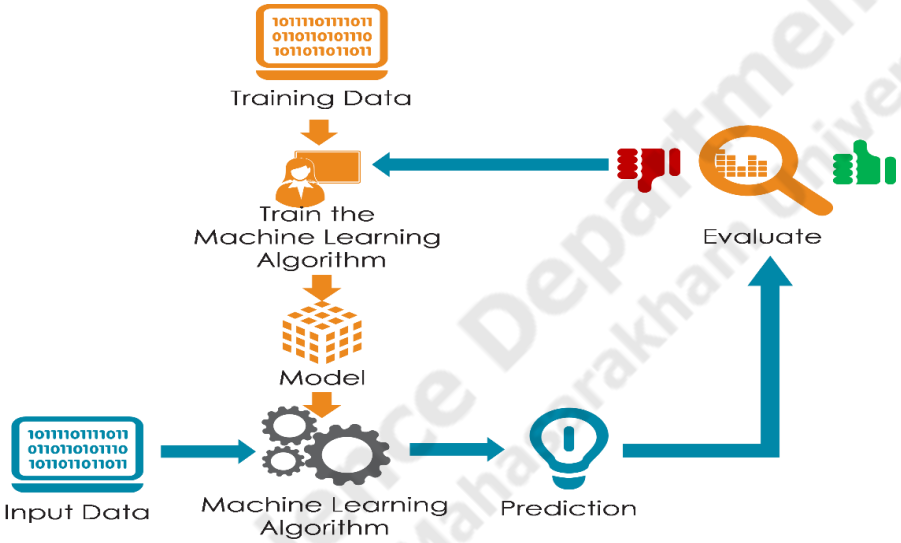
บทที่ 3

การดำเนินงาน

3.1 ภาพรวมของระบบ

1. การทำงานส่วนสร้างโมเดล



ภาพประกอบที่ 3.1 การสร้างโมเดลในการพยากรณ์ปริมาณฝุ่นละอองขนาดเล็ก(PM 2.5)

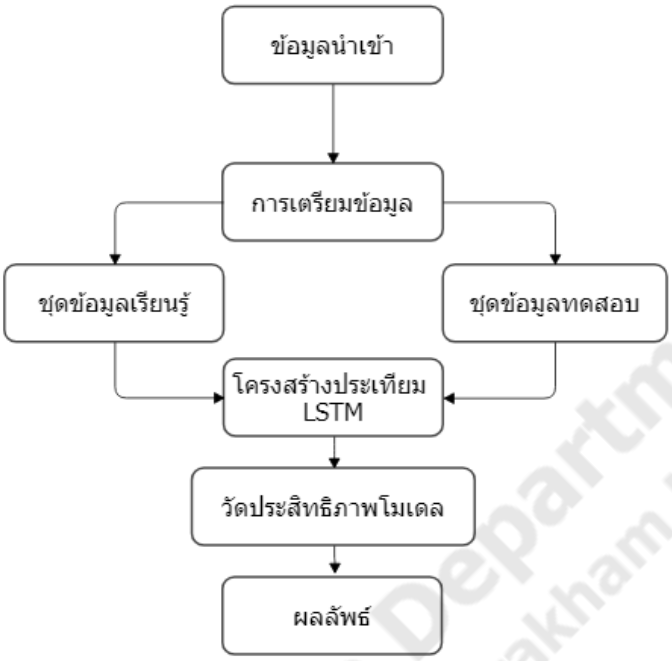
จากภาพที่ 3.1 นำชุดข้อมูลมาทำจัดเตรียมชุดข้อมูลก่อนสร้างโมเดลจากนั้นแบ่งชุดข้อมูลออกเป็นสองชุดคือ ชุดเทรนและชุดทดสอบจากนั้นนำข้อมูลชุดเทรนมาเข้าสู่ model จากนั้นนำข้อมูลชุดทดสอบเข้าสู่โมเดล จากนั้นจึงทำการวัดประสิทธิภาพของโมเดล

2. การทำงานส่วน Back-end และ Front-end

ภาพประกอบที่ 3.2 การทำงานส่วน Back-end และ Front-end

จากภาพประกอบที่ 3.2 ส่วน Back-end ที่ทำงานอยู่ด้านหลังจะคอยติดต่อกับเซิฟเวอร์หรือฐานข้อมูลเพื่อนำข้อมูลมาแสดงออกทาง Front-end เช่น Mobile App หรือ Web Service

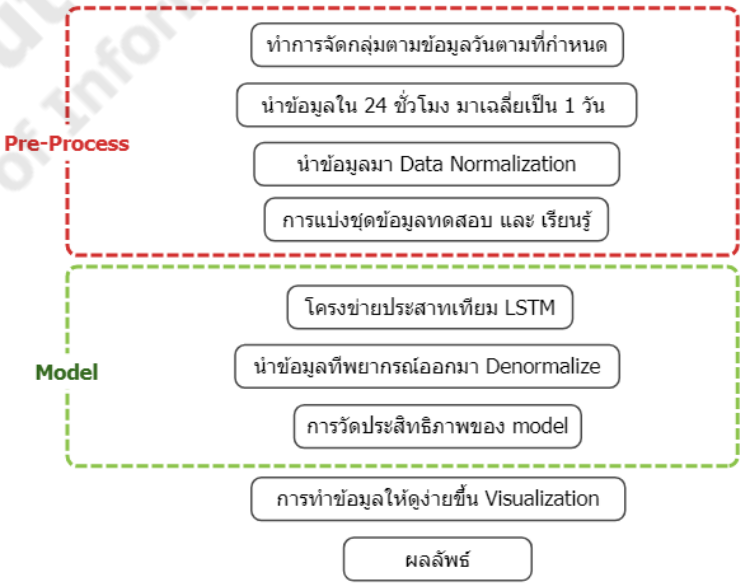
3.2 การทำงานส่วนของสร้าง Model



ภาพประกอบที่ 3.3 ตัวอย่างการทำงานของ Model

จากภาพประกอบที่ 3.3 ชุดข้อมูลที่ถูกนำเข้าจะต้องผ่านการเตรียมข้อมูล เมื่อเสร็จแล้วจะแบ่งเป็นชุดทดสอบและชุดเรียนรู้ก่อนจะนำไปสู่การพยากรณ์ด้วยโครงข่ายประสาทเทียม จากนั้นจะทำการวัดประสิทธิภาพของโมเดล และจะคำนวณผลลัพธ์

3.3 ขั้นตอนการเรียนรู้และทดสอบ



ภาพประกอบที่ 3.4 ตัวอย่างการทำงานของ Model

จากภาพที่ 3.4 การเรียนรู้และทดสอบเมื่อนำข้อมูลเข้ามาแล้วนั้น เริ่มแรกจะทำการตัดข้อมูลที่ไม่ต้องการทิ้งและจะเก็บข้อมูลที่ไว้ นำข้อมูลที่เหลือมาเฉลี่ยค่าเป็นวันจนครบ จากนั้นจะนำข้อมูลมาทำการแปลงขนาดด้วย Data Normalization

3.3.1 การนำเข้า Input

ข้อมูลที่นำเข้าไปคำนวณในการหาค่า PM 2.5 คือ ข้อมูลสภาพอากาศของมณฑลปักกิ่ง ประเทศจีน ข้อมูลทั้งหมดได้มาจาก Beijing PM2.5 Data Data Set ในเว็บ UCI (UCI Machine Learning Repository: Beijing PM2.5 Data Data Set) โดยจะมีข้อมูลดังนี้

No. หมายเลขข้อมูล
Year ปีของข้อมูล
Month เดือนของข้อมูล
Day วันที่ของข้อมูล
Hour ชั่วโมงของข้อมูล
Pm2.5 ค่าความเข้มข้นของค่าฝุ่นละอองขนาดเล็ก
TEMP ข้อมูลอุณหภูมิ
PRES ข้อมูลความดันอากาศ
cbwd ข้อมูลทิศทางลม
Iws ข้อมูลทิศทางลม
Is ข้อมูลหิมะ
Ir ข้อมูลฝน

ตารางที่ 3.1 ตารางสภาพอากาศในแต่ละช่วงเวลา

No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	Iws	Is	Ir
1	2010	4	18	10	129	6	13	1014	NW	1.79	0	0
2	2010	4	18	11	148	6	14	1013	cv	1.79	0	0
3	2010	4	18	12	159	6	16	1012	cv	2.68	0	0
4	2010	4	18	13	N/A	6	18	1011	SE	1.79	0	0
5	2010	4	18	14	N/A	6	19	1011	cv	1.79	0	0
6	2010	4	18	15	N/A	6	19	1010	SE	3.13	0	0
7	2010	4	18	16	105	6	20	1009	SE	6.26	0	0
8	2010	4	18	17	124	6	20	1009	SE	9.39	0	0

