

# ASHRAEコンペデータを用いた電力予測モデル構築

2023.8.1 長田和哉

# 目次

- ・ 背景、目的
- ・ 解析フロー
- ・ 問題の理解
- ・ データ前処理
  - 目的変数の基本情報確認
  - 説明変数の基本情報確認
  - 相関確認(相互、目的変数)
  - 欠損値除去、外れ値除去
- ・ モデル選定
  - LightGBM
- ・ 学習及び結果確認
  - ハイパーパラメータ調整：GridSearch
  - validation戦略：交差検証
  - 評価指標：RMSE,  $R^2$
- ・ まとめ

# 背景、目的

## 背景

ASHRAEコンペ： kaggleで2019年に開催された、 1448棟のビルの4種類のメーター（電力、冷水、スチーム、温水）の値を天候や位置情報から予測する回帰問題。

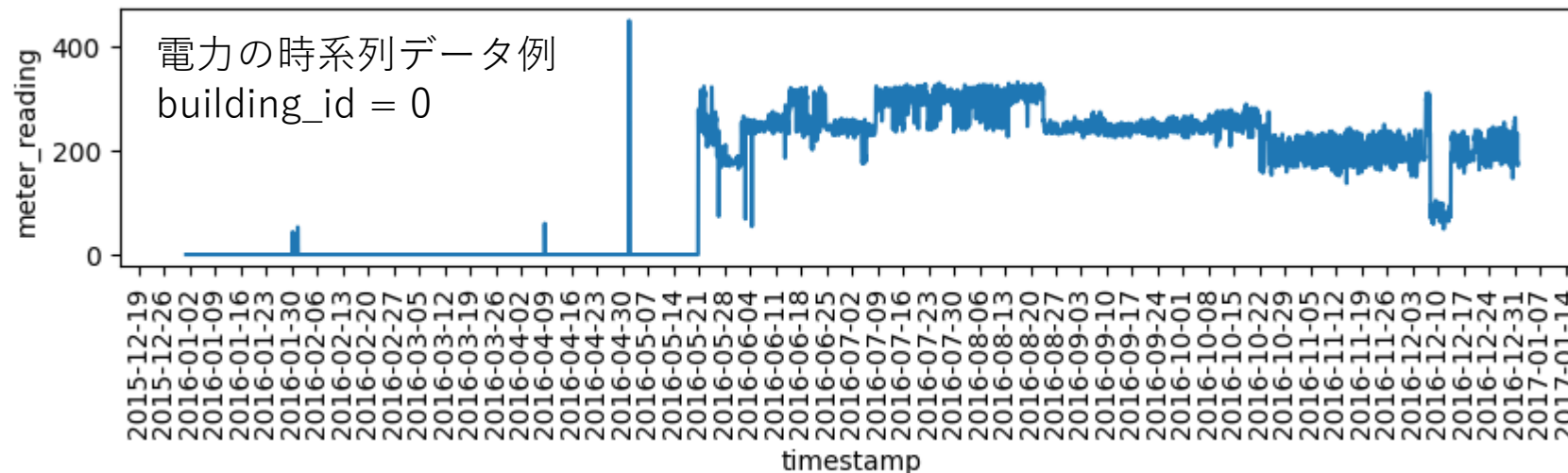
<https://www.kaggle.com/competitions/ashrae-energy-prediction>

