Covid-19 LSTM Prediction

魏燿良 4108029016
湯鴻繁 4108029017
曾德旭 4108029019

Contents

Research Purpose:

- Forecast the active case of covid 19.
- Using the data from the past 14 days, we would like to predict the active cases on the 15th day

Choose which database to use.

- First, we Import the datasets, then, we take a look at which country has the most available data
- We choose Russia: because it has 30251 available data

```
Count of country datasets
Russia                  30251
US                      26740
Japan                   18059
Mainland China          15758
India                   13182
                        ...
Cape Verde                  1
('St. Martin',)             1
 Azerbaijan                 1
Republic of Ireland         1
Channel Islands             1
Name: Country/Region, Length: 229, dtype: int64
```

Data Preprocessing

Russia datasets

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
0	501	01/31/2020	NaN	Russia	1/31/2020 23:59	2.0	0.0	0.0
1	561	02/01/2020	NaN	Russia	1/31/2020 16:13	2.0	0.0	0.0
2	633	02/02/2020	NaN	Russia	2020-01-31T16:13:45	2.0	0.0	0.0
3	702	02/03/2020	NaN	Russia	2020-01-31T16:13:45	2.0	0.0	0.0
4	772	02/04/2020	NaN	Russia	2020-01-31T16:13:45	2.0	0.0	0.0
...	...	...	...	...	...	...	...	...
30246	306399	05/29/2021	Vologda Oblast	Russia	2021-05-30 04:20:55	46023.0	1092.0	42907.0
30247	306401	05/29/2021	Voronezh Oblast	Russia	2021-05-30 04:20:55	84672.0	2910.0	80119.0
30248	306416	05/29/2021	Yamalo-Nenets Autonomous Okrug	Russia	2021-05-30 04:20:55	39063.0	419.0	37848.0
30249	306418	05/29/2021	Yaroslavl Oblast	Russia	2021-05-30 04:20:55	40903.0	605.0	38968.0
30250	306422	05/29/2021	Zabaykalsky Krai	Russia	2021-05-30 04:20:55	43126.0	669.0	41650.0

30251 rows × 8 columns

1. *remove useless columns (SNo, last update, country/region)*

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
0	501	01/31/2020	NaN	Russia	1/31/2020 23:59	2.0	0.0	0.0
1	561	02/01/2020	NaN	Russia	1/31/2020 16:13	2.0	0.0	0.0
2	633	02/02/2020	NaN	Russia	2020-01-31T16:13:45	2.0	0.0	0.0
3	702	02/03/2020	NaN	Russia	2020-01-31T16:13:45	2.0	0.0	0.0
4	772	02/04/2020	NaN	Russia	2020-01-31T16:13:45	2.0	0.0	0.0
...	...	...	...	...	...	...	...	...

	ObservationDate	Province/State	Confirmed	Deaths	Recovered
0	01/31/2020	NaN	2.0	0.0	0.0
1	02/01/2020	NaN	2.0	0.0	0.0
2	02/02/2020	NaN	2.0	0.0	0.0
3	02/03/2020	NaN	2.0	0.0	0.0
4	02/04/2020	NaN	2.0	0.0	0.0
...	...	...	...	...	...

2. Check for null values, then we fill them with “Unknown”

```
russia_datasets = russia_datasets.fillna(value="unknown")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30251 entries, 0 to 30250
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ObservationDate  30251 non-null  object
 1   Province/State   30129 non-null  object
 2   Confirmed        30251 non-null  float64
 3   Deaths           30251 non-null  float64
 4   Recovered        30251 non-null  float64
 5   Active           30251 non-null  float64
dtypes: float64(4), object(2)
memory usage: 1.4+ MB
None
ObservationDate      0
Province/State     122
Confirmed            0
Deaths               0
Recovered            0
Active               0
dtype: int64
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30251 entries, 0 to 30250
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ObservationDate  30251 non-null  object
 1   Province/State   30251 non-null  object
 2   Confirmed        30251 non-null  float64
 3   Deaths           30251 non-null  float64
 4   Recovered        30251 non-null  float64
 5   Active           30251 non-null  float64
dtypes: float64(4), object(2)
memory usage: 1.4+ MB
None
ObservationDate    0
Province/State     0
Confirmed          0
Deaths             0
Recovered          0
Active             0
dtype: int64
```

3. *Create new column “Active”*

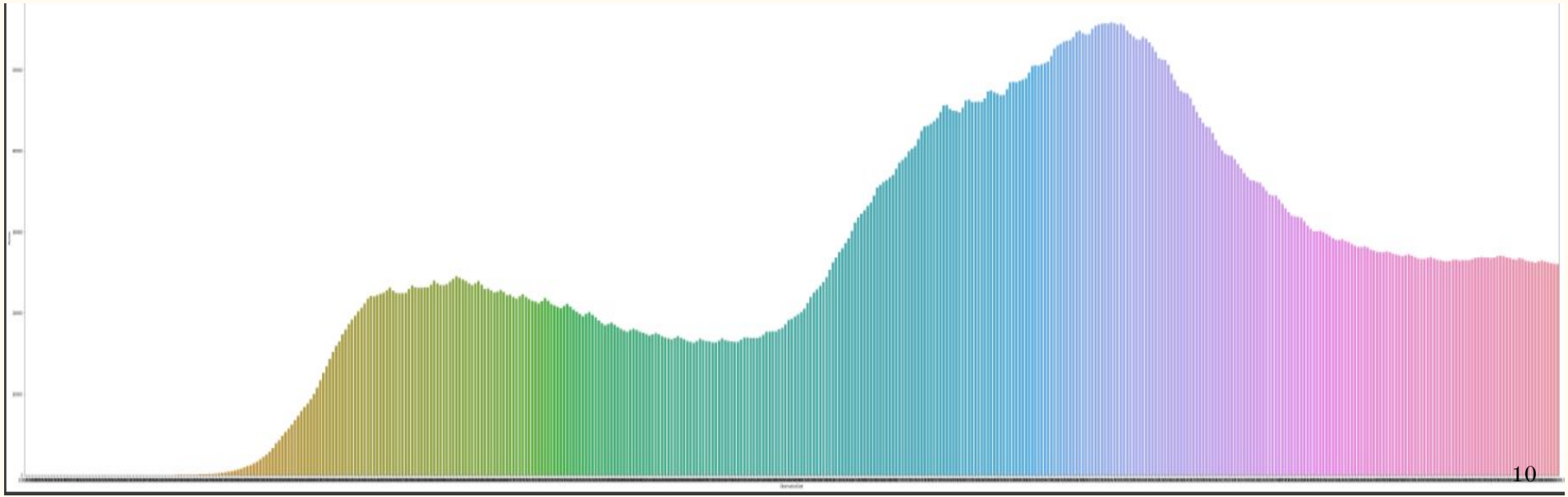
“Active” = daily active case = confirmed case – recovered

	ObservationDate	Province/State	Confirmed	Deaths	Recovered	Active
0	01/31/2020	unknown	2.0	0.0	0.0	2.0
1	02/01/2020	unknown	2.0	0.0	0.0	2.0
2	02/02/2020	unknown	2.0	0.0	0.0	2.0
3	02/03/2020	unknown	2.0	0.0	0.0	2.0
4	02/04/2020	unknown	2.0	0.0	0.0	2.0
...	...	...	...	...	...	...
30246	05/29/2021	Vologda Oblast	46023.0	1092.0	42907.0	2024.0
30247	05/29/2021	Voronezh Oblast	84672.0	2910.0	80119.0	1643.0
30248	05/29/2021	Yamalo-Nenets Autonomous Okrug	39063.0	419.0	37848.0	796.0
30249	05/29/2021	Yaroslavl Oblast	40903.0	605.0	38968.0	1330.0
30250	05/29/2021	Zabaykalsky Krai	43126.0	669.0	41650.0	807.0

