**Don Bosco Institute and Technology, Kurla(W), Mumbai-400 070**

**DEPARTMENT OF COMPUTER ENGINEERING**

**(Session 2019-20 EVEN)**

**Subject: Mini Project (CSM 605)**

**Third Year Computer Engineering**

**“Coronavirus Prediction”**

**A Project Report**

***Submitted by:***

|  |  |  |
| --- | --- | --- |
| ***Sr. No.*** | ***Roll No.*** | ***Name of Student*** |
| *1* | *05* | *Yash Bitla* |
| *2* | *07* | *Arjun Chavan* |
| *3* | *37* | *Taaha Kazi* |

**Don Bosco Institute and Technology, Kurla(W), Mumbai-400 070**

**DEPARTMENT OF COMPUTER ENGINEERING**

**DECLARATION**

We hereby declare that the project entitled “**Coronavirus Prediction**” submitted for the Mini Project (CSM 605) of third year (Computer Engineering) of Mumbai University syllabus course work is our original work/hypothesis/algorithm design/mathematical modelling and result analysis for the specific Machine Learning algorithm.

**Name:** Yash Bitla **Name:** Arjun Chavan **Name:** Taaha Kazi

**Signature of the Student- Signature of the Student- Signature of the Student-**

**Place: Mumbai**

**Date:**

**Don Bosco Institute and Technology, Kurla(W), Mumbai-400 070**

**DEPARTMENT OF COMPUTER ENGINEERING**

**CERTIFICATE**

This is to certify that the project titled “**Coronavirus Prediction**” is the bonafide work carried out by **Yash Bitla**, **Arjun Chavan & Taaha Kazi** the students of Third Year Computer Engineering of Department of Computer Engineering, Don Bosco Institute of Technology, Kurla (W), Mumbai-400 070 affiliated to Mumbai University, Mumbai, Maharashtra (India) during the academic year 2018-19, in coursework completion of subject Mini Project (CSM) of 6th semester

**Name & Signature of the Guide**

**Place: Mumbai**

**Date:**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | Abstract | i |
|  | Acknowledgement | ii |
|  | List of Figures | iii |
|  | List of Tables (optional) | iv |
|  | Timeline / Gantt Chart | v |
|  |  |  |
| **1.** | **INTRODUCTION** |  |
|  | * 1. Problem Definition   2. Project Overview/Specifications   3. Hardware Specification   4. Software Specification | 7  8  10  10 |
|  |  |  |
| **2.** | **LITERATURE SURVEY** |  |
|  | 2.1 Existing System  2.2 Proposed System  2.3 Feasibility Study | 11  12  13 |
|  |  |  |
| **3.** | **SYSTEM ANALYSIS & DESIGN** |  |
|  | 3.1 Requirement Specification  3.2 Flowcharts / DFDs / ERDs  3.3 Design and Test Steps / Criteria  3.3 Algorithms and Pseudo Code  3.3.1 Algorithms  3.3.2 Psuedo Code | 14  20  22  24  25 |
|  |  |  |
| **4.** | **RESULTS / OUTPUTS** | 28 |
| **5.** | **CONCLUSIONS / RECOMMENDATIONS** | 31 |
| **6.** | **REFERENCES** | 32 |

**Abstract**

Currently, the entire world is undergoing a pandemic of unprecedented standards. This pandemic is caused by a virus commonly known as “coronavirus”. This virus has touched almost every nation and has caused terror amongst many citizens. This virus has a spread rate of “2” under uncontrolled circumstances. That is if there are no restrictions for the people, then it will be that every person is likely to spread the virus to two others. The other major concern regarding this virus is that it remains asymptomatic for a longer duration, hence becoming difficult to track and infer if a person has contracted the virus.

Governments across the globe have implemented strong measures to curb the spread of this virus. Some have stricter measures in terms of mobility, while others have larger testings. This varied approach makes it slightly difficult to predict the spread of the virus and its future damaging power.

**List of Figures**

|  |  |  |
| --- | --- | --- |
| Figure No. | Name | Page No. |
| 1 | Overview of project | 7 |
| 2 | Flow chart | 10 |
| 3 | DFD | 11 |

**Introduction**

Currently, the entire world is undergoing a pandemic of unprecedented standards. This pandemic is caused by a virus commonly known as “coronavirus”. This virus has touched almost every nation and has caused terror amongst many citizens. This virus has a spread rate of “2” under uncontrolled circumstances. That is if there are no restrictions for the people, then it will be that every person is likely to spread the virus to two others. The other major concern regarding this virus is that it remains asymptomatic for a longer duration, hence becoming difficult to track and infer if a person has contracted the virus.

1 Problem Statement:

Creating a model that predicts the number of Covid-19 cases country wise using previous recorded data with highest accuracy possible.

2 Project Overview:

2.1 Project description

This is a project where when someone accesses the model the user will be asked for their credentials, the location, timeline etc. Based on these details the predicted number of cases will be displayed to the person.

2.2 Functional Requirements

Once the model is accessed, it will ask for credentials. After verifying the users credentials the following steps will take place.

The user will then be prompted to enter the location and a timeline.

Based on the parameters of the model, it will give an output to the user, which will give an estimate of the number of cases that people might contract based on previous history.

The user can infer the given results and make decisions accordingly.

