Innovationlab Big Data Science

Energeeks - ASHRAE Great Energy Predictor III

Adrian Uffmann
Dario Lepke
Erjona Dervishi
Tobias Weber

December 11, 2019

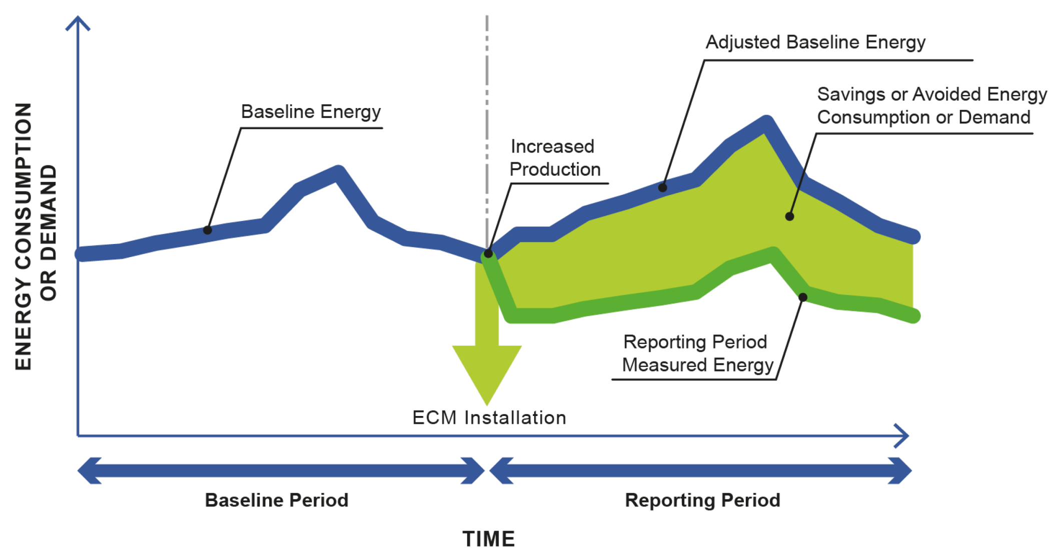
Institut for Statistics

Agenda

1. ASHRAE Great Energy Predictor III
2. Our Project
3. Retrospective
4. Next steps

ASHRAE Great Energy Predictor III

The Challenge



The Dataset

- Training Data
  - Contains $\approx$ 20 mio. rows
  - Timespan: 1 year, from 2016 to 2017
- Test Data
  - Contains $\approx$ 40 mio. rows
  - Timespan: 2 years, from 2016 to 2018
- Target
  - Hourly readings from four different meters (kWh)
    $\rightarrow$ electricity, chilledwater, steam, hotwater
- Features:
  - train.csv: building_id, timestamp, meter, meter_reading
  - building_metadata.csv: site_id, building_id, primary_use, square_feet, year_built, floor_count
  - weather_train.csv: site_id, timestamp, air_temperature, wind_direction, ...

Dataframe.head()

```
In [5]: train.head()
```

Out[5]:

	building_id	meter	timestamp	meter_reading
0	0	0	2016-01-01 00:00:00	0.0
1	1	0	2016-01-01 00:00:00	0.0
2	2	0	2016-01-01 00:00:00	0.0
3	3	0	2016-01-01 00:00:00	0.0
4	4	0	2016-01-01 00:00:00	0.0

```
In [4]: metadata.head()
```

Out[4]:

	site_id	building_id	primary_use	square_feet	year_built	floor_count
0	0	0	Education	7432	2008.0	NaN
1	0	1	Education	2720	2004.0	NaN
2	0	2	Education	5376	1991.0	NaN
3	0	3	Education	23685	2002.0	NaN
4	0	4	Education	116607	1975.0	NaN

```
In [7]: weather.head()
```

Out[7]:

	site_id	timestamp	air_temperature	cloud_coverage	dew_temperature	precip_depth_1_hr	sea_level_pressure	wind_direction	wind_speed
0	0	2016-01-01 00:00:00	25.0	6.0	20.0	NaN	1019.7	0.0	0.0
1	0	2016-01-01 01:00:00	24.4	NaN	21.1	-1.0	1020.2	70.0	1.5
2	0	2016-01-01 02:00:00	22.8	2.0	21.1	0.0	1020.2	0.0	0.0
3	0	2016-01-01 03:00:00	21.1	2.0	20.6	0.0	1020.1	0.0	0.0
4	0	2016-01-01 04:00:00	20.0	2.0	20.0	-1.0	1020.0	250.0	2.6