→データに欠損値やノイズが多く、前処理から丁寧に行う必要がある。

## 目的

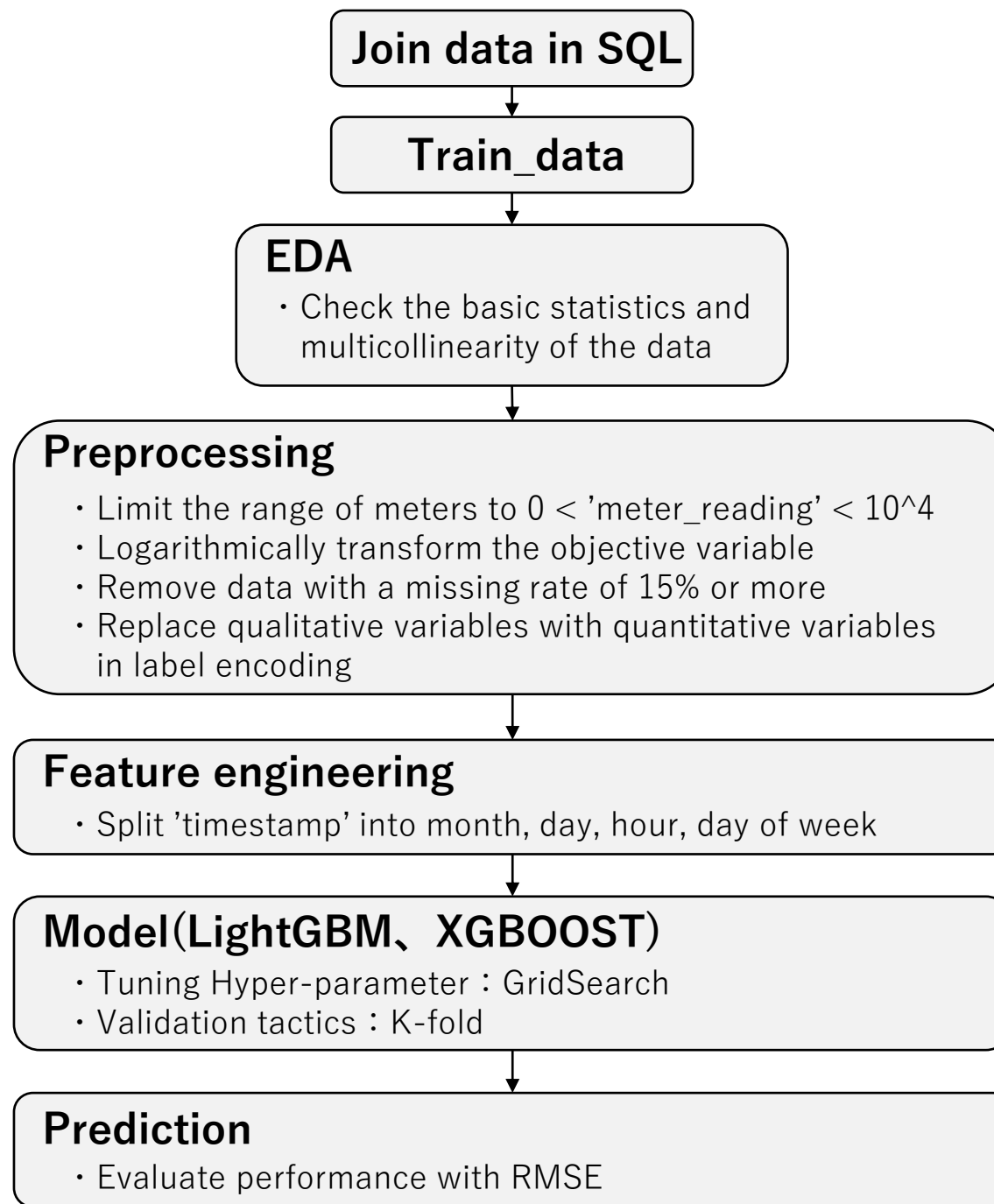
ASHRAEコンペデータを用いて電力予測モデル構築の練習を行う。（下記ルール）

- ・ 予測するのは電力のみ
- ・ trainデータの一部をtestデータとして切り出し、testデータのRMSEが最小かつ $R^2$ 最大のモデルを目指す



NEXT:今回の解析フロー

# 解析フロー



NEXT:取り組む問題の理解

# 問題の理解

## 概要

目的変数 : meter\_reading(メーターの電力読み値)  
説明変数 : air\_temperature(気温), cloud\_coverage(雲量)…等  
データ期間 : 016-01-01~2016-12-31  
ビルの情報 : 全1,449件、全16のサイト(都市)に分かれている  
予測間隔 : 1時間ごと  
追加情報 : ビルの補足情報とエリアの気象情報が利用可能

