Brick-by-Brick: Submission Documentation for TEAM TTHR

Thomas Tartiere
thomas.tartiere@gmail.com
Vancouver, Canada

Hatef Rahmani
hatef.rahmani.f@gmail.com
Vancouver, Canada

Abstract

This report presents our approach to the Brick by Brick 2024 Challenge, focusing on multi-label classification of smart building sensor data. We developed a robust preprocessing pipeline utilizing TS-Fresh for feature extraction, followed by a Random Forest Classifier with Label Powerset transformation. Our methodology effectively handled the complex task of classifying time-series data into 94 Brick schema sub-classes, addressing challenges such as irregular sampling rates and hierarchical label relationships. The final model demonstrated strong performance across cross-validation, evaluation, and leaderboard scores, highlighting the effectiveness of our feature engineering and classification approach.

ACM Reference Format:
Thomas Tartiere and Hatef Rahmani. 2025. Brick-by-Brick: Submission Documentation for TEAM TTHR. In . ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The Brick by Brick 2024 Challenge addresses a critical need in smart building management: the automated classification of IoT sensor data into standardized categories. With buildings consuming a significant portion of global energy, efficient data organization is crucial for optimizing energy use. Our task involved classifying time-series data from building sensors into 94 distinct Brick schema sub-classes, creating a standardized metadata tagging system. Key challenges included:

- Processing irregular time-series data from multiple sensors
- Addressing a multi-label classification problem with 94 possible labels
- Handling hierarchical label relationships
- Developing a solution generalizable across different buildings and time periods

Our approach centered on extensive feature engineering using TS-Fresh, followed by a Random Forest Classifier with Label Powerset transformation. We explored various models and techniques, including AutoGluon and a multi-level approach, before arriving at our final solution.

Conference'17, Washington, DC, USA
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 Methods

2.1 Feature generation

All time-series are converted to parquet files and sent to an Azure Blob Storage container to allow distributing computing. Our general idea was to convert the time-series into tabular features that can then be used in a typical classification problem. Differentiating data between different sensors require to pay attention to the magnitude of the time-series (min, max, mean, ...) but also to the shape of the curves. An additional challenge is provided by the fact that the time-series have different scale and resolution. To mitigate this issue, we generate features using the TSFresh library in two steps [1]:

- Generate "magnitude-like" features (min, max, mean, variance...) from the raw time-series
- Generate "shape-like" features (fast fourier transform, ...) from the time-series scaled using Robust scaling

This computation is performed on a Dask cluster in Microsoft Azure using Coiled (https://www.coiled.io/). With six virtual machines of size Standard_F16s_v2 on Azure, the computation took about 16h for the test data, for each of the two steps. The idea behind this process is that this could help with generalization as the shape-like features are very dependent on resolution and scale.

Through this process, we obtained 777 features. We trained a Random Forest Classifier and use the Impurity-based feature importances to identify a set of relevant features. In addition, features that lead to very large float values are dropped to prevent numerical issues. Finally, features with many NaN values are dropped. We finally select the top 60 and top 100 features to use in training and cross-validation.

2.2 Model selection and training

One of the challenges in this classification problem is that most machine learning libraries offer limited support for multi-label classification. They often rely on converting the problem into a multi-class one using a "one-vs-rest" approach, which can fail to capture relationships between labels. To address this, we use the Label Powerset method provided by the scikit-multilearn library [2]. Label Powerset is a problem transformation technique that converts a multi-label problem into a multi-class problem by training a single multi-class classifier on all unique label combinations found in the training data. With this transformation, we evaluate the macro F1-score across various machine learning models:

- Logistic Regression
- Support Vector Machines
- k-Nearest Neighbors
- Random Forest
- XGBoost

- LightGBM

For each model, we split the data into training and evaluation sets using a hierarchical approach that ensures all classes are represented in both splits. Tree-based models, particularly Random Forest and XGBoost/LightGBM, achieved significantly better results, making them the focus of our hyperparameter tuning efforts.
Models are trained using cross-validation with the IterativeStratification method from the scikit-multilearn package. This approach ensures that each fold maintains a balanced representation of higher-order label combinations. For each fold, we generate probability scores for each class and compute the binary F1-score at different thresholds. After iterating through all folds, we determine the threshold that maximizes the average binary F1-score for each class and then compute the overall macro F1-score for training.

