ASHRAE - Great Energy Predictor III

How much energy will a building consume?

Project Goals

- Minimize the number of false predictions
  - Get the most accurate prediction algorithm
  - Minimize deviation
- Define the most influential features

Situation

Our data - open data on energy consumption of buildings.

Our goal - create a Great Energy Predictor III to predict a rating of energy consumption of a building and to understand what signs have the strongest impact on it.

Problem statement

The task belongs to the class of machine learning tasks with a teacher, and is a regression construction:

Training with a teacher: we have all the necessary features, based on which the prediction is made, and the target feature itself.

Regression: we assume that the energy consumption rating is a continuous value.

Files

train.csv

- building_id
- meter (Read as {0: electricity,
  1: chilledwater,
  2: steam,
  3: hotwater})
- timestamp
- meter_reading

building_meta.csv

- site_id
- building_id
- primary_use
- square_feet
- year_built
- floor_count

weather_[train/test].csv

- site_id
- air_temperature
- cloud_coverage
- dew_temperature
- precip_depth_1_hr
- sea_level_pressure
- wind_direction
- wind_speed

test.csv

- row_id
- building_id
- meter
- timestamp

sample_submission.csv

Clearing and formatting data

Missing value

```
In [12]: metadata.isna().sum()/len(metadata)

Out[12]: site_id        0.00000
         building_id    0.00000
         primary_use    0.00000
         square_feet    0.00000
         year_built     0.53416
         floor_count    0.75500
         dtype: float64
```

building_id	0.00000	0.00000
meter	0.00000	0.00000
timestamp	0.00000	0.00000
meter_reading	0.00000	nan

No Missing values in ‘train’ and ‘test’ datasets

‘floor_count’ and ‘year_built’ variables has large percentage of missing data

Reducing memory usage

for col in <all_data_cols>:
max, min = <max, min>
choosing optimal type for such min and max

int(8/16/32/64)
float(16/32/64)
category

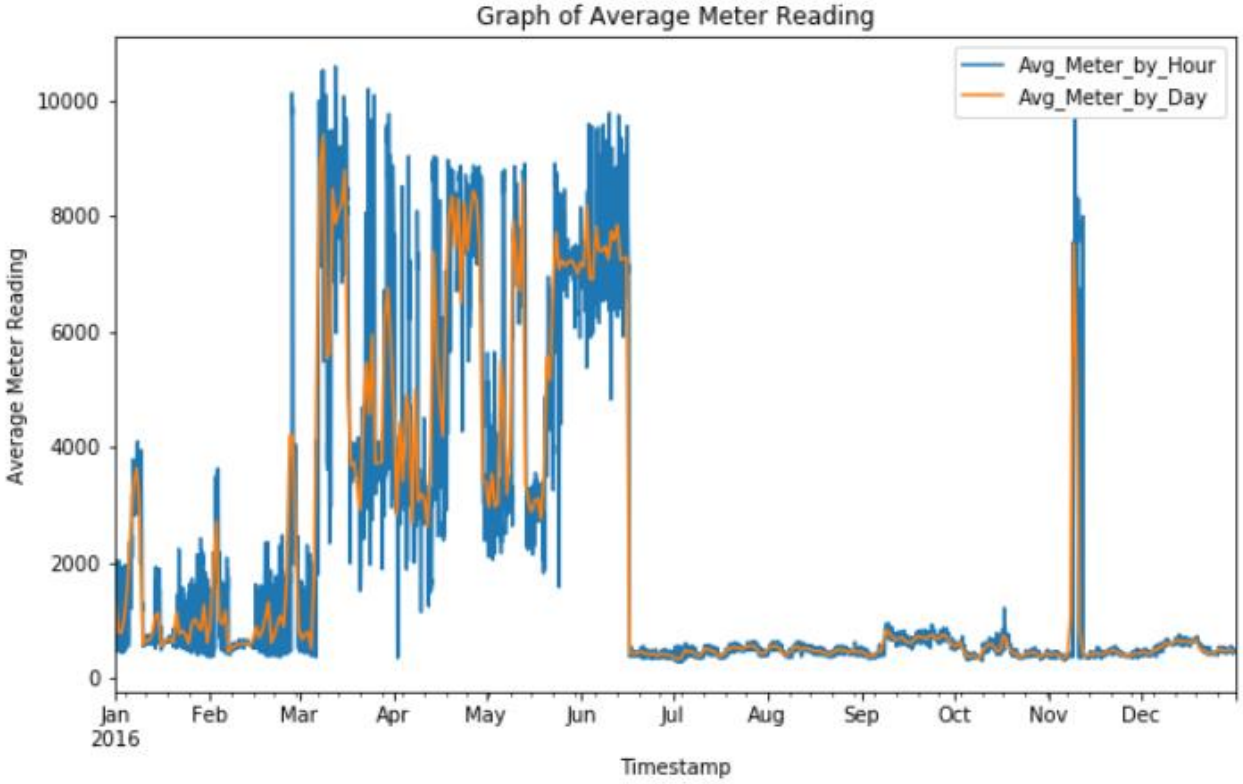
Results:

