National College of Ireland

Enhancing Occupancy Estimation with Environmental Sensor Data: A Deep Learning Approach Using TabNet

MSc Research Project
Data Analytics

Abhishek Kumar Tiwari
Student ID: X211772277

School of Computing
National College of Ireland

Supervisor: Prashanth Nayak

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Abhishek Kumar Tiwari
Student ID:	X211772277
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Prashanth Nayak
Submission Due Date:	14/08/2023
Project Title:	Enhancing Occupancy Estimation with Environmental Sensor Data: A Deep Learning Approach Using TabNet
Word Count:	6752
Page Count:	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	**Abhishek Kumar Tiwari**
Date:	14th August 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing Occupancy Estimation with Environmental Sensor Data: A Deep Learning Approach Using TabNet

Abhishek Kumar Tiwari
X211772277

Abstract

This study presents a comprehensive investigation into the application of machine learning for improving occupancy estimation using environmental sensor data. Occupancy estimation has become a pivotal aspect of several domains, including energy management, building automation, and security. However, achieving accurate and efficient occupancy estimation presents significant challenges due to the dynamic nature of occupancy and the complexity of sensor data interpretation. In this research, we employ the TabNet deep learning model, specifically designed for tabular data, with environmental factors such as temperature, humidity, pressure, and altitude as inputs. Our approach involves extensive data preprocessing, feature extraction, and implementation of a robust training methodology, including a stratified k-fold cross-validation process. The report includes a detailed evaluation of the model's performance and discusses areas for future improvement and exploration. This research contributes to ongoing efforts in the field of occupancy estimation and demonstrates the potential of machine learning techniques for this application. The study also identifies potential areas for future work aimed at enhancing the model's performance and broadening its applicability.

1 Introduction

This research report presents a comprehensive study conducted to improve occupancy estimation using environmental sensor data. The need for accurate occupancy estimation has grown significantly due to its potential applications in various domains, such as energy management, building automation, security, and personalized services. However, accurately estimating the number of people in a building or a specific area within a building remains a challenging task. This is primarily due to the dynamic nature of occupancy and the complexity of interpreting sensor data.

Previous work in this field has largely focused on leveraging machine learning techniques for occupancy estimation. Researchers such as Pal, Yang, and Dey have explored the importance of feature extraction, data preprocessing, and the use of various machine learning algorithms for occupancy estimation. However, there still remains a considerable scope for improvement, particularly in terms of accuracy and efficiency.

In this study, we aim to build upon this body of work and develop a more accurate and efficient model for occupancy estimation. Our model is based on the TabNet deep learning algorithm, which is specifically designed for tabular data. We have chosen this

model due to its compatibility with the structured nature of our dataset, which includes factors such as temperature, humidity, pressure, and altitude.

The report is organized into several sections. The 'Related Work' section provides a critical review of previous studies in this domain, discussing their strengths, weaknesses, and key contributions. The 'Methodology' and 'Implementation' sections detail our approach to data preprocessing, feature extraction, and model training. The 'Evaluation' section presents the performance of our model, including its overall accuracy and its performance at different occupancy levels. Finally, the 'Conclusion' section summarizes our findings, discusses the implications of our work, and suggests potential directions for future research.

By presenting this research, we hope to contribute to the ongoing efforts to improve occupancy estimation using environmental sensor data. We believe that our work represents a significant step forward in this field and will provide a valuable reference for future studies.

2 Related Work

Occupancy estimation in buildings is an area of growing interest, given its implications for energy management, security, and smart building automation. Achieving accurate occupancy estimation has been the subject of numerous studies, with researchers exploring various methodologies and techniques to enhance accuracy and reliability.

2.1 Pal et al.

The study (1) delves into the role of machine learning in estimating occupancy for IoT applications. Key takeaways from their research include:

- **Feature Extraction:** Emphasized extracting meaningful patterns from sensor data, proposing features like mean, variance, skewness, and kurtosis to reflect occupancy-related characteristics.
- **Data Preprocessing:** Advocated for data quality through cleaning, normalization, and managing missing values, aiming to optimize data for machine learning algorithms.
- **Machine Learning Algorithms:** Explored algorithms including SVM, Random Forests, and ANN, training models on sensor-derived features to predict occupancy.
- **Evaluation Metrics:** Utilized metrics like accuracy, precision, recall, and F1-score, gauging model efficacy in predicting occupancy.
- **Experimental Results:** Demonstrated the model's high accuracy in occupancy estimation, highlighting the robustness of their methodology.

Overall, Pal et al.'s work (1) offers insights into machine learning techniques for IoT-based occupancy estimation.

2.2 Yang et al.

Yang et al. delve into the use of machine learning for occupancy estimation(2). Key insights from the paper include:

- **Machine Learning Algorithms:** Focus on algorithms like SVM, Random Forests, ANN, and HMM, emphasizing the processing of sensor data to determine occupancy patterns.
- **Feature Selection:** Underscored the importance of selecting relevant features, considering potential features like motion patterns, light intensity, and CO2 levels. They also highlighted the significance of feature engineering for model performance.
- **Model Training and Evaluation:** Covered model training on separate training data and evaluation on testing data, likely employing metrics like accuracy, precision, recall, and F1-score.
- **Data Preprocessing:** Addressed the need for data quality improvements, including handling missing data and outliers, with potential steps like data cleaning, normalization, and managing missing values.
- **Model Optimization and Generalization:** Discussed enhancing model performance and generalizability, considering techniques like hyperparameter tuning, cross-validation, and regularization.
- **Comparison with Other Approaches:** Compared machine learning-based models to other methods like rule-based or statistical models, evaluating pros and cons in terms of accuracy, robustness, and computational efficiency.

Overall, the literature review would focus on the application of machine learning algorithms, feature selection, model training and evaluation, data preprocessing, model optimization, and the comparison of the proposed approach with other occupancy estimation methods(2).