3 Final solution

The best model was Random Forest Classifier with the following optimzied hyperprameters:

- n_estimators: 1000
- max_features: "sqrt"
- max_depth: 13
- min_samples_split: 2
- min_samples_leaf: 1
- criterion: "entropy"

This model generated the below scores (macro F1), combined with the LabelPowerSet:

- Cross validation score: 0.694
- Holdout set score: 0.725
- Leaderboard: 0.513

The leader board result is provided in figure 1.

References

[1] Braun N. Neuffer J. Maximilian, C. 2018. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). (2018).
[2] P. Szymański and T. Kajdanowicz. 2017. A scikit-based Python environment for performing multi-label classification. *ArXiv e-prints* (Feb. 2017). arXiv:1702.01460 [cs.LG]

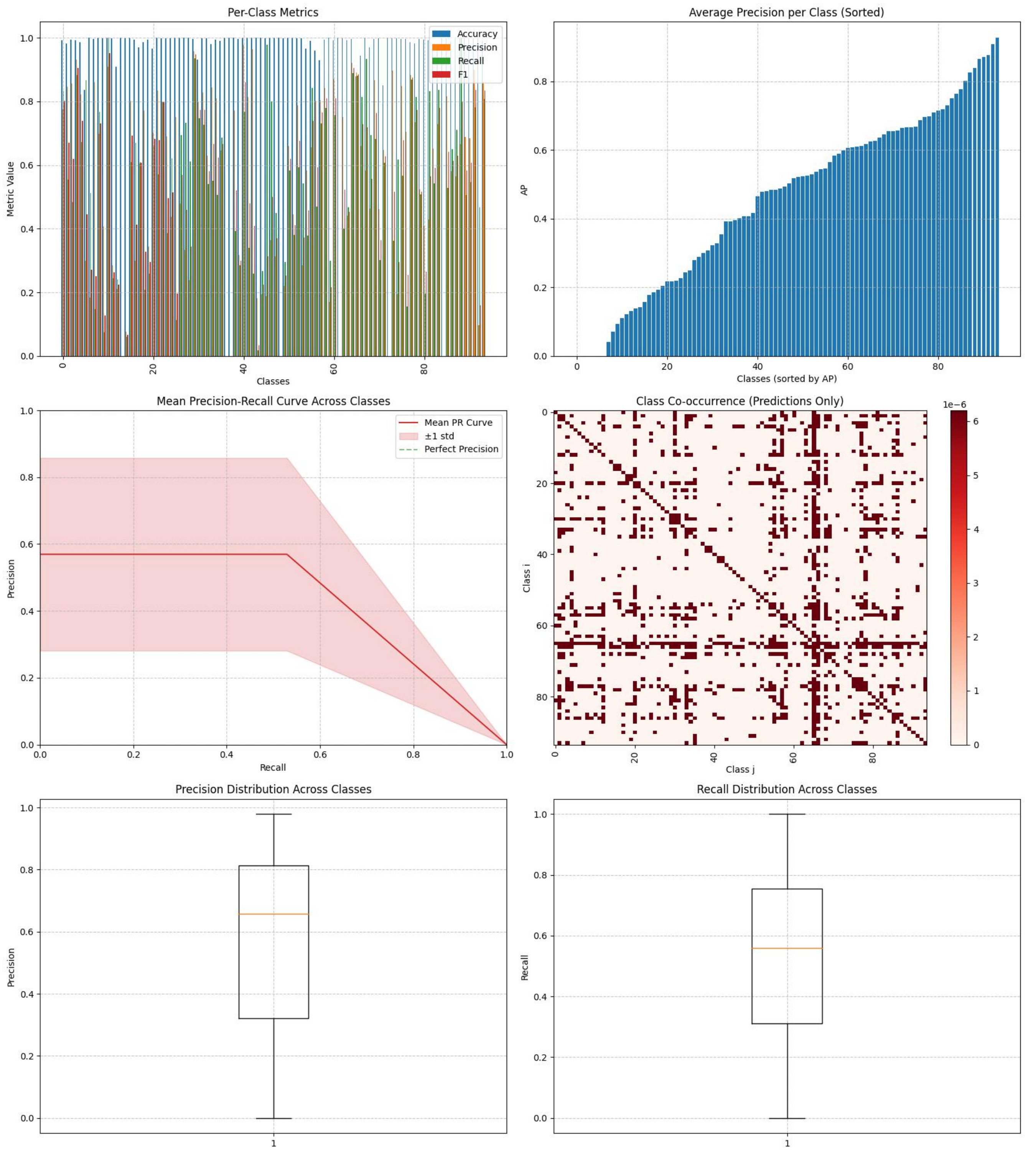

Figure 1: Visualisation of the submission.