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 결언 Conclusion & Discussion

jupyter ## 1. Import Library & Data

Logout

File Edit View Insert Cell Kernel Help Python 3

Code CellToolbar

[ ]

```
from google.colab import drive
drive.mount('/content/drive')
import pandas as pd # 데이터 분석
import numpy as np # 행렬 연산
import os # 경로 설정
import tqdm # 진행상황 파악
import datetime # 날짜타입 사용
import random # 시드 제어
from collections import defaultdict # dict 자료구조

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import KFold # 일반화

import lightgbm as lgb

# install
!pip install workalendar
from workalendar.asia import SouthKorea # 한국의 공휴일, version : 1.1.1

path = '/content/drive/My Drive/11dacon/최종 제출물/'
os.chdir(path)
path = '../data/'
os.chdir(path)

#사용한 데이터 불러오기
test = pd.read_csv("test.csv") # 대회 데이터
sub = pd.read_csv("submission.csv") # 대회 데이터
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 결언 Conclusion & Discussion

```python
# 시드를 고정하는 함수
def seed_everything(seed=0):
    random.seed(seed)
    np.random.seed(seed)
```

```python
# 온도데이터는 일시와 기온만 사용
weather['일시'] = pd.to_datetime(weather['일시'])
weather = weather.iloc[:, 1:3]
weather.columns = ['Time', 'temp']

# 16, 17, 18년 공휴일
holidays = pd.concat([
    pd.Series(np.array(SouthKorea().holidays(2018))[:, 0]),
    pd.Series(np.array(SouthKorea().holidays(2017))[:, 0]),
    pd.Series(np.array(SouthKorea().holidays(2016))[:, 0])]).reset_index(drop=True)
```

```python
def merge(train, col, hour=True, use_temp=True):
    temp = train[['Time', col]].rename(columns={col:'target'})
    temp['Time'] = pd.to_datetime(temp['Time'])
    temp = temp[temp['Time']>='2017-11-23'].reset_index(drop=True)
    temp = temp.loc[temp.index[temp['target'].astype(str)!='nan'][0]:].reset_index(drop=True)

    temp['Time'] = pd.to_datetime(temp['Time'])
    temp['month'] = temp['Time'].dt.month
    temp['week'] = temp['Time'].dt.week
    temp['weekday'] = temp['Time'].dt.weekday
    temp['day'] = temp['Time'].dt.day
    temp['hour'] = temp['Time'].dt.hour
    temp['holiday'] = temp['Time'].dt.date.isin(holidays).astype(int)
    temp['weekend'] = temp['weekday'].map({0:0, 1:0, 2:0, 3:0, 4:0, 5:1, 6:1})
    temp['is_holiday'] = (temp['weekend'] + temp['holiday']).map({0:0, 1:1, 2:1})
    temp['date'] = pd.to_datetime(temp['Time'].dt.date)
    temp['tomorrow'] = pd.to_datetime(temp['date']) + datetime.timedelta(days=1)

    for i in range(24):
        temp.loc[temp['hour']==i, 'target'] = temp.loc[temp['hour']==i, 'target'].clip(
            0, temp.loc[temp['hour']==i, 'target'].mean() + 3*temp.loc[temp['hour']==i, 'target'].std())
```

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

```
    if hour:
        temp2 = pd.DataFrame(pd.date_range('20180701', '20181201', freq='h'),
                             columns=['Time']).iloc[:24, :]
    else:
        temp2 = pd.DataFrame(pd.date_range('20180701', '20181201', freq='h'),
                             columns=['Time']).iloc[:-1]
    temp2['Time'] = pd.to_datetime(temp2['Time'])
    temp2['month'] = temp2['Time'].dt.month
    temp2['week'] = temp2['Time'].dt.week
    temp2['weekday'] = temp2['Time'].dt.weekday
    temp2['day'] = temp2['Time'].dt.day
    temp2['hour'] = temp2['Time'].dt.hour
    temp2['holiday'] = temp2['Time'].dt.date.isin(holidays).astype(int)
    temp2['weekend'] = temp2['weekday'].map({0:0, 1:0, 2:0, 3:0, 4:0, 5:1, 6:1})
    temp2['is_holiday'] = (temp2['weekend'] + temp2['holiday']).map({0:0, 1:1, 2:1})
    temp2['date'] = pd.to_datetime(temp2['Time'].dt.date)
    temp2['tomorrow'] = pd.to_datetime(temp2['date']) + datetime.timedelta(days=1)


    if hour:
        temp['next_week'] = temp['week']+1
        temp2['next_week'] = temp2['week']+1