Train data	**57%**
Building data	**73%**
Weather data	**72%**

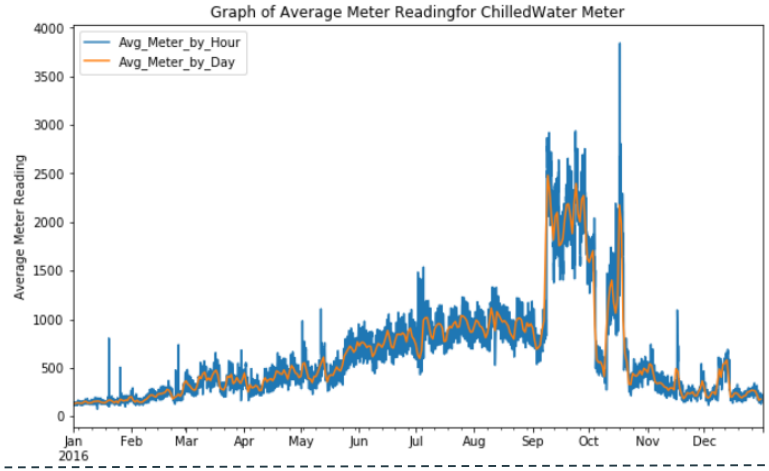
Prior data analysis

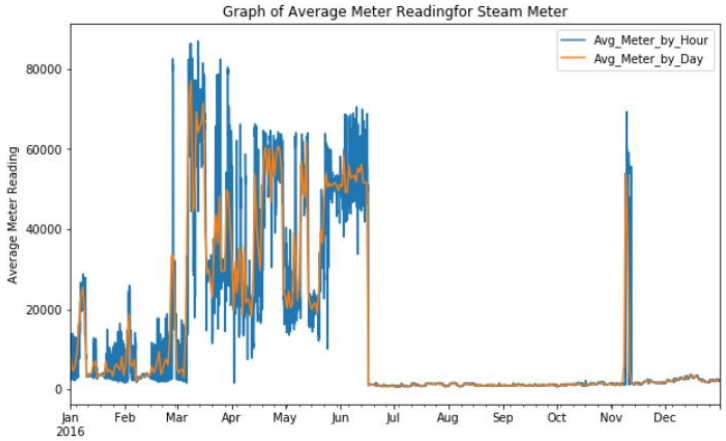
- Train Data contains records from 1st Jan to 31st Dec of 2016-2018
- Data has information about 1448 buildings
- Data has 4 meter types