2.3 Dey et al.

Dey et al. delve into inferring occupancy in buildings using machine learning(3), emphasizing its significance in energy management, security, and personalized services. Key insights from their research include:

- **Sensor Data Collection:** They discussed collecting data from various building sensors, including motion, door, light, and temperature sensors. This data serves as the primary input for the machine learning model.
- **Feature Extraction:** Highlighted the importance of extracting meaningful patterns from sensor data, considering potential features like motion patterns, occupancy duration, and temporal patterns as model inputs.
- **Machine Learning Algorithms:** Likely delved into algorithms such as SVM, Random Forests, HMM, and ANN, focusing on training these on extracted features to predict occupancy.

- **Model Training and Evaluation:** They detailed dividing the dataset into training and testing subsets and likely emphasized metrics like accuracy, precision, recall, and F1-score for evaluation.

- **Data Preprocessing:** Addressed improving data quality through methods like cleaning, normalization, outlier removal, and feature scaling.

- **Comparison and Validation:** Explored a comparison of their machine learning approach with rule-based or statistical methods, focusing on factors like accuracy, robustness, scalability, and efficiency. They also discussed validating their methods against ground truth data or manual counts.

In summary, the literature review for Dey et al.'s paper would likely cover topics like occupancy inference, sensor data collection, feature extraction, machine learning algorithms, model training, data preprocessing, and comparison/validation with other methods(3).

2.4 Vela et al.

Vela et al. investigate the potential of environmental variables, such as humidity, temperature, and pressure, in estimating occupancy within enclosed spaces (4). Their approach capitalizes on the rising trend of indirect methods for occupancy estimation. Key insights from their paper include:

- **Methodology:** The research provides an in-depth look into the data collection of environmental variables from different spaces. They emphasize preprocessing this data to enhance its suitability for machine learning models and then detail the model development and evaluation process, ensuring a systematic approach to occupancy estimation.

- **Machine Learning Algorithms:** Although the specific algorithms aren't directly listed, there's a mention of the k-Nearest Neighbors (kNN) algorithm. The use of kNN indicates their approach to modeling the relationship between environmental variables and occupancy.

- **Results and Discussion:** Vela et al. analyze their model's performance across different settings, such as gyms and living rooms, contrasting their results with existing literature. Their datasets have been made publicly available, supporting standardized research comparisons.

- **Conclusions and Further Work:** The study concludes with a reflection on their achievements, especially noting the high accuracy attained using the kNN algorithm. They also hint at ongoing research directions and potential areas of improvement in occupancy estimation.

To sum it up, Vela et al.'s study (4) offers a holistic perspective on using environmental data in tandem with machine learning for precise occupancy estimation in enclosed environments.

2.5 Masood et al.

Masood et al. explore the realm of real-time occupancy estimation in modern buildings, focusing on the use of environmental parameters(5). They emphasize the critical role of occupancy data in intelligent control decisions, particularly in ACMV system operations. The research's main insights include:

- **Limitations of Traditional Sensing Mechanisms:** Traditional mechanisms, such as cameras and wearable sensors, are not only intrusive but can also be costly. The authors advocate for a departure from these norms, suggesting more efficient alternatives.
- **Leveraging Environmental Parameters:** Masood et al. highlight that occupants' presence affects parameters like CO2, temperature, humidity, and pressure. Monitoring these can provide insights into occupancy levels. Environmental sensors come with the added advantage of being cost-effective and non-intrusive.
- **Feature Extraction and Selection:** The authors delve into the importance of extracting relevant features from sensor data, hinting at possible use of filter model feature selection techniques to pinpoint the most significant environmental indicators.
- **Real-time Estimation:** The research emphasizes the need for immediate occupancy insights, crucial for dynamic energy management and other on-the-spot building operations.
- **Conclusion and Findings:** While the paper's exact findings aren't detailed here, it's clear that Masood and his team propose a novel method challenging conventional occupancy estimation. The focus on environmental changes induced by occupants suggests a promising, non-intrusive, and cost-effective approach to real-time occupancy estimation.

In summary, Masood et al.'s research(5) underscores the potential of environmental parameters as dependable occupancy indicators, likely presenting in-depth results and validations that could be pivotal for contemporary building management.

2.6 Hailemariam et al.

Hailemariam and colleagues focus on real-time occupancy detection using multiple sensors and Decision Trees(6). Key insights from their study include:

- **Feature Extraction and Decision Trees:** The team identified the root mean square error of a passive infrared motion sensor as a significant feature. This alone achieved a 97.9
- **Multiple Sensor Types:** Merging data from varied sensors can lead to overfitting. The addition of sensors like sound and CO2 didn't always enhance results.
- **Decision Trees' Intuitiveness:** Decision Trees offer transparency, with each path providing insights into optimal feature combinations for occupancy detection.

- **Conclusions:** They highlight the challenges of integrating varied sensor data and caution against potential overfitting when using Decision Trees.

Overall, Hailemariam et al. present a nuanced approach to occupancy detection, stressing the value of feature selection, the potential of Decision Trees, and the need for careful sensor data integration(6).

2.7 Chen et al.

Chen and colleagues explore building occupancy estimation using environmental sensors, emphasizing the CDBLSTM approach(7). Key insights from their research include:

- **Methodology:** The team utilized Decision Trees to examine relationships between sensors, features, and occupancy. They viewed occupancy estimation as classification, considering metrics like classification accuracy and NRMSE.

- **Results and Discussions:** Their results highlighted the effectiveness of methods using manually extracted statistical features over raw sensory data. They noted that while using numerous sensors ensured performance, it also raised costs and maintenance demands. Features from passive infrared motion sensors, when processed efficiently, rivaled more intricate methods in accuracy.

- **Conclusions:** Chen and team emphasized the significance of feature extraction and the potential of CDBLSTM. Their findings indicated that while extensive sensor setups provide detailed data, careful feature selection and processing can achieve high accuracy with simpler configurations.

In essence, Chen et al.'s study provides a comprehensive look into the intricacies of occupancy estimation, highlighting the balance between sensor complexity and feature extraction for optimal results(7).