## 考察

- ・ビルの電力用途仮説
  - 照明、空調、PC等…日中、平日、暑い日、寒い日等に電力使用量が増大する
- ・目的変数との相関仮説
  - 気温・露点温度(～湿度) : 空調の電力量、雲量・降水量 : 照明の電力量に相関がある
- ・解析環境の選定
  - python : 今回は重回帰、アンサンブル回帰などの多変量回帰モデルが想定される。使い慣れたpythonを選定
  - GCP BigQuery : データ量が多く(2GB程度)ファイルが分割されているため、SQLでの結合処理が望ましい
  - jupyter notebook on Vertex AI Workbench : BigQueryと連携できるpython環境、ライブラリも充実している

## データ

- ・ train.csv
  - building\_id - ビルID
  - meter - メーター項目の種類 (電気、水道、蒸気、温水)
  - timestamp - 日付
  - meter\_reading - メーターの値
- ・ weather\_[train/test].csv
  - site\_id - エリアID
  - air\_temperature - 気温
  - cloud\_coverage - 雲量
  - dew\_temperature - 露点温度
  - precip\_depth\_1\_hr - 降水量
  - sea\_level\_pressure - 大気圧
  - wind\_direction - 風向
  - wind\_speed - 風速
- ・ building\_meta.csv
  - site\_id - エリアID
  - building\_id - ビルID
  - primary\_use - ビルの使用目的
  - square\_feet - 床面積
  - year\_built - 築年数
  - floor\_count - 階数

NEXT:BigQueryによるデータ整理

# データ整理

## データの流れ

### ①KaggleからCloudStrageにデータアップロード

オブジェクト					
設定   権限   保護   ライフサイクル   オブザーバビリティ					
バケット > ashrae_1 > ashrae-energy-prediction					
ファイルをアップロード   フォルダをアップロード   フォルダを作成   データ転送   保留を					
名前の接頭辞のみでフィルタ   フィルタ   オブジェクトとフォルダをフィルタ					
<input type="checkbox"/>	名前	サイズ	種類	作成日時	ス
<input type="checkbox"/>	<a href="#">building_metadata.csv</a>	44.5 KB	text/csv	2023/07/30 14:45:48	Si
<input type="checkbox"/>	<a href="#">sample_submission.csv</a>	426.8 MB	text/csv	2023/07/30 14:52:42	Si
<input type="checkbox"/>	<a href="#">test.csv</a>	1.4 GB	text/csv	2023/07/30 14:57:49	Si
<input type="checkbox"/>	<a href="#">train.csv</a>	647.2 MB	text/csv	2023/07/30 14:54:25	Si
<input type="checkbox"/>	<a href="#">weather_test.csv</a>	14.1 MB	text/csv	2023/07/30 14:52:54	Si
<input type="checkbox"/>	<a href="#">weather_train.csv</a>	7.1 MB	text/csv	2023/07/30 14:53:11	Si

### ②BigQueryでテーブルとして読み込み

Google Cloud SQL training

エクスプローラ

Workspaceのリソースを表示しています。  
スター付きのみを表示

sql-tra

energy\_prediction

building\_metadata

sample\_submission

test

test\_join

test\_join\_n0

train

train\_join

train\_join\_n0

weather\_test

weather\_train

さらに表示  
さらに表示

### ③SQLクエリでデータ結合

無題2

実行   保存   共有   スケジュール   展開

```
1 SELECT *
2 FROM `energy_prediction.train` AS t
3 JOIN `energy_prediction.weather_train` AS w ON t.timestamp = w.timestamp
4 JOIN `energy_prediction.building_metadata` AS b ON w.site_id = b.site_id AND t.building_id = b.building_id
5 ORDER BY b.building_id,t.timestamp
```

このクエリを

### ④クエリ結果を保存

行	row_id	building_id	meter	timestamp	site_id	timestamp_1
1	0	0	0	2017-01-01 00:00:00 UTC	0	2017-01-01 00:00:00 UTC
2	129	0	0	2017-01-01 01:00:00 UTC	0	2017-01-01 01:00:00 UTC
3	258	0	0	2017-01-01 02:00:00 UTC	0	2017-01-01 02:00:00 UTC
4	387	0	0	2017-01-01 03:00:00 UTC	0	2017-01-01 03:00:00 UTC
5	516	0	0	2017-01-01 04:00:00 UTC	0	2017-01-01 04:00:00 UTC
6	645	0	0	2017-01-01 05:00:00 UTC	0	2017-01-01 05:00:00 UTC
7	774	0	0	2017-01-01 06:00:00 UTC	0	2017-01-01 06:00:00 UTC
8	903	0	0	2017-01-01 07:00:00 UTC	0	2017-01-01 07:00:00 UTC
9	1032	0	0	2017-01-01 08:00:00 UTC	0	2017-01-01 08:00:00 UTC