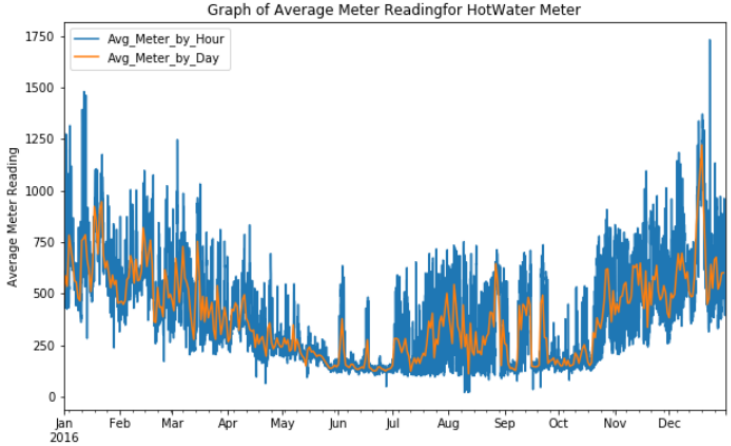
Meter readings are unstable

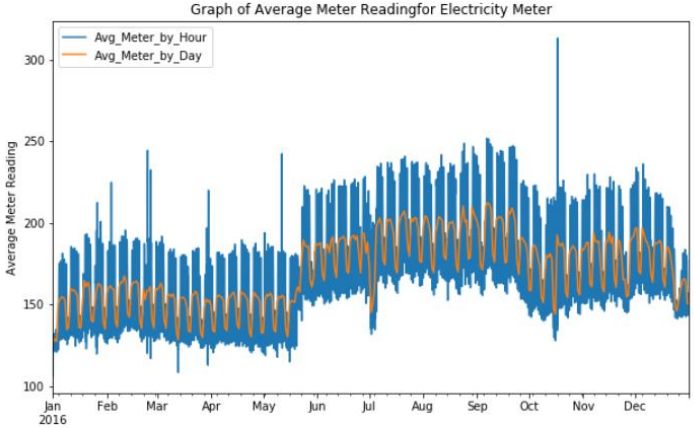


Sharp changes are observed from March to June.
And there's a strange jump in November


Graph of Average Meter Readingfor ChilledWater Meter
Avg_Meter_by_Hour
Avg_Meter_by_Day
Average Meter Reading
4000
3500
3000
2500
2000
1500
1000
500
0
Jan
2016
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec


Graph of Average Meter Readingfor Steam Meter
Avg_Meter_by_Hour
Avg_Meter_by_Day
Average Meter Reading
80000
60000
40000
20000
0
Jan
2016
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec


Graph of Average Meter Readingfor HotWater Meter
Avg_Meter_by_Hour
Avg_Meter_by_Day
Average Meter Reading
1750
1500
1250
1000
750
500
250
0
Jan
2016
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

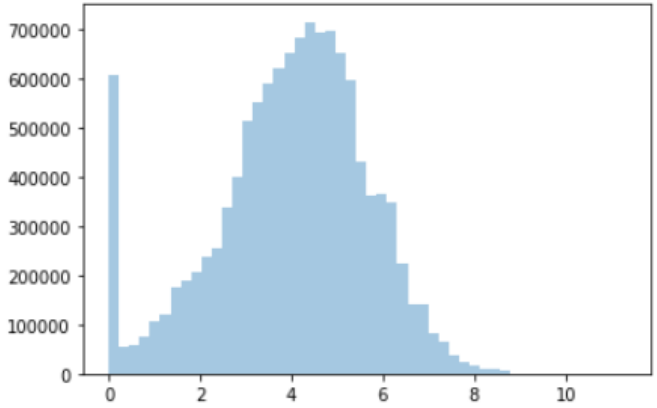

Graph of Average Meter Readingfor Electricity Meter
Avg_Meter_by_Hour
Avg_Meter_by_Day
Average Meter Reading
300
250
200
150
100
Jan
2016
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

