Department of Decision Science

Faculty of Business

University of Moratuwa

Semester 04

DA2111 - Statistical and Machine Learning

Group Assignment

Group 10

.

Table of Contents

Data Set Selection

For our assignment, we selected the **Room Occupancy Estimation** dataset, which addresses the problem of estimating the number of occupants in a room using data from non-intrusive environmental sensors. This dataset contains **10,129 records** and a mix of numerical and categorical variables, meeting the criteria for our project.

Collected over four days in a controlled 6m x 4.6m room, the dataset features readings from seven sensor nodes measuring temperature, light, sound, CO2, and PIR (motion detection). The target variable, room occupancy count, ranges from 0 to 3 people and was recorded manually, providing reliable ground truth data.

The dataset's multivariate and time-series nature makes it ideal for classification tasks, allowing us to apply concepts from the **Statistical and Machine Learning** module and explore advanced techniques in Python to optimize performance.

Dataset Repository Link:
https://archive.ics.uci.edu/dataset/864/room+occupancy+estimation

Problem Definition

1. Target Variable

The target variable for this dataset is Room_Occupancy_Count, which represents the number of people present in the room. This is a **categorical variable** with values ranging from **0 to 3**. The features, such as temperature, light, sound, CO2 levels, and motion detection (PIR), will serve as the predictors to estimate this count.

2. Business Problem/Research Question

The research question we aim to address is:
"Can we accurately predict the number of occupants in a room using environmental sensor data, enabling more efficient space utilization and energy management?"

This problem is relevant for applications in smart buildings, where occupancy estimation can help optimize HVAC systems, lighting, and other energy-consuming devices, leading to improved sustainability and operational efficiency. By developing a reliable prediction model, this project aims to contribute to advancements in smart sensor systems and intelligent building management.

Data Exploration and Preprocessing

Dataset Structure and Characteristics

- **Number of Instances**: 10,129
- **Number of Features**: 19
- **Feature Types**:
    - **Numerical**: Temperature (S1_Temp to S4_Temp), Light (S1_Light to S4_Light), Sound (S1_Sound to S4_Sound), CO2 (S5_CO2), CO2 Slope (S5_CO2_Slope).
    - **Categorical**: PIR Sensors (S6_PIR, S7_PIR), Time of Day, and Room_Occupancy_Count (Target).
    - **Date-Time**: Date, Time, Date_time.

Summary Statistics

- Descriptive statistics showed varying ranges for numerical features. For example:
    - **Temperature**: Ranged between ~24.44°C and ~29°C across sensors.
    - **Light**: Ranged from 0 to 280 Lux.
    - **Sound**: Analog voltage outputs ranged between ~0.04V and ~3.88V.
    - **CO2 Levels**: Measured between 345 ppm and 1270 ppm.
- Boxplots revealed the presence of outliers in most numerical features, such as sound and CO2 levels.

Relationships and Distributions

- Correlation analysis (via heatmap) revealed high correlations among some temperature and light sensors, necessitating removal of redundant features.

- Room occupancy counts were imbalanced, with most records corresponding to 0 occupants.

Date and Time Analysis

- Seasonal analysis revealed data collection spanned seven unique days.
- Grouping by **Time of Day** (Morning, Afternoon, Evening, and Night) revealed differences in room occupancy distribution, visualized through bar plots.

Class Distribution

- Room occupancy distribution was skewed:
  - **0 Occupants**: Majority class.
  - **1 to 3 Occupants**: Minority classes, requiring balancing during preprocessing.

Missing Values and Duplicates

- No missing values were found across the dataset.
- No duplicate rows were identified.

Feature Engineering

- Created a Date_time column by merging Date and Time. Extracted **Time of Day** (Morning, Afternoon, etc.) for temporal analysis.

Feature Selection

- Removed highly correlated features (S1_Temp, S3_Temp) to minimize redundancy and improve model interpretability.

Handling Categorical Variables

- Encoded Time_of_Day using **Label Encoding** for compatibility with machine learning models.

Scaling Numerical Features

- Standardized numerical features to ensure all variables had comparable scales, improving the performance of models sensitive to feature magnitudes.

