DATA11002 Project Report: Predicting New Particle Formation

Yaohui Ma (thezmmm)
Junhao Shao(Junhao Shao)
Kaggle Team Name: Group 48

December 19, 2025

1 Data Analysis

We performed data analysis on the training dataset using an event distribution bar chart, a mean feature correlation heatmap, and a PCA projection map. Before the actual data analysis, we performed error check on the training data set. First we checked if there are some missing values in the feature columns and then we checked if there are any duplicated values in the features. The result is we didn't find any missing values and the duplicated values.

1.1 Event distribution

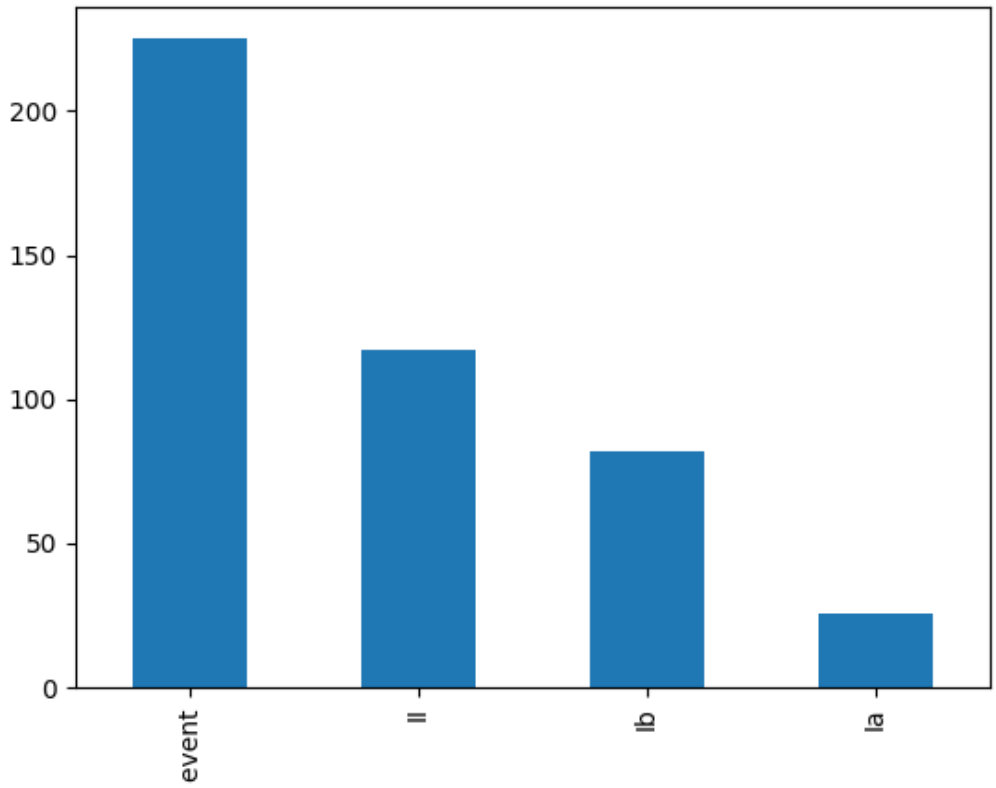

Figure 1: Event distribution

The bar chart in the Figure 1 shows the distribution of values in the column class4 in the Train data set. Each bar represents one category, and the height of the bar shows how many times that category appears in the dataset. From the figure, we can see that the category “nonevent” has the highest count, followed by “II”, then “Ib”, and finally “Ia”. This means that most of the observations in the dataset belong to the “nonevent” class, which means no NPF event took place, while the “Ia” class occurs only a few times. Overall, the chart illustrates that the data in class4 are unevenly distributed, with one dominant category “nonevent” and a few much smaller ones.However, for binary classification, the distribution is even, with 'nonevent' accounting for approximately half of the total.