Our Project

Pipeline Overview

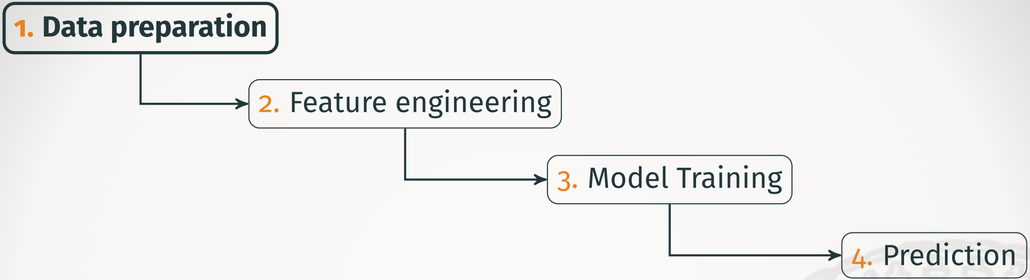
1. Data preparation
2. Feature engineering
3. Model Training
4. Prediction

Project Structure

Using Cookiecutter: `https://drivendata.github.io/cookiecutter-data-science/`

```
+-- data
|   +-- external            <- Data from third party sources.
|   +-- interim             <- Intermediate data that has been transformed.
|   +-- processed           <- The final, canonical data sets.
|   +-- raw                 <- The original, immutable data dump.
|
+-- src
    +-- data
    |   +-- make_dataset.py     <- Data preparation.
    |
    +-- features
    |   +-- build_features.py   <- Feature engineering.
    |
    +-- models
        +-- predict_model.py    <- Prediction.
        +-- train_model.py      <- Model Training.
```

Pipeline Overview



- Exclude faulty readings.
- Impute missing data.
- Align timestamps.
- Merge data frames.

Pipeline Overview

1. Data preparation

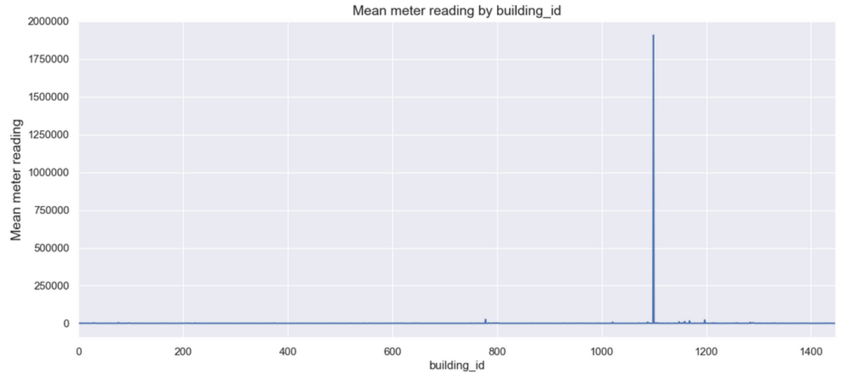
2. Feature engineering

3. Model Training

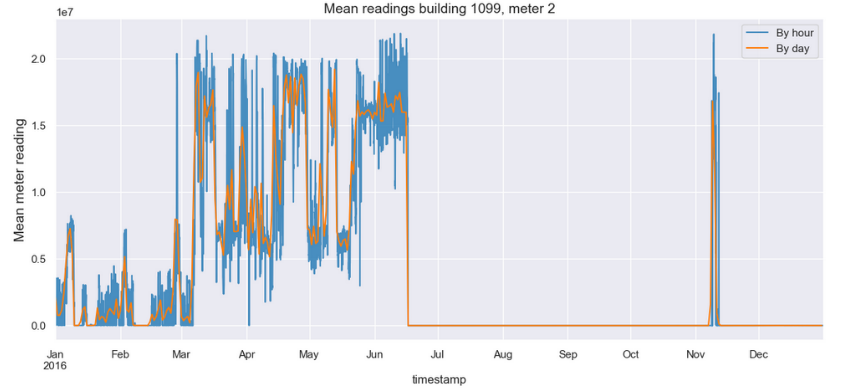
4. Prediction

- **Exclude faulty readings.**
- Impute missing data.
- Align timestamps.
- Merge data frames.