        temp_dict = temp.groupby('next_week')['target'].mean()
        temp['next_week_mean'] = temp['week'].map(temp_dict)
        temp2['next_week_mean'] = temp2['week'].map(temp_dict)
        temp_dict = temp.groupby('next_week')['target'].std()
        temp['next_week_std'] = temp['week'].map(temp_dict)
        temp2['next_week_std'] = temp2['week'].map(temp_dict)

        if use_temp:
            temp = pd.merge(temp, weather, how='left', on='Time')
            temp2 = pd.merge(temp2, weather, how='left', on='Time')

        return temp.dropna().reset_index(drop=True), temp2
    else:
        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 결언 Conclusion & Discussion

jupyter ## 3. Exploratory Data Analysis

Logout

File Edit View Insert Cell Kernel Help Python 3

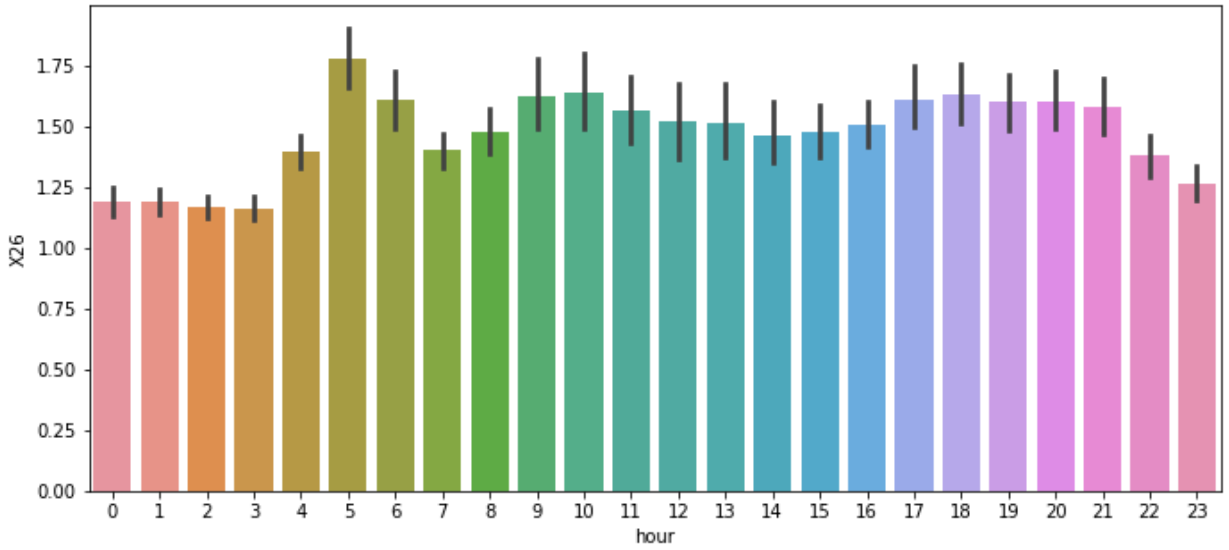
Code CellToolbar

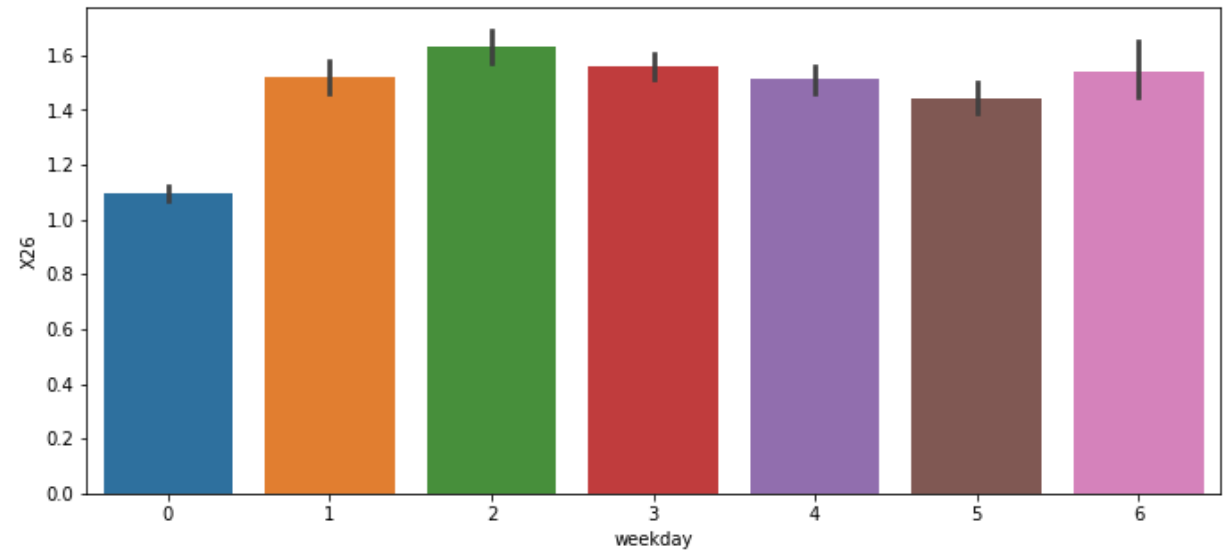
```
plt.subplots(1, 2, figsize=(25, 5))

plt.subplot(1, 2, 1)
temp_df = test[['Time', 'X26']].dropna()
temp_df['hour'] = pd.to_datetime(temp_df['Time']).dt.hour
sns.barplot(x='hour', y='X26', data=temp_df)

plt.subplot(1, 2, 2)
temp_df = test[['Time', 'X26']].dropna()
temp_df['weekday'] = pd.to_datetime(temp_df['Time']).dt.weekday
sns.barplot(x='weekday', y='X26', data=temp_df)
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 결언 Conclusion & Discussion

