

Dacon Competition Code Explanation

Mission 11.

에너지 빅데이터 활용 데이터 - 사이언스 콘테스트

20191028 3rd price

Team Don't_overfit

1. 라이브러리 및 데이터 Library & Data
2. 데이터 전처리 Data Cleansing & Pre-Processing
3. 탐색적 자료분석 Exploratory Data Analysis
4. 변수 선택 및 모델 구축 Feature Engineering & Initial Modeling
5. 모델 학습 및 검증 Model Tuning & Evaluation
6. 결과 및 결론 Conclusion & Discussion

```
[ ] 1 from google.colab import drive
    2 drive.mount('/content/drive')
    3 import pandas as pd # 데이터 분석
    4 import numpy as np # 행렬 연산
    5 import os # 경로 설정
    6 import tqdm # 진행상황 파악
    7 import datetime # 날짜타입 사용
    8 import random # 시드 제어
    9 from collections import defaultdict # dict 자료구조
   10
   11 import matplotlib.pyplot as plt
   12 import seaborn as sns
   13
   14 from sklearn.model_selection import KFold # 일반화
   15
   16 import lightgbm as lgb
   17
   18 # install
   19 !pip install workalendar
   20 from workalendar.asia import SouthKorea # 한국의 공휴일, version : 1.1.1
   21
   22 path = '/content/drive/My Drive/lldacon/최종 제출물/'
   23 os.chdir(path)
   24 path = '../data/'
   25 os.chdir(path)
   26
   27 #사용한 데이터 불러오기
   28 test = pd.read_csv("test.csv") # 대회 데이터
   29 sub = pd.read_csv("submission.csv") # 대회 데이터
   30 weather = pd.read_csv("weather_hour.csv", encoding='cp949') # 대회 데이터
```

1. 라이브러리 및 데이터 Library & Data
2. 데이터 전처리 Data Cleansing & Pre-Processing
3. 탐색적 자료분석 Exploratory Data Analysis
4. 변수 선택 및 모델 구축 Feature Engineering & Initial Modeling
5. 모델 학습 및 검증 Model Tuning & Evaluation
6. 결과 및 결론 Conclusion & Discussion



```
1 # 시드를 고정하는 함수
2 def seed_everything(seed=0):
3     random.seed(seed)
4     np.random.seed(seed)

1 # 온도데이터는 일시와 기온만 사용
2 weather['일시'] = pd.to_datetime(weather['일시'])
3 weather = weather.iloc[:, 1:3]
4 weather.columns = ['Time', 'temp']
5
6 # 16, 17, 18년 공휴일
7 holidays = pd.concat([
8     pd.Series(np.array(SouthKorea().holidays(2018))[:, 0]),
9     pd.Series(np.array(SouthKorea().holidays(2017))[:, 0]),
10    pd.Series(np.array(SouthKorea().holidays(2016))[:, 0])]).reset_index(drop=True)
```



```
1 def merge(train, col, hour=True, use_temp=True):
2     temp = train[['Time', col]].rename(columns={col: 'target'})
3     temp['Time'] = pd.to_datetime(temp['Time'])
4     temp = temp[temp['Time'] >= '2017-11-23'].reset_index(drop=True)
5     temp = temp.loc[temp.index[temp['target'].astype(str) != 'nan'][0:]].reset_index(drop=True)
6
7     temp['Time'] = pd.to_datetime(temp['Time'])
8     temp['month'] = temp['Time'].dt.month
9     temp['week'] = temp['Time'].dt.week
10    temp['weekday'] = temp['Time'].dt.weekday
11    temp['day'] = temp['Time'].dt.day
12    temp['hour'] = temp['Time'].dt.hour
13    temp['holiday'] = temp['Time'].dt.date.isin(holidays).astype(int)
14    temp['weekend'] = temp['weekday'].map({0:0, 1:0, 2:0, 3:0, 4:0, 5:1, 6:1})
15    temp['is_holiday'] = (temp['weekend'] + temp['holiday']).map({0:0, 1:1, 2:1})
16    temp['date'] = pd.to_datetime(temp['Time'].dt.date)
17    temp['tomorrow'] = pd.to_datetime(temp['date']) + datetime.timedelta(days=1)
18
19    for i in range(24):
20        temp.loc[temp['hour']==i, 'target'] = temp.loc[temp['hour']==i, 'target'].clip(
21            0, temp.loc[temp['hour']==i, 'target'].mean() + 3*temp.loc[temp['hour']==i, 'target'].std())
22
```