Exclude faulty readings



Exclude faulty readings



Pipeline Overview

1. Data preparation

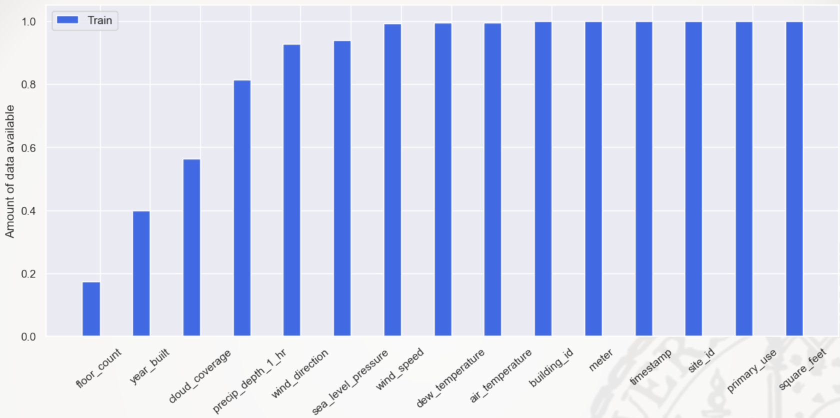
2. Feature engineering

3. Model Training

4. Prediction

- Exclude faulty readings.
- **Impute missing data.**
- Align timestamps.
- Merge data frames.

Impute missing data




```
from sklearn.impute import Imputer
from sklearn.model_selection import cross_val_score
imp = Imputer_name(missing_values=np.nan, strategy='mean, median, most_frequent, zero, knn')
```