The record with a data outlier has been identified. #1099

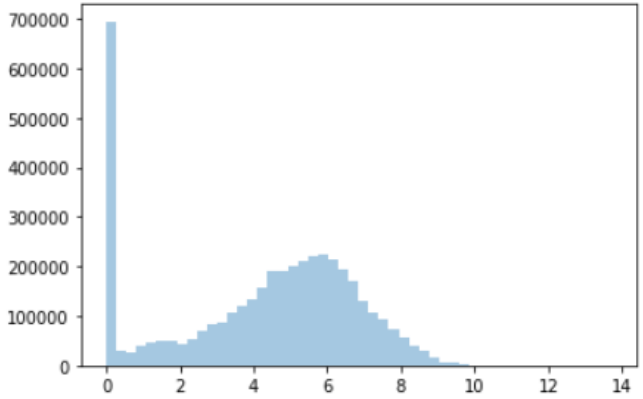
1095	17567	0.00000	93.37930	38.85189	56.69200	29.97182
1096	8783	0.00000	84.23500	28.51193	26.45800	11.73106
1097	26351	0.00000	3117.18994	301.03577	149.06900	481.29950
1098	26352	0.00000	3042.96997	198.31371	31.98350	374.66580
1099	17566	144.00000	21904700.00000	1907446.00000	985.69702	4834351.00000
1100	17567	0.00000	12656.20020	1605.92932	281.25000	2584.48804
1101	17568	0.00000	139.55800	37.55941	37.34600	33.89618
1102	17567	0.00000	7296.87988	918.08575	77.23500	1394.88428
1103	8784	8.83300	76.70000	40.44089	40.22500	13.01057
1104	26352	0.00000	29853.50000	3579.35181	271.57251	5548.45703
1105	17568	11.71870	3011.71997	450.47348	94.98200	674.85461

Some ‘dirty’ data. Missing values, outliers, negative values...

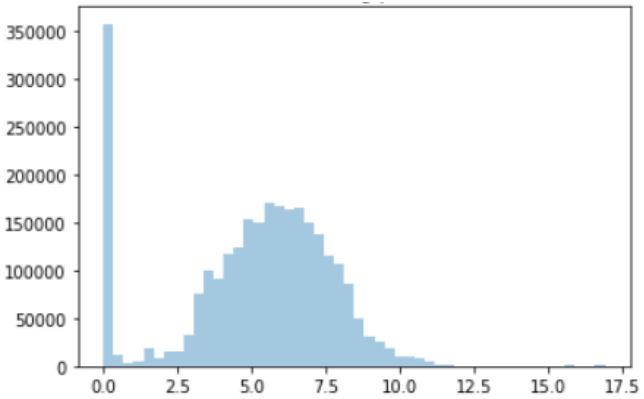
	air_temperature	cloud_coverage	dew_temperature	precip_depth_1_hr	sea_level_pressure	wind_speed
count	139718.00000	70600.00000	139660.00000	89484.00000	129155.00000	139469.00000
mean	14.41811	2.14931	7.35016	0.98305	1016.15804	3.56053
std	10.62660	2.59915	9.79023	8.46368	7.62968	2.33587
min	-28.90000	0.00000	-35.00000	-1.00000	968.20000	0.00000
25%	7.20000	0.00000	0.60000	0.00000	1011.80000	2.10000
50%	15.00000	2.00000	8.30000	0.00000	1016.40000	3.10000
75%	22.20000	4.00000	14.40000	0.00000	1020.80000	5.00000
max	47.20000	9.00000	26.10000	343.00000	1045.50000	19.00000



