss: 0.6331 - cases_loss: 0.4594 - fatalities_l
oss: 0.1737
Epoch 00037: ReduceLROnPlateau reducing learning rate to 0.0003600000170990825.
975/975 [==============================] - 2s 2ms/sample - loss: 0.6703 - cases_loss: 0.4864 - fatali
ties_loss: 0.1849 - val_loss: 0.4091 - val_cases_loss: 0.2929 - val_fatalities_loss: 0.1082
Epoch 38/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.6625 - cases_loss: 0.4875 - fatali
ties_loss: 0.1748 - val_loss: 0.3350 - val_cases_loss: 0.2262 - val_fatalities_loss: 0.1015
Epoch 39/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.6204 - cases_loss: 0.4705 - fatali
ties_loss: 0.1496 - val_loss: 0.3188 - val_cases_loss: 0.2253 - val_fatalities_loss: 0.0864
Epoch 40/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.6267 - cases_loss: 0.4696 - fatali
ties_loss: 0.1564 - val_loss: 0.3301 - val_cases_loss: 0.2244 - val_fatalities_loss: 0.0984
Epoch 41/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.6094 - cases_loss: 0.4669 - fatali
ties_loss: 0.1424 - val_loss: 0.3104 - val_cases_loss: 0.2222 - val_fatalities_loss: 0.0815
Epoch 42/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.6104 - cases_loss: 0.4741 - fatali
ties_loss: 0.1390 - val_loss: 0.2991 - val_cases_loss: 0.2190 - val_fatalities_loss: 0.0734
Epoch 43/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5948 - cases_loss: 0.4557 - fatali
ties_loss: 0.1388 - val_loss: 0.2971 - val_cases_loss: 0.2175 - val_fatalities_loss: 0.0730
Epoch 44/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5813 - cases_loss: 0.4501 - fatali
ties_loss: 0.1307 - val_loss: 0.2915 - val_cases_loss: 0.2144 - val_fatalities_loss: 0.0705
Epoch 45/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5869 - cases_loss: 0.4527 - fatali
ties_loss: 0.1338 - val_loss: 0.3210 - val_cases_loss: 0.2139 - val_fatalities_loss: 0.0995
Epoch 46/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5845 - cases_loss: 0.4433 - fatali
ties_loss: 0.1407 - val_loss: 0.3019 - val_cases_loss: 0.2144 - val_fatalities_loss: 0.0810
Epoch 47/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5739 - cases_loss: 0.4391 - fatali
ties_loss: 0.1344 - val_loss: 0.2933 - val_cases_loss: 0.2100 - val_fatalities_loss: 0.0768
Epoch 48/250
944/975 [============================>.] - ETA: 0s - loss: 0.5816 - cases_loss: 0.4532 - fatalities_l
oss: 0.1283
Epoch 00048: ReduceLROnPlateau reducing learning rate to 0.00021600000327453016.
975/975 [==============================] - 2s 2ms/sample - loss: 0.5656 - cases_loss: 0.4397 - fatali
ties_loss: 0.1253 - val_loss: 0.3020 - val_cases_loss: 0.2176 - val_fatalities_loss: 0.0784
Epoch 49/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5557 - cases_loss: 0.4340 - fatali
ties_loss: 0.1213 - val_loss: 0.2791 - val_cases_loss: 0.2071 - val_fatalities_loss: 0.0658
Epoch 50/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5477 - cases_loss: 0.4311 - fatali
ties_loss: 0.1181 - val_loss: 0.2760 - val_cases_loss: 0.2028 - val_fatalities_loss: 0.0671
Epoch 51/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5465 - cases_loss: 0.4303 - fatali
ties_loss: 0.1157 - val_loss: 0.2765 - val_cases_loss: 0.2036 - val_fatalities_loss: 0.0669
Epoch 52/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5420 - cases_loss: 0.4221 - fatali
ties_loss: 0.1197 - val_loss: 0.2813 - val_cases_loss: 0.2017 - val_fatalities_loss: 0.0738
Epoch 53/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5358 - cases_loss: 0.4200 - fatali
ties_loss: 0.1161 - val_loss: 0.2909 - val_cases_loss: 0.2196 - val_fatalities_loss: 0.0647
Epoch 54/250
944/975 [============================>.] - ETA: 0s - loss: 0.5305 - cases_loss: 0.4163 - fatalities_l
oss: 0.1142
Epoch 00054: ReduceLROnPlateau reducing learning rate to 0.0001296000002184827.
975/975 [==============================] - 2s 2ms/sample - loss: 0.5331 - cases_loss: 0.4178 - fatali
ties_loss: 0.1161 - val_loss: 0.2868 - val_cases_loss: 0.1990 - val_fatalities_loss: 0.0813
Epoch 55/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5262 - cases_loss: 0.4131 - fatali
ties_loss: 0.1127 - val_loss: 0.2710 - val_cases_loss: 0.2028 - val_fatalities_loss: 0.0623
Epoch 56/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5264 - cases_loss: 0.4176 - fatali
ties_loss: 0.1110 - val_loss: 0.2659 - val_cases_loss: 0.1973 - val_fatalities_loss: 0.0627
Epoch 57/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5248 - cases_loss: 0.4156 - fatali
ties_loss: 0.1089 - val_loss: 0.2732 - val_cases_loss: 0.2000 - val_fatalities_loss: 0.0676
Epoch 58/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5119 - cases_loss: 0.4042 - fatali
ties_loss: 0.1074 - val_loss: 0.2785 - val_cases_loss: 0.2095 - val_fatalities_loss: 0.0630
Epoch 59/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5195 - cases_loss: 0.4060 - fatali
ties_loss: 0.1132 - val_loss: 0.2604 - val_cases_loss: 0.1952 - val_fatalities_loss: 0.0594
Epoch 60/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5154 - cases_loss: 0.4042 - fatali
ties_loss: 0.1108 - val_loss: 0.2585 - val_cases_loss: 0.1927 - val_fatalities_loss: 0.0599
Epoch 61/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5215 - cases_loss: 0.4062 - fatali
ties_loss: 0.1155 - val_loss: 0.2630 - val_cases_loss: 0.1922 - val_fatalities_loss: 0.0652
Epoch 62/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5106 - cases_loss: 0.4040 - fatali
ties_loss: 0.1078 - val_loss: 0.2614 - val_cases_loss: 0.1913 - val_fatalities_loss: 0.0644
Epoch 63/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5116 - cases_loss: 0.4031 - fatali
ties_loss: 0.1079 - val_loss: 0.2588 - val_cases_loss: 0.1916 - val_fatalities_loss: 0.0615
Epoch 64/250
960/975 [============================>.] - ETA: 0s - loss: 0.5139 - cases_loss: 0.4029 - fatalities_l
oss: 0.1110
Epoch 00064: ReduceLROnPlateau reducing learning rate to 7.775999838486313e-05.
975/975 [==============================] - 2s 2ms/sample - loss: 0.5071 - cases_loss: 0.3973 - fatali
ties_loss: 0.1093 - val_loss: 0.2592 - val_cases_loss: 0.1901 - val_fatalities_loss: 0.0634
Epoch 65/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5086 - cases_loss: 0.4003 - fatali
ties_loss: 0.1078 - val_loss: 0.2540 - val_cases_loss: 0.1885 - val_fatalities_loss: 0.0600
Epoch 66/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.5021 - cases_loss: 0.3991 - fatali
ties_loss: 0.1031 - val_loss: 0.2516 - val_cases_loss: 0.1869 - val_fatalities_loss: 0.0591
Epoch 67/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4953 - cases_loss: 0.3925 - fatali
ties_loss: 0.1039 - val_loss: 0.2545 - val_cases_loss: 0.1895 - val_fatalities_loss: 0.0595
Epoch 68/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4990 - cases_loss: 0.3937 - fatali
ties_loss: 0.1048 - val_loss: 0.2521 - val_cases_loss: 0.1873 - val_fatalities_loss: 0.0594
Epoch 69/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4933 - cases_loss: 0.3950 - fatali
ties_loss: 0.1026 - val_loss: 0.2490 - val_cases_loss: 0.1853 - val_fatalities_loss: 0.0580
Epoch 70/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4899 - cases_loss: 0.3912 - fatali
ties_loss: 0.0982 - val_loss: 0.2496 - val_cases_loss: 0.1849 - val_fatalities_loss: 0.0592
Epoch 71/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4926 - cases_loss: 0.3901 - fatali
ties_loss: 0.1032 - val_loss: 0.2476 - val_cases_loss: 0.1844 - val_fatalities_loss: 0.0577
Epoch 72/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4886 - cases_loss: 0.3889 - fatali
ties_loss: 0.0999 - val_loss: 0.2629 - val_cases_loss: 0.1987 - val_fatalities_loss: 0.0586
Epoch 73/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4882 - cases_loss: 0.3896 - fatali
ties_loss: 0.0981 - val_loss: 0.2530 - val_cases_loss: 0.1893 - val_fatalities_loss: 0.0583
Epoch 74/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4883 - cases_loss: 0.3898 - fatali
ties_loss: 0.0981 - val_loss: 0.2513 - val_cases_loss: 0.1876 - val_fatalities_loss: 0.0583
Epoch 75/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4838 - cases_loss: 0.3863 - fatali
ties_loss: 0.0990 - val_loss: 0.2448 - val_cases_loss: 0.1823 - val_fatalities_loss: 0.0570
Epoch 76/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4852 - cases_loss: 0.3865 - fatali
ties_loss: 0.0987 - val_loss: 0.2481 - val_cases_loss: 0.1851 - val_fatalities_loss: 0.0574
Epoch 77/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4883 - cases_loss: 0.3919 - fatali
ties_loss: 0.0976 - val_loss: 0.2485 - val_cases_loss: 0.1850 - val_fatalities_loss: 0.0581
Epoch 78/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4867 - cases_loss: 0.3887 - fatali
ties_loss: 0.0975 - val_loss: 0.2421 - val_cases_loss: 0.1796 - val_fatalities_loss: 0.0572
Epoch 79/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4772 - cases_loss: 0.3774 - fatali
ties_loss: 0.0994 - val_loss: 0.2401 - val_cases_loss: 0.1790 - val_fatalities_loss: 0.0557
Epoch 80/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4785 - cases_loss: 0.3784 - fatali
ties_loss: 0.1000 - val_loss: 0.2460 - val_cases_loss: 0.1822 - val_fatalities_loss: 0.0584
Epoch 81/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4782 - cases_loss: 0.3808 - fatali
ties_loss: 0.1002 - val_loss: 0.2466 - val_cases_loss: 0.1815 - val_fatalities_loss: 0.0597
Epoch 82/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4848 - cases_loss: 0.3867 - fatali
ties_loss: 0.0978 - val_loss: 0.2396 - val_cases_loss: 0.1771 - val_fatalities_loss: 0.0571
Epoch 83/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4811 - cases_loss: 0.3828 - fatali
ties_loss: 0.0979 - val_loss: 0.2440 - val_cases_loss: 0.1788 - val_fatalities_loss: 0.0598
Epoch 84/250
975/975 [==============================] - 3s 3ms/sample - loss: 0.4740 - cases_loss: 0.3784 - fatali
ties_loss: 0.0957 - val_loss: 0.2379 - val_cases_loss: 0.1758 - val_fatalities_loss: 0.0568
Epoch 85/250
975/975 [==============================] - 3s 3ms/sample - loss: 0.4714 - cases_loss: 0.3750 - fatali
ties_loss: 0.0962 - val_loss: 0.2415 - val_cases_loss: 0.1805 - val_fatalities_loss: 0.0558
Epoch 86/250
975/975 [==============================] - 3s 3ms/sample - loss: 0.4727 - cases_loss: 0.3755 - fatali
ties_loss: 0.0968 - val_loss: 0.2478 - val_cases_loss: 0.1755 - val_fatalities_loss: 0.0671
Epoch 87/250
975/975 [==============================] - 3s 3ms/sample - loss: 0.4737 - cases_loss: 0.3799 - fatali
ties_loss: 0.0934 - val_loss: 0.2385 - val_cases_loss: 0.1742 - val_fatalities_loss: 0.0591
Epoch 88/250
944/975 [============================>.] - ETA: 0s - loss: 0.4779 - cases_loss: 0.3815 - fatalities_l
oss: 0.0964
Epoch 00088: ReduceLROnPlateau reducing learning rate to 4.6655999904032795e-05.
975/975 [==============================] - 3s 3ms/sample - loss: 0.4677 - cases_loss: 0.3736 - fatali
ties_loss: 0.0936 - val_loss: 0.2391 - val_cases_loss: 0.1781 - val_fatalities_loss: 0.0557
Epoch 89/250
975/975 [==============================] - 3s 3ms/sample - loss: 0.4647 - cases_loss: 0.3702 - fatali
ties_loss: 0.0953 - val_loss: 0.2340 - val_cases_loss: 0.1728 - val_fatalities_loss: 0.0561
Epoch 90/250
975/975 [==============================] - 3s 3ms/sample - loss: 0.4645 - cases_loss: 0.3703 - fatali
ties_loss: 0.0938 - val_loss: 0.2362 - val_cases_loss: 0.1751 - val_fatalities_loss: 0.0559
Epoch 91/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4649 - cases_loss: 0.3710 - fatali
ties_loss: 0.0936 - val_loss: 0.2335 - val_cases_loss: 0.1728 - val_fatalities_loss: 0.0556
Epoch 92/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4640 - cases_loss: 0.3721 - fatali
ties_loss: 0.0918 - val_loss: 0.2344 - val_cases_loss: 0.1730 - val_fatalities_loss: 0.0561
Epoch 93/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4599 - cases_loss: 0.3661 - fatali
ties_loss: 0.0935 - val_loss: 0.2375 - val_cases_loss: 0.1740 - val_fatalities_loss: 0.0583
Epoch 94/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4577 - cases_loss: 0.3666 - fatali
