

Project1 (Redo) PM2.5 Presentation

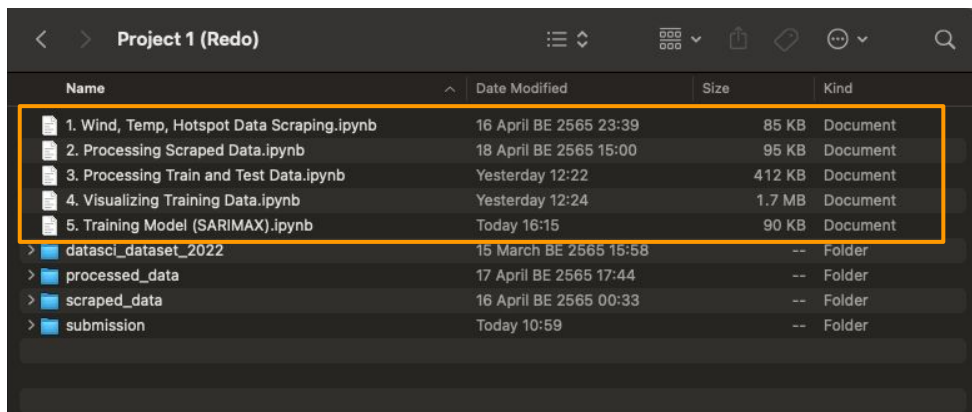
Group: ใจผมสลายชะมุง

Member

- | | | | |
|----|------------|------------|------------------|
| 1. | 6231340921 | Pongsapak | Pulthasthan |
| 2. | 6231343821 | Peeranath | Theerawatanachai |
| 3. | 6231367921 | Sahutsawat | Sivakosit |
| 4. | 6230054221 | Kharit | Tasanakowit |

รายละเอียดของสิ่งที่แก้ไข

1. จัดไฟล์ให้เป็นระเบียบมากขึ้น



IPython Notebook มีทั้งหมด 5 ไฟล์ ได้แก่

1. Wind, Temp, Hotspot Data Scraping.ipynb

การ scrape ข้อมูลต่าง ๆ

2. Processing Scraped Data.ipynb

การนำข้อมูลที่ scrape ได้มาทำให้มีความถี่ที่ต้องการ (6 ชม) และทำการ fill missing และ interpolation

3. Processing Train and Test Data.ipynb

การนำข้อมูลจาก dataset และจากการ scrape มารวมกัน โดยแยกข้อมูลออกเป็น 2 ส่วน (Train / Test)

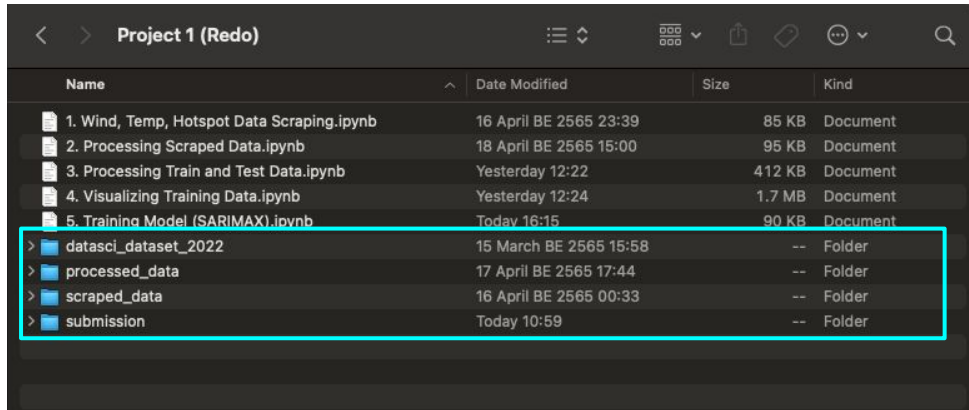
4. Visualizing Training Data.ipynb

การนำข้อมูล PM2.5 ที่ใช้ Train มาทำการ Visualize ร่วมกับข้อมูลอื่น ๆ

5. Training Model (SARIMAX).ipynb

การสร้างโมเดล SARIMAX เพื่อใช้ในการทำนาย และการปรับ Parameter ต่าง ๆ

1. จัดไฟล์ให้เป็นระเบียบมากขึ้น



Folder มีทั้งหมด 4 Folder ได้แก่

1. **datasci_dataset_2022**

โฟลเดอร์ข้อมูลที่โหลดมาจาก kaggle

2. **processed_data**

โฟลเดอร์ที่เก็บข้อมูลที่ได้จากการ scrape ที่ทำการประมวลผลแล้ว ร่วมกับ

ข้อมูลที่จะใช้ในการ Train / Test ของแต่ละจังหวัด

3. **scraped_data**

โฟลเดอร์เก็บข้อมูลดิบที่ได้จากการ scrape

4. **submission**

โฟลเดอร์ที่ใช้ในการเก็บไฟล์ csv ที่ได้จากการ forecast จาก SARIMAX ซึ่งถูก format ให้อยู่ในรูปแบบที่ใช้ส่งบน kaggle

