II. Code submission and result reproducibility

- clear.ipynb – straight forward data preparation process, separates train.csv into 3 different files by meter_id, for 38_9686 gets rid of cumulative data + small outliers cleaning. Files *submission_format.csv* and *train.csv* should be in folder *./input*
- 234_203.ipynb, 38_9686.ipynb, 334_61.ipynb – individual notebooks for each meter_id, input data is "clean" files from previous script. Run the entire notebook to get the result (or just watch it through), it takes about 5 seconds to run. Results will appear in *./output* folder. Key parameter is sigma ~ 4.5-5 – number of standard deviations.

Hardware:
- 1 core CPU
- 4GB RAM

Requirements:
- jupyter notebook
- python 3+
- pandas
- numpy
- matplotlib
- tqdm
- seaborn

III. Model documentation and write-up

1. Who are you (mini-bio) and what do you do professionally? - I graduated from Bauman Moscow State Technical University in 2013 with the master degree in biomedical engineering systems. As a part of my master thesis, I worked on medical image recognition systems. After spending 3 years in Johnson & Johnson company I switched to freelance work with some time left for casual participation in data science competitions.

2. High level summary of your approach: what did you do and why? - Hand picking and LB probing algorithm taking advantage of optimization metric. Huge penalty for FP answers convinced me to find several concrete examples of anomalies and to get public score around 80%. Other motivation for that was to gain an understanding of what is meant by anomaly. To find these examples I grouped-by observations with similar time related features and marked 4 sigma deviations as anomalies. Then by manually examining groups of anomalies I started LB probing for each meter_id separately. After a week or so I got the expected score. Then I realized that probably public and private test sets do not intersect, so I gave up. It turned out that I was wrong and private dataset had a huge overlap with the public one (or it is a bug in scoring). Anyway, my approach to finding initial candidates for anomalies might be useful.

3. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.
   1) cols = ["months","WeekDay","hours"] #334_61 – time features for group-by varied for different meter_id;

2) ds = df[(df["years"]!=2013)&(df["years"]!=2018)] – some observations had to be excluded due to the small number of occurrences;
3) *dm = dict(ds.groupby(by=cols)["Values"].mean()); dv = dict(ds.groupby(by=cols)["Values"].std()); t = dm[x]/dv[x]* – time features were picked up by maximizing average z-score. Obvious restriction: there should be enough observation in each group (>50).

4. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)? - PCA approach as described here https://www.kaggle.com/victorambonati/unsupervised-anomaly-detection

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? - LB score only

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.)? - No

8. Do you have any useful charts, graphs, or visualizations from the process? - Yes, the graph can be found at the end of the notebooks

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? - LSTM