A Simple Feature Extraction based Approach to Solving the Brick by Brick 2024 Challenge

Kaushik Gopalan
FLAME University
Pune, Maharashtra, India
kaushik.gopalan@flame.edu.in

Prithvi Dhyani
FLAME University
Pune, Maharashtra, India
prithvi.dhyani@flame.edu.in

Arunima Srikant
FLAME University
Pune, Maharashtra, India
arunima.srikant@flame.edu.in

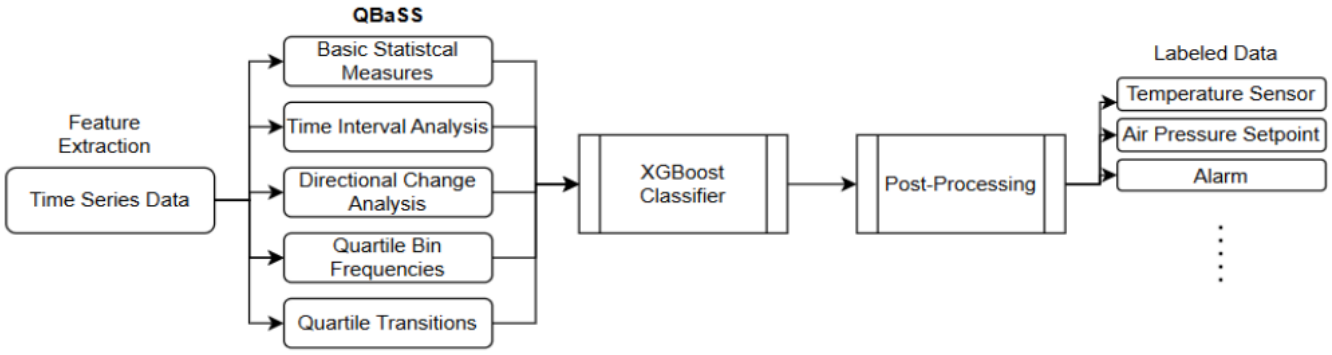


Figure 1: Illustration of QBaSS Classification Pipeline

Abstract

We present a machine-learning-based approach to efficiently classify IoT device data in the AIcrowd | Brick by Brick 2024 competition, enabling automation in smart building management. The increasing prevalence of IoT devices in modern buildings has led to an increase in time-series data from diverse sources. Efficiently classifying and organizing this data according to an ontology is crucial for applications such as energy optimization, fault detection, and predictive maintenance. However, manual classification is costly and time-consuming. To address this, we propose a Quartile-Based Shape Segmentation (QBaSS) method, which extracts lower-dimensional time-series representations for stratified classification using gradient boosting. Our results demonstrate reliable classification for most device types, achieving a Macro F1 score of 0.509 with the test data.

CCS Concepts

• **Computing methodologies** → **Boosting**; **Machine learning**; **Machine learning algorithms**; **Ensemble methods**;

Keywords

Time-series classification; XGBoost; Smart building Sensor classification

ACM Reference Format:
Kaushik Gopalan, Prithvi Dhyani, and Arunima Srikant. 2025. A Simple Feature Extraction based Approach to Solving the Brick by Brick 2024 Challenge . 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, 5 pages. https://doi.org/10.1145/3701716.3719637

1 Introduction

The 21st century has seen rapid growth in the adoption of centralized Building Management Systems(BMS). This shift has been made possible by the integration of IoT devices such as sensors and alarms for HVAC, electrical and other systems. Buildings account for roughly 40% of global energy consumption [4], hence the need to optimize their energy efficiency has further motivated work in this area. Leveraging machine learning techniques to structure the raw IoT data in real time enables automated control loops [1, 6], wherein sensor data is used to dynamically adjust building operations, which results in reduced response times and less energy wastage.

This paper documents our solution for the AIcrowd | Brick by Brick 2024 competition: https://www.aicrowd.com/challenges/brick-by-brick-2024. The Brick by Brick challenge 2024 involves classifying time-series data from IoT devices following the Brick schema version 1.2.1. Using time series data, the goal is to identify which combinations of 94 given device tags apply to a given unlabeled time series. The dataset [8] consists of variable length time series with a training set of 30,000 time series and a public test set of around 10 times that number. Each time series corresponds

WWW Companion '25, Sydney, NSW, Australia.