2. ภายใน Python Notebook มีการเขียน comment และแยกเป็น function ให้เรียบร้อยขึ้น

```
3. Processing Train and Test Data.ipynb
File Edit View Insert Runtime Tools Help Last edited on April 20
+ Code + Text Connect

General Functions

def get_current_dt():
    current_dt = datetime.now()
    return current_dt.strftime('%Y-%m-%d %H:%M:%S')

def read_processed_data(base_processed_data_path, mode):
    """
    This function will read the processed data (csv) file with the given mode, and return a pandas DataFrame
    [Input]
    - base_processed_data_path: string containing folder of processed data path
    - mode: 'temp' / 'wind' / 'hotspot'
    [Output]
    - df: pandas DataFrame
    """
    df = pd.read_csv(f'{base_processed_data_path}/{mode}_processed_data.csv')

    df['date_time'] = pd.to_datetime(df['date_time'], format='%Y-%m-%d %H:%M:%S')
    df.sort_values(by='date_time', inplace=True)
    df.drop_duplicates(inplace=True)

    return df

def read_PM25_data(base_dataset_path, province, train = True):
    """
    This function will read the PM2.5 train/test data given in dataset with the corresponding province,
    and return a pandas DataFrame
    """
    path = f'{base_dataset_path}/{province}'
    if (train):
        path += f'/train/{province.lower()}_train.csv'
    else:
        path += f'/test/{province.lower()}_test.csv'

    df = pd.read_csv(path)
    df.rename(columns={df.columns[0]: 'date_time'}, inplace=True)
    df['date_time'] = pd.to_datetime(df['date_time'], format='%Y-%m-%d %H:%M:%S')
    df.sort_values(by='date_time', inplace=True)
    df.drop_duplicates(inplace=True)

    return df
```

▼ SARIMAX Model

▼ Functions involved

```
[ ] def initialize_SARIMAX_models(df_train, exog_columns, best_order, best_seasonal_order, target = 'PM2.5'):
    """
    [Input]
    - df_train = a pandas DataFrame of all training data
    - exog_columns = list of all exog variables name
    - best_order = (p, d, q)
    - best_seasonal_order = (P, D, Q, S)
    - target = PM2.5
    [Output]
    - target_result
    - exog_result_d
    """
    # create a model for target variable
    target_mod = SARIMAX(df_train[target],
                        exog=df_train[exog_columns],
                        order=best_order, seasonal_order=best_seasonal_order,
                        enforce_stationarity=False, enforce_invertibility=False)
    target_result = target_mod.fit(dis=0)

    # create models for exog variables (except for month_x (no need model to predict))
    exog_result_d = {}
    for exog_variable in exog_columns:
        if (exog_variable.startswith('month_')):
            continue
        exog_mod = SARIMAX(df_train[exog_variable],
                        order=best_order, seasonal_order=best_seasonal_order,
                        enforce_stationarity=False, enforce_invertibility=False)
        exog_result = exog_mod.fit(dis=0)
        exog_result_d[exog_variable] = exog_result

    return target_result, exog_result_d
```

