Data Mining Research and Practice HW3

Department: IAM Student ID: 309652008 Name: 廖家緯
GitHub Link: https://github.com/Jia-Wei-Liao/Time-Series-Regression

November 18, 2021

1 引入套件

本次作業使用 numpy、pandas、sklearn、xgboost 套件

```
import numpy as np
import pandas as pd
import xgboost as xgb
from sklearn import metrics
from sklearn.linear_model import LinearRegression
```

2 讀取資料

```
origin_df = pd.read_csv(open('新竹_2020.csv'))
origin_df.drop(0, inplace=True)
origin_df.columns = [e.strip() for e in list(origin_df.columns)]
origin_df = origin_df.applymap(lambda x: x.strip())
origin_df.head(18)
```

	測站	日期	測項	00	01	02	03	04	05	06	...	14	15	16	17	18	19	20	21	22	23
1	新竹	2020/01/01 00:00:00	AMB_TEMP	15.2	15.2	15.3	15.3	15.3	15.4	15.5	...	18.1	18.2	17.9	17.3	16.7	16.4	16.2	16.1	16	15.8
2	新竹	2020/01/01 00:00:00	CH4	1.74	1.74	1.77	1.78	1.77	1.77	1.77	...	1.78	1.78	1.77	1.8	1.81	1.82	1.85	1.83	1.92	1.94
3	新竹	2020/01/01 00:00:00	CO	0.28	0.25	0.24	0.22	0.2	0.19	0.2	...	0.28	0.29	0.28	0.34	0.39	0.41	0.46	0.49	0.58	0.52
4	新竹	2020/01/01 00:00:00	NMHC	0.06	0.07	0.05	0.05	0.05	0.05	0.07	...	0.09	0.09	0.07	0.08	0.12	0.12	0.16	0.14	0.17	0.2
5	新竹	2020/01/01 00:00:00	NO	0.3	0.6	0.6	0.6	0.3	0.3	0.5	...	1.6	1.6	1.2	0.7	0.9	1.1	1.1	1.7	1.8	1.4
6	新竹	2020/01/01 00:00:00	NO2	6.7	6.9	6.4	5.9	4.8	5	6.1	...	10.9	11.1	10.7	13	15.7	17.8	21.2	22.4	22.8	22.1
7	新竹	2020/01/01 00:00:00	NOx	7	7.5	6.9	6.5	5.3	5.5	6.6	...	12.5	12.8	11.9	13.7	16.6	18.9	22.4	24.1	24.6	23.5
8	新竹	2020/01/01 00:00:00	O3	33.1	33.4	33.5	32.6	33.5	33.6	32.6	...	33.9	33.1	32.8	29	24.9	21.1	15.2	13.7	11.9	12.4
9	新竹	2020/01/01 00:00:00	PM10	36	28	22	19	14	10	14	...	19	18	5	5	11	10	5	12	12	12
10	新竹	2020/01/01 00:00:00	PM2.5	21	8	8	8	9	6	5	...	2	7	6	3	6	5	7	5	8	8
11	新竹	2020/01/01 00:00:00	RAINFALL	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
12	新竹	2020/01/01 00:00:00	RH	74	74	73	74	77	77	76	...	70	70	71	74	77	78	78	79	79	81
13	新竹	2020/01/01 00:00:00	SO2	1	1	1	0.5	0.5	0.5	0.5	...	1	0.5	1.4	1.5	1.6	1.8	1.4	1.1	1.1	1.7
14	新竹	2020/01/01 00:00:00	THC	1.8	1.81	1.82	1.83	1.82	1.82	1.84	...	1.87	1.87	1.84	1.88	1.93	1.94	2.01	1.97	2.09	2.14
15	新竹	2020/01/01 00:00:00	WD_HR	47	52	44	43	46	50	47	...	40	44	56	55	55	53	71	80	337	96
16	新竹	2020/01/01 00:00:00	WIND_DIREC	43	48	48	56	56	53	38	...	34	57	57	53	57	53	76	29	297	116
17	新竹	2020/01/01 00:00:00	WIND_SPEED	3.7	2.8	3.5	3.1	3	3.3	3.2	...	3	3.3	2.4	2.6	1.7	1.9	1.1	0.6	0.4	1.2
18	新竹	2020/01/01 00:00:00	WS_HR	3.3	2.7	2.6	2.5	2.3	2.5	2.3	...	2.3	2.4	2.4	2.2	1.9	1.7	0.7	0.6	0.1	0.5

