Aloysius Hasheem A Sendad

CSCC 45 – A

LSTM: Predicting Covid Cases

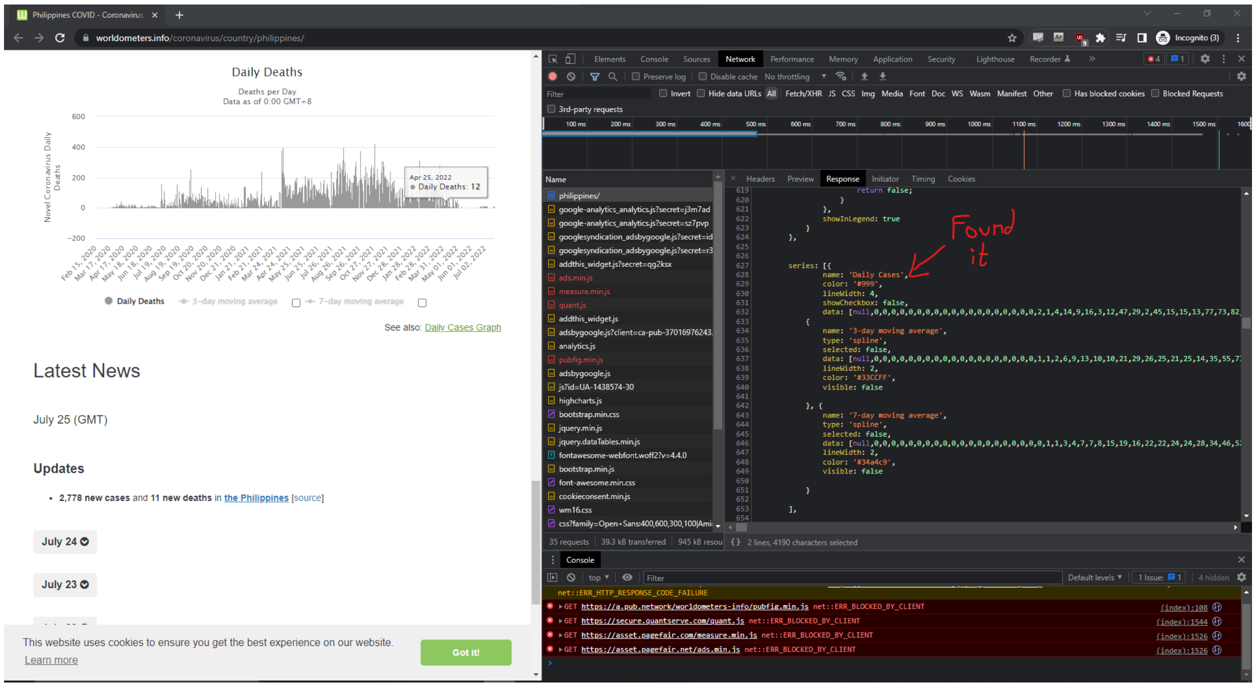
DOCUMENTATION:

Introduction

Covid 19 has already almost reached 3 years, and it has been a pain to deal with. Since more and more people are getting used to it already, and places are slowly trying to go back to normal with opening more public spaces such as f2f schooling, it would be good to know, or at least predict the cases of covid-19 in the future. Thus the idea of this project is to use machine learning to predict future cases of covid.

Method

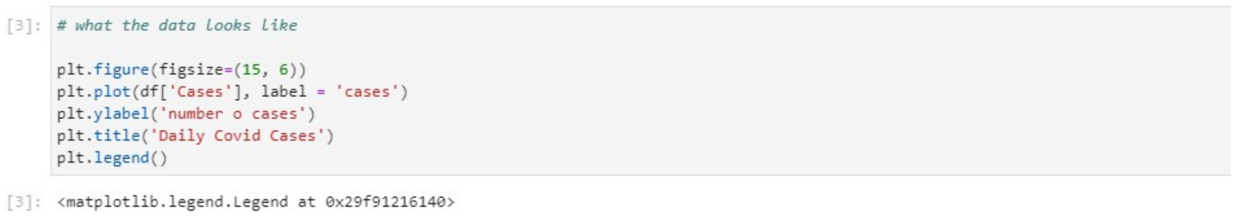
To start, we will be needing a dataset. Originally, my plan was to create a web scraping application using Beautiful Soup, but most websites present their data in the form of interactive graphs. Thus, to get the data, I manually scraped it off the website by copy-pasting it from the 'inspect element' in google chrome.