3. ใช้การ padding ข้อมูล แทนการ interpolation แบบลากจุดต่อจุดบน test data

	x	y	z
0	1.0	2.0	NaN
1	NaN	NaN	4.0
2	5.0	NaN	NaN

df

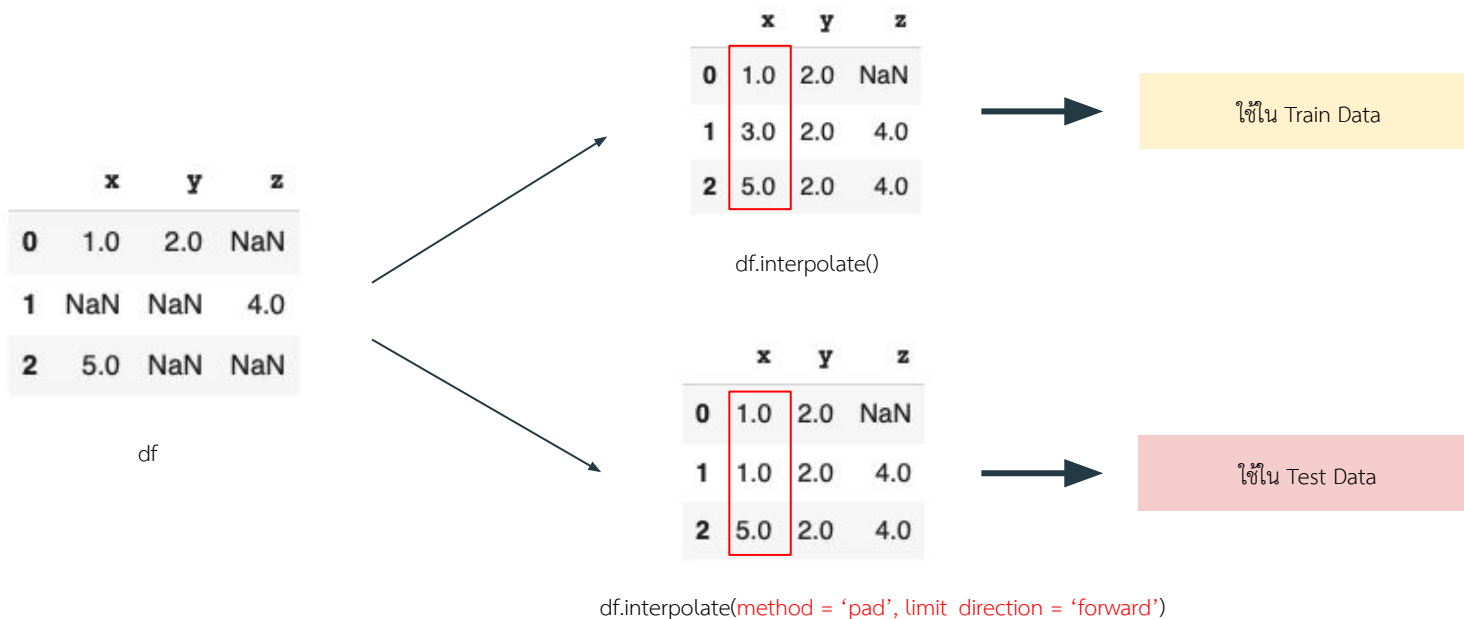
	x	y	z
0	1.0	2.0	NaN
1	3.0	2.0	4.0
2	5.0	2.0	4.0

df.interpolate()

	x	y	z
0	1.0	2.0	NaN
1	1.0	2.0	4.0
2	5.0	2.0	4.0

df.interpolate(method = 'pad', limit_direction = 'forward')

3. ใช้การ padding ข้อมูล แทนการ interpolation แบบลากจุดต่อจุดบน test data



3. ใช้การ padding ข้อมูล แทนการ interpolation แบบลากจุดต่อจุดบน test data

```
def process_wind_data(df_wind_train, df_wind_scraped, train_start_dt, train_end_dt, test_start_dt, test_end_dt, LOCS):
    # --> training data -----
    # change resolution to 1H
    train_daterange_1H = pd.date_range(start = train_start_dt, end = train_end_dt, freq = 'H').to_frame(name = 'date_time').reset_index(drop = True)
    df_wind_train = pd.merge(train_daterange_1H, df_wind_train, on = 'date_time', how = 'outer')

    # perform cyclic interpolation on wind_dir columns / simple linear interpolation on wind_speed columns
    for province in LOCS:
        df_wind_train[f'wind_dir_{province}'] = perform_cyclic_interpolation(df_wind_train[['date_time', f'wind_dir_{province}']])
        df_wind_train[f'wind_speed_{province}'] = df_wind_train[f'wind_speed_{province}'].interpolate().round(0)

    # change resolution to 6H
    df_wind_train = sample_df_freq(df_wind_train, start_dt = train_start_dt, end_dt = train_end_dt)

    # drop the first row (since there is no data)
    df_wind_train = df_wind_train.iloc[1:]
    df_wind_train.reset_index(inplace = True, drop = True)

    # --> testing data -----
    # change resolution to 1H
    test_daterange_1H = pd.date_range(start = test_start_dt, end = test_end_dt, freq = 'H').to_frame(name = 'date_time').reset_index(drop = True)
    df_wind_scraped = pd.merge(test_daterange_1H, df_wind_scraped, on = 'date_time', how = 'outer')

    # for wind_speed and wind_dir in test data. To prevent using future data, we fill the missing values with
    # the previous values only
    for col in df_wind_scraped.columns[1:]:
        df_wind_scraped[col] = df_wind_scraped[col].interpolate(method = 'pad', limit_direction = 'forward')

    # for the first entry that has missing value, we fill by using the latest values from training data
    df_wind_scraped.iloc[0, 1:] = df_wind_train.iloc[-1, 1:]

    # change resolution to 6H
    df_wind_scraped = sample_df_freq(df_wind_scraped, start_dt = test_start_dt, end_dt = test_end_dt)

    # -----
    all_wind_data = pd.concat([df_wind_train, df_wind_scraped], axis = 0)
    return all_wind_data
```