30251 rows x 6 columns

3. *Create new column “Active”*

   “Active” = daily active case = confirmed case – recovered



4. *Divide our dataset into three:*

i. One is the overall data, remove the state information
ii. One is to keep State information
iii. Last One is to keep State information and do InfoGain process

	ObservationDate	Confirmed	Deaths	Recovered	Active
0	2020-01-31	2.0	0.0	0.0	2.0
1	2020-02-01	2.0	0.0	0.0	2.0
2	2020-02-02	2.0	0.0	0.0	2.0
3	2020-02-03	2.0	0.0	0.0	2.0
4	2020-02-04	2.0	0.0	0.0	2.0
...	...	...	...	...	...
480	2021-05-25	4960174.0	117197.0	4579421.0	263556.0
481	2021-05-26	4968421.0	117595.0	4588421.0	262405.0
482	2021-05-27	4977332.0	117990.0	4598014.0	261328.0
483	2021-05-28	4986458.0	118386.0	4607276.0	260796.0
484	2021-05-29	4995613.0	118781.0	4616422.0	260410.0

485 rows × 5 columns

	Confirmed							
Province/State	Adygea Republic	Altai Krai	Altai Republic	Amur Oblast	Arkhangelsk Oblast	Astrakhan Oblast	Bashkortostan Republic	Belgorod Oblast
ObservationDate								
01/01/2021	11103.0	33124.0	13901.0	16004.0	43679.0	19187.0	19234.0	22007.0
01/02/2021	11184.0	33334.0	13972.0	16140.0	44020.0	19361.0	19403.0	22201.0
01/03/2021	11261.0	33542.0	13978.0	16273.0	44249.0	19537.0	19572.0	22386.0
01/04/2021	11341.0	33748.0	13990.0	16401.0	44504.0	19708.0	19742.0	22574.0
01/05/2021	11420.0	33957.0	14019.0	16533.0	44755.0	19877.0	19910.0	22765.0
...	...	...	...	...	...	...	...	...
12/27/2020	10676.0	32062.0	13624.0	15324.0	41859.0	18304.0	18395.0	21082.0
12/28/2020	10767.0	32276.0	13687.0	15459.0	42233.0	18481.0	18562.0	21262.0
12/29/2020	10856.0	32488.0	13719.0	15599.0	42605.0	18659.0	18728.0	21444.0
12/30/2020	10945.0	32699.0	13753.0	15735.0	42968.0	18835.0	18897.0	21631.0
12/31/2020	11025.0	32912.0	13822.0	15866.0	43327.0	19010.0	19064.0	21817.0

485 rows × 336 columns

5. *Info Gain Process*

- *drop column which infogain value less than mean(4.677)*

```
1) ('Confirmed', 'Adygea Republic') = 4.975715
2) ('Confirmed', 'Altai Krai') = 4.975715
3) ('Confirmed', 'Altai Republic') = 4.975715
4) ('Confirmed', 'Amur Oblast') = 4.975715
5) ('Confirmed', 'Arkhangelsk Oblast') = 4.975715
6) ('Confirmed', 'Astrakhan Oblast') = 4.975715
7) ('Confirmed', 'Bashkortostan Republic') = 4.975715
8) ('Confirmed', 'Belgorod Oblast') = 4.975715
9) ('Confirmed', 'Bryansk Oblast') = 4.975715
10) ('Confirmed', 'Buryatia Republic') = 4.975715
11) ('Confirmed', 'Chechen Republic') = 4.975715
12) ('Confirmed', 'Chelyabinsk Oblast') = 4.975715
```

Info Gain:

	1
count	252.000000
mean	4.677347
std	0.645663
min	0.772829
25%	4.675076
50%	4.965171
75%	4.975715
max	4.975715

Before:

485 rows × 253 columns

After:

485 rows × 189 columns

5. *Compare which datasets are more suitable*

- Standard scale
- 14 days as a sequence, batch_size = 10

Remove the "state" column

```
Train datasets
38 torch.Size([10, 14, 4])
38 torch.Size([10, 1])

Test datasets
9 torch.Size([10, 14, 4])
9 torch.Size([10, 1])
```

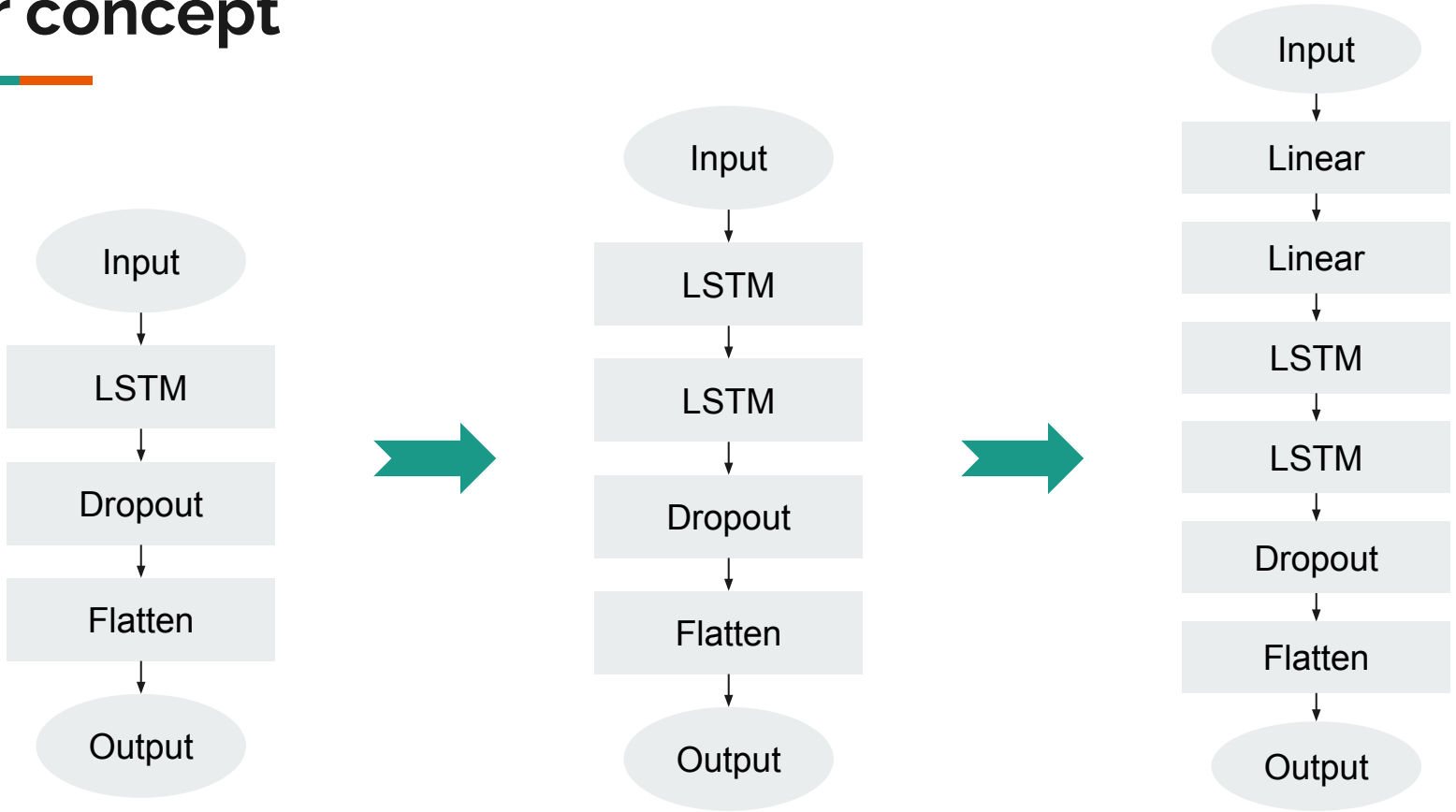
keep "state" column

```
Train datasets
39 torch.Size([10, 14, 253])
39 torch.Size([10, 1])

Test datasets
9 torch.Size([10, 14, 253])
9 torch.Size([10, 1])
```