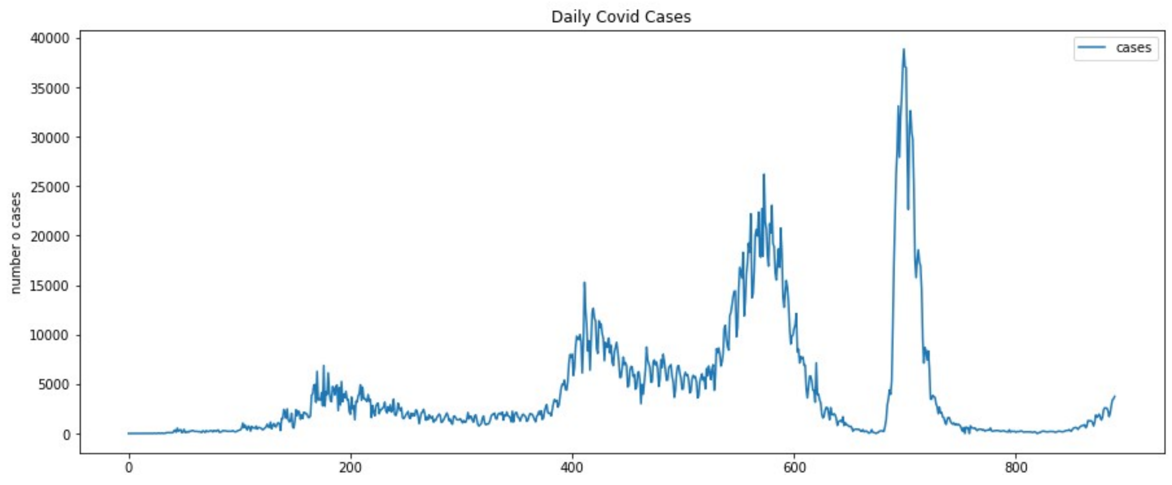
```
                    return false;
                }
            },
            showInLegend: true
        }
    },


    series: [{
        name: 'Daily Cases',
        color: '#999',
        lineWidth: 4,
        showCheckbox: false,
        data: [null,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,1,4,14,9,16,3,12,47,29,2,45,15,15,13,77,73,82,
    {
        name: '3-day moving average',
        type: 'spline',
        selected: false,
        data: [null,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,2,6,9,13,10,10,21,29,26,25,21,25,14,35,55,7
        lineWidth: 2,
        color: '#33CCFF',
        visible: false

    }, {
        name: '7-day moving average',
        type: 'spline',
        selected: false,
        data: [null,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,3,4,7,7,8,15,19,16,22,22,24,24,28,34,46,5
        lineWidth: 2,
        color: '#34a4c9',
        visible: false

    }

],
```

```
net::ERR_HTTP_RESPONSE_CODE_FAILURE
GET https://a.pub.network/worldometers-info/pubfig.min.js net::ERR_BLOCKED_BY_CLIENT    (index):108
GET https://secure.quantserve.com/quant.js net::ERR_BLOCKED_BY_CLIENT    (index):1544
GET https://asset.pagefair.com/measure.min.js net::ERR_BLOCKED_BY_CLIENT    (index):1526
GET https://asset.pagefair.net/ads.min.js net::ERR_BLOCKED_BY_CLIENT    (index):1526
```

