III. Model documentation and write-up

You can respond to these questions either in an e-mail or as an attached file (any common document format is acceptable such as plain text, PDF, DOCX, etc.) **Please number your responses.**

1. Who are you (mini-bio) and what do you do professionally?

   If you are on a team, please complete this block for each member of the team.

   I am assistant professor in computer science at University of Warsaw and co-founder and data scientist in Codility (SaaS for automated testing of programmers, used by 1k+ companies around the world). My main research area is the text algorithms and efficient data structures for texts. During last two years I'm enjoying machine learning. I'm also a big fan of data visualization and data story telling.

2. High level summary of your approach: what did you do and why?

   First I've divided the problem into three separate sub-problems depending on the prediction window (hourly, daily, weekly). I assumed that each prediction window would require different approach. Since LSTM approach was too complicated and required a lot more computational power, I've decided to use simpler neural networks. But in this case it was necessary to handle different length of inputs (different number of cold start days). The simplest solution was to divide the problem even further by the number of cold start days (to simplify the problem I was not using more than 7 days of cold start data)
   I've ended up with 21 different models (3 prediction windows * 7 cold start days).
   For each subproblem I've tried to select best neural network architecture that would solve it, and the final solution was a mixture of simple FF networks (with 1 or 2 layers) and custom made neural network (used for hourly predictions).

   The custom made neural network was created in such a way, that it analyzed the whole input (all lagging consumption and additional features) and then it computed how to multiply each lagging day to best match target day. At the end all of such partial predictions were averaged.

   Since our team was created in the very last days, we did not have chance for more in-depth cooperation, our final solution was a simple average of our best solutions. It gave suprisingly good result probably due to the fact that our approaches were complementary.

3. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.

The customized model for hourly predictions (used if number of cold start days is >= 6) helped me to obtain better score for hourly predictions:

```
def original_gen_hourly_pred_model(features, cold_start_days=None):
    input_size = len(features)
    output_size = 24

    inputs = keras.layers.Input(shape=(input_size, ), name='input')
    x = inputs
    for layer_num in range(2):
        x = keras.layers.Dense(128, activation='relu')(x)
        x = keras.layers.Dropout(0.2, seed=layer_num)(x)

    out = []
    for i in range(cold_start_days):
        j = features.index(f'consumption_lag_h_{24*(i+1):03d}')
        assert len(features[j: j+24]) == 24
        inp_i = keras.layers.Lambda(lambda x: x[:, j:j+24])(inputs)
        x_i = keras.layers.Dense(1, activation='relu')(x)
        m_i = keras.layers.multiply([inp_i, x_i])
        out.append(m_i)
    avg = keras.layers.average(out)
    avg = keras.layers.Dense(output_size, activation='relu')(avg)
    outputs = avg

    model = keras.models.Model(inputs=inputs, outputs=outputs)
    model.compile(optimizer='adam', loss='mean_absolute_error')
    return model
```

For weekly predictions it turned out that very usefull feature was calculating expected (avg) daily usage in next week. Instead of relying on training the neural network to calculate it on its own, I've simply added such feature.

```
# slen is the number of cold start hours
lc, off, dt = lag_consumption[-slen:], lag_is_day_off[-slen:], lag_dates[-slen:]
c_on = lc[off==False] # hourly consumption on working days
c_off = lc[off==True] # hourly consumption on day-off days
# I was using exponential weighted moving average to follow trend changes
mean_func = lambda x: \
   pd.Series(x.reshape(len(x)//24, 24).mean(axis=1)).ewm(1).mean().iloc[-1]
c_mean_on = mean_func(c_on) if len(c_on) > 0 else c_mean
c_mean_off = mean_func(c_off) if len(c_off) > 0 else c_mean
c_mean_w = (working_days * c_mean_on + (7 - working_days) * c_mean_off) / 7
```

The calculcated c_mean_w is the approximated (avg) daily usage in following week.

During work on the project I've noticed that the distribution of weekdays in the test set is

slightly different than the distribution in my training set, I've tried to fight this problem with either limiting validation set to the dates present in the test set or by boosting (sample) weights of some samples. Boostin weights helped to improve results for the weekly predictions:

```
# res is a train or validate dataframe,
# res['k'] - is an adjustment to the sample weight
sel_test_dates = sel_test_set['date'].unique()
a_len = res['date'].isin(sel_test_dates).sum()
b_len = len(res) - a_len
k = 8
alpha = len(sel) * k / (k * a_len + b_len)
beta = len(sel) / (k * a_len + b_len)
res['k'] = res['date'].isin(sel_test_dates).map({False: beta, True: alpha})
```

Alpha & beta parameters are selected in such a way that the scale of loss computed during training should remain the same.

I've added more in-depth solution description in the project repository.

4. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?

   At the very beginning of the contest I've tried to use (both stateful and non-stateful) LSTM but the results were rather very bad.
   I've also tried to use similar custom made neural network that creates daily predictions out of lagging days with similar achitecture as the one used for hourly predictions (for at least 6 cold start days). But unfortunately I was not able to make it working.

5. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?

   I've used ad-hoc ipython notebooks and that were using pandas, matplotlib and seaborn for visualization of both training data (`notebooks/1-*.ipynb`) and the biggest differencies obtained during validation (`notebooks/2-*.ipynb`).

6. How did you evaluate performance of the model other than the provided metric, if at all?

   Since I was training hourly / daily / weekly predictions indepently, I've slighthly modified the w_i parameters such that the NMAE would be in the same range for all prediction windows. I also had some functions to aggregate the NMAE for all models to single value that resembled LB score.

I was also experimenting with randomized selection of subsets (biased towards the dates used in test set) of validation set for computing the NMAE. You can browse the exact formulas in the validation notebook.

7. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

   Before starting the competition I've spend a lot of investigation to make the Keras deterministic (setting all the random seeds, disabling GPU, multi-threading). And on my computer all runs were 100% deterministic. After the contest I've noticed that the Keras behaves slightly differently on different OS (i.e MacOS vs Linux).

8. Do you have any useful charts, graphs, or visualizations from the process?

   Yes, I've did some initial EDA (for input data), with averge consumption for different type of the buildings. Average temperatures, etc.

9. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?

   I was very suprised with the clustering features (input data series that probably came from the same building) discovered by my team-mate (ironbar).
   Also I have not found any usefull application for holidays. Some standard holidays were used in few models, but in the rest of models the best results were obtained when I've removed all holidays data. But I am not sure if given the limited input data from the contest (no geographical info) it was possible to make the holidays very usefull.