1.2 Mean values of features

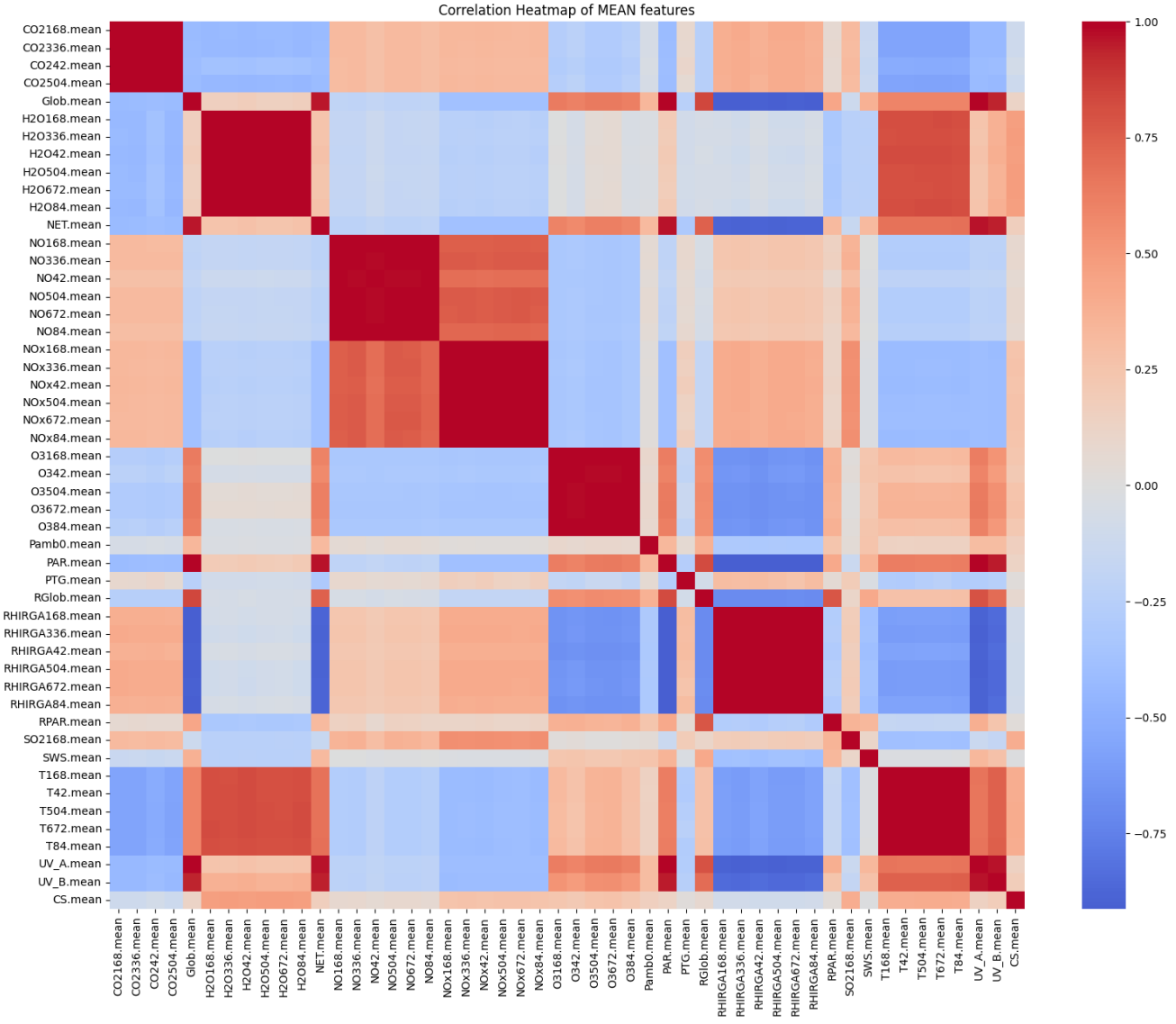

Figure 2: Mean values of features

Based on the correlation heatmap in the Figure 2, the data reveals several distinct patterns that align perfectly with known principles of atmospheric physics and chemistry. The matrix highlights relationships between different features by different color intensity.
First, we observe strong positive correlations, indicated by the dark red clusters. The most prominent relationship is between solar radiation variables, such as Global radiation, PAR, and UV, and temperature. This is logically consistent, as increased solar radiation directly drives the rise in air temperature.

Similarly, the nitrogen oxide variables, such as NO and NOx, form a tight, dark red block, suggesting they originate from common emission sources, which means when levels rise, they rise simultaneously across all measurement points. Additionally, water vapor ,H2O , correlates positively with temperature, reflecting the physical fact that warmer air has a higher capacity to hold moisture.
Second, the heatmap displays strong negative correlations, shown in the blue areas. A negative relationship is visible between Relative Humidity (RHIRGA), and temperature, and radiation (Glob, PAR, UV) : as the sun shines and temperatures rise, relative humidity typically drops. Another significant negative correlation exists between Ozone (O3) and Nitrogen Oxides (NOx), likely due to the chemical titration reaction where nitric oxide destroys ozone. Furthermore, Carbon Dioxide (CO2) shows a negative correlation with solar radiation (PAR), which can be attributed to photosynthesis; during the day, active vegetation consumes CO2, lowering its concentration compared to nighttime.
Finally, the data indicates internal consistency across measurement heights. Variables labeled with different numerical suffixes (for example, 168, 336, 504), which represent measurement heights, correlate almost perfectly with one another. This implies that the air mass is well-mixed, meaning that temperature and gas concentrations remain relatively uniform across the vertical profile of the measurement site.

