Detect, Map, Estimate: A Three-Step Framework for Historical Demand Response Analysis

Ryan Sharp
I.G.S. Energy
ryan.sharp@igs.com

Jack Van Dyke
I.G.S. Energy
jack.vandyke@igs.com

***Abstract*—A challenge of great interest to energy providers is to precisely detect demand response events and quantify load adjustments from historical data on building energy usage. We achieve a state-of-the-art F1 score and a competitive NMAE score at this task in three steps: (1) leverage a Daily Classifier to detect the days that contain a Demand Response event, (2) apply a Mapping Function to obtain 15-minute interval detections of the Demand Response Flag, and finally (3) estimate Demand Response Capacity at 15-minute intervals with our Power Regression Model. This generalizable, context-aware framework enables energy providers to confidently back-cast on their diverse historical data to measure how much load was adjusted by the buildings, facilitating the assessment of demand response behavior.**

1. Introduction

The problem statement is as follows, paraphrased from the FlexTrack Challenge 2025 overview webpage https://www.aicrowd.com/challenges/flextrack-challenge-2025.

A **Demand Response (DR) event** is when a building *increases*, *decreases*, or *shifts* its power consumption within a day to benefit the grid. Given a historical time-series data set of building energy usage, dry bulb temperature, and global horizontal radiation at 15-minute intervals, we are to back-cast on the data to:

- Detect the time periods in which DR events occurred.
- Quantify the energy that was flexed by the building for each 15-minute interval within such events.

To achieve the first bullet point, each 15-minute interval must be classified as one of three **Demand Response Flags (DRFs)**: 0 denotes no demand response, while -1 and $+1$ denote a requested decrease or increase in the building's load, respectively. This subtask is evaluated using F1 and Geometric Mean scoring.

To achieve the second bullet point, we estimate the **Demand Response Capacity** in kilowatts which can be either negative or positive. This is evaluated using Normalized Mean Absolute Error (NMAE) and Normalized Root Mean Squared Error (NRMSE) as defined on the competition website.

We note that our models are **non-causal**, meaning that when the model predicts for time t_0, it is allowed to "look ahead" and use data from time $t_1 > t_0$ to influence its prediction. This is permitted because the entire data set is considered historical to simulate the real-world scenario in which a building's historical DR events are to be assessed.

We propose two novel machine learning models to tackle this problem: (1) the Daily Classifier that detects days with Demand Response and (2) the Power Regression Model that indirectly calculates Demand Response Capacity by estimating the expected baseline power at 15-minute intervals. Our code is made publicly available at this GitHub repo.

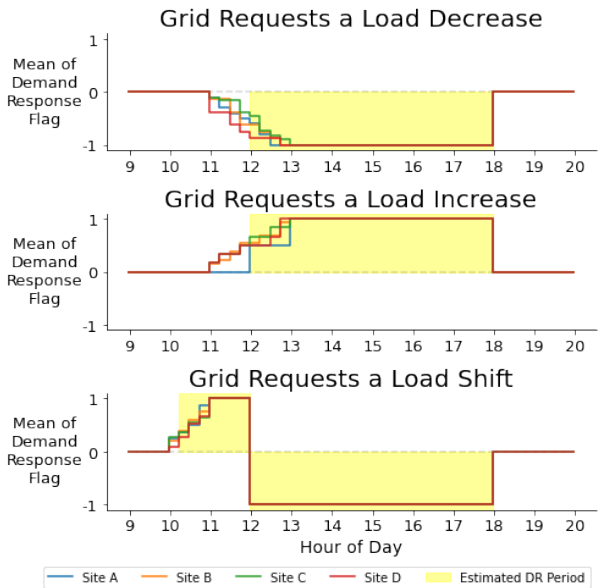

Figure 1. DRFs follow an intra-day pattern based on the day's DR event type (*decrease*, *increase*, *shift*, or no DR event). We achieve the highest F1-score in the competition by following two steps: (1) detect which DR event type has occurred (if any) for each day, and (2) have our Mapping Function output a DRF of -1 or $+1$ within the shaded yellow regions as described in Section 2.2.2. The curves in the figure are the mean of the true DRFs from the training data set grouped by DR event type, site, and time of day. No grouping contains both -1 and $+1$ DRFs.