ACM ISBN 979-8-4007-1331-6/25/04
https://doi.org/10.1145/3701716.3719637

to measurements taken over a 2 week - 8 week period, with variable sampling intervals. We describe our approach to solving this problem in the next section.

2 Methods

In recent years, deep-learning approaches to time-series classification have become prevalent, for example using MVCNNs(Multivariate CNNs) [5] to unify feature extraction and classification in a single architectural framework, while incorporating inter-temporal properties. Additionally, shape-based methods which leverage extraction of wavelets, fundamental frequencies, DTW, etc. have also been explored [3]. The majority of these methods however, either assume regular sampling rates or parametric forms. These assumptions are violated in data with low signal-to-noise-ratio such as IoT data, which often exhibits non-uniform sampling frequencies.

Taking these constraints into consideration, we characterize the task of classifying the device labels as a "time-series matching" problem: i.e. for any given test time-series for which the label is to be inferred we set out to detect which time-series in the training set are most similar both in terms of the vertical range as well as the shape of the time-series. For example, certain usage sensors and power sensors have a monotonically increasing value, certain types of temperature sensors have sinusoid like shapes whereas a rain sensor would have a rectangular shape toggling between 0 and 1.

To address this, we propose a comprehensive feature extraction framework that captures both the statistical properties and temporal dynamics of time series signals. Our approach combines basic statistical measures with more detailed analysis of temporal patterns and value distributions, presented in Table 1. The framework encompasses five distinct categories: basic statistical measures that capture the fundamental distribution of values, time interval analysis that examines the temporal spacing of measurements, directional change analysis that quantifies the frequency and direction of value changes, quartile bin frequencies that describe the distribution of values across the signal's range, and quartile transitions that characterize the pattern and frequency of value movements between different amplitude ranges. The features are calculated both on the raw data and on a filtered version (excluding values outside the 1st-99th percentile range) to provide robust characterization while retaining sensitivity to important signal characteristics.

The quartile-based shape segmentation (QBaSS) measures provide a crude idea of the shape of a given time series. A sinusoidal series would have uniform frequencies in all of the 4 quartiles and the transitions would normally occur between adjacent quartiles. On the other hand, a rectangular wave would have nearly all the values shared between Q1 and Q4, and the transitions would also be restricted between these 2 quartiles. Further, a higher frequency wave would have higher values in the transition features compared to a low frequency wave because the values would cross over between different quartiles more often. We demonstrate 3 examples of this feature extraction scheme in Figure 2. The first two time-series are roughly sinusoidal in nature, with the first having a higher frequency of oscillations compared to the second one. The final example consists of a rectangular time-series with the values toggling between 0 and 1. We can see that the quartile frequencies are roughly evenly distributed between the 4 quartiles for the sinusoidal time-series whereas the rectangular one has 0 frequencies for Q2 and Q3. Thus, we can make some inferences about the shape of the time-series based on the quartile frequencies. Further, we can

