Forecasting Room Occupancy 30 Minutes Ahead from Environmental Sensors

Zheng Ma
Brown University, DATA 1030

December 10, 2025

Github repository: `https://github.com/zhengma-cyber/finalproject`

1 Introduction

Accurate occupancy detection is a cornerstone of modern smart building management. Heating, Ventilation, and Air Conditioning (HVAC) systems are major energy consumers, often running on static schedules regardless of actual room usage. By predicting occupancy in advance, systems can adjust settings proactively, such as cooling or heating a room before people arrive, thereby balancing energy savings with occupant comfort.

This project modifies the standard occupancy detection task to a forecasting problem. Specifically, the objective is to predict whether a room will be occupied 30 minutes in the future. This 30-minute horizon is chosen to simulate the typical "start-up delay" or thermal inertia required for an HVAC system to change a room's temperature effectively.

The problem is defined as a binary classification task where the target variable is Occupancy (1 for occupied, 0 for empty) at time t. The input features are derived from environmental sensors available up to time t-30. The dataset used is the UCI "Occupancy Detection" dataset (Candanedo & Feldheim, 2016), collected from an office room over several days. In their original work, Candanedo and Feldheim focused on immediate occupancy detection, achieving accuracies ranging from 95% to 99% using Linear Discriminant Analysis (LDA) and Random Forest models. They notably observed that time-stamp information significantly improved performance, while using temperature alone yielded roughly 85% accuracy. While this project modifies the task to a more challenging 30-minute ahead forecasting problem, our final XGBoost model achieves a Test F1-score of approximately 0.94. This performance is comparable in magnitude to the original benchmarks, confirming that environmental proxies combined with lagged features remain robust predictors even when forecasting future states.

2 Exploratory Data Analysis

2.1 Class balance and baseline difficulty

The dataset exhibits significant class imbalance, with the room being unoccupied for the majority of the time, such as during nights and weekends. As a result, a naive model that predicts "Empty" (0) would achieve high accuracy but be useless for the application as it fails to detect people. Conversely, a constant "Occupied" predictor ensures comfort but wastes energy. Due to this imbalance, we utilize the F1-score and PR-AUC (Precision-Recall Area Under Curve) as our primary evaluation metrics instead of accuracy.

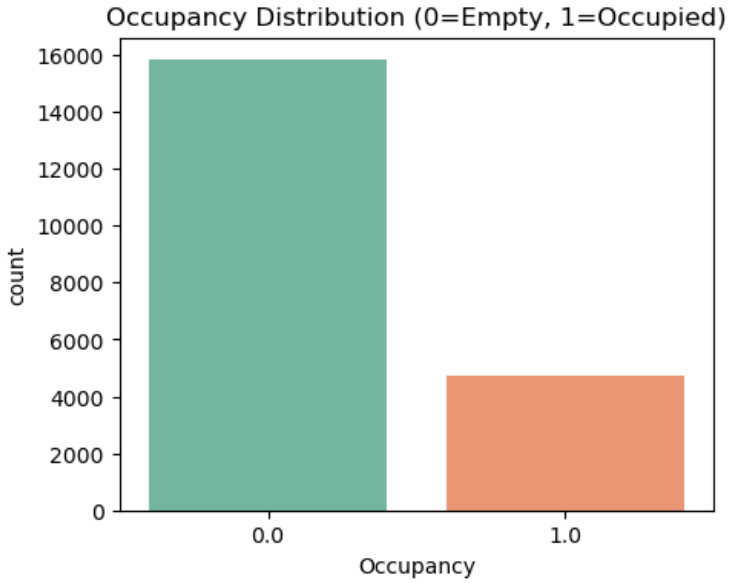
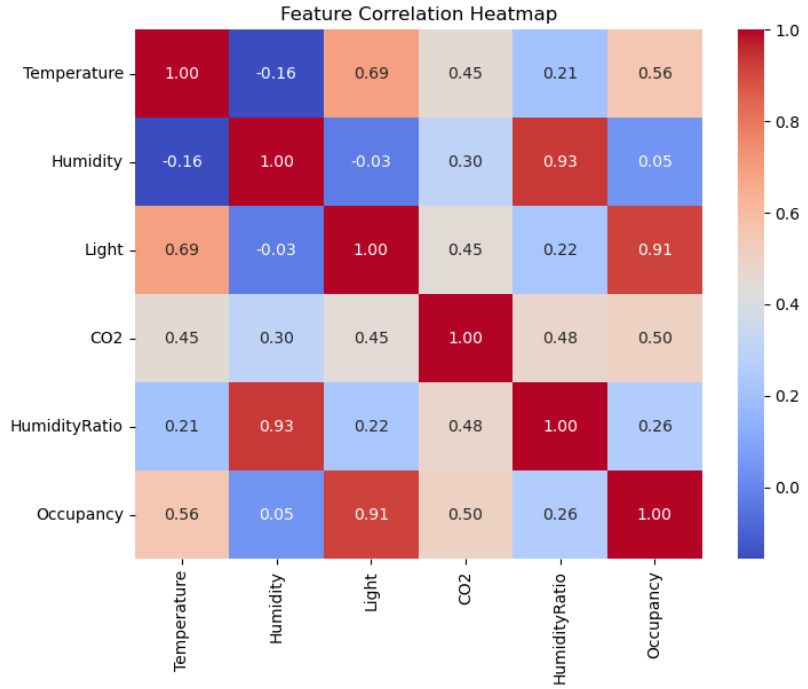


