Room Occupancy Estimation Using Advanced Machine Learning Techniques

Lavanya Chand,

Sriram Reddy Kondle,

Chandrasekhar Naidu Seelam.

AAI-530- Data Analytics and the Internet of Things

Instructor: **Ebrahim Tarshizi, Ph.D., M.B.A.**

May 08, 2023.

CONTENTS

ABSTRACT:

Person detection using non-intrusive sensors is a popular approach for occupancy estimation in IoT devices. In this project, we explore using non-intrusive sensors, including temperature, light, CO2, and humidity, for occupancy estimation in a room. We use the Room Occupancy Estimation dataset from the UCI Machine Learning Repository, which contains information on temperature, light, humidity, CO2 levels, and occupancy in a room.

We explore different machine learning algorithms, including Decision Trees and Random Forests, to predict occupancy based on sensor data. We also design an IoT system to monitor occupancy in a room using non-intrusive sensors, including controllers, gateways, and cloud services. Our team overcomes challenges related to sensor placement and data transmission through collaboration and experimentation. Finally, we discuss using deep learning methods, including Recurrent Neural Networks with Long Short-Term Memory cells, for occupancy estimation in IoT devices. Our project provides insights into the design and implementation of IoT systems for occupancy estimation using non-intrusive sensors.

I. INTRODUCTION

Project Introduction:

In a wide range of circumstances, like during natural catastrophes or military operations, where prompt knowledge of room occupancy might save lives, the capacity to identify the human presence in a room is of utmost importance. Accurate room occupancy detection can also increase security and energy efficiency in regular circumstances, like seeing a burglar in the house or controlling room amenities like lighting and air conditioning based on occupancy. Considering this, this project aims to provide a dependable room occupancy detection system that can reliably detect the presence of people in space, enabling effective source management and decision-making in a range of scenarios. The system will use cutting-edge sensors and algorithms to detect human presence to increase safety, security, and convenience.

Our team uses an advanced deep learning technique, i.e., Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM). We use two machine learning algorithms, Decision Tree and Random Forest.

Problem Statement:

Using time-series sensor data, the challenge is calculating the room's occupancy. The aim is to establish whether a human is present in the room using the sensor data. Energy savings, security, and automation are only a few uses for room occupancy estimation. The difficulty lies in creating an accurate and dependable model that can manage the complex and dynamic nature of the sensor data and offer real-time occupancy estimation.

II. LITERATURE REVIEW

Deep Learning Methods:

Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) cells is a popular deep-learning approach that can be utilized for Room Occupancy Estimation.

Recurrent Neural Networks (RNN):

Recurrent Neural Networks (RNNs) are a type of neural network designed to handle sequential data. Unlike traditional feedforward neural networks, RNNs can consider the context of previous inputs to predict the current input.

The critical feature of RNNs is their ability to maintain a "memory" of previous inputs in the form of hidden states. These hidden states are updated at each time step based on the current input and the previous hidden state. The updated hidden state is then used to predict the current input.

One of the most popular applications of RNNs is in natural language processing (NLP) tasks such as language modeling, machine translation, and speech recognition. In these tasks, RNNs are used to process sequences of words or phonemes and produce a corresponding output.

There are several variants of RNNs, including Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs). LSTM networks and GRUs are designed to address the vanishing gradient problem that can occur with traditional RNNs. The vanishing gradient problem occurs when the gradients used to update the weights in the network become very small, making it difficult to train the network effectively.

Overall, RNNs are a powerful tool for modeling sequential data and have been successfully applied in a wide range of fields, including NLP, speech recognition, image captioning, and time-series prediction.

Long Short-Term Memory (LSTM):

LSTM is a Recurrent Neural Network (RNN) architecture designed to address the vanishing gradient problem in traditional RNNs. The vanishing gradient problem arises when gradients (derivatives of the loss function concerning the model parameters) get smaller and smaller as they propagate through many time steps in a sequence, making it difficult for the model to learn long-term dependencies.

LSTM introduces the concept of a memory cell that can maintain information over long periods, allowing the model to selectively remember or forget information based on the current input. Three gates control the memory cell: the input gate, the forget gate, and the output gate.

The input gate determines how much of the current input to store in the memory cell. It takes the input vector and the previous hidden state as inputs and outputs a vector of values between 0 and 1, representing each element's relevance in the input vector.

The forget gate determines how much of the previous memory to retain. It takes the input vector and the previous hidden state as inputs. It outputs a vector of values between 0 and 1 representing the degree to which each element of the previous memory should be forgotten.

The output gate determines how much current memory is used to output. It takes the input vector and the previous hidden state as inputs. It outputs a vector of values between 0 and 1 representing the degree to which each memory cell element should be output.

The cell state is updated by combining the input and forgot gates and applying the resulting weights to the current input and previous memory. The output is then computed using the output gate to the updated memory.