2.8 Candanedo et al.

Candanedo and Feldheim investigate occupancy detection using environmental sensors, focusing on statistical learning models(8). Key insights from their study include:

- **Methodology:** They employed Decision Trees to analyze the relationship between sensors, features, and occupancy. The team viewed occupancy estimation as a classification task, using metrics such as classification accuracy and NRMSE for evaluation.

- **Results and Findings:** When all parameters were used for model training, high accuracies were observed. However, Random Forest models, especially when applied to parameters like temperature (T),humidity (Φ),CO2, and wind speed (W), showed a significant gap between training and test set accuracy. Their analysis also revealed that the error rate in Random Forest models plateaued after about 200 trees, suggesting limited benefits from adding more trees. They also identified the most influential parameters based on the Gini index.

- **Implications and Conclusions:** Candanedo and Feldheim emphasized the challenges of model overfitting and the nuances of using machine learning models like Random Forest for occupancy estimation. Their findings highlighted the importance of balancing sensor data with the right model parameters to achieve accurate occupancy estimation.

In essence, Candanedo and Feldheim provide a detailed exploration of occupancy estimation, underscoring the balance between data complexity and machine learning model parameters(8).

2.9 Agarwal et al.

Agarwal and his team delve into the intricacies of occupancy-driven energy management in smart building automation(9). Their research provides insights into:

- **Sensing Mechanism and Occupancy Detection:** They developed a cost-effective prototype using a reed switch and a PIR sensor to optimize occupancy detection. The system integrates both sensors' data, with an open door being treated as an occupancy indication based on typical office behaviors.
- **Algorithm Refinements:** Differentiating between a person leaving a room and one remaining inside after closing the door was a challenge. The team designed an algorithm leveraging the PIR sensor to handle noise and accurately determine occupancy.
- **Potential Limitations:** While their system may occasionally misinterpret a closed room's occupancy status, the goal is to maximize accuracy with the given sensors and algorithms.
- **Conclusion and Implications:** Agarwal and his team highlight the potential of hybrid sensor systems and stress the importance of efficient algorithm design in smart building automation. Their work also touches on the challenges of achieving high real-time occupancy detection accuracy.

In essence, Agarwal et al.'s research provides a comprehensive exploration of occupancy detection mechanisms and their implications in the smart building automation domain(9).

3 Data Description

3.1 Dataset Overview

This study utilizes a dataset that contains both environmental data (pressure, altitude, humidity, and temperature) and occupancy levels from two distinct locations: a fitness-gym and a living-room. The fitness-gym data was gathered over a period of six days from September 18 to October 02, 2019. Similarly, the living-room data spans 11 days, specifically from May 14 to June 4, 2020. Notably, the living-room data includes information on the number of mechanical ventilators present in that space.

3.2 Data Collection Method

The data was collected using sensors installed in the respective locations. These sensors recorded the environmental information, including pressure, altitude, humidity, and temperature, at regular intervals. The fitness-gym data collection occurred over a six-day period, while the living-room data collection spanned 11 days. The specific details of the data collection setup and sensor deployment are not provided.

3.3 Variables and Features

The dataset consists of the following variables or features:

- date: The date and time of each observation.
- pre: The pressure measurement.
- alt: The altitude measurement.
- hum: The humidity level.
- tem: The temperature reading.
- ven: The number of mechanical ventilators present (only available for living-room data).
- occ_int: The occupancy level encoded as an integer. This serves as the target variable.
- occ: The occupancy level represented as a categorical variable. The values can be 'H' (high occupancy), 'M' (medium occupancy), or 'L' (low occupancy) (only available for living-room data).

The variables pre, alt, hum, and tem are numerical variables, while ven, occ_int, and occ are categorical variables.

3.4 Missing Data Handling

According to the information provided, there are no missing values in the dataset. Each column contains the same number of non-null values as the total number of rows, indicating the absence of missing data.

3.5 Data Quality Assurance

The dataset does not provide explicit information about the steps taken to ensure data quality. However, it can be assumed that measures such as data validation checks, cleaning procedures, and outlier detection methods were employed to maintain data quality and integrity. The absence of outliers in the dataset, as mentioned, suggests that any outliers may have been detected and appropriately addressed.

Table 1: Data Description Statistics

	pre	alt	hum	tem	ven	occ_int
count	295,823.000	295,823.000	295,823.000	295,823.000	295,823.000	295,823.000
mean	94,666.124	569.81224	55.85002	30.31064	0.47417	1.751987
std	371.042	32.66009	5.19232	1.40524	0.58761	0.877814
min	93,735.630	498.26000	40.55000	25.41000	0.00000	0.000000
25%	94,431.235	545.71000	52.71000	29.66000	0.00000	2.000000
50%	94,683.160	568.26000	56.28000	30.59000	0.00000	2.000000
75%	94,939.865	590.44000	59.50000	31.08000	1.00000	2.000000
max	95,481.800	651.94000	72.65000	33.50000	2.00000	3.000000

3.6 Data Description Statistics

The dataset consists of 295,823 rows and 7 columns. The statistical summary of the numerical variables (pre, alt, hum, tem, ven, and occ int) is as follows:

The summary includes the count, mean, standard deviation, minimum, maximum, and quartile values for each numerical variable.

3.7 Data Visualization

Data visualization plays an indispensable role in data analysis and machine learning. It helps to better understand the data at hand, unveil underlying patterns, and communicate findings effectively. In our analysis, we performed several data visualization tasks using Python libraries like Matplotlib and Seaborn.