Exploratory Data Analysis (EDA)

Objectives

The main goals of EDA were:

1. **Visualizing Data Distributions**: Analyze the distributions of key features to identify patterns and trends.
2. **Spotting Outliers**: Detect anomalies or extreme values in numerical features that could impact model performance.
3. **Assessing Correlations**: Explore relationships between features to understand their interdependencies and identify redundant variables.

Visualization Techniques and Insights

Histograms for Data Distributions

Histograms were plotted for numerical features like **Temperature**, **Light**, **Sound**, and **CO2 levels**.

- **Insights**:
    - Many features, such as temperature and light intensity, exhibit near-normal or unimodal distributions, while others may show skewness..
    - Some features (eg: S1_Light and S5_C02) show extreme values, indicating possible outliers that could impact modeling.

Box Plots to Detect Outliers

Box plots were used to identify outliers for each numerical feature.

- **Insights**:
  - **Sound Levels**: Significant outliers were observed, particularly at the higher voltage ranges.
  - **CO2 Levels**: Some extreme values were identified, suggesting potential calibration issues or sudden environmental changes.

Scatter Plots and Heatmaps for Correlations

- **Scatter Plots**: Relationships between features like **Light** and **Sound** were explored.
  - Example: Higher light levels corresponded to lower sound in some cases, indicating specific environmental conditions.
- **Heatmap**:
  A correlation heatmap was generated for numerical features.
  - **High Correlations**: Observed among temperature sensors (S1_Temp, S3_Temp), and light sensors (S1_Light, S2_Light).
  - **Low Correlations**: Detected between sound and CO2 levels, suggesting independence.

Group-Based Aggregation for Trends in Categorical Data

- **Pivot Table**: The average occupancy was analyzed across **Time of Day** (Morning, Afternoon, Evening, Night).
  - **Insights**:
    - Occupancy was highest in the afternoon, while nights consistently showed lower counts.
    - Aggregation helped identify time-based trends in room usage, aiding feature engineering.

Bar Plot: Count of Room Occupancy by Time of Day

- **Insights**:
    - **0 Occupants**: The 0 occupants category dominates across all times of day, especially during **Night**.
    - **1 to 3 Occupants**: Higher occupancy counts (1, 2 and 3) are most common in the **Afternoon** and **Evening**, reflecting active periods during the day.
    - **Night**: Very few instances of 3 occupants in the **Night**, indicating less activity in late hours.

Line Plot: Average Room Occupancy Over Time

- **Insights**:
    - **Afternoon Peaks**: The highest average occupancy is observed in the **Afternoon**, suggesting peak usage during this time.
    - **Evening**: Moderate occupancy persists in the **Evening**.
    - **Morning and Night**: Both **Morning** and **Night** have significantly lower occupancy, indicating fewer people are present during these hours.
    - **Fluctuations**: Some fluctuations in occupancy, particularly in the **Morning**, could indicate specific events or patterns.

Pie Chart: Percentage Distribution of Room Occupancy

- **Insights**:
    - **Imbalanced Distribution**: The 0 occupants category represents the majority of the dataset, highlighting the need for techniques like **SMOTE** to address class imbalance.
    - **Lower Occupancy Classes**: Occupancy levels of 1,2 and 3 occupants represent a much smaller proportion of the data, making them minority classes.

Dealing with Class Imbalance

Problem Context

The dataset exhibited a significant class imbalance, with the majority of instances belonging to the 0 occupants category, while higher occupancy levels (1, 2, 3 occupants) were underrepresented. This imbalance posed a challenge for machine learning models, as they tend to favor the majority class during training, **which can lead to poor performance on minority classes.**

Approach to Handle Imbalance

To address this issue, the **Synthetic Minority Oversampling Technique (SMOTE)** was applied. SMOTE is an oversampling technique that generates synthetic samples for the minority classes by interpolating between existing samples. This helps balance the class distribution without simply duplicating data, reducing the risk of overfitting.

Impact of SMOTE

Before applying SMOTE:

- 0 occupants: ~81% of the dataset.
- 1, 2, 3 occupants: Combined ~19%, with 3 occupants being the smallest category.

After applying SMOTE:

- All classes (1, 2, 3 occupants) were evenly represented in the training set, ensuring a balanced dataset for model training.