ties_loss: 0.0907 - val_loss: 0.2336 - val_cases_loss: 0.1718 - val_fatalities_loss: 0.0566
Epoch 95/250
944/975 [============================>.] - ETA: 0s - loss: 0.4642 - cases_loss: 0.3735 - fatalities_l
oss: 0.0908
Epoch 00095: ReduceLROnPlateau reducing learning rate to 2.799360081553459e-05.
975/975 [==============================] - 2s 2ms/sample - loss: 0.4554 - cases_loss: 0.3657 - fatali
ties_loss: 0.0894 - val_loss: 0.2375 - val_cases_loss: 0.1739 - val_fatalities_loss: 0.0583
Epoch 96/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4605 - cases_loss: 0.3676 - fatali
ties_loss: 0.0925 - val_loss: 0.2333 - val_cases_loss: 0.1721 - val_fatalities_loss: 0.0560
Epoch 97/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4637 - cases_loss: 0.3706 - fatali
ties_loss: 0.0927 - val_loss: 0.2310 - val_cases_loss: 0.1703 - val_fatalities_loss: 0.0556
Epoch 98/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4594 - cases_loss: 0.3674 - fatali
ties_loss: 0.0916 - val_loss: 0.2343 - val_cases_loss: 0.1738 - val_fatalities_loss: 0.0553
Epoch 99/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4515 - cases_loss: 0.3601 - fatali
ties_loss: 0.0910 - val_loss: 0.2309 - val_cases_loss: 0.1696 - val_fatalities_loss: 0.0562
Epoch 100/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4595 - cases_loss: 0.3674 - fatali
ties_loss: 0.0917 - val_loss: 0.2307 - val_cases_loss: 0.1699 - val_fatalities_loss: 0.0557
Epoch 101/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4553 - cases_loss: 0.3635 - fatali
ties_loss: 0.0925 - val_loss: 0.2302 - val_cases_loss: 0.1693 - val_fatalities_loss: 0.0557
Epoch 102/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4494 - cases_loss: 0.3599 - fatali
ties_loss: 0.0892 - val_loss: 0.2285 - val_cases_loss: 0.1684 - val_fatalities_loss: 0.0551
Epoch 103/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4539 - cases_loss: 0.3629 - fatali
ties_loss: 0.0906 - val_loss: 0.2291 - val_cases_loss: 0.1688 - val_fatalities_loss: 0.0552
Epoch 104/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4479 - cases_loss: 0.3580 - fatali
ties_loss: 0.0895 - val_loss: 0.2295 - val_cases_loss: 0.1689 - val_fatalities_loss: 0.0555
Epoch 105/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4544 - cases_loss: 0.3624 - fatali
ties_loss: 0.0917 - val_loss: 0.2296 - val_cases_loss: 0.1683 - val_fatalities_loss: 0.0563
Epoch 106/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4539 - cases_loss: 0.3623 - fatali
ties_loss: 0.0912 - val_loss: 0.2276 - val_cases_loss: 0.1676 - val_fatalities_loss: 0.0549
Epoch 107/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4512 - cases_loss: 0.3613 - fatali
ties_loss: 0.0895 - val_loss: 0.2270 - val_cases_loss: 0.1675 - val_fatalities_loss: 0.0545
Epoch 108/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4522 - cases_loss: 0.3625 - fatali
ties_loss: 0.0895 - val_loss: 0.2291 - val_cases_loss: 0.1686 - val_fatalities_loss: 0.0555
Epoch 109/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4478 - cases_loss: 0.3585 - fatali
ties_loss: 0.0888 - val_loss: 0.2282 - val_cases_loss: 0.1673 - val_fatalities_loss: 0.0559
Epoch 110/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4501 - cases_loss: 0.3610 - fatali
ties_loss: 0.0889 - val_loss: 0.2302 - val_cases_loss: 0.1697 - val_fatalities_loss: 0.0554
Epoch 111/250
944/975 [============================>.] - ETA: 0s - loss: 0.4542 - cases_loss: 0.3632 - fatalities_l
oss: 0.0910
Epoch 00111: ReduceLROnPlateau reducing learning rate to 1.6796160707599483e-05.
975/975 [==============================] - 2s 2ms/sample - loss: 0.4459 - cases_loss: 0.3570 - fatali
ties_loss: 0.0887 - val_loss: 0.2314 - val_cases_loss: 0.1702 - val_fatalities_loss: 0.0561
Epoch 112/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4410 - cases_loss: 0.3567 - fatali
ties_loss: 0.0849 - val_loss: 0.2269 - val_cases_loss: 0.1675 - val_fatalities_loss: 0.0544
Epoch 113/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4481 - cases_loss: 0.3572 - fatali
ties_loss: 0.0905 - val_loss: 0.2272 - val_cases_loss: 0.1663 - val_fatalities_loss: 0.0559
Epoch 114/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4456 - cases_loss: 0.3580 - fatali
ties_loss: 0.0873 - val_loss: 0.2260 - val_cases_loss: 0.1661 - val_fatalities_loss: 0.0549
Epoch 115/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4463 - cases_loss: 0.3586 - fatali
ties_loss: 0.0873 - val_loss: 0.2271 - val_cases_loss: 0.1665 - val_fatalities_loss: 0.0556
Epoch 116/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4456 - cases_loss: 0.3587 - fatali
ties_loss: 0.0865 - val_loss: 0.2285 - val_cases_loss: 0.1684 - val_fatalities_loss: 0.0550
Epoch 117/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4484 - cases_loss: 0.3582 - fatali
ties_loss: 0.0911 - val_loss: 0.2262 - val_cases_loss: 0.1660 - val_fatalities_loss: 0.0552
Epoch 118/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4448 - cases_loss: 0.3559 - fatali
ties_loss: 0.0885 - val_loss: 0.2252 - val_cases_loss: 0.1653 - val_fatalities_loss: 0.0549
Epoch 119/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4435 - cases_loss: 0.3538 - fatali
ties_loss: 0.0894 - val_loss: 0.2253 - val_cases_loss: 0.1656 - val_fatalities_loss: 0.0547
Epoch 120/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4433 - cases_loss: 0.3550 - fatali
ties_loss: 0.0888 - val_loss: 0.2247 - val_cases_loss: 0.1653 - val_fatalities_loss: 0.0545
Epoch 121/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4410 - cases_loss: 0.3545 - fatali
ties_loss: 0.0862 - val_loss: 0.2252 - val_cases_loss: 0.1653 - val_fatalities_loss: 0.0550
Epoch 122/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4412 - cases_loss: 0.3546 - fatali
ties_loss: 0.0862 - val_loss: 0.2258 - val_cases_loss: 0.1649 - val_fatalities_loss: 0.0559
Epoch 123/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4451 - cases_loss: 0.3590 - fatali
ties_loss: 0.0874 - val_loss: 0.2255 - val_cases_loss: 0.1654 - val_fatalities_loss: 0.0551
Epoch 124/250
944/975 [============================>.] - ETA: 0s - loss: 0.4214 - cases_loss: 0.3390 - fatalities_l
oss: 0.0824
Epoch 00124: ReduceLROnPlateau reducing learning rate to 1.007769642455969e-05.
975/975 [==============================] - 2s 2ms/sample - loss: 0.4395 - cases_loss: 0.3547 - fatali
ties_loss: 0.0863 - val_loss: 0.2272 - val_cases_loss: 0.1666 - val_fatalities_loss: 0.0556
Epoch 125/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4452 - cases_loss: 0.3581 - fatali
ties_loss: 0.0878 - val_loss: 0.2244 - val_cases_loss: 0.1646 - val_fatalities_loss: 0.0548
Epoch 126/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4391 - cases_loss: 0.3516 - fatali
ties_loss: 0.0871 - val_loss: 0.2243 - val_cases_loss: 0.1647 - val_fatalities_loss: 0.0547
Epoch 127/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4461 - cases_loss: 0.3569 - fatali
ties_loss: 0.0891 - val_loss: 0.2252 - val_cases_loss: 0.1654 - val_fatalities_loss: 0.0548
Epoch 128/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4440 - cases_loss: 0.3568 - fatali
ties_loss: 0.0868 - val_loss: 0.2239 - val_cases_loss: 0.1642 - val_fatalities_loss: 0.0548
Epoch 129/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4432 - cases_loss: 0.3567 - fatali
ties_loss: 0.0865 - val_loss: 0.2252 - val_cases_loss: 0.1647 - val_fatalities_loss: 0.0556
Epoch 130/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4429 - cases_loss: 0.3536 - fatali
ties_loss: 0.0890 - val_loss: 0.2248 - val_cases_loss: 0.1643 - val_fatalities_loss: 0.0555
Epoch 131/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4417 - cases_loss: 0.3537 - fatali
ties_loss: 0.0879 - val_loss: 0.2242 - val_cases_loss: 0.1646 - val_fatalities_loss: 0.0547
Epoch 132/250
944/975 [============================>.] - ETA: 0s - loss: 0.4531 - cases_loss: 0.3655 - fatalities_l
oss: 0.0876
Epoch 00132: ReduceLROnPlateau reducing learning rate to 6.046617636457085e-06.
975/975 [==============================] - 2s 2ms/sample - loss: 0.4465 - cases_loss: 0.3598 - fatali
ties_loss: 0.0863 - val_loss: 0.2238 - val_cases_loss: 0.1643 - val_fatalities_loss: 0.0546
Epoch 133/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4370 - cases_loss: 0.3545 - fatali
ties_loss: 0.0837 - val_loss: 0.2236 - val_cases_loss: 0.1641 - val_fatalities_loss: 0.0546
Epoch 134/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4350 - cases_loss: 0.3516 - fatali
ties_loss: 0.0830 - val_loss: 0.2233 - val_cases_loss: 0.1637 - val_fatalities_loss: 0.0546
Epoch 135/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4391 - cases_loss: 0.3518 - fatali
ties_loss: 0.0870 - val_loss: 0.2240 - val_cases_loss: 0.1642 - val_fatalities_loss: 0.0548
Epoch 136/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4396 - cases_loss: 0.3515 - fatali
ties_loss: 0.0878 - val_loss: 0.2235 - val_cases_loss: 0.1640 - val_fatalities_loss: 0.0546
Epoch 137/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4339 - cases_loss: 0.3475 - fatali
ties_loss: 0.0864 - val_loss: 0.2240 - val_cases_loss: 0.1640 - val_fatalities_loss: 0.0551
Epoch 138/250
944/975 [============================>.] - ETA: 0s - loss: 0.4512 - cases_loss: 0.3642 - fatalities_l
oss: 0.0870
Epoch 00138: ReduceLROnPlateau reducing learning rate to 3.6279706364439334e-06.
975/975 [==============================] - 2s 2ms/sample - loss: 0.4384 - cases_loss: 0.3532 - fatali
ties_loss: 0.0848 - val_loss: 0.2240 - val_cases_loss: 0.1643 - val_fatalities_loss: 0.0548
Epoch 139/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4365 - cases_loss: 0.3517 - fatali
ties_loss: 0.0861 - val_loss: 0.2237 - val_cases_loss: 0.1637 - val_fatalities_loss: 0.0550
Epoch 140/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4415 - cases_loss: 0.3519 - fatali
ties_loss: 0.0892 - val_loss: 0.2235 - val_cases_loss: 0.1637 - val_fatalities_loss: 0.0548
Epoch 141/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4339 - cases_loss: 0.3465 - fatali
ties_loss: 0.0872 - val_loss: 0.2233 - val_cases_loss: 0.1636 - val_fatalities_loss: 0.0547
Epoch 142/250
944/975 [============================>.] - ETA: 0s - loss: 0.4369 - cases_loss: 0.3535 - fatalities_l
oss: 0.0834
Epoch 00142: ReduceLROnPlateau reducing learning rate to 2.1767824364360423e-06.
975/975 [==============================] - 2s 2ms/sample - loss: 0.4442 - cases_loss: 0.3571 - fatali
ties_loss: 0.0867 - val_loss: 0.2242 - val_cases_loss: 0.1646 - val_fatalities_loss: 0.0547
Epoch 143/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4359 - cases_loss: 0.3522 - fatali
ties_loss: 0.0837 - val_loss: 0.2237 - val_cases_loss: 0.1641 - val_fatalities_loss: 0.0546
Epoch 144/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4441 - cases_loss: 0.3578 - fatali
ties_loss: 0.0879 - val_loss: 0.2232 - val_cases_loss: 0.1637 - val_fatalities_loss: 0.0546
Epoch 145/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4360 - cases_loss: 0.3520 - fatali
ties_loss: 0.0845 - val_loss: 0.2230 - val_cases_loss: 0.1634 - val_fatalities_loss: 0.0547
Epoch 146/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4407 - cases_loss: 0.3543 - fatali
ties_loss: 0.0860 - val_loss: 0.2233 - val_cases_loss: 0.1636 - val_fatalities_loss: 0.0547
Epoch 147/250
975/975 [==============================] - 2s 2ms/sample - loss: 0.4364 - cases_loss: 0.3505 - fatali
ties_loss: 0.0860 - val_loss: 0.2231 - val_cases_loss: 0.1635 - val_fatalities_loss: 0.0547
Epoch 148/250
832/975 [========================>.....] - ETA: 0s - loss: 0.4008 - cases_loss: 0.3280 - fatalities_l
oss: 0.0728
```

4. Performance during training

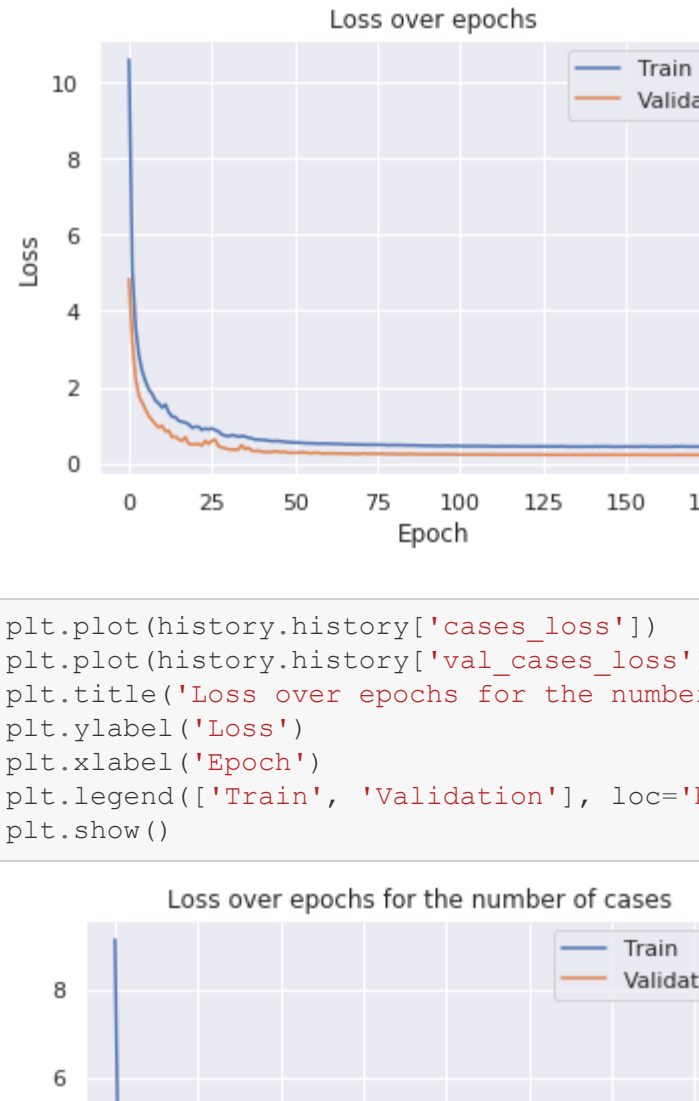
```
In [20]: plt.plot(history.history['loss'])
         plt.plot(history.history['val_loss'])
         plt.title('Loss over epochs')
         plt.ylabel('Loss')
         plt.xlabel('Epoch')
         plt.legend(['Train', 'Validation'], loc='best')
         plt.show()
```




