團隊程式說明

報名序號：112009

團隊名稱：慢靈魂

一、程式執行的安裝環境說明

- 作業系統：Windows 11
- 程式語言：Python 語言 3.7.16
- 工具軟體：
  - Keras 2.1.5
  - Statsmodels 0.13.5
  - Scikit-learn 1.0.2
  - Xgboost 1.6.2
  - Lightgbm 4.0.0
  - Catboost 1.2.1
  - Optuna 3.3.0
  - Numpy 1.21.6

二、程式執行步驟說明 – 下載套件

```
!pip  install  pmdarima
```

```
Collecting pmdarima
  Downloading pmdarima-2.0.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (1.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 8.5 MB/s eta 0:00:00
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.3.2)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (3.0.2)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.23.5)
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.5.3)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.2.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.11.2)
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (0.14.0)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (2.0.4)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (67.7.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2023.3.post1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22->pmdarima) (3.2.0)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2->pmdarima) (0.5.3)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2->pmdarima) (23.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.2->statsmodels>=0.13.2->pmdarima) (1.16.0)
Installing collected packages: pmdarima
Successfully installed pmdarima-2.0.3
```

```
!pip  install  optuna
```

```
Collecting optuna
  Downloading optuna-3.3.0-py3-none-any.whl (404 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 404.2/404.2 kB 7.0 MB/s eta 0:00:00
Collecting alembic>=1.5.0 (from optuna)
  Downloading alembic-1.12.0-py3-none-any.whl (226 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 226.0/226.0 kB 28.4 MB/s eta 0:00:00
Collecting cmaes>=0.10.0 (from optuna)
  Downloading cmaes-0.10.0-py3-none-any.whl (29 kB)
Collecting colorlog (from optuna)
  Downloading colorlog-6.7.0-py2.py3-none-any.whl (11 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from optuna) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from optuna) (23.1)
Requirement already satisfied: sqlalchemy>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from optuna) (2.0.20)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from optuna) (4.66.1)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-packages (from optuna) (6.0.1)
Collecting Mako (from alembic>=1.5.0->optuna)
  Downloading Mako-1.2.4-py3-none-any.whl (78 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 78.7/78.7 kB 11.0 MB/s eta 0:00:00
Requirement already satisfied: typing-extensions>=4 in /usr/local/lib/python3.10/dist-packages (from alembic>=1.5.0->optuna) (4.5.0)
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.10/dist-packages (from sqlalchemy>=1.3.0->optuna) (2.0.2)
Requirement already satisfied: MarkupSafe>=0.9.2 in /usr/local/lib/python3.10/dist-packages (from Mako->alembic>=1.5.0->optuna) (2.1.3)
Installing collected packages: Mako, colorlog, cmaes, alembic, optuna
Successfully installed Mako-1.2.4 alembic-1.12.0 cmaes-0.10.0 colorlog-6.7.0 optuna-3.3.0
```

```
pip  install  catboost
```

```
Collecting catboost
  Downloading catboost-1.2.1-cp310-cp310-manylinux2014_x86_64.whl (98.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.7/98.7 MB 10.5 MB/s eta 0:00:00
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from catboost) (0.20.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from catboost) (3.7.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.23.5)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.5.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from catboost) (1.11.2)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (from catboost) (5.15.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from catboost) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2023.3.post1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (4.42.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (23.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (3.1.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly->catboost) (8.2.3)
Installing collected packages: catboost
Successfully installed catboost-1.2.1
```

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
from google.colab import drive
drive.mount('/content/drive/')
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from statsmodels.tsa.api import ExponentialSmoothing
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor
from catboost import CatBoostRegressor
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt
from statsmodels.tsa.arima_model import ARIMA
import pmdarima as pm
# from catboost import CatBoostRegressor
from sklearn.linear_model import Lasso,Ridge,ElasticNet
#import autosklearn.regression
from sklearn.impute import KNNImputer
import optuna
from statsmodels.tsa.holtwinters import Holt
```

```
Mounted at /content/drive/
```

二、程式執行步驟說明 – 資料整理

將異常資料集中沒出現的日期以及爐層數補0

```
anormaly_train1 = pd.read_csv('/content/drive/MyDrive/data/anomaly_train改.csv')
for i in range(len(anormaly_train1)):
    if anormaly_train1['oven_id'][i] == '1.00E+00':
        anormaly_train1['oven_id'][i] = '1E0'
    elif anormaly_train1['oven_id'][i] == '2.00E+00':
        anormaly_train1['oven_id'][i] = '2E0'
```

```
# 產線二資料
oven,layer,date,anormaly_total_number = [],[],[],[]
for d in pd.date_range('2021/12/27','2023/02/06'):
    for i in ['2B0','2C0','2D0','2E0','2G0']:
        for j in range(1,20):
            oven.append(i)
            layer.append(j)
            date.append(d)
            anormaly_total_number.append(0)

accumulation = np.zeros(407*5*19)
lamp = np.zeros(407*5*19)

full_data = {"oven_id": oven ,'layer_id':layer,'date':date,"lamp_id":lamp,"anomaly_accumulation_hour":accumulation,"anomaly_total_number":anormaly_total_number }
full_data_1 = pd.DataFrame(full_data)
```

```
#  產線一資料
oven,layer,date,anormaly_total_number  =  [],[],[],[]
for  d  in  pd.date_range('2021/12/27','2022/09/01'):
    for  i  in  ['1B0','1C0','1D0','1E0','1G0']:
        for  j  in  range(1,20):
            oven.append(i)
            layer.append(j)
            date.append(d)
            anormaly_total_number.append(0)

accumulation  =  np.zeros(249*5*19)
lamp  =  np.zeros(249*5*19)

full_data  =  {"oven_id":  oven  ,'layer_id':layer,'date':date,"lamp_id":lamp,"anomaly_accumulation_hour":accumulation,"anomaly_total_number":anormaly_total_number  }
full_data_2  =  pd.DataFrame(full_data)
```

```
#  pd.concat()  會讓資料上下相接，但  index  也會直接相接。
all_full_data  =  pd.concat([full_data_1,full_data_2])
```

```
#建立2021/12/27-2023/02/06有異常的爐和層的資料
#  anormaly_train_1  =  anormaly_train1[['date','oven_id','layer_id','anomaly_total_number']]
anormaly_train1['date']  =  pd.to_datetime(anormaly_train1['date'],format='%Y-%m-%d')

#合併2021/12/27-2023/02/06所有爐和所有層的資料
df  =  pd.concat([anormaly_train1,all_full_data])
#  subset=['date',  'oven_id',  'layer_id']  意味著僅考慮這三列的值來判斷是否重複。
#  keep='first'  表示保留第一個出現的重複行，而刪除後續的重複行。
#  inplace=True  表示在原始DataFrame上進行操作，不返回新的DataFrame。
df.drop_duplicates(subset=['date','oven_id','layer_id'],  keep='first',  inplace=True)
df  =  df.reset_index().drop(['index'],axis=1)
```

```
#合併2021/12/27-2023/02/06所有爐和所有層的資料
df  =  pd.concat([anormaly_train1,all_full_data])
#  subset=['date',  'oven_id',  'layer_id']  意味著僅考慮這三列的值來判斷是否重複。
#  keep='first'  表示保留第一個出現的重複行，而刪除後續的重複行。
#  inplace=True  表示在原始DataFrame上進行操作，不返回新的DataFrame。
df.drop_duplicates(subset=['date','oven_id','layer_id'],  keep='first',  inplace=True)
df  =  df.reset_index().drop(['index'],axis=1)
```

```
df
```

	date	oven_id	layer_id	lamp_id	anomaly_accumulation_hour	anomaly_total_number
0	2021-12-27	1B0	5	26_49	5116.0	2
1	2021-12-27	1C0	3	45_91	4699.0	2
2	2021-12-27	1D0	14	64	3241.0	1
3	2021-12-27	1E0	1	96	4138.0	1
4	2021-12-27	1E0	8	51	3818.0	1
...	...	...	...	...	...	...
62315	2022-09-01	1G0	15	0.0	0.0	0
62316	2022-09-01	1G0	16	0.0	0.0	0
62317	2022-09-01	1G0	17	0.0	0.0	0
62318	2022-09-01	1G0	18	0.0	0.0	0
62319	2022-09-01	1G0	19	0.0	0.0	0

62320 rows × 6 columns

二、程式執行步驟說明 – 新增特徵

