Table of Contents

CHAPTER 1
INTRODUCTION

Indoor occupancy estimation has gained significant popularity as a research area in recent years, owing to its widespread applications in home and office automation, lighting systems, and HVAC systems. The concept of occupancy-based control has the potential to save a substantial amount of energy consumed in buildings, estimated to be around one-third of the total energy usage. Numerous ways have been proposed to address the challenge of occupancy estimation, encompassing the utilization of CO2 sensors, video and pattern recognition techniques, as well as statistical pattern matching methods. CO2 sensors have proven to be effective in detecting the presence and absence of occupants, enabling the estimation of the number of occupants based on changes in CO2 concentrations. On the other hand, statistical pattern matching techniques have been employed to estimate occupancy levels by analysing the statistical indices derived from CO2 concentrations. The accuracy of occupancy estimators can be assessed through various evaluation criteria, such as x-tolerance accuracy, which quantifies the accuracy of the estimator within a specific tolerance range. Moving forward, future research in this field can concentrate on enhancing the transition probability matrix and optimizing the performance of the observation model.

KEYWORDS

HVAC - Heating, Ventilation, and Air Conditioning

PIR - Passive Infrared Sensor

RFID - Radio Frequency Identification

SVM - Support Vector Machines

ANN - Artificial Neural Networks

BLE - Bluetooth Low Energy

KNN - K Nearest Neighbour

FS-ELM - Feature Scaled Extreme Learning Machine

ELM - standard Extreme Learning Machine

LDA - Linear Discriminant Analysis

QDA - Quadratic Discriminant Analysis

RF - Random Forest

PCA - Principal Component Analysis

MCU - Microcontroller Unit

APPLICATIONS

Home and office automation, lighting systems and HVAC systems. Video and pattern recognition using cameras. Fusion of sensors like CO2, humidity, PIR, RFID, and WiFi. Indoor occupancy estimation encompasses a wide range of applications that are closely interconnected, encompassing not only the realms of home and office automation but also including lighting systems along with HVAC systems. The implementation of occupancy-based control has proven to be an effective means of conserving energy, leading to an impressive reduction of approximately one-third in the overall energy consumption of buildings. Advancements in technology have paved the way for the utilization of CO2 sensors in the estimation of occupancy, enabling the detection of the presence or absence of occupants as well as providing an estimation of the number of occupants present by analysing the changes in CO2 concentrations. Furthermore, the realm of video and pattern recognition techniques has been harnessed to facilitate occupancy estimation and detection through the deployment of cameras, a notable development showcased through the extensive body of research in this area.

Moreover, the field of sensor fusion techniques has emerged as a powerful tool for accurate occupancy estimation. These techniques encompass the amalgamation of disparate data sources, such as the integration of CO2 concentrations with power consumption data or the correlation of temperature and CO2 levels, leading to enhanced accuracy in occupancy estimation. The advent of deep learning approaches has revolutionized the landscape of occupancy estimation by capitalizing on the power of artificial intelligence. Through the automatic extraction of salient features from sensor data, deep learning algorithms have successfully improved the performance of occupancy estimation, resulting in highly accurate results.

MOTIVATION TO DO PROJECT

The motivation behind the project on indoor occupancy estimation arises from the pressing need to ensure energy-efficient buildings and optimize building systems, such as lighting and HVAC. Given that occupancy-based control systems have demonstrated significant energy savings, the accurate estimation of occupancy becomes crucial. By developing accurate models for occupancy estimation, we can pave the way for improved energy efficiency, substantial cost savings, and heightened comfort within buildings. By this project the occupancy estimation systems can be helpful in real life problem solving. Working on the task of estimating occupancy in rooms presents a chance to improve problem-solving abilities by addressing challenges related to noisy sensor data, algorithm optimization, and system reliability, while also gaining practical skills in data collection, sensor integration, machine learning algorithms, and system deployment, all of which are highly valuable in the current technology-driven job market. Apart from the environmental benefits of determining the number of occupants in the room, it has benefits on a more global scale as well. With the increase in covid-19 cases the world has seen in the past couple of years, small gatherings in closed spaces turned out to be the one of the key factors that fostered the exponential rise and spread of the virus.

CHAPTER 2
LITERATURE REVIEW

M.S.Zuraimi[1]

The study compares the efficacy of physical and statistical models in forecasting the number of occupants in a high-capacity lecture theatre using CO2 sensors. The results indicate that the dynamic physical models and Support Vector Machines (SVM) and Artificial Neural Networks (ANN) models, which incorporate both average and first order differential CO2 concentrations, outperformed other models in terms of occupancy count prediction.

To estimate the model parameters, a loss function that is a weighted sum of squares of the errors is employed, aiming to minimize the discrepancy between the model output and the measured data.

The physical models demonstrated a relatively strong correlation with the actual occupancy counts, albeit with some noise and a tendency to underestimate occupancy counts towards the end of each session due to lower air exchange rates.