```
In [21]: plt.plot(history.history['cases_loss'])
         plt.plot(history.history['val_cases_loss'])
         plt.title('Loss over epochs for the number of cases')
         plt.ylabel('Loss')
         plt.xlabel('Epoch')
         plt.legend(['Train', 'Validation'], loc='best')
         plt.show()
```




```
In [22]: plt.plot(history.history['fatalities_loss'])
         plt.plot(history.history['val_fatalities_loss'])
         plt.title('Loss over epochs for the number of fatalities')
         plt.ylabel('Loss')
         plt.xlabel('Epoch')
         plt.legend(['Train', 'Validation'], loc='best')
         plt.show()
```



5. Generate predictions using the model

We can quickly check the quality of the predictions... One thing is clear, there is room for improvement!

```
In [23]: model.load_weights("best_model.h5")
```

```
In [24]: predictions = model.predict([X_temporal_test,X_demographic_test])
```

I only display what I call "temporal" inputs as we're simply trying to have a feeling of how well our model is fitting the trends.

In [25]:
```python
display_limit = 30
for inputs, pred_cases, exp_cases, pred_fatalities, exp_fatalities in zip(X_temporal_test,predictions[0][:display_limit], Y_cases_test[:display_limit], predictions[1][:display_limit], Y_fatalities_test[:display_limit]):
    print("================================================")
    print(inputs)
    print("Expected cases:", exp_cases, " Prediction:", pred_cases[0], "Expected fatalities:", exp_fatalities, " Prediction:", pred_fatalities[0] )
```

