A Unified CatBoost Framework for IoT Data Classification Using Multi-Level Feature Extraction

Chengfeng Qiu
Netease Yidun AI Lab
Hangzhou, Zhejiang, China
qiuchengf_123@163.com

Jiahui Zhou
Sun Yat-Sen University
Zhuhai, Guangdong, China
zhoujh99@mail2.sysu.edu.cn

Yongfeng Liao
University of Electronic Science and Technology of China
Chengdu, Sichuan, China
liaoyf_hz@163.com

Zhengliang Cui
Southwest Jiaotong University
Chengdu, Sichuan, China
czlbou2012@gmail.com

Dan Li
Sun Yat-Sen University
Zhuhai, Guangdong, China
lidan263@mail.sysu.edu.cn

ABSTRACT

Buildings are major energy consumers, and efficient management is crucial for reducing emissions, yet the lack of standardised data formats complicates this process. This paper presents a method for automating the classification of time-series building data, using a segmentation approach to enhance feature extraction and applying hierarchical differential features to capture temporal dynamics. The multi-label classification problem is reformulated into a unified single-label classification task by encoding multiple labels into a discrete 91-class label space. The model, trained with CatBoost and evaluated using 5-fold cross-validation, secured a top ranking in the Brick by Brick 2024 competition, demonstrating its effectiveness in promoting energy-efficient building operations.

CCS CONCEPTS

• **Computing methodologies** → **Classification and regression trees**; **Cross-validation**.

KEYWORDS

Time-Series Building Data; Segmentation Approach; Hierarchical Differential Features; 5-Fold Cross-Validation

ACM Reference Format:
Chengfeng Qiu, Jiahui Zhou, Yongfeng Liao, Zhengliang Cui, and Dan Li. 2025. A Unified CatBoost Framework for IoT Data Classification Using Multi-Level Feature Extraction. In *Companion Proceedings of the ACM Web Conference 2025 (WWW Companion '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3701716.3718479

1 INTRODUCTION

Buildings are among the largest energy consumers globally, contributing significantly to both energy use and emissions. Efficient building management is essential for mitigating the impacts of climate change, yet the lack of standardisation in building system data formats has made this a complex and costly task [2, 3]. The Brick by Brick 2024 competition, hosted by AICrowd, addresses this issue by challenging participants to automate the classification of building data, with the aim of promoting standardised, energy-efficient management solutions for sustainable building operations https://www.aicrowd.com/challenges/brick-by-brick-2024.

The competition focuses on classifying time-series data from IoT devices in buildings according to the Brick schema. The dataset consists of time-series data from three anonymised buildings in Australia, containing signals like temperature readings, setpoints, and alarms, with irregular sampling rates. Participants are required to develop algorithms that can classify this data, contributing to the broader goal of automating building management and advancing the scalability of smart building technologies [4].

2 METHODS

2.1 Data Preprocessing

The preprocessing pipeline is designed to address key challenges in the dataset while preserving its temporal characteristics and enhancing its utility for model training. Initial analysis revealed that the time span of individual samples predominantly clustered around **56**, **28**, and **14** days, with a long-tail distribution of shorter durations (less than a week). Using the full time span of each sample as raw input for feature extraction proved suboptimal, as long sequences often led to poor feature representation and limited sample diversity [7]. To mitigate these issues, we adopt a segmentation strategy, dividing each sample into daily time windows. This approach not only increases the number of training instances but also improves the granularity and robustness of the extracted features, aligning with best practices in time-series analysis.

The label processing pipeline is designed to handle the multi-label nature of the dataset, where each sample is associated with 94 distinct labels, belonging to a three-class classification problem. To streamline the label representation and facilitate model training, we first concatenate the 94 labels into a single string format, denoted as $[y1, y2, y3, ..., y94]$. This concatenated string is then encoded into a single integer value ranging from 0 to 90 using a label encoding scheme, effectively transforming the multi-label classification problem into a single-label 91-class classification task. During inference,