변수

- Time으로부터 생성하는 변수 : month, week, weekday, day, hour
- weekend와 holiday를 이용하여, 휴일을 정의
- 직전주차에 사용한 전력량의 std와 mean을 변수로 사용(only hour predict) -> 모델에 추세에 대한 정보를 부여하는 목적
- target 변수의 outlier를 제거하기 위해 평균을 기준으로 3sigma 외의 값들 clipping



Case

include temp and hour predict

exclude temp and hour predict

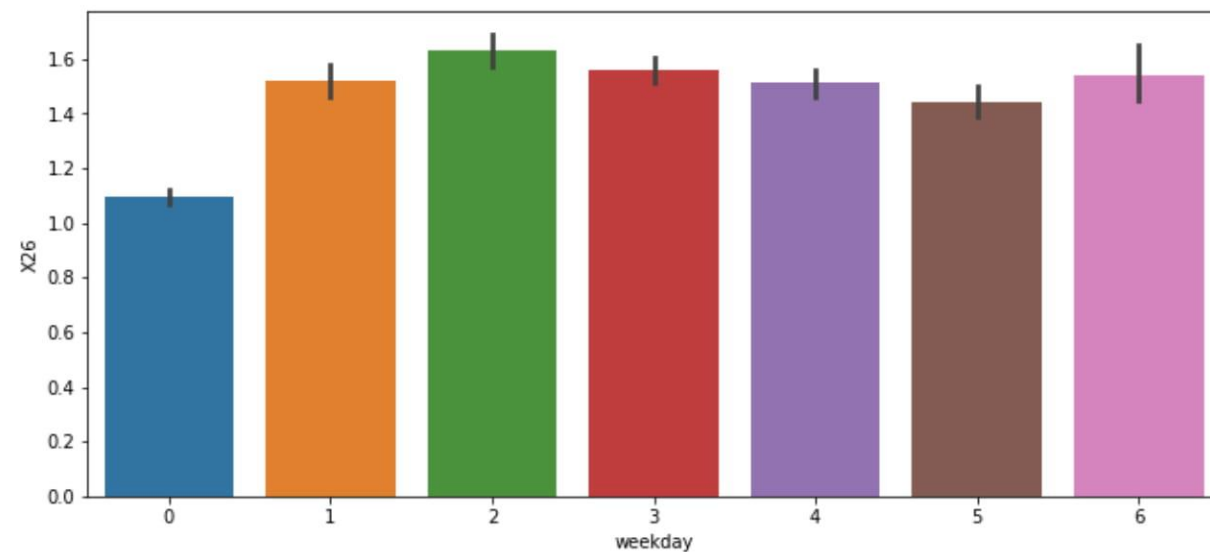
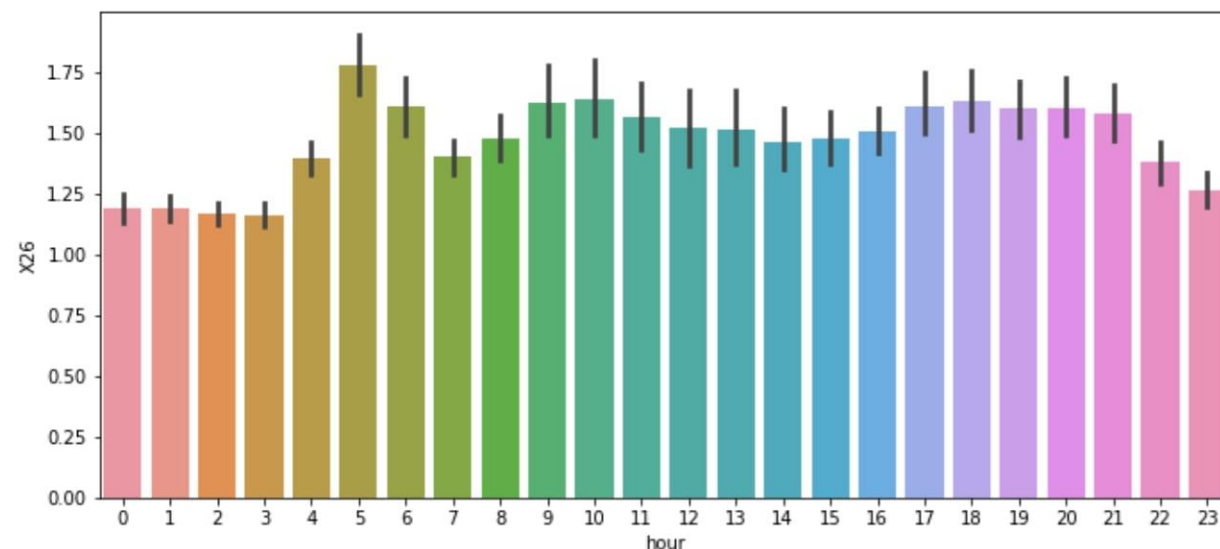
exclude temp and day, month predict

```
23     if hour:
24         temp2 = pd.DataFrame(pd.date_range('20180701', '20181201', freq='h'),
25                               columns=['Time']).iloc[:24, :]
26     else:
27         temp2 = pd.DataFrame(pd.date_range('20180701', '20181201', freq='h'),
28                               columns=['Time']).iloc[:-1, :]
29     temp2['Time'] = pd.to_datetime(temp2['Time'])
30     temp2['month'] = temp2['Time'].dt.month
31     temp2['week'] = temp2['Time'].dt.week
32     temp2['weekday'] = temp2['Time'].dt.weekday
33     temp2['day'] = temp2['Time'].dt.day
34     temp2['hour'] = temp2['Time'].dt.hour
35     temp2['holiday'] = temp2['Time'].dt.date.isin(holidays).astype(int)
36     temp2['weekend'] = temp2['weekday'].map({0:0, 1:0, 2:0, 3:0, 4:0, 5:1, 6:1})
37     temp2['is_holiday'] = (temp2['weekend'] + temp2['holiday']).map({0:0, 1:1, 2:1})
38     temp2['date'] = pd.to_datetime(temp2['Time'].dt.date)
39     temp2['tomorrow'] = pd.to_datetime(temp2['date']) + datetime.timedelta(days=1)
40
41
42     if hour:
43         temp['next_week'] = temp['week']+1
44         temp2['next_week'] = temp2['week']+1
45
46         temp_dict = temp.groupby('next_week')['target'].mean()
47         temp['next_week_mean'] = temp['week'].map(temp_dict)
48         temp2['next_week_mean'] = temp2['week'].map(temp_dict)
49         temp_dict = temp.groupby('next_week')['target'].std()
50         temp['next_week_std'] = temp['week'].map(temp_dict)
51         temp2['next_week_std'] = temp2['week'].map(temp_dict)
52
53     if use_temp:
54         temp = pd.merge(temp, weather, how='left', on='Time')
55         temp2 = pd.merge(temp2, weather, how='left', on='Time')
56
57     return temp.dropna().reset_index(drop=True), temp2
58 else:
59     return temp.dropna().reset_index(drop=True), temp2
```