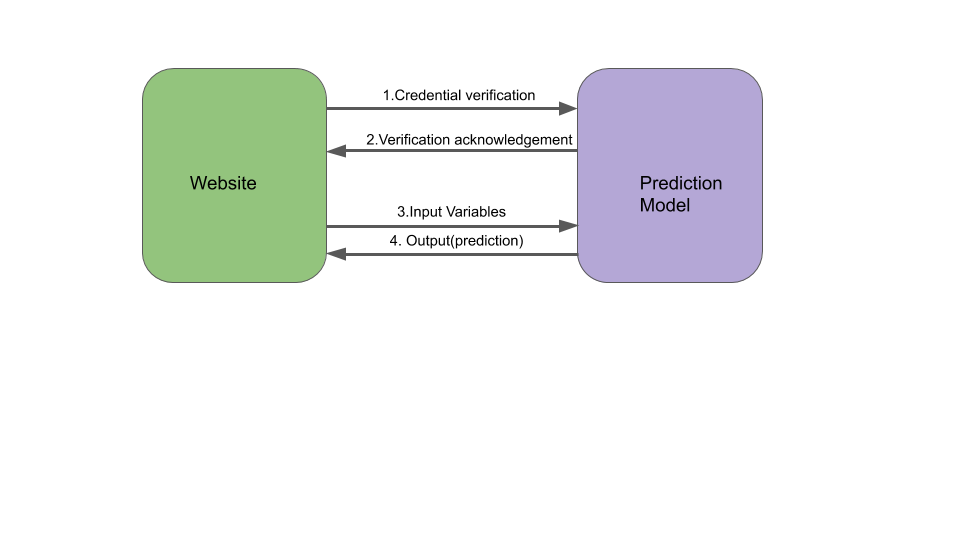
2.3 Technical Requirements

The crux of our application will be the efficiency of the model. This will depend upon the type of algorithm we deploy and the dataset. The dataset is sourced from official governmental websites and other open-source resources. Google Trends will be used to get data regarding the search history of the citizens. Google Mobility reports will also be used to understand the increase/decrease in movement of citizens.

2.3.1 Technology to be used

* TensorFlow
* Flask

2.3.2 Technical Architecture:



3.Non-functional requirements

3.1 Interface requirement

* Table to verify and validate the credentials of the user.

3.2 Security Requirements

* Secure Transfer of data from the user to the application
* Secure storage of the model and the user details

3.3 Volume requirements

Not applicable

**SYSTEM ANALYSIS AND DESIGN**

1.1 Requirement Specification

1. For Creating website

HTML, CSS, JAVASCRIPT,BOOTSTRAP:  
This are the requirnments for creating the website. Generally the specified languages are used to create the actual website.

1. HTML(Hypertext Markup Language) is used to create the frame of the website. Html is most important language while creating the website. Another important language is CSS.
2. CSS(Cascading Style Sheets) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web.
3. JavaScript is the scripting language which is used to make the website dynamic. JavaScript make pages interractive. Javascript is used to varify the user etc.JavaScript is high- level, often just-in-time compiled, and multi-paradigm language.
4. Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components. In this website we use bootstrap to make the website dynamic. Dynamic means it can open in both computer and mobile phone.
5. For creating Machine learning model
6. Python  
   Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Python community has developed many modules to help programmers implement machine learning. In this article, we will be using numpy, scipy and scikit-learn modules.

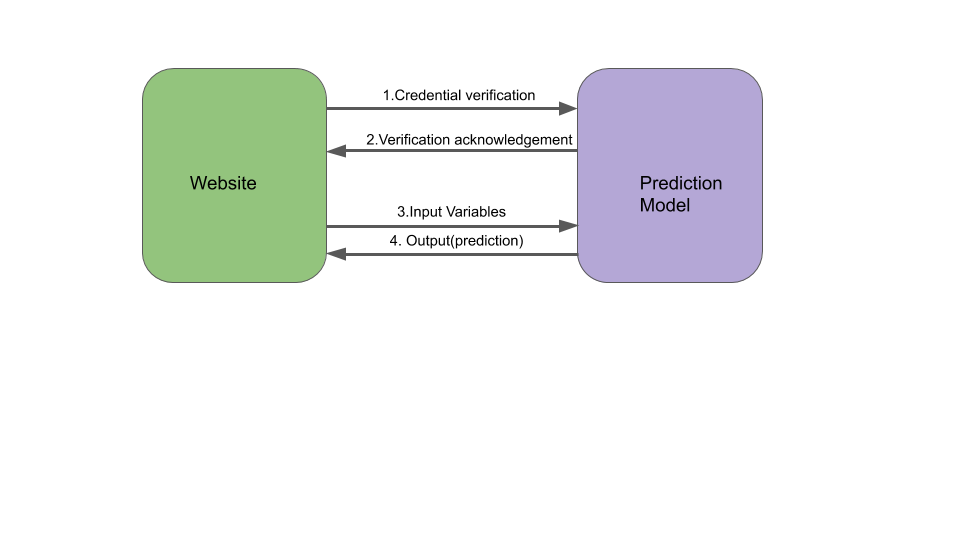
Libraries of python used in the machine learning model.

* Scikit-learn (also known as sklearn)  
  This library is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. We use this library to implement our machine learning algorithm. To use python for machine learning algorithms we have to include this library in our code.
* Pandas   
  Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. We use various methods of pandas library to create create and manipulate dataset eg. getcsv() method is used to create the dataset from the already existed csv file.
* Matplotlib  
  Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib. This library is mainly used for the data visualization. Like for creating Matrices and trees or graphs etc. We use this library in our code to visualize how our data is divided among the customers.
* SciPy  
  Scipy is a free and open-source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering. We use various methods of scipy libraries to do scientific calculations like calculating matrices and multiplying bigger float values that simple computer program cant do.