WWW Companion '25, April 28-May 2, 2025, Sydney, NSW, Australia

ACM ISBN 979-8-4007-1331-6/2025/04
https://doi.org/10.1145/3701716.3718479

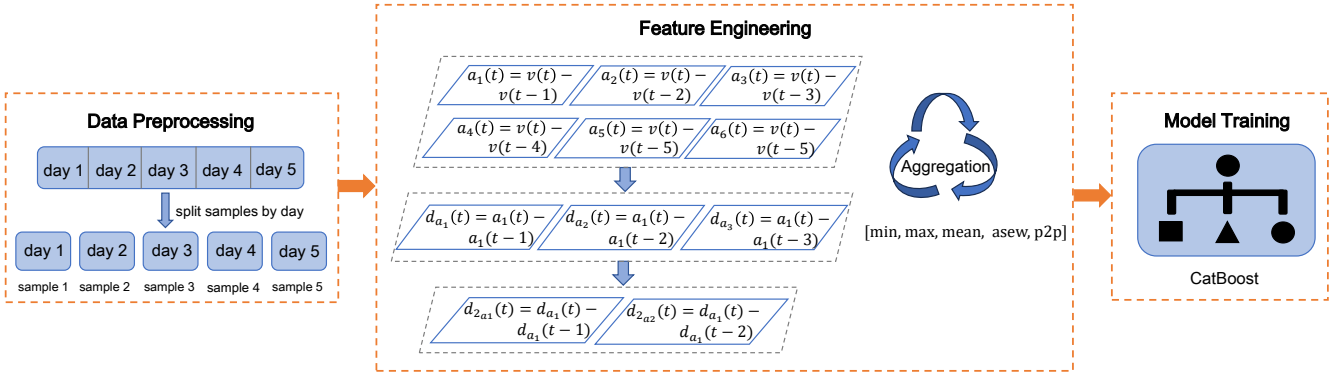

Figure 1: The architecture for classification task.

Table 1: Model Hyperparameters

Hyperparameter	Description	Value
`learning_rate`	Controls the step size during gradient descent.	0.3
`l2_leaf_reg`	L2 regularization term on weights to prevent overfitting.	10
`od_type`	Type of overfitting detector.	`Iter`
`od_wait`	Number of iterations to wait after the best iteration before stopping.	70
`bootstrap_type`	Method for sampling weights during training.	`Bernoulli`
`random_seed`	Seed for reproducibility of results.	11251
`depth`	Maximum depth of the decision trees.	5
`task_type`	Specifies the hardware for training (CPU or GPU).	`GPU`
`loss_function`	Objective function for multi-class classification.	`MultiClassOneVsAll`

the predicted class is decoded back into the original 94 labels, which are normalized to the range $[0, 1]$ for model compatibility. To handle data augmentation, where each ID corresponds to multiple samples, predictions are aggregated by computing a weighted average of the maximum and mean probabilities across augmented instances, ensuring robust and balanced final predictions.

2.2 Feature Engineering

We employ a hierarchical approach to effectively capture temporal dynamics and patterns from the time-series data. The goal of this method is to generate robust features that are sensitive to the underlying changes in the building systems, while also maintaining computational efficiency. The architecture is illustrated in figure 1. The process begins with the raw velocity feature (v), from which we generate first-order differential features through a sliding window mechanism. Specifically, we compute the differences between the current velocity and its shifted values across 1 to 6 time windows, denoted as a_n, where n represents the window index. This operation captures the immediate changes in velocity, which are crucial for understanding the dynamic behavior of the system.

Building upon the first-order features, we further derive second-order differential features d_{a_n} by applying the same sliding window technique with window sizes ranging from 1 to 3. This second-level transformation enables us to capture the acceleration patterns [1], providing insights into how the rate of change itself is evolving over time. To enhance the temporal resolution, we extend this approach to third-order differential features d_{2a_n} using smaller window sizes (1 to 2), which helps in identifying subtle variations in the acceleration patterns.

The final feature set is constructed by applying a comprehensive set of statistical aggregators $[min, max, std, mean, skew, p2p]$ to the three levels of differential features (a_n, d_{a_n}, d_{2a_n}) along with the time difference (d_t)metadata. This multi-level feature extraction strategy is particularly effective because it allows us to capture both short-term fluctuations and long-term trends in the velocity data [6], while the statistical aggregators provide a robust representation of the underlying patterns.

2.3 Training Details

We use CatBoost [5], a gradient boosting algorithm specifically designed to handle categorical features effectively. CatBoost is built on the principles of gradient boosting decision trees (GBDT) and is known for its ability to deliver robust performance even with large, heterogeneous datasets. It excels in handling categorical features by automatically encoding them, which is particularly beneficial for time-series data like ours, where categorical variables such as device types and sensor statuses are prevalent.

To ensure robust model performance and generalizability, we adopted a 5-fold cross-validation strategy. The model was trained on a system equipped with an NVIDIA 2080 Ti GPU with 10 GB of VRAM, For inference, we utilized a machine with a 24 GB CPU to handle the large-scale predictions. The hyperparameters of our

Table 2: Testing Results in Public Leaderboard Set.