```
accumulation_df  =  pd.read_csv('/content/drive/MyDrive/data/accumlation_hour.csv')
accumulation_mean  =  accumulation_df.groupby(['oven_id'])['accumulation_hour'].mean()
accumulation_max  =   accumulation_df.groupby(['oven_id'])['accumulation_hour'].max()
accumulation_min  =  accumulation_df.groupby(['oven_id'])['accumulation_hour'].min()
accumulation_mean
```

```
oven_id
1B0    8160.736842
1C0    7049.114035
1D0    7381.078947
1E0    6724.149123
1G0    2553.850877
2B0    6651.411483
2C0    7146.488038
2D0    7182.669856
2E0    5346.928230
2G0    1195.425837
Name: accumulation_hour, dtype: float64
```

```
power_df  =  pd.read_csv('/content/drive/MyDrive/data/power.csv')
```

```
power_df[['accumulation_hour0',  'accumulation_hour1']]  =  power_df['accumulation_hour'].str.split('-',  expand=True)

#  將結果轉換為數值型別（如果需要）
power_df['accumulation_hour0']  =  power_df['accumulation_hour0'].astype(int)
power_df['accumulation_hour1']  =  power_df['accumulation_hour1'].astype(int)
power_df.head()
```

	item	accumulation_hour	power_setup(other_lamp)	power_setup(lamp_1_2_60_61_62_63_121_122)	accumulation_hour0	accumulation_hour1
0	1	0-50	35.0	39.0	0	50
1	2	51-100	36.0	40.0	51	100
2	3	101-150	37.0	41.0	101	150
3	4	151-200	38.0	42.0	151	200
4	5	201-300	39.0	43.0	201	300

```
def  power_setup(accumulation_hour)  :
        for  i  in  range(len(power_df)):
                if  accumulation_hour  <=  power_df['accumulation_hour1'][i]  :
                        break
        return  power_df['power_setup(other_lamp)'][i]
```

```
def  creat_feature(oven, days):

    oven_df  =  df[df['oven_id']==  oven  ].sort_values(by='date',ascending=True)
    oven2_total_anormal  =  oven_df.groupby(['date'])['anomaly_total_number'].sum()
    oven_total_anormal  =  pd.DataFrame(oven2_total_anormal)
    oven_total_anormal['oven_id']  =  oven

    #  過去27天的資料
    oven_total_anormal['number_sum']  =  np.nan
    oven_total_anormal['number_max']  =  np.nan
    oven_total_anormal['number_min']  =    np.nan
    oven_total_anormal['number_mode']  =  np.nan
    oven_total_anormal['days']  =  np.nan

    #  每個爐的平均損壞數量
    oven_total_anormal['oven_encoder']  =  oven_total_anormal['anomaly_total_number'].sum()  /  np.sum(oven_total_anormal['anomaly_total_number']  >0  )

    #  個別爐的水冷板溫度、累積時數、Power  setup
    if  oven  ==  'IB0':
           oven_total_anormal['cooler_max']  =  30.9
           oven_total_anormal['avg_accumulation']  ,oven_total_anormal['max_accumulation']  ,oven_total_anormal['min_accumulation']  =  accumulation_mean[0]  ,accumulation_max[0],accumulation_min[0]
           oven_total_anormal['avg_power_setup']  =  power_setup(accumulation_mean[0])

    elif  oven  ==  'IC0':
           oven_total_anormal['cooler_max']  =  30.5
           oven_total_anormal['avg_accumulation']  ,oven_total_anormal['max_accumulation']  ,oven_total_anormal['min_accumulation']  =  accumulation_mean[1],accumulation_max[1],accumulation_min[1]
           oven_total_anormal['avg_power_setup']  =  power_setup(accumulation_mean[1])

    elif  oven  ==  'ID0':
           oven_total_anormal['cooler_max']  =  27.5
           oven_total_anormal['avg_accumulation']  ,oven_total_anormal['max_accumulation']  ,oven_total_anormal['min_accumulation']  =  accumulation_mean[2],accumulation_max[2],accumulation_min[2]
           oven_total_anormal['avg_power_setup']  =  power_setup(accumulation_mean[2])

    elif  oven  ==  'IE0':
           oven_total_anormal['cooler_max']  =  27
           oven_total_anormal['avg_accumulation']  ,oven_total_anormal['max_accumulation']  ,oven_total_anormal['min_accumulation']  =  accumulation_mean[3],accumulation_max[3],accumulation_min[3]
           oven_total_anormal['avg_power_setup']  =  power_setup(accumulation_mean[3])
```

```
elif  oven  ==  '1G0':
       oven_total_anormal['cooler_max']  =  30.1
       oven_total_anormal['avg_accumulation']  ,oven_total_anormal['max_accumulation']  ,oven_total_anormal['min_accumulation']  =  accumulation_mean[4],accumulation_max[4],accumulation_min[4]
       oven_total_anormal['avg_power_setup']  =  power_setup(accumulation_mean[4])

elif  oven  ==  '2B0':
       oven_total_anormal['cooler_max']  =  25.9
       oven_total_anormal['avg_accumulation']  ,oven_total_anormal['max_accumulation']  ,oven_total_anormal['min_accumulation']  =  accumulation_mean[5]  ,accumulation_max[5],accumulation_min[5]
       oven_total_anormal['avg_power_setup']  =  power_setup(accumulation_mean[5])

elif  oven  ==  '2C0':
       oven_total_anormal['cooler_max']  =  25.3
       oven_total_anormal['avg_accumulation']  ,oven_total_anormal['max_accumulation']  ,oven_total_anormal['min_accumulation']  =  accumulation_mean[6],accumulation_max[6],accumulation_min[6]
       oven_total_anormal['avg_power_setup']  =  power_setup(accumulation_mean[6])

elif  oven  ==  '2D0':
       oven_total_anormal['cooler_max']  =  26
       oven_total_anormal['avg_accumulation']  ,oven_total_anormal['max_accumulation']  ,oven_total_anormal['min_accumulation']  =  accumulation_mean[7],accumulation_max[7],accumulation_min[7]
       oven_total_anormal['avg_power_setup']  =  power_setup(accumulation_mean[7])

elif  oven  ==  '2E0':
       oven_total_anormal['cooler_max']  =  25.1
       oven_total_anormal['avg_accumulation']  ,oven_total_anormal['max_accumulation']  ,oven_total_anormal['min_accumulation']  =  accumulation_mean[8],accumulation_max[8],accumulation_min[8]
       oven_total_anormal['avg_power_setup']  =  power_setup(accumulation_mean[8])

elif  oven  ==  '2G0':
       oven_total_anormal['cooler_max']  =  26.3
       oven_total_anormal['avg_accumulation']  ,oven_total_anormal['max_accumulation']  ,oven_total_anormal['min_accumulation']  =  accumulation_mean[9],accumulation_max[9],accumulation_min[9]
       oven_total_anormal['avg_power_setup']  =  power_setup(accumulation_mean[9])
```