1.3 PCA projection

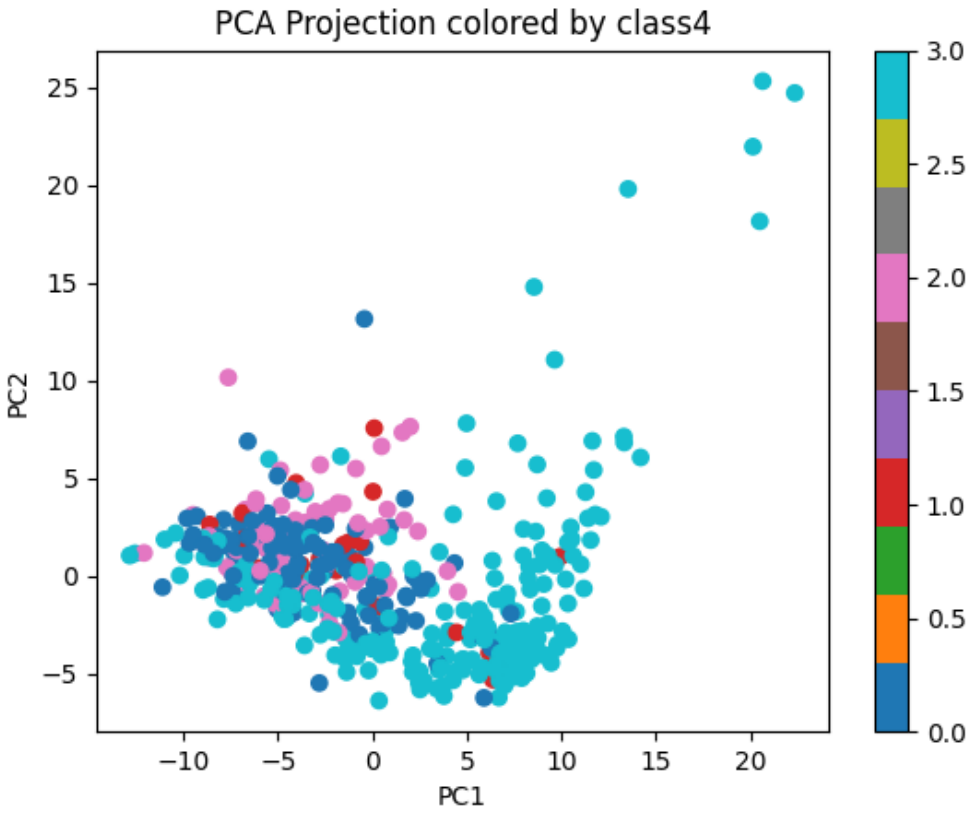

Figure 3: PCA projection

The Figure 3 shows a two-dimensional Principal Component Analysis projection of the dataset, where each point represents a single observation. The data have been projected onto the two principal components, PC1 and PC2, which are linear combinations of the original variables and are chosen to capture the maximum possible variance in the data. They represent two independent trends in the data, where several features increase or decrease together. The points are colored according to the variable class4, which take four discrete values,0,1,2,3, as indicated by the color bar on the right.

The distribution of points along the PC1 axis suggests that this component explains a substantial portion of the overall variability in the dataset. For example the turquoise points corresponding to the highest class value, is clearly shifted toward higher PC1 values. This indicates that observations belonging to this class differ systematically from the others in terms of the original features that contribute most strongly to PC1. In other words, PC1 captures a direction in feature space that is highly informative for distinguishing this particular class.

On the other hand, the remaining classes are largely concentrated in the left and central regions of the plot and show a significant degree of overlap with one another. This overlap implies that, when considering two principal components, these classes do not separate cleanly in a linear manner. The second principal component, PC2, contributes additional variance but does not appear to provide strong discriminative power between these overlapping classes, as they share similar ranges along the PC2 axis.

1.4 Histogramms for all features

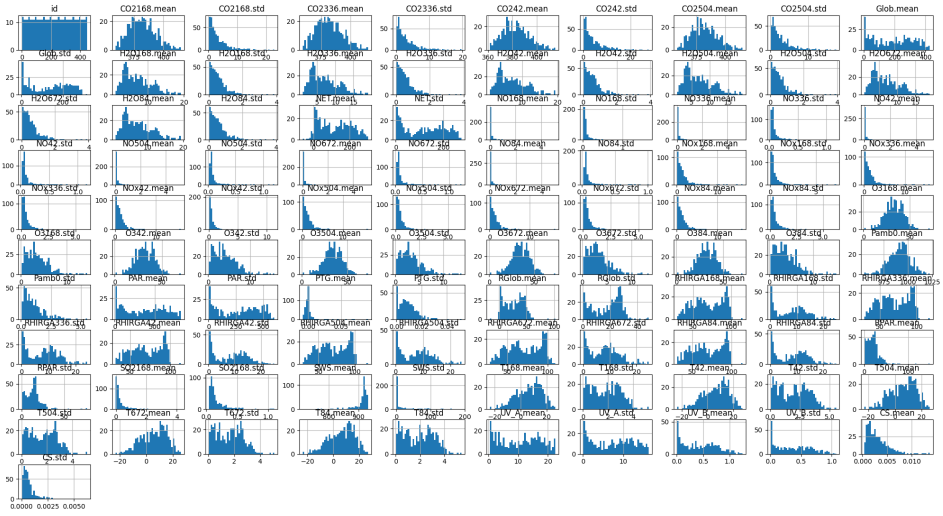

Figure 4: Feature histogramms

In the figure 4, we have list of histogramms, every histogramm corresponds to one feature. The x-axis of each histogramm corresponds to the observed values of that feature, while the y-axis corresponds to the time that value has occured in the data set. Together, the histograms provide an overview of how the different variables are distributed.
Most of the histogramms that ends with ".mean" are right-skewed, the bars cluster on the left with a long tail to the right. Which means that most of the small values occured frequently, while a few larger values extend the distribution toward the right. This suggests that for many values in the data set, typical readings are relatively low with some occasional higher observations. The variables that ends with ".std" are also mainly right-skewed. This means most of the features has small standard deviation values, and the measurements has low variability. A few variables such as temperature or UV indices show flatter or even bimodal shapes, which could reflect different measurement conditions, such as day and night cycles or differences between sampling locations.