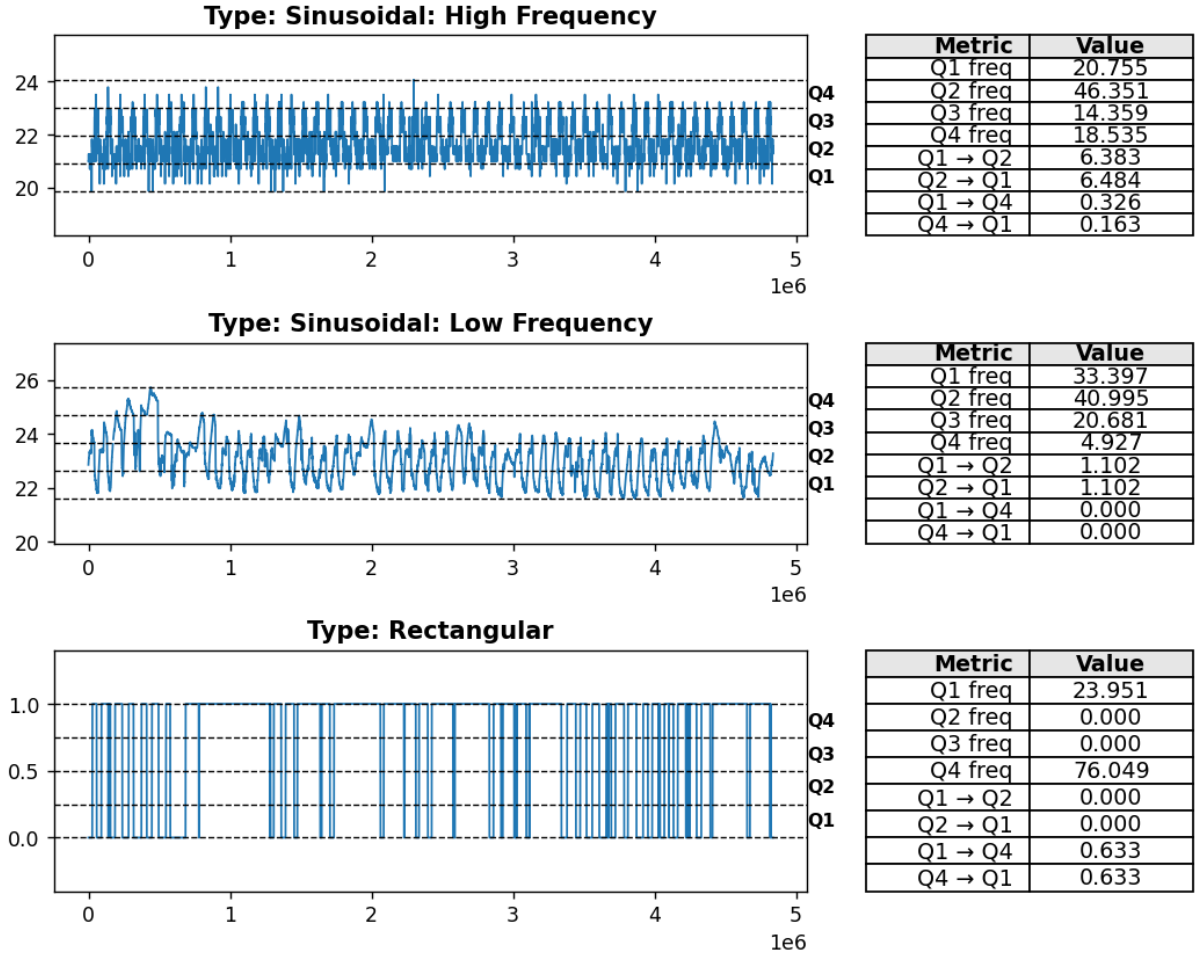
Feature Category	Features
Basic Statistical Measures(9 features)	• Minimum value • Maximum value • Mean value • Median value • Minimum non-zero value • 1st percentile • 99th percentile • Percentage of integer values
Time Interval Analysis(6 features)	• Minimum time difference • Mean time difference • Median time difference • 30,70,99th percentile of time differences
Directional Change Analysis(2 features)	• Number of positive changes per 100 samples • Number of negative changes per 100 samples
Quartile Bin Frequencies(8 features)	• Percentage in Q1 (original and filtered) • Percentage in Q2 (original and filtered) • Percentage in Q3 (original and filtered) • Percentage in Q4 (original and filtered)
Quartile Transitions(24 features)	• Q1→Q2 transitions per 100 samples • Q1→Q3 transitions per 100 samples • Q1→Q4 transitions per 100 samples • Q2→Q1 transitions per 100 samples • Q2→Q3 transitions per 100 samples • Q2→Q4 transitions per 100 samples • Q3→Q1 transitions per 100 samples • Q3→Q2 transitions per 100 samples • Q3→Q4 transitions per 100 samples • Q4→Q1 transitions per 100 samples • Q4→Q2 transitions per 100 samples • Q4→Q3 transitions per 100 samples
Total Features: 49	

Table 1: Time Series Feature Extraction Categories and Metrics

differentiate between high-frequency and low-frequency oscillations based on the inter-quartile transitions. For the 1st time-series, the number of transitions between Q1-to-Q2 and Q2-to-Q3 is above 6 times per 100 samples, while for the second sample the corresponding transitions happen close to once per 100 samples. This difference allows the model to deduce that the second time series changes with a lower frequency compared to the first.