### ⑤以降はVertex AI Workbench上のjupyter notebook で解析

NEXT:目的変数の基本情報を確認

# データ前処理：目的変数の基本情報確認

## チェックの目的

目的変数(meter\_reading)がモデル構築に値するデータであることを下記項目で事前チェック

### ①基本統計量

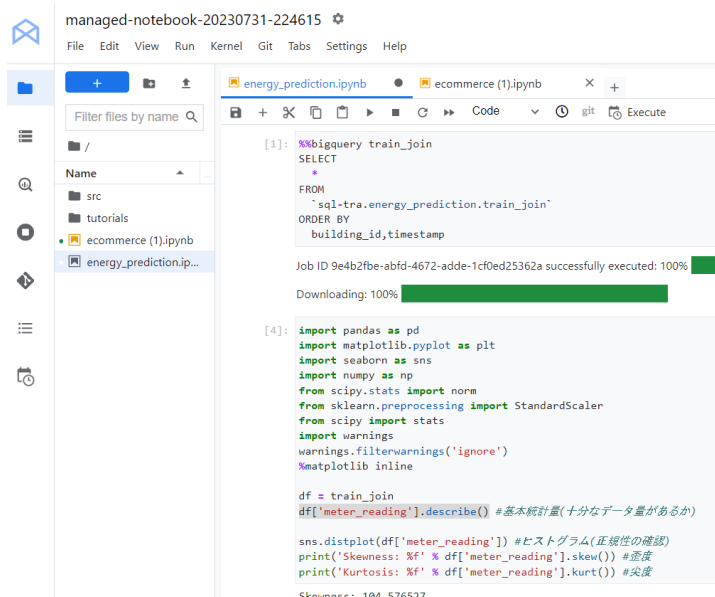
ポイント：モデル構築に十分なサンプル数があるか、平均・分散などの統計量が異常ではないか

### ②正規性

ポイント：歪度、尖度、外れ値の存在等を確認、外れ値の除去や線形回帰の条件検討として

## 結果

jupyter notebook on Vertex AI Workbenchで解析



```
[1]: %bigquery train_join
SELECT
  *
FROM
  `sql-tra.energy_prediction.train_join`
ORDER BY
  building_id,timestamp

Job ID 9e4b2fbe-abfd-4672-adde-1cf0ed25362a successfully executed: 100%
Downloading: 100%

[4]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline

df = train_join
df['meter_reading'].describe() #基本統計量(十分なデータ量があるか)

sns.distplot(df['meter_reading']) #ヒストグラム(正規性の確認)
print('Skewness: %f' % df['meter_reading'].skew()) #歪度
print('Kurtosis: %f' % df['meter_reading'].kurt()) #尖度

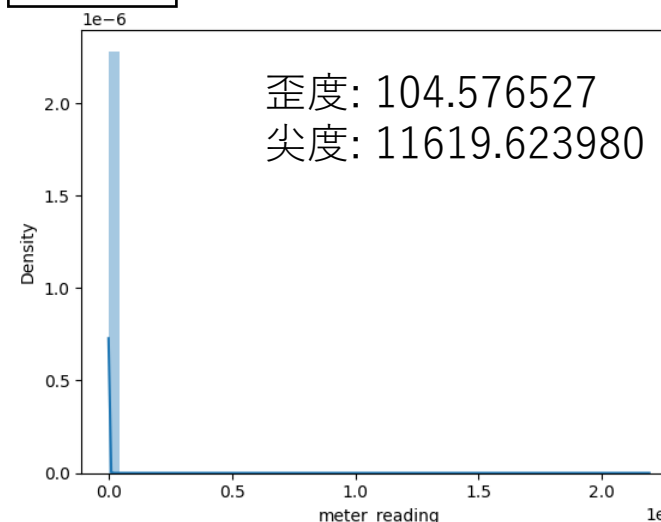
Skewness: 104.576527
```