ในการทดลองครั้งนี้ข้อมูลที่จะเก็บคือ year month day และ pm2.5

ตารางที่ 3.2 ค่าข้อมูลที่จะนำมาใช้งาน

No	year	month	day	pm2.5
1	2010	4	18	129
2	2010	4	18	148

ตารางที่ 3.2 ค่าข้อมูลที่จะนำมาใช้งาน(ต่อ)

No	year	month	day	pm2.5
3	2010	4	18	159
4	2010	4	18	N/A
5	2010	4	18	N/A
6	2010	4	18	N/A
7	2010	4	18	105
8	2010	4	18	124

จากตารางที่ 3.2 เมื่อทำการตัดคอลัมน์ที่ไม่ต้องการทิ้งไปแล้ว จะทำการรวมข้อมูลของ year month และ day เข้าด้วยกันและใช้ชื่อว่า Date

ตารางที่ 3.3 ตัวอย่างการรวมกลุ่มกันของ year month day

No	Date	pm2.5
1	2010/4/18	129
2	2010/4/18	148
3	2010/4/18	159
4	2010/4/18	N/A
5	2010/4/18	N/A
6	2010/4/18	N/A

โดยค่า PM 2.5 ที่ใช้ในการทดลองในครั้งนี้มีจำนวน 42960 ค่า ซึ่ง 42960 ค่านี้จะนำไปเฉลี่ย ตามวันที่จัดข้อมูลของ date ให้อยู่วันเดียวกันจนครบเป็นจำนวน 1790 วัน หรือจะหาค่าใน 1 วันได้โดยใช้สูตรต่อไปนี้

ตารางที่ 3.4 ตัวอย่างค่าจำนวน 24ชม.นำมาเฉลี่ย 1 วัน

pm2.5	129	148	159	181	138	109	105	124
	120	132	140	152	148	164	158	154
	159	164	170	149	154	164	156	126

$$x_0 = \frac{\text{ผลรวมของค่า} pm2.5}{\text{ตามข้อมูล} Date \text{วันเดียวกัน}} \qquad (13)$$

วิธีทำ

$$x_0 = \frac{129 + 148 + 159 + 181 + \cdots + 156 + 126}{24}$$

$$x_0 = \frac{3593}{24}$$

$$x_0 = 145.96$$

จากสมการด้านบนจะทำการหาค่าเฉลี่ยไปเรื่อยๆจนกว่าจะครบ และบางครั้งอาจจะมีค่าที่เป็น N/A จะไม่ได้สามารถหาค่าได้ดังนั้นเมื่อมีค่าใดค่านึงเป็น N/A โดยจะสมมติค่าและทำการคำนวณดังต่อไปนี้

ตารางที่ 3.5 ตัวอย่างค่าจำนวน 24ชม.นำมาเฉลี่ย 1 วัน ในกรณีที่ค่าที่หาไม่ครบจำนวน

pm2.5	129	148	159	N/A	N/A	N/A	N/A	N/A
	N/A	N/A	N/A	N/A	N/A	164	158	154
	159	164	N/A	N/A	N/A	N/A	156	126

$$x_0 = \frac{\text{ผลรวมของค่า} pm2.5}{\text{ตามข้อมูล} Date \text{วันเดียวกัน}}$$

วิธีทำ

$$x_0 = \frac{129 + 148 + 159 + 164 + 158 + 154 + 159 + 164 + 156 + 126}{10}$$

$$x_0 = \frac{1517}{10}$$

$$x_0 = 151.7$$

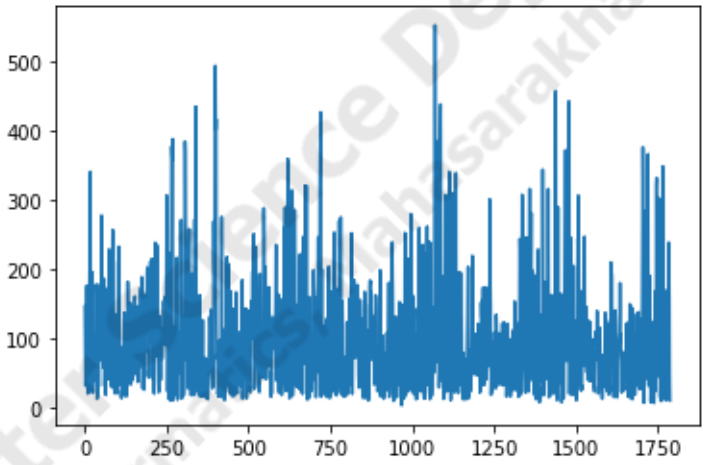
จากสูตรดังกล่าวจะเห็นว่าค่าที่สามารถนำมาคำนวณได้จะมีอยู่ 10 ค่าและค่าที่เป็น N/A จะมีอยู่ 14 ค่า ซึ่งเราจะไม่นำข้อมูล N/A มาคำนวณด้วย เพราะอาจทำให้ค่าเพี้ยนได้ เราจึงนำค่าที่เหลืออยู่มาคำนวณแทนและนับเฉพาะค่าที่เหลืออยู่เท่านั้น เมื่อครบแล้วจะได้ค่าเฉลี่ยที่ใช้ในการทดลองดังนี้

t	pm2.5
0	145.96
1	78.83
2	31.33
3	42.46
4	56.42
5	69
6	176.21
7	88.5
8	57.25
9	20

t	pm2.5
10	20.75
11	40.21
12	93.71
13	45.46
14	177.63
15	209.21
16	260.21
17	340.75
18	85.33
19	27.04

t	pm2.5
20	29.42
21	23.18
22	195.58
23	122.33
24	21.17
25	23.88
26	44.29
27	39.25
28	64.79
29	65.63

ภาพประกอบที่ 3.5 ตัวอย่างค่าเฉลี่ย pm 2.5 ในแต่ละวัน



ภาพประกอบที่ 3.6 กราฟค่าข้อมูลที่ทำการเฉลี่ยรายวันทั้งหมด 1790 ค่า

3.3.2 การจัดเตรียมข้อมูล Pre-Process

ขั้นตอนการเตรียมข้อมูล (Pre-Process) เป็นขั้นตอนเพื่อเตรียมข้อมูลก่อนเข้าประมวลผลใน Model โดยขั้นตอนการเตรียมข้อมูล จะเริ่มทำการ Data Normalization(การทำข้อมูลให้เป็นมาตรฐาน) และสามารถอธิบายได้ดังนี้

1. การแปลงขนาดชุดข้อมูลทดสอบและเรียนรู้ ก่อนหน้าให้อยู่ในช่วง 1 ถึง -1 ด้วยการทำData Normalization มาเพื่อเพิ่มค่าความแม่นยำที่มากขึ้นของโมเดล โดยใช้สูตรต่อไปนี

$$x' = 2 \times \left(\frac{x - x_{min}}{x_{max} - x_{min}} \right) - 1$$

โดย X คือค่าปัจจุบันของ Input X

$\mathbf{X_{min}}$ *คือ ค่าของ* Input X *ที่น้อยที่สุด* และค่าของที่น้อยที่สุดของ X คือ *2.96*

$\mathbf{X_{max}}$ *คือ ค่าของ* Input X *ที่มากที่สุด* และค่าที่มากที่สุดของ X คือ 552.48 กำหนดให้ X มีค่าเท่ากับ 145

<u>วิธีทำ</u>

$$x' = 2 \times \left(\frac{x - x_{min}}{x_{max} - x_{min}}\right) - \mathbf{1} \qquad (14)$$

$$x' = 2 \times (\frac{145 - 2.96}{552.48 - 2.96}) - 1$$

$$x' = 2 \times \left(\frac{143}{549.52}\right) - 1$$

$$x' = \ (2 \times 0.26) - 1$$

$$x' = \ 0.52 - 1$$

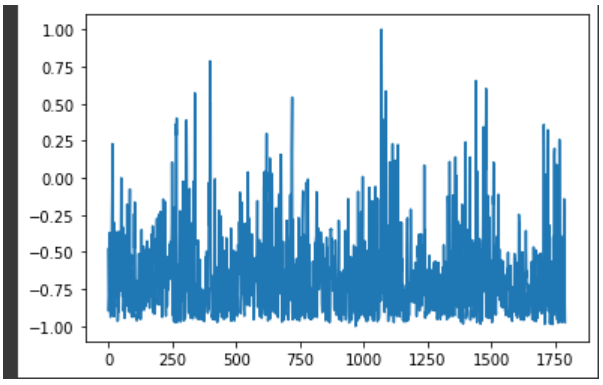
$$x' = -0.479$$

x	Pm2.5
0	-0.48
1	-0.72
2	-0.90
3	-0.86
4	-0.81
5	-0.76
6	-0.37
7	-0.69
8	-0.80
9	-0.94

x	Pm2.5
10	-0.94
11	-0.86
12	-0.67
13	-0.85
14	-0.36
15	-0.25
16	-0.06
17	0.23
18	-0.70
19	-0.91

