Supplementary Material for Event Effects Estimation on Electricity Demand Forecasting

For a detailed description of machine learning techniques, we prepare tuning parameter candidates as follows:

- In SVM, we use the Gaussian Kernel as Kernel function. In this case, we have three tuning parameters: σ, C, and ϵ. σ is used in the Gaussian Kernel given by $K(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\sigma\|\boldsymbol{x} - \boldsymbol{x}'\|^2)$, and C corresponds to the regularization parameter in the SVM problem. ϵ is used in the regression version of SVM; data in the ϵ-tube around the prediction value are not penalized. The candidates for these parameters are $\sigma = 0.001, 0.01, 0.1$, $C = 1, 10, 100$, and $\epsilon = 0.001, 0.01, 0.1$.

- The tuning parameters of random forest include number of trees, n_{trees}, and number of variables sampled at each split, n_{var}. The candidates of these parameters are $n_{trees} = 50, 100, 500$, and $n_{var} = 1, 2, 3$.

- In the lasso, the tuning parameter is the regularization parameter, λ. The candidates for the regularization parameter λ are set as follows: maximum and minimum values of λ are defined by $\lambda_{\max} = 200$ and $\lambda_{\min} = 0.01$, respectively, and 100 grids on a log scale are constructed.

- For XGBoost and LightGBM, we change two tuning parameters: number of boosting iterations, $n_{iterations}$, and learning rate, LR. We set the candidates for these parameters as $n_{iterations} = 50, 100, 500$, and LR $= 0.01, 0.05, 0.1, 0.2, 0.5, 1$. Other parameters are set to be in default; for details, see `https://github.com/dmlc/xgboost/tree/master/R-package` and `https://github.com/microsoft/LightGBM/tree/master/R-package`.

- For ARIMA (T, d, q), T is fixed as $T = 4$, as in other methods. As candidates for d and q, we use $d = 0, 1, 2$ and $q = 0, 1, 2$.

Some of the prediction methods require normalization of inputs. For lasso and SVM, we normalize the input values because these methods are expected to normalize inputs. The ensemble learning based on trees, random forest, XGBoost, and LightGBM is not normalized because these methods are scale-invariant. For ARIMA, we employ the maximum likelihood (ML) estimation, which is not scale invariant. Although normalization appears needed, there is no theoretical justification for normalizing inputs in the ARIMA model; indeed, most software functions that implement ARIMA, such as the `arima` in `R` function or `statsmodels.tsa.arima_model.ARIMA` in `python`, do not provide the option for normalization. Therefore, we do not normalize the input values for ARIMA.