2. Methods

Algorithm 1 provides a high-level outline of our entire machine learning pipeline. In later sections we cover all the specifics such as the feature sets, mapping function, post processing, etc.

Algorithm 1 High-level outline of our solution. f_C is the Daily Classifier. f_R is the Power Regression Model. d is a given day and t is a given 15-minute interval. $\hat{y}_d$ is the DR Event classification for day d, while $\hat{\mathbf{y}}_d = [\hat{y}_{d,1}, \hat{y}_{d,2}, \ldots, \hat{y}_{d,96}]$ is an array of DRF classifications for day d. These are further explained in Section 2.2.1.

Require: Training data D_{train} and test data D_{test} from the competition, and set of test sites S_{test}

$D'_{\text{train}} \leftarrow$ feature_engineering(D_{train})
$D'_{\text{test}} \leftarrow$ feature_engineering(D_{test})
Train f_C using D'_{train}
$\hat{y}_d \leftarrow f_C(D'_{\text{test}})$
$\hat{\mathbf{y}}_d \leftarrow$ mapping_function($\hat{y}_d$) // Map Daily to Interval
Partition D'_{test} by site into subsets $\{D'^{(s)}_{\text{test}} : s \in S_{\text{test}}\}$
for each partition $D'^{(s)}_{\text{test}}$ **do**
    Train $f_R^{(s)}$ on $D'^{(s)}_{\text{test}}$ filtered to $\hat{y}_d^{(s)} = 0$
    $\hat{Y}_t^{\text{Pow}(s)} \leftarrow f_R^{(s)}\left(D'^{(s)}_{\text{test}}\right)$
    $\hat{Y}_t^{\text{Cap}(s)} \leftarrow \begin{cases} 0, & \text{if } \hat{y}_{d,t}^{(s)} = 0 \\ Y_t^{\text{Pow}(s)} - \hat{Y}_t^{\text{Pow}(s)}, & \text{otherwise} \end{cases}$
end for

2.1. Data Preparation

Our models only process the features corresponding to a timestamp within the time 10:00 to 17:59 of each day because we observe that demand response events occur only during this time window. For a similar reason, our solution does not predict any demand response for the months of March, April, May, September, October, and November. If the grid were to decide that the future may call for demand response events at new times of day or for new months, then our code can easily be modified to allow for predicting during those new times and dates.

We do *not* exclude holidays or extremely high/low values of power from our training and testing sets because we noticed negligible differences by doing so.

2.2. Daily Classifier Model

2.2.1. Model Design. The end goal of this model is to classify the Demand Response Flag (DRF) for every timestamp. Although the timestamps are formatted in *15-minute intervals*, we design our classifier model to predict one value *per day*, which we call the **DR Event Type**. This value denotes whether on this day a *decrease*, *increase*, or *shift* in load is requested by the grid, or if *no* DR is requested. This prediction is subsequently mapped to an array of 15-minute interval predictions using our **Mapping Function**. Formally, let the set of possible 15-minute DRFs be

$$\mathcal{Y} = \{-1, 0, 1\},$$

as defined in the competition overview. Let the sequence of DRFs for day d be

$$\mathbf{y}_d = [y_{d,1}, y_{d,2}, \ldots, y_{d,96}], \quad y_{d,t} \in \mathcal{Y},$$

where each element corresponds to a 15-minute interval.