3.7.1 Histograms

Histograms were used to analyze the distribution of individual numeric variables in our dataset. It gives us a clear picture of data concentration and data spread and helps identify potential outliers. In Fig. 1 is an example of how we visualized the 'pre', 'alt', 'hum', 'tem', and 'occ_int' *columns.*

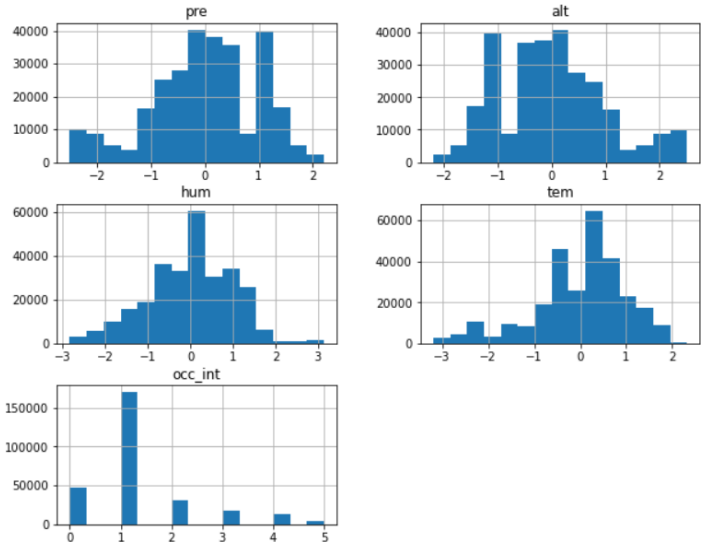

Figure 1: Histograms of independent variables

3.7.2 Correlation Matrix

A heatmap of the correlation matrix was plotted in Fig. 2 to visualize the correlation between different numerical features. High correlation between two variables might indicate a possibility of collinearity which could be taken into account during the model-building process.

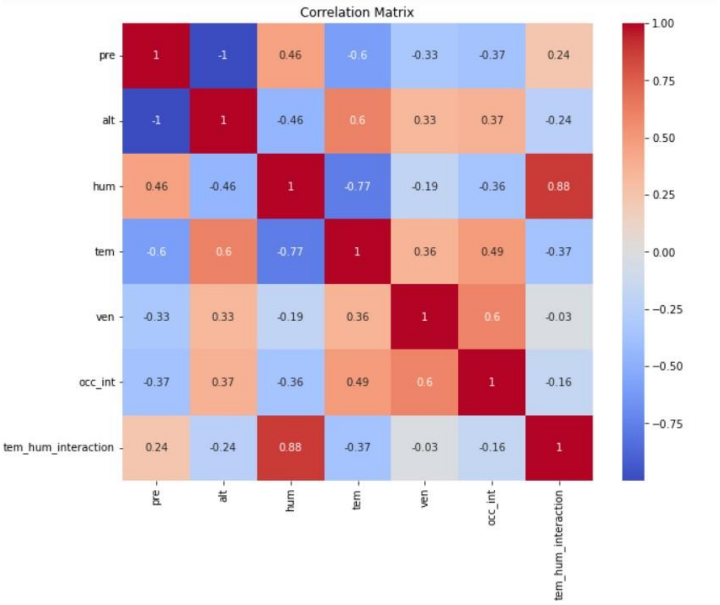

Figure 2: Correlation Matrix

3.7.3 Scatterplot

Scatter plots were used to examine the relationship between two numerical variables. For instance, we plotted scatter graphs for 'Pressure vs Temperature' and 'Humidity vs Temperature' in Fig. 3 and Fig.4 respectively.

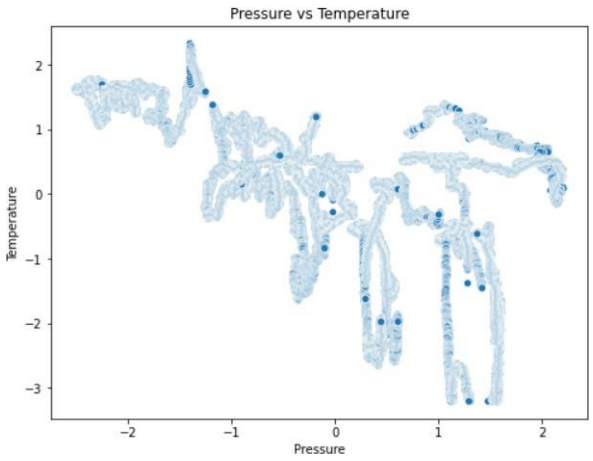

Figure 3: Scatterplot - Pressure vs Temperature

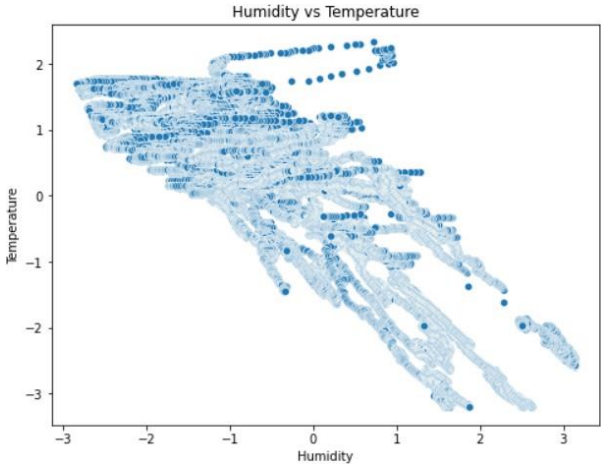

Figure 4: Scatterplot - Humidity vs Temperature

3.8 Data Limitations

The limitations of the dataset include its limited time span and the specific locations from which the data was collected. The findings and conclusions drawn from this dataset may not be generalizable to other time periods or different environments. Furthermore, it is important to note that the absence of missing values and outliers mentioned is based on the provided information, and the actual dataset may have contained missing values or outliers that were handled or addressed during preprocessing or analysis.

3.9 Ethical Considerations

****

4 Methodology

The methodology section of this data analytics study outlines the steps followed to conduct the analysis and build the TabNet model for predicting the target variable 'occ int'. The process involves data collection, data preprocessing, exploratory data ana-lysis (EDA), model building, and evaluation. The detailed methodology is as follows:

4.1 Data Collection

The dataset used in this study is collected from the "Home.csv" file. It contains several features such as pressure (pre), altitude (alt), humidity (hum), temperature (tem), vent-ilation (ven), and the target variable 'occ int', which represents the occupancy levels as integers.