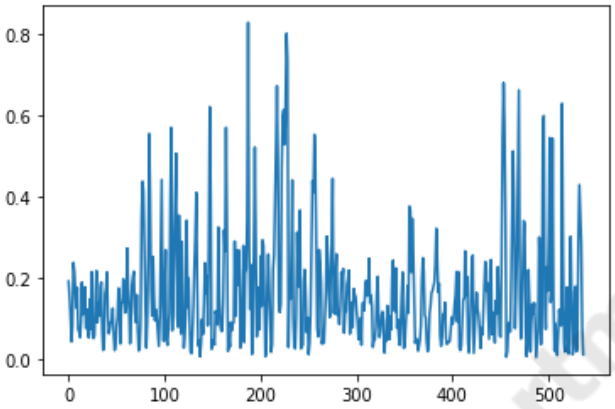
x	Pm2.5
20	-0.90
21	-0.93
22	-0.30
23	-0.57
24	-0.93
25	-0.92
26	-0.85
27	-0.87
28	-0.77
29	-0.77

ภาพประกอบที่ 3.7 ตัวอย่างข้อมูลหลังจากทำ Normalization

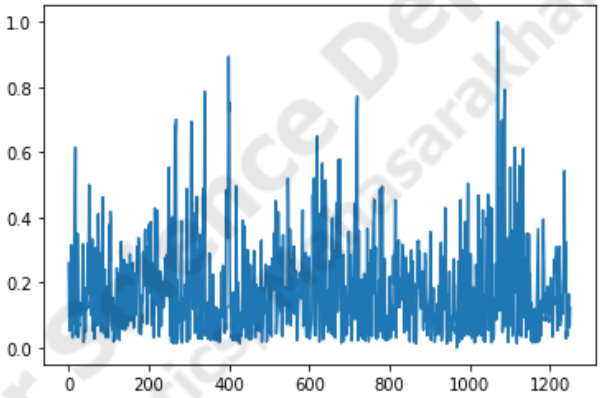


ภาพประกอบที่ 3.8 ตัวอย่างกราฟของชุดข้อมูลที่ผ่านการ normalization

2. การแบ่งชุดข้อมูลทอดสอบและเรียนรู้ จะทำหลังจากการทำ Normalization แล้ว โดยจะแบ่งเป็นชุดข้อมูลเรียนรู้ 70 เปอร์เซ็นต์ และชุดข้อมูลทดสอบ 30 เปอร์เซ็นต์



ภาพประกอบที่ 3.9 กราฟชุดข้อมูลทดสอบ



ภาพประกอบที่ 3.10 กราฟชุดข้อมูลเรียนรู้

จากภาพประกอบที่ 3.9 และ 3.10 จะแบ่งชุดข้อมูลทดสอบและเรียนรู้ได้จากค่าทั้งหมด 1790 ค่า ได้แก้ ข้อมูลชุดทดสอบ 30 เปอร์เซ็นต์จะมีข้อมูลในชุดข้อมูลทั้ง 537 ค่า และชุดข้อมูลเรียนรู้ 70 เปอร์เซ็นต์จะมีข้อมูลในชุดข้อมูลทั้งหมด 1252 ค่า

ก่อนจะนำข้อมูลเข้าสู่การพยากรณ์ล่วงหน้าด้วยโมเดลของ LSTM เราจะทำการหาค่าการพยากรณ์ของชุดข้อมูลเรียนรู้และทดสอบก่อน

```
def create_dataset(dataset, time_step=1):
  dataX, dataY = [], []
  for i in range(len(dataset)-time_step-1):
    a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----99   100
    dataX.append(a)
    dataY.append(dataset[i + time_step, 0])
  return numpy.array(dataX), numpy.array(dataY)
```

ภาพประกอบที่ 3.11 ตัวอย่างการแบ่งข้อมูลของชุดข้อมูลทดสอบและเรียนรู้

จากภาพประกอบที่ 3.11 เราจะทำการแปลงชุดข้อมูลให้อยู่ในรูปของ Dataset เพื่อทำการหาค่าพยากรณ์ของชุดข้อมูลทดสอบและเรียนรู้ โดยจะแบ่งข้อมูลที่ได้มาเป็นก่อนหน้ามาเป็น DataX และ DataY เพื่อจะนำค่าไปทำ Sliding Window

```
time_step = 5
X_train, y_train = create_dataset(train_data, time_step)
X_test, ytest = create_dataset(test_data, time_step)
print("show input  windows size = ",time_step)
for i in range(0,5):
    print("input[",i,"]", X_train[i],end="")
    print("\toutput[",i,"]",y_train[i])
print("show input  windows size = ",time_step)
for i in range(0,5):
    print("input[",i,"]",X_test[i],end="")
    print("\toutput[",i,"]",ytest[i])
```

ภาพประกอบที่ 3.12 ตัวอย่างการทำ Sliding Window ของชุดข้อมูลทดสอบและเรียนรู้

จากภาพประกอบที่ 3.12 time_step จะเป็นตัวบ่งบอกว่าจะนำกี่ค่ามาเพื่อคำนวณหาค่าถัดไป จากนั้น X_train,Y_train เป็นชุดข้อมูลเรียนรู้ที่ได้จากการทำ Dataset และ X_test,Y_test ก็เหมือนกันที่จะนำชุดข้อมูลทดสอบไปทำ Dataset แบ่งค่า X,Y จากนั้นจึงทำการแสดงค่า โดยมีการทำ Sliding Window ในการหาค่าของชุดข้อมูลทดสอบและเรียนรู้

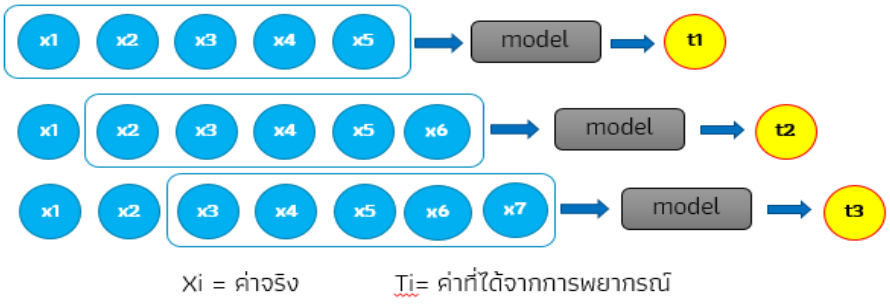
```
show input  windows size =  5
input[ 0 ] [0.26022711 0.13806595 0.05162687 0.07188091 0.0972849 ]     output[ 0 ] 0.12017760955015286
input[ 1 ] [0.13806595 0.05162687 0.07188091 0.0972849  0.12017761]     output[ 1 ] 0.3152751492211385
input[ 2 ] [0.05162687 0.07188091 0.0972849  0.12017761 0.31527515]     output[ 2 ] 0.1556631241811035
input[ 3 ] [0.07188091 0.0972849  0.12017761 0.31527515 0.15566312]     output[ 3 ] 0.09879531227252876
input[ 4 ] [0.0972849  0.12017761 0.31527515 0.15566312 0.09879531]     output[ 4 ] 0.03100888047750764
```

ภาพประกอบที่ 3.13 ตัวอย่างการ Sliding Window ของชุดข้อมูลเรียนรู้

```
show input  windows size =  5
input[ 0 ] [0.19258626 0.1550626  0.11002329 0.04443878 0.08283593]     output[ 0 ] 0.23860823991847435
input[ 1 ] [0.1550626  0.11002329 0.04443878 0.08283593 0.23860824]     output[ 1 ] 0.21753530353763287
input[ 2 ] [0.11002329 0.04443878 0.08283593 0.23860824 0.2175353 ]     output[ 2 ] 0.17484349978162764
input[ 3 ] [0.04443878 0.08283593 0.23860824 0.2175353  0.1748435 ]     output[ 3 ] 0.1272383170767215
input[ 4 ] [0.08283593 0.23860824 0.2175353  0.1748435  0.12723832]     output[ 4 ] 0.17679065366137722
```