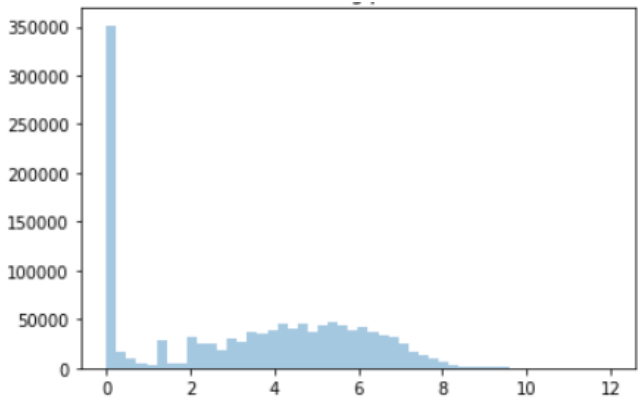
Electricity



Chilled Water

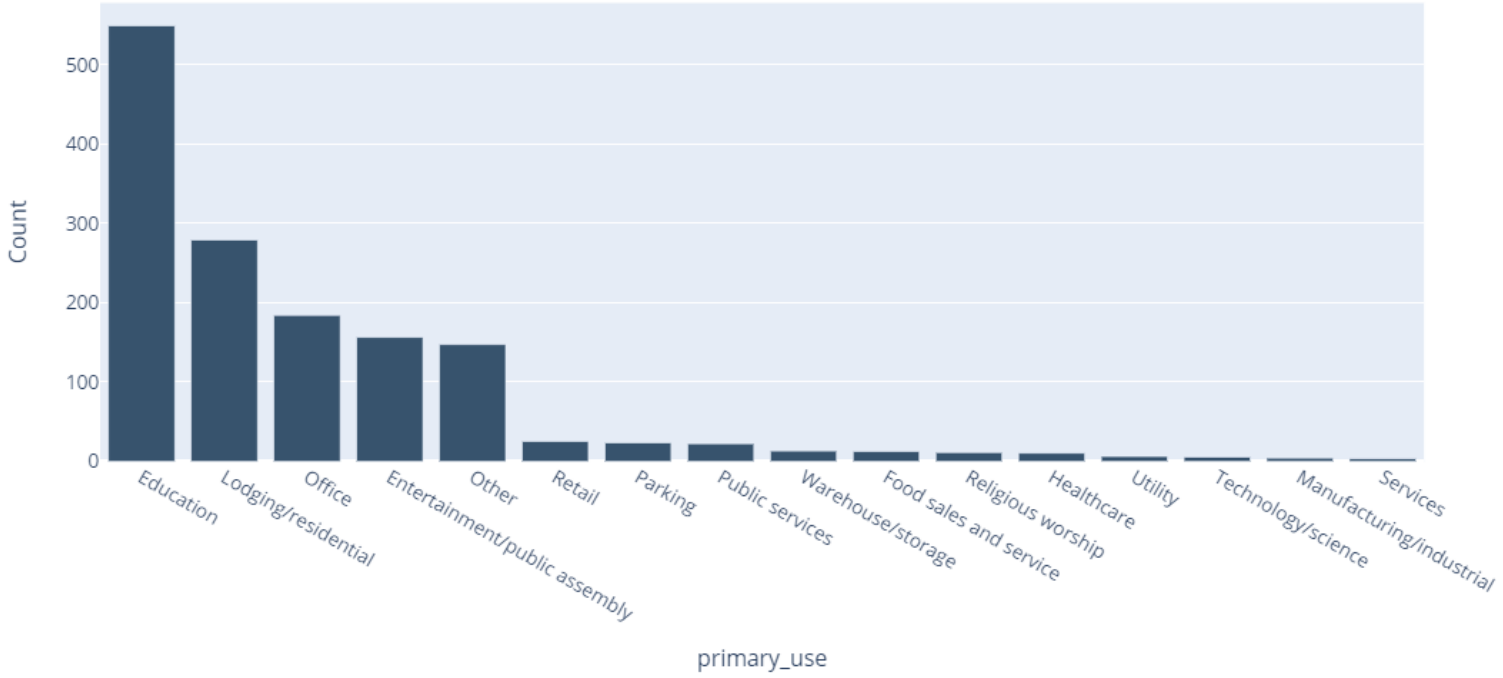


Steam



Hot Water

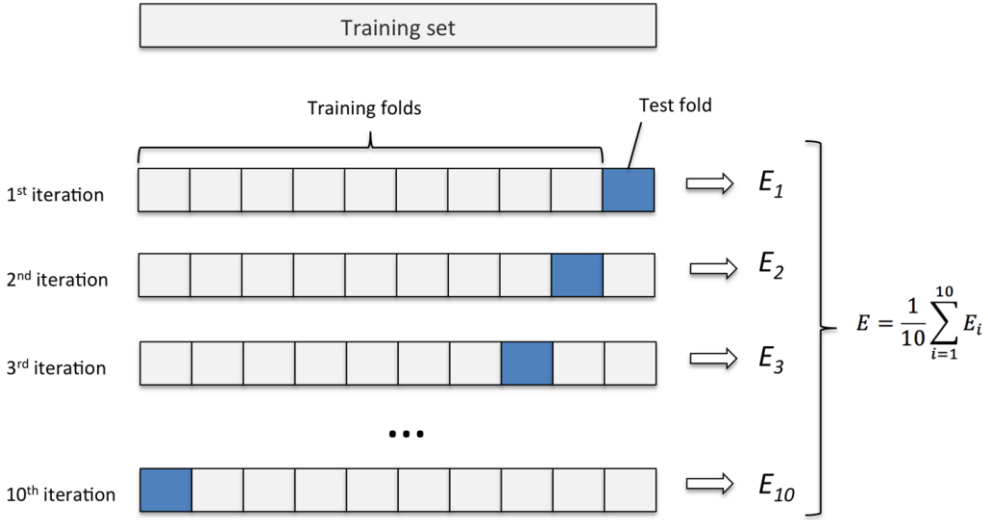
The biggest part of ‘Primary Use’ is Education, Lodging/Residential and Office