```
================================================
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [2. 0. 0. 0. 0.]
 [2. 0. 0. 0. 0.]
 [2. 0. 0. 0. 0.]
 [2. 0. 0. 0. 0.]
 [3. 0. 0. 0. 0.]]
Expected cases: 4.0  Prediction: 3.9486732 Expected fatalities: 0.0  Prediction: -0.12751679
================================================
[[ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 1.  0.  1.  0.  1.]
 [ 5.  0.  1.  0.  1.]
 [ 5.  0.  1.  0.  1.]
 [ 6.  0.  1.  0.  1.]
 [ 6.  0.  1.  0.  1.]
 [ 6.  0.  1.  0.  1.]
 [ 8.  0.  1.  0.  1.]
 [ 9.  0.  1.  0.  1.]
 [11.  0.  1.  0.  1.]
 [11.  0.  1.  0.  1.]
 [13.  0.  1.  0.  1.]
 [18.  1.  1.  0.  1.]
 [22.  1.  1.  0.  1.]
 [22.  1.  1.  0.  1.]
 [27.  2.  1.  0.  1.]]
Expected cases: 37.0  Prediction: 36.152664 Expected fatalities: 3.0  Prediction: 3.023842
================================================
[[26.  0.  0.  0.  0.]
 [29.  0.  0.  0.  0.]
 [40.  0.  0.  0.  0.]
 [51.  0.  0.  0.  0.]
 [55.  0.  0.  0.  0.]
 [57.  0.  0.  0.  0.]
 [62.  0.  0.  0.  0.]
 [62.  0.  0.  0.  0.]
 [67.  0.  0.  0.  0.]
 [79.  1.  0.  0.  0.]
 [83.  2.  0.  0.  0.]
 [83.  2.  0.  0.  0.]
 [86.  2.  0.  0.  0.]
 [87.  2.  0.  0.  0.]
 [90.  2.  0.  0.  0.]
 [90.  2.  0.  0.  0.]
 [90.  2.  0.  0.  0.]
 [90.  2.  0.  0.  0.]
 [91.  2.  0.  0.  0.]
 [91.  2.  0.  0.  0.]]
Expected cases: 91.0  Prediction: 94.00686 Expected fatalities: 2.0  Prediction: 2.3598971
================================================
[[ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 3.  0.  0.  0.  0.]
 [ 4.  0.  0.  0.  0.]
 [ 4.  0.  0.  0.  0.]
 [ 7.  0.  0.  0.  0.]
 [ 7.  0.  0.  0.  0.]
 [ 7.  0.  0.  0.  0.]
 [ 9.  0.  0.  0.  0.]
 [31.  0.  0.  0.  0.]
 [45.  0.  0.  0.  0.]
 [46.  0.  0.  0.  0.]
 [73.  0.  0.  0.  0.]
 [73.  0.  0.  0.  0.]
 [89.  0.  0.  0.  0.]]
Expected cases: 99.0  Prediction: 105.23001 Expected fatalities: 1.0  Prediction: -0.017908234
================================================
[[ 3.  1.  0.  0.  0.]
 [ 3.  1.  0.  0.  0.]
 [ 3.  1.  0.  0.  0.]
 [ 3.  1.  0.  0.  0.]
 [ 3.  1.  0.  0.  0.]
 [ 3.  1.  0.  0.  0.]
 [ 3.  1.  0.  0.  0.]
 [ 3.  1.  0.  0.  0.]
 [ 3.  1.  0.  0.  0.]
 [ 3.  1.  0.  0.  0.]
 [ 3.  1.  0.  0.  0.]
 [ 3.  1.  0.  0.  0.]
 [ 3.  1.  0.  0.  0.]
 [ 3.  1.  0.  0.  0.]
 [ 3.  1.  0.  0.  0.]
 [ 5.  1.  0.  0.  0.]
 [ 6.  1.  0.  0.  0.]
 [10.  1.  0.  0.  0.]
 [20.  1.  0.  0.  0.]
 [33.  1.  0.  0.  0.]]
Expected cases: 49.0  Prediction: 42.438465 Expected fatalities: 1.0  Prediction: 0.7619477
================================================
[[ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 2.  0.  0.  0.  0.]
 [ 2.  0.  0.  0.  0.]
 [ 3.  0.  0.  0.  0.]
 [ 3.  0.  0.  0.  0.]
 [ 3.  0.  0.  0.  0.]
 [ 5.  0.  0.  0.  0.]
 [ 7.  0.  0.  0.  0.]
 [11.  0.  0.  0.  0.]
 [13.  0.  0.  0.  0.]
 [18.  0.  0.  0.  0.]
 [24.  0.  0.  0.  0.]
 [25.  0.  0.  0.  0.]
 [26.  0.  0.  0.  0.]]
Expected cases: 38.0  Prediction: 32.168762 Expected fatalities: 0.0  Prediction: -0.072939895
================================================
[[ 21.   1.   0.   0.   0.]
 [ 21.   1.   0.   0.   0.]
 [ 23.   1.   0.   0.   0.]
 [ 36.   1.   0.   0.   0.]
 [ 36.   1.   0.   0.   0.]
 [ 51.   2.   0.   0.   0.]
 [ 62.   2.   0.   0.   0.]
 [ 69.   3.   0.   0.   0.]
 [ 80.   5.   0.   0.   0.]
 [ 80.   5.   0.   0.   0.]
 [101.   5.   0.   0.   0.]
 [109.   7.   0.   0.   0.]
 [109.   7.   0.   0.   0.]
 [119.  11.   0.   0.   0.]
 [119.  11.   0.   0.   0.]
 [144.  14.   0.   0.   0.]
 [144.  20.   0.   0.   0.]
 [175.  20.   0.   0.   0.]
 [187.  20.   0.   0.   0.]
 [187.  21.   0.   0.   0.]]
Expected cases: 208.0  Prediction: 217.8423 Expected fatalities: 21.0  Prediction: 22.681429
================================================
[[ 16.   0.   0.   0.   0.]
 [ 22.   0.   0.   0.   0.]
 [ 22.   0.   0.   0.   0.]
 [ 32.   0.   0.   0.   0.]
 [ 32.   0.   0.   0.   0.]
 [ 41.   1.   0.   0.   0.]
 [ 61.   3.   0.   0.   0.]
 [ 61.   3.   0.   0.   0.]
 [ 77.   3.   0.   0.   0.]
 [ 93.   3.   0.   0.   0.]
 [110.   3.   0.   0.   0.]
 [110.   3.   0.   0.   0.]
 [120.   3.   0.   0.   0.]
 [133.   3.   0.   0.   0.]
 [157.   4.   0.   0.   0.]
 [163.   4.   0.   0.   0.]
 [187.   4.   0.   0.   0.]
 [248.   4.   0.   0.   0.]
 [267.   4.   0.   0.   0.]
 [318.   4.   0.   0.   0.]]
Expected cases: 333.0  Prediction: 350.18784 Expected fatalities: 6.0  Prediction: 4.7573433
================================================
[[3.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [3.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [9.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [1.400e+01 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [1.800e+01 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [2.100e+01 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [2.900e+01 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [4.100e+01 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [5.500e+01 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [7.900e+01 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [1.040e+02 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [1.310e+02 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [1.820e+02 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [2.460e+02 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [3.020e+02 1.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [5.040e+02 1.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [6.550e+02 1.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [8.600e+02 1.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [1.018e+03 3.000e+00 0.000e+00 1.000e+00 0.000e+00]
 [1.332e+03 3.000e+00 0.000e+00 1.000e+00 0.000e+00]]
Expected cases: 1646.0  Prediction: 961.1802 Expected fatalities: 4.0  Prediction: 5.5971193
================================================
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]]
Expected cases: 1.0  Prediction: 1.1918738 Expected fatalities: 0.0  Prediction: -0.22034846
================================================
[[ 46.   1.   0.   0.   0.]
 [ 52.   1.   0.   0.   0.]
 [ 62.   1.   0.   0.   0.]
 [ 64.   1.   0.   0.   0.]
 [ 72.   1.   0.   0.   0.]
 [ 80.   1.   0.   0.   0.]
 [ 99.   1.   0.   0.   0.]
 [106.   1.   0.   0.   0.]
 [117.   2.   0.   0.   0.]
 [124.   2.   0.   0.   0.]
 [131.   3.   0.   0.   0.]
 [138.   3.   0.   0.   0.]
 [144.   3.   0.   0.   0.]
 [157.   4.   0.   0.   0.]
 [157.   4.   0.   0.   0.]
 [159.   4.   0.   0.   0.]
 [162.   4.   0.   0.   0.]
 [162.   4.   0.   0.   0.]
 [163.   4.   0.   0.   0.]
 [163.   4.   0.   0.   0.]]
Expected cases: 168.0  Prediction: 166.07947 Expected fatalities: 4.0  Prediction: 4.2402763
================================================
[[ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 8.  1.  0.  0.  0.]
 [ 8.  1.  0.  0.  0.]
 [ 8.  1.  0.  0.  0.]
 [ 9.  1.  0.  0.  0.]
 [ 9.  1.  0.  0.  0.]
 [10.  1.  0.  0.  0.]
 [11.  1.  0.  0.  0.]
 [11.  1.  0.  0.  0.]
 [11.  1.  0.  0.  0.]
 [14.  1.  0.  0.  0.]
 [14.  1.  0.  0.  0.]
 [21.  1.  0.  0.  0.]
 [28.  1.  0.  0.  0.]
 [30.  1.  0.  0.  0.]]
Expected cases: 41.0  Prediction: 37.24991 Expected fatalities: 1.0  Prediction: 1.166281
================================================
[[347.   0.   0.   0.   0.]
 [386.   0.   0.   0.   0.]
 [416.   0.   0.   0.   0.]
 [444.   1.   0.   0.   0.]
 [466.   1.   0.   0.   0.]
 [487.   1.   0.   0.   0.]
 [497.   2.   0.   0.   0.]
 [509.   2.   0.   0.   0.]
 [523.   2.   0.   0.   0.]
 [532.   2.   0.   0.   0.]
 [537.   2.   0.   0.   0.]
 [541.   2.   0.   0.   0.]
 [543.   3.   0.   0.   0.]
 [544.   3.   0.   0.   0.]
 [546.   4.   0.   0.   0.]
 [749.   4.   0.   0.   0.]
 [750.   4.   0.   0.   0.]
 [754.   4.   0.   0.   0.]
 [755.   5.   0.   0.   0.]
 [756.   6.   0.   0.   0.]]
Expected cases: 756.0  Prediction: 745.4007 Expected fatalities: 6.0  Prediction: 5.8771095
================================================
[[   0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.]
 [ 267.   23.    0.    0.    0.]
 [ 366.   29.    0.    0.    0.]
 [ 442.   31.    0.    0.    0.]
 [ 568.   37.    0.    0.    0.]
 [ 572.   37.    0.    0.    0.]
 [ 643.   40.    0.    0.    0.]
 [ 904.   48.    0.    0.    0.]
 [1076.   55.    0.    0.    0.]]
Expected cases: 1014.0  Prediction: 928.2089 Expected fatalities: 55.0  Prediction: 44.024376
================================================
[[1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [2. 1. 0. 0. 0.]
 [2. 1. 0. 0. 0.]
 [2. 1. 0. 0. 0.]
 [2. 1. 0. 0. 0.]
 [2. 1. 0. 0. 0.]
 [3. 1. 0. 0. 0.]
 [3. 1. 0. 0. 0.]
 [3. 1. 0. 0. 0.]
 [3. 1. 0. 0. 0.]
 [3. 1. 0. 0. 0.]
 [3. 1. 0. 0. 0.]
 [3. 1. 0. 0. 0.]
 [3. 1. 0. 0. 0.]
 [3. 1. 0. 0. 0.]
 [3. 1. 0. 0. 0.]
 [3. 1. 0. 0. 0.]
 [3. 1. 0. 0. 0.]]
Expected cases: 3.0  Prediction: 3.0475361 Expected fatalities: 1.0  Prediction: 1.1656009
================================================
[[ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 4.  0.  0.  0.  0.]
 [ 4.  0.  0.  0.  0.]
 [ 4.  0.  0.  0.  0.]
 [ 8.  0.  0.  0.  0.]
 [ 8.  0.  0.  0.  0.]
 [13.  0.  0.  0.  0.]]
Expected cases: 23.0  Prediction: 16.787846 Expected fatalities: 0.0  Prediction: -0.12304173
================================================
[[2.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [2.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [3.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [4.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [4.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [4.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [8.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [9.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [1.700e+01 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [1.700e+01 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [2.400e+01 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [5.000e+01 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [7.400e+01 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [9.400e+01 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [1.210e+02 1.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [1.390e+02 1.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [1.810e+02 5.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [2.190e+02 4.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [6.280e+02 4.000e+00 0.000e+00 0.000e+00 0.000e+00]
 [1.013e+03 4.000e+00 0.000e+00 0.000e+00 0.000e+00]]
Expected cases: 1342.0  Prediction: 726.8143 Expected fatalities: 6.0  Prediction: 7.161662
================================================
[[ 9.  0.  0.  0.  0.]
 [10.  0.  0.  0.  0.]
 [10.  0.  0.  0.  0.]
 [10.  0.  0.  0.  1.]
 [10.  0.  0.  0.  1.]
 [11.  0.  0.  0.  1.]
 [11.  0.  0.  0.  1.]
 [16.  0.  0.  0.  1.]
 [16.  0.  0.  0.  1.]
 [17.  0.  0.  0.  1.]
 [18.  0.  0.  0.  1.]
 [18.  0.  0.  0.  1.]
 [18.  0.  0.  0.  1.]
 [18.  0.  0.  0.  1.]
 [18.  0.  0.  0.  1.]
 [18.  0.  0.  0.  1.]
 [18.  0.  0.  0.  1.]
 [20.  1.  0.  0.  1.]
 [22.  1.  0.  0.  1.]
 [22.  1.  0.  0.  1.]]
Expected cases: 23.0  Prediction: 21.922379 Expected fatalities: 1.0  Prediction: 1.0475128
================================================
[[  0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.]
 [  5.   0.   0.   0.   0.]
 [  6.   0.   0.   0.   0.]
 [ 12.   0.   0.   0.   0.]
 [ 29.   0.   0.   0.   0.]
 [ 39.   0.   0.   0.   0.]
 [ 46.   0.   0.   0.   0.]
 [ 78.   0.   0.   0.   0.]
 [ 83.   0.   0.   0.   0.]
 [131.   0.   0.   0.   0.]
 [157.   0.   0.   0.   0.]
 [196.   0.   0.   0.   0.]
 [242.   0.   0.   0.   0.]]
Expected cases: 381.0  Prediction: 269.27927 Expected fatalities: 1.0  Prediction: 0.11550885
================================================
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]]
Expected cases: 1.0  Prediction: 1.2144642 Expected fatalities: 0.0  Prediction: -0.2051672
================================================
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [2. 0. 0. 0. 0.]
 [2. 0. 0. 0. 0.]
 [3. 0. 0. 0. 0.]
 [3. 0. 0. 0. 0.]]
Expected cases: 3.0  Prediction: 4.040323 Expected fatalities: 0.0  Prediction: -0.1370603
================================================
[[ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 5.  0.  0.  0.  0.]
 [ 7.  0.  0.  0.  0.]
 [ 7.  0.  0.  0.  0.]
 [13.  0.  0.  0.  0.]
 [19.  0.  0.  0.  0.]
 [26.  0.  0.  0.  0.]
 [32.  0.  0.  0.  0.]]
Expected cases: 35.0  Prediction: 41.180374 Expected fatalities: 2.0  Prediction: -0.07844072
================================================
[[16.  0.  0.  0.  0.]
 [16.  0.  0.  0.  0.]
 [16.  0.  0.  0.  0.]
 [16.  0.  0.  0.  0.]
 [16.  0.  0.  0.  0.]
 [16.  0.  0.  0.  0.]
 [16.  0.  0.  0.  0.]
 [16.  0.  0.  0.  0.]
 [16.  0.  0.  0.  0.]
 [16.  0.  0.  0.  0.]
 [16.  0.  0.  0.  0.]
 [16.  0.  0.  0.  0.]
 [16.  0.  0.  0.  0.]
 [16.  0.  0.  0.  0.]
 [16.  0.  0.  0.  0.]
 [16.  0.  0.  0.  0.]
 [18.  0.  0.  0.  0.]
 [30.  0.  0.  0.  0.]
 [30.  0.  0.  0.  0.]
 [31.  0.  0.  0.  0.]]
Expected cases: 38.0  Prediction: 37.558662 Expected fatalities: 0.0  Prediction: -0.061804075
================================================
[[  0.   0.   0.   0.   0.]
 [  1.   0.   0.   0.   0.]
 [  1.   0.   0.   0.   0.]
 [  3.   0.   0.   0.   0.]
 [  6.   0.   0.   0.   0.]
 [ 11.   0.   0.   0.   0.]
 [ 26.   0.   0.   0.   0.]
 [ 34.   0.   0.   0.   0.]
 [ 43.   0.   0.   0.   0.]
 [ 50.   0.   0.   0.   0.]
 [ 50.   0.   0.   0.   0.]
 [ 58.   0.   0.   0.   0.]
 [ 69.   0.   0.   0.   0.]
 [ 85.   0.   0.   0.   0.]
 [103.   0.   0.   0.   0.]
 [134.   0.   0.   0.   0.]
 [156.   0.   0.   0.   0.]
 [171.   5.   0.   0.   0.]
 [180.   0.   0.   0.   0.]
 [220.   1.   0.   0.   0.]]
Expected cases: 250.0  Prediction: 234.38689 Expected fatalities: 1.0  Prediction: 1.2801368
================================================
[[  3.   0.   0.   0.   0.]
 [  4.   0.   0.   0.   0.]
 [  4.   0.   0.   0.   0.]
 [  7.   0.   0.   0.   0.]
 [  7.   0.   0.   0.   0.]
 [  7.   0.   0.   0.   0.]
 [  9.   0.   0.   0.   0.]
 [ 31.   0.   0.   0.   0.]
 [ 45.   0.   0.   0.   0.]
 [ 46.   0.   0.   0.   0.]
 [ 73.   0.   0.   0.   0.]
 [ 73.   0.   0.   0.   0.]
 [ 89.   0.   0.   0.   0.]
 [ 99.   1.   0.   0.   0.]
 [ 99.   1.   0.   0.   0.]
 [190.   1.   0.   0.   0.]
 [228.   3.   0.   0.   0.]
 [331.   4.   0.   0.   0.]
 [331.   4.   0.   0.   0.]
 [387.   5.   0.   0.   0.]]
Expected cases: 418.0  Prediction: 510.16367 Expected fatalities: 5.0  Prediction: 6.1775312
================================================
[[ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 2.  0.  0.  0.  0.]
 [10.  0.  0.  0.  0.]]
Expected cases: 12.0  Prediction: 14.965948 Expected fatalities: 1.0  Prediction: -0.17472637
================================================
[[117.   1.   0.   0.   0.]
 [119.   1.   0.   0.   0.]
 [119.   1.   0.   0.   0.]
 [121.   1.   0.   0.   0.]
 [121.   1.   0.   0.   0.]
 [121.   1.   0.   0.   0.]
 [121.   1.   0.   0.   0.]
 [121.   1.   0.   0.   0.]
 [121.   1.   0.   0.   0.]
 [121.   1.   0.   0.   0.]
 [121.   1.   0.   0.   0.]
 [121.   1.   0.   0.   0.]
 [121.   1.   0.   0.   0.]
 [121.   1.   0.   0.   0.]
 [121.   1.   0.   0.   0.]
 [121.   1.   0.   0.   0.]
 [121.   1.   0.   0.   0.]
 [122.   1.   0.   0.   0.]
 [122.   1.   0.   0.   0.]
 [125.   1.   0.   0.   0.]]
Expected cases: 125.0  Prediction: 125.46737 Expected fatalities: 1.0  Prediction: 0.89293855
================================================
[[  3.   1.   0.   0.   0.]
 [  3.   1.   0.   0.   0.]
 [  3.   1.   0.   0.   0.]
 [  3.   1.   0.   0.   0.]
 [  4.   1.   0.   0.   0.]
 [  6.   1.   0.   0.   0.]
 [  9.   1.   0.   0.   0.]
 [  9.   1.   0.   0.   0.]
 [ 14.   1.   0.   0.   0.]
 [ 17.   1.   0.   0.   0.]
 [ 17.   1.   0.   0.   0.]
 [ 28.   1.   0.   0.   0.]
 [ 31.   1.   0.   0.   0.]
 [ 35.   1.   0.   0.   0.]
 [ 52.   1.   0.   0.   0.]
 [ 64.   1.   0.   0.   0.]
 [ 90.   1.   0.   0.   0.]
 [120.   1.   0.   0.   0.]
 [140.   1.   0.   0.   0.]
 [175.   1.   0.   0.   0.]]
Expected cases: 175.0  Prediction: 205.62349 Expected fatalities: 1.0  Prediction: 1.3014854
================================================
[[ 58.   0.   0.   0.   0.]
 [ 67.   0.   0.   0.   0.]
 [ 72.   0.   0.   0.   0.]
 [ 75.   0.   0.   0.   0.]
 [ 77.   0.   0.   0.   0.]
 [ 81.   0.   0.   0.   0.]
 [ 84.   0.   0.   0.   0.]
 [ 84.   0.   0.   0.   0.]
 [ 85.   0.   0.   0.   0.]
 [ 85.   0.   0.   0.   0.]
 [ 89.   0.   0.   0.   0.]
 [ 89.   0.   0.   0.   0.]
 [ 91.   0.   0.   0.   0.]
 [ 93.   0.   0.   0.   0.]
 [ 93.   0.   0.   0.   0.]
 [ 93.   0.   0.   0.   0.]
 [102.   0.   0.   0.   0.]
 [106.   0.   0.   0.   0.]
 [108.   0.   0.   0.   0.]
 [110.   0.   0.   0.   0.]]
Expected cases: 110.0  Prediction: 112.03973 Expected fatalities: 0.0  Prediction: -0.025774306
================================================
[[ 9.  0.  0.  0.  0.]
 [ 9.  0.  0.  0.  0.]
 [ 9.  0.  0.  0.  0.]
 [ 9.  0.  0.  0.  0.]
 [ 9.  0.  0.  0.  0.]
 [ 9.  0.  0.  0.  0.]
 [ 9.  0.  0.  0.  0.]
 [ 9.  0.  0.  0.  0.]
 [ 9.  0.  0.  0.  0.]
 [ 9.  0.  0.  0.  0.]
 [ 9.  0.  0.  0.  0.]
 [13.  0.  0.  0.  0.]
 [13.  0.  0.  0.  0.]
 [13.  0.  0.  0.  0.]
 [15.  0.  0.  0.  0.]
 [20.  0.  0.  0.  0.]
 [23.  0.  0.  0.  0.]
 [36.  0.  0.  0.  0.]
 [40.  0.  0.  0.  0.]
 [51.  0.  0.  0.  0.]]
Expected cases: 85.0  Prediction: 61.38384 Expected fatalities: 0.0  Prediction: -0.04931904
```

6. Apply the model to predict future trends

The following functions will be used to get the 13 previous days from a given date and demographic information, predict the number of cases and fatalities, before iterating again. Therefore, it will use the prediction for the next day as part of the data for the one afterwards.

In [26]:
```python
#Will retrieve the number of cases and fatalities for the past 6 days from the given date
def build_inputs_for_date(country, province, date, df):
    start_date = date - timedelta(days=20)
    end_date = date - timedelta(days=1)