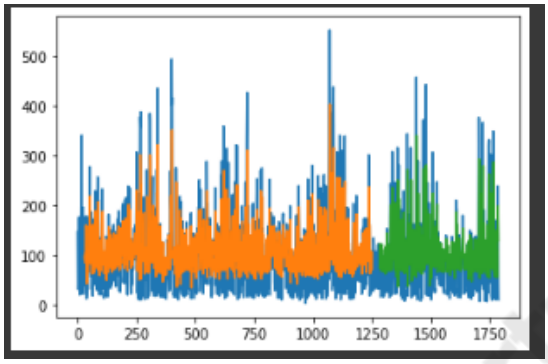
ภาพประกอบที่ 3.14 ตัวอย่างการ Sliding Window ของชุดข้อมูลทดสอบ

จากภาพประกอบที่ 3.13 และ 3.14 จะเป็นค่าจริง 5 ค่า พยากรณ์ค่าล่าสุด 1 ค่า โดยค่าล่าสุดจะทำการมาแทนที่ค่าถัดไป จากภาพดังต่อไปนี้



ภาพประกอบที่ 3.15 ตัวอย่างการ Sliding Window ของการหาค่า Dataset

จากภาพประกอบที่ 3.15 เป็นตัวอย่างของการ Sliding Window ซึงการสไลด์แบบนี้จะเป็นการสไลด์แบบ Integrate โดยการสไลด์แบบนี้จะเป็นการนำค่าที่ได้มานั้นมาพยากรณ์ต่อข้างหลังโดยมีค่าจริงคอยรอบรับเสมอ



ภาพประกอบที่ 3.16 กราฟการพยากรณ์ของชุดข้อมูลเรียนรู้และทดสอบ

จากภาพประกอบที่ 3.16 กราฟจะแสดงข้อมูลดังนี้

สีฟ้า แสดงข้อมูลทั้งหมด

สีส้ม แสดงข้อมูลค่าที่ถูกพยากรณ์ด้วยชุดข้อมูลเรียนรู้

สีเขียว แสดงข้อมูลที่ถูกพยากรณ์ด้วยชุดข้อมูลทดสอบ

3.3.3 การพยากรณ์ด้วยโครงข่ายประสาทเทียม LSTM

หลังจากเตรียมข้อมูลและแยกชุดข้อมูลทอดสอบและเรียนรู้จะเริ่มทำการพยากรณ์โดยนำชุดข้อมูลเรียนรู้ไปเพิ่มการเรียนรู้ของโมเดล หลังจากนั้นจะทำการนำข้อมูลชุดทดสอบไปพยากรณ์กับโมเดลที่ได้ทำการเรียนรู้ หลังจากนั้นจะแสดง output ออกมา

1.การเรียกใช้โมเดล LSTM โดยการใช้ Sequential() เพื่อที่จะทำการ add layer ไปแต่ละเรื่อยๆ แบบตรงไปตรงมา

```
model=Sequential()
model.add(LSTM(50,return_sequences=True,input_shape=(30,1)))
model.add(LSTM(50,return_sequences=True))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(loss='mean_squared_error',optimizer='adam')
```

ภาพประกอบที่ 3.17 ตัวอย่างโค้ดในการ สร้างโมเดลในแต่ละ Layer

จากภาพประกอบที่ 3.17 จะเริ่มสร้าง Sequential ก่อนเป็นอย่างแรก หลังจากนั้นจะทำการสร้าง Layer แต่ละชั้น โดย Layer แรกที่จะทำการสร้างคือ Layer สำหรับ input และ โดยกำหนดให้ค่าที่จะทำมาพยากรณ์นั้นเป็น 30 ค่าและ output ที่ตั้งการ 1 ค่า ค่า return_sequences

จะเป็นตัวที่คอย set ค่า Defualt ให้กับโมเดล สุดท้ายค่า 10 นั้นจะเป็นจำนวนของ Hidden Node ที่ใช้ในการพยากรณ์ สำหรับ Dense คือ Fully Connected Layer หรือเชื่อมต่อกันหมด ก็คือทั้ง 10 ตัว ก็จะเชื่อมกัน 10 ตัวในชั้นถัดไป หลังจาก add จนเสร็จก็จะ compile model โดย loss function ที่ใช้ mean squared error(MSE) และ optimizer ที่ชื่อว่า adam และเราสามารถโชว์ว่าโมเดลแต่ละ Layer เป็นอย่างไรโดยใช้คำสั่ง model.sumary()

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 30, 50)            10400
_________________________________________________________________
lstm_1 (LSTM)                (None, 30, 50)            20200
_________________________________________________________________
lstm_2 (LSTM)                (None, 50)                20200
_________________________________________________________________
dense (Dense)                (None, 1)                 51
=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
```

ภาพประกอบที่ 3.18 ตัวอย่างของโมเดล LSTM ในแต่ละ Layer

```
model.fit(X_train,y_train,validation_data=(X_test,ytest),epochs=100,batch_size=64,verbose=1)
```

ภาพประกอบที่ 3.19 ตัวอย่างโค้ดในการเรียกใช้ Model

จากภาพประกอบที่ 3.19 หลังจากสร้างโมเดล LSTM แล้วนั้น ต่อไปจะเป็นการกำหนดค่าเบื้องต้นของ Model โดยมีพารามิเตอร์ต่อไปนี้

X_train, Y_train ค่าที่เก็บข้อมูลเรียนรู้ไว้หลังจากแปลงอาเรย์ให้อยู่ในรูปแบบ 2 มิติ

Validation_data เป็นพารามิเคอร์ที่เก็บค่า X และ Y ของข้อมูลทดสอบไว้

Epoch เป็นค่าที่ใช้สำหรับการวนซ้ำของโมเดล

Batch_size เป็นพารามิเตอร์ที่กำหนดการคาดการของการพยากรณ์

Verbose เป็นค่าที่จะแสดงการทำงานค่า epoch ในแต่ละรอบ

```
Epoch 1/100
20/20 [==============================] - 6s 97ms/step - loss: 0.0218 - val_loss: 0.0204
Epoch 2/100
20/20 [==============================] - 1s 52ms/step - loss: 0.0200 - val_loss: 0.0208
Epoch 3/100
20/20 [==============================] - 1s 52ms/step - loss: 0.0202 - val_loss: 0.0204
Epoch 4/100
20/20 [==============================] - 1s 50ms/step - loss: 0.0201 - val_loss: 0.0216
Epoch 5/100
20/20 [==============================] - 1s 50ms/step - loss: 0.0204 - val_loss: 0.0204
Epoch 6/100
20/20 [==============================] - 1s 51ms/step - loss: 0.0198 - val_loss: 0.0204
Epoch 7/100
20/20 [==============================] - 1s 52ms/step - loss: 0.0197 - val_loss: 0.0202
Epoch 8/100
20/20 [==============================] - 1s 51ms/step - loss: 0.0197 - val_loss: 0.0202
Epoch 9/100
20/20 [==============================] - 1s 52ms/step - loss: 0.0197 - val_loss: 0.0201
Epoch 10/100
20/20 [==============================] - 1s 51ms/step - loss: 0.0196 - val_loss: 0.0203
```

ภาพประกอบที่ 3.20 ตัวอย่าง Epoch ในการรันแต่ละรอบ

จากภาพประกอบที่ 3.20 ค่า Epoch ที่ใช้คือ 100 และแต่ละการวนซ้ำของ Epoch จะแสดงเวลาและค่า Loss Function ที่เกิดในช่วงเวลานั้นๆ

5. หลังจากเราลองทำการทดลองกับค่าที่มีอยู่จริงแล้วนั้นต่อไปเราจะพยากรณ์ล่วงหน้าโดยพยากรณ์ข้อมูลที่ไม่มีอยู่จริง โดยใช้ input 30 ค่า และจะทำการ sliding window เพื่อหาค่าล่วงหน้า 7 วัน

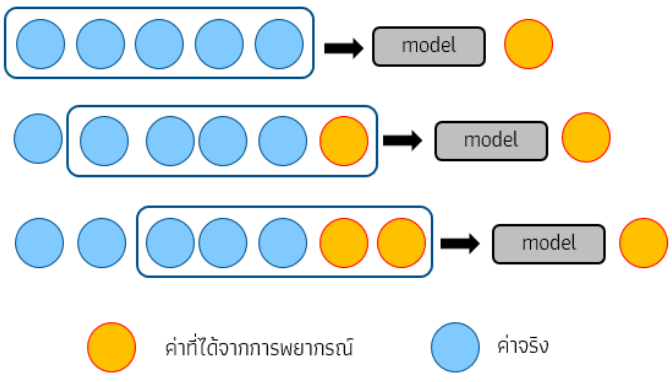
```
#print(temp_input)
x_input=np.array(temp_input[1:])
print("{} day input {}".format(i,x_input))
x_input=x_input.reshape(1,-1)
x_input = x_input.reshape((1, n_steps, 1))
#print(x_input)
yhat = model.predict(x_input, verbose=0)
print("{} day output {}".format(i,yhat))
temp_input.extend(yhat[0].tolist())
temp_input=temp_input[1:]
#print(temp_input)
lst_output.extend(yhat.tolist())
```

ภาพประกอบที่ 3.21 ตัวอย่างโค้ดการพยากรณ์ล่วงหน้า 7 วันล่าสุด

จากภาพประกอบที่ 3.21 นำค่า X_input ที่ได้จากอาเรย์ temp_input เราจะนำค่านี้มากทำการพยากรณ์ล่วงหน้า 1 หน่วยเวลา เป็นเวลา 7 วัน หลังจากได้ค่าที่ต้องนำไปพยากรณ์แล้วจะนำค่าดังกล่าวไปทำการแปลงอาเรย์เพื่อนำไปเข้าโมเดล หลังจากนั้นจะเก็บไค่าพยากรณ์ที่ได้ไว้ใน yhat และจะนำค่านั้นไปต่อท้ายค่านำเข้าเพื่อทำ sliding window ในการหาค่าต่อไป