4.2 Data Pre-processing

Data preprocessing is a crucial step in any data analytics project to ensure that the data is in a suitable format for analysis and modeling. In this study, the dataset is preprocessed to address various issues such as handling categorical variables, removing

irrelevant columns, and dealing with missing values. The specific steps taken during data preprocessing are as follows:

4.2.1 Dropping the variable 'occ'

The 'occ' column in the dataset represents the occupancy levels, and it is categorical in nature with three possible categories: 'L' (Low), 'M' (Medium), and 'H' (High). This variable, however, is derived from the target variable itself and doesn't provide additional distinct information to our model. To prevent any potential data leakage and enhance the model's ability to generalize from the remaining independent features, we decided to drop the 'occ' column from our dataset prior to training.

4.2.2 Dropping Irrelevant Column 'date'

The 'date' column in the dataset contains timestamp information for each data point. Since the focus of the analysis is on predicting occupancy levels, the 'date' column does not provide any meaningful information for the model. As a result, it is dropped from the dataset to avoid unnecessary complexity and reduce dimensionality.

4.2.3 Removing the Outliers

In our dataset, we calculated the Z-score for each numerical variable to identify and remove the outliers. The Z-score is a statistical measurement that describes a value's relationship to the mean of a group of values. It's measured in terms of standard deviations from the mean.To begin with, we calculated Z-scores for all the numerical variables in our dataset. These scores provide us with an idea of how much a data point deviates from the mean in terms of standard deviations. After calculating these scores, we set a threshold of 3 standard deviations, commonly considered in statistics to be the boundary for outliers.We then identified the rows in our data that contained outliers; that is, data points that had a Z-score greater than 3 or less than -3. We used a logical OR operator (—) to check for either condition.

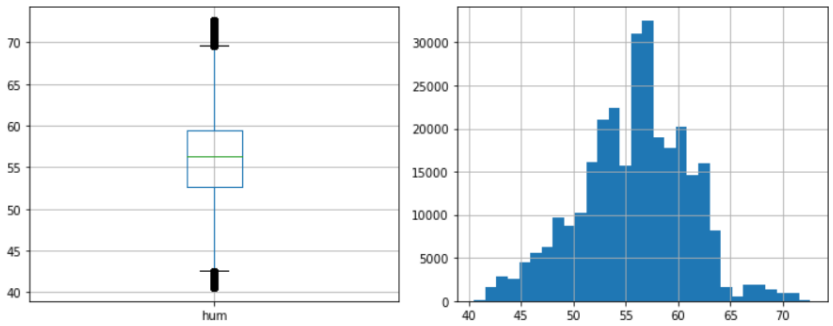

Figure 5: Outliers in 'hum' variable

Finally, we removed these identified outliers from the DataFrame by negating the condition (using), thus preserving only those rows that do not contain outliers. By removing these outliers, we are able to create a dataset that is more representative of the underlying patterns we want our machine learning model to learn. This leads to more robust and accurate models.After removing the outliers, we were left with a DataFrame that is ready to be used in the subsequent steps of our analysis.

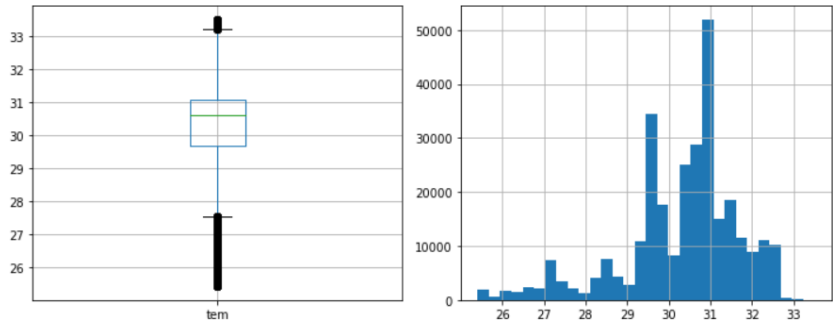

Figure 6: Outliers in 'temp' variable

4.2.4 Handling Missing Values in 'occ'

After mapping the 'occ' column to numeric values, missing values are identified in this column. Missing values indicate instances where the occupancy level is not recorded or is unknown. To avoid potential biases and inaccuracies in the model, rows with missing 'occ' values are removed from the dataset. The decision to remove these rows is based on the assumption that the missing values are not informative or meaningful for predicting occupancy levels. After removing the rows with missing 'occ' values, the dataset becomes clean and contains no missing values.

4.3 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a crucial step in the data analytics process that involves the systematic examination and visualization of data to gain insights and understand the underlying patterns, relationships, and characteristics of the dataset. In this study, EDA is performed to better understand the dataset related to occupancy levels and other variables. The specific visualizations used in the EDA process are as follows:

4.3.1 Count Plot of Occupancy Levels

A count plot is a type of bar plot that shows the number of occurrences of each category in a categorical variable. In this analysis, a count plot is generated for the 'occ' column, which represents the occupancy levels. The count plot visually displays the distribution of occupancy levels, providing valuable information about the data imbalance and the relative frequency of each occupancy level (Low, Medium, High). This helps to identify any potential biases in the dataset and assess the prevalence of different occupancy levels.

4.3.2 Correlation Matrix Heatmap

A correlation matrix is a table that shows the correlation coefficients between pairs of variables in a dataset. In this study, a correlation matrix heatmap is generated using the 'corr' function, which calculates the correlation between numerical variables. The heatmap provides a visual representation of the correlation values, with colors indicating the strength and direction of the correlations. This allows for the identification of potential relationships between variables, including positive and negative correlations. For instance, a strong positive correlation between two variables suggests that they tend

to increase or decrease together, while a strong negative correlation indicates an inverse relationship.