    str_start_date = start_date.strftime("%Y-%m-%d")
    str_end_date = end_date.strftime("%Y-%m-%d")
    df = df.query("Country_Region=='"+country+"' and Province_State=='"+province+"' and Date>='"+str_start_date+"' and Date<='"+str_end_date+"'")

    #preparing the temporal inputs
    temporal_input_data = np.transpose(np.reshape(np.asarray([df["ConfirmedCases"],
                                                 df["Fatalities"],
                                                 df["restrictions"],
                                                 df["quarantine"],
                                                 df["schools"]]),
                                     (5,sequence_length)), (1,0) ).astype(np.float32)

    #preparing all the demographic inputs
    total_population = float(province_df.iloc[i].total_pop)
    density = float(province_df.iloc[i].density)
    hospibed = float(province_df.iloc[i].hospibed)
    lung = float(province_df.iloc[i].lung)
    centenarian_pop = float(province_df.iloc[i]["age_100+"])
    demographic_input_data = [total_population,density,hospibed,lung,centenarian_pop]

    return [np.array([temporal_input_data]), np.array([demographic_input_data])]
```

In [27]:
```python
#Take a dataframe in input, will do the predictions and return the dataframe with extra rows
#containing the predictions
def predict_for_region(country, province, df):
    begin_prediction = "2020-04-01"
    start_date = datetime.strptime(begin_prediction,"%Y-%m-%d")
    end_prediction = "2020-05-14"
    end_date = datetime.strptime(end_prediction,"%Y-%m-%d")

    date_list = [start_date + timedelta(days=x) for x in range((end_date-start_date).days+1)]
    for date in date_list:
        input_data = build_inputs_for_date(country, province, date, df)
        result = model.predict(input_data)

        #just ensuring that the outputs is
        #higher than the previous counts
        result[0] = np.round(result[0])
        if result[0]<input_data[0][0][-1][0]:
            result[0]=np.array([[input_data[0][0][-1][0]]])

        result[1] = np.round(result[1])
        if result[1]<input_data[0][0][-1][1]:
            result[1]=np.array([[input_data[0][0][-1][1]]])

        #We assign the quarantine and school status
        #depending on previous values
        #e.g Once a country is locked, it will stay locked until the end
        df = df.append({"Country_Region":country,
                        "Province_State":province,
                        "Date":date.strftime("%Y-%m-%d"),
                        "restrictions": 1 if any(input_data[0][0][2]) else 0,
                        "quarantine": 1 if any(input_data[0][0][3]) else 0,
                        "schools": 1 if any(input_data[0][0][4]) else 0,
                        "total_pop": input_data[1][0],
                        "density": input_data[1][0][1],
                        "hospibed": input_data[1][0][2],
                        "lung": input_data[1][0][3],
                        "age_100+": input_data[1][0][4],
                        "ConfirmedCases":round(result[0][0][0]),
                        "Fatalities":round(result[1][0][0])},
                       ignore_index=True)
    return df
```

In [28]:
```python
#The functions that are called here need to optimise, sorry about that!
copy_df = train_df
with tqdm(total=len(list(copy_df.Country_Region.unique()))) as pbar:
    for country in copy_df.Country_Region.unique():
        for province in copy_df.query("Country_Region=='"+country+"'").Province_State.unique():
            copy_df = predict_for_region(country, province, copy_df)
        pbar.update(1)
```

```
100%|██████████| 313/313 [31:21<00:00,  6.01s/it]
```

7. Check the model's performance for the beginning of April

In [29]:
```python
groundtruth_df = gpd.read_file("/kaggle/input/covid19-global-forecasting-week-4/train.csv")

groundtruth_df["ConfirmedCases"] = groundtruth_df["ConfirmedCases"].astype("float")
groundtruth_df["Fatalities"] = groundtruth_df["Fatalities"].astype("float")
#The country_region got modifying in the enriched dataset by @optimo,
# so we have to apply the same change to this Dataframe.
groundtruth_df["Country_Region"] = [ row.Country_Region.replace("'","").strip(" ") if row.Province_State=="" else str(row.Country_Region+"_"+row.Province_State).replace("'","").strip(" ") for idx,row in groundtruth_df.iterrows()]

last_date = groundtruth_df.Date.unique()[-1]
```

In [30]:
```python
#to remove annoying warnings from pandas
pd.options.mode.chained_assignment = None

def get_RMSLE_per_region(region, groundtruth_df, display_only=False):
    groundtruth_df["ConfirmedCases"] = groundtruth_df["ConfirmedCases"].astype("float")
    groundtruth_df["Fatalities"] = groundtruth_df["Fatalities"].astype("float")