```
x_input = x_input.reshape((1, n_steps,1))
yhat = model.predict(x_input, verbose=0)
print(yhat[0])
temp_input.extend(yhat[0].tolist())
print(len(temp_input))
lst_output.extend(yhat.tolist())
```

ภาพประกอบที่ 3.22 ตัวอย่างโค้ดในการนำ Output ที่ได้มาทำ Sliding Window



ภาพประกอบที่ 3.23 ตัวอย่างการเลื่อนข้อมูล หรือ Sliding Window

จากภาพประกอบที่ 3.23 เป็นตัวอย่างการเลื่อนข้อมูลหรือ Sliding Window โดยจากภาพจะเห็นว่า output ที่ได้มานั้นจะถูกนำมาคิดค่าพยากรณ์ต่อทันที เมื่อจะหาค่าถัดไปและค่าที่อยู่แรกสุดจะถูกผลักออกทันที

```
1 day input [0.04079924 0.01137356 0.06516596 0.12298006 0.09515577 0.08418256
 0.62887611 0.2547678  0.03972558 0.01956253 0.07937837 0.17795531
 0.15611807 0.01486752 0.17242321 0.30260955 0.13595502 0.01330252
 0.02827923 0.16794657 0.18001165 0.02032683 0.04374727 0.30442932
 0.42893798 0.35379968 0.28395691 0.07846848 0.01288397 0.10032599]
1 day output [[0.14152917]]
2 day input [0.01137356 0.06516596 0.12298006 0.09515577 0.08418256 0.62887611
 0.2547678  0.03972558 0.01956253 0.07937837 0.17795531 0.15611807
 0.01486752 0.17242321 0.30260955 0.13595502 0.01330252 0.02827923
 0.16794657 0.18001165 0.02032683 0.04374727 0.30442932 0.42893798
 0.35379968 0.28395691 0.07846848 0.01288397 0.10032599 0.14152917]
2 day output [[0.15377517]]
```

ภาพประกอบที่ 3.24 ตัวอย่างค่า input และ output ที่ทำ Sliding Window

4. หลังจากหาค่าพยากรณ์ได้แล้ว แต่ข้อมูลที่พยากรณ์ออกมายังไม่ได้กลับไปอยู่ในค่าที่สามารถวัดประสิทธิภาพของโมเดลได้ ดังนั้นจะทำการ Denormalize

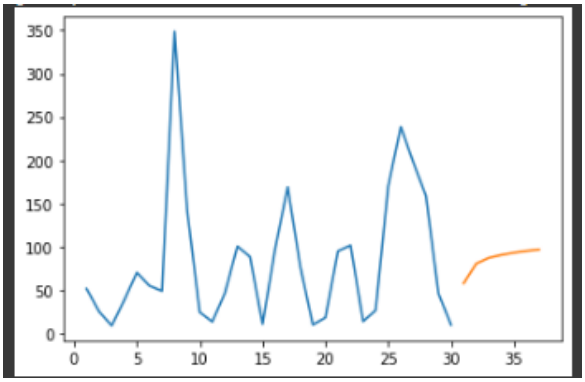
```
pre =scaler.inverse_transform(lst_output).tolist()
actual = scaler.inverse_transform(df1).tolist()
```

ภาพประกอบที่ 3.25 ตัวอย่างโค้ดในการทำ Denormalize