※ソースは資料末尾のgithub参照

### 基本統計量

```
count    2.012560e+07
mean      2.124813e+03
std       1.535796e+05
min        0.000000e+00
25%       1.830000e+01
50%       7.875000e+01
75%       2.678430e+02
max       2.190470e+07
Name: meter_reading
dtype: float64
```

### 正規性



- ・ サンプル数7000万データ⇒如何なるモデルでも十分な水準
- ・ 最大値 $10^7$ と平均値 $10^3$ に4桁の差⇒電力使用量の差としては考えにくい
- ・ 尖度、歪度⇒正規分布から大きく乖離して尖っている、0付近にデータが多い⇒ **$10^7$ と0近傍の外れ値を除去して予測モデルを作成する必要がある**

NEXT:外れ値の除去 & 対数変換

# データ前処理：目的変数の基本情報確認

## チェックの目的

目的変数(meter\_reading)がモデル構築に値するデータであることを下記項目で事前チェック

### ①基本統計量

ポイント：モデル構築に十分なサンプル数があるか、平均・分散などの統計量が異常ではないか

### ②正規性

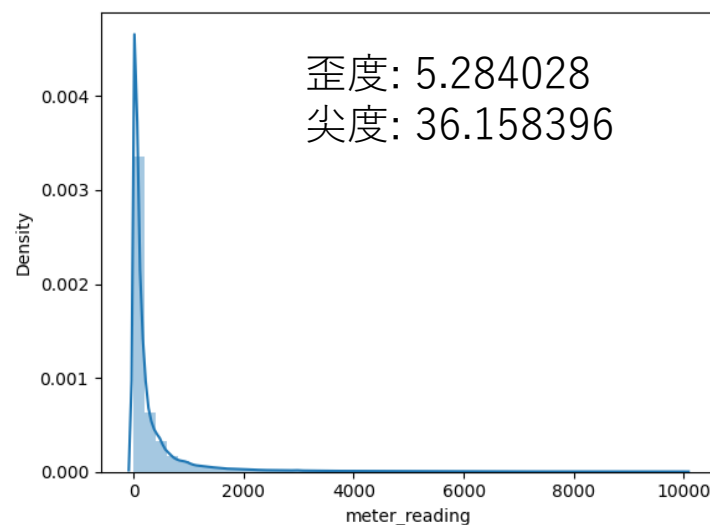
ポイント：歪度、尖度、外れ値の存在等を確認、外れ値の除去や線形回帰の条件検討として

**結果** meter\_readingの値を0以上10000以下に設定⇒再度基本統計量と正規性を確認

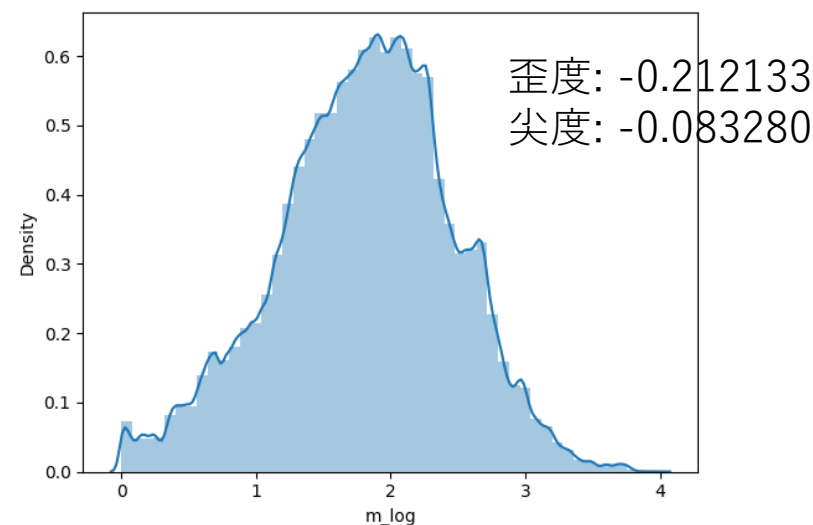
#### 基本統計量

```
count  1.148767e+07
mean    1.778589e+02
std     3.737603e+02
min     4.000000e-04
25%     2.335000e+01
50%     6.856670e+01
75%     1.780000e+02
max     9.966900e+03
Name: meter_reading
dtype: float64
```