ข้อมูล wind direction และ wind speed

3. ใช้การ padding ข้อมูล แทนการ interpolation แบบลากจุดต่อจุดบน test data

```
def process_temp_data(df_temp_train, df_temp_scraped, train_start_dt, train_end_dt, test_start_dt, test_end_dt, LOCS):
    # --> training data -----
    # change resolution to 1H
    train_daterange_1H = pd.date_range(start = train_start_dt, end = train_end_dt, freq = 'H').to_frame(name = 'date_time').reset_index(drop = True)
    df_temp_train = pd.merge(train_daterange_1H, df_temp_train, on = 'date_time', how = 'outer')

    # perform simple linear interpolation on temp columns
    for col in df_temp_train.columns[1:]:
        df_temp_train[col] = df_temp_train[col].interpolate().round(1)

    # change resolution to 6H
    df_temp_train = sample_df_freq(df_temp_train, start_dt = train_start_dt, end_dt = train_end_dt)

    # drop the first row (since there is no data)
    df_temp_train = df_temp_train.iloc[1:]
    df_temp_train.reset_index(inplace = True, drop = True)

    # --> testing data -----
    # change resolution to 1H
    test_daterange_1H = pd.date_range(start = test_start_dt, end = test_end_dt, freq = 'H').to_frame(name = 'date_time').reset_index(drop = True)
    df_temp_scraped = pd.merge(test_daterange_1H, df_temp_scraped, on = 'date_time', how = 'outer')

    # to prevent using future data, fill the missing values with only its previous values (in test data)
    for col in df_temp_scraped.columns[1:]:
        df_temp_scraped[col] = df_temp_scraped[col].interpolate(method = 'pad', limit_direction = 'forward')

    # for the first entry that has missing value, we fill by using the latest values from training data
    df_temp_scraped.iloc[0, 1:] = df_temp_train.iloc[-1, 1:]

    # change resolution to 6H
    df_temp_scraped = sample_df_freq(df_temp_scraped, start_dt = test_start_dt, end_dt = test_end_dt)
    # -----

    all_temp_data = pd.concat([df_temp_train, df_temp_scraped], axis = 0)
    return all_temp_data
```

ข้อมูล temp surface

3. ใช้การ padding ข้อมูล แทนการ interpolation แบบลากจุดต่อจุดบน test data

```
def process_PM25_data(df_pm25_train, df_pm25_test, train_start_dt, train_end_dt, test_start_dt, test_end_dt):  
    # ensure that the resolution of both the train/test data is 1H  
    df_pm25_train = change_resolution_to1H(df_pm25_train, train_start_dt, train_end_dt)  
    df_pm25_test = change_resolution_to1H(df_pm25_test, test_start_dt, test_end_dt)  
  
    # for train data -> fill missing values by interpolation  
    df_pm25_train['PM2.5'] = df_pm25_train['PM2.5'].interpolate().round(0)  
  
    # for test data -> fill missing values with its previous value (avoid using future data)  
    df_pm25_test['PM2.5'] = df_pm25_test['PM2.5'].interpolate(method = 'pad', limit_direction = 'forward')  
  
    # change the resolution to 6H for further use to forecast  
    df_pm25_train = sample_df_freq(df_pm25_train, train_start_dt, train_end_dt, freq = '6h')  
    df_pm25_test = sample_df_freq(df_pm25_test, test_start_dt, test_end_dt, freq = '6h')  
  
    df_pm25_processed = pd.concat([df_pm25_train, df_pm25_test], axis = 0)  
    return df_pm25_processed
```

ข้อมูล PM2.5

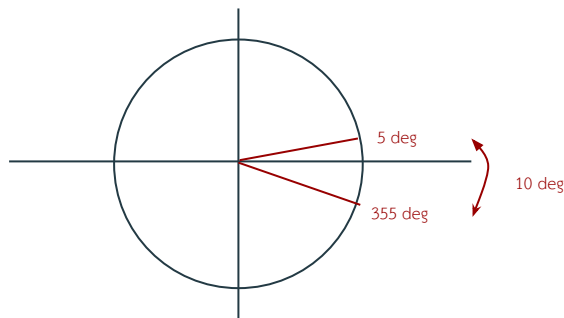
4. แก้ไขการ Processing ข้อมูล

4.1 ข้อมูล wind_direction

เนื่องจากข้อมูล wind_direction ที่ scrape มาได้ ซึ่งเป็นมุมของทิศทางลม (หน่วย: องศา) ซึ่งมีค่าอยู่ในช่วง $[0, 360)$

ดังนั้นในการทำ interpolation เติมข้อมูล wind_direction ที่ missing ไปใน train data เราจะมีการทำ interpolation แบบ cyclic ซึ่งจะมีการพิจารณาความใกล้/ไกลของมุมตามมุมของวงกลม