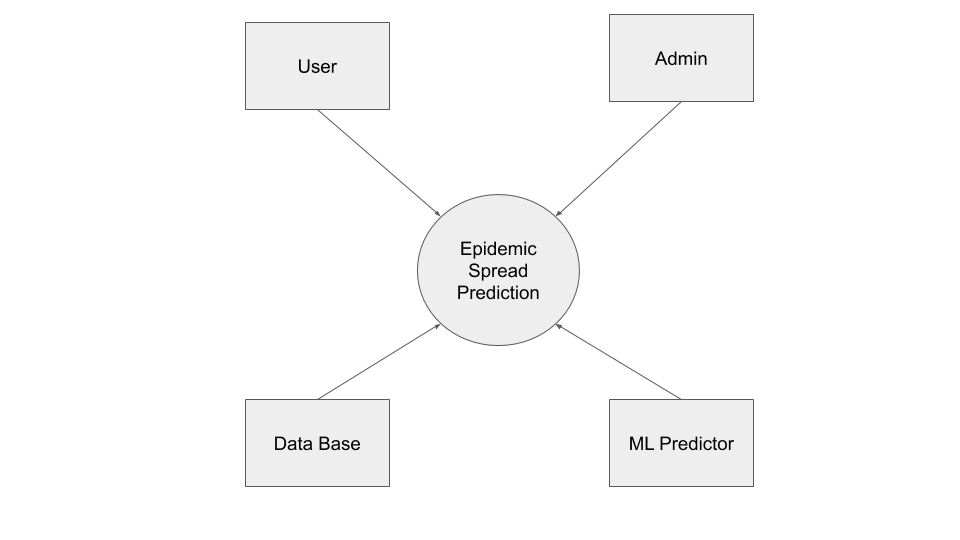
1. Softwares used to create the project
2. Tensorflow:  
   TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.
3. PyCharm:  
   PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language.

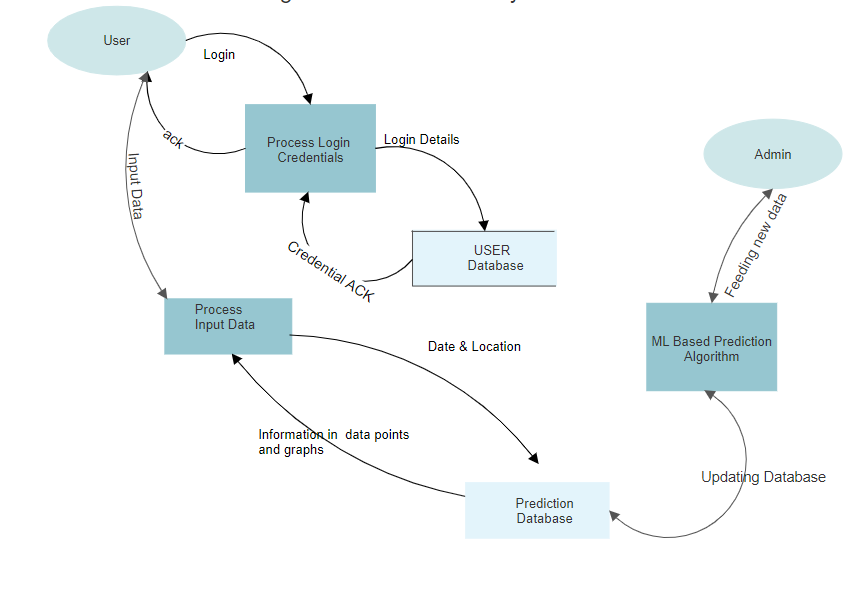
1.2 Flowcharts,DFD’s

Flowchart:



Data Flow Diagram(DFD)





1.3 Design and test steps

Steps to create design

1. The very first step is to create the website and create a connection to our database.
2. We collect data from the open source dataset
3. We clean the data set which values of states also mentioned.
4. Then we import various libraries from python like pandas, scikit-learn, numpy, scipy, matplotlib, and seaborn.
5. There are various methods defined in these libraries to do operation on the dataset and to create models.
6. We then format the given data into a time series based dataset, where we take the first seven days as input and then the value of the eight day as output.
7. Then we split the data into training and testing.
8. We then initialize the stacked LSTM model.
9. We feed in the values for the number of stacked layers, the loss function and activation function.
10. We deploy ReLu, and adam as the activation function and loss functions.
11. Other factors such as number of epochs, pivot points for dates are set and initialized.
12. Then the model is trained and the results are stored for the website to access later.
13. And then we send this json file to our web-interface file using function json.dumps()
14. This json file contains productID of the recommendations of the specific user.
15. Then as per the users needs the results can be accessed from the website

1.4 Algorithm and pseudo code

Algorithm:

Univariate LSTM:

Time series prediction problems are a difficult type of predictive modeling problem.Predicting viruses is a time series based problem.

Unlike regression predictive modeling, time series also adds the complexity of a sequence dependence among the input variables.

A powerful type of neural network designed to handle sequence dependence is called recurrent neural networks. The Long Short-Term Memory network or LSTM network is a type of recurrent neural network used in deep learning because very large architectures can be successfully trained.