    #we only take data until the 30th of March 2020 as the groundtruth was not available for later dates.
    groundtruth = groundtruth_df.query("Country_Region=='"+region+"' and Date>='2020-04-01' and Date<='"+last_date+"'")
    predictions = copy_df.query("Country_Region=='"+region+"' and Date>='2020-04-01' and Date<='"+last_date+"'")

    RMSLE_cases = np.sqrt(mean_squared_log_error( groundtruth.ConfirmedCases.values, predictions.ConfirmedCases.values ))
    RMSLE_fatalities = np.sqrt(mean_squared_log_error( groundtruth.Fatalities.values, predictions.Fatalities.values ))

    if display_only:
        print(region)
        print("RMSLE on cases:",np.mean(RMSLE_cases))
        print("RMSLE on fatalities:",np.mean(RMSLE_fatalities))
    else:
        return RMSLE_cases, RMSLE_fatalities
```

In [31]:
```python
def get_RMSLE_for_all_regions(groundtruth_df):
    RMSLE_cases_list = []
    RMSLE_fatalities_list = []
    for region in groundtruth_df.Country_Region.unique():
        RMSLE_cases, RMSLE_fatalities = get_RMSLE_per_region(region, groundtruth_df, False)
        RMSLE_cases_list.append(RMSLE_cases)
        RMSLE_fatalities_list.append(RMSLE_fatalities)
    print("RMSLE on cases:",np.mean(RMSLE_cases_list))
    print("RMSLE on fatalities:",np.mean(RMSLE_fatalities_list))
```

We can first check whether one of the outputs is globally harder to predict than the other.

In [32]:
```python
get_RMSLE_for_all_regions(groundtruth_df)
```

```
RMSLE on cases: 0.49494465612276073
RMSLE on fatalities: 0.3770080063743591
```

Now, let's check how we are performing on two groups:

- Countries known to have an outbreak
- Countries with relatively "few" cases

```
In [33]: badly_affected_countries = ["France","Italy","United Kingdom","Spain","Iran","Germany"]
         for country in badly_affected_countries:
             get_RMSLE_per_region(country, groundtruth_df, display_only=True)
```

```
France
RMSLE on cases: 0.6055889849865056
RMSLE on fatalities: 0.917603893928617
Italy
RMSLE on cases: 0.21993188939467181
RMSLE on fatalities: 0.2790982583478945
United Kingdom
RMSLE on cases: 0.7023263322485703
RMSLE on fatalities: 1.0985807570598574
Spain
RMSLE on cases: 0.3452272671599796
RMSLE on fatalities: 0.4424718447195897
Iran
RMSLE on cases: 0.2918973895938459
RMSLE on fatalities: 0.24915402316605503
Germany
RMSLE on cases: 0.36357028547960063
RMSLE on fatalities: 0.8511859973565415
```

```
In [34]: healthy_countries = ["Taiwan*","Singapore","Kenya","Slovenia","Portugal", "Israel"]
         for country in healthy_countries:
             get_RMSLE_per_region(country, groundtruth_df, display_only=True)
```

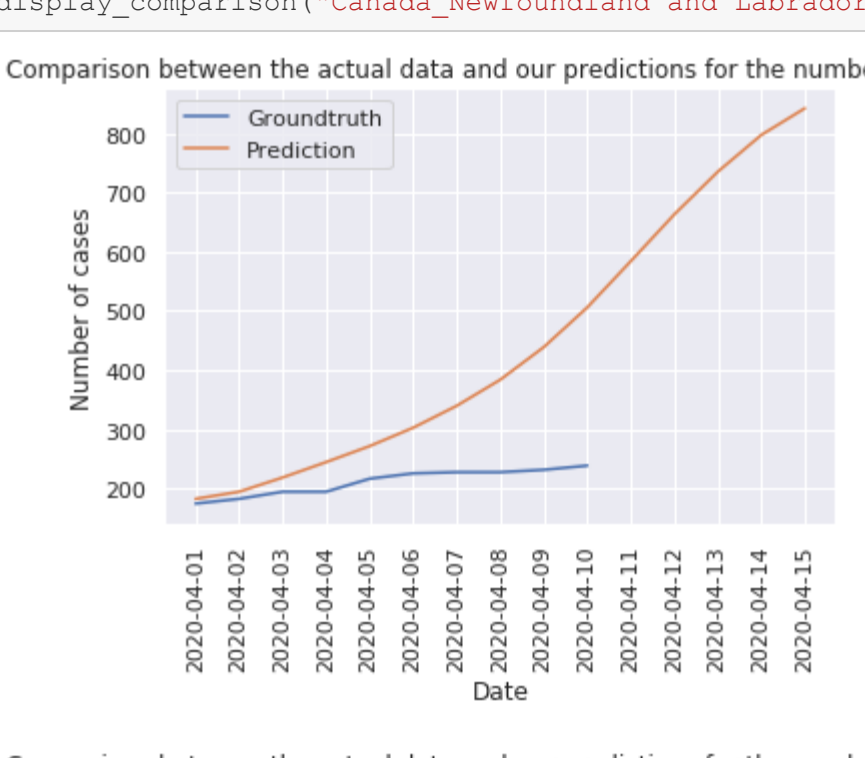
```
Taiwan*
RMSLE on cases: 0.4725040374958486
RMSLE on fatalities: 0.4312974401612734
Singapore
RMSLE on cases: 0.42418201374459735
RMSLE on fatalities: 0.3024619370927203
Kenya
RMSLE on cases: 0.4533514542584227
RMSLE on fatalities: 1.074759899089988
Slovenia
RMSLE on cases: 0.12575939557457347
RMSLE on fatalities: 0.1639770020679584
Portugal
RMSLE on cases: 0.4632140222221402
RMSLE on fatalities: 0.2443228659880887
Israel
RMSLE on cases: 0.4725838196781893
RMSLE on fatalities: 0.22815509227844558
```

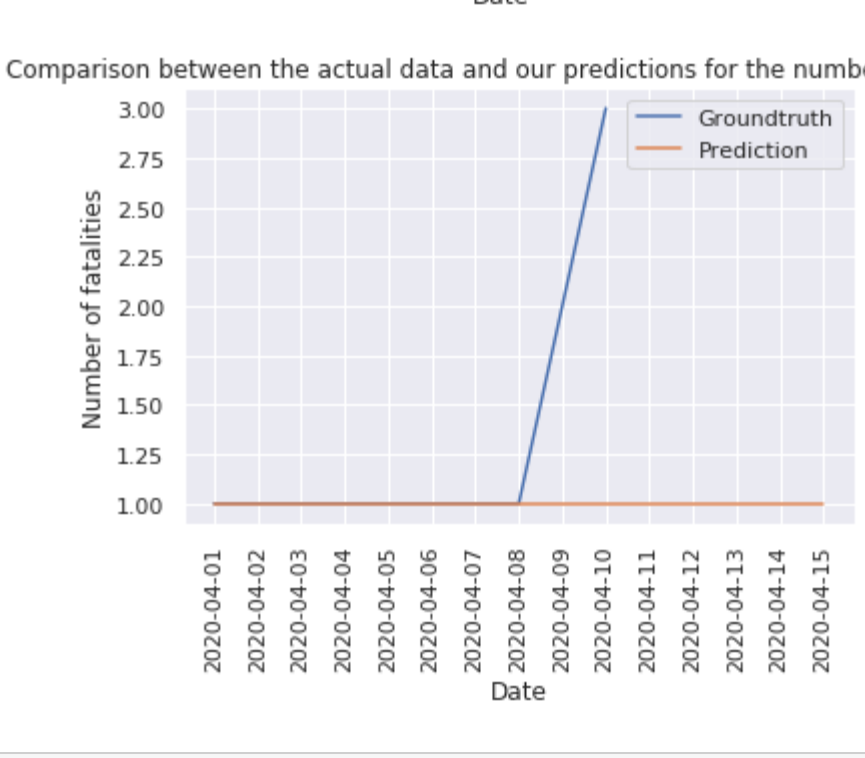
8. Outputs: Observing the curves

I am using the original training CSV file instead of the enriched dataset as it is up-to-date with the latest stats from this week.

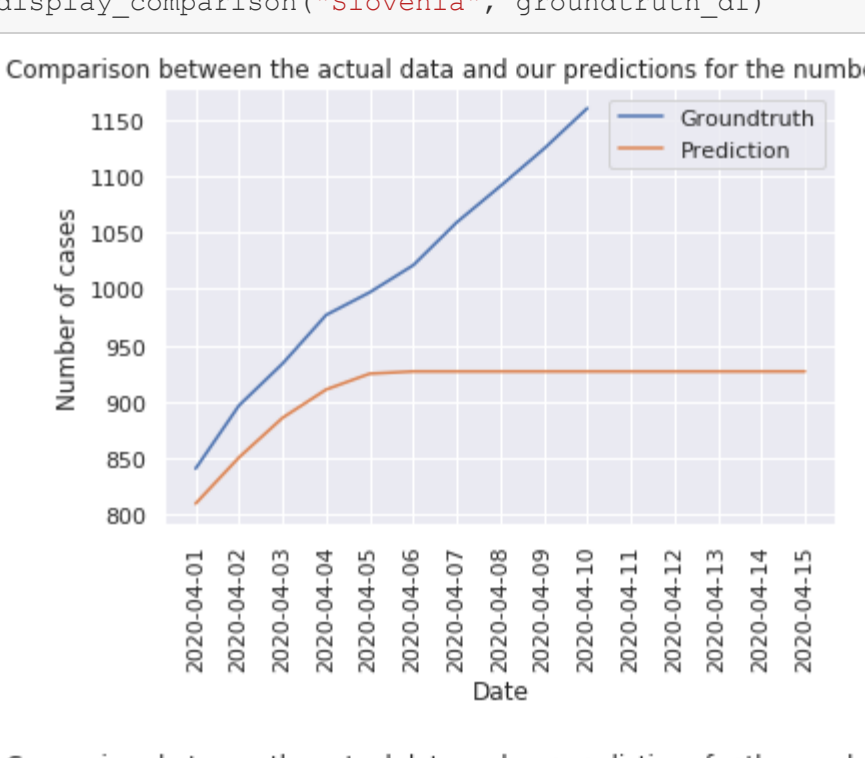
```
In [35]: def display_comparison(region,groundtruth_df):
             groundtruth = groundtruth_df.query("Country_Region=='"+region+"' and Date>='2020-04-01' and Date<
         ='2020-04-15'")
             prediction = copy_df.query("Country_Region=='"+region+"' and Date>='2020-04-01' and Date<='2020-04-
         15'")

             plt.plot(groundtruth.ConfirmedCases.values)
             plt.plot(prediction.ConfirmedCases.values)
             plt.title("Comparison between the actual data and our predictions for the number of cases")
             plt.ylabel('Number of cases')
             plt.xlabel('Date')
             plt.xticks(range(len(prediction.Date.values)),prediction.Date.values,rotation='vertical')
             plt.legend(['Groundtruth', 'Prediction'], loc='best')
             plt.show()