LSTM Model

Our concept


Input
LSTM
Dropout
Flatten
Output
Input
LSTM
LSTM
Dropout
Flatten
Output
Input
Linear
Linear
LSTM
LSTM
Dropout
Flatten
Output

We made 8 LSTM models

1. single lstm model (with 4 features)

```
LSTMNetwork3(
  (lstm1): LSTM(4, 4, batch_first=True)
  (dropout): Dropout(p=0.2, inplace=False)
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc2): Linear(in_features=56, out_features=1, bias=True)
)
```

2. single lstm model (with 253 features)

```
LSTMNetwork3(
  (lstm1): LSTM(253, 253, batch_first=True)
  (dropout): Dropout(p=0.2, inplace=False)
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc2): Linear(in_features=3542, out_features=1, bias=True)
)
```

We made 8 LSTM models

3. single lstm model (with 189 features)

```
LSTMNetwork4(
  (lstm1): LSTM(189, 189, batch_first=True)
  (dropout): Dropout(p=0.2, inplace=False)
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc2): Linear(in_features=2646, out_features=1, bias=True)
)
```

4. double lstm (linear + lstm with 253 features)

```
LSTMNetwork2(
  (fc_b1): Linear(in_features=253, out_features=126, bias=True)
  (fc_b2): Linear(in_features=126, out_features=64, bias=True)
  (lstm1): LSTM(64, 64, batch_first=True)
  (fc1): Linear(in_features=64, out_features=32, bias=True)
  (lstm2): LSTM(32, 32, batch_first=True)
  (dropout): Dropout(p=0.2, inplace=False)
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc2): Linear(in_features=448, out_features=1, bias=True)
)
```

We made 8 LSTM models

5. double lstm (linear + lstm with 189 features)

```
LSTMNetwork2(
  (fc_b1): Linear(in_features=189, out_features=94, bias=True)
  (fc_b2): Linear(in_features=94, out_features=64, bias=True)
  (lstm1): LSTM(64, 64, batch_first=True)
  (fc1): Linear(in_features=64, out_features=32, bias=True)
  (lstm2): LSTM(32, 32, batch_first=True)
  (dropout): Dropout(p=0.2, inplace=False)
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc2): Linear(in_features=448, out_features=1, bias=True)
)
```

6. double lstm (253 features)

```
LSTMNetwork3(
  (lstm1): LSTM(253, 253, batch_first=True)
  (fc1): Linear(in_features=253, out_features=64, bias=True)
  (lstm2): LSTM(64, 64, batch_first=True)
  (dropout): Dropout(p=0.2, inplace=False)
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc2): Linear(in_features=896, out_features=1, bias=True)
)
```

We made 8 LSTM models

7. double lstm (189 features)

```
LSTMNetwork3(
  (lstm1): LSTM(189, 189, batch_first=True)
  (fc1): Linear(in_features=189, out_features=64, bias=True)
  (lstm2): LSTM(64, 64, batch_first=True)
  (dropout): Dropout(p=0.2, inplace=False)
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc2): Linear(in_features=896, out_features=1, bias=True)
)
```

8. double lstm (4 features)

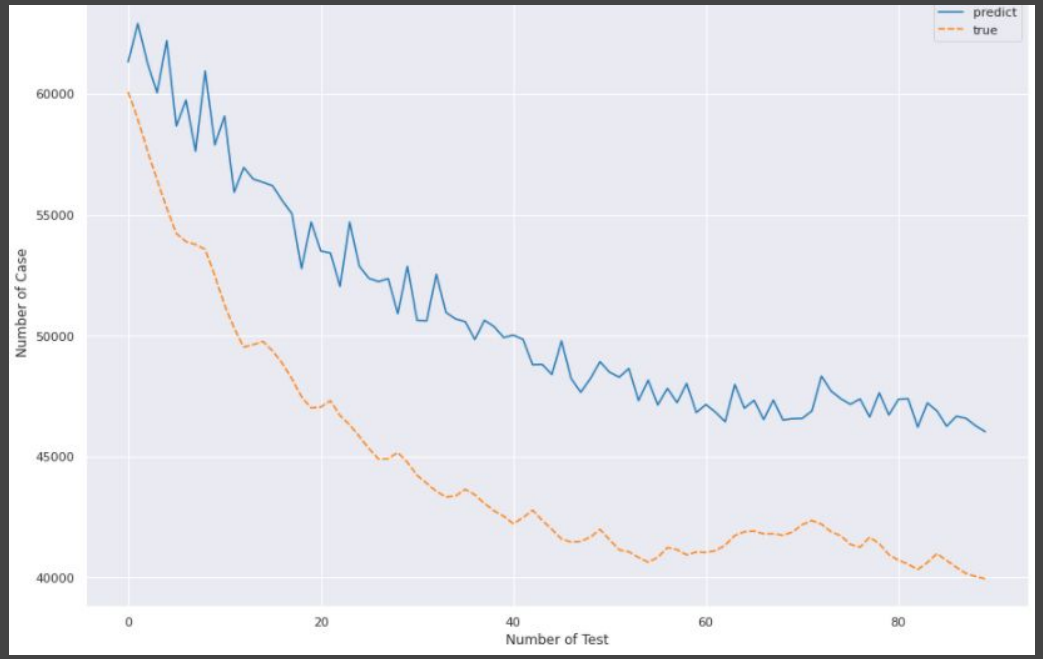
```
LSTMNetwork(
  (lstm1): LSTM(4, 64, batch_first=True)
  (fc1): Linear(in_features=64, out_features=32, bias=True)
  (lstm2): LSTM(32, 32, batch_first=True)
  (dropout): Dropout(p=0.2, inplace=False)
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc2): Linear(in_features=448, out_features=1, bias=True)
)
```

- Loss function = MSE Loss
- Optimizer = Adam

Validation Test

For Each Models

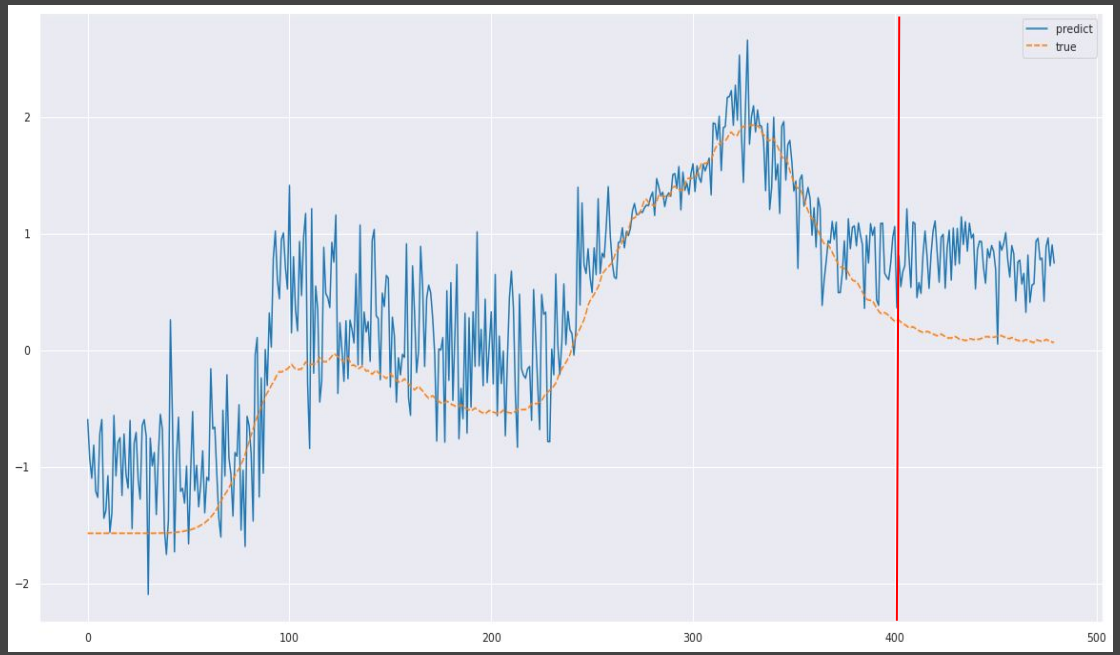
1. Single LSTM Model Result



Learning Rate = 0.01
Using Dataset1 (Drop State)
epoch 2000

	predict	true	difference
count	90.000000	90.000000	90.000000
mean	50793.844444	44533.188889	6260.655556
std	4556.452940	4837.874691	1205.478311
min	46028.000000	39947.000000	1241.000000
25%	47253.750000	41271.500000	5732.000000
50%	48809.000000	42214.000000	6406.000000
75%	52875.250000	46601.750000	7016.000000
max	62908.000000	60081.000000	8978.000000