4.3.3 Scatterplots of Pressure vs. Temperature and Humidity vs. Temperature

Scatterplots visualize the relationship between two numerical variables. In this study, scatterplots display the correlation between pressure and temperature, as well as humidity and temperature. Data points on these plots represent pressure or humidity (x-axis) against temperature (y-axis) and are colored by occupancy levels (Low, Medium, or High). This color-coding reveals how occupancy varies with pressure and humidity at different temperatures. These plots illuminate trends or clusters, highlighting potential variable interrelations.

Through exploratory data analysis (EDA) and such visualizations, researchers can deeply understand the dataset, pinpoint patterns, and decide on data preprocessing and model choices. EDA is pivotal in data analytics, setting the foundation for future analysis and promoting data-driven decisions.

4.4 Data Modelling with TABnet

This study predicts occupancy levels using environmental features and employs the TabNet model for its prowess with tabular data and intricate feature relationships. The modeling process includes:

- **Feature Selection:** Data is categorized into environmental features and the numeric 'occ_int' representing occupancy levels (Low, Medium, High).
- **Validation Method:** Stratified k-fold cross-validation ensures each subset retains the class label proportions, balancing occupancy level distributions.
- **Model Training:** Using the TabNetClassifier from pytorch tabnet, the model trains over 50 epochs with an early stopping mechanism set at a patience of 10.
- **Evaluation:** Post-training, model accuracy is gauged with a test set. A confusion matrix and classification report detail model performance across occupancy levels.
- **Feature Importance:** TabNet's built-in mechanism unveils feature significance, visualized with a bar plot.

Overall, TabNet's capabilities, combined with a structured approach, make it adept for predicting occupancy levels based on environmental indicators.

4.5 Model Evaluation

The TABnet model is central to this study, aiming to predict occupancy levels based on environmental features. The evaluation process encompasses:

- **Training the TABnet Model:**
  - Dataset Split: Dividing into train_features (environmental variables) and train_target (numerical occupancy levels).

  - Hyperparameters: Configuring a maximum of 50 epochs, employing early stopping with a patience of 10 to avoid overfitting, and setting batch sizes (32 for standard batch size and 16 for virtual batch size) to optimize GPU memory.

- **Model Evaluation Process:**
  - Model Training: Leveraging specified hyperparameters and gauging performance on the test dataset.
  - Data Splitting: Similar to training, partitioning into train features for independent variables and train_target for numerical occupancy representations.
  - Batch Configuration: Setting batch sizes of 32 and virtual batches of 16 for efficient data processing.

- **Evaluation Metrics:**
  - Accuracy Score: The TABnet model achieved approximately 82.61
  - Confusion Matrix: A tabular representation comparing model predictions to actual labels.
  - Classification Report: Includes metrics like precision, recall, and F1-score for each occupancy level.

5 Implementation

5.1 Importing Libraries

The project's foundation lies in importing key Python libraries essential for data handling, modeling, and visualization:

- **Pandas:** A cornerstone library for data manipulation and analysis in Python, offering versatile data structures and I/O functions.
- **PyTorch TABnet:** Implements the TabNet model, tailored for tabular data, streamlining its creation, training, and evaluation.
- **Scikit-learn:** A renowned machine learning library, utilized here for data splitting, model evaluation, and other ML tasks.
- **Matplotlib:** Python's primary plotting library, employed for various visualizations in our exploratory data analysis.
- **Seaborn:** An enhancement over Matplotlib, it offers advanced graphics and is used for intricate visualizations like heatmaps and scatterplots.

Each of these libraries plays a crucial role in the project and is used at different stages of the implementation, from data loading and preprocessing to exploratory data analysis, modeling, and evaluation.

5.2 Loading Data

The data for this project is stored in a CSV (Comma Separated Values) file named ”Home.csv”. This file is loaded into a pandas DataFrame using the read_csv() function. The line df = pd.read_csv(”Home.csv”) reads the CSV file and stores the data in the variable df as a DataFrame. This format allows for easy data manipulation and analysis using pandas’ extensive functionality.

5.3 Data Preprocessing

- **Creating Interaction Term: Temperature x Humidity:**
  - Formed an interaction term between temperature and humidity.
  - Objective: Capture their combined influence on ’occ int’.
  - Method: Multiplying the two variables.
- **Dropping Redundant Columns:**
  - Removed ’date’ column: unrelated to the prediction task.
  - Removed ’occ’ column: duplicated information with ’occ int’.
  - Objective: Avoid model bias and redundancy.
- **Managing Missing Data:**
  - Screened dataset for missing values using pandas’ isnull() function.
  - Objective: Prevent model outcomes distortion.
  - Possible strategies: Central tendency imputation, row/column removal, etc.
- **Handling Outliers:**
  - Detected outliers using methods like Z-score or IQR.
  - Objective: Prevent distortion in model predictions.
  - Action: Outliers were removed or adjusted.
- **Feature Scaling:**
  - Standardized data features.
  - Tool: sklearn’s StandardScaler.
  - Objective: Ensure consistent influence of each feature on the model.

These preprocessing steps lay the groundwork for efficient modeling.

5.4 Exploratory Data Analysis (EDA)

- **Visualizing Data Distributions:**
  - Used histograms to visualize data distributions.
  - Tools: matplotlib and seaborn.
  - Objective: Detect skewness and outliers.
- **Understanding Variable Correlations:**
  - Created correlation matrix heatmaps.
  - Tool: seaborn.
  - Objective: Visualize correlations between features and guide feature selection.
- **Scatterplots: Relationships Between Variables:**
  - Constructed scatterplots for 'pressure vs. temperature' and 'humidity vs. temperature'.
  - Objective: Identify relationships, trends, and outliers between variables.

Through EDA, a deeper understanding of the dataset and variable relationships was acquired, which will inform subsequent modeling strategies.