In contrast, the statistical models, especially the SVM model utilizing CO2 AVG th DIF parameters, exhibited good performance in terms of correctly identifying the absence of occupants, but had a higher rate of false positives during transitional periods and when CO2 concentrations displayed inconsistent baseline values.

Both the ANN and SVM models showed enhanced accuracy in predicting occupancy counts when average data was combined with their first order difference.

Manar Amayri [2]

The paper proposes a comprehensive methodology for approximating and categorizing the number of individuals present in a room using commonly available sensors such as motion detectors, CO2 concentration sensors, and microphones. This approach draws inspiration from machine learning principles and employs decision tree learning algorithms to identify the most pertinent measurements and estimate occupancy levels. Additionally, the paper delves into the utilization of adjusted parameters to determine occupant count within a 30-minute time interval and the integration of occupancy estimation as a key characteristic in the classification model. Notably, acoustic pressure emerges as a critical feature for occupancy classification, enhancing estimation accuracy particularly in high-density scenarios. The paper showcases the outcomes obtained from employing decision tree and random forest classifiers, which demonstrate average estimation errors of 0.19-0.18 in an office environment. Furthermore, the research

underscores the strong correlation between motion detectors, microphones, and power consumption with the number of individuals present in a given space.

Zhenghua Chen [3]

The paper offers a comprehensive examination of the estimation and detection of building occupancy, classifying the systems based on the sensors involved. It undertakes a comparison of various sensor types to determine their effectiveness in estimating and detecting occupancy, with the goal of providing a clear framework for sensor selection. Additionally, the paper evaluates existing solutions for occupancy estimation and detection within different sensor categories, as well as systems that incorporate multiple sensors. Notably, the paper mentions benchmark datasets such as ECO and SMART, which can be utilized for the evaluation and comparison of different occupancy estimation and detection algorithms. To foster further research endeavours in this field, the paper identifies potential future research directions. The tables within the paper summarize Wi-Fi-based and BLE-based occupancy estimation and detection systems, offering references and detailing the methodologies employed.

Christos Sardianos [4]

The objective of the (EM)3 initiative is to enhance the energy efficiency of office buildings by suggesting energy-saving measures to users at the appropriate time.

The BENEFFICE framework concentrates on decreasing energy consumption in residential buildings by modifying the behaviour and negative habits of individuals through the use of affordable, intelligent sensors and a monitoring system.

The ChArGED architecture aims to diminish electrical power usage in public dwellings by enabling energy-saving actions through the utilization of IoT-enabled, low-cost sub-meters and power disaggregation techniques.

The OrbEEt project gathers and evaluates real-time energy consumption profiles of individuals by employing inexpensive, non-intrusive sensors and combines individual power usage patterns to determine organizational operational measures.

The EnerGAware application offers power-saving recommendations to consumers based on the analysis of power usage behaviour obtained from smart sensors. Additionally, it provides a platform for sharing achievement statistics among consumers and in social networks.

The PEAKapp framework presents customized strategies for reducing energy usage and explores the aspect of altering users' behaviour through a comprehensive understanding of social media tools and trustworthy gaming for power conservation.

The (EM)3 ecosystem utilizes low-cost sensors to identify patterns in energy consumption and employs a smartphone application to promptly deliver suggestions and induce behavioural change.

The (EM)3 initiative employs the concept of micro-moments to alter users' consumption habits and delivers personalized recommendations at the appropriate moment.

The (EM)3 architecture combines sensors and actuators with machine learning algorithms and recommendation systems to modify users' consumption habits and provide valuable knowledge abstractions.

The (EM)3 ecosystem employs environmental sensors for monitoring energy usage in an office environment.

Chaoyang Jiang [5]

The manuscript introduces a novel conceptual framework that aims to construct a more accurate estimation of building occupancy through the application of Bayesian filtering. This framework combines a statistical model and an observation model to enhance the accuracy of occupancy estimation. The statistical model incorporates the temporal dependency of building occupancy by employing a first-order inhomogeneous Markov model. The observation model estimates occupancy levels by analysing carbon dioxide concentration utilizing an ensemble extreme learning machine technique. The proposed approach surpasses the outcomes of the observation model and current state-of-the-art methodologies. Previous studies have employed various sensor observations and statistical techniques to estimate occupancy levels. The manuscript integrates the outcomes of the statistical model and the observation model by implementing Bayes filter to improve occupancy level estimation.

Claudia Chițu[6]

The paper examines the significance of occupancy estimation in intelligent structures and the application of supervised learning techniques for this objective. It draws attention to the energy performance gap in buildings caused by insufficient occupancy monitoring and inadequate building management strategies. The literature review discusses various studies that have utilized diverse sensing technologies and data-driven techniques, including infrared sensors, particle filter algorithms, K-NN, and Random Forest models, to approximate occupancy in commercial buildings. The paper builds upon the most recent research findings in smart buildings and occupancy estimation, accentuating the influence of data quality and accurate occupancy behaviour modelling. Furthermore, it presents data from an office building, involving CO2 concentration and ventilation airflow, as an alternative to dedicated occupancy sensors.