Figure 1: Class distribution of the Occupancy label over time, showing clear daily and weekly patterns.

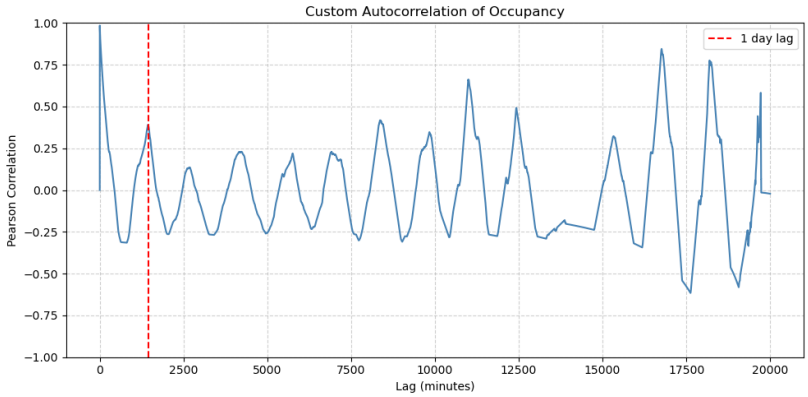
2.2 Correlation structure and key signals

Exploratory analysis guided our feature selection and model design (Figure 2). The correlation matrix (Figure 2a) shows a near-perfect linear relationship between Humidity and Humidity Ratio, prompting us to drop the former to avoid multicollinearity. Meanwhile, Light and CO2 showed the strongest association with Occupancy, identifying them as primary predictors.

Furthermore, the data exhibits strong temporal dependencies (Figure 2b), confirming that readings are not independent snapshots. This evidence validates the use of a forecasting framework with sliding window lags (t-60 to t-30) to capture inertial dynamics, rather than treating the problem as static classification.



(a) Correlation Matrix



(b) Time-Series Evidence

Figure 2: (a) High correlation between Humidity and HumidityRatio supports feature selection. (b) Temporal structure confirms the need for time-series modeling.

3 Methods

3.1 Problem setup and evaluation metric

We forecast occupancy y_t using features X_{t-30} to account for a 30-minute HVAC reaction time. The primary evaluation metric is the F1-score, selected to balance the trade-off between false negatives (occupant discomfort) and false positives (energy waste).

3.2 Feature engineering

Data from three recording periods were concatenated, and samples crossing temporal gaps were strictly removed to ensure integrity. For each prediction at time t, we extracted features from the contiguous [t-60, t-30] interval. These include window statistics (mean, std) and strided sensor lags sampled every 5 minutes (lags at 30, 35, ..., 60) to capture trends without excessive dimensionality. Time features (hour, day) were added, resulting in a compact set of approximately 38 features.