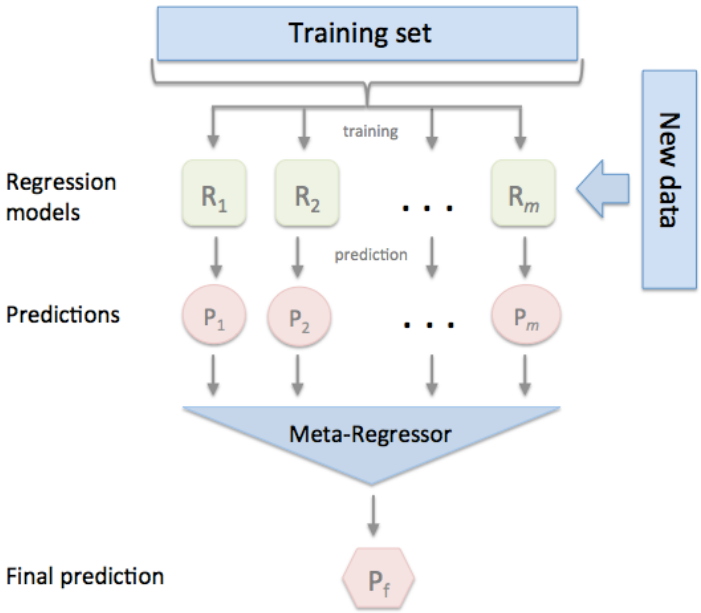
Some theory

KFold + Stacking Regression

1) KFlod
2) Stack Regression
3) Prediction



KFold + Stacking regression




```
lightgbm = LGBMRegressor(
        objective='regression',
        num_leaves=1024, feature_fraction=0.8)

ridge = Ridge(alpha=0.3)
lasso = Lasso(alpha=0.3)

model = StackingRegressor(
        regressors=(lightgbm, ridge, lasso),
        meta_regressor=lightgbm,
        use_features_in_secondary=True)

model.fit(
        np.array(train_features),
        np.array(train_target))
```

Linear Regression (Ridge + Lasso)

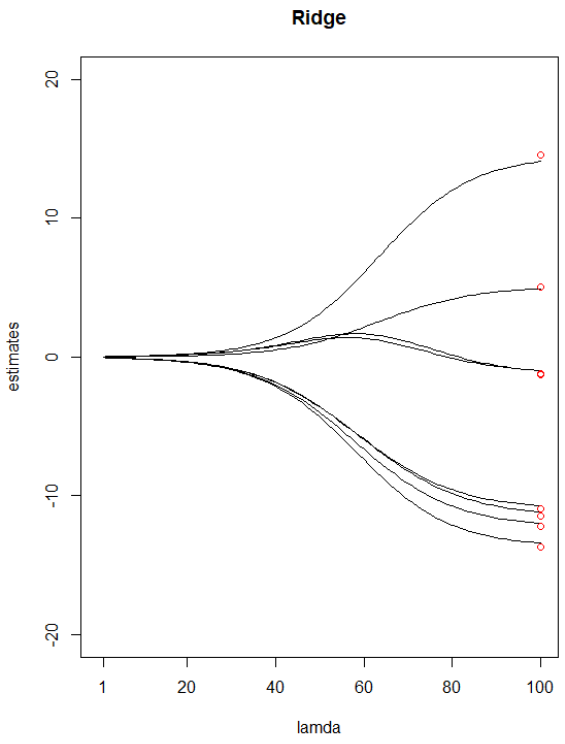
Ridge and Lasso regression are the techniques to reduce model complexity and prevent over-fitting which may result from linear regression.