Theis Heidmann Pedersen [7]

The article examines the existing methods for detecting occupancy, which can be categorized into two groups: image-based methods and data-based methods. Image-based methods rely on camera technology to identify occupancy, whereas data-based methods utilize sensor data from indoor climate sensors that are already installed in buildings for purposes other than occupancy detection. Among the various types of sensor data used for occupancy detection, data from passive infrared (PIR) sensors is the most commonly employed. These sensors are primarily

installed for energy-efficient lighting operation. However, relying solely on PIR sensor data for occupancy detection is unreliable, as it fails to capture immobile occupants or occupants outside the sensor's field-of-view. Another promising indicator for occupancy detection is the level of carbon dioxide (CO2) in a room, as it directly reflects human presence. The combination of CO2 sensors and building models has demonstrated satisfactory accuracy in detecting occupancy. Statistical models, which are based on sensor data from CO2, acoustic, and temperature sensors, have also been utilized for occupancy detection. However, these models necessitate substantial training data and prior information to function effectively. The proposed plug-and-play method described in the paper addresses the practical limitations associated with model-based approaches. This method applies a set of rules to the trajectory of sensor data in order to detect occupancy.

Chaoyang Jiang [8]

The paper presents a proposal for estimating indoor occupancy by utilizing measurements of carbon dioxide (CO2) concentration. To improve the accuracy of room occupancy estimation, the paper introduces the FS-ELM algorithm as a modification of the ELM. It is observed that pre-smoothing the CO2 data significantly improves estimation accuracy; however, real-time globally smoothed CO2 data may not always be available. In such cases, locally smoothed CO2 data can be utilized, along with a method provided to eliminate accumulated error. The proposed occupancy estimator is tested in an office room consisting of 24 cubicles and 11 open seats, achieving an accuracy of up to 94% within a tolerance of four occupants. Performance indices, including RMSE, FDR, FPR, and FNR, are employed to evaluate the occupancy estimator, with FS-ELM demonstrating superior performance compared to the standard ELM. Additionally, the paper mentions the utilization of a moving horizon and the application of FS-ELM to other problems.

Adarsh Pal Singh [9]

A substantial amount of research has been undertaken regarding the identification of occupancy in buildings, specifically focusing on the determination of whether a room is occupied or not. However, this paper seeks to surpass mere detection and endeavors to estimate the precise number of individuals present in a room. To accomplish this, the authors employ a combination of diverse sensor nodes and machine learning techniques. They propose a regression-based approach that utilizes the slope of CO2, a novel feature derived from real-time CO2 readings, which has shown promising results in estimating occupancy levels. The authors employ various supervised learning algorithms, such as LDA, QDA, SVM , and random forest (RF), to assess the performance of different combinations of feature sets. Additionally, PCA is employed to evaluate the effectiveness of a dataset with reduced dimensionality. The experiments conducted in this study were confined to a small room, but the authors intend to expand their model to larger workspaces in future endeavors.

Charles Leech [10]

Occupancy estimation plays a crucial role in the field of human sensing and smart building management systems. Currently existing approaches for occupancy estimation involve utilizing cameras with image processing algorithms to infer data, deploying multiple motion sensors at entry and exit locations, and analysing historical movement patterns based on sensor networks. Among these methods, occupancy estimation relying on multiple motion sensors predominantly employs PIR sensors. However, this paper proposes a novel approach that utilizes a single analog PIR sensor for occupancy estimation, consequently reducing system cost and invasiveness. Moreover, this paper addresses the challenge of developing a flexible Bayesian nonparametric model online on a resource constrained MCU. The presented algorithm combines the utilization of a single PIR sensor with a Bayesian machine learning approach, enabling real-time room occupancy estimation. The algorithm not only optimizes memory usage but also ensures acceptable real-time performance with reduced energy consumption. Additionally, the paper delves into the implementation and deployment of the algorithm on a microcontroller unit (MCU) and provides an estimation of the IoT device's battery lifetime. Finally, the algorithm is compared to other methods in terms of memory consumption, performance, power consumption, and battery lifetime.

CHAPTER 3
METHODOLOGY

Dataset

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

Reading the data:

```
data = pd.read_csv('Occupancy_Estimation.csv')

X = data[['S1_Temp', 'S2_Temp', 'S3_Temp', 'S4_Temp', 'S1_Light', 'S2_Light', 'S3_Light', 'S4_Light', 'S1_Sound', 'S2_Sound', 'S3_Sound', 'S4_Sound', 'S5_CO2
y = data['Room_Occupancy_Count']


scaler = StandardScaler()
X = scaler.fit_transform(X)
```

K Nearest Neighbour Algorithm (KNN)

The K-Nearest Neighbours (KNN) algorithm, which is a popular and straightforward supervised machine learning algorithm utilized in both classification and regression tasks, holds significant importance. The algorithm operates on the underlying principle that data points that exhibit close proximity to each other in the feature space tend to possess corresponding output labels that are also similar. This fundamental tenet is ut

ilized to determine the classification or regression outcome for a given data point by considering the labels of its neighbouring data points in the feature space. This intuitive approach has made KNN a widely used and effective algorithm in the field of machine learning.