The set of unique DRFs for day d is defined as

$$U_d = \text{unique}(\mathbf{y}_d) \subseteq \mathcal{Y}.$$

We define the corresponding **DR Event Type** as

$$y_d = \begin{cases} -1, & \text{if } U_d = \{0, -1\} \text{ (a } \textit{decrease} \text{ in load)}, \\ 0, & \text{if } U_d = \{0\} \text{ (no demand response)}, \\ 1, & \text{if } U_d = \{0, 1\} \text{ (an } \textit{increase} \text{ in load)}, \\ 2, & \text{if } U_d = \{-1, 0, 1\} \text{ (a } \textit{shift} \text{ in load)}. \end{cases}$$

Finally, let f_C denote the Daily Classifier model that maps an input feature vector $\mathbf{x}_d$ (representing the features corresponding to day d) to a discrete prediction:

$$f_C(\mathbf{x}_d) = \hat{y}_d \in \{-1, 0, 1, 2\}.$$

Our algorithm of choice for f_C is the Extreme Gradient Boosting (XGBoost) classifier [1]. Since XGBoost cannot directly predict negative class labels, it predicts values in the set $\{0, 1, 2, 3\}$, from which we subtract 1 to obtain the desired label range.

2.2.2. Mapping Function. After f_C predicts the DR Event Type $\hat{y}_d$ for each day of the test set, we map these daily classifications to $\hat{\mathbf{y}}_d = [\hat{y}_{d,1}, \hat{y}_{d,2}, \ldots, \hat{y}_{d,96}]$, an array of 15-minute interval DRF classifications for an entire day. The mapping works as follows:

- $\hat{y}_d = -1$: The day receives a -1 DRF prediction from 12:00 to 17:59, and 0 at all other times.
- $\hat{y}_d = 0$: The day receives a 0 DRF prediction for the entire day.
- $\hat{y}_d = 1$: The day receives a 1 DRF prediction from 12:00 to 17:59, and 0 at all other times.
- $\hat{y}_d = 2$: The day receives a 1 DRF prediction from 10:15 to 11:59, a -1 from 12:00 to 17:59, and 0 at all other times.

This mapping is visualized by the yellow shaded regions in Figure 1.

2.2.3. Organization of the Training Data. We train our Daily Classifier using all sites containing labeled DRFs (i.e. sites A, B, C and D). After doing so, the Daily Classifier is ready to receive input features to predict for any site. We perform array aggregation of the 15-minute interval features within each day before passing that array to the Daily Classifier as input. Within this array, the order of the features does not matter as long as it remains fixed. We oversample the minority classes (DR Event Type values -1, 1 and 2) using SMOTE [2] with random seed 94.

2.2.4. Features.

- **Classifier Features:**
  - temp_corr_dev, power_zscore_sh, power_zscore_sh_diff_t, power_zscore_sh_diff_wdt, power_zscore_sh_peek_diff, power_zscore_sh_diff, power_zscore_sh_peek_diff_t, power_zscore_sh_hourly_std, power_share_zscore_sh, power_share_zscore_sh_diff, power_share_zscore_sh_diff_t, power_share_zscore_sh_diff_wdt, power_share_zscore_sh_peek_diff, power_share_zscore_st_hourly_std, power_zscore_sh_peek4_diff, power_zscore_sh_lag4_diff, power_share_zscore_sh_peek4_diff, power_share_zscore_sh_lag4_diff, power_share_zscore_st_peek4_diff_t, season, month
- **Feature Partitions**
  - t: time
  - h: hour
  - wd: day of week
  - m: month
  - s: season
- **Feature Explanations**
  - Power Share: The power of that row divided by the sum of power within a day's working hours (10:00-17:45).
  - Corr: Weather variable's correlation to power for the site
  - Corr Dev: Weather variable's correlation to power multiplied by the deviation in weather variable from site's median for monthly or seasonal partition
  - Diff: The difference in a feature from the previous value of that partition.

2.2.5. Intuition of Model Design. The benefits of predicting Demand Response Flags using this *Daily Classifier* rather than a *15-minute interval classifier* are:

- **Wider context is visible to the model.** At the 15-minute interval level of granularity, features can be noisy and misleading, as the broader picture is not visible. For example, it is possible for the DRF to be -1 while Demand Response Capacity is actually positive (this occurs in 263 timestamps in the train data set of sites A, B and C). "Zooming out" to the entire day provides greater context for the model, allowing it to consider the "shape" of the load throughout the day - reducing the likelihood that those noisy timestamps will be misclassified.
- **Facilitation of creating an "intra-day shape" of DRF predictions that resembles that of the true flags.** Figure 1 shows that when a DR event is requested by the grid, the event lasts until 17:59 and its type (increase, decrease, or shift) does not change or cancel within a day. For this reason, it is appropriate for our Mapping Function to produce such static patterns or "shapes" of DRF predictions within each day. Although, the exact beginning time of the event may still be misclassified – this is an area for potential further improvement.