5.5 Data Modelling with TabNet

- **Dataset Split: Features vs Target:**
  - Segmented dataset into features (e.g., 'pressure', 'altitude') and target variable ('occ int').
  - Tool: pandas.
- **Stratified K-Fold Cross-Validation:**
  - Employed Stratified K-Fold for reliable model performance estimates.
  - Importance: Essential for imbalanced datasets.
- **Training TabNet:**
  - Trained the TabNet model on feature-target relationships.
  - Tool: PyTorch TabNet library.
  - Emphasized on hyperparameter tuning, such as setting learning rate and epochs.

These modeling steps ensure the model's effective learning and accurate performance assessment.

5.6 Training the model

The model employs stratified k-fold cross-validation, where the dataset is split into k ”folds”. Training occurs k times, rotating testing across the folds to ensure each is tested once. Stratification ensures each fold maintains the original class proportions, crucial for imbalanced datasets. Key hyperparameters for the TabNet model:

Learning Rate (learning rate = 0.001): Dictates optimizer step size. Smaller rates increase accuracy but slow training, while larger rates might miss the optimum.

Max_epochs (30): Maximum cycles through the dataset, allowing each sample to update the model.

Patience (10): For early stopping. Training halts if no validation performance improvement occurs over 10 epochs.

Batch size (32): Number of training samples per batch, after which model parameters are updated.

Virtual batch size (16): Size of mini-batches for aggregation during forward pass, impacting model capacity. Post-training, the model’s performance is gauged using a confusion matrix and a classification report, detailing precision, recall, and f1-score. The finalized model is saved, enabling future loading and predictions on new data.

5.7 Analyzing Feature Importance

5.7.1 Accessing Feature Importance Scores from the TabNet Model

The TabNet model offers a beneficial feature that calculates the importance of each feature in predicting the target variable. This information, accessible through the feature_importances_ attribute, provides insights into which features are the most influential in the model’s predictions.

5.7.2 Visualization of Feature Importance Scores

The feature importance scores are represented through a bar plot, where each feature’s importance is directly proportional to the length of its corresponding bar. In this plot:

- The feature ’hum’ (humidity) is the most influential in predicting occupancy levels, with its importance score lying midway between 0.20 and 0.25.
- The features ’pre’ (pressure) and ’tem hum interaction’ (the interaction term between temperature and humidity) have a similar degree of importance, with scores just above 0.15. This indicates that these factors also play a significant role in occupancy level prediction.
- The ’tem’ (temperature) feature has a moderate impact, with an importance score just above 0.10.

- The 'alt' (altitude) feature has the least influence on the model's prediction, with an importance score just above 0.05. This suggests that altitude might not be a critical factor in predicting occupancy levels.

This visualization provides an insightful way to understand which features the model deems most important when predicting occupancy levels, offering potential avenues for further exploration or model refinement.

5.8 Predictions

- **Making predicting with the Model:** After the TabNet model is loaded, it can be used to make predictions on new unseen data. The predict() function of the model is used for this purpose. This function takes the features of the test set as input and outputs the predicted labels.

- **Testing Process:** In the case of this project, the test set data is passed into the predict() function of the loaded model. The process might look something like this: predictions = tabnet_model.predict(test_features). Here, test_features contains the independent variables (environmental factors) of the test data.

- **Output:** The output, predictions, is an array of the predicted occupancy levels for each instance in the test set. Each prediction is the model's best guess for the occupancy level based on the environmental factors of that instance.

- **Usage of Predictions:** These predictions can then be used for various purposes, such as evaluating the performance of the model (by comparing the predictions to the true labels), visualizing the results, or making decisions based on the predicted occupancy levels.

6 Model Evaluation

6.1 Accuracy Calculation

Accuracy, one of the most straightforward metrics for evaluating classification models, is calculated as the number of correct predictions made by the model divided by the total number of predictions. In Python, the accuracy_score() function from the sklearn.metrics module calculates accuracy, taking the actual labels and the predicted labels as arguments. For this project, the accuracy score sheds light on how frequently the model correctly predicts the occupancy levels. Although accuracy is an easily comprehensible metric, it may not always be the best measure for imbalanced datasets, where performance on the minority class can be more crucial.

The model achieved an overall accuracy of 0.86, implying that it made correct predictions for 86% of the instances in the test set.

6.2 Confusion Matrix

The confusion matrix provides a more granular view of the model's performance across different classes. Here is the confusion matrix for our model:

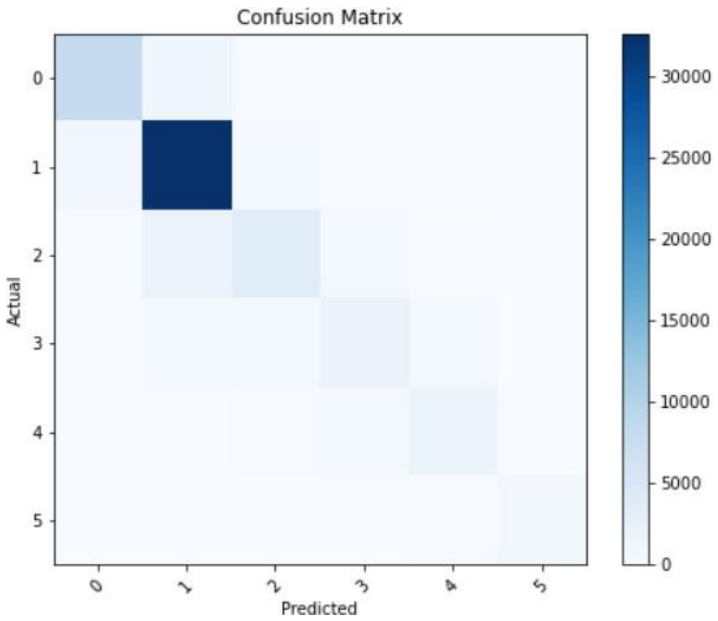

Figure 7: Confusion Matrix

Each row of the matrix corresponds to the actual class, while each column corresponds to the predicted class. The diagonal elements (from the top left to the bottom right) of the matrix show the number of correct predictions for each class.

