III. Model documentation and write-up

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

- I worked as a data scientist in a IoT startup located in Frankfurt. My daily work is to develop machine learning models for the collected data from different sensors and make analysis. I have a master degree in computer science and communication engineering. I learned machine learning all from online open courses and books.

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

- I think there are three key points, which give me a good result, especially in private leaderboard. The first one is the way I split the data to training data and validation data. Because the provided data is time series dataset and the performance is based on 625 buildings, of which only small amount data is provided, so I use all 758 buildings' data, which cover 4 weeks time, as training dataset. For the other 625 buildings dataset, based on time, I select a very small part of most recent dataset as validation data to find the best iteration number of the training models and then use all data for retraining. The second thing is I build two kinds of models, one I called 'hourly' model and another 'daily' model. For hourly predictions of next 24 hours, I used hourly model and for daily and weekly predictions of next 7 days and 2 weeks, I used daily models to forecast. The last thing is the way I normalize the consumption values, especially for daily model. Because for some cold start buildings, only 1 day of data is provided, for daily model, we have only 1 value available. Normal normalization methods will fail in this case. So I normalize the daily consumption based on minimal, maximal and mean hour consumption values, which need to be multiplied by 24. As to prediction models, I build one LSTM based neural network and one lightgbm model. I tried removing anomaly consumptions but with no success. Not much feature engineering work is involved, but only historical consumption values of each building.

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

1. The first impactful part is absolutely the data split strategy I used in this competition. As shown, I used all the 758 buildings data, which contains 4 weeks' data as training data and select a small part of buildings' data, which are provided as cold start data, as validation data to do local validation.

```python
def split_data(df, mode='hourly', id_start=758, id_end=1400):
    tr_val_idx = []
    ids_ = df.series_id.unique().tolist()[id_start:id_end]
    df_tmp = df[df.series_id.isin(ids_)].copy()
    for i, id_ in enumerate(ids_):
        df_tmp2 = df_tmp[df_tmp.series_id == id_]
        nr = df_tmp2.shape[0]
        if id_ in df.series_id.unique().tolist()[:758]:
            nr_val = 0
        else:
            if mode == 'hourly':
                nr_val = min(24, np.power(2, nr // 24 - 1))
            elif mode == 'daily':
                nr_val = min(3, nr // 3)
            elif mode == 'weekly':
                nr_val = max(0, nr - 4)
        tr_val_idx += ['tr'] * (nr - nr_val) + ['val'] * nr_val
    df_tmp['train_val'] = tr_val_idx
    return df_tmp
```

2. The second thing, which is very impactful is how I normalize the consumption values. I calculate the minimal, maximal and mean hour consumption values for each building and then based on the type of our built model, I select the factor we need to multiply to do normalization for that model. For example, for daily model, because what we need to predict is daily consumption, so I multiply 24 to the minimal, maximal and mean hour consumption of each building and then normalize the daily consumption value to the range between -1 and 1. This helps a lot because in this competition the performance metrics is normalized mean absolute error.

```python
def normalize_consumptions(df, mode='hourly'):
    if mode == 'hourly':
        df['consumption'] = (df['consumption'] - df['con_hour_min'] + 1e-5) / (df['con_hour_max'] - df['con_hour_min'] + 1e-5) * 2 - 1
    elif mode == 'daily':
        df['consumption'] = (df['consumption'] - df['con_hour_min'] * 24 + 1e-2) / (df['con_hour_max'] * 24 - df['con_hour_min'] * 24 + 1e-2) * 2 - 1
    elif mode == 'weekly':
        df['consumption'] = (df['consumption'] - df['con_hour_min'] * 24 * 7 + 1e-2) / (df['con_hour_max'] * 24 * 7 - df['con_hour_min'] * 24 * 7 + 1e-2) * 2 - 1
    else:
        print('Wrong mode...')
        return
    for col in tqdm(df.columns):
        if 'consumption_prev_hour_' in col:
            df[col] = (df[col] - df['con_hour_min'] + 1e-5) / (df['con_hour_max'] - df['con_hour_min'] + 1e-5) * 2 - 1
        elif 'consumption_prev_day_' in col:
            df[col] = (df[col] - df['con_hour_min'] * 24 + 1e-2) / (df['con_hour_max'] * 24 - df['con_hour_min'] * 24 + 1e-2) * 2 - 1
        elif 'consumption_prev_week_' in col:
            df[col] = (df[col] - df['con_hour_min'] * 24 * 7 + 1e-3) / (df['con_hour_max'] * 24 * 7 - df['con_hour_min'] * 24 * 7 + 1e-3) * 2 - 1

    return df
```

3. The other technique which helps me a lot is to write one custom evaluate function and use it as model metrics. This is possible in this competition because of the way I split the dataset. The training and validation dataset is fixed in my case, so I can calculate the minimal, maximal and mean values for each building in the training dataset and validation dataset in advance and use them to calculate the normalized mean absolute error for validation dataset.

```
def nmae(preds, train_data):
    ys = train_data.get_label()
    inv_ys = (ys + 1) / 2 * (val_maxs - val_mins) + val_mins
    inv_preds = (preds + 1) / 2 * (val_maxs - val_mins) + val_mins
    nmae = np.mean(abs(inv_preds - inv_ys) / val_means)
    return 'nmae', nmae, False

lgb_model_hour = lgb.train(params, lgb_train, num_boost_round=100000, valid_sets=[lgb_val],
                           early_stopping_rounds=500, verbose_eval=500, feval=nmae#, fobj=nmae_obj,
                    )
```

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

- One thing I spend quite a lot of time but didn't helps at all, is trying to find the anomaly values in the consumption data. From what I saw, there are at least two types of abnormal values, one is some obvious imputing values, the pattern of which is there exists constant value in certain time range. Another type of anomaly values is some extreme consumption values. But after I remove those two types of anomaly values and impute the second type of anomaly values using their previous consumption values. The performance score in the public leaderboard is improved a very little bit. But in private leaderboard, the performance scores are much worse than the performance scores without doing anything to obvious anomaly consumption values. My guess is for test data, the ground truth consumption data, which we need to predict, also contains those two types of anomaly values that I mentioned, in which case, if I remove those anomaly values, the performance score in some parts of test data can be worse.

