1. 比赛介绍

【赛题背景】

- 能源是制约公司、行业乃至整个社会可持续、健康发展的重要因素。解决能源问题是提高能源利用效率的关键方法之一。人工智能技术（AI）作为挖掘数字价值的重要技术，正为提高能源利用效率注入全新动能，例如对于能耗预测的走势判断设备的异常，加大能源结构管理和减少异常消耗能源。本次大赛旨在利用AI技术，通过对能耗进行预测，以提高能源利用效率。然而，在行业内的数据存在各种形式的问题，这给预测带来了巨大的挑战。如何将前沿技术与行业热点话题深度结合，加速以数强实、数实融合是本次大赛同时要解决的问题。携AI技术积淀与行业积累，施耐德电气带来“AI大施杯”算法大赛。面向个人开发者、高等院校、科研机构、企业单位等招募选手，共同壮大数智赋能的创新生态圈，推动业界效率的提升与可持续目标的实现。
- 中国建筑节能协会统计发布的报告显示，国内建筑碳排放量占全国排放总量的50.6%，而随着城镇化快速推进和产业结构深度调整，其碳排放总量与比例均可能进一步提高，更将成为实现节能减排与双碳目标落地的重要阵地之一。紧贴这一关键场景，首届“AI大施杯”算法大赛将以智能建筑领域能耗预测为主题，要求参赛选手使用AI算法构建预测模型，基于已有数据实现未来特定时限内的综合能耗预测。

大赛将于3月2日正式启动，分为初赛和决赛两大赛段，通过初赛两轮评测通道的参赛队伍将再参与一轮决赛评测，选出总排名前十的队伍入围5月底的决赛答辩，从而决出一至三等奖及优胜奖，获胜者除获得现金奖励之外，更有业务合作与职业发展机会。

2. 赛题说明

【赛题介绍】

- 能耗预测是时序预测领域的一个经典应用方向，准确的预测结果能够更好的帮助商业楼宇响应 “双碳”目标，实现节能和环保。本赛题要求选手使用人工智能算法构建精准的预测模型，并实现未来7天小时级别的综合能耗预测。

【赛题任务】

- 根据主办方提供的能耗和天气历史数据，训练AI模型对4种不同类型的能耗（分别为总能耗，空调能耗，照明能耗及插座能耗）实现未来7* 24小时的能耗预测。
- 注：总能耗=空调能耗+照明能耗+插座能耗

【比赛机制】

- 本次比赛将会分为初赛和决赛两个阶段，初赛包含两个测评通道，分为测评通道A和B，决赛包含一个测评通道。三个测评通道将会按时间顺序依次测评三个自然周的真实结果和选手的预测结果。每个测评通道开启后，将会更新最新的历史能耗数据供选手使用。其中初赛的测评通道选手可每天提交五次，决赛测评通道选手可每天提交三次。

【测评规则】

- 本次比赛，采用R^2指标衡量算法预测的综合效果，具体计算方法请参考赛题官网。

3. 目标任务

【任务说明】

- 根据主办方提供的能耗和天气历史数据，训练AI模型对3种不同类型的耗能（分别为总耗能，空调耗能，照明及插座耗能）实现未来7*24小时的能耗预测。

【提交格式示例】

- 将本笔记输出的.csv文件压缩为test.zip，将其提交到AI Studio，平台进行在线评分，实时排名。

【评价方法】

- 最终得分：每个测评通道取前100名为有效分数队伍，第一名100分，第二名99分，第三名98分，以此类推。决赛权重60%，测评通道B权重30%，测评通道A权重10%。

4. 大赛baseline代码

- 本大赛提供测评通道A的参考代码供选手学习。
- 本baseline代码旨在更为清晰地说明数据读取、模型训练以及评测流程，让选手能更专注于预测效果的提升。
- 本baseline只提供基本参考方法，并不对预测效果负责。

导入所需要包：

In [1]:
```
import pandas as pd
import lightgbm as lgb
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt
from sklearn.multioutput import MultiOutputRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from lightgbm import LGBMRegressor
import numpy as np
import datetime
from chinese_calendar import is_workday
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

读入能耗数据：

In [2]:
```
sub_sheet_name = ["二层插座", "二层照明", "一层插座", "一层照明", "空调"]
data_dict = {}
for name in sub_sheet_name:
    data_dict[name] = pd.read_csv('Data_A/' + name + '_EC.csv')
    data_dict[name]["time"] = pd.to_datetime(data_dict[name]["time"])
data_dict
```

Out[2]:

读入天气数据

In [3]:
```
wh = pd.read_csv("Data_A/天气.csv")
wh["日期"] = pd.to_datetime(wh["日期"])
```

