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

My main focus of work is the use of ML in marketing for small companies. As a rule, they have problems with the availability of big data, so I often encounter the need to find or buy data that will help a company at the stage without a qualitative data processing history.

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

My solution contains fastai v1 pipeline with embeddings and additional features based on time, temperature and schedule for each house. For example, I added a rolling mean temperature for different periods of time. At the start of the competition, scaling to each building played a critical role, however, I also abandoned the idea of teaching separate models for them, as some winners did in the last competition in predicting electricity consumption. A significant point was the rejection of the use of features based on the history of the house, as well as target encodings.

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

1) This function was part of the fastai v0.7 library, but at the time of solving the problem was not available in the current version v1. Nevertheless, its transfer helped quickly add almost all features based on time to the model.

```
def add_datepart(df, fldname, drop=True, time=False):
    fld = df[fldname]
    fld_dtype = fld.dtype
    if isinstance(fld_dtype, pd.core.dtypes.dtypes.DatetimeTZDtype):
        fld_dtype = np.datetime64
    if not np.issubdtype(fld_dtype, np.datetime64):
        df[fldname] = fld = pd.to_datetime(fld, infer_datetime_format=True)
    targ_pre = re.sub('[Dd]ate$', '', fldname)
    attr = ['Year', 'Month', 'Week', 'Day', 'Dayofweek', 'Dayofyear',
            'Is_month_end', 'Is_month_start', 'Is_quarter_end',
            'Is_quarter_start', 'Is_year_end', 'Is_year_start']
    if time: attr = attr + ['Hour', 'Minute', 'Second']
    for n in attr: df[targ_pre + n] = getattr(fld.dt, n.lower())
    df[targ_pre + 'Elapsed'] = fld.astype(np.int64) // 10 ** 9
    if drop: df.drop(fldname, axis=1, inplace=True)
```

2) Here is the part of the code that processes the temperature information in the submission_format.csv. With a large number of data gaps, there was a great temptation to make a simple temperature filling with medians. Also, the temperature for daily and weekly predictions was averaged, which did not fit my model. However, I added both the average temperature per day and the average temperature per week in addition to the existing rolling mean for the hourly data to use these small additional data from submission_format.csv.

```
w_submit = submit.loc[submit.prediction_window=='weekly', ['series_id', 'temperature']]
w_submit.loc[:, 'temperature_w2'] = w_submit.temperature.shift(-1)
w_submit.columns = ['series_id', 'temperature_w', 'temperature_w2']
w_submit.drop_duplicates(subset='series_id', keep='first', inplace=True)
w_test = join_df(w_test, w_submit, 'series_id')
ids = w_test.series_id.unique()
for _id in w_test.series_id.unique():
    mask = w_test.series_id==_id
    index = w_test.loc[mask, :].index[0]+168
    w_test.loc[mask&(w_test.index>=index), 'temperature_w'] = w_test.temperature_w2
w_test.drop('temperature_w2', axis=1, inplace=True)
w_test.loc[:, 'temperature_d'] = np.nan
X.loc[:, 'temperature_d'] = np.nan
X.loc[:, 'temperature_w'] = np.nan
```

3) Using the fastai v1 pipeline helped me in the competition, although it provided additional difficulties because at the time of writing the best solution the library was still in the alpha version and changed daily. However, its use has allowed me to use adamw optimizer, 1cycle fit, and general feature normalization.

```
data = TabularDataBunch.from_df(models_path, X_train, X_val, test_df=X_test,
    dep_var='consumption', tfms=tfms, cat_names=cat_names,
    cont_names=cont_names, log_output=False, bs=2048)
learn = get_tabular_learner(data, [512, 512, 256, 64, 8, 1], emb_szs=emb_szs,
    ps=[0.0, 0.0, 0.0, 0.0, 0.0], emb_drop=0.0, y_range=None,
    use_bn=True, loss_func=F.l1_loss, true_wd=True,
    callback_fns=[ShowGraph, SaveModelCallback], wd=0.001, bn_wd=False)
learn.fit_one_cycle(50, 0.01)
```

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

I tried to apply a vast number of models: LightGBM, Pytorch NN with a complete history for each hour, Facebook Prophet per series_id with stacking, LSTM and even elements of non-ML solutions. Honestly, I am happy that my best solution is one neural network with a small number of features since this is what I would like to use in practice at the customer site.

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

I used the visualizations provided by the organizers, the most useful was the visualization of the predictions for individual buildings.

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

Validation was complicated by 4 points: different time windows for prediction, different depth of history, completeness of temperature data, as well as the presence of very non-standard buildings in the data, each of which could significantly influence the final result. It was a major headache during the entire competition. The model was very easily retrained for too good training data, and validation could not predict movement on the leaderboard. For example, my two best models, which showed a very close final result of 0.2597, 0.2598, showed entirely different results of 0.2854, 0.2895 on the public leaderboard and fundamentally different results on local validation: 0.4784, 0.5157.

In the middle of the competition to solve this problem, I even created 150 datasets with augmentation, so that the training data would better match the test data. This approach did not improve the result, which is extremely positive for the simplicity of the final model.

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.)?

My model was trained using the same 11gb Nvidia 1080ti, as well as 128Gb RAM. The amount of RAM should not be a bottleneck, but a video card with an appropriate amount of memory is essential to avoid reducing batch size.

it is worth paying attention to the fact that at the moment the fastai v1 library is frequently updated, and therefore with further use of the model I would consider translating it into pure pytorch v1.

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

I used visualizations based on the LSTM pipeline provided by the Drivendata organizers. Although my solution did not use any parts of the baseline-pipeline code, it was useful.

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

Models will be greatly assisted by more data, as well as the availability of quality deferred validation. I would also like to take into account the geo-data, as it seriously affects the patterns of electricity consumption by the hour, on weekends, and allows you to highlight holidays.