In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math

from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split, GridSearchCV

from xgboost import XGBRegressor
from sklearn.ensemble import RandomForestRegressor
```

In [2]:

```
#creating features
def io_times_daygap_diff1(df,method,bycolumn,calcolumn):
    import math
    if method == 'max':
        test_t = df.groupby(bycolumn)[calcolumn].max().reset_index(
                name = 'history_date'+calcolumn+'max')
    elif method == 'min':
        test_t = df.groupby(bycolumn)[calcolumn].min().reset_index(
                name = 'history_date'+calcolumn+'min')
    elif method == 'range':
        test_t = pd.DataFrame(df.groupby(bycolumn)[calcolumn].max()
                 - df.groupby(bycolumn)[calcolumn].min())
        test_t.rename(columns = {calcolumn: 'history_date'+calcolumn +'range
        test_t[bycolumn] = test_t.index
        test_t.reset_index(drop=True,inplace = True)
    elif method == 'std':
        test_t = df.groupby(bycolumn)[calcolumn].std().reset_index(
                name = 'history_date'+calcolumn+'std')
    elif method == 'avg':
        test_t = df.groupby(bycolumn)[calcolumn].mean().reset_index(
                name =  'history_date'+calcolumn+'avg')
    elif method == 'skew':
        test_t = df.groupby(bycolumn)[calcolumn].skew().reset_index(
                name =  'history_date'+calcolumn+'skew')
    elif method == 'kurt':
        test_t = df.groupby(bycolumn)[calcolumn].apply(pd.DataFrame.kurt).re
                name = 'history_date'+calcolumn+'kurt')
    return test_t
```

In [3]:

```
#define features and load datasets
features = ['season', 'month', 'hour', 'hour_sin', 'hour_cos', 'hourlyAverag
            'hourlyHumidity',
            'hourlyUV_Index',
            'NT',
            'ST',
            'hourlyCoolingLoad',
            'T-1',
            'T-2',
            'T-3',
            'T-4',
            'T-5',
            'Max',
            'Min',
            'Range',
            'Std',
            'Kurt',
            'Skew',
            'Median']

train = pd.read_csv("datasets/hourly_training.csv")

verifying = pd.read_csv("datasets/hourly_verifying.csv")

test = pd.read_csv("datasets/hourly_testing.csv")
test
```

Out[3]:

	Timestamp	hour	hour_sin	hour_cos	hourlyAverage_OAT	hourlyHumidity	hou
0	2021-09-24 0:00	0	0.000000	1.000000	28.954708	80.921200	
1	2021-09-24 1:00	1	0.258819	0.965926	28.262833	85.972092	
2	2021-09-24 2:00	2	0.500000	0.866025	28.747625	82.472033	
3	2021-09-24 3:00	3	0.707107	0.707107	28.895333	81.559533	
4	2021-09-24 4:00	4	0.866025	0.500000	28.618458	82.609533	
...	...	...	...	...	...	...	
163	2021-09-30 19:00	19	-0.965926	0.258819	29.724292	85.423775	
164	2021-09-30 20:00	20	-0.866025	0.500000	29.926167	83.867875	
165	2021-09-30 21:00	21	-0.707107	0.707107	29.770333	86.922117	
166	2021-09-30 22:00	22	-0.500000	0.866025	29.572625	89.122967	
167	2021-09-30 23:00	23	-0.258819	0.965926	28.091083	86.288329	

168 rows × 22 columns

In [4]:

```
#define features for training, verifing, and testing
train_input = train[[#'season', 'month',
                     'hour', 'hour_sin', 'hour_cos',
    'hourlyAverage_OAT', 'hourlyHumidity', 'hourlyUV_Index',
            'T-1',
            'T-2',
            'T-3',
            'T-4',
            'T-5',
            'Max',
            'Min',
            'Range',
            'Std',
            'Kurt',
            'Skew',
            'Median']]


train_output = train.hourlyCoolingLoad

verifying_input = verifying[[#'season', 'month',
                             'hour', 'hour_sin', 'hour_cos',
    'hourlyAverage_OAT', 'hourlyHumidity', 'hourlyUV_Index',
            'T-1',
            'T-2',
            'T-3',
            'T-4',
            'T-5',
            'Max',
            'Min',
            'Range',
            'Std',
            'Kurt',
            'Skew',
            'Median']]

verifying_output = verifying.hourlyCoolingLoad

test_input = test[[#'season', 'month',
                   'hour', 'hour_sin', 'hour_cos',
    'hourlyAverage_OAT', 'hourlyHumidity', 'hourlyUV_Index',
            'T-1',
            'T-2',
            'T-3',
            'T-4',
            'T-5',
            'Max',
            'Min',
            'Range',
            'Std',
            'Kurt',
            'Skew',
            'Median']]
test_output = test.hourlyCoolingLoad
```

In [9]:

```
params={
    'booster':'gbtree',
        'objective': 'reg:linear',
        'eval_metric': 'rmse',
    'n_estimators':800,
    'max_depth':11,
    'min_child_weight':7,
        'gamma':1.2,
        'subsample':0.5,
        'colsample_bytree':0.88,
    'alpha': 0.1,
    'lambda':2,
        'eta': 0.057,
    'scale_pos_weight':1,
        'seed':0,
    'silent':0
}

model = XGBRegressor(**params)
model.fit(train_input, train_output)

"""
importances  = pd.DataFrame(model.feature_importances_)
importances.to_csv('imporatnces.csv')

print(importances)

"""

predictions = model.predict(test_input)
rmse = math.sqrt(mean_squared_error(test_output, predictions))

importances  = pd.DataFrame(model.feature_importances_)
print(importances)

pd.DataFrame(predictions).to_csv('predictions.csv')

print(rmse)
print(model.score(test_input, test_output)*100)
```

```
[15:27:10] WARNING: /Users/runner/work/xgboost/xgboost/python-package/build/t
emp.macosx-11.0-arm64-cpython-38/xgboost/src/objective/regression_obj.cu:213:
reg:linear is now deprecated in favor of reg:squarederror.
[15:27:10] WARNING: /Users/runner/work/xgboost/xgboost/python-package/build/t
emp.macosx-11.0-arm64-cpython-38/xgboost/src/learner.cc:767:
Parameters: { "silent" } are not used.

           0
0   0.102257
1   0.016148
2   0.185613
3   0.001468
4   0.000472
5   0.000814
6   0.397745
7   0.115369
8   0.000842
9   0.000665
10  0.000625
11  0.096054
12  0.006748
13  0.022019
14  0.017441
15  0.005092
16  0.028342
17  0.002286
102.66273009571589
99.82505614740832
```