Figure 1. Origin data

1. 資料集包含 AMB_TEMP、CH4、CO、NMHC、NO、NO2、NOx、O3、PM10、PM2.5、RAINFALL、RH、SO2、THC、WD_HR、WIND_DIREC、WIND_SPEED、WS_HR 共 18 種屬性 (汙染物)。

2. 特殊符號含意:

   - # 表示儀器檢核為無效值
   - * 表示程式檢核為無效值
   - x 表示人工檢核為無效值
   - A 是指因儀器疑似故障警報所產生的無效值
   - 空白表示缺失值

3. 資料來源: https://airtw.epa.gov.tw/CHT/Query/His_Data.aspx

3 資料前處理

3.1 取出 10 到 12 月的資料

```
month = ['10', '11', '12']
index = origin_df['日期'].map(lambda x: True if x[5:7] in month else False)
origin_df = origin_df[index]
df = origin_df.iloc[:, 3:]
```

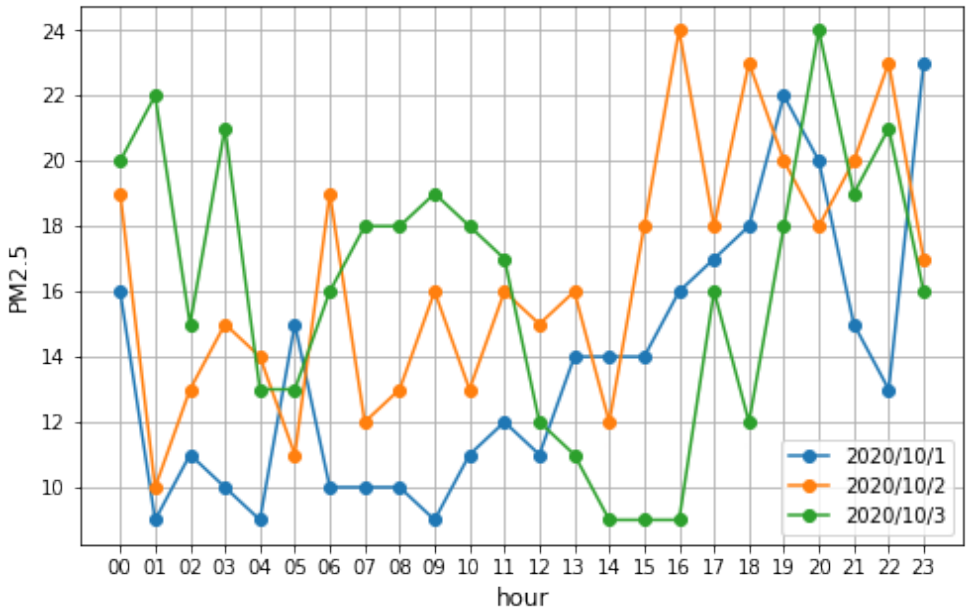

Figure 2. PM2.5 of 10/1 to 10/3

3.2 填補缺失值

以前後一小時平均值取代缺失值以及無效值 (前一小時仍有空值，再往前取前一小時)

3.2.1 檢查是否為特殊符號

```
def check_special_symbol(df, i, j, symbol_list):
    for s in symbol_list:
        if str(df.iloc[i, j]).rfind(s) != -1:
            return True
    else:
        return False
```

3.2.2 找尋前後填補值及無效值

以遞迴方式找尋填補值

- 若第 0 小時為特殊符號，需遞迴至前一天第 23 小時；若第 23 小時為特殊符號，需遞迴至後一天第 0 小時
- 若皆為特殊符號則回傳 $(-1, -1)$

```
def get_lower_number(df, i, j, symbol_list):
    n, m = df.shape
    if i < 0:
        return -1, -1

    elif j==-1:
        return get_lower_number(df, i-18, m-1, symbol_list)

    elif check_special_symbol(df, i, j, symbol_list):
        return get_lower_number(df, i, j-1, symbol_list)

    else:
        return i, j

def get_hight_number(df, i, j, symbol_list):
    n, m = df.shape
    if i > n:
        return -1, -1

    elif j==len(df.columns):
        return get_hight_number(df, i+18, 0, symbol_list)

    elif check_special_symbol(df, i, j, symbol_list):
        return get_hight_number(df, i, j+1, symbol_list)

    else:
        return i, j
```