To maintain the Brick schema hierarchy, we formulate our problem as a multi-class classification problem where each class corresponds to a unique combination of device labels with value 1 which occur in the training data. There are a total of 91 such unique combinations. We use all of the features described above to train an XGBoost classifier [2] to classify the labels corresponding to a time series. XGBoost is a versatile and popular choice for classification problems and has been shown to be effective in a wide range of applications [7]. The XGBoost model is configured for multi-class classification using the 'multi:softmax' objective function. The model architecture uses trees with a maximum depth of 13 levels and maintains a learning rate (eta) of 0.02. Both row and column subsampling are set to 0.92, meaning each tree uses 92% of the data points and features respectively. The gamma value of 0.0875 defines the minimum loss reduction needed for node splitting, while the minimum sum of instance weight is set to 1. The number of classes is dynamically set based on the unique classes in the dataset.

The hyperparameters for the XGBoost model were chosen based on a random search strategy across a handful of important parameters that we deemed likely to influence the model's skill. The search space encompasses tree depth ranging from 6 to 15 levels; subsample ratios ranging from 0.5 to 1.0; column sampling rates varying between 0.5 and 1.0; and gamma values from 0 to 0.15. The procedure maintains a fixed learning rate (eta) of 0.02 and a minimum child weight of 1, while employing an upper bound of 800 boosting rounds with an early stopping threshold of 20 rounds. We found the model to be somewhat insensitive to the hyperparameters in the random search, with most random combinations resulting in F1 scores within 20% of the best observed hyperparameter set. We found that the tree depth seemed to have the most influence on the model accuracy, with models with smaller tree depths performing generally worse than models with higher tree depths.



Metric	Value
Q1 freq	20.755
Q2 freq	46.351
Q3 freq	14.359
Q4 freq	18.535
Q1 → Q2	6.383
Q2 → Q1	6.484
Q1 → Q4	0.326
Q4 → Q1	0.163

Metric	Value
Q1 freq	33.397
Q2 freq	40.995
Q3 freq	20.681
Q4 freq	4.927
Q1 → Q2	1.102
Q2 → Q1	1.102
Q1 → Q4	0.000
Q4 → Q1	0.000

Metric	Value
Q1 freq	23.951
Q2 freq	0.000
Q3 freq	0.000
Q4 freq	76.049
Q1 → Q2	0.000
Q2 → Q1	0.000
Q1 → Q4	0.633
Q4 → Q1	0.633

Figure 2: Quartile based features for different types of time series

The model training process runs for 1000 boosting rounds, with multi-class log loss used as the evaluation metric. The model was trained using a laptop with an AMD Ryzen 5 processor and without a GPU, and still was able to complete training and inference on the test set within a few minutes without the need for parallel processing.

Finally, we utilize a post-processing step where we apply minimum value, mean value and maximum value limit checks to improve the classification of 2 specific devices: Wind Direction Sensors and Solar Radiance Sensors. The post-processing was based on manual observation of training samples of these sensors, where we noticed that the ranges of these sensors lay within a narrow band of values. For the Solar Radiance sensors, we noticed that the minimum values ranged between -1 and 1; the max values between 500 and 1100; and the mean values between 100 and 200. For the Wind Direction sensors, the maximum values ranged between 301 and 302.9, while the minimum values were below 50. Further, we observed that the respective sensors were the only sensors which exhibited this behavior, so we tagged all samples which satisfied the range checks as the respective sensor class. This resulted in a small improvement of the F1 score in the test set from .504 to .509.

The full code for this methodology has been made public at https://github.com/kaushik-gopalan/brick-submission.