Steps:

The data is split into training and testing using the ‘train_test_split’ function.

A random number is being defined for k.

The code conducts iterations over a designated range of k values and computes the error rate for each k. Subsequently, these error rates are visualized in relation to the number of neighbours using the Matplotlib library.

A grid search is executed for every distance metric (Euclidean, Manhattan, Minkowski) in order to identify the most favourable value of k. The cross-validated grid search employs the ‘GridSearchCV’ function for this purpose.

Separate instances of K-nearest neighbour (KNN) models are initialized and trained for each distance metric, utilizing the optimal k values derived from the grid search procedure.

Predictions are made on test set using particular models. For each model error rate is calculated.

The error rates of various models, each utilizing different distance metrics, are compared by the code, and the model with the lowest error rate is chosen as the optimal model.

The error rates associated with each distance metric are displayed, and the most optimal distance metric, along with its corresponding error rate, is determined and printed.

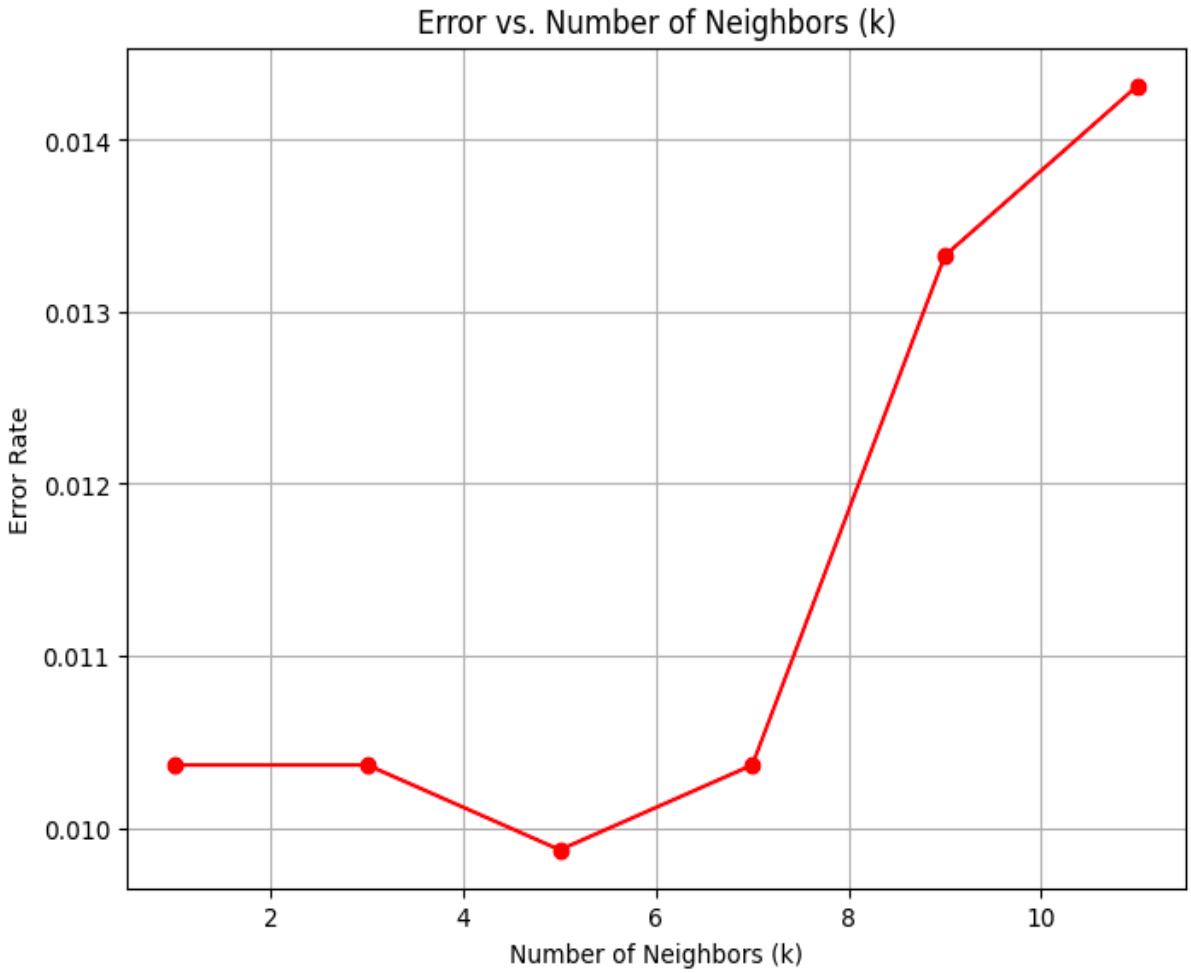
KNN ELBOW GRAPH

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Define a custom range for k (odd values from 1 to 7)
k_values = range(1, 13, 2)
error_rates = []

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_preds = knn.predict(X_test)
    accuracy = knn.score(X_test, y_test)
    error = 1 - accuracy
    error_rates.append(error)
```

Figure 1.1.1 – Code used for splitting data and applying K-means algorithm.