```
wh["time"] = wh["日期"] + pd.to_timedelta(wh["小时"], "h")
wh.drop(["Unnamed: 0", "日期", "小时"], axis=1, inplace=True)
wh
```

处理缺失数据

- 由于大赛提供的数据表存在不同程度的数据缺失问题，将数据对其、合并，并将缺失数据删除。
- 注意：能耗表与天气表时间均是北京时间。

In [4]:
```
data = pd.merge(data_dict["二层插座"],data_dict["二层照明"],on='time', how="inner", suffixes=("_socket_2","_light_2"))
data = pd.merge(data,data_dict["一层插座"],on='time', how="inner", suffixes=("","_socket_1"))
data = pd.merge(data,data_dict["一层照明"],on='time', how="inner", suffixes=("","_light_1"))
data = pd.merge(data,data_dict["空调"],on='time', how="inner", suffixes=("","_air"))
data = data.rename(columns={"value":"value_socket_1"})
data
```

将一层、二层插座能耗相加，一层、二层照明能耗相加构造数据：

In [5]:
```
data["value_socket"] = data["value_socket_2"] + data["value_socket_1"]
data["value_light"] = data["value_light_2"] + data["value_light_1"]
data.drop(['value_socket_2','value_light_2','value_socket_1','value_light_1'], axis=1, inplace=True)
data
```

将数据尺度调整为每小时，并根据赛题要求选定数据范围，构造特征：

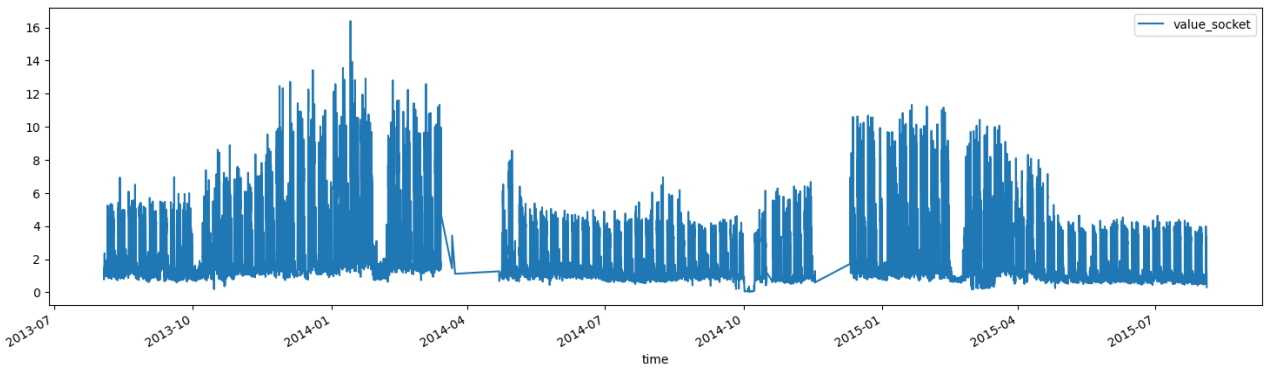
In [6]:
```
data = data[(data['time'] >= '2013-8-03 00:00:00') & (data['time'] <= '2015-08-03 23:00:00')]
if_not_workday = []
for da in data['time']:
    if_not_workday.append(is_workday(da))