```
day [1] [58.091140329241746]
day [2] [80.73311095952987]
day [3] [87.46253166317939]
day [4] [90.82444564700126]
day [5] [93.38875195384026]
day [6] [95.48538106560707]
day [7] [97.06665142774581]
```

ภาพประกอบที่ 3.26 ตัวอย่าง output หลังทำ Denormalize

จากภาพประกอบที่ 3.26 ค่าที่ได้จากการพยากรณ์นั้นจะยังไม่สามารถนำไปใช้ได้จึงต้องมาทำการ Denormalize เพื่อกลับเป็นค่าเดิมตามภาพประกอบที่ 3.27



ภาพประกอบที่ 3.27 กราฟแสดงข้อมูล output ที่พยากรณ์ล่วงหน้า 7 วัน

จากภาพประกอบที่ 3.27 เป็นกราฟที่แสดงข้อมูลผลลัพธ์ โดยเส้นสีฟ้าจะเป็นค่าจริงที่ทำการพยากรณ์ และเส้นสีแดงเป็นค่าที่ได้จากการพยากรณ์ล่วงหน้า 7 วัน

3.4 การวัดประสิทธิภาพของ model

หลังจากได้ค่าการพยากรณ์ด้วยโครงข่ายประสาทเทียมแล้ว จะทำการวัดประสิทธิภาพโมดลที่ได้ทำการพยากรณ์นั้นสามารถเชื่อถือได้มากน้อยเพียงใด

1. Root Mean Square Error (RMSE)

$$RMSE = \sqrt{(\frac{1}{N}\sum_{i=1}^{N}(T_i - F_i)^2)}$$

โดยที่ N = จำนวนข้อมูล

T = ข้อมูลเป้าหมาย

F = ข้อมูลพยากรณ์

และกำหนดให้ i = 1

<u>วิธีทำ</u>

$$RMSE = \sqrt{\frac{1}{506}\left((79.42 - 122.29) + (62.13 - 72.89) + (100.92 - 238.67) + \cdots + 10.04 - 129.48\right)^2}$$

$$RMSE = \sqrt{\frac{1}{506}(-42.87)^2 + (-10.76)^2 + (-2.35)^2 + \cdots + (-119.44)^2}$$

$$RMSE = \sqrt{\frac{1}{506}\,1838.53 + 115.87 + 5.54 + \cdots + 14265.98}$$

$$RMSE = \sqrt{\frac{1}{506}\,(37815.99)}$$

$$RMSE = \sqrt{74.73}$$

$$RMSE = 8.64$$

Root Mean Square Error (RMSE) มีคุณสมบัติที่คล้ายกับ MSE แต่มีเพิ่มเติมตรงที่จะนำ MSE มาใส่ Square Root ซึ่งอาจทำให้ตีความง่ายกว่าเนื่องจากหน่วยของค่า Error จะไม่มีเลขยกกำลัง 2 เหมือนกับ MSE

2. Mean Square Error (MSE)

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\frac{(T_i - F_i)^2}{100}$$

โดยที่ N = จำนวนข้อมูลทั้งหมด

T = ข้อมูลเป้าหมาย

F = ข้อมูลพยากรณ์

และกำหนดให้ i = 1

<u>วิธีทำ</u>

$$MSE = \frac{1}{506}\{\frac{\big((79.42 - 122.29) + (62.13 - 72.89) + (100.92 - 238.67) + \cdots + 10.04 - 129.48\big)^2}{100}\}$$

$$MSE = \frac{1}{506}\{\frac{(-42.87)^2 + (-10.76)^2 + (-2.35)^2 + \cdots + (-119.44)^2}{100}\}$$

$$MSE = \frac{1}{506}\,(\frac{1838.53 + 115.87 + 5.54 + \cdots + 14265.98}{100})$$

$$MSE = \frac{1}{506}\,(37815.99)$$

$$MSE = 74.73$$

Mean Square Error (MSE) ค่านี้จะ Sensitive กับ Outlier มากเนื่องจากมีการนำค่า Error มายกกำลังสอง จึงควรระวังในการใช้งานหากข้อมูลมี Outlier เยอะ

3. Mean Absolute Percentage Error (MAPE)

$$MAPE = \frac{1}{n}\sum_{i=1}^{N}\left|\frac{T_i - F_i}{T_i}\right| \times 100$$

โดยที่ N = จำนวนข้อมูล

T = ข้อมูลเป้าหมาย

F = ข้อมูลพยากรณ์.

และกำหนดให้ i = 1

<u>วิธีทำ</u>

$$MAPE = \frac{1}{506}\left(\left|\left(\frac{(79.42 - 122.29)}{79.42}\right)\right| \times 100\right) + \left(\left|(\frac{(62.13 - 72.89)}{62.13})\right| \times 100\right) + \cdots + \left(\left|(\frac{10.04 - 129.48}{10.04})\right| \times 100\right)$$

$$MAPE = \frac{1}{506}\,((-0.53) + (-0.17) + (-0.02) + \cdots + (-11.89))$$

$$MAPE = \frac{1}{506}(797.29)$$

$$MAPE = 1.57$$