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

- No

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

- No, just used the provided evaluation performance metrics.

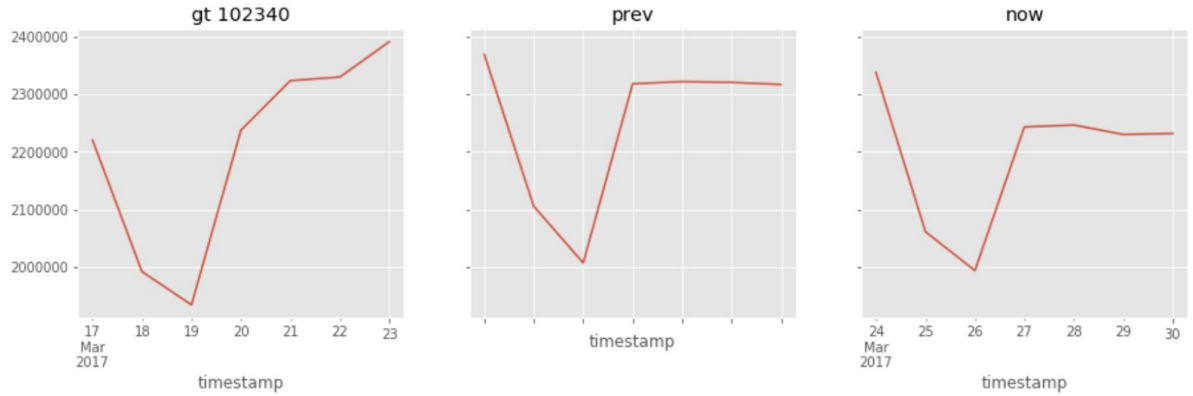
7. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

- The memory requirement should be no less than 16 GB. The overall running time to regenerate result is about three and half hours. For my best submission, which gives the best private score, I don't use XGBoost model. But from my later experiment, it seems, adding XGBoost models helps the performance a little bit. So I added them here.

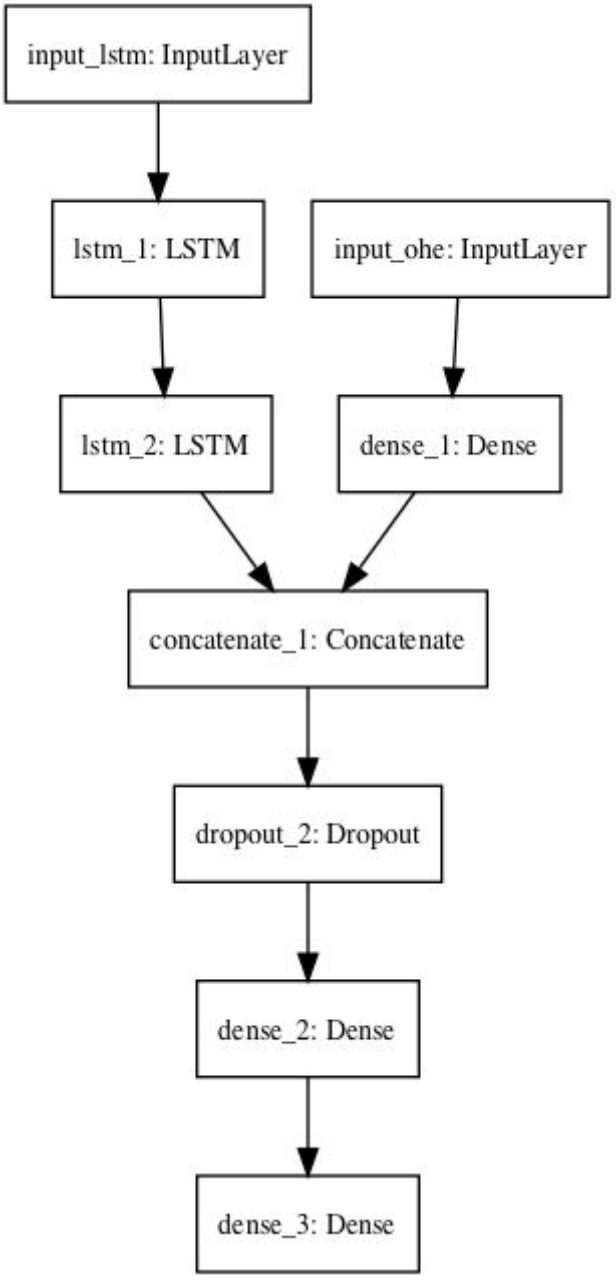
8. Do you have any useful charts, graphs, or visualizations from the process?

- I find it's very useful to plot the prediction and compare it with the ground historical data and previous predictions to see if the new prediction makes sense or have any

obvious improvement.



- And the neural network architecture is as following:



Two kinds of inputs for the network, one for lag consumption values and one for other one-hot encoded categorical features. I didn't use temperature for neural network models, but use it for tree models.

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 will definitely try ARIMA model.