data['workday'] = if_not_workday
data['hour'] = data['time'].dt.hour
data['week'] = data['time'].dt.weekday
data['day'] = data['time'].dt.day
data['month'] = data['time'].dt.month
data['year'] = data['time'].dt.year
data = data.groupby(["year",'month','day','week','hour','workday'])[['value_socket','value_light','value_air']].sum().reset_index()
data
```

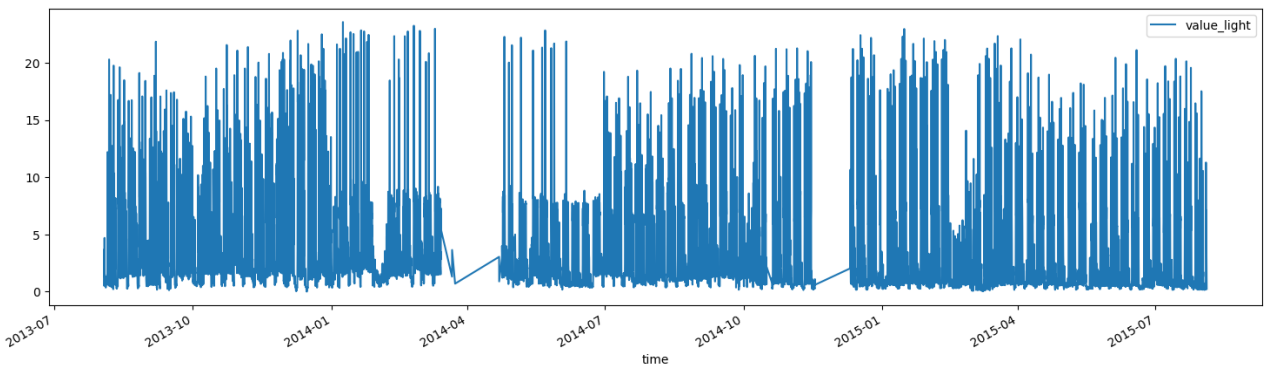
同样处理天气数据，并进行merge：

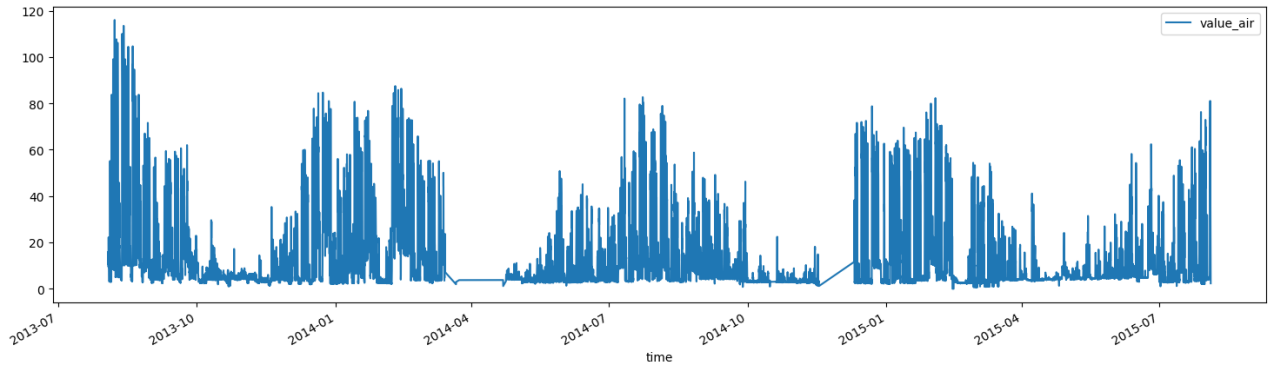
In [7]:
```
wh['hour'] = wh['time'].dt.hour
wh['week'] = wh['time'].dt.weekday
wh['day'] = wh['time'].dt.day
wh['month'] = wh['time'].dt.month
wh['year'] = wh['time'].dt.year
data = pd.merge(data,wh,on=['hour','week','day','month','year'], how="inner")
data = data.rename(columns={"温度(C°)":"temp","湿度(%)":"humidity","降雨量(mm)":"rainfall",
                            "大气压(hPa)":"atmos","风向":"wind_direction","风向角度(°)":"wind_angle",
                            "风速(KM/H)":"wind_speed","云量(%)":"cloud"})
le = LabelEncoder()
data['wind_direction'] = le.fit_transform(data['wind_direction'])
data['wind_direction'] = data['wind_direction'].astype('category')
data['workday'] = le.fit_transform(data['workday'])
data['workday'] = data['workday'].astype('category')