Model Selection

Algorithm Choice and Justification

Why Classification Was Selected

The problem at hand was **predicting the number of occupants in a room based on environmental sensor data**. It is inherently a **classification task** because:

- **Target Variable**: The Room_Occupancy_Count variable is categorical, representing discrete classes (0, 1 ,2 ,3 occupants).

- **Goal**: The objective is to assign each instance to one of these predefined categories based on the feature values (e.g., temperature, light, sound, CO2 levels, and PIR motion detection).

Characteristics of the Data

The dataset's characteristics influenced the selection of machine learning algorithms:

- **Multivariate Nature**: The data contains a mix of numerical features (eg: temperature, light, sound, CO2 levels) and binary features (eg: PIR motion detection). Algorithms like Random Forest and Logistic Regression handle such mixed data types effectively.

- **Imbalanced Classes**: The majority of instances belong to the 0 occupants class, which poses challenges for model performance. Random Forest can handle class imbalance better by leveraging ensemble techniques, while Logistic Regression serves as a robust baseline.

- **Sensor Variability**: Features like sound, light, and CO2 levels are time-dependent and influenced by sensor calibration. Models like Random Forest are less sensitive to feature scaling and can handle such variability better, while Logistic Regression benefits from proper preprocessing (eg: scaling).

Algorithms Selected and Justification

1. **Random Forest Classifier**

- **Robustness with High-Dimensional Data**: Random Forest can handle datasets with multiple features and does not assume linear relationships between predictors and the target variable, making it ideal for capturing complex sensor interactions.

- **Feature Importance**: It provides feature importance scores, helping identify which sensors (eg: light or PIR) contribute most to predicting room occupancy. This is particularly useful in the problem context for sensor optimization in smart buildings.

- **Performance with Non-Linear Data**: As the relationships between environmental variables (eg: CO2 levels and occupancy) are likely non-linear, Random Forest excels at capturing such patterns.

2. **Logistic Regression**

- **Baseline Model**: Logistic Regression is straightforward and interpretable, serving as an excellent benchmark for comparison with more complex models like Random Forest.

- **Linearity**: It assumes linear relationships between features and the target variable, which might hold for some predictors (eg: light levels and occupancy).

- **Computational Efficiency**: Logistic Regression is computationally light, making it a practical choice for large datasets like this one.

- **Probabilistic Output**: Logistic Regression provides probabilities for each class, which can be useful for threshold-based decision-making in smart building systems.

Model Training and Evaluation

Data Splitting

To ensure reliable evaluation and validate model performance, the dataset was split into three subsets:

- **Training Set (60%)**: Used to train the machine learning models after applying SMOTE to address class imbalance.
- **Validation Set (20%)**: Used during training to assess model performance and fine-tune decisions (if necessary).
- **Testing Set (20%)**: Reserved for evaluating the models on unseen data, providing an unbiased assessment of their generalization capability.
- **Stratified Sampling**: The splitting process maintained proportional representation of each class (0, 1 ,2 ,3 occupants) to ensure consistent class distribution across subsets.

Training

Two machine learning models were selected for training:

Random Forest Classifier:

- Trained on the preprocessed and balanced training dataset.
- Leveraged ensemble techniques to improve prediction robustness and handle non-linear feature interactions effectively.

Logistic Regression:

- Trained using standardized features for numerical stability.
- Provided a baseline for comparison, offering simplicity and interpretability while handling linear relationships between features and the target variable.

Evaluation Metrics

The models were evaluated on the test set using key classification metrics:

- **Accuracy**: Percentage of correctly predicted instances across all classes.
- **Precision**: Assesses the proportion of true positives among predicted positives for each class.
- **Recall**: Evaluates the model's ability to identify all true positives, especially critical for minority classes.
- **F1-Score**: Balances precision and recall into a single metric, ideal for imbalanced datasets.
- **ROC-AUC**: Measures the model's ability to distinguish between classes, providing an overall sense of its discriminative power.
- **Confusion Matrix**: Detailed the performance of the models for each class, showing true positives, true negatives, false positives, and false negatives.

Hyperparameter Tuning

Objective

The objective of hyperparameter tuning was to optimize the **Random Forest Classifier** by fine-tuning its hyperparameters to improve model performance on the classification task.