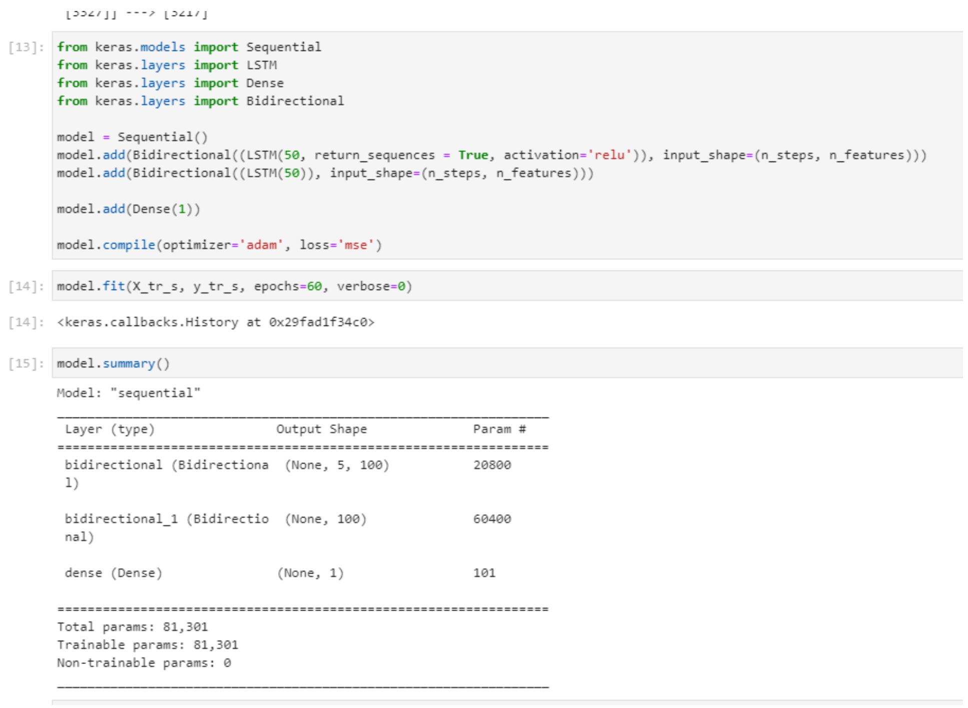
Here is how the model looks like in graph:



```
[3]: # what the data looks like

     plt.figure(figsize=(15, 6))
     plt.plot(df['Cases'], label = 'cases')
     plt.ylabel('number o cases')
     plt.title('Daily Covid Cases')
     plt.legend()

[3]: <matplotlib.legend.Legend at 0x29f91216140>
```



To create the model, I will be using LSTM to create and test the model from the training, test, and validation data.