3 Results

The classification accuracy of the proposed method relative to the competition baseline is shown in Table 2. The QBaSS method outperforms all the baseline methods by well over a factor of 10 in F1-score. The precision is also higher than the baseline methods by a large margin, though the "Random Uniform" method has a similar recall value as the proposed method. On the training dataset with a train-test split of 80-20 percent, the proposed method achieves a macro F1 score of ≈ 0.75.

Table 2: Results in Public Leaderboard Set. The main metric is the (macro) F1-score

Method	Precision	Recall	**F1-score**	mAP
Zero	0.0000	0.0000	0.0000	0.0000
Random Uniform	0.0231	0.4993	0.0356	0.0233
Random Proportional	0.0239	0.0224	0.0097	0.0234
Mode	0.0000	0.0106	0.0001	0.0000
One	0.0231	1.0000	0.0396	0.0231
QBaSS	0.578	0.494	**0.509**	0.462

The largest time cost in the proposed method is for feature extraction, which takes well over an hour on a mid-range laptop for the training and testing data combined. Training and inference of the XGBoost model for the test set require only a few minutes on a laptop without the need for a GPU. It is possible to reduce the time taken for feature extraction using parallel processing, though we have not attempted it at this stage. Figure 3 shows the performance of the proposed method on a class-wise basis. The performance is highly varied, with some classes achieving zero skill while others exhibit perfect accuracy. Figure 4 demonstrates that the proposed method delivers relatively high precision - typically more than 0.6 for most classes, while the recall is somewhat lower.

4 Discussion

In this manuscript, we present a machine-learning-based approach to efficiently classify IoT device data as part of the Brick by Brick 2024 competition. While the proposed approach achieves reasonable accuracy in the public dataset, there are several caveats that must be mentioned here. First, the performance in the training set (0.75 F1 score) was substantially superior to that in the test dataset. This is understandable since the training dataset is substantially smaller than the test one, but it does suggest a degree of overfitting in the model. Further, the robustness of the model is limited by other factors beyond our control. For example, if a temperature sensor is set to measure readings in degree Celcius in the test building, and an identical sensor is set to Fahrenheit in another, the model will not be able to identify the latter correctly. Further work, and larger training datasets, will be required to address the shortcomings in our approach.

Acknowledgments

We thank Mihir Hasabnis for helping us in brainstorming our approach.

References

[1] Wael Alsafery, Omer Rana, and Charith Perera. 2023. Sensing within smart buildings: A survey. *Comput. Surveys* 55, 13s (2023), 1–35.

[2] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[3] Li Ge and Li-Juan Ge. 2016. Feature extraction of time series classification based on multi-method integration. *Optik* 127, 23 (2016), 11070–11074. doi:10.1016/j.ijleo.2016.08.089

[4] Mengda Jia, Ali Komeily, Yueren Wang, and Ravi S Srinivasan. 2019. Adopting Internet of Things for the development of smart buildings: A review of enabling technologies and applications. *Automation in Construction* 101 (2019), 111–126.

[5] Chien-Liang Liu, Wen-Hoar Hsaio, and Yao-Chung Tu. 2019. Time Series Classification With Multivariate Convolutional Neural Network. *IEEE Transactions on Industrial Electronics* 66, 6 (2019), 4788–4797. doi:10.1109/TIE.2018.2864702

[6] Chrysi K Metallidou, Kostas E Psannis, and Eugenia Alexandropoulou Egyptiadou. 2020. Energy efficiency in smart buildings: IoT approaches. *IEEE Access* 8 (2020), 63679–63699.

[7] Didrik Nielsen. 2016. *Tree boosting with xgboost-why does xgboost win" every" machine learning competition?* Master's thesis. NTNU.

[8] 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. *Thirty-Eighth Annual Conference on Neural Information Processing Systems* (2024).