3.3 Data splitting and cross-validation

To prevent data leakage, we utilized a strict chronological split: the first 80% for training/validation and the final 20% for testing. Hyperparameter tuning employed a 5-fold TimeSeriesSplit. All features were standardized using StandardScaler within a pipeline to ensure statistics were derived solely from training folds.

3.4 Models and hyperparameter tuning

We evaluated a Baseline classifier (constant "Occupied") against Logistic Regression, KNN, SVM, Random Forest, and XGBoost. Hyperparameters were tuned via GridSearchCV, with the exception of XGBoost, which utilized a manual loop with Early Stopping (patience=50) to dynamically optimize the number of estimators. Table 1 details the specific configurations.

Table 1: Summary of machine learning models and hyperparameter grids tuned via cross-validation.

Model	Hyperparameter Grid / Settings
Baseline	strategy='constant', constant=1 (Always predicts Occupied)
Logistic Regression	$C \in \{0.001, 0.01, 0.1, 1, 10, 50, 100\}$ Penalty $\in$ {L1, L2, Elastic-Net} L1 Ratio (Elastic-Net) $\in \{0.2, 0.5, 0.8\}$
K-Nearest Neighbors	$n\_neighbors \in \{5, 30\}$ weights='distance', $p = 2$ (Euclidean)
SVM (RBF Kernel)	$C \in \{1, 10, 100\}$, γ = 'scale'
Random Forest	$n\_estimators \in \{200, 300\}$ $max\_depth \in \{20, 30, \text{None}\}$ $min\_samples\_split \in \{2, 5\}$, $min\_samples\_leaf \in \{1, 2\}$
XGBoost	$max\_depth \in \{3, 5, 7\}$ $learning\_rate \in \{0.01, 0.05, 0.1\}$ $subsample = 0.8$, $colsample\_bytree = 0.8$ *Trees ($n\_estimators$) determined dynamically via Early Stopping (max=1000, patience=50).

3.5 Handling class imbalance

Class imbalance was addressed by applying class weights (e.g., class_weight='balanced' for Random Forest, scale_pos_weight for XGBoost) to penalize misclassification of the minority occupied class

more heavily.

3.6 Uncertainty quantification

We quantified uncertainty using two approaches. Splitting Uncertainty (across CV folds) yielded a mean F1 of 0.6768 with a high standard deviation of 0.3333 for XGBoost, indicating significant non-stationarity in occupancy patterns. In contrast, Randomness Uncertainty (varying seeds) was negligible (0.7622 ± 0.0288). This confirms that the primary challenge stems from the temporal dynamics of the environment rather than algorithmic instability.

4 Results

4.1 Baseline and overall model performance

The baseline model, which always predicts "Occupied", achieved a Recall of 1.0 but a low Precision of 0.2691, resulting in an F1-score of 0.4241. This sets a clear benchmark, indicating that useful models must significantly exceed this F1 score. Table 2 summarizes the final performance on the held-out test set. XGBoost and Random Forest were the top performers, both achieving F1-scores above 0.93.

Table 2: Test performance of all models on the held-out 20% time period.

Model	Precision	Recall	F1-score	PR–AUC
XGBoost	0.9158	0.9617	0.9382	0.9362
Random Forest	0.9132	0.9590	0.9355	0.9541
KNN	0.8982	0.9253	0.9115	0.8926
LogReg (L2 / Ridge)	0.8456	0.9389	0.8898	0.8786
LogReg (ElasticNet)	0.8434	0.9380	0.8882	0.8781
LogReg (L1 / Lasso)	0.8293	0.9344	0.8787	0.8686
SVM (RBF)	0.7678	0.9435	0.8466	0.9442
Baseline	0.2691	1.0000	0.4241	0.2691

4.2 Cross-validation vs test performance

Figure 3 compares the Cross-Validation (CV) distribution against the final Test F1 score. The results show a "Test > Validation" phenomenon, where the final test scores are generally higher than the average CV scores. This is likely due to the non-stationarity of the data, as specific time periods contained in some CV folds were significantly harder to predict or contained concept drift, resulting in high standard deviations for CV. However, the models generalized well to the final test period. XGBoost was selected as the deployment model due to its optimal balance of F1-score and its superior Recall of 0.9617, which minimizes the risk of false negatives.