1. 라이브러리 및 데이터 Library & Data
2. 데이터 전처리 Data Cleansing & Pre-Processing
3. 탐색적 자료분석 Exploratory Data Analysis
4. 변수 선택 및 모델 구축 Feature Engineering & Initial Modeling
5. 모델 학습 및 검증 Model Tuning & Evaluation
6. 결과 및 결론 Conclusion & Discussion


```
1 plt.subplots(1, 2, figsize=(25, 5))
2
3 plt.subplot(1, 2, 1)
4 temp_df = test[['Time', 'X26']].dropna()
5 temp_df['hour'] = pd.to_datetime(temp_df['Time']).dt.hour
6 sns.barplot(x='hour', y='X26', data=temp_df)
7
8 plt.subplot(1, 2, 2)
9 temp_df = test[['Time', 'X26']].dropna()
10 temp_df['weekday'] = pd.to_datetime(temp_df['Time']).dt.weekday
11 sns.barplot(x='weekday', y='X26', data=temp_df)
12 plt.show()
```

Time 별 전력사용량 변화 위주로 관찰



1. 라이브러리 및 데이터 Library & Data
2. 데이터 전처리 Data Cleansing & Pre-Processing
3. 탐색적 자료분석 Exploratory Data Analysis
4. 변수 선택 및 모델 구축 Feature Engineering & Initial Modeling
5. 모델 학습 및 검증 Model Tuning & Evaluation
6. 결과 및 결론 Conclusion & Discussion



Code



CellToolbar

```
1 submit_dict = defaultdict()
2 lgb_params = {
3     'objective': 'regression',
4     'boosting_type': 'gbdt',
5     'metric': 'rmse',
6     'n_jobs': -1,
7     'learning_rate': 0.03,
8     'num_leaves': 2**9,
9     'max_depth': -1,
10    'tree_learner': 'serial',
11    'min_child_weight': 5,
12    'subsample': 0.7,
13    'reg_alpha': 0.2,
14    'reg_lambda': 0.2,
15    'verbose': -1,
16    'seed': 0
17 }
18
```

Modeling

- 각각의 모델을 만들어 총 200개의 모델 생성
- 베이스모델은 xgboost로 시작
- 속도가 느렸기 때문에, lightgbm으로 변환 (성능의 상승)
 - > xgboost의 경우 overfit의 문제가 있었을 것이라 추정
- kfold, loocv, validation subset 구축 등의 모든 방법이 cv==lb가 맞지 않는 경우 발생
- LB를 기준으로 파라미터 조절을 수행
- 보다 robust한 모델을 구축하기 위해, smape metrics 에서 weight가 강한 hour 예측 모델은 온도 포함/온도 미포함 결과를 앙상블하고 시드앙상블 수행
- 결론적으로 시드앙상블을 안한 모델의 private score가 좋았습니다.