3.2.3 搜尋缺失值或無效值進行填補

遇缺失值，尋找前後值的平均進行填補。若後面皆為特殊符號，直接以前面的值填補

```
def filled_x(df):
    symbol_list = ['#', '*', 'x', 'A']
    n, m = df.shape
    for i in range(n):
        for j in range(m):
            if check_special_symbol(df, i, j, symbol_list):
                lower_index = get_lower_number(df, i, j-1, symbol_list)
                hight_index = get_hight_number(df, i, j+1, symbol_list)

                if lower_index == (-1, -1):
                    lower_index = hight_index

                elif hight_index == (-1, -1):
                    hight_index = lower_index

                df.iloc[i, j] = str((float(df.iloc[lower_index]) +
                                     float(df.iloc[hight_index])) / 2)

    return df

df = filled_x(df).astype(float)
```

3.3 將無降雨以 0 取代

```
df[df=='NR'] = 0
```

4 分割訓練集及測試集

- 訓練集: 10、11 月，測試集: 12 月
- 訓練資料 : 測試資料 $= (31+30)\times 18 : 31\times 18 = 1098 : 558$

```
train_index = (31+30)*18
train_df = df.iloc[:train_index, :]
test_df = df.iloc[train_index:, :]
```

4.1 製作時間序列資料

將資料集做轉換，列代表 18 種屬性，行代表每小時監測的數據

```
train_X = np.hstack([train_df.iloc[0+18*i:18+18*i, :]
                     for i in range(len(train_df) // 18)])
test_X = np.hstack([test_df.iloc[0+18*i:18+18*i, :]
                     for i in range(len(test_df) // 18)])
```

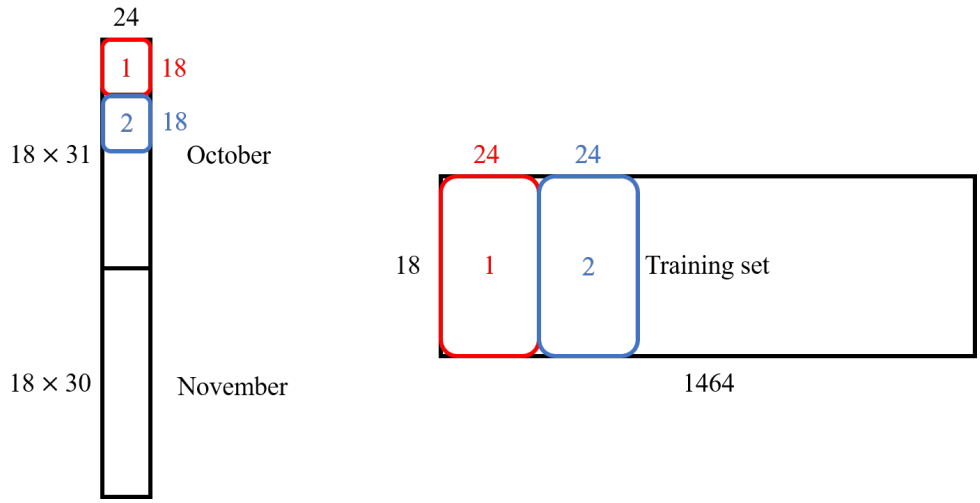

Figure 3. Data transformation

4.2 預測目標

```
def get_time_series_label(df, pred_hour_unit, PM2p5=9):
    return np.vstack([df[PM2p5, pred_hour_unit+i]
                      for i in range(df.shape[1]-pred_hour_unit)])
```

4.2.1 預測未來第 1 個小時

以每 6 小時為一單位，預測第 7 小時 (未來第 1 小時)