data
```

绘制曲线：

In [8]:
```
data.plot("time","value_socket",figsize=(18,5))
data.plot("time","value_light",figsize=(18,5))
data.plot("time","value_air",figsize=(18,5))
data.drop('time', axis=1, inplace=True)
```







多步直接预测

- 这里采用多步直接预测来构建数据；

In [9]:
```
data_socket = data.copy()
for i in range(7*24):
    data_socket['value_socket_{}'.format(i)] = data_socket['value_socket'].shift(-i-1)
data_socket.dropna(inplace=True)
data_socket
```

划分训练集、验证数据集：

In [10]:
```
targets = [item for item in data_socket.columns if 'value_socket_' in item]

X_train_socket = data_socket.drop(targets, axis=1)[: int(len(data_socket) * 0.95)]
y_train_socket = data_socket[targets][: int(len(data_socket) * 0.95)]

X_test_socket = data_socket.drop(targets, axis=1)[int(len(data_socket) * 0.95) :]
y_test_socket = data_socket[targets][int(len(data_socket) * 0.95) :]

X_train_socket
```

In [11]:
```
y_train_socket
```

自回归模型构建

- 这里我们使用Auto-regressive distributed lags来构建模型，并使用LightGBM进行预测；

In [12]:
```
model_socket = MultiOutputRegressor(lgb.LGBMRegressor(objective='regression')).fit(X_train_socket, y_train_socket)
```

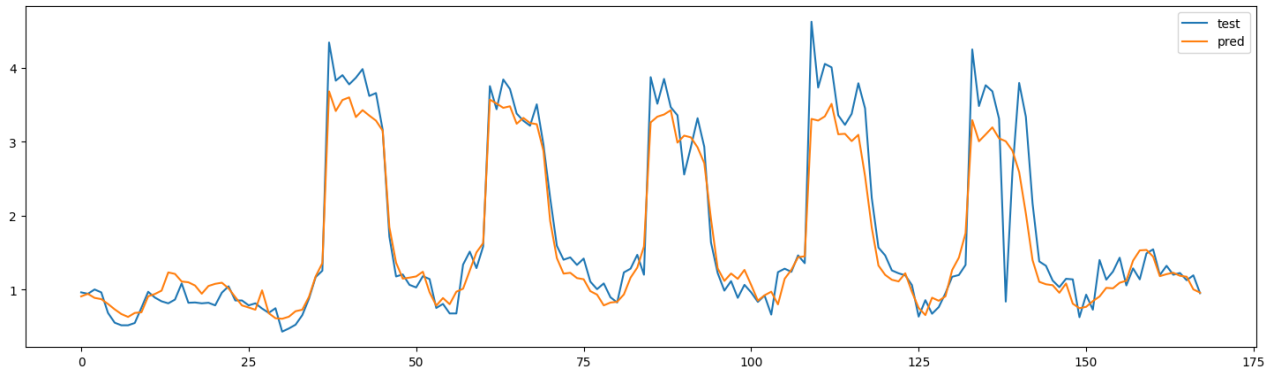
使用划分出的测试集测试：

In [13]:
```
pred_socket = pd.DataFrame(model_socket.predict(X_test_socket), columns=targets)
pred_socket.index = y_test_socket.index
pred_socket
```

In [14]:
```
y_test_socket
```

In [15]:
```
pred_num_socket = pred_socket.loc[15000,:].tolist()
y_test_num_socket = y_test_socket.loc[15000,:].tolist()
plt.figure(figsize=(18,5))
plt.plot(y_test_num_socket, label="test")
plt.plot(pred_num_socket, label="pred")
plt.legend()
```

Out[15]:
```
<matplotlib.legend.Legend at 0x7f34ee736b90>
```



分别计算七天的R2，并按照大赛评分标准计算R2_T_socket：

In [16]:
```
R2_list_socket = []
weight_socket = [0.25, 0.15, 0.15, 0.15, 0.1, 0.1, 0.1]
for day in range(7):
    day_list = []
    for i in range(day*24,(day+1)*24):
        day_list.append('value_socket_{}'.format(i))
    pred_day_socket = pred_socket[day_list]
    test_day_socket = y_test_socket[day_list]
    R2_list_socket.append(r2_score(test_day_socket, pred_day_socket))
R2_T_socket = np.multiply(np.array(weight_socket), np.array(R2_list_socket)).sum()
R2_T_socket
```

Out[16]:
```
0.8893487303167646
```

同样的方法预测照明能耗，并计算R2_T_light：

In [17]:
```
data_light = data.copy()
for i in range(7*24):
    data_light['value_light_{}'.format(i)] = data_light['value_light'].shift(-i-1)
data_light.dropna(inplace=True)

targets = [item for item in data_light.columns if 'value_light_' in item]

X_train_light = data_light.drop(targets, axis=1)[: int(len(data_light) * 0.95)]
y_train_light = data_light[targets][: int(len(data_light) * 0.95)]

X_test_light = data_light.drop(targets, axis=1)[int(len(data_light) * 0.95) :]
y_test_light = data_light[targets][int(len(data_light) * 0.95) :]

model_light = MultiOutputRegressor(lgb.LGBMRegressor(objective='regression')).fit(X_train_light, y_train_light)

pred_light = pd.DataFrame(model_light.predict(X_test_light), columns=targets)
pred_light.index = y_test_light.index