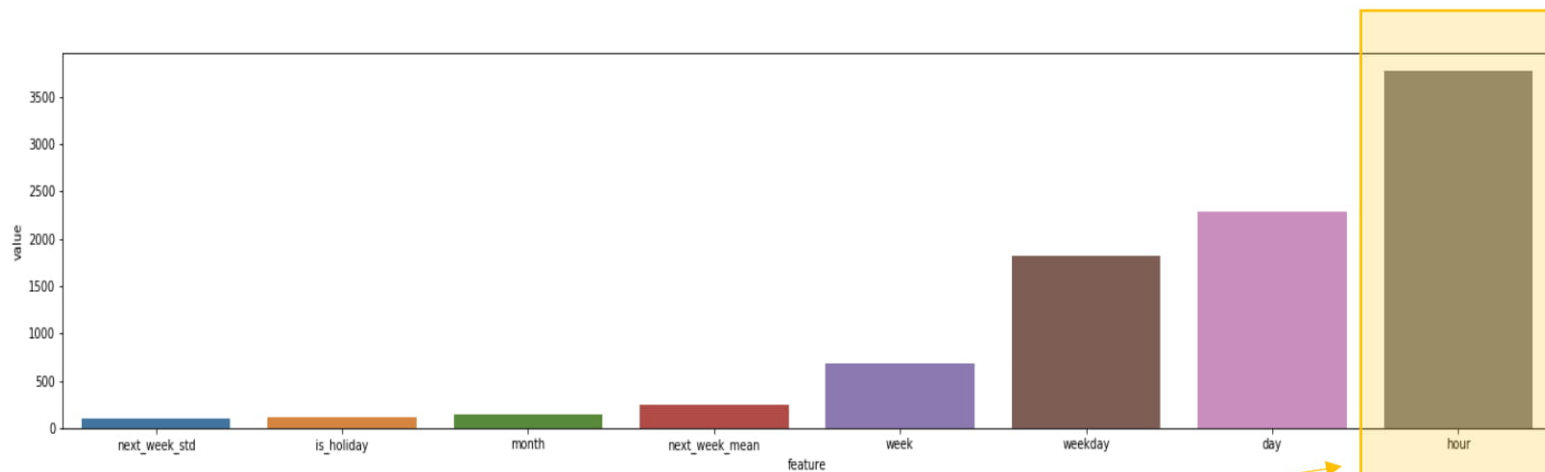


```
19 for idx in tqdm.tqdm_notebook(list(range(200))):
20     hours=True
21     using_temp=True
22     train_df, test_df = merge(test, sub['meter_id'][idx], hour=hours, use_temp=using_temp)
23     glo_pred1 = np.zeros(len(test_df))
24
25     for SEED in [42, 43, 44, 45, 46]:
26         lgb_params['seed'] = SEED
27         pred1 = np.zeros(len(test_df))
28
29         feature = [i for i in train_df.columns if i not in ['Time', 'target', 'holiday', 'weekend', 'date', 'tomorrow', 'next_week']]
30         kf = KFold(n_splits=5, random_state=SEED, shuffle=True)
31
32         for trn_idx, val_idx in kf.split(train_df):
33             tt = lgb.Dataset(train_df.loc[trn_idx, feature], train_df.loc[trn_idx, ['target']])
34             vv = lgb.Dataset(train_df.loc[val_idx, feature], train_df.loc[val_idx, ['target']])
35             model = lgb.train(lgb_params, tt, 100, valid_sets=[tt, vv], early_stopping_rounds=50, verbose_eval=0)
36             pred1 += model.predict(test_df[feature])/5
37         pred1[pred1<0] = train_df['target'].min()
38         glo_pred1 += pred1/5
39
40     using_temp=False
41     train_df, test_df = merge(test, sub['meter_id'][idx], hour=hours, use_temp=using_temp)
42     glo_pred2 = np.zeros(len(test_df))
43
44     for SEED in [42, 43, 44, 45, 46]:
45         lgb_params['seed'] = SEED
46         pred2 = np.zeros(len(test_df))
47
48         feature = [i for i in train_df.columns if i not in ['Time', 'target', 'holiday', 'weekend', 'date', 'tomorrow', 'next_week']]
49         kf = KFold(n_splits=5, random_state=SEED, shuffle=True)
50
51         for trn_idx, val_idx in kf.split(train_df):
52             tt = lgb.Dataset(train_df.loc[trn_idx, feature], train_df.loc[trn_idx, ['target']])
53             vv = lgb.Dataset(train_df.loc[val_idx, feature], train_df.loc[val_idx, ['target']])
54             model = lgb.train(lgb_params, tt, 200, valid_sets=[tt, vv], early_stopping_rounds=50, verbose_eval=0)
55             pred2 += model.predict(test_df[feature])/5