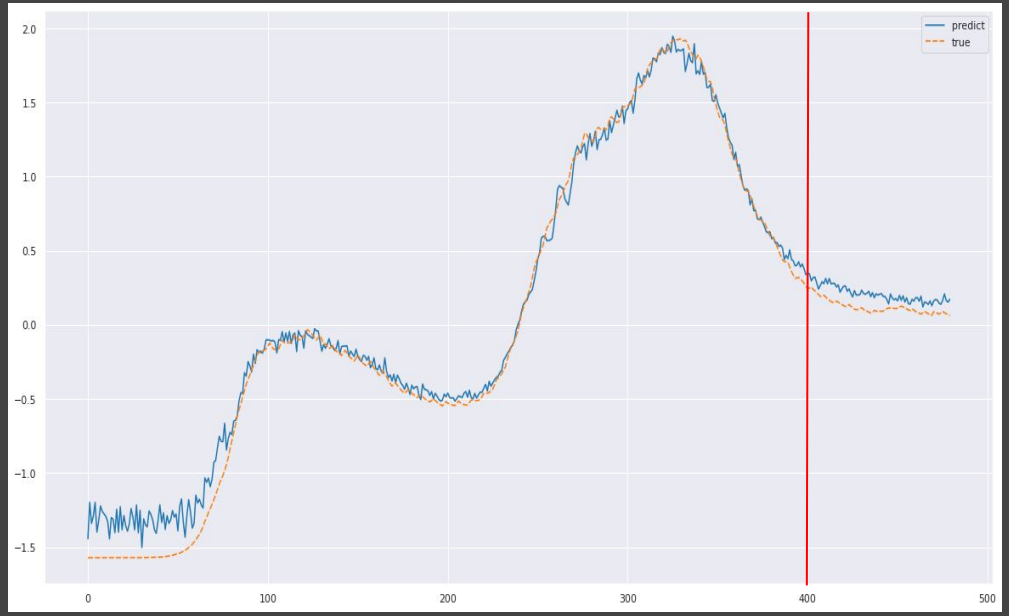
1. Single LSTM Model Result



Learning Rate = 0.01
Using Dataset2 (Use State)
epoch 2000

	predict	true	difference
count	480.000000	480.000000	480.000000
mean	312863.687500	258016.543750	68992.885417
std	148046.993264	154422.014669	51048.821962
min	-103132.000000	0.000000	2.000000
25%	204033.500000	173340.000000	25168.000000
50%	335866.000000	249238.500000	62670.500000
75%	411490.000000	358601.250000	106353.000000
max	688681.000000	558146.000000	250588.000000

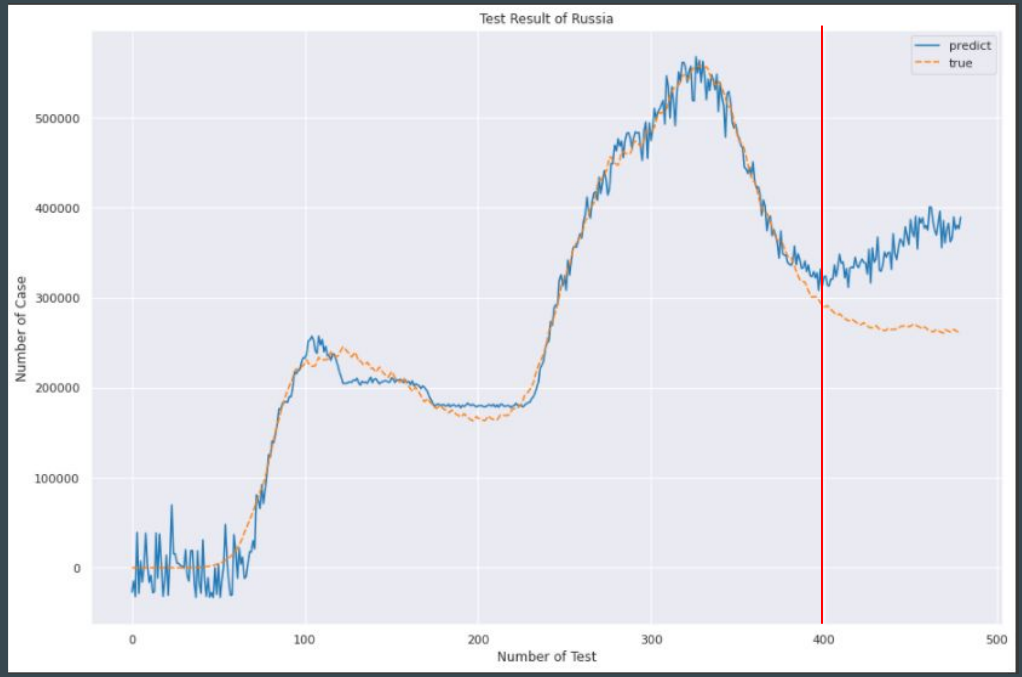
1. Single LSTM Model Result



Learning Rate = 0.001
Using Dataset2(Use State) + Info gain
epoch 2000

	predict	true	difference
count	480.000000	480.000000	480.000000
mean	267876.683333	258016.543750	13542.847917
std	143116.317052	154422.014669	13132.837032
min	11005.000000	0.000000	2.000000
25%	179415.250000	173340.000000	4646.750000
50%	253289.000000	249238.500000	9841.500000
75%	349657.250000	358601.250000	16891.000000
max	560914.000000	558146.000000	59580.000000

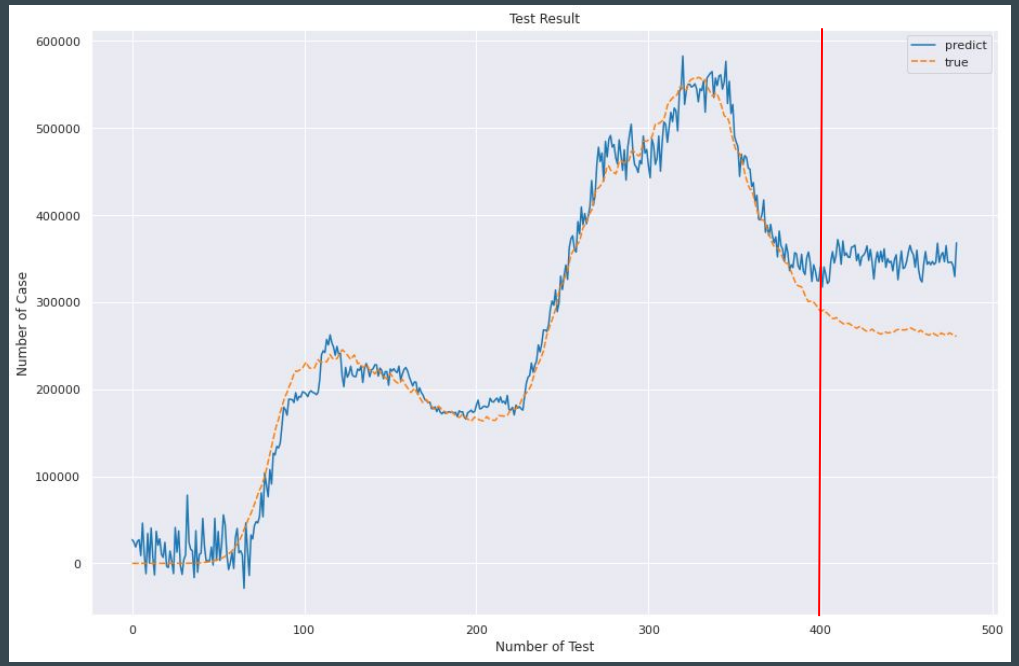
2. Double LSTM Result (Linear + LSTM)