```
    #  過去22天的資料
    for  i  in  range(days,len(oven_total_anormal)):
            oven_total_anormal['number_sum'][i]  =  np.sum(oven_total_anormal['anomaly_total_number'][i-days:i])
            oven_total_anormal['number_max'][i]  =  np.max(oven_total_anormal['anomaly_total_number'][i-days:i])
            oven_total_anormal['number_min'][i]  =  np.min(oven_total_anormal['anomaly_total_number'][i-days:i])
            #找眾數
            vals,counts  =  np.unique(  oven_total_anormal[i-days:i][  oven_total_anormal['anomaly_total_number'][i-days:i]>0  ]['anomaly_total_number'],return_counts=True)
            try:
                    index  =  np.argmax(counts)
                    oven_total_anormal['number_mode'][i]  =  vals[index]
            except:  #  找不到眾數
                    oven_total_anormal['number_mode'][i]  =  0
            oven_total_anormal['days'][i]  =  np.sum(oven_total_anormal['anomaly_total_number'][i-days:i]  >0  )
    return  oven_total_anormal

def  rmse(y,yhat):
    return  np.sqrt(mean_squared_error(y,yhat))
```

二、程式執行步驟說明 – 分訓練集、測試集

以oven 2舉例，若是oven 1就把其中22都改成27即可

```
prediction_days  =  22

train_df  =  creat_feature(oven='1B0',days=prediction_days)[:-prediction_days]
test_df  =  creat_feature(oven='1B0',days=prediction_days)[-prediction_days:]

for  i  in  ['1C0','1D0','1E0','1G0','2B0','2C0','2D0','2E0','2G0']:
        training_df  =  creat_feature(oven=i,days=prediction_days)[:-prediction_days]
        testing_df  =  creat_feature(oven=i,days=prediction_days)[-prediction_days:]
        #將產生完新特徵後的資料上下連接
        train_df  =  pd.concat( [train_df,  training_df  ])
        test_df  =  pd.concat( [test_df,  testing_df  ])
```

```
train_df
```

date	anomaly_total_number	oven_id	number_sum	number_max	number_min	number_mode	days	oven_encoder	cooler_max	avg_accumulation	max_accumulation	min_accumulation	avg_power_setup
2021-12-27	2	1B0	NaN	NaN	NaN	NaN	NaN	6.555556	30.9	8160.736842	9430	2702	75.5
2021-12-28	0	1B0	NaN	NaN	NaN	NaN	NaN	6.555556	30.9	8160.736842	9430	2702	75.5
2021-12-29	0	1B0	NaN	NaN	NaN	NaN	NaN	6.555556	30.9	8160.736842	9430	2702	75.5
2021-12-30	0	1B0	NaN	NaN	NaN	NaN	NaN	6.555556	30.9	8160.736842	9430	2702	75.5
2021-12-31	0	1B0	NaN	NaN	NaN	NaN	NaN	6.555556	30.9	8160.736842	9430	2702	75.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...

二、程式執行步驟說明 – 機器學習模型實驗

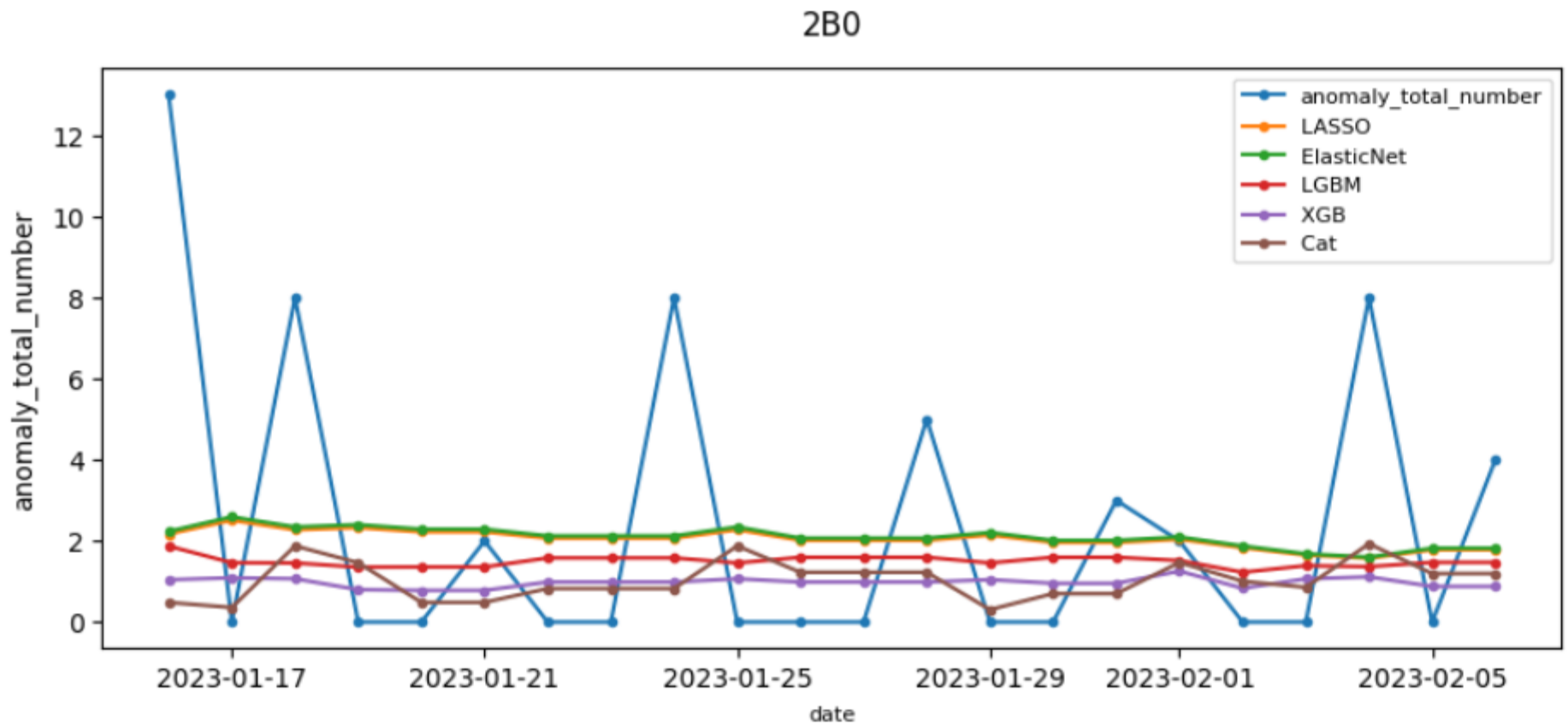
以oven 2舉例，若是oven 1就把其中22都改成27， ['2B0','2C0','2D0','2E0','2G0‘]改成 ['1B0','1C0','1D0','1E0','1G0’]即可

```
train_x = train_df.drop(['anomaly_total_number','oven_id'],axis=1)
train_y = train_df['anomaly_total_number']
lasso_preds, en_preds, lgb_preds, xgb_preds, cat_preds, actual = [],[],[],[],[],[]

#for i in ['1B0','1C0','1D0','1E0','1G0']:
for i in ['2B0','2C0','2D0','2E0','2G0']:
    print("============================ oven "+i+" ============================")
    test_1 = test_df[test_df['oven_id']==i]
    test_1_x = test_1.drop(['anomaly_total_number','oven_id'],axis=1)
    test_1_y = test_1['anomaly_total_number']

    # 執行KNN插補
    # KNNImputer 是一個用於填充缺失值的類， n_neighbors 參數設置為 3 表示將考慮每個缺失值周圍的 3 個最近鄰來進行填充。
    imputed_data = KNNImputer(n_neighbors=3).fit_transform(train_x)
    # imputed_data = train_x.fillna(0)

    if i == '1G0' or i == '2G0':      # 設定G爐預測值為0
        lasso_pred,en_pred,lgb_pred,xgb_pred,cat_pred=np.zeros(prediction_days),np.zeros(prediction_days),np.zeros(prediction_days),np.zeros(prediction_days),np.zeros(prediction_days)
    else:
        lasso_pred = Lasso().fit(imputed_data, train_y).predict(test_1_x)
        en_pred = ElasticNet().fit(imputed_data, train_y).predict(test_1_x)
        xgb_pred = XGBRegressor(learning_rate=0.01).fit(train_x, train_y).predict(test_1_x)
        lgb_pred = LGBMRegressor(learning_rate=0.01,max_depth=5).fit(train_x, train_y).predict(test_1_x)
        cat_pred = CatBoostRegressor(random_state=42,silent=True).fit(train_x, train_y).predict(test_1_x)
```