2.3. Power Regression Model

2.3.1. Brief Overview. Now that the test set has been classified into periods *with* and *without* demand response, we turn to estimating the Demand Response Capacity within those periods of demand response. We leverage our Power Regression Model f_R to compute $\hat{Y}_t^{\text{Power}}$, the expected building power for time t (at 15-minute granularity) were no demand response event requested.

$$f_R(x_t) = \hat{Y}_t^{\text{Power}}$$

Thus, the estimated Demand Response Capacity is the true building power minus expected building power:

$$\hat{Y}_{\text{t}}^{\text{Capacity}} = Y_t^{\text{Power}} - \hat{Y}_t^{\text{Power}}$$

f_R is an ensemble of three regression models, each with their own unique set of features. We partition the test set by site and train and run inference with this model within those partitions.

2.3.2. Organization of the Training and Testing Data. For site s, we train f_R on the data corresponding to site s on the days where $\hat{y}_d = 0$. We subsequently predict on the data for site s on the days where $\hat{y}_d \neq 0$. Thus, $\hat{Y}_t^{\text{Capacity}(s)}$ remains zero on days with no detected Demand Response event.

2.3.3. Ensemble. We employ a weighted ensemble of XGBoost models, each trained with distinct feature subsets and hyperparameter settings, where a dominant model provides a robust global baseline and secondary models act as targeted specialists that correct its systematic errors. This design leverages hypothesis diversity and complementary feature signals (temperature-based, irradiance-based, temporal lags, etc.), producing less correlated errors and improved handling of heteroskedasticity and regime shifts across sites and hours. By combining models that prioritize different loss characteristics and inductive biases, the ensemble preserves the low bias of the best performer while using corrective terms to reduce residual bias and variance, producing more calibrated, generalizable, and interpretable capacity estimations than any single model. The hyperparameters were tuned with Optuna [3]. Feature selection was performed per model using custom code.

TABLE 1. MODEL PERFORMANCE METRICS ON THE PRIVATE TEST SET, ORDERED BY NMAE ASCENDING.

Rank	Team or Participant	Normalized MAE	Normalized RMSE	Geometric Mean Score	F1 Score
1	flex_king	**0.698**	1.104	0.651	0.624
2	zch	*0.706*	1.107	0.651	0.629
3	WollongongOrBust	0.731	*1.086*	*0.749*	**0.720**
4	ningjia	0.779	**1.052**	0.743	*0.702*
5	DTU	0.889	1.233	0.637	0.581
6	improvers	0.890	1.175	0.578	0.523
7	pluto	0.915	1.229	0.711	0.629
8	liberifatali	0.940	1.285	0.563	0.487
9	Phaedrus	0.989	1.359	**0.819**	0.457
10	danglchris	0.991	1.223	0.618	0.532

2.3.4. Features.

- **Time Features:**
  - season, month, week, day_of_week, hour, quarter_hour
- **Weather Features:**
  - temp, temp_lag, temp_lag2, temp_lag3, temp_lag4, temp_median_mt, temp_peek, temp_peek2, temp_power_corr_st, temp_pull2, temp_pull3, temp_shift, temp_shift3, temp_shift4, irr, irr_lag, irr_lag2, irr_lag3, irr_lag4, irr_median_mt, irr_median_st, irr_peek, irr_peek2, irr_peek3, irr_peek4, irr_power_corr_st, irr_power_corr_st_pab, irr_pull, irr_pull2, irr_pull3, irr_pull4, irr_shift, irr_shift2, irr_shift3, irr_shift4
- **Power Features:**
  - baseline_pow, li_feature, mean_usage_sdt, mean_usage_sdt_corr_dev_st, usage_lag, usage_lag_dow, usage_lag_dow2, usage_peek, usage_peek2, usage_peek_dow
- **Feature Partitions**
  - t: time
  - d: day of week
  - dow: day of week
  - m: month
  - s: season
  - pab: power above baseline
- **Feature Explanations**
  - li_feature: This is the value from time t of the linear interpolation of building power between time 9:45 and 18:15 of the day.
  - Lag: Past values partitioned by time
  - Peek: Future values partitioned by time
  - Shift: Past values of current date
  - Pull: Future values of current date
  - Corr: Weather variable's correlation to power for the site
  - Corr Dev: Weather variable's correlation to power multiplied by the deviation in weather variable from site's median for monthly or seasonal partition