For instance, the first row corresponds to occupancy level 0. The model correctly predicted this level 8,035 times (as seen in the first element of the row). However, it also incorrectly predicted this level as 1 1,268 times and as 2 224 times. Similarly, you can interpret the other rows.

The confusion matrix helps us understand the types of errors the model is making. For instance, it appears that the model sometimes confuses the 0 occupancy level with the 1 and 2 levels.

6.3 Classification Report

The classification report provides additional insightful metrics, such as precision, recall, and the F1 score.

Class	Precision	Recall	F1-Score	Support
0	0.89	0.84	0.87	9527
1	0.90	0.96	0.93	34105
2	0.74	0.60	0.66	6254
3	0.69	0.58	0.63	3572
4	0.66	0.69	0.67	2566
5	0.67	0.66	0.66	813
Accuracy: 0.86 Macro Avg (Precision, Recall, F1-Score): (0.76, 0.72, 0.74) Weighted Avg (Precision, Recall, F1-Score): (0.85, 0.86, 0.85)				

Table 2: Classification Report

- For the occupancy level 0, the model demonstrated a precision of 0.89, recall of 0.84, and an F1 score of 0.87. This suggests that the model is quite adept at predicting this level of occupancy, with fewer false positives (as indicated by the high precision) and fewer false negatives (as indicated by the high recall).

- For occupancy level 1, the precision was 0.90, recall was 0.96, and the F1 score was 0.93. These high scores imply that the model is highly effective at predicting this occupancy level.

- For the other levels (2 to 5), the metrics vary, with precision ranging from 0.74 to 0.67, recall from 0.60 to 0.66, and F1 score from 0.66 to 0.67. The slightly lower scores for these levels suggest that the model has some difficulty accurately predicting them.

By interpreting these metrics collectively, we can gain a more comprehensive understanding of the model's performance. For instance, while the model exhibits relatively high precision and recall for the 0 and 1 occupancy levels, the performance slightly decreases for levels 2 to 5. This observation can guide potential improvements to the model or adjustments in the feature engineering process. Perhaps the model needs more representative samples for the classes with lower performance, or it may require adjustments to better capture the characteristics of these classes.

The weighted avg row in the classification report takes into account the number of instances for each class during the calculation of average metrics. It shows that, on average, taking the distribution of classes into account, our model has a precision, recall, and F1-score of 0.85, 0.86, and 0.85 respectively.

7 Conclusion and Future Work

The research undertaken by the team sought to enhance occupancy prediction through environmental sensor data, drawing inspiration from foundational studies by Pal, Yang, and Dey. These prior works emphasized the significance of feature extraction, data preprocessing, and the application of machine learning.

In response to this foundation, the researchers prioritized data preprocessing, creating interaction terms for temperature and humidity, optimizing data quality, and discarding irrelevant data. Using the TabNet model, tailored for tabular data and built with prominent Python libraries, they achieved an accuracy of 0.86. Notably, the model demonstrated proficiency in predicting lower occupancy levels but faced challenges with higher levels, signifying areas for refinement.

Through an analysis of features, humidity emerged as the paramount predictor, succeeded by pressure and the interaction between temperature and humidity. This sheds light on potential avenues for further research or model enhancement.

The interpretability of the model, facilitated by TabNet's feature importance scores, resonates with the literature's emphasis on the transparency of machine learning models. Additionally, the model's preservation using PyTorch ensures its adaptability in real-world scenarios.

However, challenges arose when predicting higher occupancy levels, pointing to a need for enhanced feature engineering or alternative modeling techniques. Broadening the sensor data spectrum or applying the model in diverse environments could amplify its precision.

In summation, this study underscores the potential of machine learning, particularly the TabNet model, for advanced occupancy prediction. While the results are promising, there are avenues for model refinement and expanded application, laying the groundwork for continued research in this essential domain.

References

[1] A. P. Singh, V. Jain, S. Chaudhari, F. A. Kraemer, S. Werner, and V. Garg, "Machine learning-based occupancy estimation using multivariate sensor nodes," in *2018 IEEE Globecom Workshops (GC Wkshps)*, 2018, pp. 1–6.

[2] Z. Yang, N. Li, B. Becerik-Gerber, and M. Orosz, "A multi-sensor based occupancy estimation model for supporting demand driven hvac operations," in *Proceedings of the 2012 Symposium on Simulation for Architecture and Urban Design*, ser. SimAUD '12. San Diego, CA, USA: Society for Computer Simulation International, 2012.

[3] A. Dey, X. Ling, A. Syed, Y. Zheng, B. Landowski, D. Anderson, K. Stuart, and M. E. Tolentino, "Namatad: Inferring occupancy from building sensors using machine learning," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, 2016, pp. 478–483.

[4] A. Vela, J. Alvarado-Uribe, M. Davila, N. Hernandez-Gress, and H. G. Ceballos, "Estimating occupancy levels in enclosed spaces using environmental variables: A fitness gym and living room as evaluation scenarios," *Sensors*, vol. 20, no. 22, p. 6579, 2020.

[5] M. K. Masood, Y. C. Soh, and V. W. . Chang, "Real-time occupancy estimation using environmental parameters," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–8.

[6] E. Hailemariam, R. Goldstein, R. Attar, and A. Khan, "Real-time occupancy detection using decision trees with multiple sensor types," in *Proceedings of the 2011 Symposium on Simulation for Architecture and Urban Design*, ser. SimAUD '11. San Diego, CA, USA: Society for Computer Simulation International, 2011, p. 141–148.

[7] Z. Chen *et al.*, "Building occupancy estimation with environmental sensors via cd-blstm," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 12, pp. 9549–9559, 2017.

[8] L. Candanedo and V. Feldheim, "Accurate occupancy detection of an office room from light, temperature, humidity and co_2 measurements using statistical learning models," *Energy and Buildings*, vol. 112, pp. 28–39, 2016.

[9] Y. Agarwal *et al.*, "Occupancy-driven energy management for smart building automation," in *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*. ACM, 2010, pp. 1–6.