```
        lasso_preds.append(np.round(np.sum(lasso_pred)))
        en_preds.append(np.round(np.sum(en_pred)))
        lgb_preds.append(np.round(np.sum(lgb_pred)))
        xgb_preds.append(np.round(np.sum(xgb_pred)))
        cat_preds.append(np.round(np.sum(cat_pred)))

        actual.append(np.sum(test_1_y))

        print('Actual number :',actual[-1] ,'| lasso :',lasso_preds[-1],'| ElasticNet :', en_preds[-1],'| LGBM :',lgb_preds[-1],'| XGB :',xgb_preds[-1],'| Cat :',cat_preds[-1])
        plt.figure(figsize = (10,4), dpi = 100, linewidth = 2)
        plt.plot( test_1.index, test_1_y , 'p-', label= 'anomaly_total_number',marker='.')
        plt.plot( test_1.index , lasso_pred , 'p-', label= 'LASSO ',marker='.')
        plt.plot( test_1.index , en_pred , 'p-', label= 'ElasticNet ',marker='.')
        plt.plot( test_1.index , lgb_pred , 'p-', label= 'LGBM ',marker='.')
        plt.plot( test_1.index , xgb_pred , 'p-', label= 'XGB ',marker='.')
        plt.plot( test_1.index , cat_pred , 'p-', label= 'Cat ',marker='.')
        plt.title(i , x = 0.5, y = 1.03)
        plt.yticks(fontsize = 10)
        plt.xlabel("date", fontsize = 8, labelpad = 5)
        plt.ylabel("anomaly_total_number", fontsize = 10, labelpad = 5)
        plt.legend(loc = "best", fontsize = 8)
        plt.plot()

print("="*110)
print('lasso RMSE=',rmse(lasso_preds,actual))
print('ElasticNet RMSE=',rmse(en_preds,actual))
print('LGBM RMSE=',rmse(lgb_preds,actual))
print('XGB RMSE=',rmse(xgb_preds,actual))
print('Cat RMSE=',rmse(cat_preds,actual))
```

```
============================== oven 2B0 ==============================
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000266 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 211
[LightGBM] [Info] Number of data points in the train set: 3060, number of used features: 11
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
Actual number : 39 | lasso : 37.0 | ElasticNet : 39.0 | LGBM : 32.0 | XGB : 26.0 | Cat : 51.0
============================== oven 2G0 ==============================
Actual number : 0 | lasso : 0.0 | ElasticNet : 0.0 | LGBM : 0.0 | XGB : 0.0 | Cat : 0.0
======================================================================================================
lasso RMSE= 9.033271832508971
ElasticNet RMSE= 8.23407554009556
LGBM RMSE= 14.345731072343439
XGB RMSE= 18.88385553852814
Cat RMSE= 16.92926460304759
```



二、程式執行步驟說明 – 時間序列實驗

```
def  rmse(y,yhat):
    return  np.sqrt(mean_squared_error(y,yhat))

def  Moving_Average(  y  ,  window_size  ,  predict_period):
    train_y_df  =  pd.DataFrame(y)
    roll  =  train_y_df.rolling(window_size).mean().values
    ma  =  list(  roll.reshape(  len(y)  )  )
    for  i  in  range(predict_period):
        ma.append(np.mean(ma[-window_size:]))
    ma_pred  =  ma[-predict_period:]
    return  ma_pred

def  best_window_size(  valid_y  ,training_y,  predict_period  =  22):
    best_rmse  ,  best_window_size_1  =  100,  100
    best_diff  ,  best_window_size_2  =  100,  100

    for  window  in  range(2,22):
        valid_pred  =  Moving_Average(  y  =  training_y  ,  window_size  =  window  ,  predict_period  =  predict_period)
        #  RMSE  =  rmse(  valid_y  ,  valid_pred  )
        diff  =  abs(  np.sum(valid_y)  -  np.ceil(np.sum(valid_pred)  ))

        #  if  RMSE  <  best_rmse  :
        #      best_rmse  =  RMSE
        #      best_window_size_1  =  window

        if  diff  <  best_diff  :
            best_diff  =  diff
            best_window_size_2  =  window
```

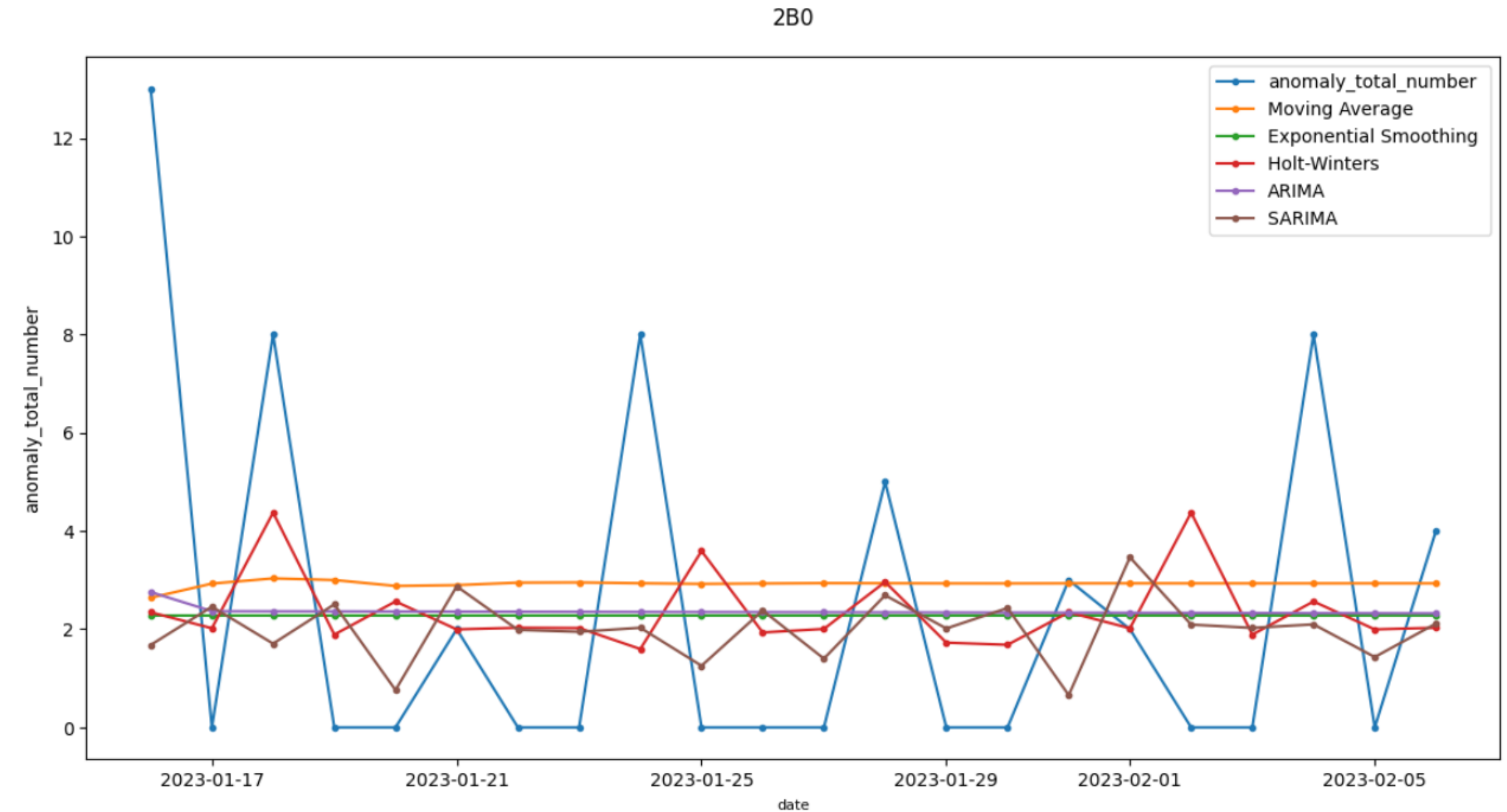
```python
def period_split(data, period):
    '''
    period : 以幾天作為一期
    data: 訓練資料
    '''
    period_data = []
    for j in range( int(len(data) /period) ,0,-1):
        period_data.append( np.sum( data[ len(data) -period*j : len(data) -period*(j-1) ]) )
    prediction_period = int(22/period)+1
    return prediction_period,period_data

