AI/ML for 5G-Energy Consumption Competition Solution

Ning Jia

https://www.linkedin.com/in/ning-jia-04b55b241/

Contents

Competition Challenges

Mapping signals to tag names

I found it challenging to associate HL-2A signals and tag names. So, I ended up only using J-Text and C-Mod data.

Fewer samples of C-Mod

The test dataset is for C-Mod, but we only have 20 samples of C-Mod. We need to use about 2000 J-Text samples to generalize the signals well.

Competition Solution

Summary

Given the nature of this competition, I’d prefer an easy solution that doesn’t depend on complicated algorithms or features and too many parameter tunings. Hopefully, this relatively simple solution will generalize well for unseen data.

The solution, from training to test submission, is implemented with an ipython notebook. It should take less than 20 minutes to complete running the whole notebook.

Algorithm

Tree-based algorithms or deep learning like autoencoders work well for event/anomaly detection in high-frequency signals.

Here, LightGBM with 5 StratifiedKFold folds is used. The easy tree-based algorithm already performs well; it seems we should not bother with deep learning for this dataset.

Signal Extraction and Preprocessing

I use the provided jddb package to read the data files.

However, some tag names of C-Mod specified by the **ITU data - signal.csv** are not available in the C-Mod dataset.

I only use the following 25 signals:

MIR signals:

'poloidal Mirnov probes_01',

'poloidal Mirnov probes_02',

'poloidal Mirnov probes_03',

'poloidal Mirnov probes_05',

'poloidal Mirnov probes_06',

SXR signals:

soft-X-ray from 1 to 20.

It turns out that the model can have good performance with features extracted from these 25 signals.

Because J-Text and C-Mod signals are in different scales, all signals will be **normalized** after extraction.

Observation Window

The problem is phrased as a binary classifier for the following status of the short file. So only the latest observation is relevant. We will use an observation window defined as last 300 data points in the shot files, which is **60 ms** given the sample rate of 5000 Hz.

Feature Engineering

We only need Scipy.signal and Numpy to build features.

We will build 4 features from two perspectives for each normalized signal in the observation window.

From the **frequency domain**, we will calculate the spectral entropy. Spectral entropy encodes the frequency distributions and has information like the dominated frequencies.

From the **time domain**, we will calculate the standard deviations and rolling window based standard deviations. These features capture the variance of the signals from different scales.

So, we end up with a total of 25*4=100 features. The features can be fine-tuned further with proper feature selection approaches.

Sample Weights

In the training set, we have 1443 samples from J-TEXT (we dropped some J-TEXT files because they don't have enough observations defined by the observation window) and 20 samples from C-Mod. In the test set, we have about 500 samples from C-Mod.

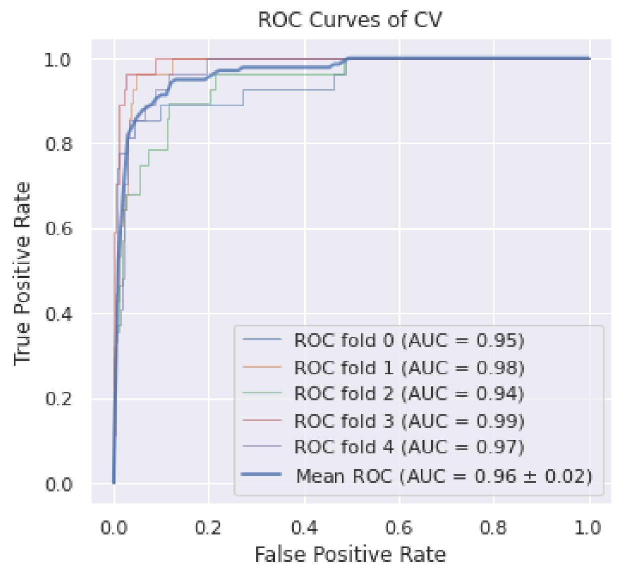
I put more weight on C-Mod samples to focus on the C-Mod data. The solution uses 15 for C-Mod samples and 1 for J-Text.

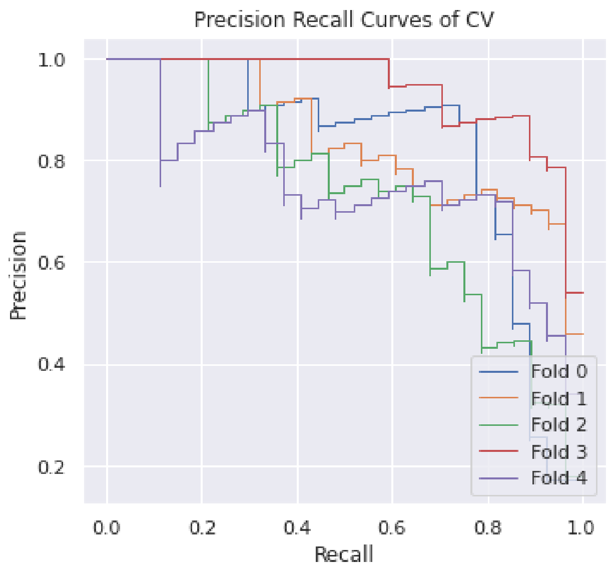
Loss function

Since we treat this competition as a binary classification problem, I use AUC as the loss function and set 'is_unbalance' True. The AUC generally works well for binary classifiers if the data is not extremely imbalanced.

Model performance

Blew shows the AUC and precision-recall curve for each fold.





True positive and false positive rates at 60 ms before disruption: 0.91 and 0.1 respectively.

Mean AUC (area of under ROC curve) for 5 folds: 0.97.

F1 Score

The competition uses F1 score to evaluate the performance. I don't think it's a good choice. Reasons:

1. The model will have hard predictions, which means it cannot output the uncertainties of the predictions, and we must tune the threshold to optimize the F1 score.
2. It cannot evaluate how close the predictions are to the ground truth regarding time difference.

The average precision of detected events, which is averaged over timestamp error tolerance thresholds, is a good metric for this competition. Please refer to Kaggle site: https://www.kaggle.com/code/metric/event-detection-ap/notebook

The model's output is un-calibrated probabilities. F1 score requires hard predictions like 1 or 0.   To maximize the F1 score, I pick two thresholds tuned by calculating the F1 score on out-out-sample (or OOF) predictions for two final submissions.

Performance on Test Datasets

Public F1 score: 0.857142857

Private F1 score: 0.897959183