Learning Rate = 0.001
Using Dataset2(Use State)
epoch 1500

	predict	true	difference
count	480.000000	480.000000	480.000000
mean	271202.266667	258016.543750	25617.922917
std	161453.818990	154422.014669	31035.515480
min	-33041.000000	0.000000	69.000000
25%	180510.500000	173340.000000	5982.750000
50%	255339.000000	249238.500000	13737.500000
75%	383033.000000	358601.250000	30084.000000
max	567892.000000	558146.000000	137542.000000

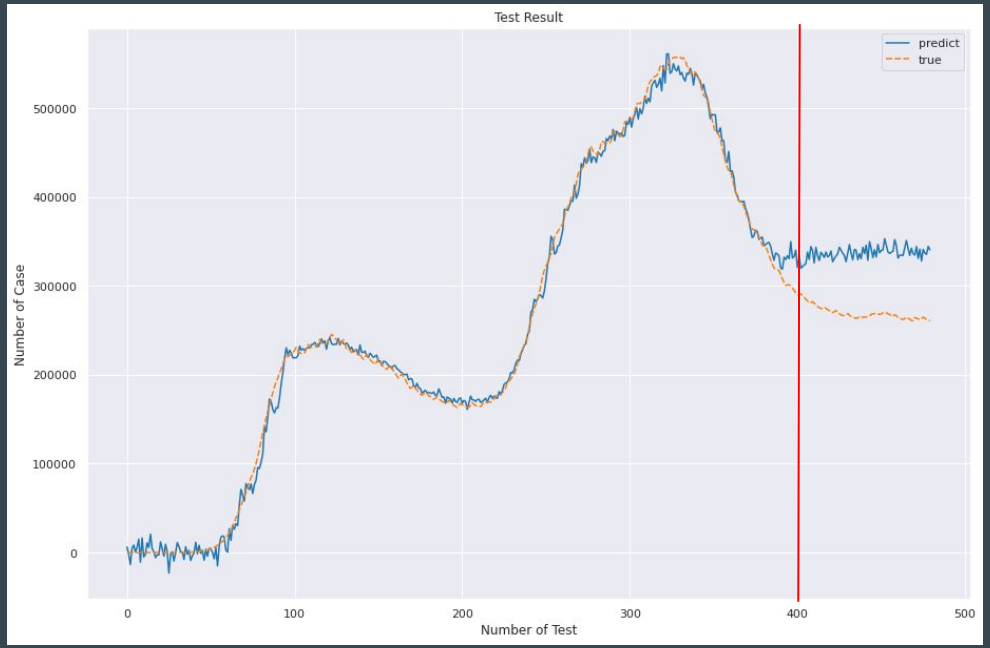
2. Double LSTM Result (Linear + LSTM)



Learning Rate = 0.001
Using Dataset2(Use State) + Info Gain
epoch 2000

	predict	true	difference
count	480.00000	480.000000	480.000000
mean	274016.81250	258016.543750	27153.656250
std	158645.05205	154422.014669	26921.253818
min	-28659.00000	0.000000	120.000000
25%	177880.75000	173340.000000	7751.750000
50%	267140.50000	249238.500000	17242.500000
75%	368612.75000	358601.250000	35594.000000
max	582546.00000	558146.000000	107285.000000

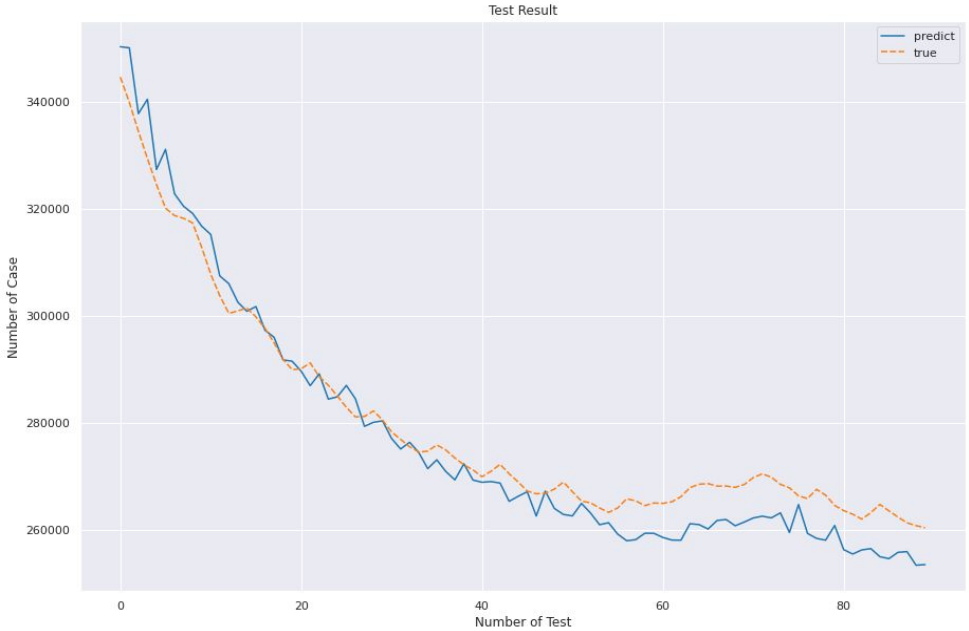
2. Double LSTM Result (Linear + LSTM)



Learning Rate = 0.0001 (Lower)
Using Dataset2(Use State) + info gain
epoch 2000

	predict	true	difference
count	480.000000	480.000000	480.000000
mean	269201.379167	258016.543750	17538.260417
std	156488.396674	154422.014669	23759.037179
min	-23148.000000	0.000000	7.000000
25%	174522.500000	173340.000000	3011.250000
50%	247029.000000	249238.500000	7024.000000
75%	353711.750000	358601.250000	17646.500000
max	561125.000000	558146.000000	86377.000000

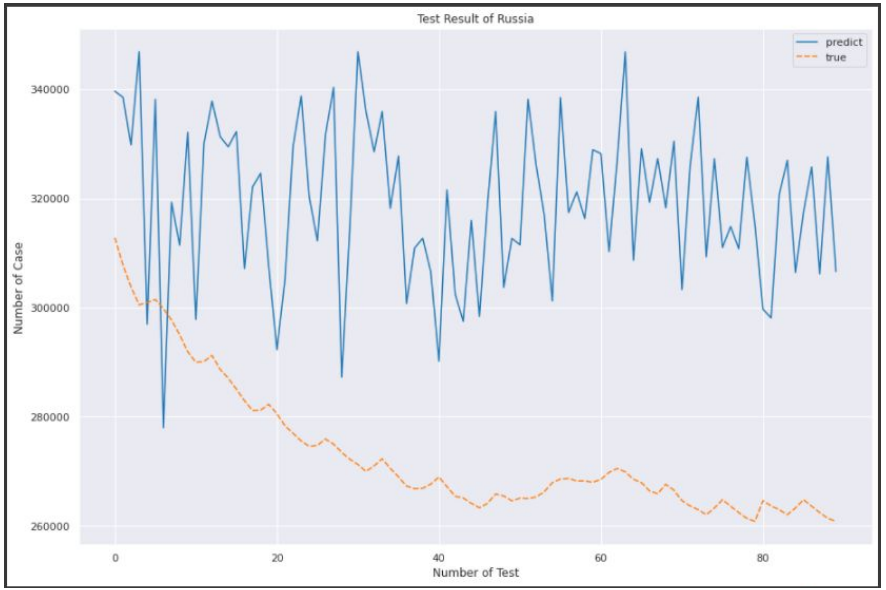
2. Double LSTM Result (LSTM only)