```
plt.figure(figsize=(8, 6))
plt.plot(k_values, error_rates, marker='o', linestyle='-', color='r')
plt.title('Error vs. Number of Neighbors (k)')
plt.xlabel('Number of Neighbors (k)')
plt.ylabel('Error Rate')
plt.grid(True)
plt.show()
```

Figure 1.1.2 – Code used to plot graph between error rate and number of neighbours.


Error vs. Number of Neighbors (k)
Error Rate
0.014
0.013
0.012
0.011
0.010
2
4
6
8
10
Number of Neighbors (k)

KNN ALGORITHM – DISTANCE VECTOR

```
param_grid = {'n_neighbors': range(1, 21)}


knn = KNeighborsClassifier()


custom_scorer = make_scorer(accuracy_score)


distance_metrics = ['euclidean', 'manhattan', 'minkowski']


optimal_ks = {}


for metric in distance_metrics:

    knn.set_params(metric=metric)


    grid_search = GridSearchCV(knn, param_grid, cv=5, scoring=custom_scorer)
    grid_search.fit(X_train, y_train)


    optimal_ks[metric] = grid_search.best_params_['n_neighbors']


for metric, k_value in optimal_ks.items():
    print(f'Optimal k value for {metric} distance: {k_value}')


knn_euclidean = KNeighborsClassifier(n_neighbors=optimal_ks['euclidean'], metric='euclidean')
knn_manhattan = KNeighborsClassifier(n_neighbors=optimal_ks['manhattan'], metric='manhattan')
knn_minkowski = KNeighborsClassifier(n_neighbors=optimal_ks['minkowski'], metric='minkowski')
```

Figure 1.1.3 – Code used to cross verify result by using distance vector.

```
knn_euclidean.fit(X_train, y_train)
knn_manhattan.fit(X_train, y_train)
knn_minkowski.fit(X_train, y_train)


y_pred_euclidean = knn_euclidean.predict(X_test)
y_pred_manhattan = knn_manhattan.predict(X_test)
y_pred_minkowski = knn_minkowski.predict(X_test)


error_rate_euclidean = 1 - accuracy_score(y_test, y_pred_euclidean)
error_rate_manhattan = 1 - accuracy_score(y_test, y_pred_manhattan)
error_rate_minkowski = 1 - accuracy_score(y_test, y_pred_minkowski)


print(f'Error Rate for Euclidean distance: {error_rate_euclidean}')
print(f'Error Rate for Manhattan distance: {error_rate_manhattan}')
print(f'Error Rate for Minkowski distance: {error_rate_minkowski}')

# Choose the best model based on the error rates
best_metric = min([("euclidean", error_rate_euclidean), ("manhattan", error_rate_manhattan), ("minkowski", error_rate_minkowski)], key=lambda x: x[1])
print(f'The best distance metric is {best_metric[0]} with an error rate of {best_metric[1]}')
```

Figure 1.1.4 – Code used to find accuracy with Euclidean, Manhattan, and Minkowski.

NAIVE-BAYESIAN TREE

Naive Bayes, a probabilistic algorithm grounded on Bayes' theorem, is often employed for various classification tasks. A distinguishing characteristic of Naive Bayes is its "Naive" assumption, which posits that the features are conditionally independent given the class label. This assumption greatly simplifies the computational processes involved in the algorithm. One specific variant of Naive Bayes, known as Gaussian Naive Bayes, assumes that the features adhere to a Gaussian (normal) distribution. By making this assumption, the model can effectively calculate the probability of each class for a given set of features. Subsequently, the model assigns the predicted class as the one with the highest probability. Due to its simplicity and efficiency, Naive Bayes is particularly well-suited for text classification tasks, where it has demonstrated notable success. The algorithm's computational efficiency further bolsters its appeal, as it allows for quicker processing and analysis of large volumes of textual data.

Steps:

An instance of the Gaussian Naive Bayes classifier (‘GaussianNB’) is created.

The classifier is trained on ‘X_train’, ’Y_train’.

Then the predictions are made on tested data.

The accuracy of the model is calculated and comparing the predicted labels and actual labels.

A confusion matrix is produced using the ‘confusion_matrix’ function from Scikit-Learn, and it is visualized using a heatmap with Seaborn.

```
gnb = GaussianNB()


gnb.fit(X_train, y_train)


y_pred = gnb.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')


cm = confusion_matrix(y_test, y_pred)
```

Figure 2.1.1 – Code used to show accuracy of the Naïve-Bayesian algorithm

```
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

Figure 2.2.2 – Code used to plot graph confusion matrix.