While Logistic Regression is less dependent on hyperparameter adjustments, Random Forest benefits significantly from tuning parameters such as the number of trees (n_estimators) and tree depth (max_depth).

Methodology

Grid Search was employed to systematically explore combinations of hyperparameters for Random Forest. The following parameters were tuned:

- **n_estimators**: The number of decision trees in the forest.
- **max_depth**: The maximum depth of each tree, controlling overfitting.
- **min_samples_split**: The minimum number of samples required to split a node.
- **min_samples_leaf**: The minimum number of samples required to form a leaf node.

Parameter Changes and Effects

- **n_estimators**: Increased from the default (100) to 150. This improved performance by reducing variance in predictions through additional decision trees.
- **max_depth**: Optimized to 15, balancing the complexity of the model and avoiding overfitting.
- **min_samples_split**: Set to 4 to ensure nodes are split only when necessary, preventing overly complex trees.
- **min_samples_leaf**: Adjusted to 2, ensuring that leaf nodes are not too small, improving generalization.

Comparison and Conclusion

Performance Comparison

Metric	Random Forest (Default)	Random Forest (Tuned)
Accuracy	99.90%	99.93%
Precision	99.90%	99.94%
Recall	99.90%	99.93%
F1-Score	99.90%	99.93%
ROC-AUC	99.92%	99.95%

Insights

- **Improved Metrics**: The tuned model achieved slightly better accuracy, precision, recall, and F1-scores, demonstrating enhanced predictive performance.

- **Generalization**: The optimized parameters improved the model's ability to handle unseen data while maintaining robust performance across all classes.

- **Balance Between Overfitting and Underfitting**: Tuning effectively balanced model complexity, reducing overfitting without sacrificing accuracy.

Strengths and Weaknesses

Random Forest Classifier:

Strengths:

- Outperformed Logistic Regression in all metrics, particularly for minority classes, due to its ability to capture complex relationships and non-linear patterns.
- Provided feature importance insights, aiding interpretability and feature selection.
- Robust to outliers and noise in the dataset.

Weaknesses:

- Computationally expensive compared to Logistic Regression, especially during hyperparameter tuning.
- Less interpretable compared to simpler models.

Logistic Regression:

Strengths:

- Simple and interpretable, providing a clear understanding of how features influence predictions.
- Computationally efficient and fast to train, even on large datasets.

Weaknesses:

- Struggled with minority class predictions, as evidenced by lower recall and F1-scores for underrepresented occupancy levels.
- Assumes linear relationships between features and the target variable, limiting its ability to capture complex patterns.

Challenges and Solutions

Class Imbalance:

Challenge: The majority of instances belonged to the 0 occupants class, leading to biased predictions.

Solution: Applied **SMOTE** to balance the training dataset, which significantly improved recall and F1-scores for minority classes.

Feature Correlations:

Challenge: Strong correlations among features, such as temperature and light sensors, risked redundancy and overfitting.

Solution: Performed feature selection by removing highly correlated variables, simplifying the models without compromising accuracy.

Computational Complexity:

Challenge: Random Forest's training and tuning were computationally intensive.

Solution: Optimized hyperparameters with a limited grid search and reduced dataset size for faster evaluation during development

Concluding Remarks

Model Effectiveness

Both models effectively addressed the room occupancy classification problem, achieving high accuracy and reliability. The **Random Forest Classifier**, with its superior performance across all metrics, proved to be the best-suited model for this task.

Logistic Regression, though slightly less effective, served as a robust and interpretable baseline.

Real-World Application

The models demonstrate strong potential for real-world applications in smart buildings, where accurate occupancy estimation can optimize energy management systems.

Recommendations for Future Work:

- **Longer Data Collection Periods**: Expanding the dataset to cover more days and diverse conditions would improve model robustness.

- **Additional Features**: Incorporating new environmental features, such as humidity or motion intensity, could further enhance model predictions.

- **Model Optimization**: Experimenting with advanced models like Gradient Boosting or Neural Networks may yield additional performance gains.

- **Deployment**: Building a real-time implementation pipeline using the trained models would showcase their effectiveness in live settings

Details of Infrastructure:

- Laptop: MacBook Air
- Chip: Apple M1
- GPU: Apple M1
- Memory: 8GB