1.5 Distrbution differences of the top 3 features across different Class 4

We used boxplot for showing the distribution diffirences. The box shows the interquartile range, which means the middle 50 percent of the values, and the line inside the box is the median. The whiskers extend to the minimum and maximum values that aren't considered outliers.

1.5.1 Feature 1: RHIRGA84

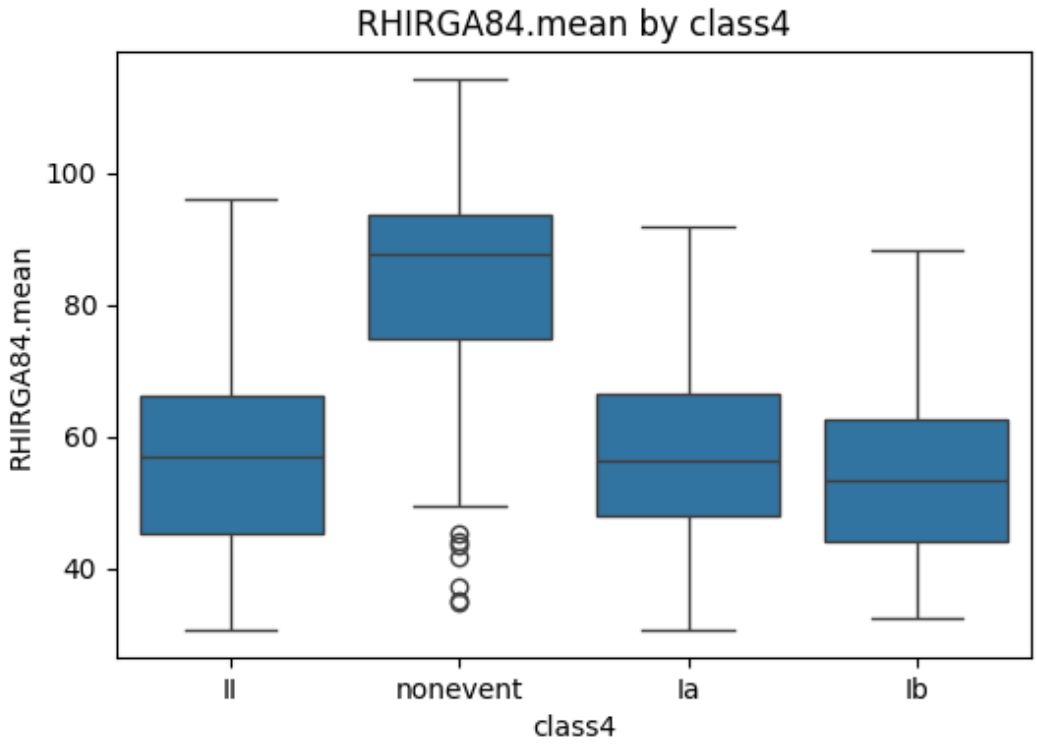

Figure 5: Distribution of RHIRGA84

Based on the Figure 5, the RHIRGA84 values in “nonevent” group has a higher and wider distribution, indicating generally larger values, and the distribution differs from ”II”,”Ia” and ”Ib”. The event groups (II, Ia, Ib) have lower and similar distributions, centered around 50–60.