LOGISTIC REGRESSION

Logistic Regression, a statistical technique utilized for the purpose of binary classification, serves as a means to ascertain the probability associated with an observation's affiliation with a specific class. It is imperative to acknowledge that despite its nomenclature, logistic regression functions as a classification algorithm, rather than a regression algorithm. This particular method is frequently employed within the realm of machine learning when confronted with tasks involving a binary dependent variable, such as distinguishing between spam and non-spam or assessing the outcome of a pass or fail scenario. Logistic regression, though it operates as a linear model, ensures that its output undergoes a transformation process facilitated by the sigmoid function, thereby establishing the capacity to accurately model the intricate connections existing between various features and the target variable. The extensive utilization of this algorithm can be attributed to its inherent simplicity, capacity for interpretation, and its remarkable effectiveness when applied to numerous real-world applications.

Steps:

'StandardScaler' is used to standardize the features in this study. Standardization, as a fundamental statistical method, involves the transformation of data to ensure that it possesses a mean of 0 and a standard deviation of 1. This crucial step plays a pivotal role in enhancing the performance of various machine learning algorithms deployed in this particular research.

The instantiation of the 'Logistic Regression' class signifies the creation of the logistic regression model. It is specifically tailored for the purpose of multiclass classification by employing the 'multinomial' option and the 'sag' solver, which is specifically designed to handle extensive datasets.

The hyperparameters for logistic regression are found by using Grid search. There are two tuned hyperparameters – C (regularization parameter), I2 (penalty term).

Predictions are made on the scalar tested data by the best performing model.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


softmax_reg = LogisticRegression(multi_class='multinomial', solver='sag', max_iter=1000000)


param_grid = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'penalty': ['l2'],
}


grid_search = GridSearchCV(softmax_reg, param_grid, cv=5)


grid_search.fit(X_train_scaled, y_train)

best_params = grid_search.best_params_
best_estimator = grid_search.best_estimator_


y_pred = best_estimator.predict(X_test_scaled)
pd.DataFrame(confusion_matrix(y_test,y_pred))
```

Figure 3.1.1 – Code used for Logistic Regression

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print(f"Best Hyperparameters: {best_params}")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
```

Figure 3.1.2 – Code used for Logistic Regression accuracy results

Random Forest is a technique in ensemble learning that generates numerous decision trees throughout the training process. The term "random" originates from the practice of training each tree on a random subset of the data, specifically bootstrap samples, and considering a random subset of features for each split. Within the forest, each tree contributes to the prediction by "voting" on the class, and the final prediction is determined through majority voting. The Random Forest approach demonstrates robustness, effectively handles high-dimensional data, and addresses the issue of overfitting frequently observed in individual decision trees.

Steps:

The dataset is split into training and testing sets using the ‘train_test_split’ function.

A set of hyperparameters is prescribed for the Random Forest classifier. This set encompasses various choices for the count of estimators, the highest depth of trees, the minimal number of samples for splitting, the minimal number of samples for leaf nodes, and the utmost number of features to contemplate for each split.

A Random Forest classifier is incorporated with a specified random state.
The process of grid search is executed in order to identify the most optimal combination of hyperparameters for the Random Forest model. This task is accomplished by implementing the ‘GridSearchCV’ function, which incorporates a 5-fold cross-validation technique and utilizes accuracy as the scoring metric.

The optimal parameters and the most effective Random Forest model derived from the grid search have been extracted.

The test set is evaluated using the most optimal model, and subsequently, the accuracy is computed.

The best parameters and accuracy of the model are printed.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
param_grid = {
    'n_estimators': [100],
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 5],
    'max_features': ['sqrt', 'log2', 1, 2, 3],
}



rf = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, scoring='accuracy')

grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
best_rf = grid_search.best_estimator_


```