#### 正規性



対数変換



正規分布に近づいた

・ 正規分布に近づいたが依然として0近傍に偏りあり⇒対数変換

NEXT:説明変数の基本情報を確認

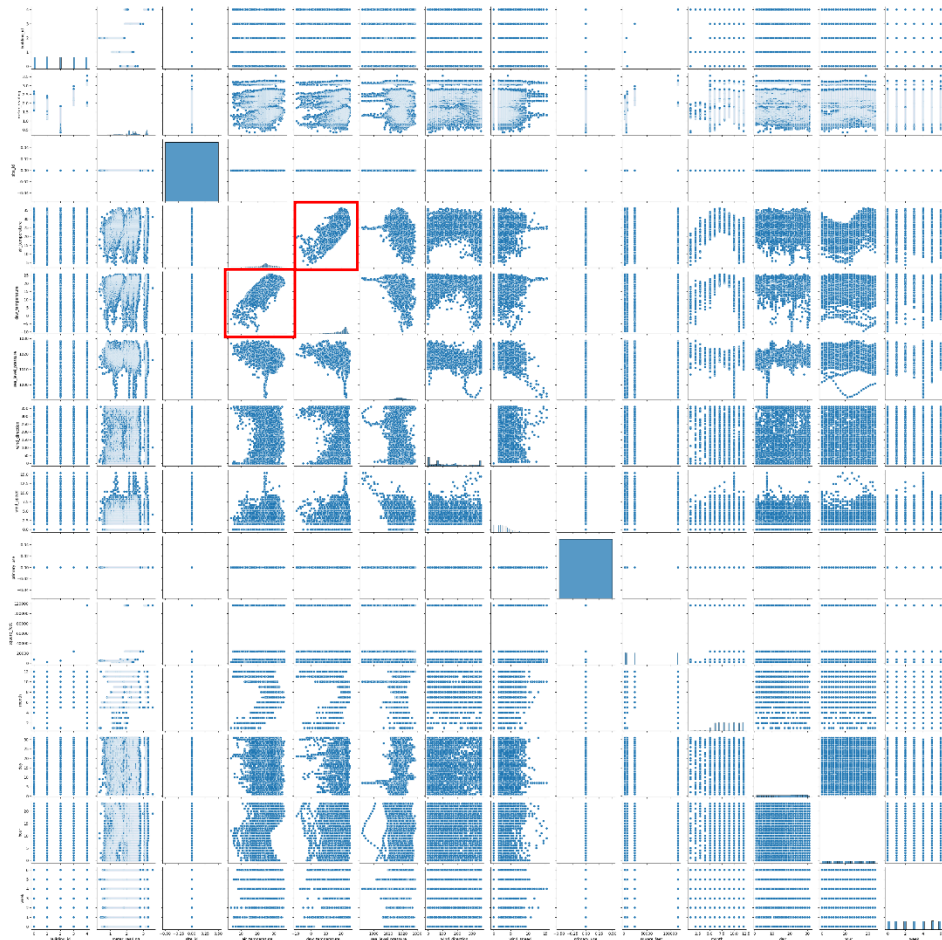


# データ前処理：説明変数の基本情報確認

## チェックの目的

説明変数間の多重共線性を除去するため、多変量連関図により変数同士の相関及びデータの分布を確認

※質的変数を数値にエンコード、今回はラベルエンコーディングを採用



- データ量が多いため `building_id < 10` をサンプリング
- `air_temperature`(気温)と`dew_temperature`(露点温度)に相関が見られる  
→ 同じ温度指標のため。今回はどちらもそのまま使用

NEXT:欠損値、外れ値の除去

# データ前処理：欠損値、外れ値の除去

## 欠損値、外れ値の除去

下記手順で実施

- ①目的変数(meter\_reading、メーターの電力読み値)の欠損値及び外れ値を除去(実施済)
  - 待機電力を考慮すると0及び負の値は除去すべき外れ値であると言える
  - また単変量解析により明らかに桁の大きいデータも削除する
- ②説明変数の内、一定の欠損値割合を持つデータを除去
  - 今回は15%以上の欠損データは削除
- ③残った欠損値は 0 で置換
- ④更に目的変数との二変量解析(可視化)により目視で外れ値を除去