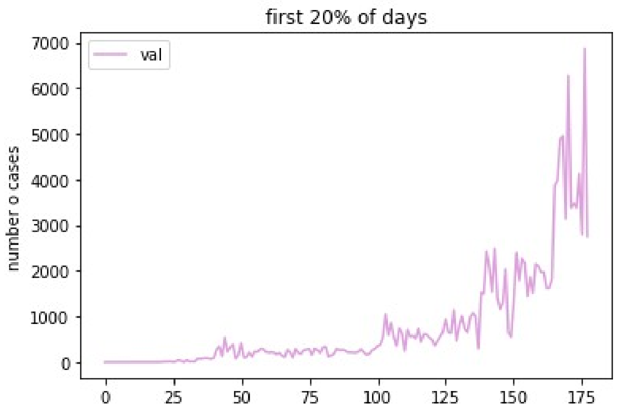


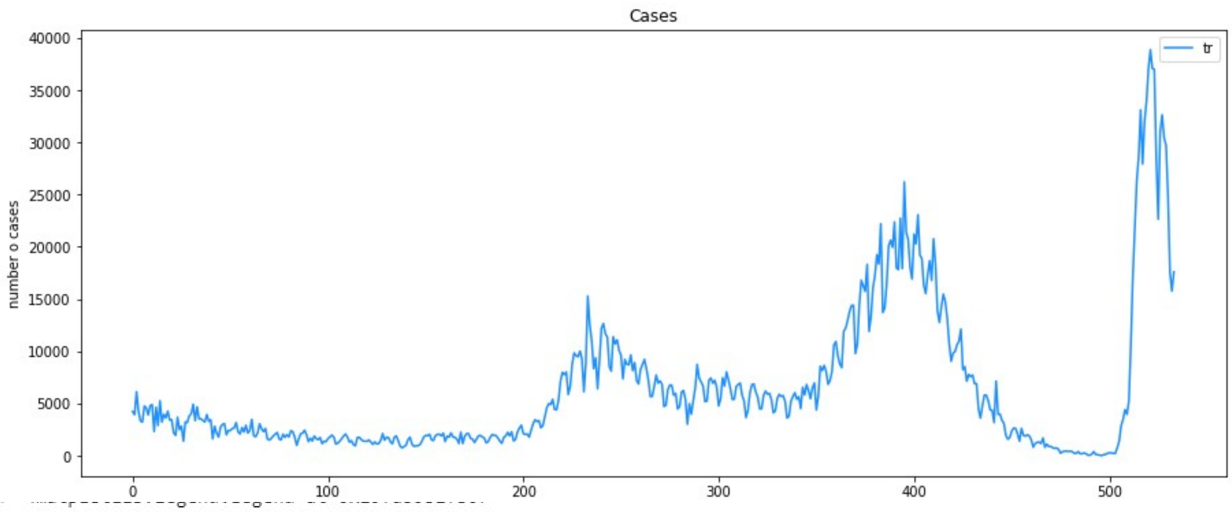
```
[5327]] ---> [5217]

[13]: from keras.models import Sequential
      from keras.layers import LSTM
      from keras.layers import Dense
      from keras.layers import Bidirectional

      model = Sequential()
      model.add(Bidirectional((LSTM(50, return_sequences = True, activation='relu')), input_shape=(n_steps, n_features)))
      model.add(Bidirectional((LSTM(50)), input_shape=(n_steps, n_features)))

      model.add(Dense(1))

      model.compile(optimizer='adam', loss='mse')

[14]: model.fit(X_tr_s, y_tr_s, epochs=60, verbose=0)

[14]: <keras.callbacks.History at 0x29fad1f34c0>

[15]: model.summary()

      Model: "sequential"
      _________________________________________________________________
       Layer (type)                Output Shape              Param #
      =================================================================
       bidirectional (Bidirectiona  (None, 5, 100)           20800
       l)

       bidirectional_1 (Bidirectio  (None, 100)              60400
       nal)

       dense (Dense)               (None, 1)                 101

      =================================================================
      Total params: 81,301
      Trainable params: 81,301
      Non-trainable params: 0
      _________________________________________________________________
```

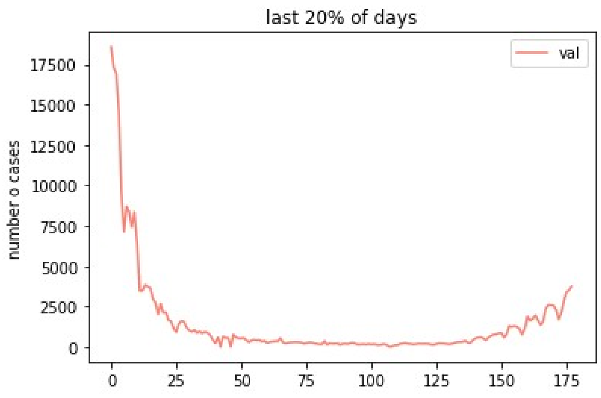
The data is split by:

- 60% - training
- 20% - testing
- 20% - validation

The data for the validation and testing set are from the first and last 20% of the data respectively.