เช่น มุม 5 องศา กับ 355 องศา แท้จริงแล้วต่างกันอยู่ 10 องศา ไม่ใช่ 350 องศา



4. แก้ไขการ Processing ข้อมูล

4.1 ข้อมูล wind_direction

จากหลักการในหน้าที่แล้ว เราจึงทำ function ในการทำ interpolation สำหรับข้อมูล wind direction โดยเฉพาะ

```
def perform_cyclic_interpolation(df):  
    ...  
    This function computes cyclic interpolation on wind direction data  
    [Input]  
    df - a pandas DataFrame which contain 2 columns ['date_time', 'wind_dir']  
    [Output]  
    a pandas Series contain values calculated by cyclic interpolation on wind_data  
    ...  
  
    df = df.copy()  
    df.columns.values[1] = 'wind_dir'  
  
    df2 = df.loc[df['wind_dir'].notna(), ['date_time', 'wind_dir']].copy()  
    df2.rename(columns = {'wind_dir': 'wind_dir_2'}, inplace = True)  
    df2['wind_dir_2'] = np.rad2deg(np.unwrap(np.deg2rad(df2['wind_dir_2'])))  
  
    df3 = pd.merge(df, df2, on = 'date_time', how = 'left')  
    df3['wind_dir_2'] = df3['wind_dir_2'].interpolate().round(0)  
    df3['wind_dir_2'] %= 360  
    df3.drop(columns = 'wind_dir', inplace = True)  
    df3.rename(columns = {'wind_dir_2': 'wind_dir'}, inplace = True)  
  
    return df3['wind_dir']
```

	date_time	wind_dir	wind_dir_simple_interpolated	wind_dir_cyclic_interpolated
0	2020-07-01 00:00:00	5.0	5.0	5.0
1	2020-07-01 06:00:00	NaN	175.0	355.0
2	2020-07-01 12:00:00	345.0	345.0	345.0
3	2020-07-01 18:00:00	NaN	197.5	18.0
4	2020-07-02 00:00:00	50.0	50.0	50.0

ตัวอย่างข้อมูล
wind_dir ที่มี
Missing values

ผลลัพธ์ที่ได้จากการทำ
Interpolation ธรรมดา

ผลลัพธ์ที่ได้จากการทำ
Interpolation แบบ cyclic

4. แก้ไขการ Processing ข้อมูล

4.1 ข้อมูล wind_direction

จากหลักการในหน้าที่แล้ว เราจึงทำ function ในการทำ interpolation สำหรับข้อมูล wind direction โดยเฉพาะ

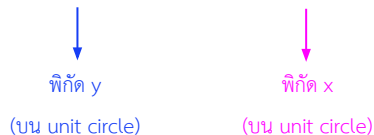
```
def perform_cyclic_interpolation(df):  
    ...  
    This function computes cyclic interpolation on wind direction data  
    [Input]  
    df - a pandas DataFrame which contain 2 columns ['date_time', 'wind_dir']  
    [Output]  
    a pandas Series contain values calculated by cyclic interpolation on wind_data  
    ...  
  
    df = df.copy()  
    df.columns.values[1] = 'wind_dir'  
  
    df2 = df.loc[df['wind_dir'].notna(), ['date_time', 'wind_dir']].copy()  
    df2.rename(columns = {'wind_dir': 'wind_dir_2'}, inplace = True)  
    df2['wind_dir_2'] = np.rad2deg(np.unwrap(np.deg2rad(df2['wind_dir_2'])))  
  
    df3 = pd.merge(df, df2, on = 'date_time', how = 'left')  
    df3['wind_dir_2'] = df3['wind_dir_2'].interpolate().round(0)  
    df3['wind_dir_2'] %= 360  
    df3.drop(columns = 'wind_dir', inplace = True)  
    df3.rename(columns = {'wind_dir_2': 'wind_dir'}, inplace = True)  
  
    return df3['wind_dir']
```

	date_time	wind_dir	wind_dir_simple_interpolated	wind_dir_cyclic_interpolated
0	2020-07-01 00:00:00	5.0	5.0	5.0
1	2020-07-01 06:00:00	NaN	175.0	355.0
2	2020-07-01 12:00:00	345.0	345.0	345.0
3	2020-07-01 18:00:00	NaN	197.5	18.0
4	2020-07-02 00:00:00	50.0	50.0	50.0

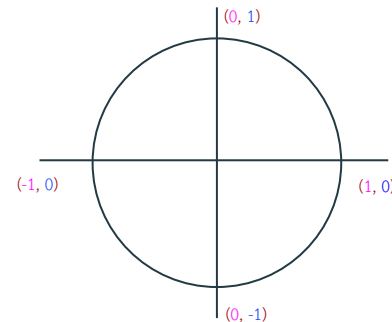
4. แก้ไขการ Processing ข้อมูล

4.1 ข้อมูล wind_direction

หลังจากมีการทำ cyclic interpolation ของ wind direction เรียบร้อยแล้ว อีกขั้นตอนในการพิจารณาความใกล้เคียงของมุมในแง่ของตัวเลข (numerical) ที่จะนำไป train model ต่อไป จึงได้นำข้อมูล wind_dir มา process โดยมีการใช้ function sin, cos เกิดเป็น column `wind_dir_sin` และ `wind_dir_cos` ตามลำดับ



```
#wind_dir_sin and wind_dir_cos
df['wind_dir_sin'] = np.sin(df['wind_dir']*2*np.pi/360).round(4)
df['wind_dir_cos'] = np.cos(df['wind_dir']*2*np.pi/360).round(4)
```