Learning Rate = 0.001
Using Dataset1(Drop State)
epoch 1500

	predict	true	difference
count	90.000000	90.000000	90.000000
mean	277101.955556	279592.022222	4452.688889
std	24284.933647	20237.180879	2956.079238
min	253396.000000	260409.000000	70.000000
25%	260315.000000	265947.000000	1896.250000
50%	266704.000000	269890.500000	4086.000000
75%	286991.250000	288245.000000	6986.750000
max	350304.000000	344629.000000	11069.000000

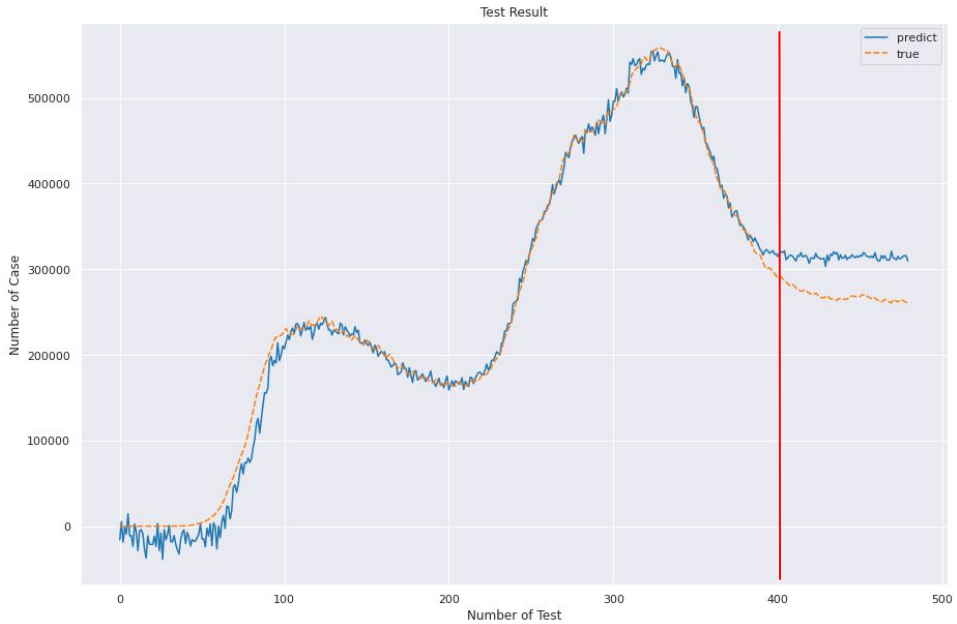
2. Double LSTM Result (LSTM only)



Learning Rate = 0.001
Using Dataset2(Use State)
epoch 1500

	predict	true	difference
count	90.000000	90.000000	90.000000
mean	319177.322222	273162.277778	46589.822222
std	14688.102909	12396.270724	15897.297145
min	277947.000000	260795.000000	4000.000000
25%	308811.750000	264757.000000	37699.250000
50%	319300.500000	268186.000000	46938.500000
75%	329570.500000	276666.250000	58568.500000
max	346896.000000	312819.000000	77039.000000

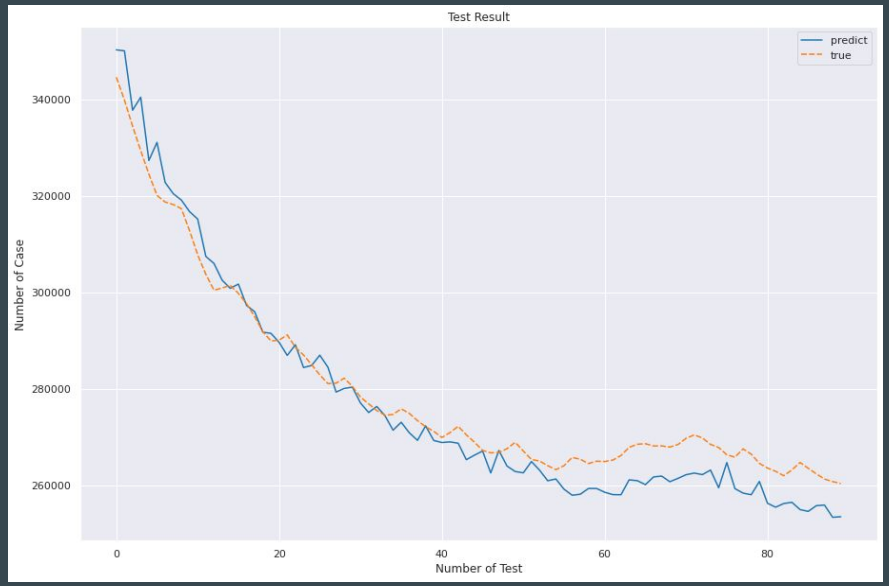
2. Double LSTM Result (LSTM only)



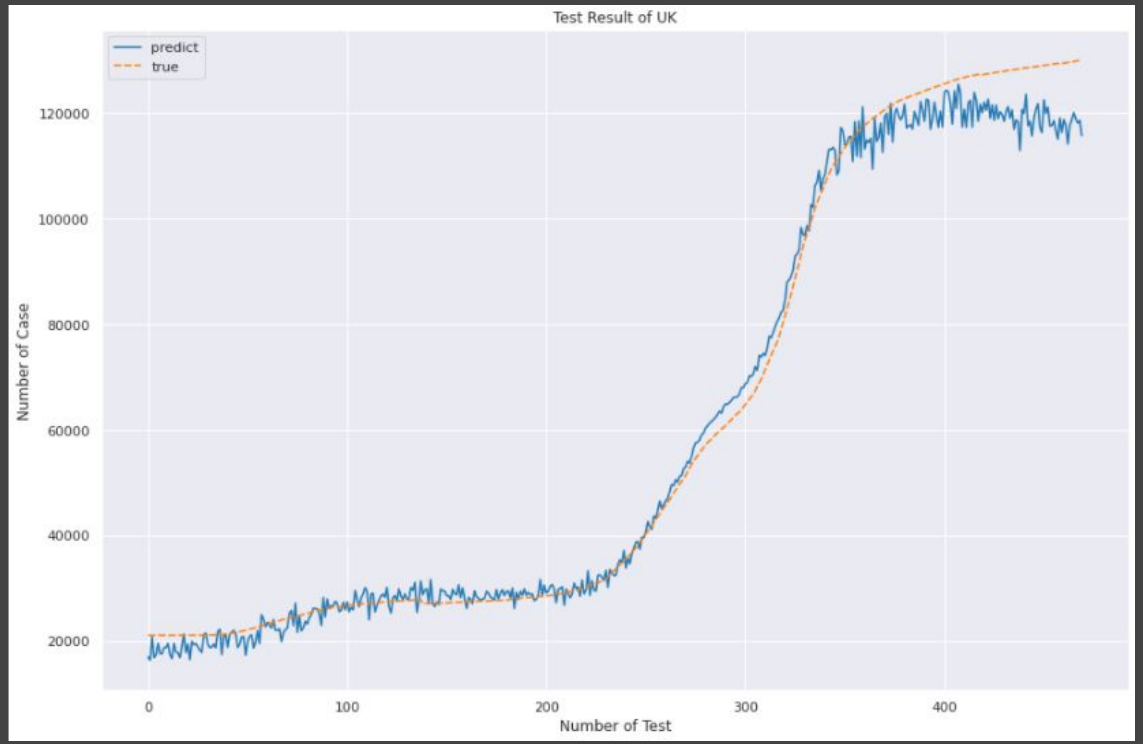
Learning Rate = 0.001
Using Dataset2(Use State) + Info Gain
epoch 1000

	predict	true	difference
count	480.000000	480.000000	480.000000
mean	261880.897917	258016.543750	15640.420833
std	161200.046599	154422.014669	16075.558367
min	-38440.000000	0.000000	2.000000
25%	173243.500000	173340.000000	3761.750000
50%	260381.000000	249238.500000	8441.500000
75%	357489.250000	358601.250000	23596.000000
max	554822.000000	558146.000000	60185.000000

- Double LSTM works better. Therefore, we save this model and test it on other countries.