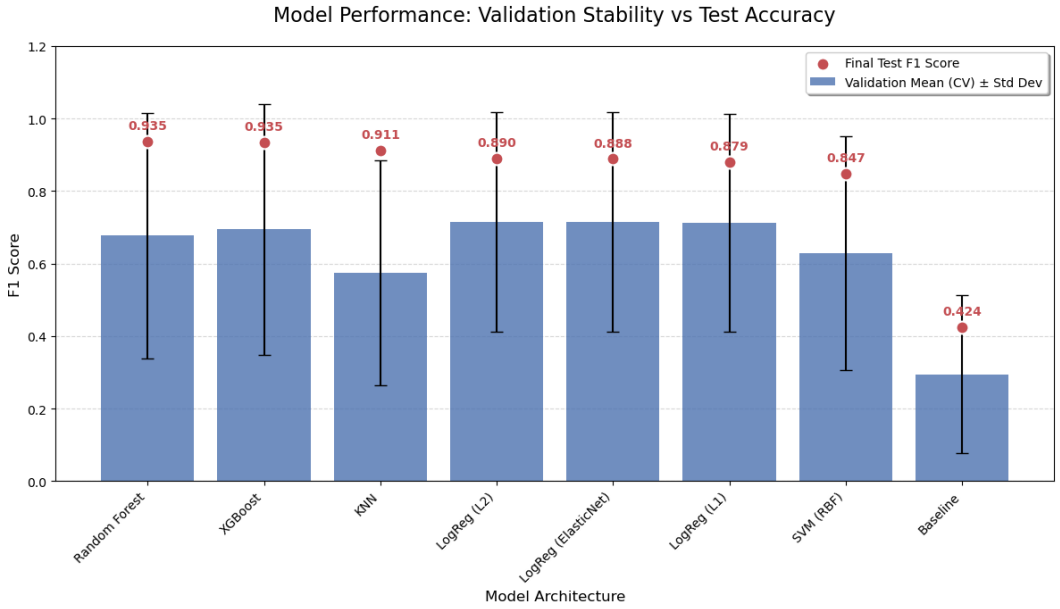


Figure 3: Average cross-validation F1 (bars with error bars) and test F1 (dots).

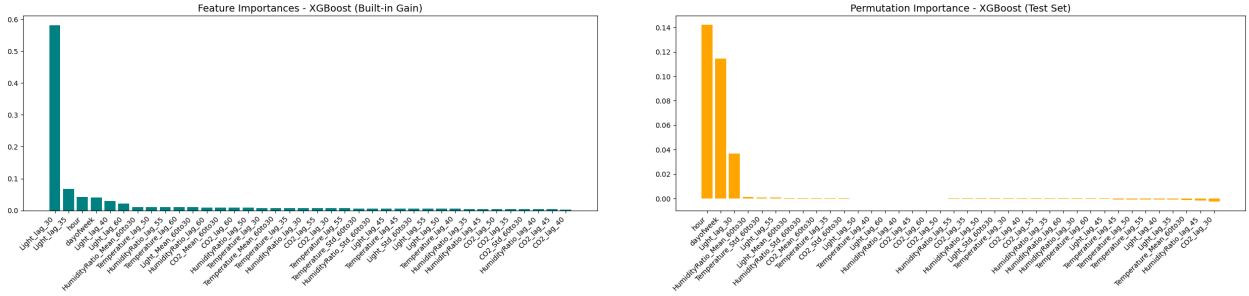
4.3 Uncertainty vs baseline

Our uncertainty quantification revealed that the variation due to time splitting was substantial, with a standard deviation of approximately 0.3447 for XGBoost. This reflects the high non-stationarity of room usage patterns. In contrast, Randomness Uncertainty from varying seeds was negligible (≈ 0.03).