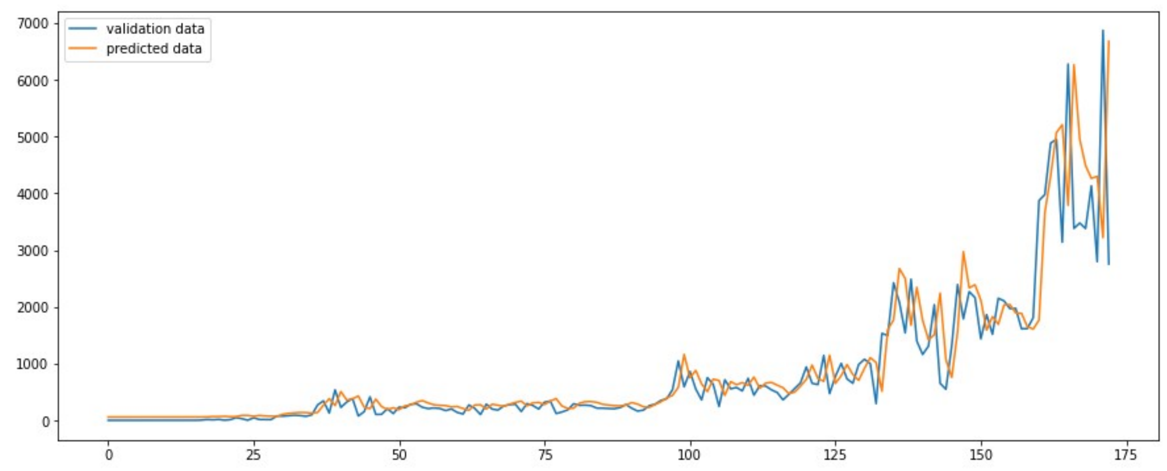


Results

After setting up the model, and tuning with the parameters, here is what the predictions look like in the Validation and Testing sets.

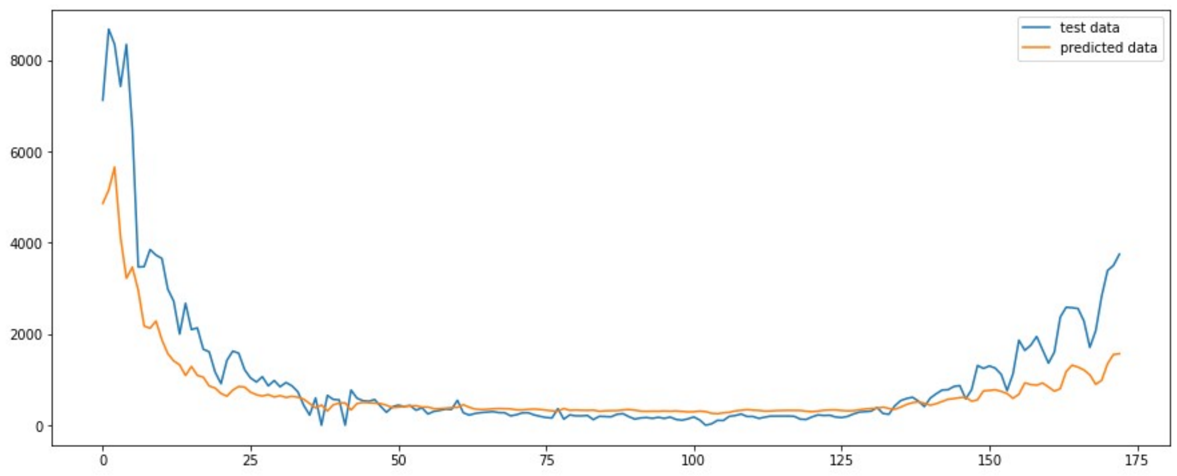
```
[18]: fig = plt.figure(figsize=(15,6))
      plt.plot(y_va, label = 'validation data')
      plt.plot(pred_new, label = 'predicted data')
      plt.legend()
```

```
[18]: <matplotlib.legend.Legend at 0x24c7ba1b700>
```