R2_list_light = []
weight_light = [0.25, 0.15, 0.15, 0.15, 0.1, 0.1, 0.1]
for day in range(7):
    day_list = []
    for i in range(day*24,(day+1)*24):
        day_list.append('value_light_{}'.format(i))
    pred_day_light = pred_light[day_list]
    test_day_light = y_test_light[day_list]
    R2_list_light.append(r2_score(test_day_light, pred_day_light))
R2_T_light = np.multiply(np.array(weight_light), np.array(R2_list_light)).sum()
R2_T_light
```

Out[17]: 0.8156069059987674

同样的方法预测空调能耗，并计算R2_T_air：

In [18]:
```
data_air = data.copy()
for i in range(7*24):
    data_air['value_air_{}'.format(i)] = data_air['value_air'].shift(-i-1)
data_air.dropna(inplace=True)

targets = [item for item in data_air.columns if 'value_air_' in item]

X_train_air = data_air.drop(targets, axis=1)[: int(len(data_air) * 0.95)]
y_train_air = data_air[targets][: int(len(data_air) * 0.95)]

X_test_air = data_air.drop(targets, axis=1)[int(len(data_air) * 0.95) :]
y_test_air = data_air[targets][int(len(data_air) * 0.95) :]

model_air = MultiOutputRegressor(lgb.LGBMRegressor(objective='regression')).fit(X_train_air, y_train_air)

pred_air = pd.DataFrame(model_air.predict(X_test_air), columns=targets)
pred_air.index = y_test_air.index

R2_list_air = []
weight_air = [0.25, 0.15, 0.15, 0.15, 0.1, 0.1, 0.1]
for day in range(7):
    day_list = []
    for i in range(day*24,(day+1)*24):
        day_list.append('value_air_{}'.format(i))
    pred_day_air = pred_air[day_list]
    test_day_air = y_test_air[day_list]
    R2_list_air.append(r2_score(test_day_air, pred_day_air))
R2_T_air = np.multiply(np.array(weight_air), np.array(R2_list_air)).sum()
R2_T_air
```

Out[18]: 0.5846897834934842

计算总能耗，并计算R2_T_total：

In [19]:
```
for i in range(7*24):
    y_test_air = y_test_air.rename(columns={"value_air_{}".format(i):"value_total_{}".format(i)})
    y_test_light = y_test_light.rename(columns={"value_light_{}".format(i):"value_total_{}".format(i)})
    y_test_socket = y_test_socket.rename(columns={"value_socket_{}".format(i):"value_total_{}".format(i)})
    pred_air = pred_air.rename(columns={"value_air_{}".format(i):"value_total_{}".format(i)})
    pred_light = pred_light.rename(columns={"value_light_{}".format(i):"value_total_{}".format(i)})
    pred_socket = pred_socket.rename(columns={"value_socket_{}".format(i):"value_total_{}".format(i)})
y_test_total = y_test_air + y_test_light + y_test_socket
pred_total = pred_air + pred_light + pred_socket

y_test_total = y_test_air + y_test_light + y_test_socket
pred_total = pred_air + pred_light + pred_socket
R2_list_total = []
weight_total = [0.25, 0.15, 0.15, 0.15, 0.1, 0.1, 0.1]
for day in range(7):
    day_list = []
    for i in range(day*24,(day+1)*24):
        day_list.append('value_total_{}'.format(i))
    pred_day_total = pred_total[day_list]
    test_day_total = y_test_total[day_list]
    R2_list_total.append(r2_score(test_day_total, pred_day_total))
R2_T_total = np.multiply(np.array(weight_total), np.array(R2_list_total)).sum()
R2_T_total
```

Out[19]: 0.7028840658029298

预测未来七天的各项能耗，并输出用于评测的.csv文件：

In [20]:
```
final_test_input = data.iloc[-1:]
final_pred_socket = model_socket.predict(final_test_input)
final_pred_light = model_light.predict(final_test_input)
final_pred_air = model_air.predict(final_test_input)
results = pd.DataFrame()
results['total'] = final_pred_air[0] + final_pred_light[0] + final_pred_socket[0]
results['socket'] = final_pred_socket[0]
results['light'] = final_pred_light[0]
results['air'] = final_pred_air[0]
results.to_csv('pre.csv', index = False)
results
```

You can do

- 特征工程
- 更丰富的时序处理方法
- 更多的模型
- 调参
- ...