Method	Accuracy	Precision	Recall	**F1-score**	mAP
Zero	0.9769	0.0000	0.0000	0.0000	0.0000
Random Uniform	0.5005	0.0231	0.4993	0.0356	0.0233
Random Proportional	0.9563	0.0239	0.0224	0.0097	0.0234
Mode	0.9664	0.0000	0.0106	0.0001	0.0000
One	0.0231	0.0231	1.0000	0.0396	0.0231
Our Method (*partial features*)	0.989	0.681	0.602	0.604	0.557
Our Method	**0.9895**	**0.6921**	**0.6146**	**0.6160**	**0.5738**

Table 3: The performance in training set

Method	Micro F1	Accuracy	Precision	Recall	mAP
Our Method (*partial features*)	0.8009	0.9878	0.8529	0.7984	0.7497
Our Method	0.8007	0.7524	0.8534	0.8030	0.9874

model are summarized in Table 1. The code is publicly available in https://github.com/Qiucf-king/www2025_chanllenge_yddm.

3 RESULTS

The experimental results on the public leaderboard set, as presented in Table 2, demonstrate the performance of various baseline models provided by the competition organizers and our proposed method. The primary evaluation metric is the macro F1-score, which provides a balanced measure of precision and recall.

Baseline models yield poor performance in Table 2, with F1-scores ranging from 0 to 0.0396. In contrast, our proposed method demonstrates significant improvements across all metrics. The Our Method (partial features) version, which utilizes only 6 engineered features, achieves an F1-score of 0.604. The final version of Our Method, which extends the feature set to 11 features, further boosts performance to an F1-score of 0.6160, along with superior accuracy, precision, recall, and mAP. This success stems from effective data preprocessing, which enhances sample augmentation and data utilization, and advanced feature engineering, where differential and derived features are constructed to capture high-expression patterns.

Besides, we evaluate our models on the training set to provide a more comprehensive analysis. Table 3 presents the performance of our two models. The results show that both models perform well, with the model (*partial features*) achieving a Micro F1 of 0.8009, accuracy of 0.9878, and recall of 0.7984. The full feature set model, while slightly lower in Micro F1, exhibits improved recall (0.8030) and a notably higher mAP of 0.9874, indicating a better fit for the training data.

The training time for Our Method with partial features is approximately 40 minutes, while the full feature model takes about 85 minutes. For inference, the partial features model requires around 20 minutes, and the full feature model takes about 40 minutes. Furthermore, in accordance with the competition requirements, Figure 2 provides a visual representation of our submission results, offering further insight into the model's performance across different scenarios.

4 CONCLUSION

This paper presents a novel framework for automating the classification of time-series building data to promote energy-efficient management in buildings. By leveraging a hierarchical feature extraction approach, we effectively capture the temporal dynamics of building systems, enhancing both the accuracy and efficiency of classification. The transformation of the multi-label classification problem into a unified single-label task, through encoding labels into a discrete 91-class space, further streamlines the process. The use of CatBoost, proved to be particularly effective in managing the complexity of time-series data. Our model achieved outstanding results, securing top place in the Brick by Brick 2024 competition, underscoring the potential of this approach in real-world applications. This work not only demonstrates the feasibility of automating building data classification but also offers a significant contribution toward standardizing data formats for energy-efficient operations in buildings.

REFERENCES

[1] Marília Barandas, Duarte Folgado, Letícia Fernandes, Sara Santos, Mariana Abreu, Patrícia Bota, Hui Liu, Tanja Schultz, and Hugo Gamboa. 2020. TSFEL: Time series feature extraction library. *SoftwareX* 11 (2020), 100456.

[2] Dan Li, Yuxun Zhou, Guoqiang Hu, and Costas J Spanos. 2018. Identifying unseen faults for smart buildings by incorporating expert knowledge with data. *IEEE Transactions on Automation Science and Engineering* 16, 3 (2018), 1412–1425.

[3] Dan Li, Yuxun Zhou, Guoqiang Hu, and Costas J Spanos. 2019. Handling incomplete sensor measurements in fault detection and diagnosis for building HVAC systems. *IEEE Transactions on Automation Science and Engineering* 17, 2 (2019), 833–846.