United Kingdom

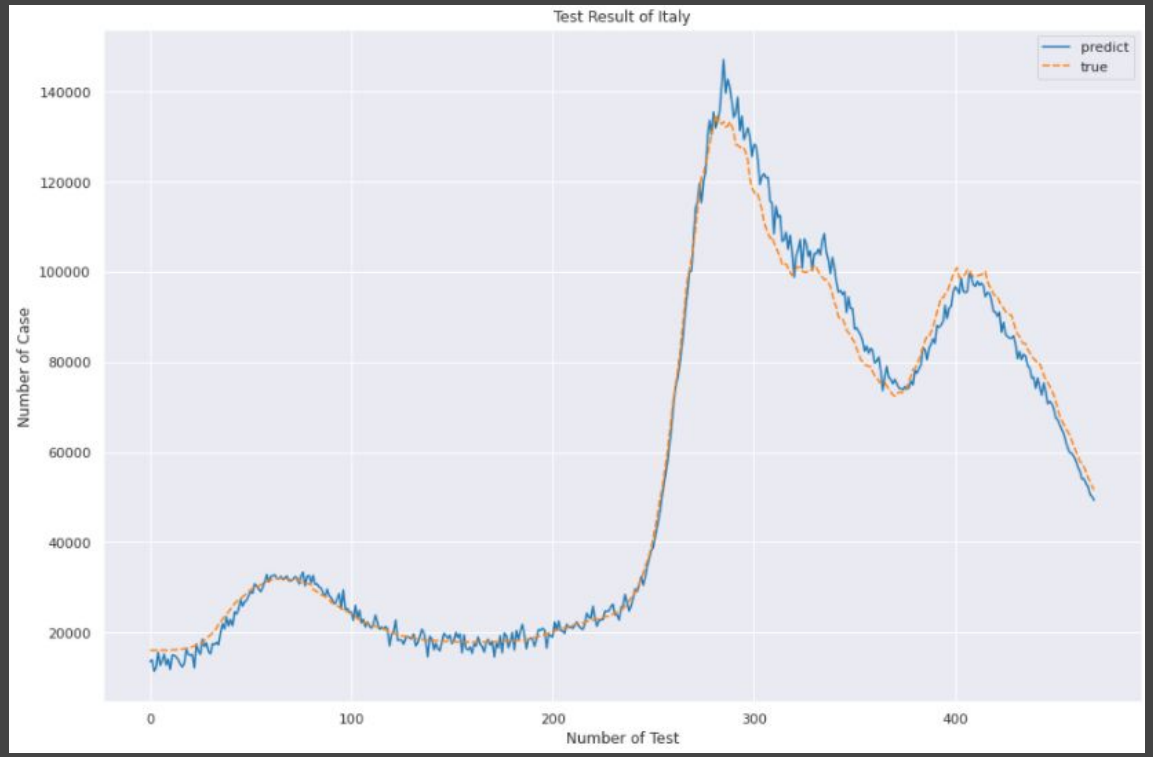


	predict	true	difference
0	16958	21075	4117
1	16369	21075	4706
2	20771	21075	304
3	16866	21075	4209
4	17362	21075	3713
...	...	...	...
465	120092	129777	9685
466	118916	129851	10935
467	118125	129935	11810
468	118625	130036	11411
469	115728	130117	14389

470 rows × 3 columns

	predict	true	difference
count	470.000000	470.000000	470.000000
mean	60031.178723	61167.368085	3048.887234
std	40508.874485	42608.491574	3018.583585
min	16369.000000	21075.000000	2.000000
25%	27445.750000	27152.500000	931.250000
50%	33463.000000	33470.500000	2059.500000
75%	113117.750000	114399.250000	3796.500000
max	125497.000000	130117.000000	15476.000000

Italy

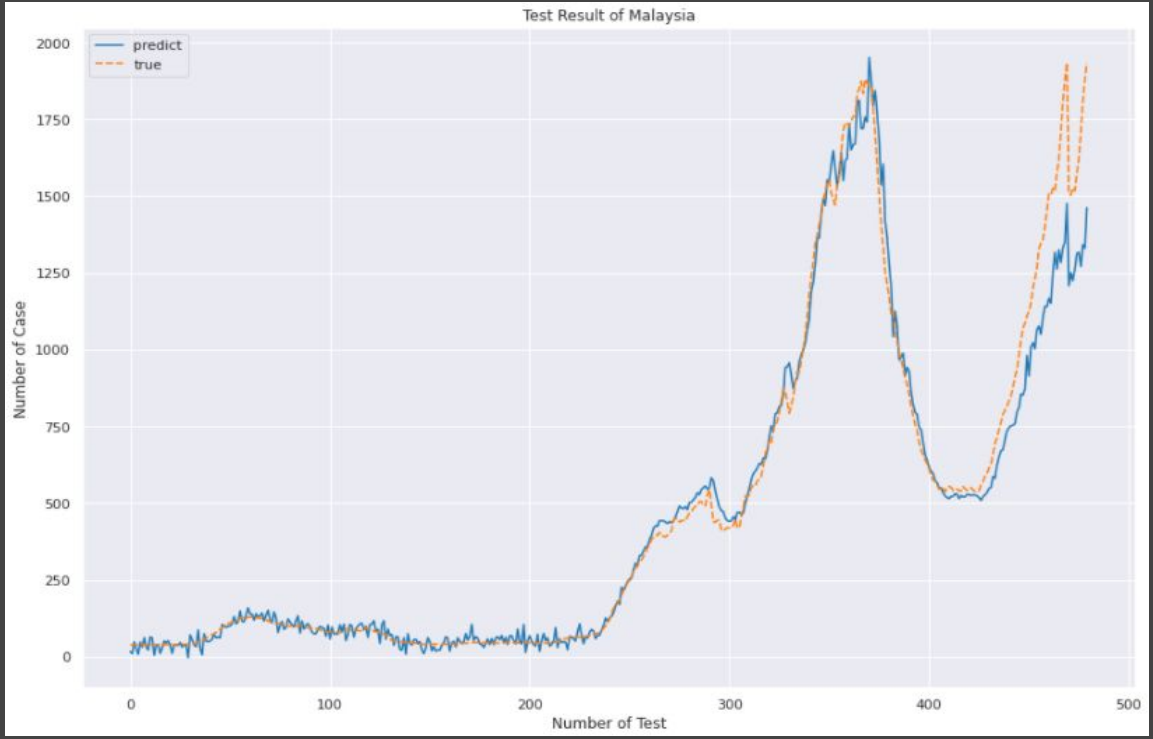


	predict	true	difference
0	13507	16026	2519
1	13915	16026	2111
2	11368	16026	4658
3	12166	16026	3860
4	15535	16026	491
...	...	...	...
465	53029	55558	2529
466	52265	54361	2096
467	50576	53354	2778
468	50110	52333	2223
469	49307	51698	2391

470 rows × 3 columns

	predict	true	difference
count	470.000000	470.000000	470.000000
mean	54737.629787	54574.691489	2587.014894
std	38465.988979	37233.695132	2324.501073
min	11368.000000	16026.000000	6.000000
25%	20621.250000	20482.750000	854.250000
50%	32332.500000	31699.500000	2062.000000
75%	87539.250000	89513.500000	3376.750000
max	147109.000000	134846.000000	13752.000000