Pipeline Overview

1. Data preparation

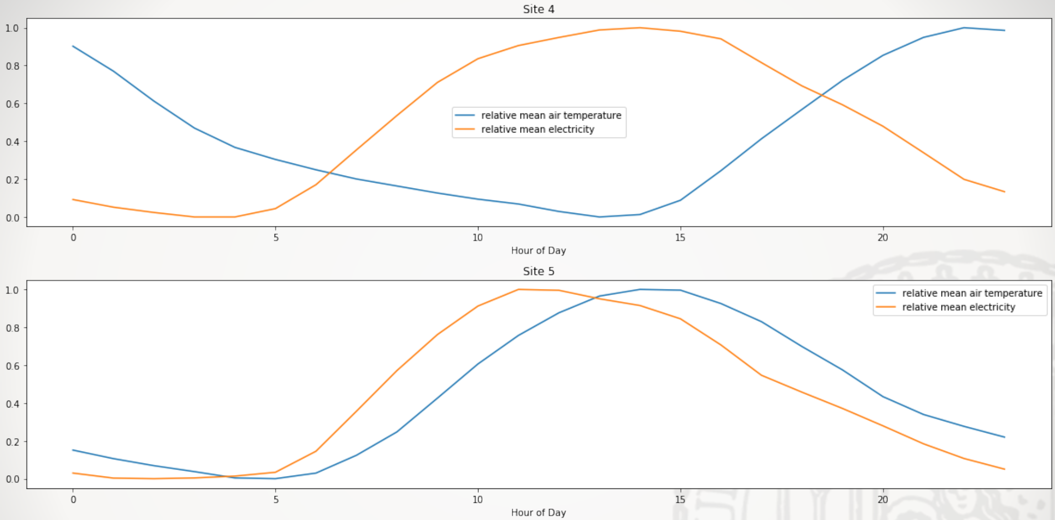
2. Feature engineering

3. Model Training

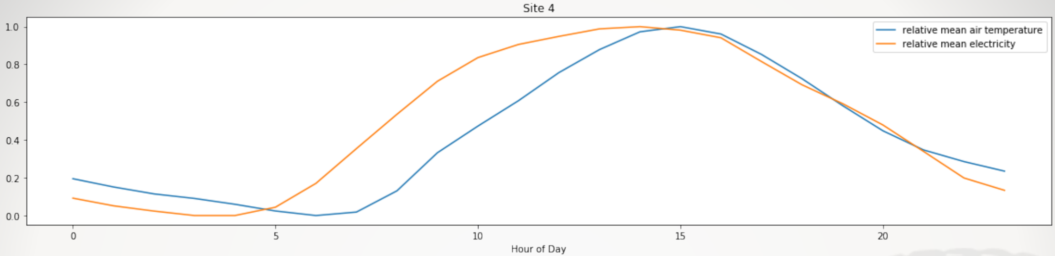
4. Prediction

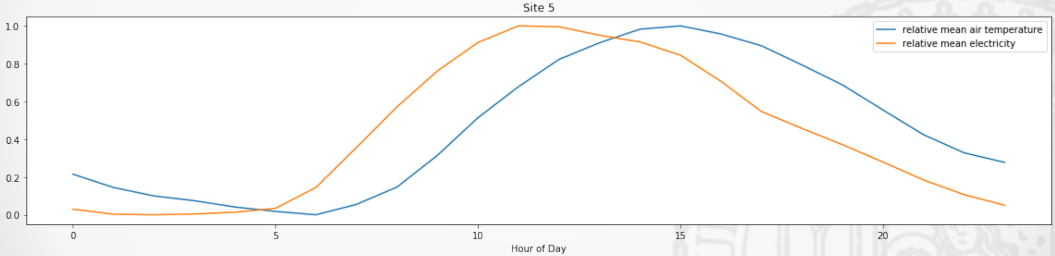
- Exclude faulty readings.
- Impute missing data.
- **Align timestamps.**
- Merge data frames.

Without Timestamp-Alignment



With Timestamp-Alignment





Pipeline Overview

1. **Data preparation**
2. Feature engineering
3. Model Training
4. Prediction

- Exclude faulty readings.
- Impute missing data.
- Align timestamps.
- **Merge data frames.**

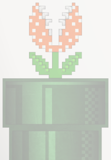
Pipeline Overview

1. Data preparation
2. **Feature engineering**
3. Model Training
4. Prediction

- Encode categorical data.
- Transform year_built into age.
- Logarithmic scaling of square_feet.
- Add lag features.
- Encode cyclic data.

Pipeline Overview

1. Data preparation
2. Feature engineering
3. **Model Training**
4. Prediction



Which frameworks were used?

dmlc XGBoost

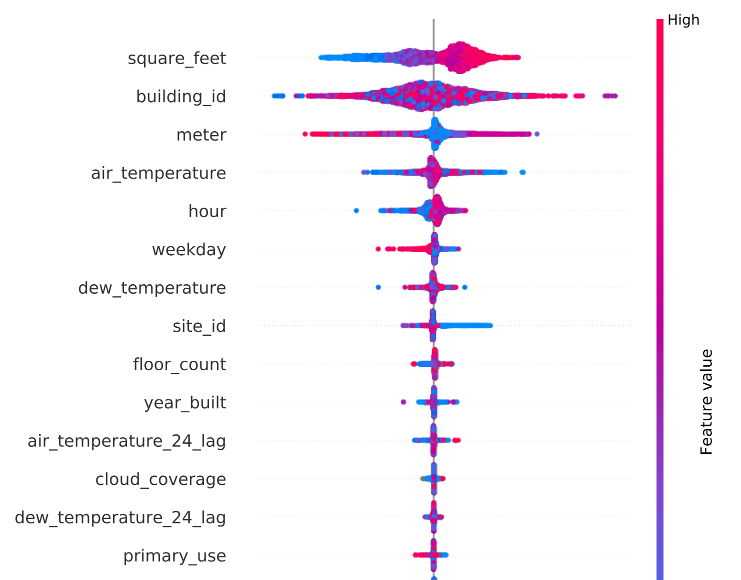
Microsoft LightGBM

Yandex CatBoost

What's the best model? I

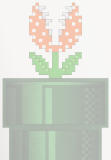
- Favorite framework so far: LightGBM
  - → Extreme fast
  - → Low RAM Usage
- Cross Validation (4-Fold w/o shuffle)
- Mean-stacking strategy
- Bayesian Hyperparameter Optimization with hyperopt
- RMSLE: 1.06