56         pred2[pred2<0] = train_df['target'].min()
57         glo_pred2 += pred2/5
58     submit_dict[idx] = glo_pred1*0.6 + glo_pred2*0.4
59
60 submit_df = pd.DataFrame(submit_dict)
61 submit_df.columns = sub['meter_id']
62 submit_df = submit_df.T.reset_index()
63 submit_df.columns = sub.columns[:25]
64 submit_df_hour = submit_df.copy()
```



```
18 for idx in tqdm.tqdm_notebook(list(range(200))):
19     hours=False
20     train_df, test_df = merge(test, sub['meter_id'][idx], hour=hours)
21     glo_pred = np.zeros(len(test_df))
22
23     for SEED in [42, 43, 44, 45, 46]:
24         lgb_params['seed'] = SEED
25         pred = np.zeros(len(test_df))
26
27         feature = [i for i in train_df.columns if i not in ['Time', 'target', 'holiday', 'weekend', 'date', 'tomorrow']]
28         kf = KFold(n_splits=5, random_state=SEED, shuffle=True)
29
30         for trn_idx, val_idx in kf.split(train_df):
31             tt = lgb.Dataset(train_df.loc[trn_idx, feature], train_df.loc[trn_idx, ['target']])
32             vv = lgb.Dataset(train_df.loc[val_idx, feature], train_df.loc[val_idx, ['target']])
33             model = lgb.train(lgb_params, tt, 200, valid_sets=[tt, vv], early_stopping_rounds=50, verbose_eval=0)
34             pred += model.predict(test_df[feature])/5
35             pred[pred<0] = train_df['target'].min()
36             glo_pred += pred/5
37         submit_dict[idx] = glo_pred
38
39 submit_df = pd.concat([pd.DataFrame(submit_dict).loc[:23],
40                         pd.concat([pd.DataFrame([j for i in range(10) for j in str(i) * 24], columns=['house']),
41                                   pd.DataFrame(submit_dict).loc[:239]], 1).groupby('house').sum().reset_index(drop=True),
42                         pd.concat([test_df['Time'].dt.to_period('m'),
43                                   pd.DataFrame(submit_dict)], 1).groupby('Time').sum().reset_index(drop=True)])],
44                        axis=1)
45 submit_df.columns = sub['meter_id']
46 submit_df = submit_df.T.reset_index()
47 submit_df.columns = sub.columns
48 submit_df_day = submit_df.copy()
```