1.5.2 Feature 2: RHIRGA168

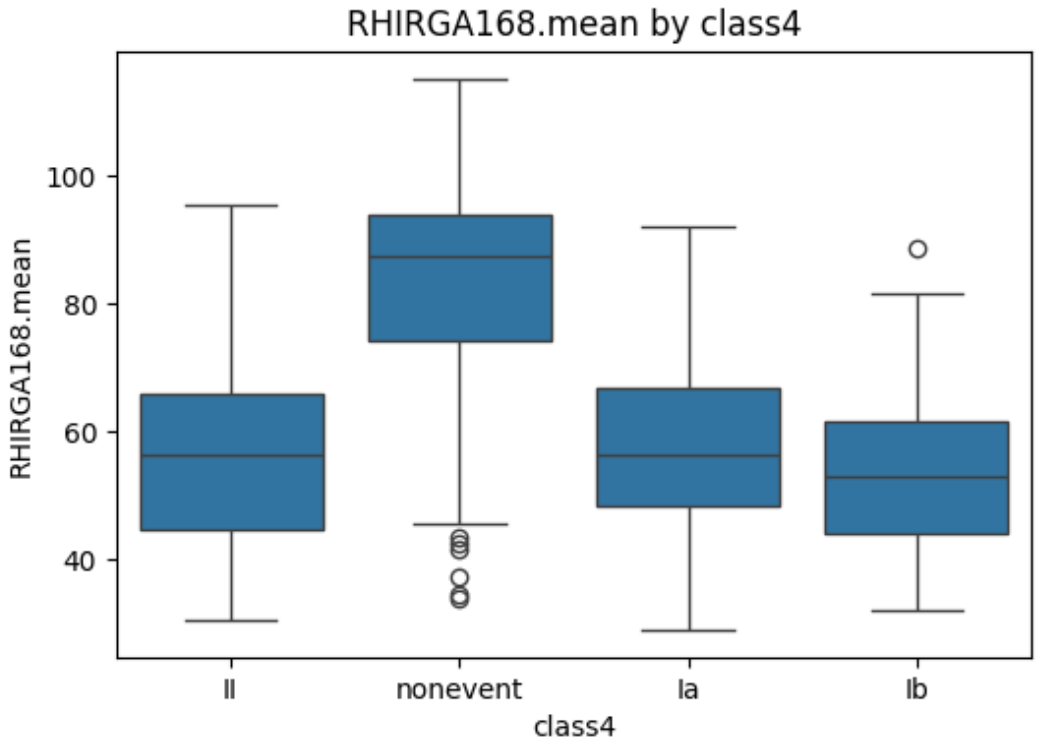

Figure 6: Distribution of RHIRGA168

Based on the Figure 6, we can find out that the distribution of RHIRGA168 is similar to RHIRGA84. They are both systematically higher in the nonevent group and lower and similar across the event classes (II, Ia, Ib), which indicating a consistent distribution difference between event and nonevent cases.

1.6 Feature 3: RHIRGA336

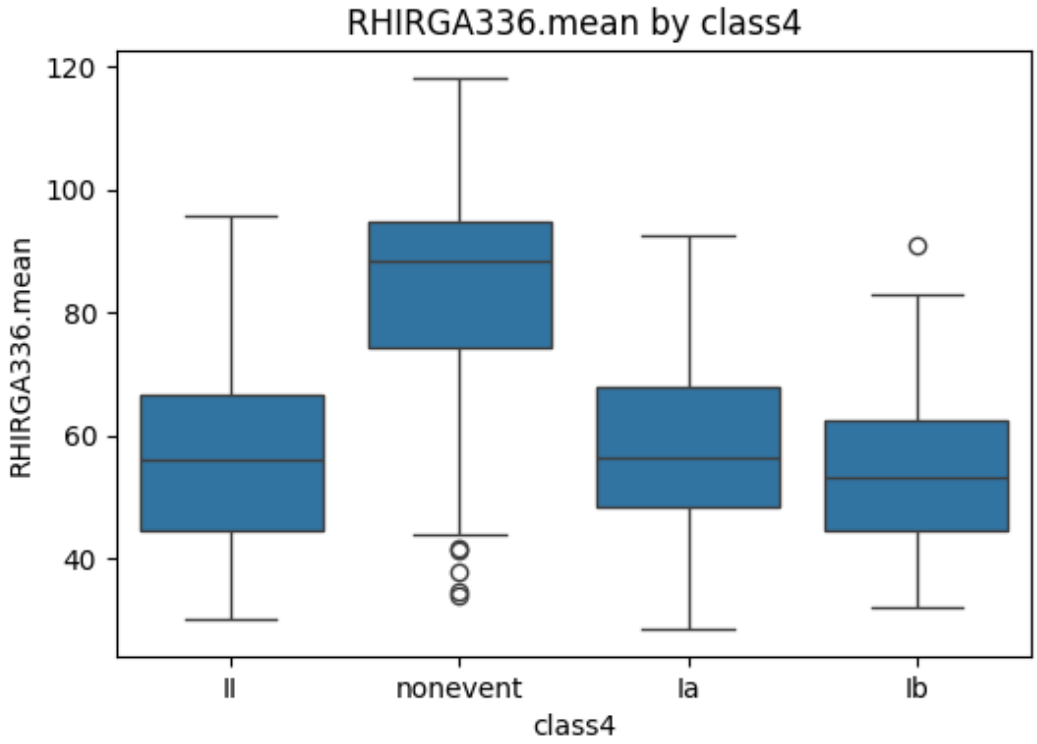

Figure 7: RHIRGA336

Based on the Figure 7, we found out that all three most important features has pretty similar distributions. Which means these three variables measure a very similar phenomenon or are strongly correlated with each other. This information is valuable, because all it tells us that if one feature changes, the other two will probably change same way.

2 Data Preprocessing

Before training any machine learning models, we applied a systematic data preprocessing pipeline to ensure that the input data was clean, consistent, and suitable for supervised learning algorithms. All preprocessing steps were implemented using standard tools from the scikit-learn library to improve reproducibility and methodological correctness.

2.1 Column Removal

We first removed several columns that were not suitable as predictive features:

- id: an identifier column that carries no predictive information and could introduce noise if included.
- date: a timestamp-like variable that was not explicitly modeled as a time-series feature and was therefore excluded.
- partlybad: All data entries with the same partlybad value are considered invalid.

2.2 Target Variable

The dataset includes a target variable named **class4**, which represents a four-class classification problem.

We explicitly distinguished between training and test datasets based on the presence of this column:
Training data: contains class4
Test data: does not contain class4

For the training data, the target variable class4 was transformed using **label encoding**, mapping the original class labels to integer values. This encoding step is necessary because most machine learning algorithms in scikit-learn expect numerical target labels rather than categorical strings.

The features (X) and target (y) were then separated to avoid data leakage during subsequent preprocessing steps.