Overall, LSTM can selectively store or discard information based on the current input, allowing it to learn long-term dependencies and avoid the vanishing gradient problem. LSTMs are effective in various applications, including speech recognition, language modeling, and image captioning.

RNN with LSTM Architecture:

RNNs with LSTM (Long Short-Term Memory) are commonly used in sequential data analysis tasks such as speech recognition, natural language processing, and time series analysis. They are well-suited for such tasks because they can model the temporal dependencies in the input data.

One of the main advantages of using an LSTM architecture with RNNs is their ability to handle vanishing and exploding gradients effectively. This is a common issue when training RNNs with conventional architectures, where the gradients either vanish or explode as they propagate through time.

LSTM units are specifically designed to address this issue. They do this by introducing a gating mechanism that allows the network to remember or forget information at each time step selectively. The gates are implemented as nonlinear functions trained alongside the rest of the network parameters.

The input gate determines how much new information is allowed to enter the memory cell, the forget gate determines how much of the previous cell state should be forgotten, and the output gate determines how much of the cell state should be output at the current time step. The LSTM architecture also includes a cell state that acts as a memory for the network. This cell state is updated at each time step, with information being added or removed based on the input and gate values.

RNN with LSTM can detect a person's presence in a room by analyzing time-series data, such as sensor data from cameras, microphones, or other devices. The model can be trained on a large

dataset of sensor data to learn the patterns and features associated with the presence of a person in a room.

Once trained, the model can analyze real-time data from sensors in a room to determine if a person is present. The model can also be used to trigger actions, such as turning on lights or alerting security personnel, based on the presence or absence of a person in the room.

The LSTM component of the model can help to capture long-term dependencies and patterns in the sensor data, which can be useful in detecting the presence of a person over extended periods. Additionally, the RNN component of the model can help to analyze sequences of sensor data, which can be useful in detecting subtle changes in the sensor data that may indicate the presence or absence of a person.

Overall, RNN with LSTM can be a powerful tool for detecting the presence of a person in a room and triggering actions based on that information.

Machine Learning methods for time-series management:

The best-suited algorithm for time series requirement would be the Decision Tree and Random Forest algorithm. These algorithms can handle time series data by analyzing the relationship between the input and target variables over time.

Decision Tree:

A decision tree is a supervised machine-learning algorithm commonly used for classification and regression analysis. It is a tree-like model that represents decisions and their possible consequences. In a decision tree, each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label or a numerical value.

Decision trees are constructed recursively, starting from the root node, by selecting the best data split attribute. The best split is determined by measuring the attribute's impurity or information gain. The impurity measures how mixed the class labels are at a particular node. In contrast, the information gained measures the impurity reduction achieved by splitting the data on a particular attribute.

Once the decision tree is constructed, it can be used to predict the class label or the numerical value of a new sample by traversing the tree from the root to a leaf node based on the values of the attributes of the sample.

Random Forest:

Random Forest is an ensemble learning method that uses multiple decision trees to improve the performance of a single decision tree. Random Forest is a popular algorithm for classification and regression analysis, and it is often used in machine learning competitions.

In a Random Forest, a set of decision trees is constructed using a random subset of the training data and a random subset of the attributes at each tree node. The randomness in the data and attributes ensures that each decision tree is different, and the ensemble of trees can capture the complexity of the data better than a single decision tree.

During the prediction phase, the sample is passed through each decision tree, and the majority vote or the average of the predictions of all the trees is taken as the final prediction.

Random Forest has several advantages over a single decision tree, such as reduced overfitting, improved accuracy, and increased robustness to noise and outliers. Once the decision tree is constructed, it can be used to predict the class label or the numerical value of a new sample by traversing the tree from the root to a leaf node based on the values of the attributes of the sample.

III. Methodology

IoT System Design:

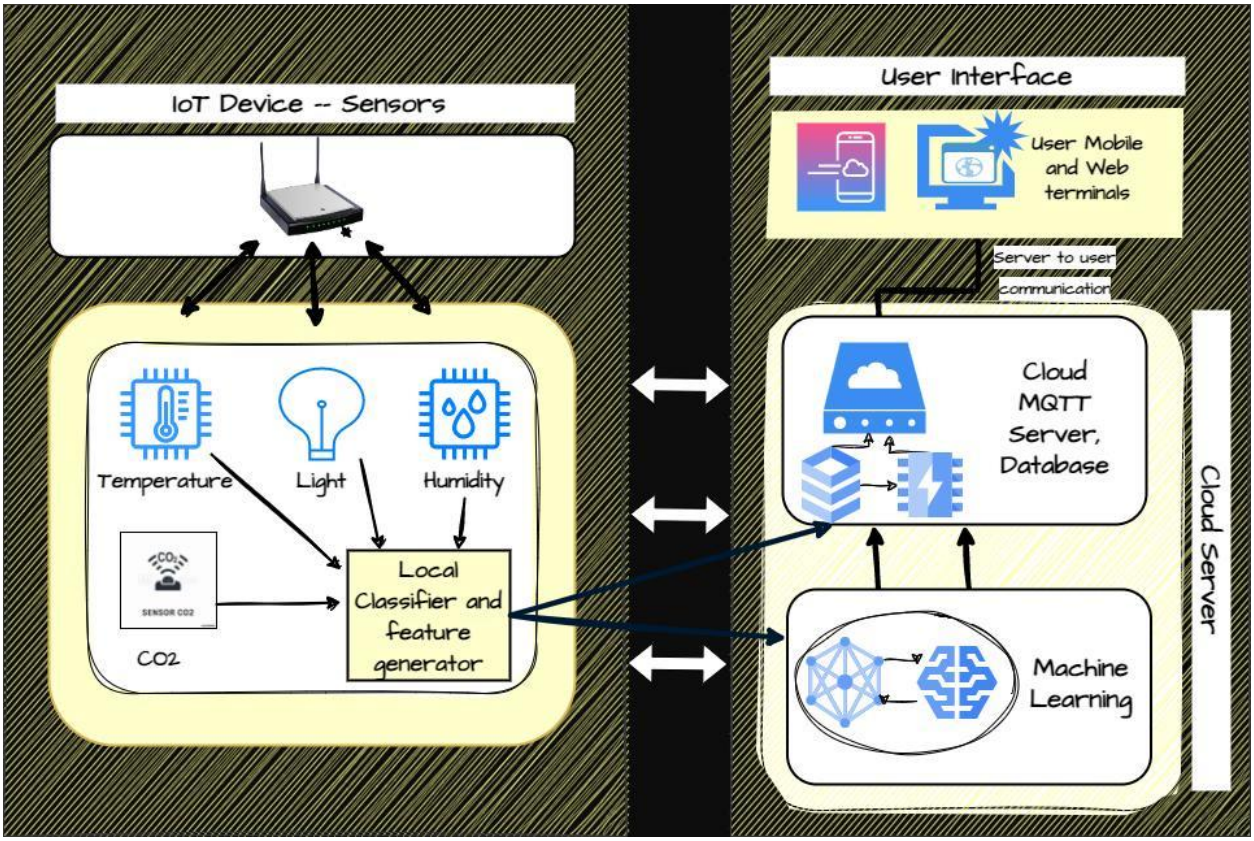


Fig 1.1 Room Occupancy Estimation System Design

- **Sensors:** The IoT system for estimating room occupancy would require sensors to detect human presence. The sensors include PIR (passive infrared), ultrasonic, or microwave sensors. PIR sensors are the most used as they are inexpensive and easy to install.
- **Data Acquisition:** The sensors will collect data on human presence in the room and send it to a microcontroller, such as an Arduino or Raspberry Pi, to process the data.
- **Communication:** The microcontroller will communicate with a central server, either through Wi-Fi, Ethernet, or GSM module. The server will receive the data from the microcontroller and perform further processing.
- **Data Processing:** The server will analyze the data to determine the occupancy of the room. This can be done through a machine learning algorithm, such as RNN with LSTM or decision tree and random forest. The algorithm will learn from the data and make predictions about the occupancy of the room based on the sensor data.

- **Notification:** Once the occupancy of the room is determined, the IoT system can send notifications to the user through various channels, such as SMS, email, or push notifications. For example, if the room is occupied, the system can turn on the lights or adjust the temperature.
- **Dashboard:** The system can also display the room's occupancy status on a dashboard that the user can access through a web or mobile application. The dashboard can provide real-time data on the occupancy of the room and historical data for analysis.

Overall, an IoT system for room occupancy estimation can provide a smart and efficient way of managing the occupancy of a room, which can have various applications in homes, offices, hospitals, and other industries.

Dataset

1. Source of Dataset:

We will use Room Occupancy Estimation Data Set from the UCI Machine Learning Repository as the primary dataset for our IoT application/system. The ground truth of the occupancy count in the room was noted manually and made publicly available through the UCI repository.

2. Data Collection:

The data was collected in a controlled manner for 4 days, with the occupancy in the room varying between 0 and 3 people.

3. Observations:

The dataset contains a total of 20561 instances, the precise number of occupants in a room using multiple non-intrusive environmental sensors like temperature, light, sound, CO2, and PIR. About Dataset Binary classification (room occupancy) from Temperature, Humidity, Light, and CO2. Occupancy was obtained from time-stamped pictures that were taken every minute.

4. The Attribute Informationin the dataset are:

• Date: YYYY/MM/DD

• Time: HH: MM: SS

• Temperature: In degrees Celsius

• Light, in Lux

• Humidity in %

• Carbon dioxide (CO2) (in PPM)

• Humidity Ratio, Derived quantity from temperature and relative humidity, in kgwatervapor/kg-air.