Malaysia

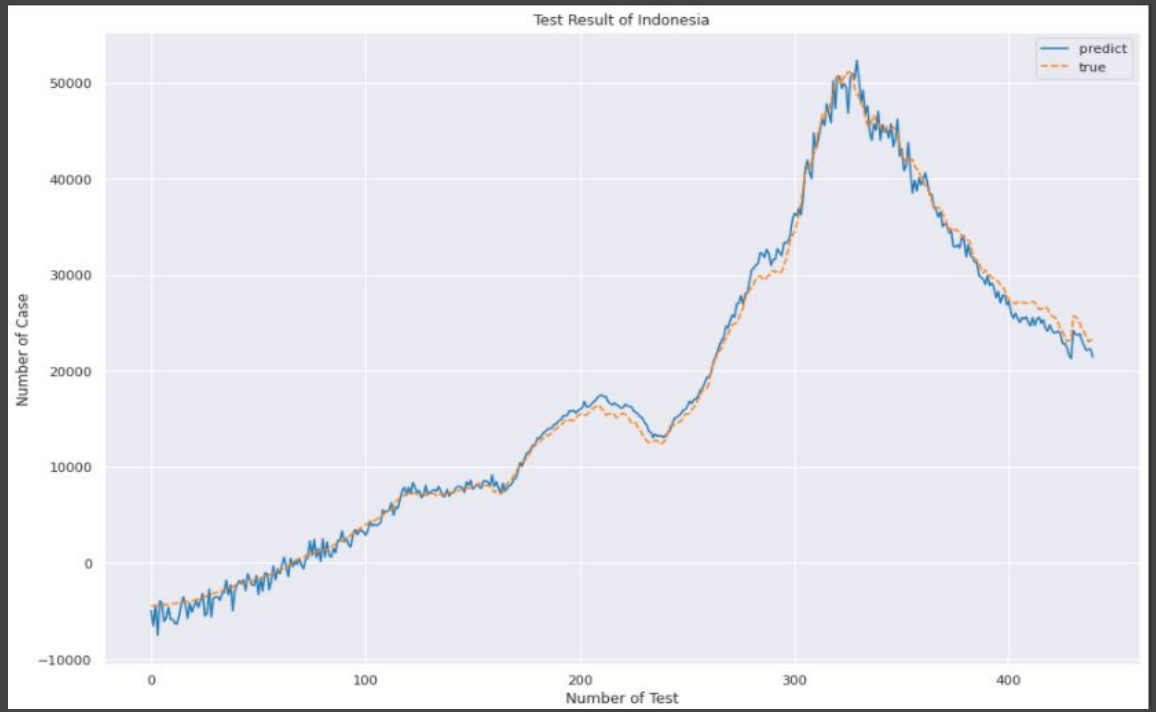


	predict	true	difference
0	17	39	22
1	10	39	29
2	48	39	9
3	26	39	13
4	8	39	31
...	...	...	...
475	1317	1621	304
476	1271	1710	439
477	1342	1810	468
478	1330	1878	548
479	1462	1934	472

480 rows × 3 columns

	predict	true	difference
count	480.000000	480.000000	480.000000
mean	454.264583	472.022917	49.679167
std	497.858800	537.163844	83.700013
min	-3.000000	39.000000	0.000000
25%	63.750000	51.750000	9.000000
50%	147.500000	129.500000	20.000000
75%	680.250000	699.500000	44.250000
max	1951.000000	1934.000000	548.000000

Indonesia

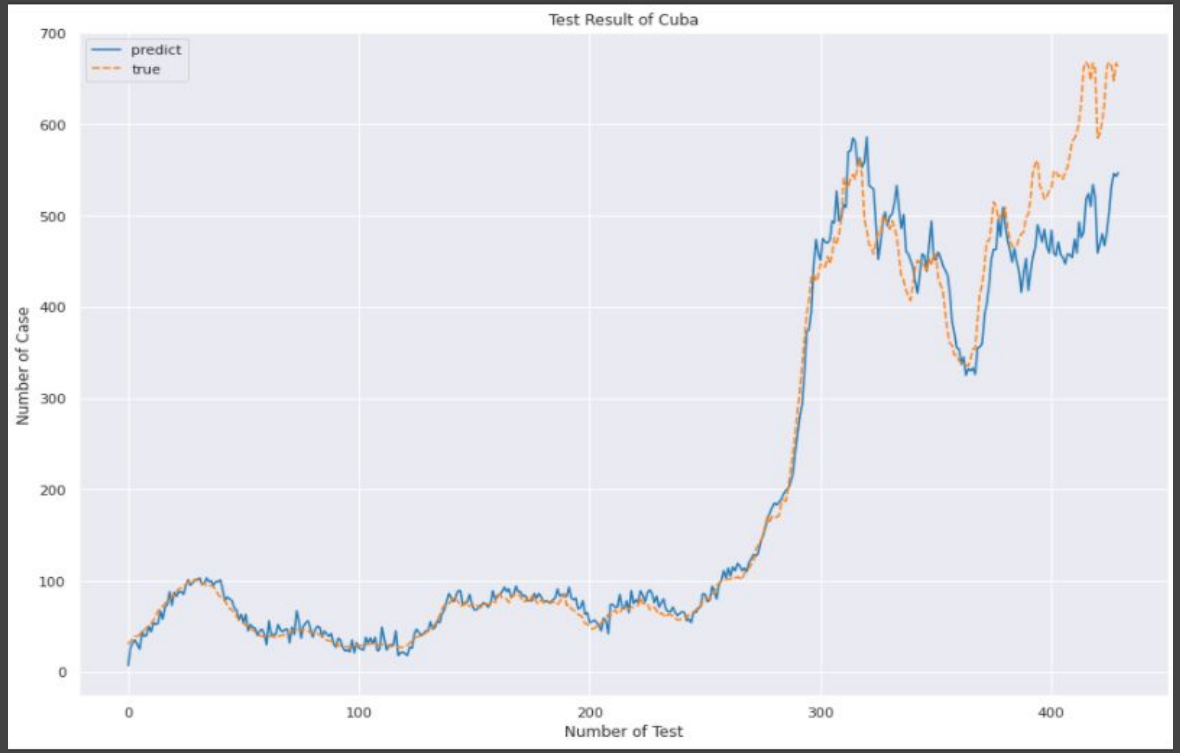


	predict	true	difference
0	-4945	-4472	473
1	-6554	-4460	2094
2	-4401	-4436	35
3	-7527	-4421	3106
4	-3937	-4397	460
...	...	...	...
435	22657	24091	1434
436	22147	23564	1417
437	22247	23055	808
438	22285	23154	869
439	21447	23347	1900

440 rows × 3 columns

	predict	true	difference
count	440.000000	440.000000	440.000000
mean	17667.179545	17722.247727	872.177273
std	15729.230236	15718.311960	671.052670
min	-7527.000000	-4472.000000	0.000000
25%	5365.250000	5344.500000	391.750000
50%	15843.000000	14907.000000	679.000000
75%	29363.500000	29605.000000	1273.500000
max	52320.000000	51152.000000	4345.000000

Cuba



	predict	true	difference
0	7	30	23
1	26	34	8
2	32	36	4
3	35	39	4
4	30	39	9
...	...	...	...
425	503	668	165
426	532	664	132
427	546	648	102
428	543	667	124
429	547	661	114

430 rows × 3 columns

	predict	true	difference
count	430.000000	430.000000	430.000000
mean	198.893023	205.946512	21.332558
std	187.646851	204.049264	33.191310
min	7.000000	27.000000	0.000000
25%	57.000000	57.000000	4.000000
50%	86.000000	81.000000	8.000000
75%	438.000000	432.750000	19.750000
max	586.000000	668.000000	181.000000

CONCLUSION

1. Add more data for training different situations
2. Good performance after self data preprocessing
3. Lower LR for smoother performance and learning
4. Too many epochs can lead to overfitting
5. Try different models, more layers is better

Reference:

1.How to Develop LSTM Models for Time Series Forecasting:

https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

2.Stacked LSTM

https://machinelearningmastery.com/stacked-long-short-term-memory-networks/

3.Python nn.LSTM屬性代碼示例

https://vimsky.com/zh-tw/examples/detail/python-attribute-torch.nn.LSTM.html

4.Predicting Stock Prices Using Deep Learning Models

https://medium.com/swlh/predicting-stock-prices-using-deep-learning-models-310b41cec90a

謝謝聆聽