def best_period_MA(train_y, valid_y, period_split_day,max_window):

    best_period,best_diff = 100,100
    best_window_diff , best_window_size_2 = 100,100
    for i in range(2,period_split_day):
        prediction_period, period_train_data = period_split(train_y,i)
        _ , period_valid_data = period_split(valid_y,i)

        for window in range(2,max_window):
            valid_pred = Moving_Average( y = period_train_data , window_size = window , predict_period = prediction_period)
            window_diff = abs( np.sum(period_valid_data) - np.ceil(np.sum(valid_pred) ))

            if window_diff < best_window_diff :
                best_window_diff = window_diff
                best_window_size_2 = window

        valid_pred = Moving_Average( period_train_data , window_size = best_window_size_2 , predict_period = prediction_period )
        diff = abs( np.sum(period_valid_data) - np.ceil(np.sum(valid_pred) ))
        if diff < best_diff :
            best_diff = diff
            best_period = i

    print("Period MA |",'best diff =',best_diff,'best period =',best_period,"best window size =",best_window_size_2)
    return best_period,best_window_size_2
```

```
def  period_predict(training_data,period_split_day,window_size):
    prediction_period,  period_train_data  =  period_split(training_data,period_split_day)
    prediction  =  Moving_Average(  y  =  period_train_data  ,  window_size  =  window_size  ,  predict_period  =  prediction_period)
    return  prediction

def  sarima_model(data,  Seasonality,frequency):
    model  =  pm.auto_arima(  data,  start_p=1,  start_q=1,
                                        information_criterion='aic',
                                        test='adf',              #  use  adftest  to  find  optimal  'd'
                                        max_p=10,  max_q=10,  #  maximum  p  and  q
                                        m=frequency,                              #  frequency  of  series
                                        d=None,                        #  let  model  determine  'd'
                                        seasonal=  Seasonality,          #  No  Seasonality
                                        start_P=0,
                                        D=1,
                                        trace=False,
                                        error_action='ignore',
                                        suppress_warnings=True,
                                        stepwise=True)
    pred  =  model.predict(n_periods=22,  return_conf_int=False)
    return  pred
```

```
#oven_2 = ['1B0','1C0','1D0','1E0','1G0']
oven_2 = ['2B0','2C0','2D0','2E0','2G0']
ma_preds ,holt_preds ,holt_winter_preds,period_preds,sarima_preds,arima_preds ,actual= [],[],[],[],[],[],[]

for i in oven_2:
        print("============================ oven "+i+" ============================")
        oven_df = df[df['oven_id']==i].sort_values(by='date',ascending=True)
        oven2_total_anormal = oven_df.groupby(['date'])['anomaly_total_number'].sum()
        x = oven2_total_anormal.index
        y =   oven2_total_anormal.values
        train_y = oven2_total_anormal.values[:-44]
        valid_y = oven2_total_anormal.values[-44:-22]
        test_y = oven2_total_anormal.values[-22:]

        test_x = oven2_total_anormal.index[-22:]
        actual.append(np.sum(test_y))

        # Moving Average
        best_window = best_window_size( valid_y ,train_y)
        MA_pred = Moving_Average( train_y , window_size = best_window , predict_period=22)
        ma_preds.append(np.ceil(np.sum(MA_pred)))

        # Exponential Smoothing (單指数平滑法)
        holt_pred = ExponentialSmoothing( y[:-22] ).fit().forecast(22)
        holt_preds.append(np.ceil(np.sum(holt_pred)))

        # holt's winter (三指数平滑法)
        holt_winter_pred = ExponentialSmoothing( y[:-22] , seasonal_periods=15, seasonal='add' ).fit().forecast(22)
        holt_winter_preds.append(np.ceil(np.sum(holt_winter_pred)))

        # 以多天為一期的 Moving Average
        period , window = best_period_MA(train_y, valid_y, period_split_day = 22,max_window = 11)
        period_pred = period_predict(training_data = y[:-22] , period_split_day = period, window_size = period)
        period_preds.append(np.ceil(np.sum(period_pred)))

        # ARIMA
        arima_pred = sarima_model( y[:-22] ,Seasonality = False ,frequency = 11)
        arima_preds.append(np.ceil(np.sum(arima_pred)))
```

```
        #  SARIMA
        sarima_pred  =  sarima_model(  y[:-22]  ,Seasonality  =  True,  frequency  =  11)
        sarima_preds.append(np.ceil(np.sum(sarima_pred)))


        print('Actual  number  :',actual[-1]  ,'|  Exponential  Smoothing  :',holt_preds[-1],'|  Holt-Winters  :',holt_winter_preds[-1],
                    '|  MA  :',ma_preds[-1]  ,'|  Period  MA  :',  period_preds[-1],
                    "|  ARIMA  :",arima_preds[-1],"|  SARIMA  :",sarima_preds[-1])

        plt.figure(figsize  =  (14,7),  dpi  =  100,  linewidth  =  2)
        plt.plot(  test_x,  test_y  ,  'p-',  label=  'anomaly_total_number',marker='.')
        plt.plot(  test_x  ,  MA_pred  ,  'p-',  label=  'Moving  Average',marker='.')
        plt.plot(  test_x  ,  holt_pred  ,  'p-',  label=  'Exponential  Smoothing  ',marker='.')
        plt.plot(  test_x  ,  holt_winter_pred  ,  'p-',  label=  'Holt-Winters  ',marker='.')
        plt.plot(  test_x  ,  arima_pred  ,  'p-',  label=  'ARIMA',marker='.')
        plt.plot(  test_x  ,  sarima_pred  ,  'p-',  label=  'SARIMA  ',marker='.')
        plt.title(i  ,  x  =  0.5,  y  =  1.03)
        plt.yticks(fontsize  =  10)
        plt.xlabel("date",  fontsize  =  8,  labelpad  =  5)
        plt.ylabel("anomaly_total_number",  fontsize  =  10,  labelpad  =  5)
        plt.legend(loc  =  "best",  fontsize  =  10)
        plt.plot()

print("="*110)
print('MA  RMSE=',rmse(ma_preds,actual))
print('Exponential  Smoothing  RMSE=',rmse(holt_preds,actual))
print('holt-winter  RMSE=',rmse(holt_winter_preds,actual))
print('Period  MA  RMSE=',rmse(period_preds,actual))
print('ARIMA  RMSE=',rmse(arima_preds,actual))
print('SARIMA  RMSE=',rmse(sarima_preds,actual))
```