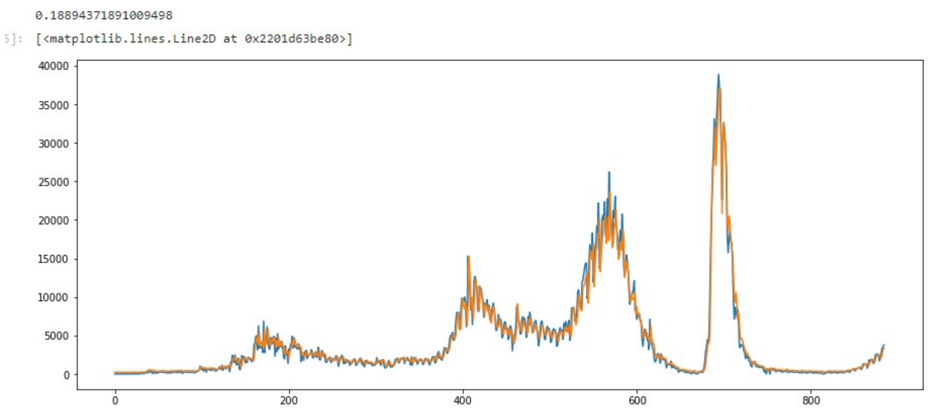
```
[20]: fig = plt.figure(figsize=(15,6))
      plt.plot(y_te, label = 'test data')
      plt.plot(pred_new, label = 'predicted data')
      plt.legend()
```

```
[20]: <matplotlib.legend.Legend at 0x24c7bfcb040>
```



Now that I have defined my model, I will use the same model to predict future cases. To do that, I will use the entire dataset as the training dataset.

Here is the dataset and predicted set:



```
    0.18894371891009498
5]: [<matplotlib.lines.Line2D at 0x2201d63be80>]
```

I think I would’ve gotten better results with higher epoch, but I settled with this.

To predict future sets, I used this model, together with the last n numbers in the dataset to predict the next value. After, I simply used recursion to predict the next 100 days

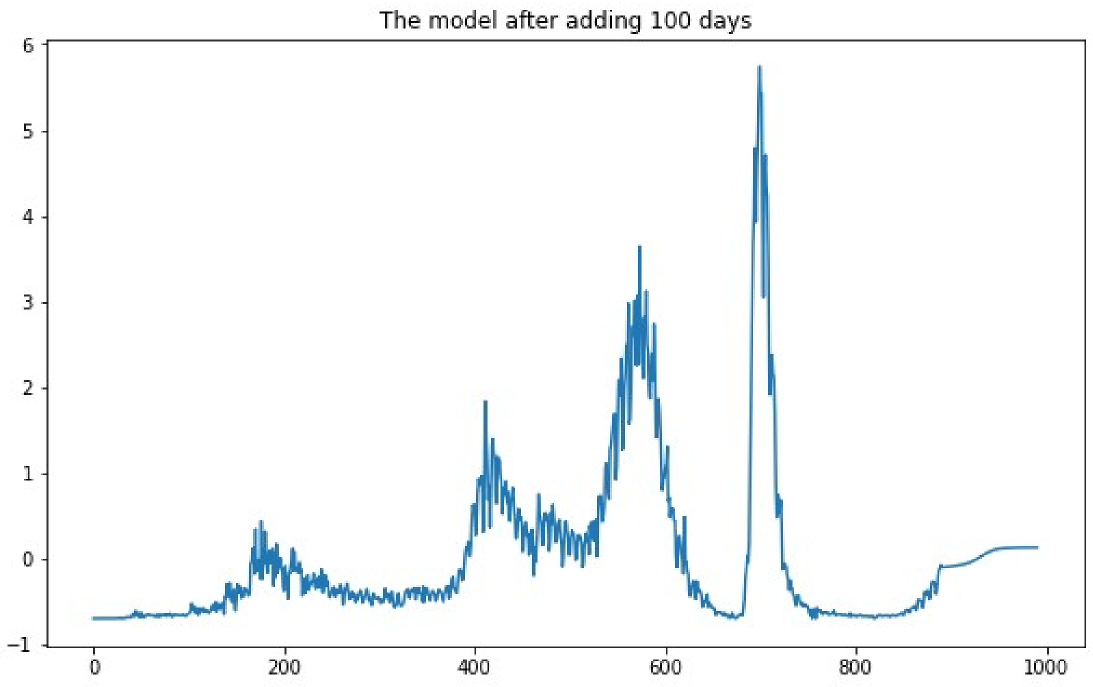
```
[27]: Dta = np.array([])
      Dta = np.append(Dta, scaled_new_data)

      print(Dta.shape)

      # @param
      #   rec = number of times recursion occurs

      def pred_(model, data, n_step, rec):
          global Dta
          # get the last set in the data
          l = data.shape[0]
          X = data[l-n_step:l]

          y = forecast_lstm(model, 1, X)

          #print(X, y)
          data = np.append(data, y)

          if rec <= 0:
              return data
          else:
              rec = rec-1
              data = pred_(model, data, n_step, rec)

          return data

      (890,)
```

```
[28]: a = pred_(model_1, Dta, n_steps, 100)
```