**Code of the actual python program:**

import math

import pickle

import os

import pandas as pd

import folium

import numpy as np

import matplotlib

import json

matplotlib.use('nbagg')

import matplotlib.pylab as plt

import seaborn as sns

from matplotlib import rcParams

import plotly as py

import cufflinks

from plotly.subplots import make\_subplots

import plotly.express as px

import plotly.graph\_objects as go

from tqdm import tqdm\_notebook as tqdm

import warnings

import tensorflow as tf

from numpy import array

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import mean\_squared\_error

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping

from tensorflow.keras.layers import Dropout

from tensorflow.keras.layers import LSTM

from tensorflow.keras.utils import plot\_model

from tensorflow.keras import Input

from tensorflow.keras.layers import BatchNormalization

from dateutil.relativedelta import relativedelta

import datetime

warnings.filterwarnings("ignore")

# Reading COVID-19 Raw data

train = pd.read\_csv("DAT/train.csv")

# covid\_master=pd.read\_csv('covid\_19\_data.csv')

submission = pd.read\_csv("DATA/submission.csv")

# covid\_open=pd.read\_csv('COVID19\_open\_line\_list.csv')

test = pd.read\_csv("DATA/test.csv")

# train = pd.read\_csv("../input/covid19-global-forecasting-week-1/train.csv")

print(train.isna().sum())

# We will fill the missing states with a value 'NoState'

train = train.fillna('NoState')

test = test.fillna('NoState')

# changing the data type

train = train.rename(columns={'ConfirmedCases': 'Confirmed', 'Fatalities': 'Deaths', 'Country\_Region': 'Country/Region',

'Province\_State': 'Province/State', 'Date': 'ObservationDate'})

num\_cols = ['Confirmed', 'Deaths']

for col in num\_cols:

temp = [int(i) for i in train[col]]

train[col] = temp

print(train.head(2))

# Creating list of all regions of all countries

unique\_regions = train['Country/Region'].unique()

states\_per\_regions = []

for reg in tqdm(unique\_regions):

states\_per\_regions.append(train[train['Country/Region'] == reg]['Province/State'].unique())

print('No of unique regions:', len(unique\_regions))

# function to create training data for LSTM

# We will take last 7 days Cases as input and 8th day's case as output

def create\_train\_dataset(target, n\_steps, train, pivot\_date):

train = train.query("ObservationDate<" + pivot\_date)

x = []

y = []

for k in tqdm(range(len(unique\_regions))):

for state in states\_per\_regions[k]:

# print(unique\_regions[k],state)

temp = train[(train['Country/Region'] == unique\_regions[k]) & (train['Province/State'] == state)]

sequence = list(temp[target])

for i in range(len(sequence)):

end\_ix = i + n\_steps

if end\_ix > len(sequence) - 1:

break

seq\_x, seq\_y = sequence[i:end\_ix], sequence[end\_ix]

if (seq\_y != 0):

x.append(seq\_x)

y.append(seq\_y)

return array(x), array(y)

def create\_countrywise\_newly\_added\_train\_dataset(target, n\_steps, train, pivot\_date):

train = train.query("ObservationDate<" + pivot\_date)

x = []

y = []

for k in tqdm(range(len(unique\_regions))):

# print(unique\_regions[k],state)

temp = train[(train['Country/Region'] == unique\_regions[k])]

sequence = list(temp[target])

for i in range(len(sequence)):

end\_ix = i + n\_steps

if end\_ix > len(sequence) - 1:

break

seq\_x, seq\_y = sequence[i:end\_ix], sequence[end\_ix]

if (seq\_y != 0):

x.append(seq\_x)

y.append(seq\_y)

return array(x), array(y)

# function to create test dataset

# our supervised problem is now given last 7 days data predict the no of cases for 8th day

# target : 'Confirmed'/'Deaths'

def create\_test\_dataset(target, n\_steps, train, pivot\_date):

train = train.query("ObservationDate<" + pivot\_date)

x = []

regs = []

for k in tqdm(range(len(unique\_regions))):

for state in states\_per\_regions[k]:

# regs.append((unique\_regions[k],state))

temp = train[(train['Country/Region'] == unique\_regions[k]) & (train['Province/State'] == state)]

sequence = temp[target]

# print(sequence[len(sequence)-n\_steps:len(sequence)+1])

x.append(sequence[len(sequence) - n\_steps:len(sequence) + 1])

regs.append((unique\_regions[k], state))

return array(x), regs

def create\_countrywise\_newly\_added\_test\_dataset(target, n\_steps, train, pivot\_date):

train = train.query("ObservationDate<" + pivot\_date)

x = []

regs = []

for k in tqdm(range(len(unique\_regions))):

temp = train[(train['Country/Region'] == unique\_regions[k])]

sequence = temp[target]

# print(sequence[len(sequence)-n\_steps:len(sequence)+1])

x.append(sequence[len(sequence) - n\_steps:len(sequence) + 1])

regs.append(unique\_regions[k])

return array(x), regs

def get\_newly\_added(world\_data\_):

world\_data\_ = world\_data\_.sort\_values(['Country/Region', 'ObservationDate'])

temp = [0 \* i for i in range(len(world\_data\_))]

world\_data\_['New Confirmed'] = temp

world\_data\_['New Death'] = temp

for i in tqdm(range(1, len(world\_data\_))):

if (world\_data\_['Country/Region'].iloc[i] == world\_data\_['Country/Region'].iloc[i - 1]):

if (world\_data\_['Deaths'].iloc[i] < world\_data\_['Deaths'].iloc[i - 1]):

world\_data\_['Deaths'].iloc[i] = world\_data\_['Deaths'].iloc[i - 1]

if (world\_data\_['Confirmed'].iloc[i] < world\_data\_['Confirmed'].iloc[i - 1]):