```
MA RMSE= 7.1693793315739685
Exponential Smoothing RMSE= 4.1952353926806065
holt-winter RMSE= 3.794733192202055
Period MA RMSE= 20.174241001832016
ARIMA RMSE= 7.835815209663893
SARIMA RMSE= 11.636150566231086
```



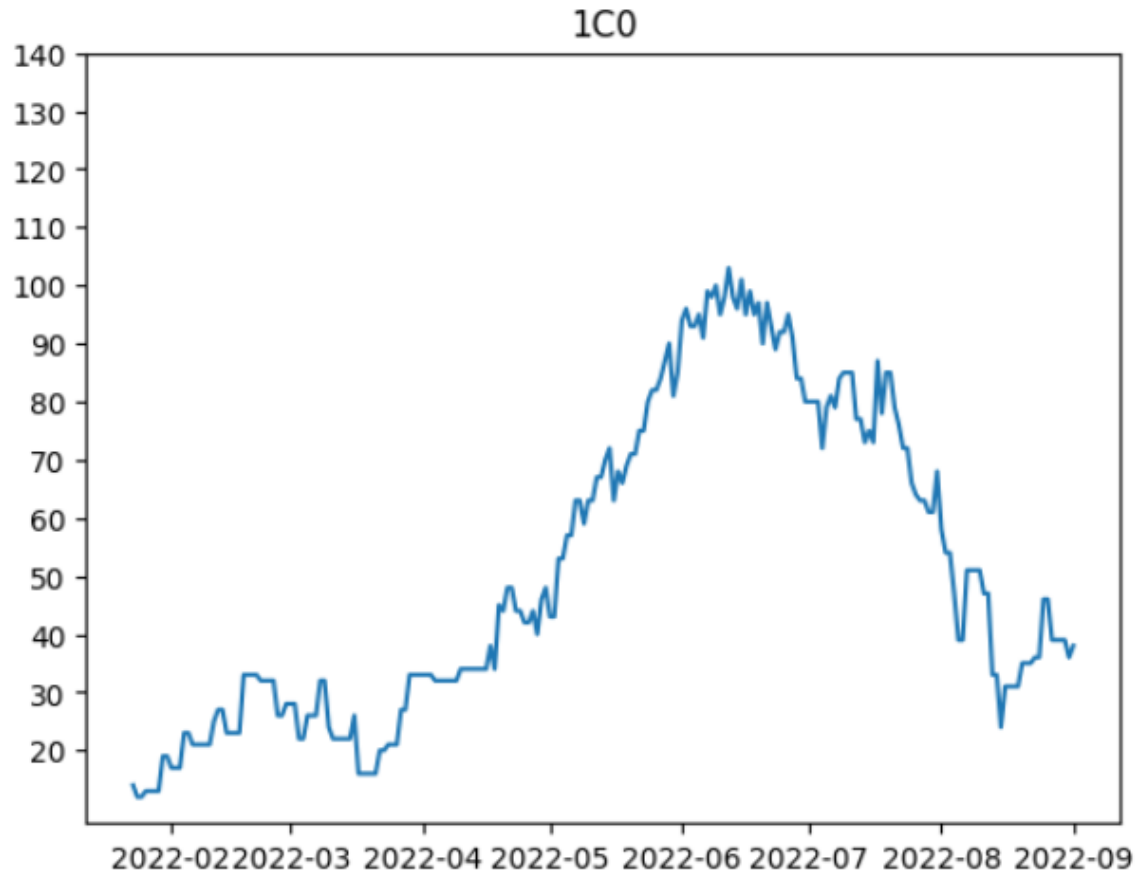
繪製趨勢圖

- 以1C0舉例，其他爐則改成該爐編號即可

二、程式執行步驟說明 – 繪製趨勢圖

以1C0舉例，其他爐則改成該爐編號即可

```
for  i  in  ['1C0']:
    plt.plot(train_df[train_df['oven_id']==i]['number_sum'])
    plt.title(i)
    plt.yticks([i  for  i  in  range(20,150,10)])
```



機器學習模型最終預測結果

- 以oven 1舉例，若是oven 2就把其中22都改成27、
- ['1B0','1C0','1D0','1E0','1G0’]改成 ['2B0','2C0','2D0','2E0','2G0‘]即可

二、程式執行步驟說明 – 機器學習模型最終預測結果

以1C0舉例，其他爐則改成該爐編號即可

```
def creat_feature(oven, days):
    oven_df = df[df['oven_id']== oven ].sort_values(by='date',ascending=True)
    oven2_total_anormal = oven_df.groupby(['date'])['anomaly_total_number'].sum()
    oven_total_anormal = pd.DataFrame(oven2_total_anormal)

    for i in range(27): # 產線一預測後27天，加入後27天的 row
        oven_total_anormal.loc[249+i] = np.nan

    oven_total_anormal['oven_id'] = oven
```

新增特徵時須也新增未來特徵預測，以便產出未來異常數量預測

```
    # 過去的資料
    oven_total_anormal['number_sum'] = np.nan
    oven_total_anormal['number_max'] = np.nan
    oven_total_anormal['number_min'] = np.nan
    oven_total_anormal['number_mode'] = np.nan
    oven_total_anormal['days'] = np.nan

    # 每個爐的平均損壞數量
    oven_total_anormal['oven_encoder'] = oven_total_anormal['anomaly_total_number'].sum() / np.sum(oven_total_anormal['anomaly_total_number'] >0 )

    # 個別爐的水冷板溫度、累積時數、Power setup
    if oven == '1B0':
        oven_total_anormal['cooler_max'] = 30.9
        oven_total_anormal['avg_accumulation'] ,oven_total_anormal['max_accumulation'] ,oven_total_anormal['min_accumulation'] = accumulation_mean[0] ,accumulation_max[0],accumulation_min[0]
        oven_total_anormal['avg_power_setup'] = power_setup(accumulation_mean[0])

    elif oven == '1C0':
        oven_total_anormal['cooler_max'] = 30.5
        oven_total_anormal['avg_accumulation'] ,oven_total_anormal['max_accumulation'] ,oven_total_anormal['min_accumulation'] = accumulation_mean[1],accumulation_max[1],accumulation_min[1]
        oven_total_anormal['avg_power_setup'] = power_setup(accumulation_mean[1])

    elif oven == '1D0':
        oven_total_anormal['cooler_max'] = 27.5
        oven_total_anormal['avg_accumulation'] ,oven_total_anormal['max_accumulation'] ,oven_total_anormal['min_accumulation'] = accumulation_mean[2],accumulation_max[2],accumulation_min[2]
        oven_total_anormal['avg_power_setup'] = power_setup(accumulation_mean[2])
```

```
elif  oven  ==  '1E0':
       oven_total_anormal['cooler_max']  =  27
       oven_total_anormal['avg_accumulation']  ,oven_total_anormal['max_accumulation']  ,oven_total_anormal['min_accumulation']  =  accumulation_mean[3],accumulation_max[3],accumulation_min[3]
       oven_total_anormal['avg_power_setup']  =  power_setup(accumulation_mean[3])

elif  oven  ==  '1G0':
       oven_total_anormal['cooler_max']  =  30.1
       oven_total_anormal['avg_accumulation']  ,oven_total_anormal['max_accumulation']  ,oven_total_anormal['min_accumulation']  =  accumulation_mean[4],accumulation_max[4],accumulation_min[4]
       oven_total_anormal['avg_power_setup']  =  power_setup(accumulation_mean[4])

elif  oven  ==  '2B0':
       oven_total_anormal['cooler_max']  =  25.9
       oven_total_anormal['avg_accumulation']  ,oven_total_anormal['max_accumulation']  ,oven_total_anormal['min_accumulation']  =  accumulation_mean[5]  ,accumulation_max[5],accumulation_min[5
       oven_total_anormal['avg_power_setup']  =  power_setup(accumulation_mean[5])

elif  oven  ==  '2C0':
       oven_total_anormal['cooler_max']  =  25.3
       oven_total_anormal['avg_accumulation']  ,oven_total_anormal['max_accumulation']  ,oven_total_anormal['min_accumulation']  =  accumulation_mean[6],accumulation_max[6],accumulation_min[6]
       oven_total_anormal['avg_power_setup']  =  power_setup(accumulation_mean[6])

elif  oven  ==  '2D0':
       oven_total_anormal['cooler_max']  =  26
       oven_total_anormal['avg_accumulation']  ,oven_total_anormal['max_accumulation']  ,oven_total_anormal['min_accumulation']  =  accumulation_mean[7],accumulation_max[7],accumulation_min[7]
       oven_total_anormal['avg_power_setup']  =  power_setup(accumulation_mean[7])

elif  oven  ==  '2E0':
       oven_total_anormal['cooler_max']  =  25.1
       oven_total_anormal['avg_accumulation']  ,oven_total_anormal['max_accumulation']  ,oven_total_anormal['min_accumulation']  =  accumulation_mean[8],accumulation_max[8],accumulation_min[8]
       oven_total_anormal['avg_power_setup']  =  power_setup(accumulation_mean[8])

elif  oven  ==  '2G0':
       oven_total_anormal['cooler_max']  =  26.3
       oven_total_anormal['avg_accumulation']  ,oven_total_anormal['max_accumulation']  ,oven_total_anormal['min_accumulation']  =  accumulation_mean[9],accumulation_max[9],accumulation_min[9]
       oven_total_anormal['avg_power_setup']  =  power_setup(accumulation_mean[9])
```