• Occupancy, 0 or 1, 0 for not occupied, 1 for occupied status.

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
0	2015-02-02 14:19:00	23.7000	26.272	585.200000	749.200000	0.004764	1
1	2015-02-02 14:19:59	23.7180	26.290	578.400000	760.400000	0.004773	1
2	2015-02-02 14:21:00	23.7300	26.230	572.666667	769.666667	0.004765	1
3	2015-02-02 14:22:00	23.7225	26.125	493.750000	774.750000	0.004744	1
4	2015-02-02 14:23:00	23.7540	26.200	488.600000	779.000000	0.004767	1
5	2015-02-02 14:23:59	23.7600	26.260	568.666667	790.000000	0.004779	1
6	2015-02-02 14:25:00	23.7300	26.290	536.333333	798.000000	0.004776	1
7	2015-02-02 14:25:59	23.7540	26.290	509.000000	797.000000	0.004783	1
8	2015-02-02 14:26:59	23.7540	26.350	476.000000	803.200000	0.004794	1
9	2015-02-02 14:28:00	23.7360	26.390	510.000000	809.000000	0.004796	1

The dataset provides information about an experimental testbed for occupancy estimation in a 6m × 4.6m room using non-intrusive sensors. The setup consists of sensor nodes and one edge node in a star configuration, with the sensor nodes transmitting data to the edge every 30s using wireless transceivers. The non-intrusive sensors used in the experiment are temperature, light, CO2, and humidity. The data were collected for four days in a controlled manner, with the occupancy in the room varying between 0 and 3 people, and the ground truth of the occupancy count in the room was noted. This information provides a detailed understanding of the experimental setup and the data collection process for occupancy estimation using non-intrusive sensors.

Execution:

Importing the dataset

In [1]:

```
import pandas as pd

# Read the dataset into a Pandas DataFrame
df = pd.read_csv('Occupancy.csv')

df.head(10)
```

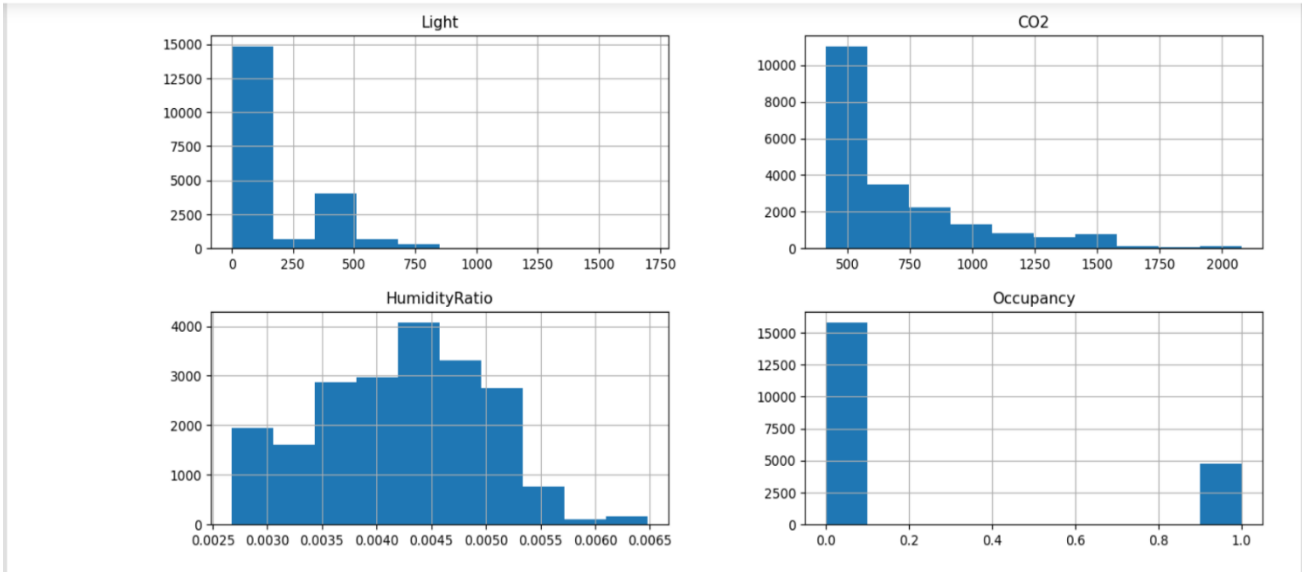
Out[1]:

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
0	2015-02-02 14:19:00	23.7000	26.272	585.200000	749.200000	0.004764	1
1	2015-02-02 14:19:59	23.7180	26.290	578.400000	760.400000	0.004773	1
2	2015-02-02 14:21:00	23.7300	26.230	572.666667	769.666667	0.004765	1
3	2015-02-02 14:22:00	23.7225	26.125	493.750000	774.750000	0.004744	1
4	2015-02-02 14:23:00	23.7540	26.200	488.600000	779.000000	0.004767	1
5	2015-02-02 14:23:59	23.7600	26.260	568.666667	790.000000	0.004779	1
6	2015-02-02 14:25:00	23.7300	26.290	536.333333	798.000000	0.004776	1
7	2015-02-02 14:25:59	23.7540	26.290	509.000000	797.000000	0.004783	1
8	2015-02-02 14:26:59	23.7540	26.350	476.000000	803.200000	0.004794	1
9	2015-02-02 14:28:00	23.7360	26.390	510.000000	809.000000	0.004796	1

The given code reads in a dataset from a CSV file named "Occupancy.csv" using the pandas library and stores it as a pandas Data Frame object called "df." It then displays the first ten rows of the data frame using the "head()" method.

Exploratory Data Analysis

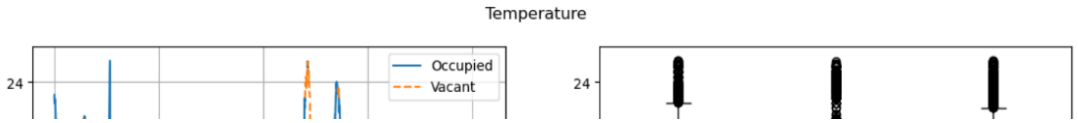
```
In [5]: 1 features = ["Temperature", "Humidity","Light","CO2","HumidityRatio","Occupancy"]
        2 df[features].hist(figsize=(15, 10))

Out[5]: array([[<AxesSubplot:title={'center':'Temperature'}>,
                <AxesSubplot:title={'center':'Humidity'}>],
               [<AxesSubplot:title={'center':'Light'}>,
                <AxesSubplot:title={'center':'CO2'}>],
               [<AxesSubplot:title={'center':'HumidityRatio'}>,
                <AxesSubplot:title={'center':'Occupancy'}>]], dtype=object)
```

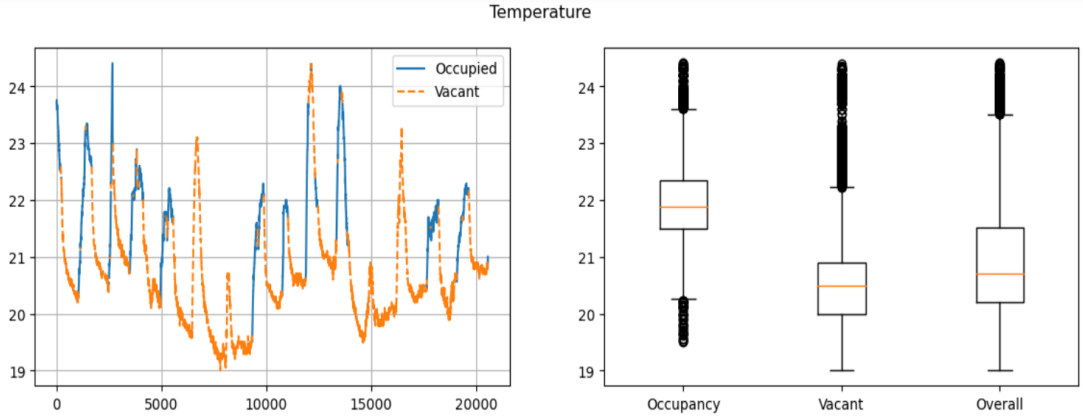


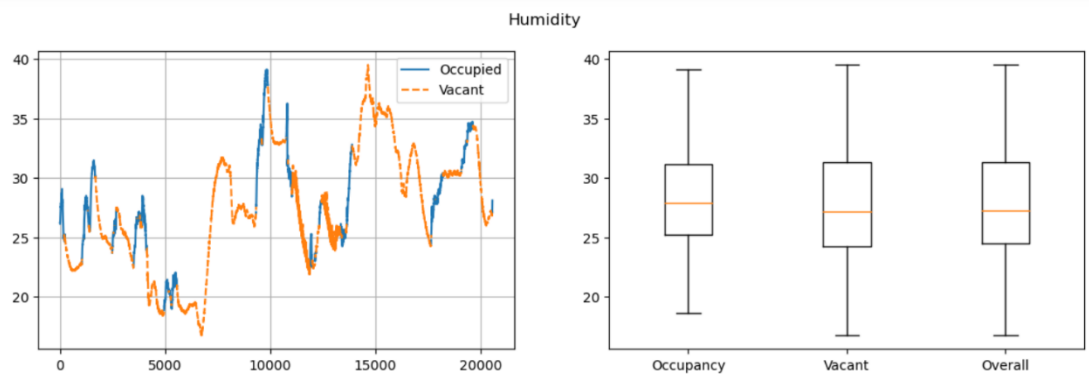
This code reads in a dataset from a CSV file and selects a subset of columns to create a histogram visualization. The columns included in the visualization are temperature, humidity, light, CO2, humidity ratio, and occupancy. The hist() function creates a histogram for each column, and the figsize parameter specifies the size of the overall figure.