4. แก้ไขการ Processing ข้อมูล

4.2 ข้อมูล hotspot

เนื่องจากข้อมูล hotspot ที่ไป scrape มาได้จาก <http://asmc.asean.org/asmc-haze-hotspot-daily-new/#Hotspot>

นั้นเป็นข้อมูลจำนวน hotspot ของประเทศในแถบอาเซียน ซึ่งความถี่ของข้อมูลที่ได้มาจะอยู่ในระดับวัน (daily) ที่มีการแบ่งเป็น 2 ช่วงเวลาคือ ช่วงกลางวัน (day) กับช่วงกลางคืน (night)

	date	hotspot_thailand_day	hotspot_thailand_night	hotspot_myanmar_day	hotspot_myanmar_night
0	2017-07-01	0.0	0.0	0.0	0.0
1	2017-07-02	0.0	0.0	0.0	0.0
2	2017-07-03	0.0	0.0	0.0	0.0
3	2017-07-04	0.0	0.0	0.0	0.0
4	2017-07-05	0.0	0.0	0.0	0.0

4. แก้ไขการ Processing ข้อมูล

4.2 ข้อมูล hotspot

เนื่องจากข้อมูล PM2.5 ที่จะทำนาย จะต้องทำนายทุก ๆ 6 ชั่วโมง เราจึงมีการเปลี่ยน resolution ของข้อมูลจากรายวัน (daily) ให้เป็นราย 6 ชั่วโมง (6 hourly) (0:00, 6:00, 12:00, 18:00) โดยมีหลักการดังนี้

4. [For Train Duration (2017-07-01 to 2020-06-30)]

```
at any date x
- 0:00 and 6:00 we will use the hotspot counts at "night" time of date x divided by 12 (hours)
- 12:00 and 18:00 we use the hotspot counts at "day" time of date x divided by 12 (hours)
```

[For Test Duration (2020-07-01 to 2021-07-01)]

```
to prevent using the future data
at any date x
- 0:00 and 6:00 we will use the hotspot counts at "night" time of date "x-1" divided by 12 (hours)
- 12:00 and 18:00 we will use the hotspot counts at "day" time of date "x-1" divided by 12 (hours)
```

Note: after divided by 12 we will round it up

Train Data

→ ช่วง 0:00, 6:00 จะใช้จำนวน hotspot ในช่วงกลางคืน หารด้วย 12 ของวันนั้น

ส่วนช่วง 12:00, 18:00 จะใช้จำนวน hotspot ในช่วงกลางวัน หารด้วย 12 ของวันนั้น

4. แก้ไขการ Processing ข้อมูล

4.2 ข้อมูล hotspot

เนื่องจากข้อมูล PM2.5 ที่จะทำนาย จะต้องทำนายทุก ๆ 6 ชั่วโมง เราจึงมีการเปลี่ยน resolution ของข้อมูลจากรายวัน (daily) ให้เป็นราย 6 ชั่วโมง (6 hourly) (0:00, 6:00, 12:00, 18:00) โดยมีหลักการดังนี้

4. [For Train Duration (2017-07-01 to 2020-06-30)]

```
at any date x
- 0:00 and 6:00 we will use the hotspot counts at "night" time of date x divided by 12 (hours)
- 12:00 and 18:00 we use the hotspot counts at "day" time of date x divided by 12 (hours)
```

[For Test Duration (2020-07-01 to 2021-07-01)]

```
to prevent using the future data
at any date x
- 0:00 and 6:00 we will use the hotspot counts at "night" time of date "x-1" divided by 12 (hours)
- 12:00 and 18:00 we will use the hotspot counts at "day" time of date "x-1" divided by 12 (hours)
```

Note: after divided by 12 we will round it up

Test Data

เพื่อป้องกันการใช้ข้อมูลในอนาคต

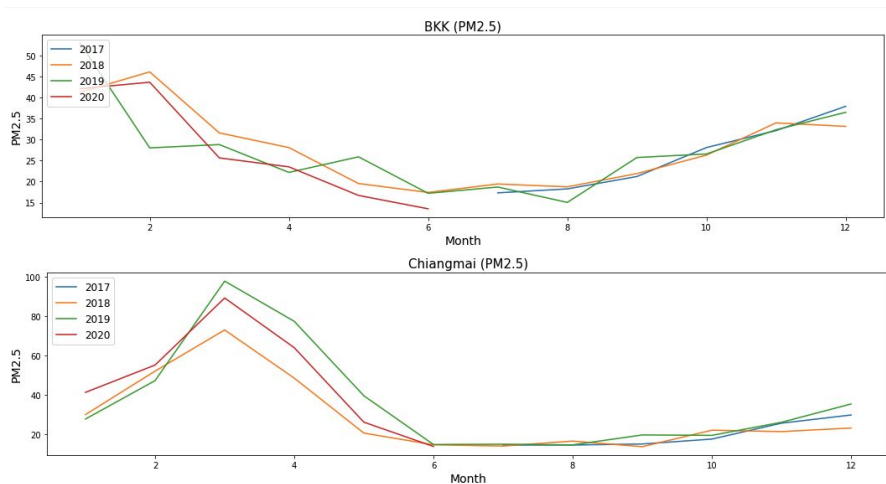
ช่วง 0:00, 6:00 จะใช้จำนวน hotspot ในช่วง
กลางคืน หารด้วย 12 ของวันก่อนหน้า