             plt.plot(groundtruth.Fatalities.values)
             plt.plot(prediction.Fatalities.values)
             plt.title("Comparison between the actual data and our predictions for the number of fatalities")
             plt.ylabel('Number of fatalities')
             plt.xlabel('Date')
             plt.xticks(range(len(prediction.Date.values)),prediction.Date.values,rotation='vertical')
             plt.legend(['Groundtruth', 'Prediction'], loc='best')
             plt.show()
```

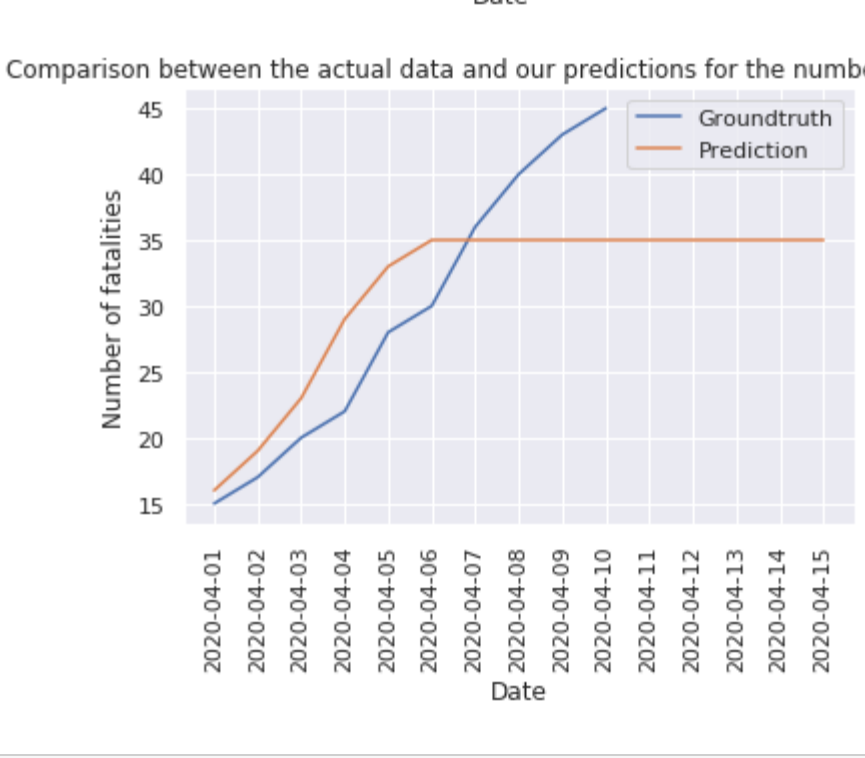
```
In [36]: display_comparison("Canada_Newfoundland and Labrador", groundtruth_df)
```



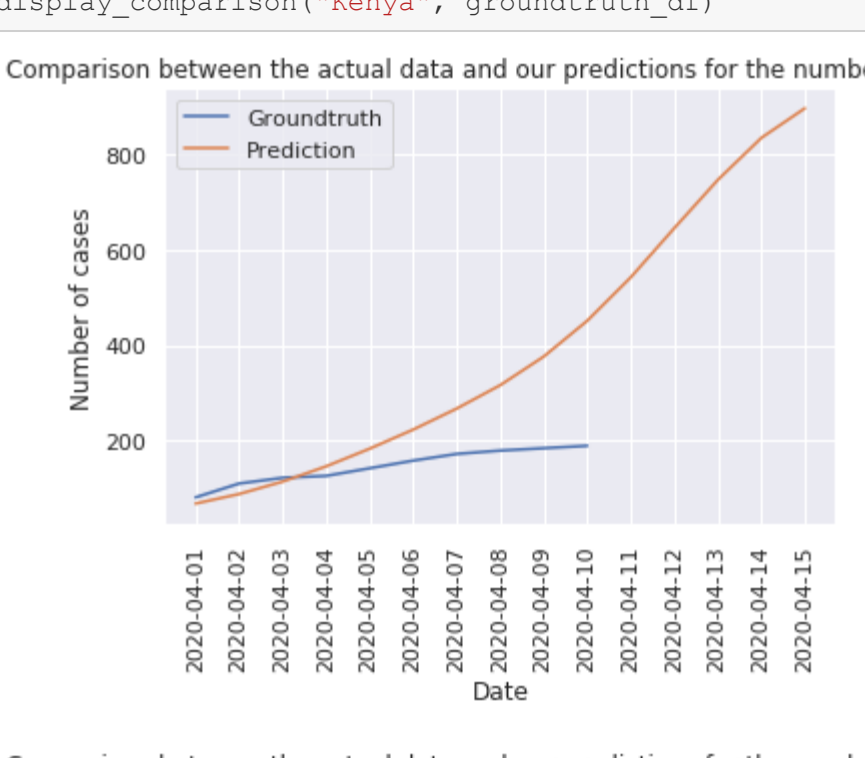


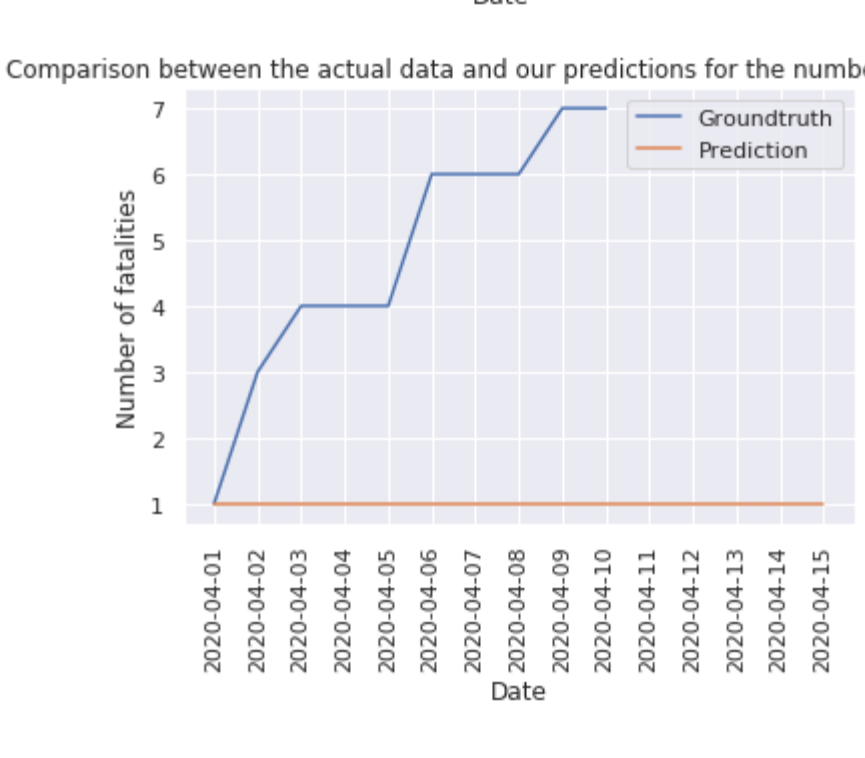

```
In [37]: display_comparison("Slovenia", groundtruth_df)
```






```
In [38]: display_comparison("Kenya", groundtruth_df)
```

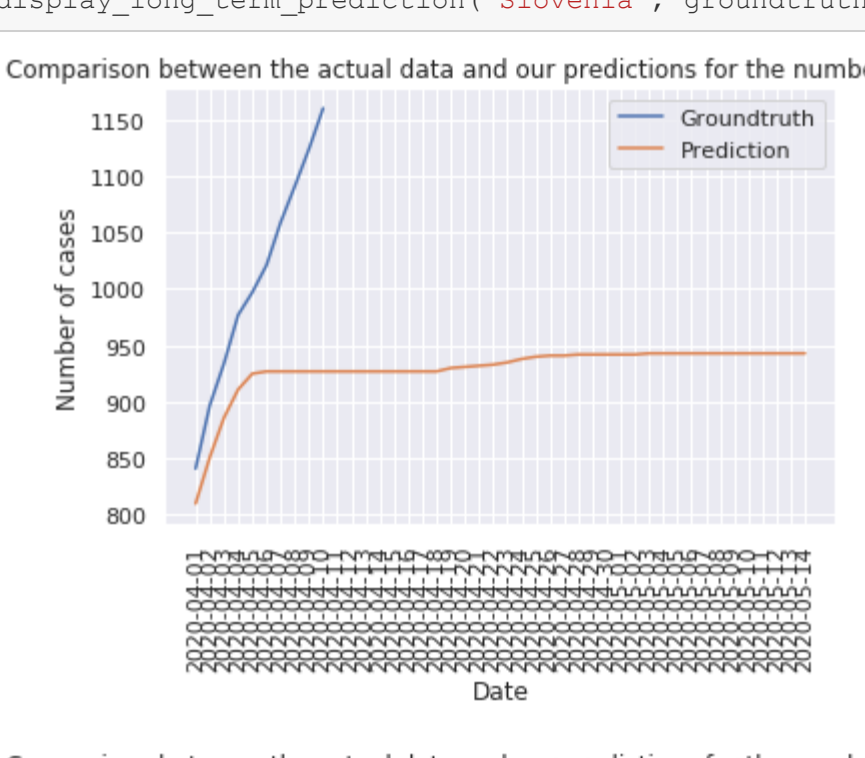


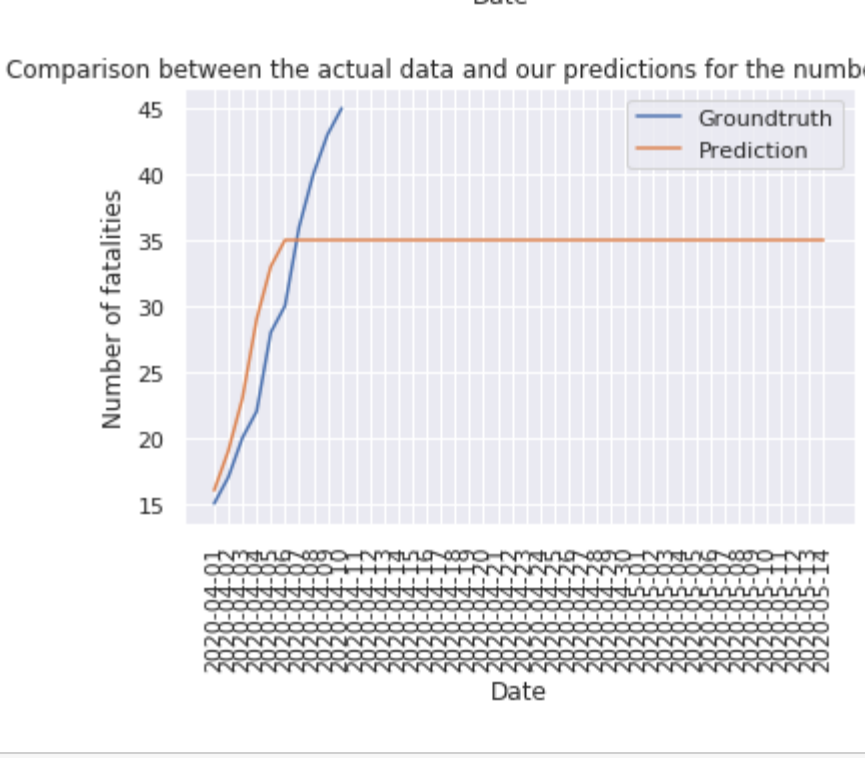


Now let's see how well the model perform on a much longer period.

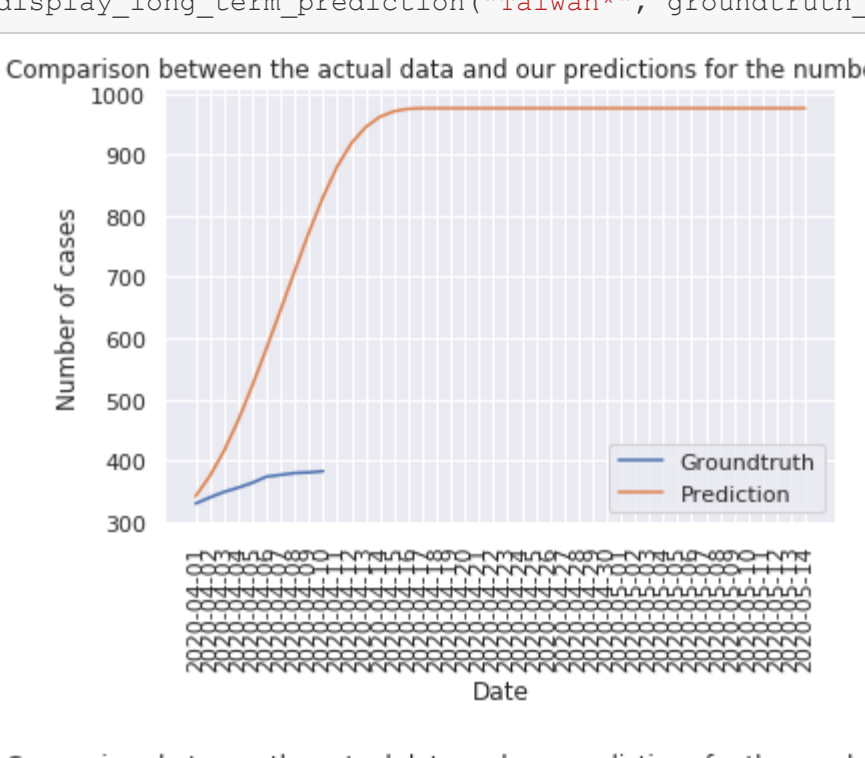
```
In [39]: def display_long_term_prediction(region,groundtruth_df):
             groundtruth = groundtruth_df.query("Country_Region=='"+region+"' and Date>='2020-04-01' and Date<
         ='2020-04-15'")
             prediction = copy_df.query("Country_Region=='"+region+"' and Date>='2020-04-01' and Date<='2020-05-
         14'")