SHAP Values

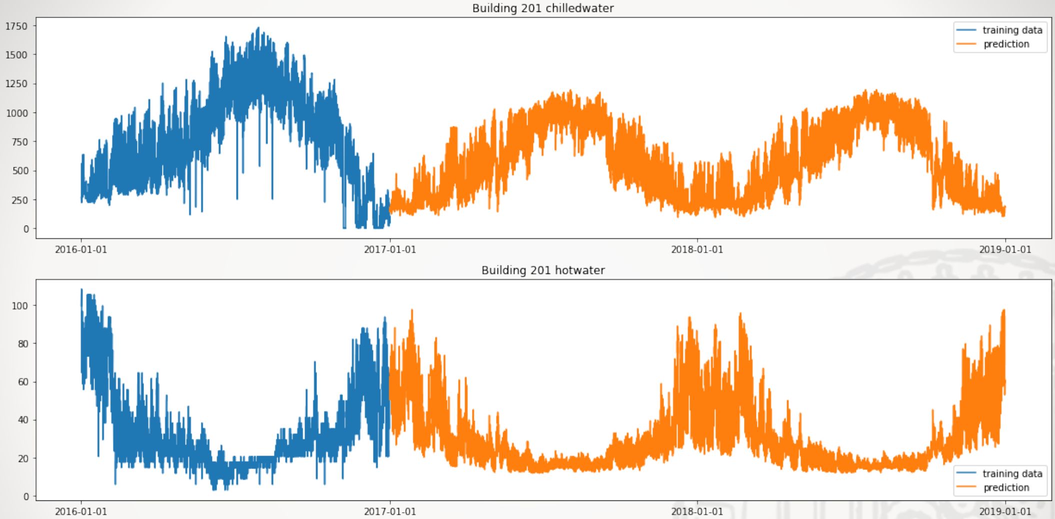

square_feet
building_id
meter
air_temperature
hour
weekday
dew_temperature
site_id
floor_count
year_built
air_temperature_24_lag
cloud_coverage
dew_temperature_24_lag
primary_use
High
Feature value

Pipeline Overview

1. Data preparation
2. Feature engineering
3. Model Training
4. **Prediction**



Prediction


Building 201 chilledwater
1750
1500
1250
1000
750
500
250
0
2016-01-01
2017-01-01
2018-01-01
2019-01-01
training data
prediction
Building 201 hotwater
100
80
60
40
20
0
2016-01-01
2017-01-01
2018-01-01
2019-01-01
training data
prediction

Leaks & a broken Leaderboard

- Identification of sites and buildings via timestamp
- Public availability of energy consumption

→ Scraping of 1 mio. test labels

→ Data Science competition → Web Scraping competition

- Allegedly no leaked data in private Leaderboard

Retrospective

Collaboration

https://github.com/energeeks/ashrae-energy-prediction

GitHub

DISCORD

Problems & Success stories

Problems:

- Underestimation of tickets
- Sprints were a bit difficult to plan (constantly had new ideas)

Success stories:

- Ticket system (Kanban Board)
- Team members have different backgrounds and strengths
- Python notebooks

Next steps

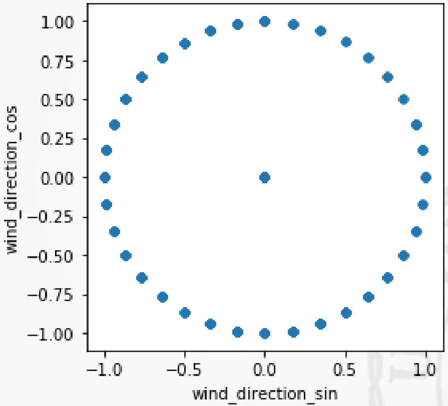
Next steps

- Phase I: Kaggle Challenge
  - Include more features
  - Model stacking/ blending
  - December 19, 2019 - Final submission deadline.
- Phase II: Create web-application
  - User gives building metadata
  - Use model from Phase I to predict future energy consumption
  - Include weather APIs for weather forecasting

Thank you for your attention! :)

Cyclic-Encoding

```
df["wind_direction_sin"] = np.sin(2 * np.pi * df["wind_direction"] / 360)
df["wind_direction_cos"] = np.cos(2 * np.pi * df["wind_direction"] / 360)
df.loc[df["wind_direction"].isna(), ["wind_direction_sin", "wind_direction_cos"]] = 0
df.loc[df["wind_speed"] == 0, ["wind_direction_sin", "wind_direction_cos"]] = 0
```



What's the best model? II

Parameters	**Score: 1.07**	**Score: 1.06**
Boosting Type	GBDT	DART
Early Stopping	YES	NO
Number of Leaves	3480	3630
Learning Rate	0.05	0.05
Training Time	Low	High