ส่วนช่วง 12:00, 18:00 จะใช้จำนวน hotspot ใน
ช่วงกลางวัน หารด้วย 12 ของวันก่อนหน้า

4. แก้ไขการ Processing ข้อมูล

4.3 ข้อมูล month ของแต่ละ timestep

เนื่องจากทางกลุ่มได้ทำการ visualize ปริมาณ PM2.5 เฉลี่ยในแต่ละเดือนของแต่ละปี ของแต่ละจังหวัดออกมาแล้ว พบว่ามีแนวโน้มที่น่าสนใจดังรูป



4. แกะไขการ Processing ข้อมูล

4.3 ข้อมูล month ของแต่ละ timestep

เราได้เพิ่ม column month_{x} โดย $1 \leq x \leq 12$ ซึ่งเป็นการบ่งบอกเดือนของแต่ละ timestep ซึ่งอาจนำไปใช้ทดลองเป็น feature ที่จะใส่เข้าไป model ในอนาคต (month เป็น nominal value)

	PM2.5	wind_speed	wind_dir	temp	hotspot_thailand	hotspot_myanmar	hotspot_cambodia	hotspot_lao_pdr	hotspot_p_malaysia	month_1	...	month_5	month_6	month_7	month_8	month_9	month_10	month_11	month_12
date_time																			
2020-07-01 00:00:00	9.0	36.0	262.0	30.9	0.0	0.0	0.0	0.0	0.0	0	...	0	0	1	0	0	0	0	0
2020-07-01 06:00:00	11.0	55.0	270.0	28.3	0.0	0.0	0.0	0.0	0.0	0	...	0	0	1	0	0	0	0	0
2020-07-01 12:00:00	15.0	51.0	260.0	29.7	2.0	0.0	0.0	0.0	0.0	0	...	0	0	1	0	0	0	0	0
2020-07-01 18:00:00	14.0	35.0	265.0	29.5	2.0	0.0	0.0	0.0	0.0	0	...	0	0	1	0	0	0	0	0
2020-07-02 00:00:00	18.0	60.0	275.0	28.7	0.0	0.0	0.0	0.0	0.0	0	...	0	0	1	0	0	0	0	0
...
2021-06-30 18:00:00	15.0	29.0	270.0	26.0	0.0	0.0	4.0	2.0	0.0	0	...	0	1	0	0	0	0	0	0
2021-07-01 00:00:00	15.0	20.0	265.0	28.4	0.0	0.0	0.0	0.0	0.0	0	...	0	0	1	0	0	0	0	0
2021-07-01 06:00:00	14.0	21.0	275.0	26.5	0.0	0.0	0.0	0.0	0.0	0	...	0	0	1	0	0	0	0	0
2021-07-01 12:00:00	14.0	17.0	245.0	32.3	0.0	0.0	5.0	0.0	1.0	0	...	0	0	1	0	0	0	0	0
2021-07-01 18:00:00	14.0	21.0	245.0	31.2	0.0	0.0	5.0	0.0	1.0	0	...	0	0	1	0	0	0	0	0

1464 rows x 23 columns

5. Model

Model ที่เราใช้ในการทำนายยังคงใช้ SARIMAX เช่นเดิม ซึ่งจะสร้าง 1 SARIMAX Model ต่อการทำนายปริมาณฝุ่น PM2.5 ในแต่ละจังหวัดเช่นเดิม ตามที่ได้ทำไว้ตั้งแต่รอบแรก โดยมีหลักการทำงานดังนี้

- Initiate SARIMAX model for predicting target (PM2.5) variable
- Initiate SARIMAX models for predicting each exog variables

(At timestep t , we want to predict timestep $t+1$, $t+2$, ..., $t+12$)

- Append target data at timestep t into SARIMAX target result
- Append each exog data at timestep t into its SARIMAX exog result
- Predict the next 12 timesteps of all exog variables
- Use predicted exog variables to predict the next 12 timesteps of target (PM2.5) variable

จะใช้ข้อมูลถึง timestep t
เท่านั้นในการ predict
timestep $t+1$ ถึง $t+12$

→ (ค่า exog ที่ใช้ในการ predict target ก็จะใช้ค่า
exog ที่มาจากการ predict ด้วยข้อมูลถึง
timestep t เท่านั้น)

ไม่มีการใช้ข้อมูลในอนาคต

Note: เนื่องจากข้อมูล month เป็น fact สามารถกรอกล่วงหน้าได้ ดังนั้นจะไม่มีการทำนายค่า month
(ในกรณีที่มีการใช้ month เป็น exog variables)