world\_data\_['Confirmed'].iloc[i] = world\_data\_['Confirmed'].iloc[i - 1]

world\_data\_['New Confirmed'].iloc[i] = world\_data\_['Confirmed'].iloc[i] - world\_data\_['Confirmed'].iloc[

i - 1]

world\_data\_['New Death'].iloc[i] = world\_data\_['Deaths'].iloc[i] - world\_data\_['Deaths'].iloc[i - 1]

else:

world\_data\_['New Confirmed'].iloc[i] = world\_data\_['Confirmed'].iloc[i]

world\_data\_['New Death'].iloc[i] = world\_data\_['Deaths'].iloc[i]

return world\_data\_

# Countrywise timeseries data with Newly added Incident Each Day

covid\_timeseries = train.groupby(['ObservationDate', 'Country/Region', 'Province/State'])['Confirmed', 'Deaths'].sum()

covid\_timeseries = covid\_timeseries.reset\_index().sort\_values('ObservationDate')

covid\_timeseries = get\_newly\_added(covid\_timeseries)

print(covid\_timeseries[covid\_timeseries['Country/Region'] == 'India'].tail())

# Maintain the date format for pivot\_date and forcast\_start\_date

# Pivot\_date : data of date less than the given date will be used for training

# Forcast\_start\_date : Date from which forcasting will be started

n\_steps = 7

pivot\_date = "'2020-04-02'"

forcast\_start\_date = '2020-04-02'

print('Preparing datasets with Cumulative Confirmed Incidents..')

X\_c, y\_c = create\_train\_dataset('Confirmed', n\_steps, train, pivot\_date)

print('Preparing datasets with Newly Confirmed Incidents..')

X\_nc, y\_nc = create\_train\_dataset('New Confirmed', n\_steps, covid\_timeseries, pivot\_date)

test\_confirmed, regs = create\_test\_dataset('Confirmed', n\_steps, train, pivot\_date)

test\_nc, reg\_nc = create\_test\_dataset('New Confirmed', n\_steps, covid\_timeseries, pivot\_date)

print('Preparing datasets with Deaths Incidents..')

X\_d, y\_d = create\_train\_dataset('Deaths', n\_steps, train, pivot\_date)

test\_deaths, regs = create\_test\_dataset('Deaths', n\_steps, train, pivot\_date)

print('Datasets prepared sucessfully.')

# Split the train data in to train and val data

X\_train\_c, X\_val\_c, y\_train\_c, y\_val\_c = train\_test\_split(X\_c, y\_c, test\_size=0.30, random\_state=42)

X\_train\_d, X\_val\_d, y\_train\_d, y\_val\_d = train\_test\_split(X\_d, y\_d, test\_size=0.30, random\_state=42)

X\_train\_nc, X\_val\_nc, y\_train\_nc, y\_val\_nc = train\_test\_split(X\_c, y\_c, test\_size=0.30, random\_state=42)

# Reshapping the Confirmed data for LSTM

X\_train\_c = X\_train\_c.reshape((X\_train\_c.shape[0], 1, X\_train\_c.shape[1]))

X\_val\_c = X\_val\_c.reshape((X\_val\_c.shape[0], 1, X\_val\_c.shape[1]))

X\_train\_nc = X\_train\_nc.reshape((X\_train\_nc.shape[0], 1, X\_train\_nc.shape[1]))

X\_val\_nc = X\_val\_nc.reshape((X\_val\_nc.shape[0], 1, X\_val\_nc.shape[1]))

X\_test\_c = test\_confirmed.reshape((test\_confirmed.shape[0], 1, test\_confirmed.shape[1]))

X\_test\_nc = test\_nc.reshape((test\_nc.shape[0], 1, test\_nc.shape[1]))

print(X\_train\_c.shape, y\_train\_c.shape, X\_val\_c.shape, y\_val\_c.shape, X\_test\_c.shape, X\_test\_nc.shape)

# Reshapping the d\_confirmed data for LSTM

X\_train\_d = X\_train\_d.reshape((X\_train\_d.shape[0], 1, X\_train\_d.shape[1]))

X\_val\_d = X\_val\_d.reshape((X\_val\_d.shape[0], 1, X\_val\_d.shape[1]))

X\_test\_d = test\_deaths.reshape((test\_deaths.shape[0], 1, test\_deaths.shape[1]))

print(X\_train\_d.shape, y\_train\_d.shape, X\_val\_d.shape, y\_val\_d.shape, X\_test\_d.shape)

print(X\_train\_c[100])

print(X\_train\_d[1])

# Initializing model components

epochs = 10

batch\_size = 32

n\_hidden = 32

timesteps = X\_train\_c.shape[1]

input\_dim = X\_train\_c.shape[2]

n\_features = 1

print(timesteps)

print(input\_dim)

print(len(X\_train\_c))

# Stacked LSTM Model

model\_c = Sequential()

model\_c.add(LSTM(50, activation='relu', input\_shape=(n\_features, n\_steps), return\_sequences=True))

model\_c.add(LSTM(150, activation='relu'))

model\_c.add(Dense(1, activation='relu'))

model\_c.summary()

# Compiling the model

model\_c.compile(optimizer='adam', loss=tf.keras.losses.MeanSquaredLogarithmicError())

#To reduce overfitting

callbacks = [ReduceLROnPlateau(monitor='val\_loss', patience=5, verbose=1, factor=0.6),

EarlyStopping(monitor='val\_loss', patience=20),

ModelCheckpoint(filepath='best\_model.h5', monitor='val\_loss', save\_best\_only=True)]

