**Objective**

To take a dataset having daily statewise distribution of confirmed covid cases and related details like deceased and recovered numbers and predict the curves.

**Procedure**

The machine learning model used here is a neural network model with 6 layers, each layer decreasing in node number by half. The preprocessing involves splitting the dataset based on state entered and splitting the resulting dataset into half, the first half serving as the training dataset and the second half serving as the testing dataset.

**Program**

from pandas import read\_csv

from keras.models import Sequential

from keras.layers import Dense

from keras.wrappers.scikit\_learn import KerasRegressor

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import KFold

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import Pipeline

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

dataframe = read\_csv("/content/sample\_data/state\_level\_daily.csv", delim\_whitespace=False, header=0)

SS=input()

dataframe=dataframe.loc[dataframe['State'] == SS]

dataframe=dataframe.sort\_values(by='Confirmed')

dataset = dataframe.values

#dataframe

A=(int)(len(dataframe)/2)

X\_train = dataset[0:A,3]

y\_train = dataset[0:A,5]

X\_test=dataset[A:,3]

y\_test=dataset[A:,5]

X\_train=X\_train.reshape(-1,1)

X\_test=X\_test.reshape(-1,1)

X\_train = tf.convert\_to\_tensor(X\_train, dtype=tf.int64)

y\_train = tf.convert\_to\_tensor(y\_train, dtype=tf.int64)

X\_test = tf.convert\_to\_tensor(X\_test, dtype=tf.int64)

y\_test = tf.convert\_to\_tensor(y\_test, dtype=tf.int64)

#plt.scatter(X\_test,y\_test)     #Confirmed vs Deceased

#plt.scatter(dataset[A:,1],X\_test)    #Confirmed vs time

#plt.scatter(dataset[A:,1],y\_test)    #Deceased vs time

model = Sequential()

model.add(Dense(60, input\_dim=1, kernel\_initializer='normal', activation='relu'))

model.add(Dense(30, kernel\_initializer='normal', activation='relu'))

model.add(Dense(15, kernel\_initializer='normal', activation='relu'))

model.add(Dense(7, kernel\_initializer='normal', activation='relu'))

model.add(Dense(3, kernel\_initializer='normal', activation='relu'))

model.add(Dense(1, kernel\_initializer='normal', activation='relu'))

model.compile(loss='mean\_squared\_error', optimizer='adam')

model.fit(X\_train, y\_train,  epochs=500, batch\_size=300)

y\_test=model.predict(X\_test)

y\_test=y\_test.mean(axis=1)

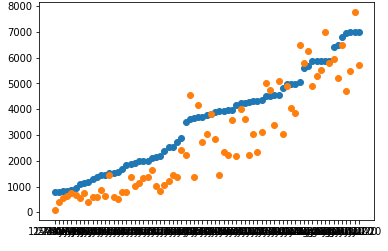
#plt.scatter(X\_test,y\_test)     #Confirmed vs Deceased

#plt.scatter(dataset[A:,1],X\_test)    #Confirmed vs time

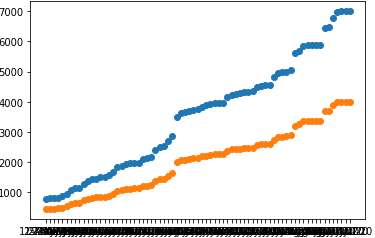
#plt.scatter(dataset[A:,1],y\_test)    #Deceased vs time

**Output graphs**

**Before Training**



**After Training**



**Link**

<https://colab.research.google.com/drive/1WL9R_NWKfEJ9e6758KFkWiSVxBAZvMLz?usp=sharing>