A. MODEL SUMMARY

General Guidelines

A1. Background on you/your team

- Competition Name: ASHRAE – Great Energy Predictor III
- Team Name: eagle4
- Private Leaderboard Score: 1.234
- Private Leaderboard Place: 3rd
- Name: Xavier Capdepon
- Location: New York, USA
- Email: xcapdepon@hotmail.fr

A2. Background on you/your team

- What is your academic/professional background?
  Academic: MS Civil Engineering (France) + Advanced Master of Corporate Finance (France)/ Professional: currently Data Scientist/Machine learning Engineer for a start-up.
- Did you have any prior experience that helped you succeed in this competition?
  I have spent many years on Kaggle learning how to build ML models and handling data. In my job, I worked in creating ML models related to computer vision in the past and currently for NLP projects and put code in production since mid-2016.
- What made you decide to enter this competition?
  The fact that the competition shall not require expensive resources (ex: GPU) since it was a tabular data competition.
- How much time did you spend on the competition?
  From 0 to 8 hours per days (I am guessing an average of 2 hours / day over the last month)
- If part of a team, how did you decide to team up?
  NA

- If you competed as part of a team, who did what?
  NA

A3. Summary

- The training method(s) you used (Convolutional Neural Network, XGBoost)
  Decision trees: Lightgbm (on CPU), Catboost (on GPU),
  Neural Network : Dense layers model (Keras on CPU) and CNN layers model (Keras on CPU)
- The most important features:
  This is the most difficult questions. In this competition, it seems that the different models could use different sets of features and achieve approximately the same results. Additionally, the fact that we are training and testing on the same building also push for overfitting and a poor generalization. The building_id was usually the most important features followed by square feet, hour, temperature.

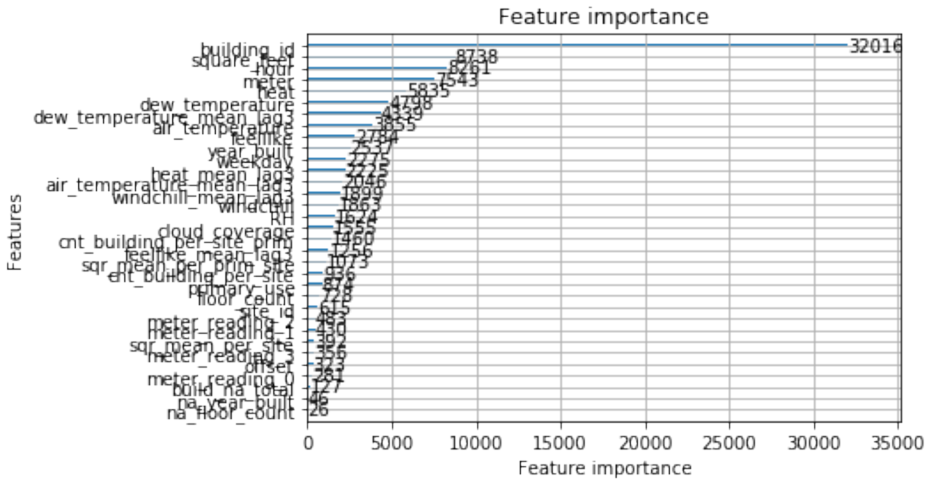
- The tool(s) you used:
  Python Notebook on AWS EC2 instances
- How long it takes to train your model:
  Without parallelization of tasks, on a cumulated basis, it takes about 12 hours.

A4. Features Selection / Engineering

- What were the most important features?

  The building_id was the most important feature in most of my model except for Catboost which recognized the meter as being the most important feature. Since the test set contains the same buildings, this feature was quite essential but it also means that the models are somehow overfitting to this feature and doesn't generalize well. Without verifying it, I also suspected the decision tree models to recognize the building_id by concatenating other features such as the square_feet, primary use and the site_id.

  Also regarding the square_feet, a Kaggler wrote that it was likely used to estimate the missing information of the meter_reading on some of the building training data which can explain its importance.



The chart is extracted from level1—submission_withoutleak001—lightgbm.

- How did you select features?
  the features were selected by looking at CV and LOFO analysis.
- Did you make any important feature transformations?
  Among the most important features, the best features obtained by transformation such as heat, feelike and RH were provided from the meteocalc package.
- Did you find any interesting interactions between features?
  the most interesting one was square_feet x floor_count (I didn't use it in my winning submission as I discovered it really late)
- Did you use external data? (if permitted)
  No except for the leaky data that were downloaded from public kernels

A5. Training Method(s)

- What training methods did you use?
  In the final submission, I only used regression.
  I attempted to use transfer learning using the leak and some clustering techniques at the beginning of the competition but it didn't give me anything.
- Did you ensemble the models?
  Yes
- If you did ensemble, how did you weight the different models?
  Yes, the leak is used here as an hold out set. The predictions from the different models are including in a new model (level 2) as features. The meter raw feature

from the test data is added as a features and the model (Lightgbm) is trained using the leaky rows as target.