# fit the model

hist = model\_c.fit(X\_train\_c, y\_train\_c, epochs=epochs, batch\_size=batch\_size, validation\_data=(X\_val\_c, y\_val\_c),

verbose=2,

shuffle=True, callbacks=callbacks)

# Stacked LSTM Model

model\_d = Sequential()

model\_d.add(LSTM(50, activation='relu', input\_shape=(n\_features,n\_steps),return\_sequences=True))

model\_d.add(LSTM(50, activation='relu'))

model\_d.add(Dense(1))

model\_d.summary()

# Compiling the model

model\_d.compile(optimizer='adam', loss=tf.keras.losses.MeanSquaredLogarithmicError())

callbacks = [ReduceLROnPlateau(monitor='val\_loss', patience=5, verbose=1, factor=0.6),

EarlyStopping(monitor='val\_loss', patience=20),

ModelCheckpoint(filepath='best\_model.h5', monitor='val\_loss', save\_best\_only=True)]

# fit the model

hist=model\_d.fit(X\_train\_d,y\_train\_d, epochs=epochs, batch\_size=batch\_size, validation\_data=(X\_val\_d, y\_val\_d), verbose=2,

shuffle=True,callbacks=callbacks)

# Stacked LSTM Model

model\_nc = Sequential()

model\_nc.add(LSTM(50, activation='relu', input\_shape=(n\_features,n\_steps),return\_sequences=True))

model\_nc.add(LSTM(50, activation='relu'))

model\_nc.add(Dense(1))

model\_nc.summary()

model\_nc.compile(optimizer='adam', loss=tf.keras.losses.MeanSquaredLogarithmicError())

callbacks = [ReduceLROnPlateau(monitor='val\_loss', patience=5, verbose=1, factor=0.6),

EarlyStopping(monitor='val\_loss', patience=20),

ModelCheckpoint(filepath='best\_model.h5', monitor='val\_loss', save\_best\_only=True)]

# fit the model

hist=model\_nc.fit(X\_train\_nc,y\_train\_nc, epochs=epochs, batch\_size=batch\_size, validation\_data=(X\_val\_nc, y\_val\_nc), verbose=2,

shuffle=True,callbacks=callbacks)

# batch size number of inputs per gradient

def pred(model,data):

y\_pred=model.predict(data)

#y\_pred=[math.ceil(i) for i in y\_pred]

return y\_pred

# Utility method for Forcasting

# model - trained model on Confirmed/Deaths data

# start\_date - Starting date of forcasting

# num\_days - Number of days for which forcasting is required

def forcast(model,data,start\_date,num\_days):

res\_=dict()

for i in range(len(data)):

res\_[i]=[]

y\_pred=pred(model,data)

dates=[]

date1 = datetime.datetime.strptime(start\_date, "%Y-%m-%d")

for j in range(1,num\_days+1):

for i in range(len(data)):

cur\_window=list(data[i][0][1:n\_steps+1])

#print(j,i,cur\_window[-1])

res\_[i].append(cur\_window[-1])

cur\_window.append(y\_pred[i])

data[i][0]=cur\_window

y\_pred=pred(model,data)

dates.append(date1.strftime("%Y-%m-%d"))

date1+=relativedelta(days=1)

res=pd.DataFrame(pd.DataFrame(pd.DataFrame(res\_).values.T))

res.columns=dates

res['Country/State']=regs

return res

def forcast\_(model,data,start\_date,num\_days):

res\_=[]

for i in list(data['Country/Region']):

res\_.append(i)

y\_pred=pred(model,data)

dates=[]

date1 = datetime.datetime.strptime(start\_date, "%Y-%m-%d")

for j in range(1,num\_days+1):

for i in range(len(data)):

cur\_window=list(data[i][0][1:n\_steps+1])

#print(j,i,cur\_window[-1])

res\_[i].append(cur\_window[-1])

cur\_window.append(y\_pred[i])

data[i][0]=cur\_window

y\_pred=pred(model,data)

dates.append(date1.strftime("%Y-%m-%d"))

date1+=relativedelta(days=1)

res=pd.DataFrame(pd.DataFrame(pd.DataFrame(res\_).values.T))

res.columns=dates

res['Country/State']=res\_

return res

# Utility method for submission

def prepare\_submission(res\_c,res\_d,res\_nc,test,pivot\_date):

test=test.query("Date>="+pivot\_date)

index=dict()

for i in range(len(res\_c)):

index[res\_c.iloc[i]['Country/State']]=i

pred\_c=[]

pred\_d=[]

pred\_nc=[]

for i in tqdm(range(len(test))):

if((test.iloc[i]['Country\_Region'],test.iloc[i]['Province\_State']) in index):

loc=index[(test.iloc[i]['Country\_Region'],test.iloc[i]['Province\_State'])]

#print(res.iloc[loc][test.iloc[i]['Date']])

pred\_c.append(res\_c.iloc[loc][test.iloc[i]['Date']])

pred\_d.append(res\_d.iloc[loc][test.iloc[i]['Date']])

pred\_nc.append(res\_nc.iloc[loc][test.iloc[i]['Date']])

test['ConfirmedCases']=pred\_c

test['Fatalities']=pred\_d

test['New Confirmed']=pred\_nc

res\_regional=test

res=test.drop(columns=['Province\_State','Country\_Region','Date','New Confirmed'])

return res,res\_regional

# Call only when forcast and submission data are available

def get\_countrywise\_forcast\_(target,country\_name,state\_name,num\_days):

temp=covid\_timeseries[(covid\_timeseries['Country/Region']==country\_name)&(covid\_timeseries['Province/State']==state\_name)].query("ObservationDate>="+pivot\_date)

x\_truth=temp.ObservationDate

y\_truth=temp[target]

pred\_=res\_regional[(res\_regional['Country\_Region']==country\_name) & ((res\_regional['Province\_State']==state\_name))]

x\_pred=pred\_.Date[0:num\_days]

y\_pred=pred\_[target][0:num\_days]

return list(x\_truth),list(y\_truth),list(x\_pred),list(y\_pred)