```python
submit_dict = defaultdict()
lgb_params = {
        'objective':'regression',
        'boosting_type':'gbdt',
        'metric':'rmse',
        'n_jobs':-1,
        'learning_rate':0.03,
        'num_leaves': 2**9,
        'max_depth':-1,
        'tree_learner':'serial',
        'min_child_weight':5,
        'subsample':0.7,
        'reg_alpha':0.2,
        'reg_lambda':0.2,
        'verbose':-1,
        'seed': 0
    }
```

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.

```
for idx in tqdm.tqdm_notebook(list(range(200))):
    hours=True
    using_temp=True
    train_df, test_df = merge(test, sub['meter_id'][idx], hour=hours, use_temp=using_temp)
    glo_pred1 = np.zeros(len(test_df))

    for SEED in [42, 43, 44, 45, 46]:
        lgb_params['seed'] = SEED
        pred1 = np.zeros(len(test_df))

        feature = [i for i in train_df.columns if i not in ['Time', 'target', 'holiday', 'weekend', 'date', 'tomorrow', 'next_week']]
        kf = KFold(n_splits=5, random_state=SEED, shuffle=True)

        for trn_idx, val_idx in kf.split(train_df):
            tt = lgb.Dataset(train_df.loc[trn_idx, feature], train_df.loc[trn_idx, ['target']])
            vv = lgb.Dataset(train_df.loc[val_idx, feature], train_df.loc[val_idx, ['target']])
            model = lgb.train(lgb_params, tt, 100, valid_sets=[tt, vv], early_stopping_rounds=50, verbose_eval=0)
            pred1 += model.predict(test_df[feature])/5
        pred1[pred1<0] = train_df['target'].min()
        glo_pred1 += pred1/5

    using_temp=False
    train_df, test_df = merge(test, sub['meter_id'][idx], hour=hours, use_temp=using_temp)
    glo_pred2 = np.zeros(len(test_df))

    for SEED in [42, 43, 44, 45, 46]:
        lgb_params['seed'] = SEED
        pred2 = np.zeros(len(test_df))

        feature = [i for i in train_df.columns if i not in ['Time', 'target', 'holiday', 'weekend', 'date', 'tomorrow', 'next_week']]
        kf = KFold(n_splits=5, random_state=SEED, shuffle=True)

        for trn_idx, val_idx in kf.split(train_df):
            tt = lgb.Dataset(train_df.loc[trn_idx, feature], train_df.loc[trn_idx, ['target']])
            vv = lgb.Dataset(train_df.loc[val_idx, feature], train_df.loc[val_idx, ['target']])
            model = lgb.train(lgb_params, tt, 200, valid_sets=[tt, vv], early_stopping_rounds=50, verbose_eval=0)
            pred2 += model.predict(test_df[feature])/5
        pred2[pred2<0] = train_df['target'].min()
        glo_pred2 += pred2/5
    submit_dict[idx] = glo_pred1*0.6 + glo_pred2*0.4

submit_df = pd.DataFrame(submit_dict)
submit_df.columns = sub['meter_id']
submit_df = submit_df.T.reset_index()
submit_df.columns = sub.columns[:25]
submit_df_hour = submit_df.copy()
```