2.3 Missing Value Imputation

Although no missing values were found during the missing value check in the EDA process, for the sake of program robustness, we applied **median imputation** using 'SimpleImputer(strategy="median")'.

We chose median imputation for the following reasons:
It is robust to outliers compared to mean imputation.
It preserves the distribution of numerical features reasonably well.
It avoids discarding samples, which could reduce the effective training set size.
Imputation was applied **only to the feature matrix**, never to the target variable.

2.4 Feature Scaling

After imputation, all features were standardized using **z-score normalization (StandardScaler)**, transforming each feature to have zero mean and unit variance.

Standardization was applied because:
Many machine learning algorithms (e.g., logistic regression, SVMs, distance-based models) are sensitive to feature scales.
It ensures that all features contribute comparably to the learning process.
It improves numerical stability and convergence during training.

2.5 Output Generation and Reproducibility

The processed features were converted back into a structured DataFrame and, for the training set, the encoded target variable was appended as the final column. The fully preprocessed datasets were saved as CSV files for later use.

To ensure reproducibility and consistent preprocessing during inference, all fitted preprocessing objects—including:

1. the label encoder,
2. missing-value imputer,
3. the feature scaler

were saved to disk using 'joblib'. These objects can later be reloaded and applied to unseen data, ensuring that training and test data undergo identical transformations.

3 Model Selection

3.1 Model Comparison

Given the relatively small dataset (450 training samples) and a high-dimensional numerical feature space composed primarily of summary statistics (mean and standard deviation) extracted from multiple environmental sensor signals, we evaluated three supervised learning approaches:

- Logistic Regression (implemented by Junhao)
- LightGBM (LGBM) (implemented by Yaohui)
- XGBoost (implemented by Yaohui)

All models were trained using the same preprocessed feature set to ensure a fair comparison.

Logistic regression was used as a baseline model due to its simplicity and interpretability. However, its performance was significantly worse than that of the tree-based ensemble methods, achieving an accuracy of approximately 0.5. We believe this poor performance is mainly due to the high dimensionality and structure of the feature space.

As a result, the model struggled to learn a discriminative representation in this high-dimensional setting, even with regularization. This outcome suggests that the classification task is fundamentally non-linear and not well suited for purely linear models.

LightGBM was evaluated as an alternative gradient boosting framework with a different tree growth strategy. In the four-class classification task, its performance was comparable to but slightly weaker than that of XGBoost, potentially due to its higher sensitivity to small dataset sizes.

However, when the task was reformulated as a binary classification problem, LightGBM unexpectedly surpassed XGBoost and achieved improved performance. A plausible explanation is that:

Binary classification simplifies the decision boundary and reduces class overlap.

LightGBM's leaf-wise growth strategy can exploit simpler class separations more aggressively.

In this setting, LightGBM was better able to utilize the available feature information without overfitting.

XGBoost consistently outperforms the other two models, when using the same model for both four-class and two-class classification tasks. It achieved substantially higher classification performance and demonstrated stable behavior across different task formulations.

The advantages of XGBoost in this context include:

Its ability to automatically capture non-linear relationships and feature interactions.

Strong built-in regularization mechanisms, which are crucial given the limited dataset size (450 samples).

Robustness to correlated features and noise.

3.2 Performance Plateau with XGBoost

Despite these strengths, we observed that XGBoost's performance plateaued at around 0.74 when it is used for both the four-class and binary classification tasks. Further hyperparameter tuning and feature-level adjustments —such as adjusting tree depth, learning rate, number of estimators, and regularization parameters— led to only marginal improvements, suggesting that XGBoost was already close to the performance ceiling imposed by the data and feature representation.

We believe the observed performance ceiling is mainly due to the following factors:

1. Limited sample size: With only 450 training samples and a relatively large number of correlated features, model capacity is constrained by the risk of overfitting.

2. Highly correlated features: Many features are derived from the same sensors and differ only by aggregation window (e.g., mean vs standard deviation), which limits the amount of independent information available.

3. Class complexity in four-class classification: The four-class task requires the model to separate subtle class boundaries, which may not be fully distinguishable given the available features.

As a result, further tuning of XGBoost primarily increased model complexity without improving generalization.

Interestingly, when reformulating the problem as a binary classification task, replacing XGBoost with LightGBM led to a noticeable performance improvement, surpassing the XGBoost plateau.

A likely explanation is that:

Binary classification simplifies the decision boundary, reducing class ambiguity.

LightGBM's leaf-wise tree growth can exploit simpler class separations more aggressively.

In this reduced-complexity setting, LightGBM was able to capture fine-grained splits that XGBoost's more conservative tree growth did not exploit as effectively.

3.3 Final Model Selection Summary

Based on the empirical comparisons and performance analysis, we selected different models for the two task formulations:

Four-class classification: **XGBoost** was chosen as the final model due to its consistently strong performance, robustness, and stable generalization.

Binary classification: **LightGBM** was selected as the final model, as it surpassed XGBoost and achieved the best overall performance in this setting.

This final selection reflects a data-driven approach to model choice. Rather than relying on a single algorithm, we evaluated multiple models and selected the best-performing one for each task formulation. The results also suggest that further performance improvements would likely require additional data or more informative features rather than further tuning of existing models.