A6. Interesting findings

- What was the most important trick you used?
  Unfortunately, using the leaky target on the test data for ensembling work better than a simple average of a large number of prediction without leak. However, to my opinion, it sounds like diversifying the models and training data inputs pushed my final submission to the top (instead of using a single type of model such as Lightgbm)
- What do you think set you apart from others in the competition?
  I don't exactly know.
  I was trying to answer the question: how can I leverage the leak? how can I correct my prediction to make sure that the last 11% of the test set (private leaderboard) is of high quality?
  Additionally, given my experience on Kaggle on this kind of regression and the score method, I knew that a way to reduce efficiently the error on such test set was to diversify a maximum the models and set of features in every models and aggregate them into a single submission.
- Did you find any interesting relationships in the data that don't fit in the sections above?
  Yes, the cleaning of the data seems to be really important. As shown by many posts on the forum, many data has been inputted with 0s. A Kaggler showed that some building meter_reading were inputed based on another similar building. It also appeared that all the values in the meter_reading weren't all in kWh.

  Also, I noticed at the end of the competition that decision tree models such as Catboost seemed to pick up on square_feet*floor_count which tells me that the square feet is most likely the ground floor area of the building, not the total square footage of the building.

  The only feature that I used toward the end and I haven't seen on the forum is to calculate a kind of solar horizontal radiation using the latitude (based on forum post and site_id):

```
latitude_dict = {0 :28.5383,
1 :50.9097,
2 :33.4255,
3 :38.9072,
4 :37.8715,
5 :50.9097,
6 :40.7128,
7 :45.4215,
8 :28.5383,
9 :30.2672,
10 :40.10677,
11 :45.4215,
12 :53.3498,
13 :44.9375,
14 :38.0293,
15: 40.7128,}

train_df['latitude'] = train_df['site_id'].map(latitude_dict)
train_df['solarHour'] = (train_df['hour']-12)*15
train_df['solarDec'] = -23.45*np.cos(np.deg2rad(360*(train_df['doy']+10)/365))
train_df['horizsolar'] =
np.cos(np.deg2rad(train_df['solarHour']))*np.cos(np.deg2rad(train_df['solarDec'
]))*np.cos(np.deg2rad(train_df['latitude'])) +
np.sin(np.deg2rad(train_df['solarDec']))*np.sin(np.deg2rad(train_df['latitude']))

train_df['horizsolar'] = train_df['horizsolar'].apply(lambda x: 0 if x <0 else x)
```

A7. Simple Features and Methods

Many customers are happy to trade off model performance for simplicity. With this in mind:

- Is there a subset of features that would get 90-95% of your final performance?
  I believe the work can be done with a simple model with a few features.

- Which features?
  the features that shall be kept are: building_id, square_feet, meter, heat, feelike, hour, site_id, air_temperature, dew_temperature, primary_house, year_built.

- What model that was most important?
  The easiest to use is Lightgbm for its speed, performance on CPU and its simplicity to use. Catboost required in my case to buy GPU hours, so it was expensive and CNN / DNN requires some data manipulation, it isn't possible to know what features are the most important and it is much longer to train and to predict.
- What would the simplified model score?
  A lightgbm can provide a private score in the range of 1.29 to 1.31 in my case without using the leak. CNN and Catboost gave similar results

- * Try and restrict your simple model to fewer than 10 features and one training method.

A8. Model Execution Time

Many customers care about how long the winning models take to train and generate predictions:

- How long does it take to train your model?

  In total without parallelization of the prediction (generate data, level 1 predictions and ensembling (level 2)), it would take on an accumulated basis about 12 hours.

Generate data (data FE, cleaning and leak): 4.5 hours (one single groupby in pandas takes 3.5 hours but I believe it can be multi-threaded)
LightGbm: 30 mins to 1.5 hours on CPU / total : about 3 hours
Catboost: 1 hour on GPU
Dense NN: ~30 mins
CNN NN: 3.5 hours
Ensembling: 30 mins.

- How long does it take to generate predictions using your model?
  the time to predict takes from 5 minutes using Lightgbm to 60+ minutes with the CNN model.
- How long does it take to train the simplified model (referenced in section A6)?
  my simplest model would take about 30 mins to train
- How long does it take to generate predictions from the simplified model?
  my simplest model would take about 5 mins to predict

A9. References

Citations to references, websites, blog posts, and external sources of information where appropriate.

Kaggle kernels and Kaggle discussions