```
for idx in tqdm.tqdm_notebook(list(range(200))):
    hours=False
    train_df, test_df = merge(test, sub['meter_id'][idx], hour=hours)
    glo_pred = np.zeros(len(test_df))

    for SEED in [42, 43, 44, 45, 46]:
        lgb_params['seed'] = SEED
        pred = np.zeros(len(test_df))

        feature = [i for i in train_df.columns if i not in ['Time', 'target', 'holiday', 'weekend', 'date', 'tomorrow']]
        kf = KFold(n_splits=5, random_state=SEED, shuffle=True)

        for trn_idx, val_idx in kf.split(train_df):
            tt = lgb.Dataset(train_df.loc[trn_idx, feature], train_df.loc[trn_idx, ['target']])
            vv = lgb.Dataset(train_df.loc[val_idx, feature], train_df.loc[val_idx, ['target']])
            model = lgb.train(lgb_params, tt, 200, valid_sets=[tt, vv], early_stopping_rounds=50, verbose_eval=0)
            pred += model.predict(test_df[feature])/5
        pred[pred<0] = train_df['target'].min()
        glo_pred += pred/5
    submit_dict[idx] = glo_pred

submit_df = pd.concat([pd.DataFrame(submit_dict).loc[:23],
                       pd.concat([pd.DataFrame([j for i in range(10) for j in str(i) * 24], columns=['house']),
                                  pd.DataFrame(submit_dict).loc[:239]], 1).groupby('house').sum().reset_index(drop=True),
                       pd.concat([test_df['Time'].dt.to_period('m'),
                                  pd.DataFrame(submit_dict)], 1).groupby('Time').sum().reset_index(drop=True)])

submit_df.columns = sub['meter_id']
submit_df = submit_df.T.reset_index()
submit_df.columns = sub.columns
submit_df_day = submit_df.copy()
```

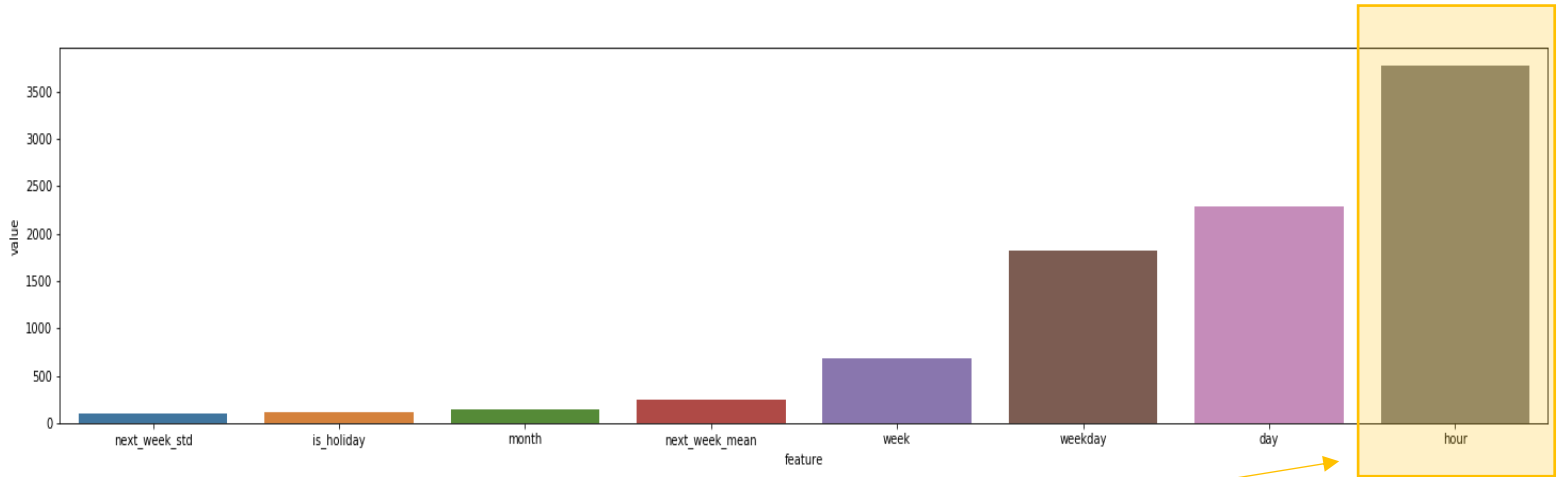
1. 라이브러리 및 데이터 Library & Data
2. 데이터 전처리 Data Cleansing & Pre-Processing
3. 탐색적 자료분석 Exploratory Data Analysis
4. 변수 선택 및 모델 구축 Feature Engineering & Initial Modeling
5. 모델 학습 및 검증 Model Tuning & Evaluation
6. 결과 및 결언 Conclusion & Discussion

jupyter ## 6.Conclusion & Discussion

Logout

File Edit View Insert Cell Kernel Help Python 3

Code CellToolbar



* 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.