5. Model

Hyperparameter ที่ใช้จะมาจากการทำ Random Search / Grid Search จากการทำในรอบแรก

คือใช้ค่า $(p, d, q) = (1, 0, 1)$ และ $(P, D, Q, s) = (0, 1, 1, 12)$

โดยเรามีการ train Model โดยใช้ list ของ exog variables ที่แตกต่างกันไป โดยมีผลลัพธ์ดังนี้

No. (in code)	Exog Variables ที่ใช้	Final RMSE
0.1	wind_speed, wind_dir, temp	8.5169
0.2	wind_speed, wind_dir_sin, wind_dir_cos, temp	8.5271
3	wind_speed, wind_dir_sin, wind_dir_cos, temp, hotspot (แตกต่างกันไปในแต่ละจังหวัด), month column	9.9183
4	wind_speed, wind_dir_sin, wind_dir_cos, temp, month column	8.5396





5. Model

รายละเอียดของค่า RMSE ของแต่ละจังหวัดจากการ predict ของแต่ละ model

Model No.	BKK	Chiangmai	Rayong	Saraburi	Khonkaen	Surat	Total RMSE
0.1	6.0741	9.7507	7.5361	10.7188	10.1522	5.4867	8.5169
0.2	6.1158	9.7592	7.5783	10.7516	10.1081	5.4792	8.5271
3	6.6024	12.0486	7.8922	12.5936	12.4263	5.3786	9.9183
4	6.0023	9.8863	7.5922	10.6020	10.3298	5.3518	8.5396

5. Model

Captured screen (RMSE) ของ Kaggle ที่ได้จากการส่ง
และค่า RMSE ที่ได้จากการ run code

2	โจมตีทางอากาศ	  	8.52706	1	8h
	Your First Entry! Welcome to the leaderboard!				

```
Province [1/6]: BKK  
(start at: 2022-04-21 10:59:12)  
Parameters: order = (1, 0, 1), seasonal_order = (0, 1, 1, 12)  
Exog Columns: ['wind_speed', 'wind_dir_sin', 'wind_dir_cos', 'temp']  
100% ██████████ 1464/1464 [50:05<00:00, 2.05s/it]  
Test on SARIMAX with RMSE = 6.1158  
(finish at: 2022-04-21 11:49:45)
```

```
Province [2/6]: Chiangmai  
(start at: 2022-04-21 11:49:45)  
Parameters: order = (1, 0, 1), seasonal_order = (0, 1, 1, 12)  
Exog Columns: ['wind_speed', 'wind_dir_sin', 'wind_dir_cos', 'temp']  
100% ██████████ 1464/1464 [49:31<00:00, 2.03s/it]  
Test on SARIMAX with RMSE = 9.7592  
(finish at: 2022-04-21 12:39:39)
```

```
Province [3/6]: Rayong
(start at: 2022-04-21 12:39:39)
Parameters: order = (1, 0, 1), seasonal_order = (0, 1, 1, 12)
Exog Columns: ['wind_speed', 'wind_dir_sin', 'wind_dir_cos', 'temp']
100% ██████████ 1464/1464 [55:57<00:00, 2.29s/it]
Test on SARIMAX with RMSE = 7.5783
(finish at: 2022-04-21 13:36:02)
```

```
Province [4/6]: Saraburi  
(start at: 2022-04-21 13:36:02)  
Parameters: order = (1, 0, 1), seasonal_order = (0, 1, 1, 12)  
Exog Columns: ['wind_speed', 'wind_dir_sin', 'wind_dir_cos', 'temp']  
100% ██████████ 1464/1464 [1:00:28<00:00, 2.48s/it]  
Test on SARIMAX with RMSE = 10.7516  
(finish at: 2022-04-21 14:36:59)
```

```
Province [5/6]: Khonkaen  
(start at: 2022-04-21 14:36:59)  
Parameters: order = (1, 0, 1), seasonal_order = (0, 1, 1, 12)  
Exog Columns: ['wind_speed', 'wind_dir_sin', 'wind_dir_cos', 'temp']  
100% ██████████ 1464/1464 [56:31<00:00, 2.32s/it]  
Test on SARIMAX with RMSE = 10.1081  
(finish at: 2022-04-21 15:33:56)
```

```
Province [6/6]: Surat  
(start at: 2022-04-21 15:33:56)  
Parameters: order = (1, 0, 1), seasonal_order = (0, 1, 1, 12)  
Exog Columns: ['wind_speed', 'wind_dir_sin', 'wind_dir_cos', 'temp']  
100% ██████████ 1464/1464 [39:44<00:00, 1.63s/it]  
Test on SARIMAX with RMSE = 5.4792  
(finish at: 2022-04-21 16:13:56)
```

Final test on SARIMAX with RMSE = 8.5271

Total of 94248 predicted rows

Thank you!!