削除 {

variable	Total(欠損値の数)	Percent(欠損値割合)
floor_count	8580691	0.746948
year_built	6322085	0.550337
cloud_coverage	5041037	0.438822
precip_depth_1_hr	2437674	0.212199
sea_level_pressure	958198	0.083411
wind_direction	609513	0.053058
wind_speed	22521	0.001960
dew_temperature	5077	0.000442
air_temperature	3368	0.000293
hour	0	0.000000
day	0	0.000000
month	0	0.000000
building_id	0	0.000000
square_feet	0	0.000000
primary_use	0	0.000000
meter_reading	0	0.000000
site_id	0	0.000000
week	0	0.000000

# モデル選定

## 多変量の回帰に対応したモデル選定

→下表の手法からメリデメを考慮して選択

→今回の説明変数には質的変数ある

→一般的にエンコーディングの手間が少なく済むアンサンブル学習系を採用：**LightGBM**

手法 \ メリデメ	メリット	デメリット
線形回帰系	<ul style="list-style-type: none"><li>・ロジックが単純で実装が楽</li><li>・説明変数の寄与率がわかる</li><li>・動作が高速</li></ul>	<ul style="list-style-type: none"><li>・非線形な挙動には対応できない</li><li>・残差の正規性が要求される</li><li>・基本的には量的変数のみ可</li></ul>
アンサンブル学習系	<ul style="list-style-type: none"><li>・非線形を表現できる</li><li>・質的/量的変数を同時処理可</li><li>・説明変数の寄与率がわかる</li><li>・大規模でも比較的高速なものがある</li></ul>	<ul style="list-style-type: none"><li>・データが少ないと過学習しやすい</li></ul>
NN系	<ul style="list-style-type: none"><li>・非線形含めた優れた表現力</li><li>・画像データの入力も可</li></ul>	<ul style="list-style-type: none"><li>・データが少ないと過学習しやすい</li><li>・説明変数の寄与度が不明</li><li>・学習に時間がかかる</li><li>・基本的には量的変数のみ可</li></ul>

NEXT:ハイパーパラメータの調整結果と学習結果の確認

# 学習及び結果確認

## 学習条件

ハイパーパラメータ調整：GridSearch  
Validation戦略：交差検証法 CV=5  
評価指標：RMSE,R^2

→

params = {'reg\_alpha': [0.0001, 0.003, 0.1],  
'reg\_lambda': [0.0001, 0.1],  
'num\_leaves': [2, 3, 4, 6],  
'colsample\_bytree': [0.4, 0.7, 1.0],  
'subsample': [0.4, 1.0],  
'subsample\_freq': [0, 7],  
'min\_child\_samples': [0, 2, 5, 10]}

## 結果

・最適パラメータ

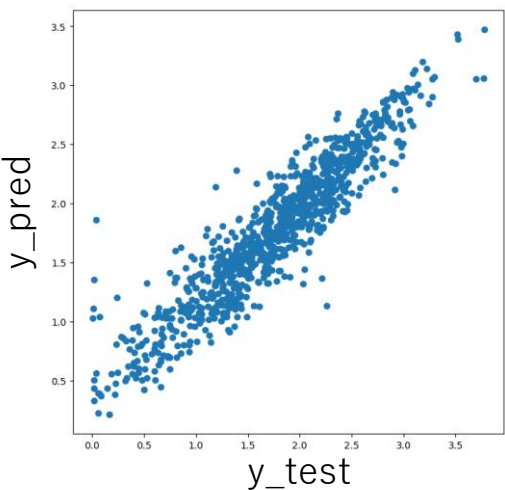
'colsample\_bytree': 0.4, 'min\_child\_samples': 0,  
'num\_leaves': 3, 'reg\_alpha': 0.0001, 'reg\_lambda': 0.1,  
'subsample': 0.4, 'subsample\_freq': 7