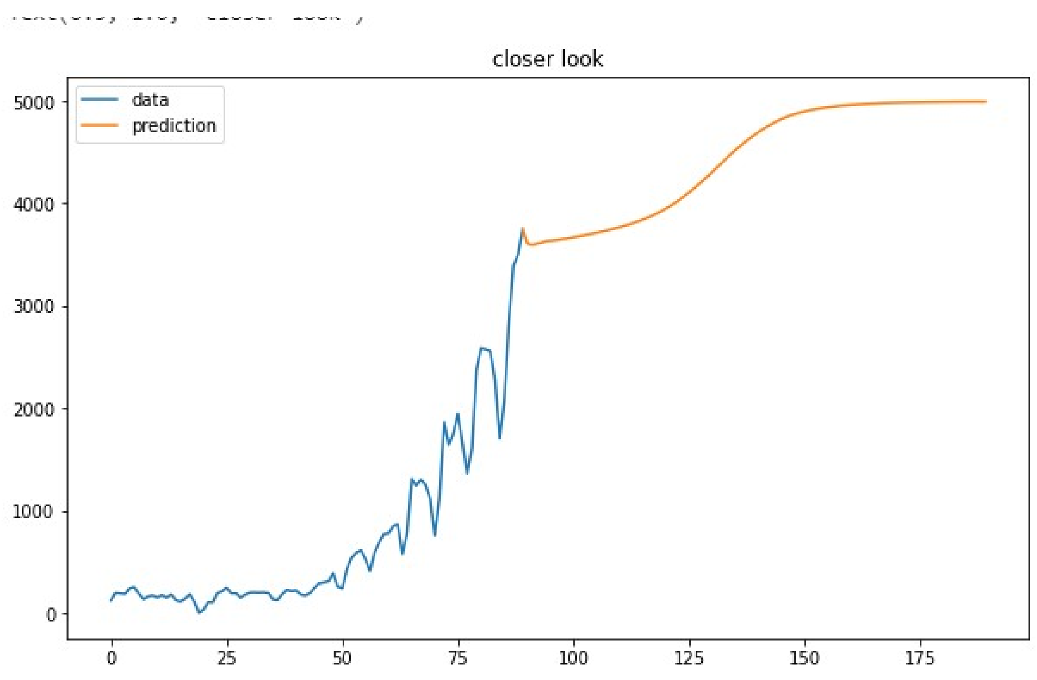
As for the prediction of the next 100 days, here is what the data looks like:

```
[29]: plt.figure(figsize=(10,6))
      plt.plot(a)
      plt.title('The model after adding 100 days')
```

```
[29]: Text(0.5, 1.0, 'The model after adding 100 days')
```



And here is a closer look at the data:



Discussion

Upon the creation of this project, I learned that LSTM models are not really good at extrapolating, which is why I only added a prediction for the next 100 days, and still, I will not quite consider it very reliable. However, I think that this is still an educated prediction of future cases based on existing data.

To summarize, what I created was an attempt to predict future Covid cases using an LSTM model with the training set of the middle 60% of the data. After tweaking the model's parameters with the test and validation set, I used the same model and used the entire data as training set in order to predict future sets. With this model, I predicted future values for the next 100 days. Finally, results are then displayed on Anvil.