Quantitatively, the XGBoost model's mean cross-validation F1-score (0.6938) is approximately 1.16 standard deviations above the baseline's mean performance (0.2945), considering the high variance inherent in the time-series splits. While the variance is large, the consistent superiority on the held-out test set confirms the model's predictive value.

4.4 Global interpretability

To robustly identify critical signals, we compared three global metrics: XGBoost's built-in gain, Permutation Importance, and Mean Absolute SHAP values. As shown in Figure 4 and Figure 5, these metrics exhibit strong consensus. Light-based features (specifically lags at t-30 and t-35) overwhelmingly dominate, confirming light usage as the primary proxy for presence. CO2 statistics and Time-of-Day serve as secondary predictors. Notably, Temperature and Humidity rank consistently low, diverging from immediate detection literature. This suggests that while thermal variables define current comfort, their high thermal inertia limits their utility as leading indicators for a 30-minute forecast horizon.



(a) XGBoost Built-in Gain

(b) Permutation Importance

Figure 4: Comparison of model-intrinsic (Gain) and model-agnostic (Permutation) feature importance metrics. Both identify Light as the dominant signal.

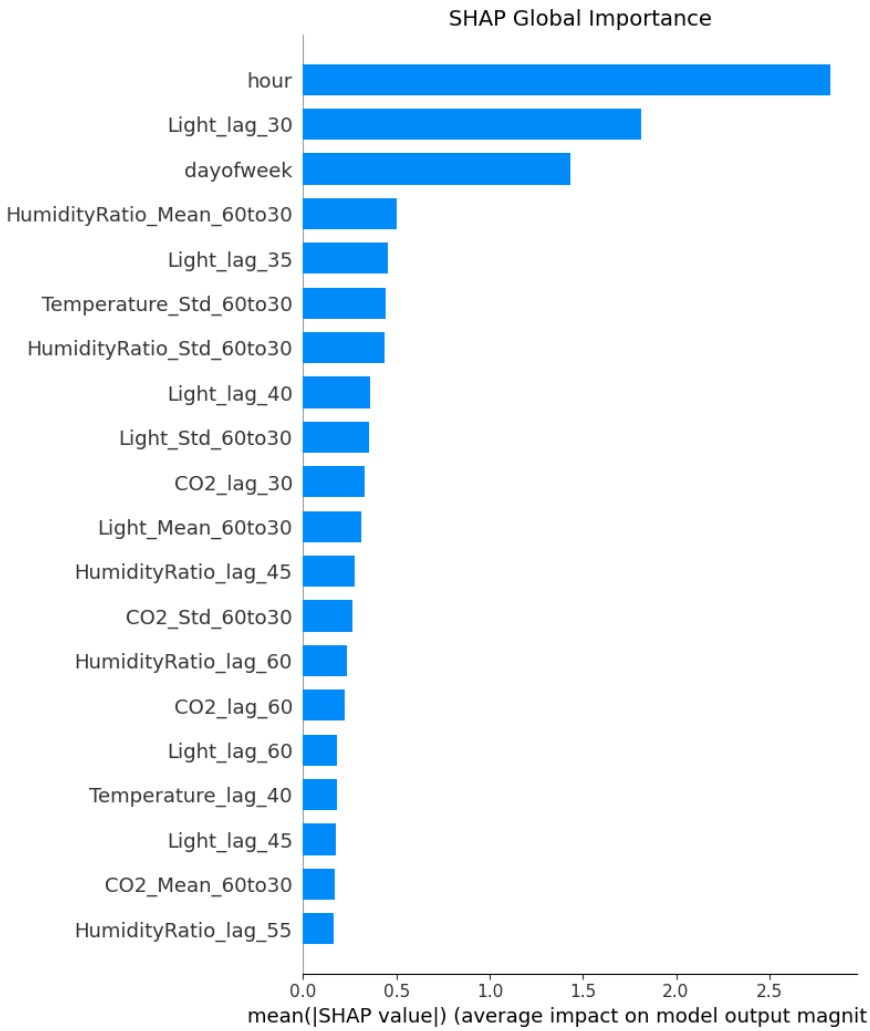


Figure 5: Mean Absolute SHAP Value (Global Importance). This metric aligns with the others, showing the magnitude of impact each feature has on the model output.