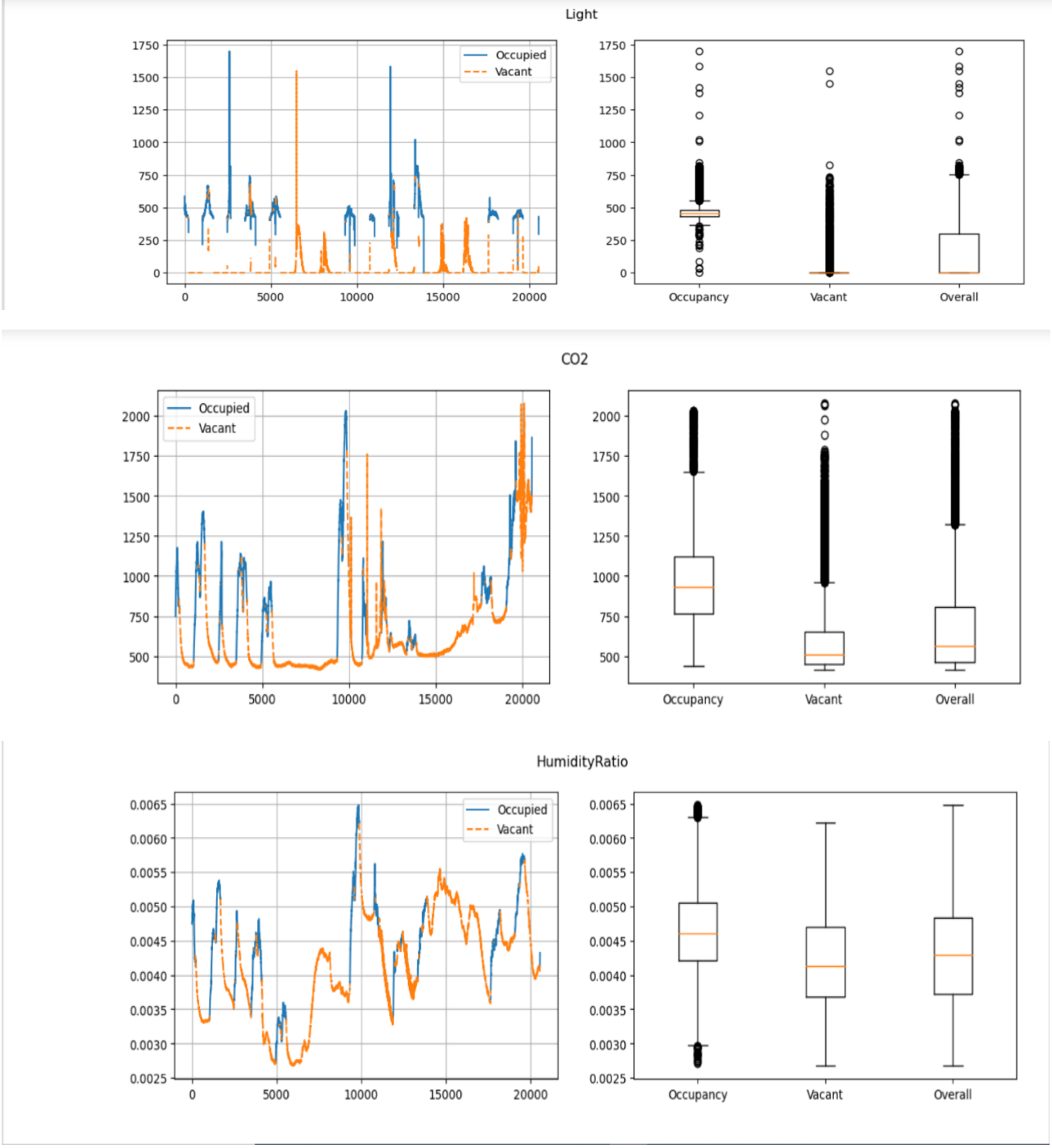
Data Visualization:

In [9]:
```
import matplotlib.pyplot as plt
import numpy as np


def occupancy_plot(df, cat):
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14,4))

    fig.suptitle(cat)
    ax1.plot(np.where(df.Occupancy==1, df[cat], None), label='Occupied')
    ax1.plot(np.where(df.Occupancy==0, df[cat], None), label='Vacant', ls='--')
    ax1.grid()
    ax1.legend()

    ax2.boxplot([df[cat][df.Occupancy==1], df[cat][df.Occupancy==0], df[cat]])
    ax2.set_xticklabels(['Occupancy', 'Vacant', 'Overall'])


for i in range(1, 6):
    occupancy_plot(df, df.columns[i])
```