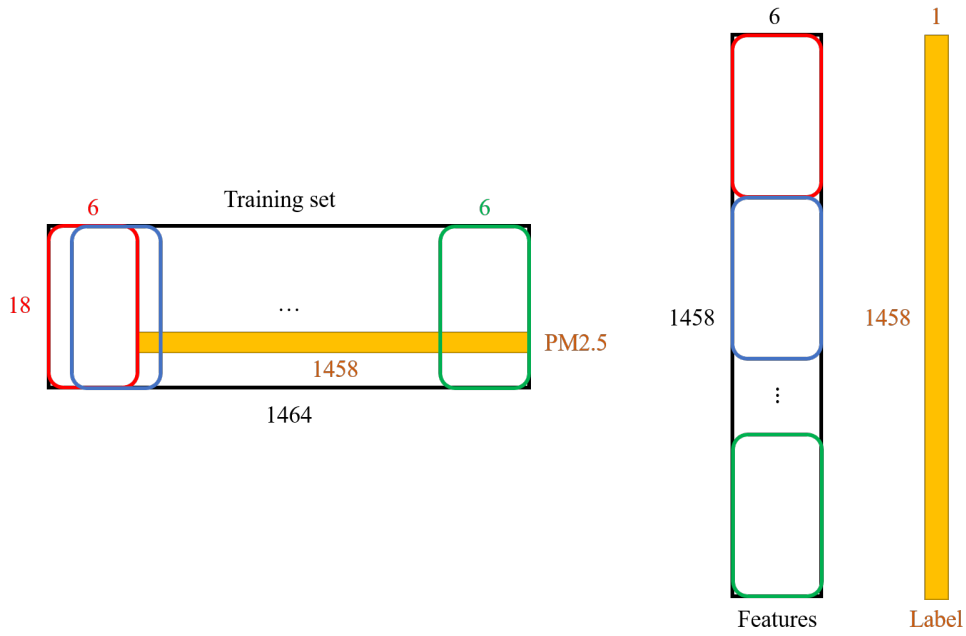

Figure 4. Features and Label

```
train_Y_c6_p6 = get_time_series_label(train_X, pred_hour_unit=6)
test_Y_c6_p6 = get_time_series_label(test_X, pred_hour_unit=6)
```

4.2.2 預測未來第 6 個小時

以每 6 小時為一單位，預測第 12 小時 (未來第 6 小時)

```
train_Y_c6_p11 = get_time_series_label(train_X, pred_hour_unit=11)
test_Y_c6_p11 = get_time_series_label(test_X, pred_hour_unit=11)
```

4.3 資料特徵選取

```
def get_time_series_data(df, cut_unit, pred_hour_unit, PM2p5):
    return np.vstack([df[PM2p5, 0+i:cut_unit+i].reshape(1, -1)
                      for i in range(df.shape[1]-pred_hour_unit)])
```

4.3.1 使用 PM2.5 屬性

```
train_X_c6_p6_f1 = get_time_series_data(train_X, 6, 6, 9)
test_X_c6_p6_f1 = get_time_series_data(test_X, 6, 6, 9)
train_X_c6_p11_f1 = get_time_series_data(train_X, 6, 11, 9)
test_X_c6_p11_f1 = get_time_series_data(test_X, 6, 11, 9)
```

4.3.2 使用 18 種屬性

```
train_X_c6_p6_f18 = get_time_series_data(train_X, 6, 6, range(18))
test_X_c6_p6_f18 = get_time_series_data(test_X, 6, 6, range(18))
train_X_c6_p11_f18 = get_time_series_data(train_X, 6, 11, range(18))
test_X_c6_p11_f18 = get_time_series_data(test_X, 6, 11, range(18))
```

5 模型建立及預測

採用 Linear Regression 、XGBoost 兩種模型

```
def Modeling(data_list, model):
    mae_list = []
    for tr_X, tr_Y, ts_X, ts_Y in data_list:
        model.fit(tr_X, tr_Y)
        prediction = model.predict(ts_X)
        mae = metrics.mean_absolute_error(ts_Y, prediction)
        mae_list.append(round(mae, 4))

    return mae_list

lr_model = LinearRegression()
lr_score = Modeling(data_list, lr_model)
xgboost_model = xgb.XGBRegressor(objective='reg:squarederror', verbosity=2)
xgboost_score = Modeling(data_list, xgboost_model)
```