[4] Arian Prabowo, Xiachong Lin, Imran Razzak, Hao Xue, Emily W Yap, Matthew Amos, and Flora D Salim. 2024. BTS: Building Timeseries Dataset: Empowering Large-Scale Building Analytics. *arXiv preprint arXiv:2406.08990* (2024).

[5] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: unbiased boosting with categorical features. *Advances in neural information processing systems* 31 (2018).

[6] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and Jun Zhou. 2024. Timemixer: Decomposable multiscale mixing for time series forecasting. *arXiv preprint arXiv:2405.14616* (2024).

[7] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2023. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In *International Conference on Learning Representations.*

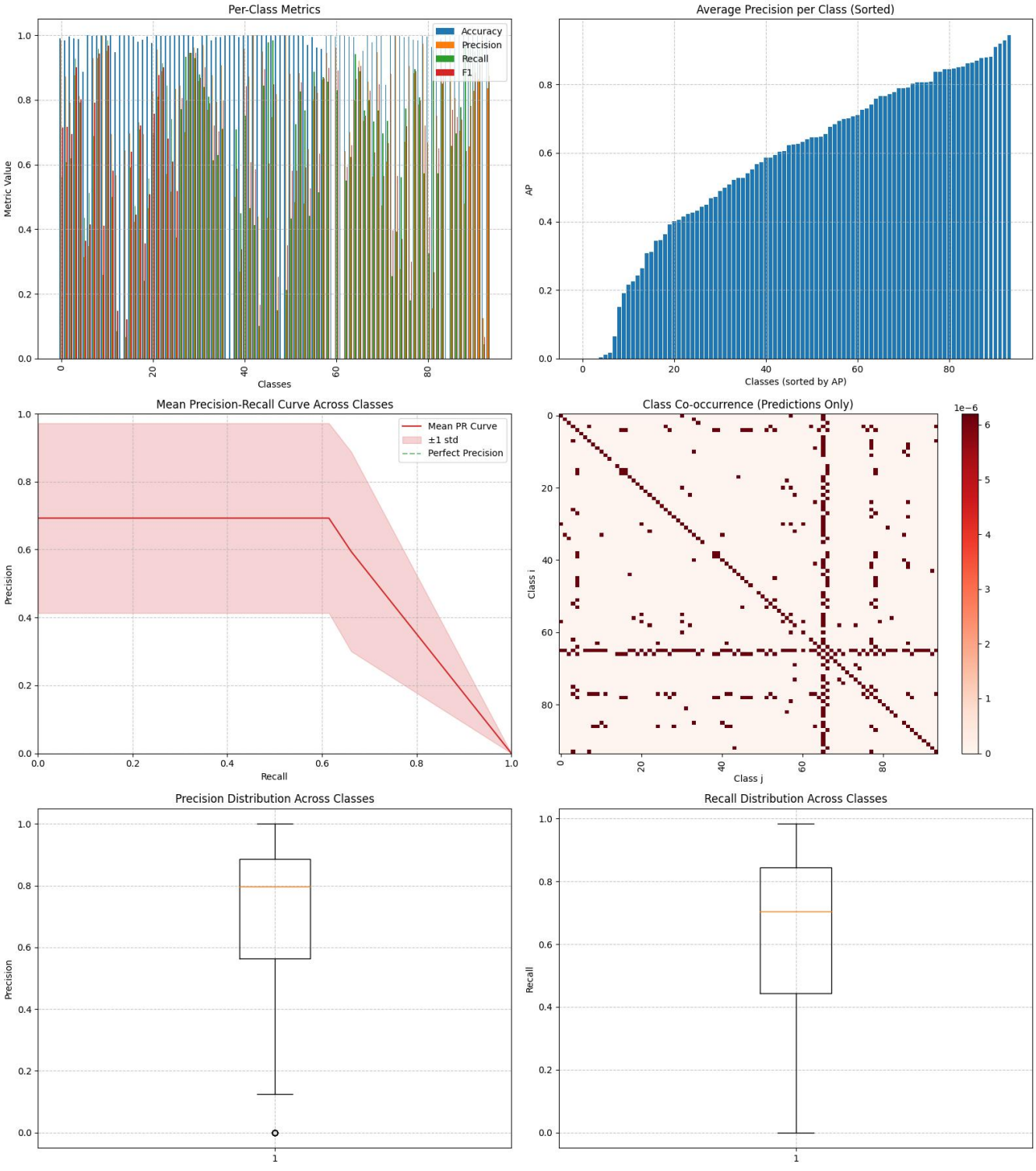

Figure 2: The visualization of the submission.