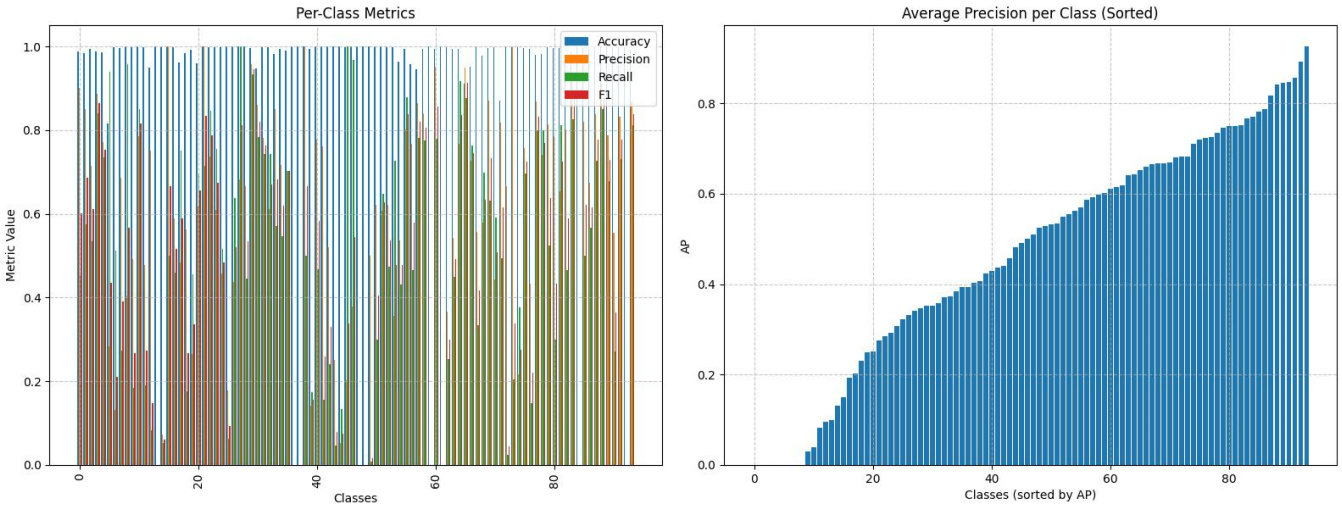


Figure 3: Summary of classification metrics for the *QBaSS* model

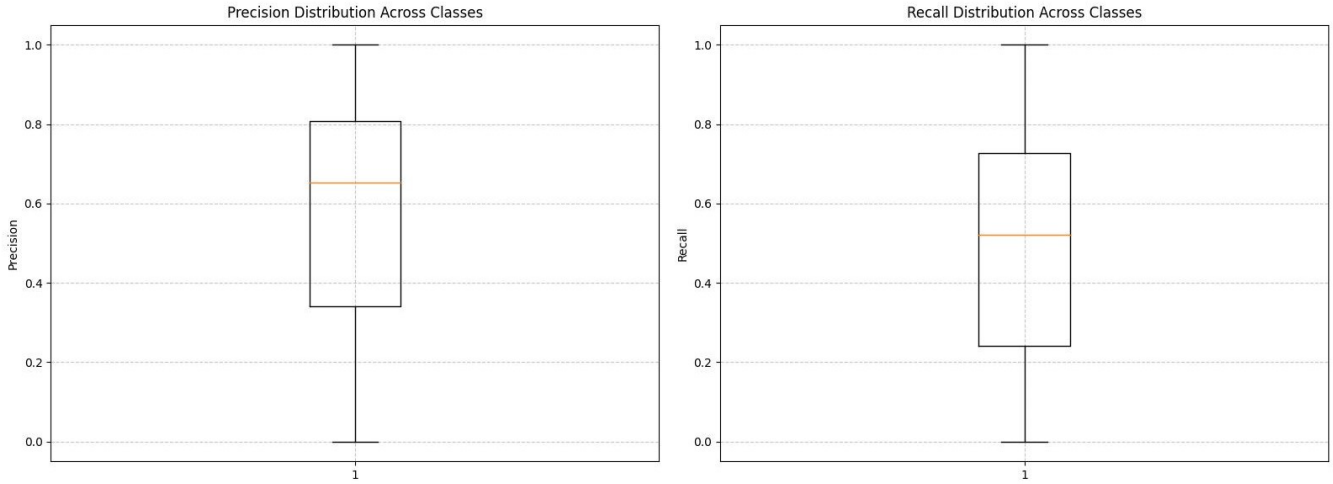


Figure 4: Distribution of precision and recall for the *QBaSS* model