5.1 預測結果

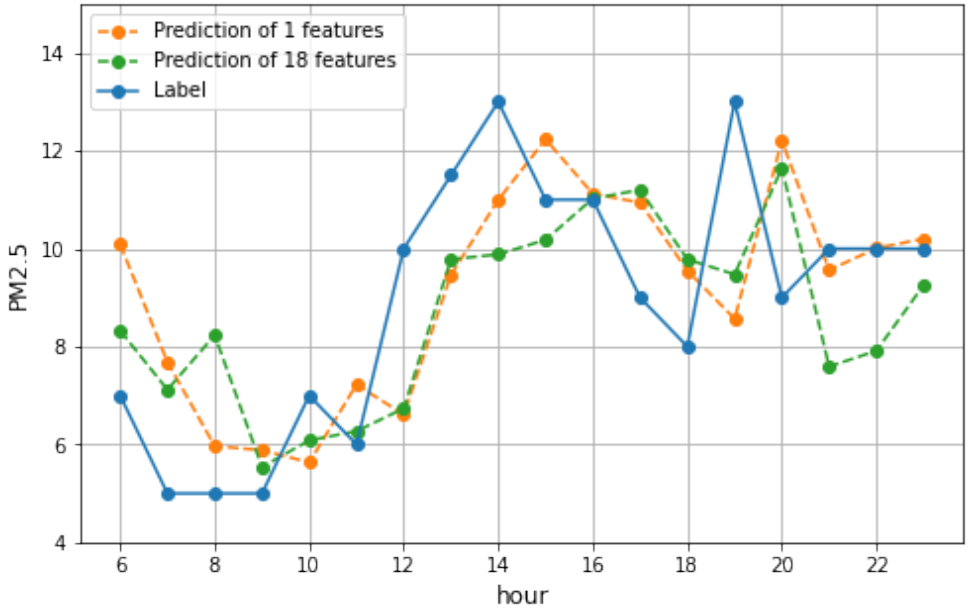

Figure 5. Linear Regression predicted PM2.5 in 2020/12/1

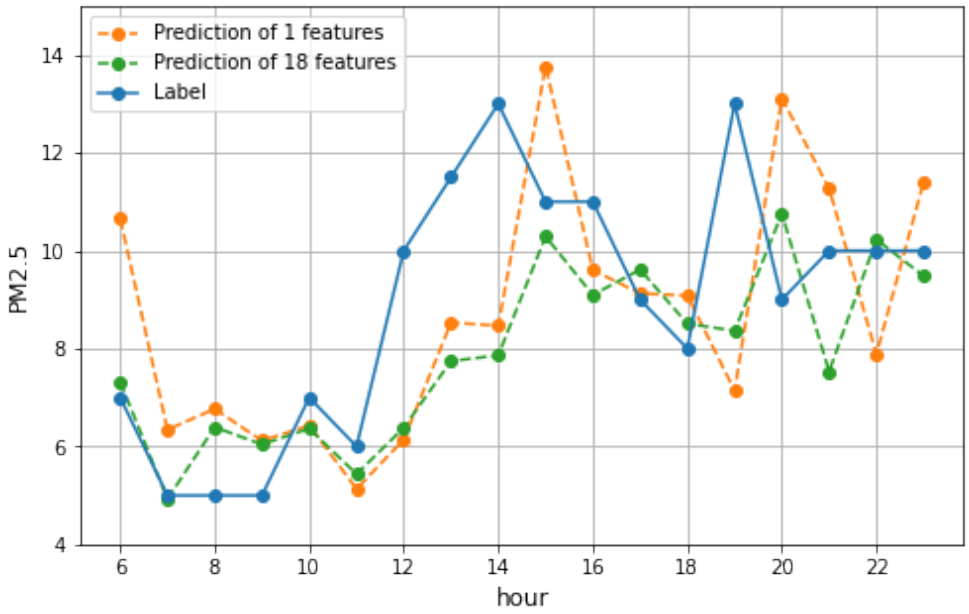

Figure 6. Xgboost predicted PM2.5 in 2020/12/1

6 模型評估

6.1 Mean Absolute Error (MAE)

設 $\{(x_i, y_i)\}_{i=1}^{n}$ 表示 n 筆 data，f 為模型，定義

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|f(x_i) - y_i|$$

Predict future hour	Future 1 hour		Future 6 hour	
Features	Only PM2.5	18 features	Only PM2.5	18 features
Linear Regression Model	2.5727	2.2964	4.0234	3.6254
Xgboost Model	0.6177	0.0864	1.0306	0.0771