# Call only when forcast and submission data are available

def get\_countrywise\_forcast(country\_name,state\_name,num\_days):

temp=train[(train['Country/Region']==country\_name)&(train['Province/State']==state\_name)].query("ObservationDate>="+pivot\_date)

x\_truth=temp.ObservationDate

y\_truth=temp.Confirmed

pred\_=res\_regional[(res\_regional['Country\_Region']==country\_name) & ((res\_regional['Province\_State']==state\_name))]

x\_pred=pred\_.Date[0:num\_days]

y\_pred=pred\_.ConfirmedCases[0:num\_days]

return list(x\_truth),list(y\_truth),list(x\_pred),list(y\_pred)

# Call only when forcast and submission data are available

def get\_countrywise\_forcast\_Deaths(country\_name,state\_name,num\_days):

temp=train[(train['Country/Region']==country\_name)&(train['Province/State']==state\_name)].query("ObservationDate>="+pivot\_date)

x\_truth=temp.ObservationDate

y\_truth=temp.Deaths

pred\_=res\_regional[(res\_regional['Country\_Region']==country\_name) & ((res\_regional['Province\_State']==state\_name))]

x\_pred=pred\_.Date[0:num\_days]

y\_pred=pred\_.Fatalities[0:num\_days]

return list(x\_truth),list(y\_truth),list(x\_pred),list(y\_pred)

# Utility Method to convert newly added prediction to cumulative [Not Accurate]

def get\_cumulative\_confirmed\_cases(world\_data\_):

world\_data\_=world\_data\_.sort\_values(['Country\_Region','Date'])

temp=[0\*i for i in range(len(world\_data\_))]

world\_data\_['Cumulative Confirmed']=world\_data\_['New Confirmed']

for i in tqdm(range(1,len(world\_data\_))):

if(world\_data\_['Country\_Region'].iloc[i]!=world\_data\_['Country\_Region'].iloc[i-1]):

world\_data\_['Cumulative Confirmed'].iloc[i]=world\_data\_['ConfirmedCases'].iloc[i]

for i in tqdm(range(1,len(world\_data\_))):

if(world\_data\_['Country\_Region'].iloc[i]==world\_data\_['Country\_Region'].iloc[i-1]):

world\_data\_['Cumulative Confirmed'].iloc[i]=world\_data\_['Cumulative Confirmed'].iloc[i]+world\_data\_['Cumulative Confirmed'].iloc[i-1]

return world\_data\_

# num\_days = Num of days for which Forcasting is required

#forcast\_start\_date='2020-04-01'

res\_confirmed = forcast(model\_c,X\_test\_c,forcast\_start\_date,num\_days=50)

res\_deaths = forcast(model\_d,X\_test\_d,forcast\_start\_date,num\_days=50)

res\_new\_confirmed = forcast(model\_nc,X\_test\_nc,forcast\_start\_date,num\_days=50)

# res\_regional contains submission data along with extra columns

sub,res\_regional=prepare\_submission(res\_confirmed,res\_deaths,res\_new\_confirmed,test,pivot\_date)

sub.to\_csv('submission.csv',index=None)

sub.head()

x\_truth\_Ge,y\_truth\_Ge,x\_pred\_Ge,y\_pred\_Ge=get\_countrywise\_forcast\_('New Confirmed','Germany','NoState',15)

x\_truth\_In,y\_truth\_In,x\_pred\_In,y\_pred\_In=get\_countrywise\_forcast\_('New Confirmed','India','NoState',15)

x\_truth\_Sp,y\_truth\_Sp,x\_pred\_Sp,y\_pred\_Sp=get\_countrywise\_forcast\_('New Confirmed','Spain','NoState',15)

x\_truth\_It,y\_truth\_It,x\_pred\_It,y\_pred\_It=get\_countrywise\_forcast\_('New Confirmed','Italy','NoState',15)

x\_truth\_Ge,y\_truth\_Ge,x\_pred\_Ge,y\_pred\_Ge = get\_countrywise\_forcast('Germany','NoState',15)

x\_truth\_In,y\_truth\_In,x\_pred\_In,y\_pred\_In = get\_countrywise\_forcast('India','NoState',15)

x\_truth\_Sp,y\_truth\_Sp,x\_pred\_Sp,y\_pred\_Sp = get\_countrywise\_forcast('Spain','NoState',15)

x\_truth\_It,y\_truth\_It,x\_pred\_It,y\_pred\_It = get\_countrywise\_forcast('Italy','NoState',15)

res\_In\_truth = {x\_truth\_In[i]: y\_truth\_In[i] for i in range(len(x\_truth\_In))}

res\_In\_pred = {x\_pred\_In[i]: y\_pred\_In[i] for i in range(len(x\_pred\_In))}

res\_Ge\_truth = {x\_truth\_Ge[i]: y\_truth\_Ge[i] for i in range(len(x\_truth\_Ge))}