4 Optimization Attempts and Methodological Exploration

4.1 Four-Class First vs. Binary First Strategy

Since the Kaggle evaluation score is computed by jointly considering the performance of both the four-class and binary classification tasks, achieving strong results on both tasks is essential. We therefore explored different modeling strategies to balance performance across these two objectives.

We considered two alternative approaches:

1. Approach 1: Independent Modeling

   Train two separate models:

   one model directly optimized for the four-class classification task,

   and another model trained independently for the binary classification task.

2. Approach 2: Hierarchical Classification

   First apply a binary classifier to distinguish event vs. non-event. Then, restrict the dataset to samples predicted as event and apply a second-stage multi-class classifier to distinguish between the event subcategories.

Empirically, the independent modeling approach (Approach 1) consistently outperformed the hierarchical strategy in terms of the combined Kaggle score.

We attribute the inferior performance of the hierarchical approach to two main factors:

1. Error propagation from the binary classifier: The initial binary classification stage did not achieve sufficiently high accuracy. As a result, misclassified samples at this stage could not be recovered in the subsequent multi-class classification, leading to compounded errors.

2. Reduced effective training data for multi-class learning: After filtering the data using the binary classifier, the remaining samples available for the three-class classification were significantly fewer. Given the already limited dataset size, this reduction further constrained the model's ability to learn reliable class boundaries, resulting in underfitting.

Based on these observations, we adopted the **independent modeling strategy**, training separate models for four-class and binary classification. This approach provided better overall performance and avoided the cascading error and data scarcity issues inherent in the hierarchical pipeline.

4.2 Hyperparameter Tuning

To further improve model performance, we conducted systematic hyperparameter optimization for the XGBoost model using **Optuna**, a state-of-the-art automated hyperparameter optimization framework.

We optimized the XGBoost model using a five-fold stratified cross-validation scheme to ensure robust performance estimation and to avoid overfitting to a single validation split. The optimization objective was to minimize multiclass log-loss, which is well aligned with the evaluation metric used in the Kaggle competition.

The following groups of hyperparameters were explored:

- Model capacity and structure: max_depth, min_child_weight, gamma
- Learning dynamics: learning_rate, n_estimators
- Sampling and regularization: subsample, colsample_bytree, reg_alpha, reg_lambda

The search space was intentionally kept broad to allow Optuna to explore both conservative and expressive model configurations.

Despite this extensive tuning process, the performance gain obtained through hyperparameter optimization was relatively modest, typically in the range of 0.002–0.004 in terms of the evaluation score.

While the tuned model consistently outperformed the default-parameter baseline, the improvement was significantly smaller than initially expected.

Although hyperparameter optimization did not lead to large absolute gains, this step was still essential for confirming that the model was operating near its optimal regime. The results also reinforced an important insight: beyond a certain point, further improvements are more likely to come from better data representations or additional data rather than from more aggressive tuning.

4.3 Class Weights

During the data analysis phase, we observed that the class distribution in the training dataset was highly imbalanced. For example, some event classes (e.g., Ia) occurred much less frequently than others, while the non-event class constituted roughly half of the data.

To address this imbalance, we introduced class weights proportional to the inverse of their frequencies as shown in the table 1.

Class	Weight
Ia	1/0.057778
Ib	1/0.182222
II	1/0.26
nonevent	1/0.5

Table 1: Class weights used in the model.

These weights were passed to the model training procedure so that misclassifying rare classes would incur a higher penalty, encouraging the model to pay more attention to underrepresented classes.

Adding class weights resulted in a measurable improvement in performance, with the score increasing by approximately 0.006–0.01. While the improvement is modest, it demonstrates that compensating for class imbalance can enhance model sensitivity to rare but important events, especially in a multi-class classification setting.

4.4 Multi-Round Training

In an effort to further stabilize model predictions and potentially improve performance, we experimented with multi-round training. The idea was to train the same classifier multiple times with different random seeds and then aggregate the results across rounds and cross-validation folds.

We applied Stratified K-Fold cross-validation (5 folds) to maintain class balance in each split.

The training was repeated for three rounds, each with a different random seed to introduce variation in the training process.

For each fold, we used a LightGBM classifier with calibrated probabilities (CalibratedClassifierCV) to improve probability estimates.

Class imbalance was addressed with scale_pos_weight calculated from the training fold.

Despite the rationale, this multi-round strategy did not improve performance. In fact, it occasionally caused a negative effect, slightly reducing the validation scores.

The dataset is relatively small (450 samples), so repeated re-sampling and multiple rounds introduced additional variance rather than reducing it.

Aggregating predictions across rounds did not provide new independent information since the model was already close to its capacity on this data.

Calibration and reweighting within each round could not compensate for the inherent limits of the feature set and sample size.

Multi-round training can be beneficial in larger datasets or in models prone to instability, but in this specific task, the combination of a small dataset and high-dimensional features limited its effectiveness. This experiment demonstrates the importance of empirically validating advanced training strategies rather than assuming they always provide gains.

4.5 Why No Explicit Feature Selection Was Applied

Although feature selection is often beneficial in high-dimensional settings, we deliberately chose not to apply an explicit feature selection step in this project. This decision was motivated by several considerations related to dataset size, feature structure, and model choice.