1. 라이브러리 및 데이터 Library & Data
2. 데이터 전처리 Data Cleansing & Pre-Processing
3. 탐색적 자료분석 Exploratory Data Analysis
4. 변수 선택 및 모델 구축 Feature Engineering & Initial Modeling
5. 모델 학습 및 검증 Model Tuning & Evaluation
6. 결과 및 결론 Conclusion & Discussion



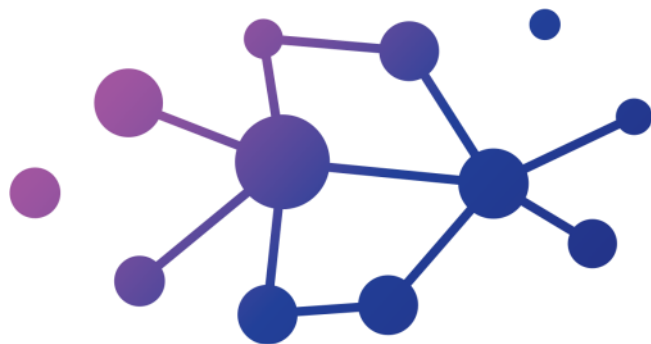
* hour가 가장 높은 영향력을 지님

* hour를 기반으로 여러가지 변수를 만들고 싶었으나, 대부분 overfit 요소가 되었습니다.

+ 각각의 경우에 보다 세밀하게 outlier를 선정하는 것을 생각해 보았습니다.

+ raw data를 학습하는 과정에서 std, mean, quantile 등 다양한 통계량을 통해서 비슷한 meter_id를 탐지하고, 비슷한 meter_id 별로 예측하는 것을 생각해 보았습니다.

THANK YOU



대회 참가해보기