4.5 Local interpretability with SHAP

SHAP analysis elucidates feature influence mechanics. The Beeswarm plot (Figure 6) confirms directionality: high Light values strongly drive the model toward "Occupied" predictions, with elevated CO2 acting as a secondary indicator. Locally, the Waterfall plot (Figure 7) decomposes a specific positive instance, demonstrating that Light_lag_30 contributes the largest additive force. This confirms the model leverages physically valid relationships—specifically the link between lighting and human presence—rather than spurious correlations.

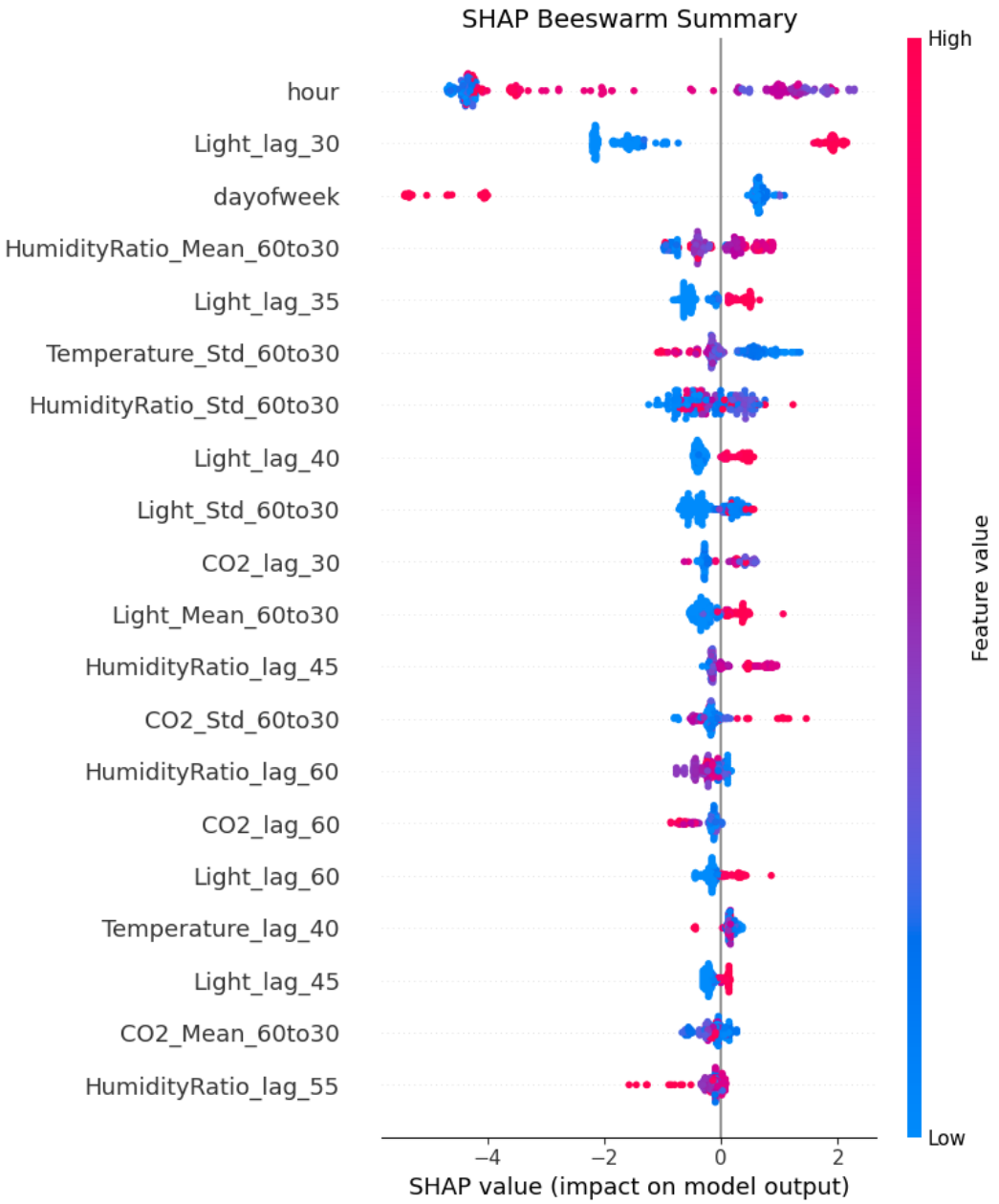


Figure 6: SHAP Beeswarm Summary. Global feature directionality: Red dots (high feature value) on the right indicate positive contribution to occupancy probability.

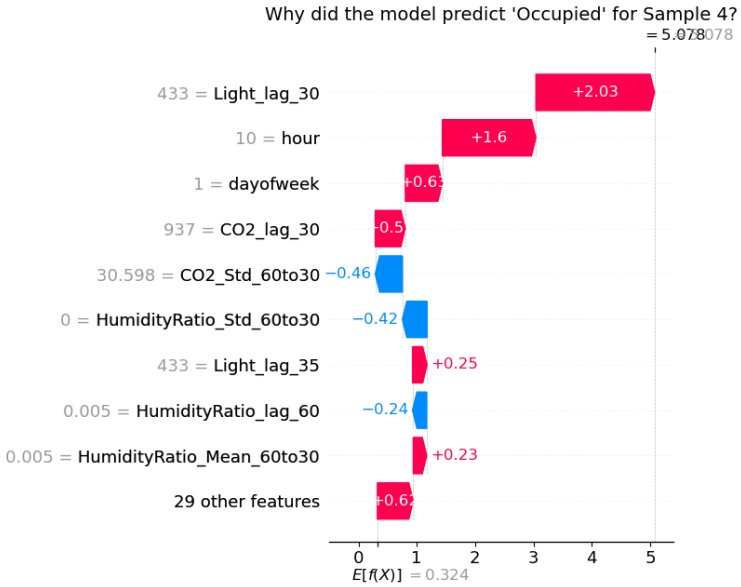


Figure 7: Local Waterfall Plot (Occupied Sample). Light features push the probability up for a specific positive sample, validating the model's logic.

4.6 Model choice

I recommend deploying the XGBoost model. While Random Forest achieved a comparable Test F1-score (0.9355 vs 0.9382), XGBoost demonstrated superior performance across multiple metrics. Notably, it achieved a higher mean cross-validation F1-score (0.6938) compared to Random Forest (0.6774), indicating stronger generalization capability on the historical training splits. Furthermore, XGBoost offered a slightly higher Recall. In an HVAC control context, missing an occupant (False Negative) is more detrimental to user experience than slightly over-conditioning the room. Combined with the robustness provided by the early stopping procedure, XGBoost represents the most reliable candidate for deployment.

5 Outlook

Despite strong performance, the current approach has limitations. First, reliance on single-room data limits spatial transferability, preventing generalization to other buildings without retraining. Second, the high cross-validation variance ($\sigma \approx 0.34$) indicates instability due to non-stationary occupancy patterns. Third, the rigid [t-60, t-30] feature window likely misses complex, longer-term temporal behaviors beyond the one-hour horizon.

Future work could address these issues by integrating external calendars (e.g., Google Calendar) to reduce uncertainty during transition periods. Furthermore, adopting deep sequence models like LSTMs or Transformers could capture long-term dependencies better than simple windowed lags. Finally, dynamic thresholding techniques could be implemented to adaptively maximize Recall during working hours while preserving Precision during off-hours.

References

[1] L. Candanedo and V. Feldheim. Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models. Energy and Buildings, 112, 2016.

[2] UCI Machine Learning Repository. Occupancy Detection Data Set. Available at: `http://archive.ics.uci.edu/dataset/357/occupancy+detection`.