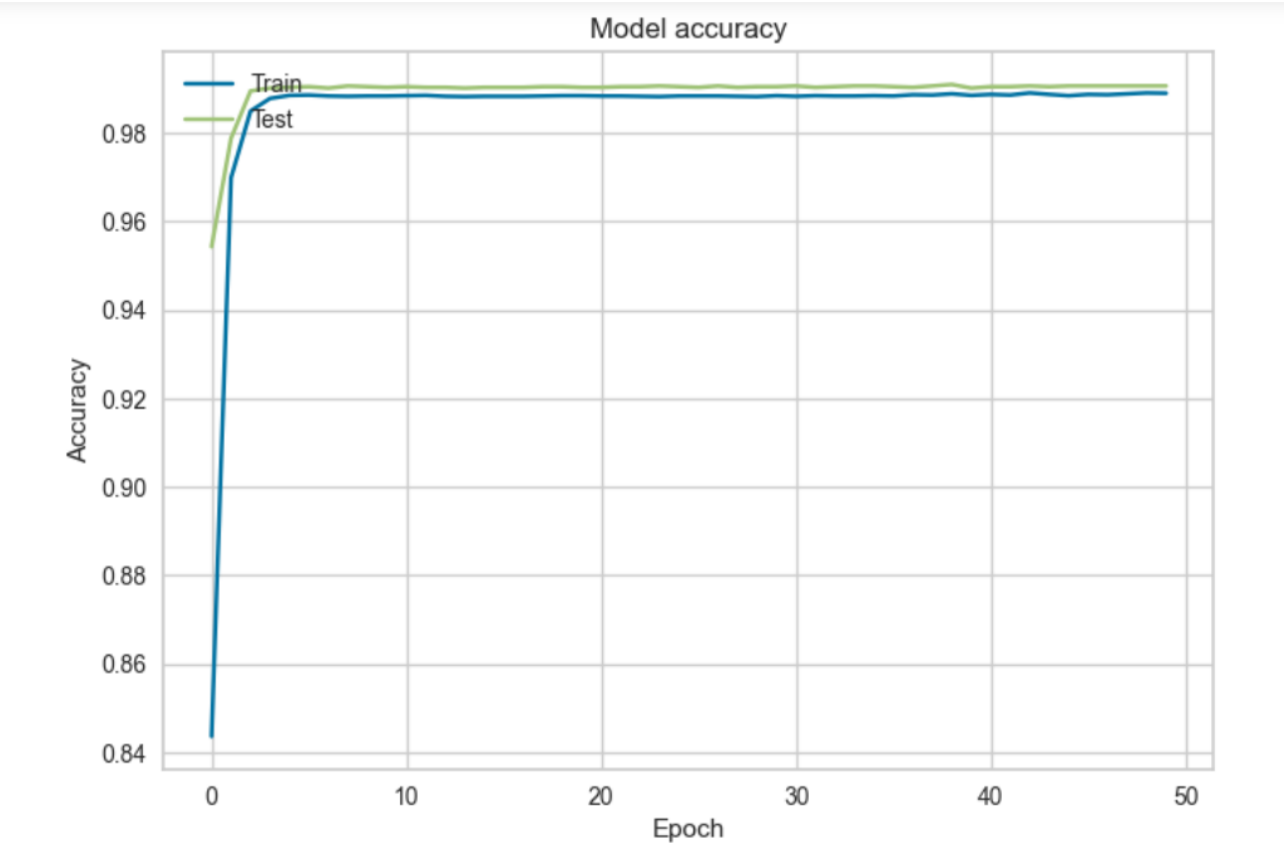
The left subplot shows a line plot of the feature values against time for occupied and vacant periods. Solid lines and the vacant periods by dashed lines mark the occupied periods.

The right subplot shows a box plot of the feature values for occupied, vacant, and overall periods.

These plots allow us to observe the relationship between each feature and occupancy status. They can help identify patterns and correlations that can inform the development of a model for room occupancy estimation.

IV. Results & Discussion:

Predicting occupancy using RNN with LSTM:




```
193/193 [==============================] - 1s 3ms/step
Accuracy of LSTM based RNN model: 99.06%
```

This code builds and trains an LSTM model to predict occupancy based on temperature, humidity, light, CO2, and humidity ratio data. It uses the Keras library to build the LSTM model and trains it using binary cross-entropy as the loss function and accuracy as the evaluation metric. The code also plots the model's training and validation accuracy and calculates the model's accuracy on the test data. The final output prints the accuracy of the LSTM model.