Mean Absolute Percentage Error (MAPE) เป็นค่าที่มีความแม่นยำโดยคำนวณเปอร์เซ็นต์ในการพยากรณ์ โดยไม่คำนึงถึงเครื่องหมาย ค่าที่ได้ต่ำ มีความแม่นยำสูง

4. Absolute Percentage Error (APE)

$$APE = \left(\left|\frac{T_i - F_i}{T_i}\right| * 100\right) \quad (15)$$

โดยที่ N = จำนวนข้อมูล

T = ข้อมูลเป้าหมาย

F = ข้อมูลพยากรณ์.

และกำหนดให้ i = 1

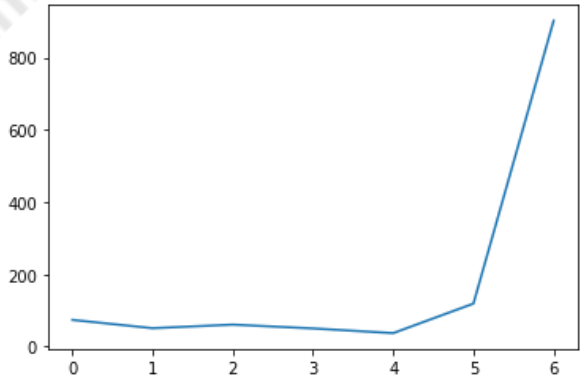
$$APE = \left(\left|\frac{27 - 43.31}{27}\right| * 100\right)$$

$$APE = \left(\left|\frac{-16.31}{27}\right| * 100\right)$$

$$APE = (0.604 * 100)$$

$$APE = 60.4$$

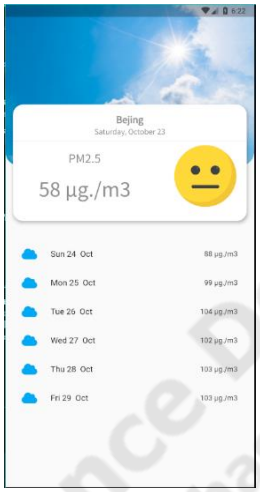
Absolute Percentage Error (APE) เป็นค่าที่คำนวณเปอร์เซ็นต์ในการพยากรณ์ โดยไม่คำนึงถึงเครื่องหมายใดๆทั้งสิ้น และสามารถทำออกมาเป็นกราฟได้



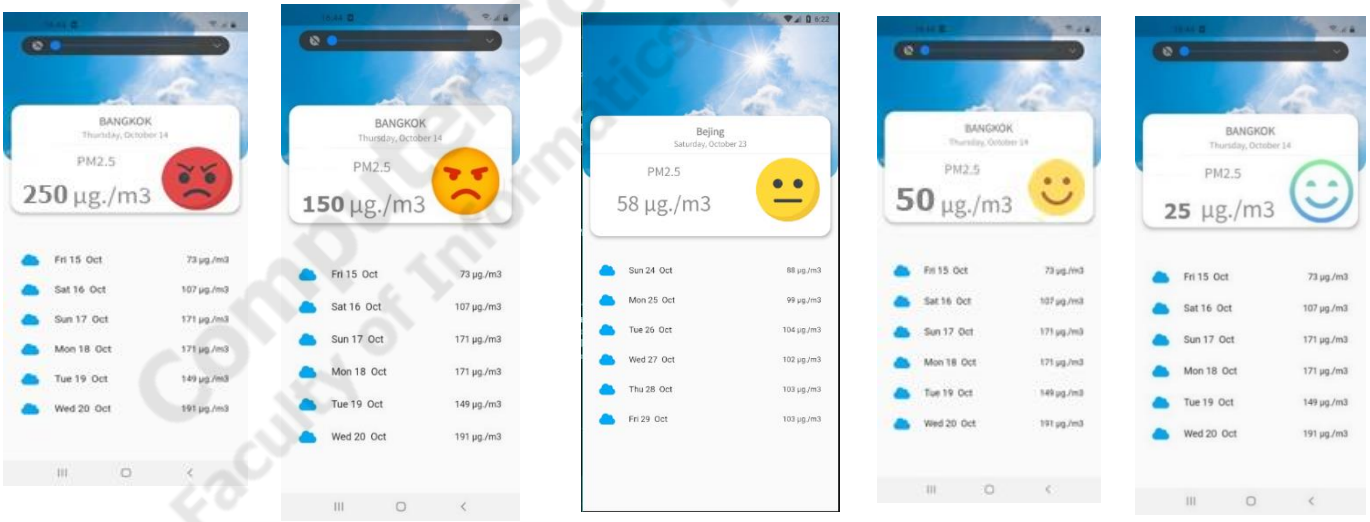
ภาพประกอบที่ 3.28 กราฟตัวอย่างของ APE

3.5 ผลลัพธ์

เมื่อทำการพยากรณ์และการวิเคราะห์ทางสถิติแล้วนั้นจะเป็นการเชื่อมข้อมูลที่ได้เชื่อมกับ Flask ที่เป็น Framework ที่ใช้ในการสร้าง Web Application ที่นำข้อมูลการพยากรณ์ขึ้นมานำเสนอในรูปแบบแอปพลิเคชัน โดยใช้ JSON ช่วย โดยในตอนนี้แอปพลิเคชันจะแสดงข้อมูลของค่า PM 2.5 ใน 7 วัน



ภาพประกอบที่ 3.29 ตัวอย่างของแอปพลิเคชั้นที่ใช้ Flask ในการสร้าง



ภาพประกอบที่ 3.30 ภาพตัวอย่างของแอปพลิเคชัน

จากภาพประกอบที่ 3.30 จะเป็นตัวอย่างแอปพลิเคชันโดยแต่ละรูปจะแสดงถึงค่าความเข้มข้นของค่าย pm 2.5 แตกต่างกัน โดยจำแนกจากซ้ายไปขวาดังนี้

หน้าโกรธสีแดง จะเป็นไอคอนที่เสดงเมื่อค่า pm 2.5 มากกว่า 200 ขึ้นไป

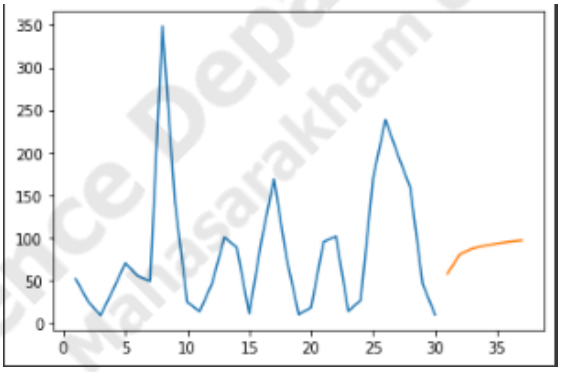
หน้าโกรธสีเหลือง จะเป็นไอคอนที่เสดงเมื่อค่า pm 2.5 มากกว่า 100 แต่ไม่เกิน 200

หน้านิ่ง จะเป็นไอคอนที่เสดงเมื่อค่า pm 2.5 มากกว่า 50 แต่ไม่เกิน 100
หน้ายิมสีเหลือง จะเป็นไอคอนที่เสดงเมื่อค่า pm 2.5 มากกว่า 25 แต่ไม่เกิน 50
หน้ายิ้มสีเขียว จะเป็นไอคอนที่เสดงเมื่อค่า pm 2.5 น้อยกว่า 25

3.6 สรุปผล

จากการทดลองข้อมูลที่ใช้ทั้งหมด 42960 ค่าและนำมาเฉลี่ยเป็นข้อมูลกายใน 1 วันเหลือ 1790 ค่า ทำให้การแบ่งชุดข้อมูล Train เป็น 70 เปอร์เซ็นต์ และชุดข้อมูล Test อีก 30 เปอร์เซ็นต์ ซึ่งเป็นข้อมูลจริงที่ใช้ Train จำวนว 1221 ค่าและชุดข้อมูลที่ใช้ test จำนวน 537 ค่า เมื่อนำชุดข้อมูล Train และ Test เข้าโมเดล LSTM เพื่อทำการพยากรณ์ค่าฝุ่นละอองขนาดเล็กล่วงหน้า หลังจากพยากรณ์เสร็จแล้วจะนำค่าที่พยากรณ์ออกมาได้ มาเปรียบเทียบกับค่าจริง ดังภาพต่อไปนี้

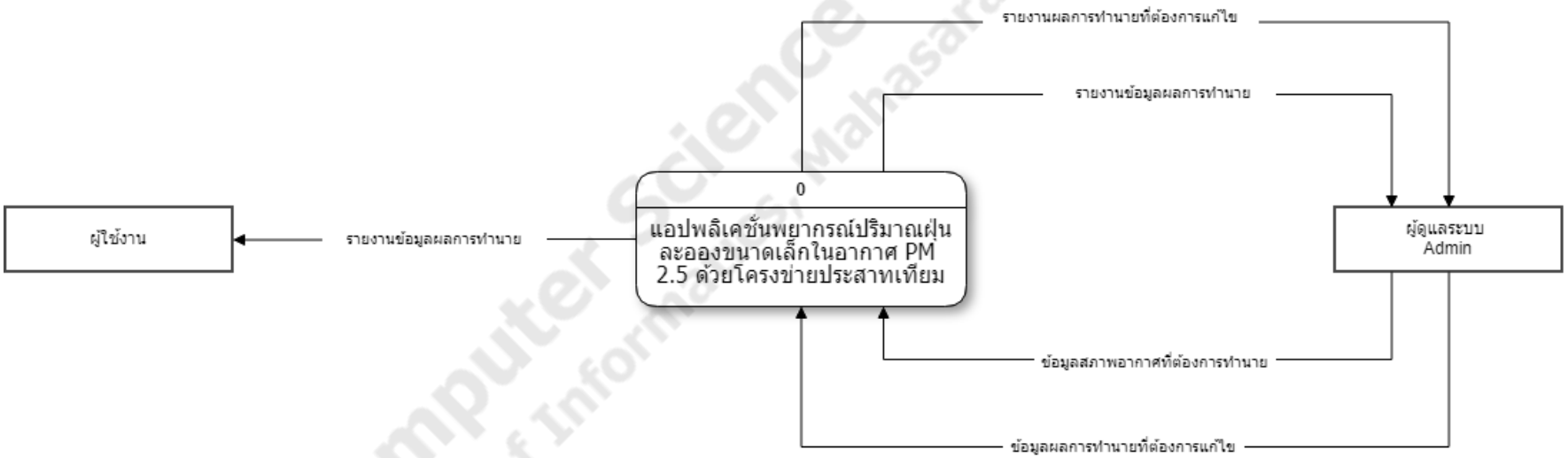
```
day [1] [58.091140329241746]
day [2] [80.73311095952987]
day [3] [87.46253166317939]
day [4] [90.82444564700126]
day [5] [93.38875195384026]
day [6] [95.48538106560707]
day [7] [97.06665142774581]
```



ภาพประกอบที่ 3.31 ตัวอย่างการเปรียบเทียบข้อมูลที่จริงและข้อมูลที่พยากรณ์ออกมา

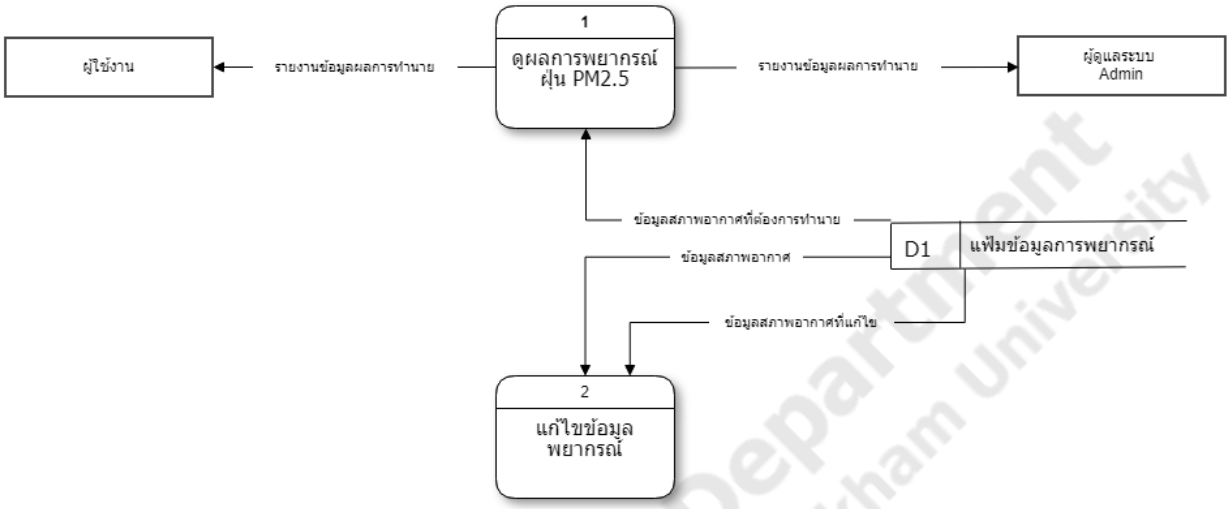
3.7 Context Diagram

3.7.1 Context Diagram



ภาพประกอบที่ 3.32 Context Diagram

3.7.2 Data Flow Diagram Level 1



ภาพประกอบที่ 3.33 Data Flow Diagram Level 1

3.7.3 External Entity Description

ตารางที่ 3.6 ตารางแสดง External Entity Description

Name	Description	Input Data Flow	Output Data Flow
ผู้ดูแลระบบ	ผู้ดูแลระบบ	- รายงานการทำนายผล - แก้ไขข้อมูลการพยากรณ์	- ข้อมูลการทำนายผล - ข้อมูลพยากรณ์ที่แก้ไข