・評価指標

	学習	テスト
RMSE	0.26	0.26
R^2	0.85	0.85

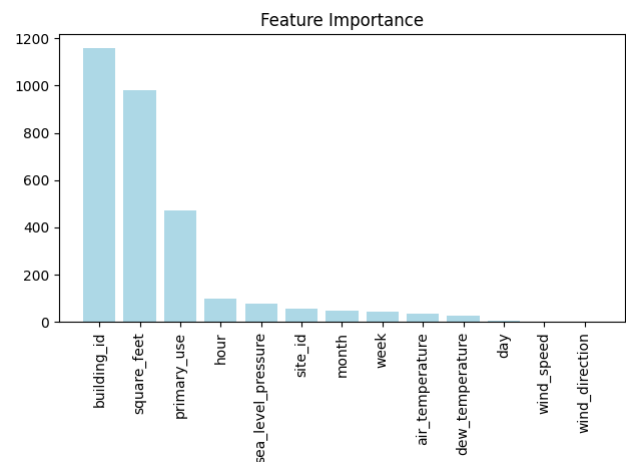
学習、テスト共に値に評価指標に差なし  
→過学習していない

・予測と実値の比較



R^2 = 0.85なので、  
本データセットに対しては  
予測精度の高いモデルを構築できた

・特徴量重要度



building\_id、square\_feetといった  
個々のビルの特徴や位置情報が  
電力使用量にとって重要度が高い

# まとめ

- ・ KaggleのASHRAEコンペデータを使って電力予測モデル構築の練習を行った
- ・ 目的変数、説明変数共に欠損値や異常値が多く、データの前処理を丁寧に実施する必要があった
- ・ LightGBMにより比較的高い精度で予測モデルを構築することができた
- ・ 特徴量重要度により、ビル毎の特性や位置情報が、電力消費量にとって重要であることがわかった
- ・ 更なるモデル精度向上に向けて下記改善点が挙げられる

## 前処理

欠損値や異常値を線形補完により穴埋め

ラベルエンコード→ターゲットエンコードに変更

## 特徴量エンジニアリング

気温、露点温度、風速などから不快指数などの特徴量を作成

## モデル改良

ハイパーパラメータの調整としてoptuna等のベイズ最適化の方法を用いる

NNや他のアンサンブル手法を複数組み合わせ

site\_id等のデータブロック毎に学習範囲やモデルを組み合わせる

# 参考資料

- ・ ASHRAE - Great Energy Predictor III

<https://www.kaggle.com/competitions/ashrae-energy-prediction/data>

- ・ 「データ前処理」 - Kaggle人気チュートリアル<https://qiita.com/hkthirano/items/12e046b3e02961d8460d>

- ・ Kaggleのエネルギー使用量予測コンペ（ASHRAEコンペ）に参加しました

<https://www.deep-percept.co.jp/blog/category02/202006051085/>

- ・ Kaggleコンペ、ASHRAE - Great Energy Predictor IIIの概要

<https://book.st-hakky.com/docs/ashrae-overview/>

- ・ 1st Place Solution Team Isamu & Matt

<https://www.kaggle.com/c/ashrae-energy-prediction/discussion/124709>

- ・ XGBoostとLightGBMの違い

<https://tech.datafluct.com/entry/20220223/1645542001>

- ・ LightGBMのearly\_stoppingの仕様が変わったので、使用法を調べてみた

[https://qiita.com/c60evaporator/items/2b7a2820d575e212bcf4#%E5%8F%82%E8%80%83-%E7%AC%AC3%E3%81%AEearly\\_stopping%E6%8C%87%E5%AE%9A%E6%96%B9%E6%B3%95](https://qiita.com/c60evaporator/items/2b7a2820d575e212bcf4#%E5%8F%82%E8%80%83-%E7%AC%AC3%E3%81%AEearly_stopping%E6%8C%87%E5%AE%9A%E6%96%B9%E6%B3%95)

- ・ Vertex AI\_workbenchとGitHubを連携させ、ファイルのクローンを作る方法

[https://blog.since2020.jp/ai/vertex-ai\\_workbench\\_github/](https://blog.since2020.jp/ai/vertex-ai_workbench_github/)

- ・ Cloud Console を使用してユーザー管理ノートブック インスタンスを作成する

<https://cloud.google.com/vertex-ai/docs/workbench/user-managed/create-user-managed-notebooks-instance-console-quickstart?hl=ja>