```
    #  過去的資料
    for  i  in  range(days,len(oven_total_anormal)):
            oven_total_anormal['number_sum'][i]  =  np.sum(oven_total_anormal['anomaly_total_number'][i-days:i])
            oven_total_anormal['number_max'][i]  =  np.max(oven_total_anormal['anomaly_total_number'][i-days:i])
            oven_total_anormal['number_min'][i]  =  np.min(oven_total_anormal['anomaly_total_number'][i-days:i])
            vals,counts  =  np.unique(  oven_total_anormal[i-days:i][  oven_total_anormal['anomaly_total_number'][i-days:i]>0  ]['anomaly_total_number'],return_counts=True)
            try:
                    index  =  np.argmax(counts)
                    oven_total_anormal['number_mode'][i]  =  vals[index]
            except:  #  找不到眾數
                    oven_total_anormal['number_mode'][i]  =  0
            oven_total_anormal['days'][i]  =  np.sum(oven_total_anormal['anomaly_total_number'][i-days:i]  >0  )
    return  oven_total_anormal

def  rmse(y,yhat):
    return  np.sqrt(mean_squared_error(y,yhat))
```

```
prediction_days = 27

train_df = creat_feature(oven='1B0',days=prediction_days)[:-prediction_days]
test_df = creat_feature(oven='1B0',days=prediction_days)[-prediction_days:]

for i in ['1C0','1D0','1E0','1G0','2B0','2C0','2D0','2E0','2G0']:
    training_df = creat_feature(oven=i,days=prediction_days)[:-prediction_days]
    testing_df = creat_feature(oven=i,days=prediction_days)[-prediction_days:]
    train_df = pd.concat( [train_df, training_df ])
    test_df = pd.concat( [test_df, testing_df ])
```

```
train_df
```

date	anomaly_total_number	oven_id	number_sum	number_max	number_min	number_mode	days	oven_encoder	cooler_max	avg_accumulation	max_accumulation	min_accumulation	avg_power_setup
2021-12-27	2.0	1B0	NaN	NaN	NaN	NaN	NaN	6.555556	30.9	8160.736842	9430	2702	75.5
2021-12-28	0.0	1B0	NaN	NaN	NaN	NaN	NaN	6.555556	30.9	8160.736842	9430	2702	75.5
2021-12-29	0.0	1B0	NaN	NaN	NaN	NaN	NaN	6.555556	30.9	8160.736842	9430	2702	75.5
2021-12-30	0.0	1B0	NaN	NaN	NaN	NaN	NaN	6.555556	30.9	8160.736842	9430	2702	75.5
2021-12-31	0.0	1B0	NaN	NaN	NaN	NaN	NaN	6.555556	30.9	8160.736842	9430	2702	75.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2023-02-02	0.0	2G0	0.0	0.0	0.0	0.0	0.0	3.000000	26.3	1195.425837	3464	0	47.0
2023-02-03	0.0	2G0	0.0	0.0	0.0	0.0	0.0	3.000000	26.3	1195.425837	3464	0	47.0
2023-02-04	0.0	2G0	0.0	0.0	0.0	0.0	0.0	3.000000	26.3	1195.425837	3464	0	47.0
2023-02-05	0.0	2G0	0.0	0.0	0.0	0.0	0.0	3.000000	26.3	1195.425837	3464	0	47.0
2023-02-06	0.0	2G0	0.0	0.0	0.0	0.0	0.0	3.000000	26.3	1195.425837	3464	0	47.0

3280 rows × 13 columns

```
train_x = train_df.drop(['anomaly_total_number','oven_id'],axis=1)
train_y = train_df['anomaly_total_number']
lasso_preds, en_preds, lgb_preds, xgb_preds, cat_preds, actual = [],[],[],[],[],[]

#lgb_params = {'n_estimators': 900, 'learning_rate': 0.08108549380709631, 'reg_lambda': 7.400420560246068, 'subsample': 0.6262267316079506, 'min_child
#xgb_params = {'n_estimators': 1300, 'learning_rate': 0.024658403720628174, 'booster': 'gbtree', 'lambda': 8.65039444200474, 'alpha': 7.8410257575344
#cat_params = {'iterations': 600, 'learning_rate': 0.08077046387065928, 'depth': 7, 'l2_leaf_reg': 9.566779128474014, 'bagging_temperature': 3.857521

for i in ['1B0','1C0','1D0','1E0','1G0']:
# for i in ['2B0','2C0','2D0','2E0','2G0']:
    print("============================== Test prediction on oven "+i+" ==============================")
    test_1 = test_df[test_df['oven_id']==i]
    test_1_x = test_1.drop(['anomaly_total_number','oven_id'],axis=1)

    # 執行KNN插補
    imputed_data =  KNNImputer(n_neighbors=3).fit_transform(train_x)
    # imputed_data = train_x.fillna(0)

    # if i == '1G0' or i == '2G0':     # 設定G爐預測值為0
    #     lasso_pred,en_pred,lgb_pred,xgb_pred=np.zeros(prediction_days),np.zeros(prediction_days),np.zeros(prediction_days),np.zeros(prediction_days)
    # else:
    #     lasso_pred = Lasso().fit(imputed_data, train_y).predict(test_1_x)
    #     en_pred = ElasticNet().fit(imputed_data, train_y).predict(test_1_x)
    #     lgb_pred = XGBRegressor(learning_rate=0.01).fit(train_x, train_y).predict(test_1_x)
    #     xgb_pred = LGBMRegressor(learning_rate=0.01,max_depth=5).fit(train_x, train_y).predict(test_1_x)

    lasso_pred = Lasso().fit(imputed_data, train_y).predict(test_1_x)
    en_pred = ElasticNet().fit(imputed_data, train_y).predict(test_1_x)
    xgb_pred = XGBRegressor(random_state= 42).fit(train_x, train_y).predict(test_1_x)
    lgb_pred = LGBMRegressor(random_state= 42).fit(train_x, train_y).predict(test_1_x)
    cat_pred = CatBoostRegressor(random_state= 42).fit(train_x, train_y).predict(test_1_x)
```

```
        lasso_pred = Lasso().fit(imputed_data, train_y).predict(test_1_x)
        en_pred = ElasticNet().fit(imputed_data, train_y).predict(test_1_x)
        xgb_pred = XGBRegressor(random_state= 42).fit(train_x, train_y).predict(test_1_x)
        lgb_pred = LGBMRegressor(random_state= 42).fit(train_x, train_y).predict(test_1_x)
        cat_pred = CatBoostRegressor(random_state= 42).fit(train_x, train_y).predict(test_1_x)

        lasso_preds.append(np.round(np.sum(lasso_pred)))
        en_preds.append(np.round(np.sum(en_pred)))
        lgb_preds.append(np.round(np.sum(lgb_pred)))
        xgb_preds.append(np.round(np.sum(xgb_pred)))
        cat_preds.append(np.round(np.sum(cat_pred)))
        print('Lasso :',lasso_preds[-1],'| ElasticNet :', en_preds[-1],'| LGBM :',lgb_preds[-1],'| XGB :',xgb_preds[-1],'| Cat :',cat_preds[-1])
result = pd.DataFrame()
result["Lasso"] = lasso_preds
result["ElasticNet"] = en_preds
result["LGBM"] = lgb_preds
result["XGB"] = xgb_preds
result["Cat"] = cat_preds
result.index = ['1B0','1C0','1D0','1E0','1G0']
print(result)
```

```
949:    learn: 2.0050411        total: 1.53s    remaining: 80.5ms
950:    learn: 2.0050310        total: 1.53s    remaining: 78.9ms