ผู้ใช้งาน	ผู้ใช้งานสามารถเลือกช่วงเวลาในการทำนายผลได้	- รายงานการทำนายผล	- ข้อมูลการทำนายผล

3.8 Data Store Description

ตารางที่ 3.7 ตารางแสดง Data Store Description

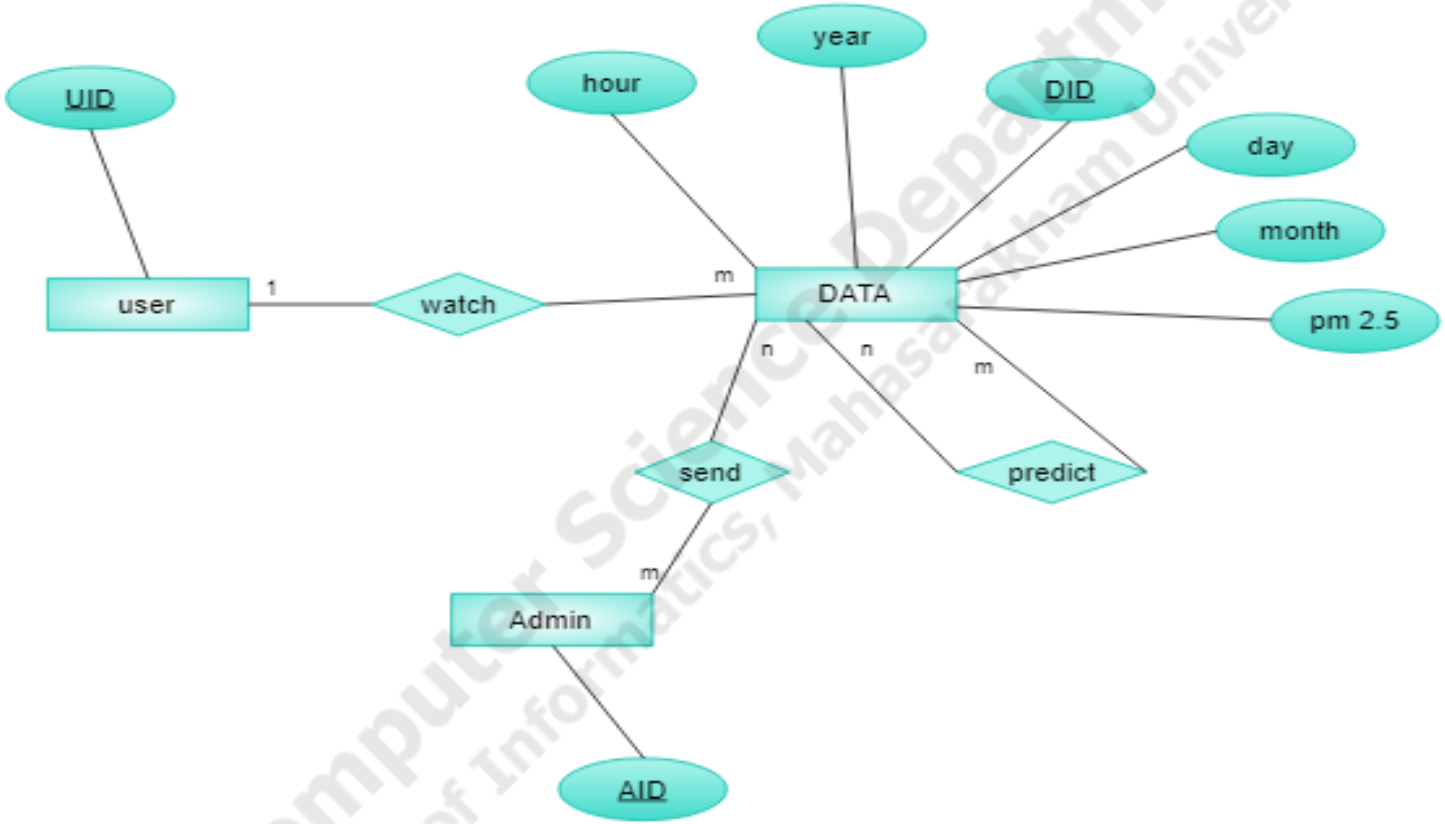
ID	Name	Description	Data Structure
D1	แฟ้มเก็บข้อมูลการพยากรณ์	เก็บข้อมูลผลลัพธ์การพยากรณ์	วัน + เดือน + ปี + ชั่วโมง + ผลการพยากรณ์

3.9 Data Structure Description

ตารางที่ 3.8 ตารางแสดง Data Structure Description

Name	Description	Source	Destination	Data Structure
รายงานข้อมูลการทำนาย	รายงานข้อมูลการทำนาย	Process 1 ดูผลการพยากรณ์ฝุ่น PM2.5	ผู้ดูแลระบบ ผู้ใช้งาน	รหัสการพยากรณ์ + วัน + เดือน + ปี + ชั่วโมง + ผลการพยากรณ์
ข้อมูลสภาพอากาศที่แก้ไข	ข้อมูลสภาพอากาศที่แก้ไข	Process 2 แก้ไขข้อมูลพยากรณ์	D1 แฟ้มข้อมูลการพยากรณ์	รหัสการพยากรณ์ + วัน + เดือน + ปี + ชั่วโมง + ผลการพยากรณ์
ข้อมูลสภาพอากาศที่ต้องการทำนาย	ข้อมูลสภาพอากาศที่ต้องการทำนาย	D1 แฟ้มข้อมูลการพยากรณ์	Process 1 ดูผลการพยากรณ์ฝุ่น PM2.5 Process 2 แก้ไขข้อมูลพยากรณ์	รหัสการพยากรณ์ + วัน + เดือน + ปี + ชั่วโมง + ผลการพยากรณ์

3.10 Entity Relationship Diagram



ภาพประกอบที่ 3.34 Entity Relationship Diagram

3.11 Data Table Description

ตารางที่ 3.9 ตารางแสดงข้อมูลผู้ใช้งาน

Attribute Name	Type	Size	Description	Key type	References
UID	Integer	10	รหัสผู้ใช้งาน	PK	U01

ตารางที่ 3.10 ตารางแสดงข้อมูลการทำนาย

Attribute Name	Type	Size	Description	Key type	References
DID	Integer	10	รหัสข้อมูล	PK	D01
Day	Integer	2	วัน	-	1
Month	Integer	2	เดือน	-	1
Year	Integer	5	ปี	-	2020
Hour	Integer	2	ชั่วโมง	-	5
PM 2.5	Integer	5	PM 2.5	-	207

ตารางที่ 3.11 ตารางแสดงข้อมูลของผู้ดูแล

Attribute Name	Type	Size	Description	Key type	References
AID	Integer	10	รหัสผู้ดูแล	PK	A1

บทที่ 4

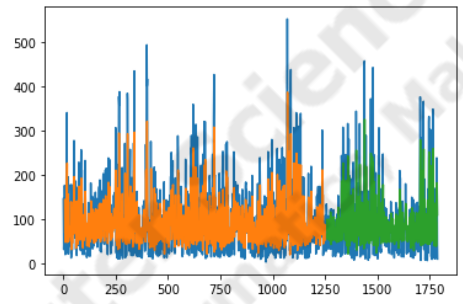
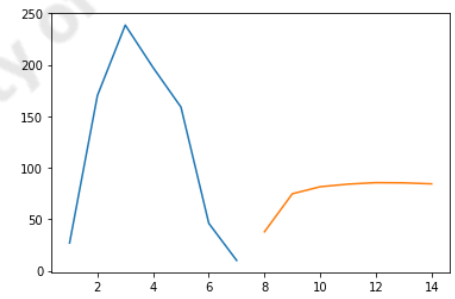
ผลการทดลอง

ในบทนี้จะประกอบไปด้วยตัวอย่างการวัดประสิทธิภาพของโมเดล โดยจะเปลี่ยนค่า Sliding Window ต่างๆดังนี้

4.1 การทดลองวัดประสิทธิภาพของโมเดล

โดยจะทดลองเปลี่ยนค่า Sliding Window ดังต่อไปนี้ 7 14 21 30 เพื่อเช็คว่าค่าที่เปลี่ยนของ Sliding Window จะเปลี่ยนค่าที่ได้จากการวัดประสิทธิภาพมากน้อยเพียงใด โดยจะทำทั้งการพยากรณ์ค่าที่ได้จากชุดข้อมูลทดสอบและเรียนรู้เปรียบเทียบกับข้อมูลจริงและค่าที่ทำการพยากรณ์ล่วงหน้า7วัน

ตารางที่ 4.1 การทดลองโดยให้ค่า Slide Window มีค่าเท่ากับ 7

<table>
<tr><th>ค่าที่ใช้</th><th>ตัวอย่างภาพ</th><th colspan="3">ค่าที่ได้</th></tr>
<tr><td rowspan="6">7</td><td colspan="4">กราฟข้อมูลชุดเรียนรู้และทดสอบ</td></tr>
<tr><td rowspan="2"> 
</td><td colspan="3">Train RSME : 103.15
Train MSE : 10638.9
Train MAPE : 12.24</td></tr>
<tr><td colspan="3">Test RSME : 103.15
Test MSE : 10202.79
Test MAPE : 11.9</td></tr>
<tr><td colspan="4">กราฟแสดงการพยากรณ์ล่วงหน้า 7 วัน</td></tr>
<tr><td rowspan="2"> 
</td><td>Day(yhat)</td><td>y</td><td>ape</td></tr>
<tr><td>37.88
74.79
81.6
84.15
85.6
85.37
84.45</td><td>27
170.25
238.67
197.38
159
46.08
10.04</td><td>40.31
56.06
65.8
57.36
46.16
85.26
741.1</td></tr>
</table>