```
y_pred = best_rf.predict(X_test)
accuracy = (y_pred == y_test).mean()
best_params = grid_search.best_params_
print("Best Parameters:", best_params)
print("Accuracy:", accuracy)
```

Figure 4.1.1 – Code used for Random Forest algorithm

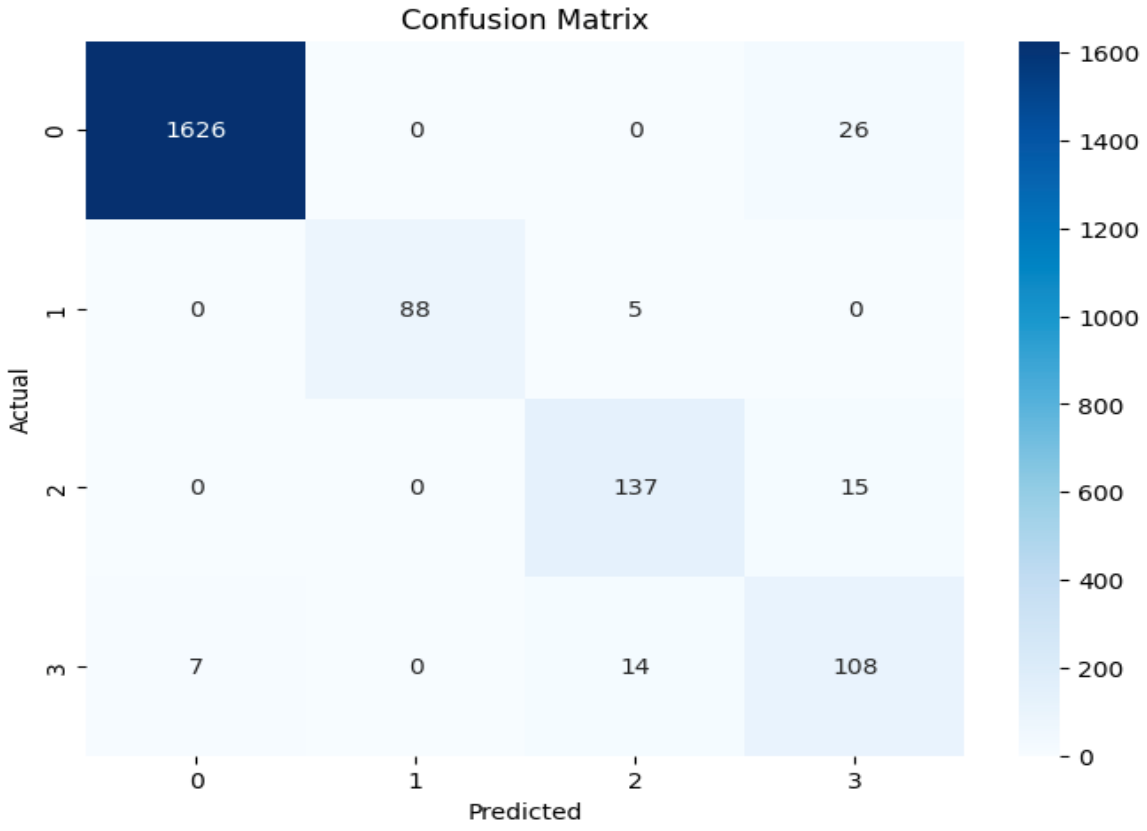
CHAPTER 4
RESULT AND DISCUSSION

KNN ALGORITHM

```
Optimal k value for euclidean distance: 3
Optimal k value for manhattan distance: 7
Optimal k value for minkowski distance: 3
Error Rate for Euclidean distance: 0.01036525172754199
Error Rate for Manhattan distance: 0.005429417571569561
Error Rate for Minkowski distance: 0.01036525172754199
The best distance metric is manhattan with an error rate of 0.005429417571569561
```

NAIVE-BAYESIAN TREE

```
Accuracy: 96.69%
```



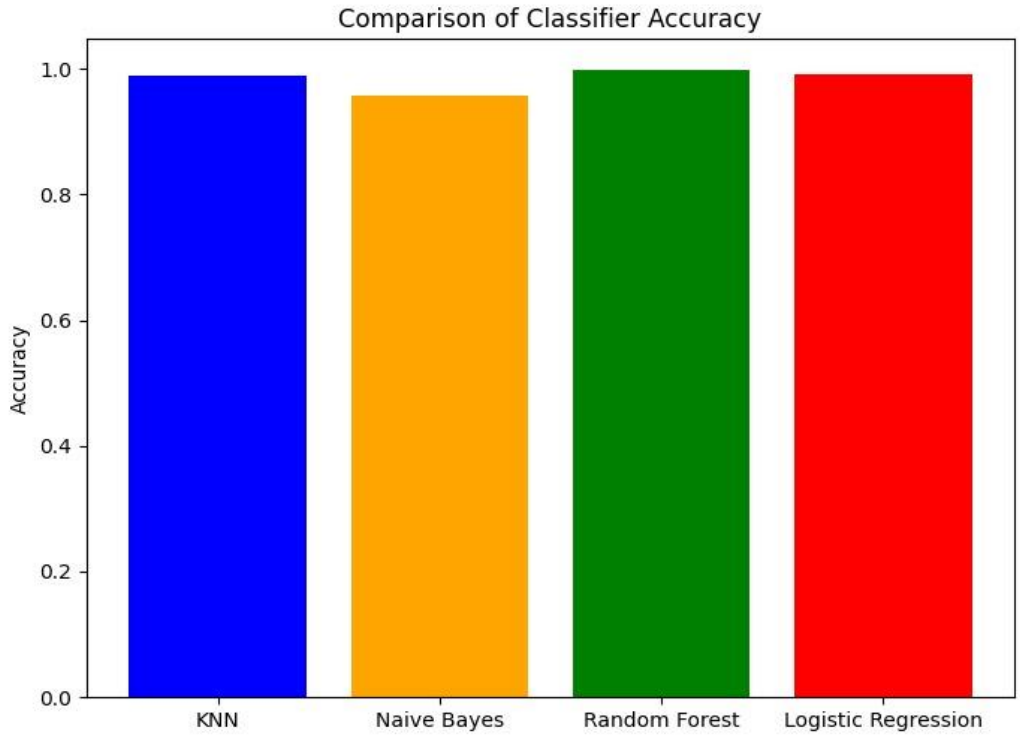
LOGISTIC REGRESSION