Lasso : 3.0 | ElasticNet : 2.0 | LGBM : 1.0 | XGB : 1.0 | Cat : 1.0
     Lasso  ElasticNet  LGBM    XGB   Cat
1B0   40.0        40.0  85.0  106.0  72.0
1C0   29.0        29.0  34.0   45.0  31.0
1D0   26.0        25.0  24.0   39.0  25.0
1E0   24.0        23.0  22.0   16.0  22.0
1G0    3.0         2.0   1.0    1.0   1.0
```

時間序列模型最終預測結果

二、程式執行步驟說明 – 時間序列模型最終預測結果

```
#oven_all = ['1B0','1C0','1D0','1E0','1G0','2B0','2C0','2D0','2E0','2G0']
oven_1 = ['1B0','1C0','1D0','1E0','1G0']
ma_preds ,holt_preds ,holt_winter_preds,period_preds,sarima_preds,arima_preds ,actual= [],[],[],[],[],[],[]

for i in oven_1:
    print("============================ oven "+i+" ============================")
    oven_df = df[df['oven_id']==i].sort_values(by='date',ascending=True)
    oven2_total_anormal = oven_df.groupby(['date'])['anomaly_total_number'].sum()
    x = oven2_total_anormal.index
    y = oven2_total_anormal.values
    train_y = oven2_total_anormal.values[:-27]
    valid_y = oven2_total_anormal.values[-27:]

    # Moving Average
    best_window = best_window_size(valid_y ,train_y)
    MA_pred = Moving_Average( y , window_size = best_window , predict_period=27)
    ma_preds.append(np.ceil(np.sum(MA_pred)))

    # Exponential Smoothing (單指數平滑法)
    holt_pred = ExponentialSmoothing( y ).fit().forecast(27)
    holt_preds.append(np.ceil(np.sum(holt_pred)))

    # holt's winter (三指數平滑法)
    holt_winter_pred = ExponentialSmoothing( y , seasonal_periods=15, seasonal='add' ).fit().forecast(27)
    holt_winter_preds.append(np.ceil(np.sum(holt_winter_pred)))

    # 以多天為一期的 Moving Average
    period , window = best_period_MA(train_y, valid_y, period_split_day = 27, max_window = 14)
    period_pred = period_predict(training_data = y , period_split_day = period, window_size = window)
    period_preds.append(np.ceil(np.sum(period_pred)))

    # ARIMA
    arima_pred = sarima_model( y ,Seasonality = False ,frequency = 14)
    arima_preds.append(np.ceil(np.sum(arima_pred)))
```

```
        #  SARIMA
        sarima_pred  =  sarima_model(  y  ,Seasonality  =  True,  frequency  =  14)
        sarima_preds.append(np.ceil(np.sum(sarima_pred)))


        print('Exponential  Smoothing  :',holt_preds[-1],'|  Holt-Winters  :',holt_winter_preds[-1],
                    '|  MA  :',ma_preds[-1]  ,'|  Period  MA  :',  period_preds[-1],
                    "|  ARIMA  :",arima_preds[-1],"|  SARIMA  :",sarima_preds[-1])


print("="*110)
result  =  pd.DataFrame()
result["Exponential  Smoothing"]  =  holt_preds
result["Holt-Winters"]  =  holt_winter_preds
result["MA"]  =  ma_preds
result["Period  MA"]  =  period_preds
result["ARIMA"]  =  arima_preds
result["SARIMA"]  =  sarima_preds
result.index  =  ['1B0','1C0','1D0','1E0','1G0']
print(result)
```

```
============================== oven 1B0 ==============================
Moving Average | best diff= 6.0 best window size= 26
Period MA | best diff = 0.0 best period = 11 best window size = 2
Exponential Smoothing : 59.0 | Holt-Winters : 59.0 | MA : 49.0 | Period MA : 75.0 | ARIMA : 60.0 | SARIMA : 53.0
============================== oven 1C0 ==============================
```

```
==============================================================================================================
     Exponential Smoothing  Holt-Winters    MA  Period MA  ARIMA  SARIMA
1B0                   59.0          59.0  49.0       75.0   60.0    53.0
1C0                   40.0          37.0  26.0       40.0   30.0    33.0
1D0                   46.0          45.0  26.0       28.0   40.0    61.0
1E0                   36.0          36.0  37.0       32.0   32.0    37.0
1G0                    2.0           2.0   0.0        0.0    2.0     1.0
```

```
#oven_all = ['1B0','1C0','1D0','1E0','1G0','2B0','2C0','2D0','2E0','2G0']
oven_2 = ['2B0','2C0','2D0','2E0','2G0']
ma_preds ,holt_preds ,holt_winter_preds,period_preds,sarima_preds,arima_preds ,actual= [],[],[],[],[],[],[]

for i in oven_2:
    print("============================ oven "+i+" ============================")
    oven_df = df[df['oven_id']==i].sort_values(by='date',ascending=True)
    oven2_total_anormal = oven_df.groupby(['date'])['anomaly_total_number'].sum()
    x = oven2_total_anormal.index
    y = oven2_total_anormal.values
    train_y = oven2_total_anormal.values[:-22]
    valid_y = oven2_total_anormal.values[-22:]

    # Moving Average
    best_window = best_window_size(valid_y ,train_y)
    MA_pred = Moving_Average( y , window_size = best_window , predict_period=22)
    ma_preds.append(np.ceil(np.sum(MA_pred)))

    # Exponential Smoothing (單指数平滑法)
    holt_pred = ExponentialSmoothing( y ).fit().forecast(22)
    holt_preds.append(np.ceil(np.sum(holt_pred)))

    # holt's winter (三指数平滑法)
    holt_winter_pred = ExponentialSmoothing( y , seasonal_periods=15, seasonal='add' ).fit().forecast(22)
    holt_winter_preds.append(np.ceil(np.sum(holt_winter_pred)))

    # 以多天為一期的 Moving Average
    period , window = best_period_MA(train_y, valid_y, period_split_day = 22, max_window = 14)
    period_pred = period_predict(training_data = y , period_split_day = period, window_size = window)
    period_preds.append(np.ceil(np.sum(period_pred)))

    # ARIMA
    arima_pred = sarima_model( y ,Seasonality = False ,frequency = 14)
    arima_preds.append(np.ceil(np.sum(arima_pred)))
```

```
        #  SARIMA
        sarima_pred  =  sarima_model(  y  ,Seasonality  =  True,  frequency  =  14)
        sarima_preds.append(np.ceil(np.sum(sarima_pred)))


        print('Exponential  Smoothing  :',holt_preds[-1],'|  Holt-Winters  :',holt_winter_preds[-1],
                  '|  MA  :',ma_preds[-1]  ,'|  Period  MA  :',  period_preds[-1],
                  "|  ARIMA  :",arima_preds[-1],"|  SARIMA  :",sarima_preds[-1])


print("="*110)
result  =  pd.DataFrame()
result["Exponential  Smoothing"]  =  holt_preds
result["Holt-Winters"]  =  holt_winter_preds
result["MA"]  =  ma_preds
result["Period  MA"]  =  period_preds
result["ARIMA"]  =  arima_preds
result["SARIMA"]  =  sarima_preds
result.index  =  ['2B0','2C0','2D0','2E0','2G0']
print(result)
```

```
============================== oven 2B0 ==============================
Moving Average | best diff= 17.0 best window size= 4
Period MA | best diff = 0.0 best period = 11 best window size = 10
Exponential Smoothing : 51.0 | Holt-Winters : 50.0 | MA : 54.0 | Period MA : 60.0 | ARIMA : 50.0 | SARIMA : 44.0
============================== oven 2C0 ==============================

==============================================================================================================
     Exponential Smoothing  Holt-Winters    MA  Period MA  ARIMA  SARIMA
2B0                   51.0          50.0  54.0       60.0   50.0    44.0
2C0                   47.0          46.0  56.0       59.0   47.0    52.0
2D0                   46.0          46.0  45.0       51.0   42.0    52.0
2E0                   38.0          38.0  46.0       21.0   27.0    42.0
2G0                    1.0           1.0   0.0        0.0    1.0     1.0
```