             plt.plot(groundtruth.ConfirmedCases.values)
             plt.plot(prediction.ConfirmedCases.values)
             plt.title("Comparison between the actual data and our predictions for the number of cases")
             plt.ylabel('Number of cases')
             plt.xlabel('Date')
             plt.xticks(range(len(prediction.Date.values)),prediction.Date.values,rotation='vertical')
             plt.legend(['Groundtruth', 'Prediction'], loc='best')
             plt.show()

             plt.plot(groundtruth.Fatalities.values)
             plt.plot(prediction.Fatalities.values)
             plt.title("Comparison between the actual data and our predictions for the number of fatalities")
             plt.ylabel('Number of fatalities')
             plt.xlabel('Date')
             plt.xticks(range(len(prediction.Date.values)),prediction.Date.values,rotation='vertical')
             plt.legend(['Groundtruth', 'Prediction'], loc='best')
             plt.show()
```

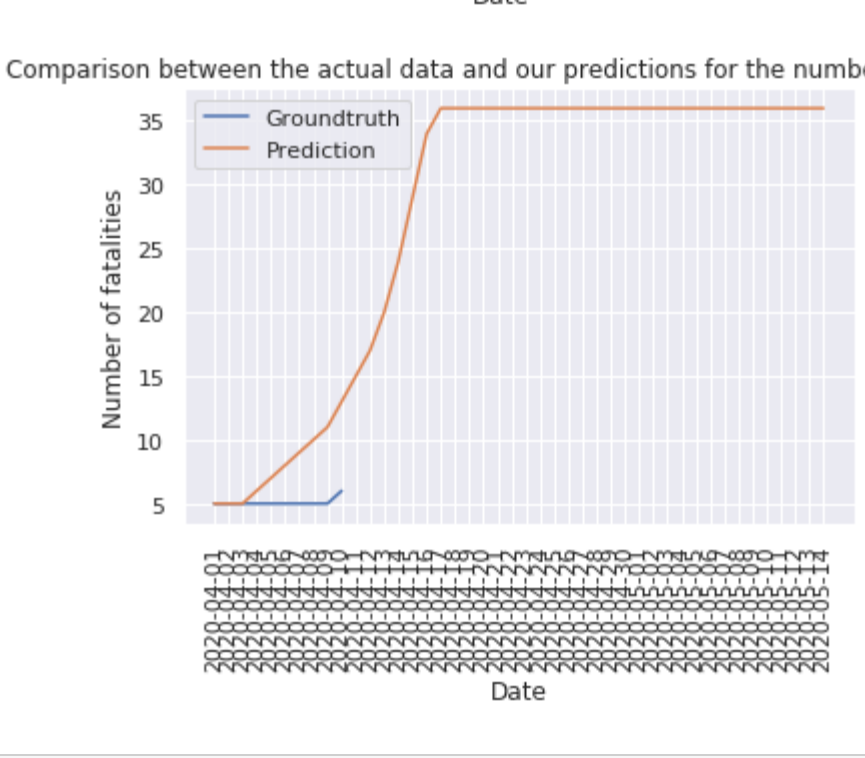
```
In [40]: display_long_term_prediction("Slovenia", groundtruth_df)
```



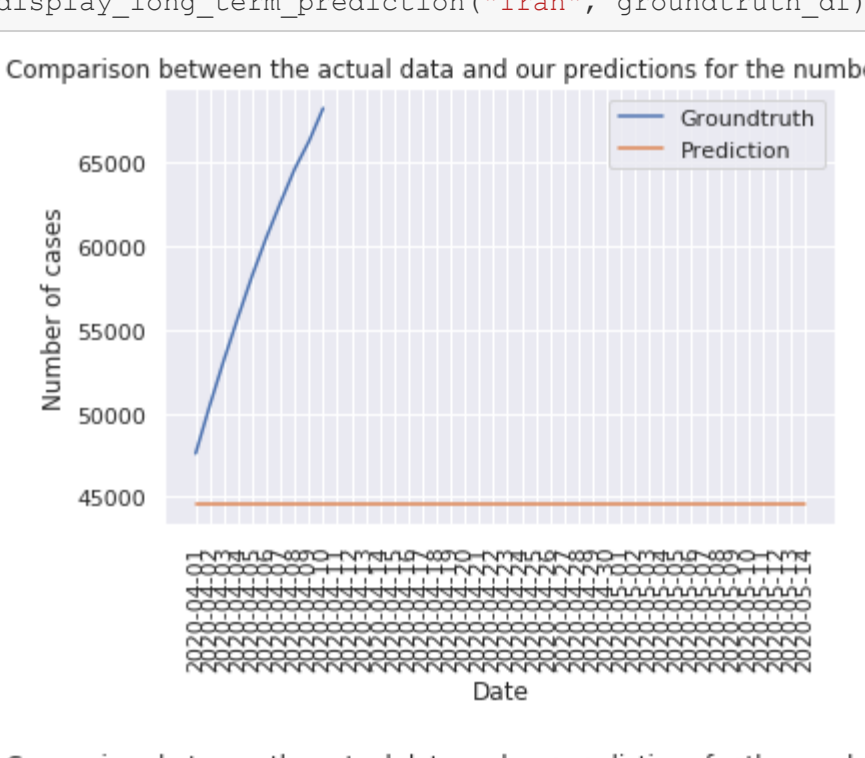


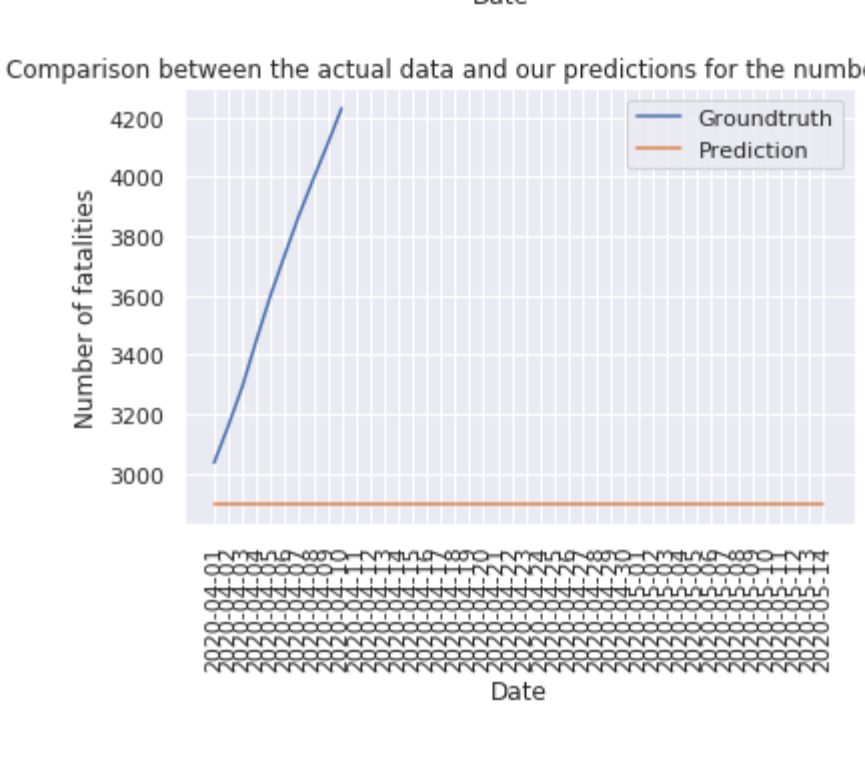

```
In [41]: display_long_term_prediction("Taiwan*", groundtruth_df)
```






```
In [42]: display_long_term_prediction("Iran", groundtruth_df)
```





9. Generating the submission file

```
In [43]: test_df = gpd.read_file("/kaggle/input/covid19-global-forecasting-week-4/test.csv")
         #The country_region got modifying in the enriched dataset by @optimo,
         # so we have to apply the same change to the test Dataframe.
         test_df["Country_Region"] = [ row.Country_Region if row.Province_State=="" else row.Country_Region+"_"+
         row.Province_State for idx,row in test_df.iterrows() ]
         test_df.head()
```

Out[43]:

	ForecastId	Province_State	Country_Region	Date	geometry
0	1		Afghanistan	2020-04-02	None
1	2		Afghanistan	2020-04-03	None
2	3		Afghanistan	2020-04-04	None
3	4		Afghanistan	2020-04-05	None
4	5		Afghanistan	2020-04-06	None

Just need to do this little trick to extract the relevant date and the forecastId and add that to the submission file.

```
In [44]: submission_df = pd.DataFrame(columns=["ForecastId","ConfirmedCases","Fatalities"])
         with tqdm(total=len(test_df)) as pbar:
             for idx, row in test_df.iterrows():
                 #Had to remove single quotes because of countries like Cote D'Ivoire for example
                 country_region = row.Country_Region.replace("'","").strip(" ")
                 province_state = row.Province_State.replace("'","").strip(" ")
                 item = copy_df.query("Country_Region=='"+country_region+"' and Province_State=='"+province_stat
         e+"' and Date=='"+row.Date+"'")
                 submission_df = submission_df.append({"ForecastId":row.ForecastId,
                                                       "ConfirmedCases":int(item.ConfirmedCases.values[0]),
                                                       "Fatalities":int(item.Fatalities.values[0])},
                                                      ignore_index=True)
                 pbar.update(1)

100%|██████████| 13459/13459 [03:34<00:00, 62.84it/s]
```

```
In [45]: submission_df.sample(20)
```

Out[45]:

	ForecastId	ConfirmedCases	Fatalities
10528	10529	954	44
9529	9530	0	0
5626	5627	981	0
10196	10197	987	47
8367	8368	1938	329
4995	4996	567	13
3881	3882	978	27
3323	3324	861	10
8092	8093	5	1
6418	6419	930	36
6159	6160	914	74
13274	13275	866	0
10595	10596	6741	135
1223	1224	933	41
2790	2791	943	35
11621	11622	75833	1550
7652	7653	949	37
10362	10363	960	42
1867	1868	970	1
8928	8929	983	0

```
In [46]: submission_df.to_csv("submission.csv",index=False)
```

10. Conclusion

As there is still "little" data available for such kind of model, it was clearly an ambitious approach but I was curious to see how far we could go with such solution. It appears that the model was reasonably good at predicting the earlier stages of the spread when the numbers are still in the few hundreds, but then it seems to plateau far too early. It also appears to be performing awfully in countries when the outbreaks are already at an advanced stage. When looking back at the version notes, it seems that adding information about the regions improved the model's performance in a minor with significant manner. Information about the restrictions on populations over time showed more impact than demographic information such as the number of hospital beds or density of population. While many other models have demonstrated far better results, I think this "naive" model could be interesting on the long run as more data become available, mostly due to how easy it is to integrate more new inputs to the model.

If you found this notebook helpful, please give it an upvote. It will be greatly appreciated!