2.3.5. Post Processing. We apply simple, conservative post-processing to the raw capacity estimations to ensure operational validity: they are clipped to site-specific bounds derived from observed maximum power in the training data (limiting capacity to realistic percentages of each site's max), and rows with a $+1$ DRF classification are additionally constrained so that they cannot produce negative capacity values. These safeguards prevent implausible or unsafe outputs, preserve consistency with historical behavior, and reduce downstream risk from extreme or out-of-distribution model errors.

3. Results and Discussion

3.1. Performance Metrics

The most notable achievement of our solution is that it obtained the **highest F1 score** on the private test set out of any team. This indicates that our classification pipeline (f_C and Mapping Function) is the **state-of-the-art** at detecting historical demand response events, because it achieves the best precision-recall trade-off across all three classes of DRFs.

Furthermore, our solution achieved the **second-highest Geometric Mean score**, the **second-lowest NRMSE**, and the **third-lowest NMAE** on the private test set out of any team. The NRMSE and NMAE scores prove that our solution is highly competitive at estimating the Demand Response Capacity.

A complete table of these metrics for the top ten teams is shown in Table 1.

3.2. Computational Requirements

We used an AMD Ryzen 5 2600 processor and no GPU for training and inference. The time and memory required by our models to produce the third-place submission is listed in Table 2 below. **Our Daily Classifier f_C takes 0.01 seconds to perform inference on the entire private testing set which contains 289,440 rows of data,** or 3,015 days.

TABLE 2. TIME AND MEMORY REQUIRED BY MODELS f_C AND f_R TO PRODUCE OUR THIRD-PLACE SOLUTION.

Model	Phase	Time (seconds)	Max memory (MiB)
f_C	Training	21	1,070
	Inference	0.01	1,110
f_R	Training	70	2,030
	Inference	0.5	1,900

3.3. Limitations

- As our models are non-causal, they cannot directly be used to make real-time predictions for the current moment. Instead, they would have to wait until the current day ends before they can predict for the day. Thus, a potential next step for our solution is to adapt the code to allow real-time predictions.
- The predicted DRFs from the Mapping Function may be *incorrect* during the first 1 or 2 hours of the DR event *even if* our Daily Classifier *correctly* classifies the DR event type. This can be seen in Figure 1. For example, when the grid requests a load decrease, the DRF can switch from 0 to -1 at any point between 11:00 and 13:00, while our Mapping Function estimates that the switch occurs at 12:00. Although we did not have success at using machine learning to estimate the exact "start time" of the DR event, this may be a potential next step for improvement.
- The leaderboard in Table 1 suggests that our Power Regression Model f_R has further room for improvement. We look forward to learning from the other winning teams to potentially improve our solution for the benefit of many.

4. Conclusion

Our solution advances the community's ability to detect historical Demand Response events, as seen by our state-of-the-art F1 score. Furthermore, our competitive regression scores suggest that we may have discovered potential improvements for the estimation of Demand Response Capacity. Moving forward, we hope to (1) adapt this solution for a causal, real-time scenario and (2) potentially collaborate with other winning teams to make further performance improvements. We are grateful for the organizers and sponsors of the Flextrack Challenge 2025 for fostering innovation that will benefit future energy grids.

References

[1] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, p. 785–794, ACM, Aug. 2016.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, p. 321–357, June 2002.

[3] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," 2019.