res\_Ge\_pred = {x\_pred\_Ge[i]: y\_pred\_Ge[i] for i in range(len(x\_pred\_Ge))}

res\_Sp\_truth = {x\_truth\_Sp[i]: y\_truth\_Sp[i] for i in range(len(x\_truth\_Sp))}

res\_Sp\_pred = {x\_pred\_Sp[i]: y\_pred\_Sp[i] for i in range(len(x\_pred\_Sp))}

res\_It\_truth = {x\_truth\_It[i]: y\_truth\_It[i] for i in range(len(x\_truth\_It))}

res\_It\_pred = {x\_pred\_It[i]: y\_pred\_It[i] for i in range(len(x\_pred\_It))}

json.dump(res\_In\_truth, open("WEB\_INTERFACE\In\_truth.txt",'w'))

json.dump(res\_In\_pred, open("WEB\_INTERFACE\In\_pred.txt",'w'))

json.dump(res\_Ge\_truth, open("WEB\_INTERFACE\Ge\_truth.txt",'w'))

json.dump(res\_Ge\_pred, open("WEB\_INTERFACE\Ge\_pred.txt",'w'))

json.dump(res\_It\_truth, open("WEB\_INTERFACE\It\_truth.txt",'w'))

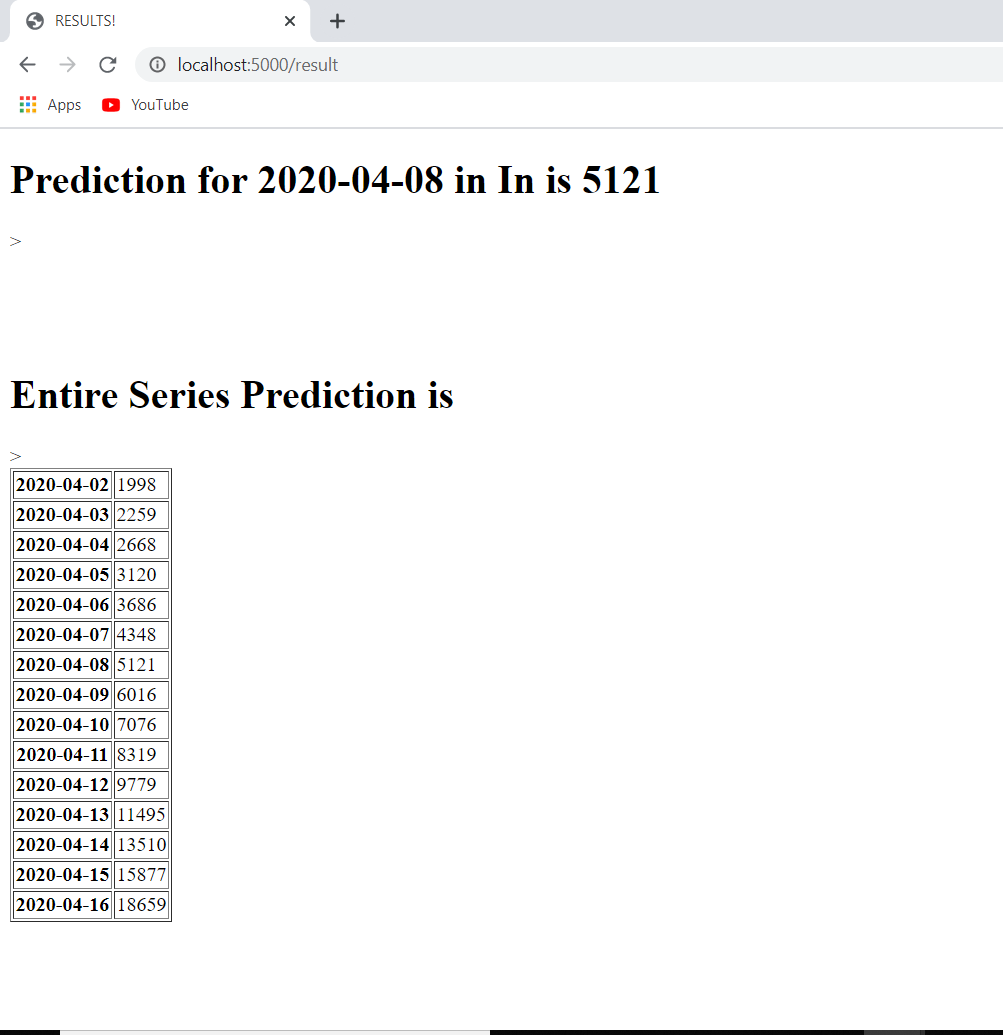
json.dump(res\_It\_pred, open("WEB\_INTERFACE\It\_pred.txt",'w'))

json.dump(res\_Sp\_truth, open("WEB\_INTERFACE\Sp\_truth.txt",'w'))

json.dump(res\_Sp\_pred, open("WEB\_INTERFACE\Sp\_pred.txt",'w'))

**Output of the program:**

***The website***

****

**Conclusion:**

Prediction systems are of great use for predicting future number of cases in a time dependent series.

* The effectiveness of the system relies on the algorithm it uses to find future patterns.
* A univariate stacked LSTM is used to predict the values.
* In the beginning, the data is cleaned and prepared for training.
* In the most crucial part, a comprehensive amount of study is done about overall system design and the prediction approach.
* Finally, we come to the conclusion that a stacked LSTM gives better results than a normal LSTM.

**References**

* https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
* https://www.kaggle.com/arpandas65/covid-19-projection-using-lstm