Table 1. Training set MAE

Predict future hour	Future 1 hour		Future 6 hour	
Features	Only PM2.5	18 features	Only PM2.5	18 features
Linear Regression Model	2.5224	2.6959	4.5794	6.0882
Xgboost Model	3.0083	2.9761	5.0085	4.6741

Table 2. Test set MAE

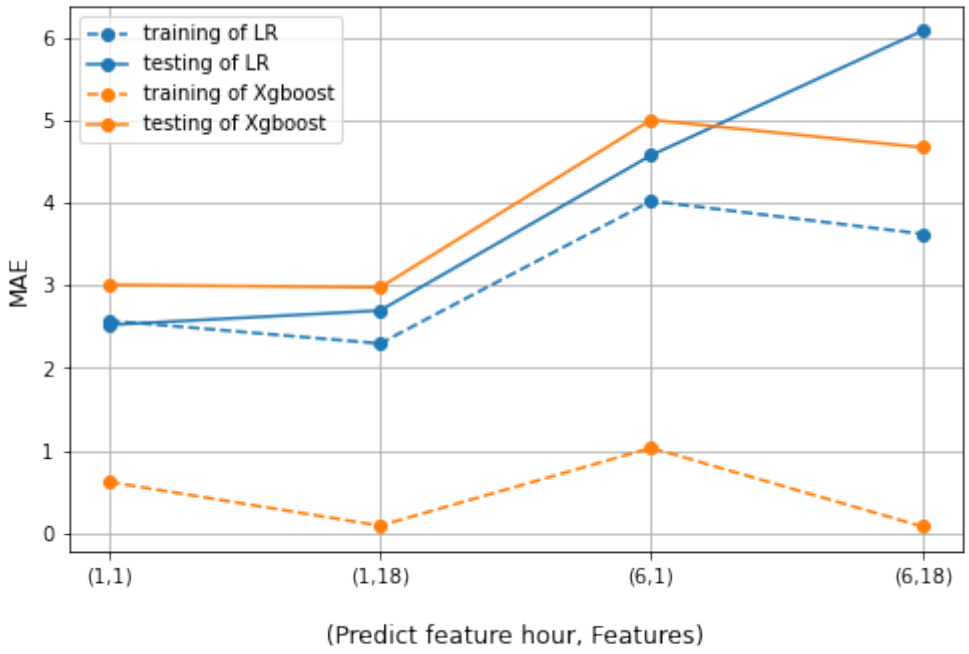

Figure 7. MAE of Linear Regression and Xgboost Models

6.2 R-squared (coefficient of determination)

設 $\{(x_i, y_i)\}_{i=1}^n$ 表示 n 筆 data，f 為模型，定義

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i, \tag{1}$$

$$SS_{tot} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \bar{y})^2, \tag{2}$$

$$SS_{res} = \frac{1}{n}\sum_{i=1}^{n} (f(x_i) - y_i)^2, \tag{3}$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{4}$$

則 (4) 稱為 R-squared 值或判定係數

Predict future hour	Future 1 hour		Future 6 hour	
Features	Only PM2.5	18 features	Only PM2.5	18 features
Linear Regression Model	0.8493	0.823	0.4728	0.2654
Xgboost Model	0.765	0.7873	0.393	0.4166

Table 3. Test set R-squared

7 總結

根據 MAE 指標的定義，MAE 越低越好。從我們實驗的數據可以發現，Linear Regression 只使用 PM2.5 特徵預測未來第一小時的效果最好。若使用 18 種特徵，在 training set 上的表現雖然不錯，但 test set 明顯表現不佳，表示已經 over-fitting。而 Xgboost 並不會因為多種特徵而導致 over-fitting，但其表現比 Linear Regression 還要差。另外，當我們預測的時間離我們選取的範圍越遠時，模型的表現也會越來越差，符合我們的直覺。

R-squared 指標表示模型的解釋能力，其值越高越好。從上表可以發現，對於 Linear Regression 而言，只使用 PM2.5 特徵比使用 18 種特徵的解釋力來的高，當模型為預測第小時，R-squared 差異更明顯，因此建立模型的時候不是越多特徵越好，特徵工程必須認真做，才能達到好的結果。