The training set contains only 450 samples, which already limits the amount of information available for learning robust decision boundaries. Applying aggressive feature selection in such a low-sample regime risks removing features that contain weak but complementary predictive signals. Given that many features represent different statistical summaries (mean and standard deviation) of the same sensor signals at different time windows, removing features could inadvertently discard useful contextual information.In this setting, the cost of losing potentially informative features was considered higher than the cost of retaining redundant ones.

While the feature space is relatively high-dimensional, the features are not arbitrary. Most features are derived in a structured way (e.g., sensor $\times$ time window $\times$ summary statistic). This means that redundancy across features is expected and systematic rather than noisy.

Tree-based ensemble models such as XGBoost and LightGBM are well known for their ability to:1)Handle correlated features,2)Automatically select informative splits during training,3)Down-weight irrelevant features through regularization.As a result, explicit feature selection was less critical than it would be for linear models.

Our experiments showed that even without explicit feature selection, XGBoost achieved strong and stable performance, approaching an apparent performance ceiling. Additional efforts focused on model complexity control and validation strategy yielded diminishing returns, suggesting that the model was already effectively exploiting the available feature set.

With a larger dataset, feature selection or dimensionality reduction techniques (e.g., model-based feature importance pruning or PCA) could be explored more safely. However, given the current dataset size and the strong built-in regularization of the chosen models, we judged that retaining the full feature set was the more conservative and robust choice.

5 Results and Insights

5.1 Final Model and Performance

After a series of modeling and optimization steps, our final solution combined:

Four-class classification: XGBoost

Binary classification: LightGBM

This combination achieved a final Kaggle score of **0.75185**, **ranking 8th** in the competition leaderboard.

5.2 Insights

Through experimentation, we observed that XGBoost consistently delivered strong performance on the four-class task, effectively capturing non-linear interactions among the many sensor-derived features. However, despite extensive hyperparameter tuning, the four-class XGBoost model appeared to reach a performance plateau around 0.74, suggesting that the available data and feature set impose a natural limit on achievable accuracy. In contrast, reformulating the binary task with LightGBM allowed for a notable improvement beyond this plateau, likely due to simpler decision boundaries and LightGBM's ability to exploit them efficiently.

We also investigated alternative approaches, including logistic regression and feature selection. Logistic regression performed poorly, achieving only a baseline score of around 0.50, likely due to the

high feature-to-sample ratio and the inherently non-linear relationships present in the data. Feature selection was considered but ultimately not applied, as the relatively small dataset (450 samples) raised concerns that aggressive feature filtering would remove informative signals and reduce the model's capacity to generalize.

Additional strategies, such as multi-stage classification, multiple training rounds, cross-validation, and the use of class weights, were applied to ensure robust and calibrated predictions. While some optimization efforts, including hyperparameter tuning via Optuna, yielded only minor improvements in log-loss, they helped confirm the stability and reliability of the final model.

Overall, the results demonstrate that careful preprocessing, the choice of complementary algorithms for four-class and binary tasks, and methodical evaluation strategies were critical to achieving competitive performance. The combination of XGBoost for the four-class task and LightGBM for the binary task balanced model expressiveness and generalization, producing a solution that performed consistently on unseen test data. Notably, our score remained stable before and after the leaderboard freeze, while many other teams experienced declines, contributing to a substantial **jump of 20 ranks** in the final placement.

The final score of 0.75185 slightly exceeded our initial expectations and reflects both the strengths and limitations of the dataset and feature set. While XGBoost for the four-class task reached a natural performance ceiling, LightGBM for the binary task helped overcome this limit, providing a modest but meaningful improvement. These observations suggest that further gains would likely require additional data, more informative features, or alternative problem formulations, even though the current models already perform competitively.

6 Self Evaluation

6.1 Project Self-Evaluation

Suggested Grade: 5

We believe this project merits a grade of 5 based on the grading criteria provided for the course.

The project follows a clear and systematic machine learning workflow, including data preprocessing, exploratory analysis, model selection, hyperparameter tuning, and performance evaluation. We carefully justified all major design choices, such as the selection of XGBoost for the four-class task and LightGBM for the binary task, based on empirical comparisons and validation results rather than convenience or default settings.

Feature engineering and model parameter selection were conducted in a methodologically sound manner, using cross-validation and controlled experiments. While the dataset size limited the potential for aggressive feature selection or more complex modeling, these limitations were explicitly acknowledged and reflected upon in the report.

The final report is written as a self-contained technical document that clearly explains what was done, why decisions were made, and what insights were gained. The discussion goes beyond reporting results and includes critical reflection on model performance, observed performance ceilings, and potential directions for future improvement.

Overall, the project demonstrates a solid understanding of supervised learning methods, careful experimental design, and clear technical communication, which together justify a top-grade evaluation.

6.2 Team Self-Evaluation

Group Grade: 5

All team members contributed actively throughout the project. While some efforts, such as the time spent experimenting with logistic regression, did not lead to significant improvements, these attempts still provided valuable insights and helped guide our subsequent decisions. Overall, the group worked collaboratively, shared responsibilities fairly, and maintained productive discussions, which enabled us to achieve a strong final solution and a competitive score on the challenge.