Machine Learning method for Time Series management:

Logistic Regression:

```
Confusion Matrix:
[[4630   61]
 [   6 1301]]

Accuracy Score: 0.9888296098699566

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      4691
           1       0.96      1.00      0.97      1307

    accuracy                           0.99      5998
   macro avg       0.98      0.99      0.98      5998
weighted avg       0.99      0.99      0.99      5998
```

Decision Tree:

```
Confusion Matrix:
[[4657   34]
 [  29 1278]]

Accuracy Score: 0.9894964988329443

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      4691
           1       0.97      0.98      0.98      1307

    accuracy                           0.99      5998
   macro avg       0.98      0.99      0.98      5998
weighted avg       0.99      0.99      0.99      5998
```

Random Forest:

```
Confusion Matrix:
[[4666   25]
 [  16 1291]]

Accuracy Score: 0.9931643881293765

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      1.00      4691
           1       0.98      0.99      0.98      1307

    accuracy                           0.99      5998
   macro avg       0.99      0.99      0.99      5998
weighted avg       0.99      0.99      0.99      5998
```

Accuracy scores of ML methods:

- Logistic Regression=98.88%
- Decision Tree=98.94%
- Random Forest=99.31%

V. Conclusion

The conclusion suggests that after comparing the performance of various machine learning models in predicting room occupancy using sensor data, the Random Forest Classifier was found to be the most accurate model, with an accuracy rate of 0.993. However, it is important to note that the model's accuracy may vary depending on the characteristics of new datasets, and it is essential to test the model's performance before applying it to real-world scenarios.

Additionally, the conclusion also highlights the successful implementation of an LSTM-based RNN approach in predicting room occupancy based on time series data. The model achieved a high accuracy rate of 99.03%, and it was effective in identifying patterns and trends in the sensor data to make accurate predictions. Overall, the study suggests that both traditional machine learning models like Random Forest and advanced techniques like LSTM-based RNN can be used to predict room occupancy accurately, depending on the specific characteristics of the dataset and the application requirements**.**

VI. Future scope:

There are several future scopes for Room Occupancy Estimation using Advanced Machine Learning Techniques. Some of them are:

Implementation of more advanced deep learning techniques: Currently, deep learning techniques like LSTM are being used to predict room occupancy. However, there are many more advanced deep learning techniques available that can be explored to achieve better accuracy.

Integration of other data sources: Room occupancy estimation can be enhanced by integrating other data sources such as weather data, events data, and social media data. These data sources can provide additional insights into occupancy patterns and help to improve accuracy.

Real-time occupancy monitoring: With the help of advanced machine learning techniques, real-time occupancy monitoring can be achieved. This can be useful in a variety of applications such as building automation, energy management, and security.

Implementation of Explainable AI: Explainable AI can help to understand the factors that contribute to room occupancy. This can help in identifying the root cause of occupancy patterns and help in making more informed decisions.

Integration with IoT devices: With the increasing adoption of IoT devices, integrating occupancy estimation with these devices can provide a more comprehensive view of occupancy patterns. This can help to optimize the use of resources and improve the overall efficiency of buildings.

Overall, the future of Room Occupancy Estimation using Advanced Machine Learning Techniques looks promising with the potential to achieve higher accuracy and provide valuable insights into occupancy patterns.

VII. References

[1] Singh, A. P., Jain, V., Chaudhari, S., Kraemer, F. A., Werner, S., and Garg, V. (2018). Machine Learning-Based Occupancy Estimation Using Multivariate Sensor Nodes. In 2018 IEEE Globecom Workshops (GC Wkshps).

[2] Singh, A. P. (2020). Machine Learning for IoT Applications: Sensor Data Analytics and Data ReductionTechniques (Master's Thesis). web2py.iiit.ac.in/research_centres/publications/view_publication/mastersthesis/872

[3] Jeon, Y., Cho, C., Seo, J., Kwon, K., Park, H., Oh, S., & Chung, I.-J. (2017). IoT-based occupancy detection system in indoor residential environments. Sensors (Basel, Switzerland), 17(10), 2271. https://doi.org/10.3390/s17102271

[4] De Santis, A., Cacace, J., & Marano, S. (2020). Room occupancy estimation through WiFi-based indoor localization and machine learning. Applied Sciences, 10(9), 3072. https://doi.org/10.3390/app10093072

[5] Gargiulo, F., Saggese, A., Vozella, A., & Verde, F. (2020). Machine learning for occupancy detection in smart buildings: A review. Energies, 13(18), 4804. https://doi.org/10.3390/en13184804

[6] Li, D., Li, Z., Li, Q., & Wang, X. (2020). Room occupancy detection based on deep learning for energy-efficient building applications: A review. Applied Energy, 279, 115859. https://doi.org/10.1016/j.apenergy.2020.115859

[7] Lu, S., Hui, P., Yu, W., & Ning, H. (2019). Deep learning-based indoor occupancy estimation using passive WiFi sensing. Sensors, 19(22), 4817. https://doi.org/10.3390/s19224817

[8] Zhang, X., Duan, Y., Jin, Y., & Xu, Y. (2020). An LSTM-based method for room occupancy prediction using environmental factors. Energy and Buildings, 210, 109766. https://doi.org/10.1016/j.enbuild.2020.109766