Ridge:

$$\sum_{i=1}^{M}(y_i-\hat{y}_i)^2=\sum_{i=1}^{M}\left(y_i-\sum_{j=0}^{p}w_j\times x_{ij}\right)^2+\lambda\sum_{j=0}^{p}w_j^2$$

Lasso:

$$\sum_{i=1}^{M}(y_i-\hat{y}_i)^2=\sum_{i=1}^{M}\left(y_i-\sum_{j=0}^{p}w_j\times x_{ij}\right)^2+\lambda\sum_{j=0}^{p}|w_j|$$



Merging Data & Determination of the correlation

Merging our data and make it ‘clean’

```
train = pd.merge(train,metadata,on='building_id',how='left')
test  = pd.merge(test,metadata,on='building_id',how='left')
```

```
Training Data Shape (20216100, 14)
Testing Data Shape (41697600, 13)
```

```
train = pd.merge(train,weather_train,on=['site_id','timestamp'],how='left')
test  = pd.merge(test,weather_test,on=['site_id','timestamp'],how='left')
```

```
Training Data Shape (20216100, 21)
Testing Data Shape (41697600, 20)
```

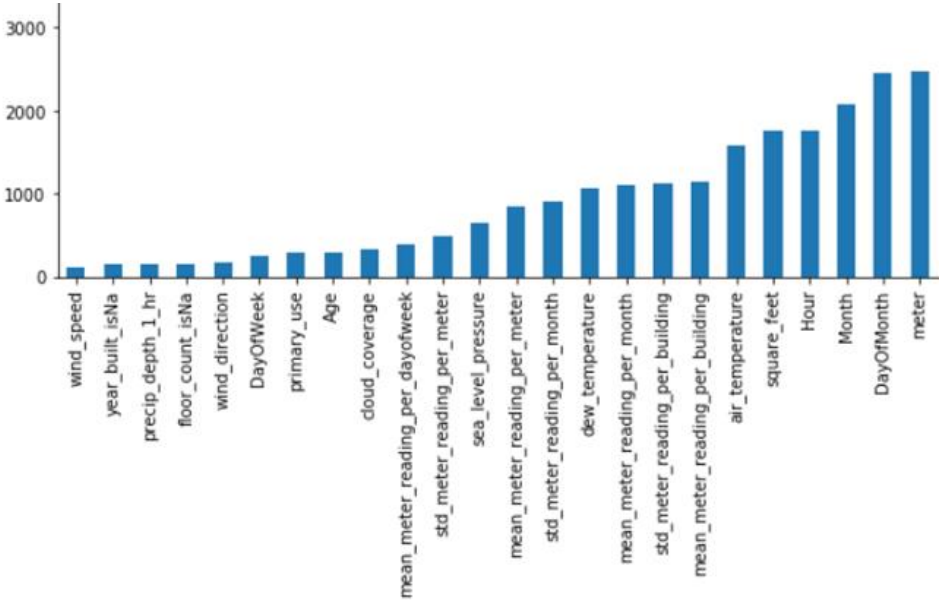
Correlation of values has a significant impact on the data (*multicollinearity)*

Delete data that has a correlation coefficient greater than 0.9

```
corr_matrix = train.corr().abs()
```

Final steps

The most influential values



Laureates!

1658	Vladimir Khomenok		1.106
Your Best Entry ↑			

...

1960	Marina Kiryakina	1.130	1	3h

...

3414	Emre Çetin	4.367	1	1mo
3415	masahito429	4.452	1	25d
3416	Romain Faurès	4.583	1	4d
3417	Yang Wang	4.647	2	6d
	sample_submission.csv	4.699		
3418	Fingal Plumpton	4.697	1	10d
3419	Brandon	4.697	1	2mo
3420	person	4.699	1	1d

Data Leak & Results Leak

Models trained with data leaks and submissions from open kernels show (unexpected!) better results
It seems, everyone is using the same pattern. They use similar algorithms

- reducing memory usage
- clearing data
- adding leaked data for validation
- adjusting weights using other submissions

423	stdy	0.979
424	takashi	0.979
425	KagKor_newbie	0.979
426	westpole	0.979
427	Sasha	0.979