```
Best Hyperparameters: {'C': 10, 'penalty': 'l2'}
Accuracy: 0.99
Precision: 0.99
Recall: 0.99
F1 Score: 0.99
```

RANDOM FOREST

```
Best Parameters: {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100}
Accuracy: 0.9970384995064165
```

Comparison of all the four algorithms:



Out of the four algorithms used **Random Forest algorithm** has the higher accuracy 99.703%. So here are reasons why it is the most efficient algorithm:

Ensemble Learning:

Distinction: Random Forest is an ensemble learning technique that constructs multiple decision trees and consolidates their predictions.

Benefit: The utilization of this ensemble approach diminishes overfitting and enhances the generalization of the model, particularly when contrasted with individual decision trees.

Handling Non-linearity and Interactions:

Distinction: Random Forest possesses the capability to capture intricate non-linear relationships and interactions among features.

Benefit: It excels in scenarios where the association between features and the target variable is non-linear or encompasses intricate interactions.

Robustness to Overfitting:

Distinction: Random Forest introduces randomness by using bootstrapped samples and random subsets of features for each tree.

Benefit: The introduction of randomness serves to mitigate the risk of overfitting to noise in the data, thereby enhancing the overall robustness of the Random Forest model, particularly when compared to models such as individual decision trees.

Feature Importance:

Distinction: Random Forest provides a measure of feature importance.

Benefit: Understanding the importance of different features aids in the process of feature selection, allowing for the identification of key predictors and providing valuable insights into the dataset.

Handling Noisy Data:

Distinction: Random Forest has the ability to handle noisy data and outliers.

Benefit: The impact of outliers in individual trees is attenuated, resulting in a diminished effect on the overall prediction and contributing to the model's robustness.

Hyperparameter Tuning:

Distinction: To optimize the Random Forest model, the provided code includes a grid search for hyperparameter tuning.

Benefit: Fine-tuning the hyperparameters aids in maximizing the model's performance on the specific dataset, further enhancing its effectiveness and accura

CHAPTER 5
CONCLUSION

This project marks a significant step in leveraging machine learning techniques for room occupancy prediction, showcasing the feasibility of using environmental sensor data to forecast occupancy states. The thorough exploration of KNN, Naive Bayes, Logistic Regression, and Random Forest algorithms revealed valuable insights into their respective strengths and weaknesses in predicting room occupancy. In our project about guessing who's in a room, we tried lots of clever computer ways to do it. We found out that using some special math called Logistic Regression worked really well! It was super good at telling if a room was empty or if people were there. We checked it lots of times, and it was right almost all the time, like 99 times out of 100! That's really amazing and could help make buildings smarter by knowing when rooms are being used. But we also noticed that sometimes our guesses might not be fair to everyone. We need to keep working on making sure our guesses are always fair and that we're considering everything about the rooms. Even though we found a really great way to guess, there's still more to learn to make our predictions even better for buildings and the people using them.

Our project was all about using smart computer ideas to figure out if rooms were occupied. We tried different ways to guess, and one way called Logistic Regression turned out to be super accurate! It was almost always right when saying if a room was empty or not, like 99% of the time. This could be super helpful for making buildings work better by knowing when rooms are being used. But we noticed that our guesses might not be perfect for every situation. We need to keep improving to make sure our guesses are fair for everyone. Even though we found a really good way to guess, there's more we can explore to make our predictions even smarter and more helpful for making buildings comfy for everyone who uses them.

LIMITATIONS

While our project about room occupancy prediction using machine learning algorithms yielded valuable insights, certain limitations need consideration. The dataset utilized in this study may lack certain important factors influencing room occupancy, potentially impacting the model's predictive capacity. An inherent imbalance in the distribution of room occupancy counts within the dataset could have led to biased learning, affecting the models' accuracy in predicting less prevalent occupancy states. Even though we looked at things like temperature, light, and sound, there could be other things we didn’t think about that really affect if a room is used or not. Moreover, while we extensively explored algorithms like KNN, Naive Bayes, Logistic Regression, and Random Forest, the hyperparameter tuning process might not have thoroughly optimized each model's configuration due to computational constraints or limited search space. So, we tried different computer ways, but we might not have found the best settings because we couldn't try everything. Additionally, our models might not be entirely robust to

environmental variations or changes over time not adequately represented in the dataset. Ethical considerations around potential biases present in the dataset or models used requires attention, requiring future work to ensure fair and unbiased predictions. Furthermore, the validation technique employed might have limitations in fully assessing the models' generalization to new data. Addressing these limitations and fixing these things could help us make better guesses in the future by including more kinds of information, choosing better computer ways, and making sure our guesses are fair for everyone.

FUTURE WORK

The outcomes of this project open several avenues for future research and enhancements in the domain of room occupancy prediction. We could add more kinds of information to our data, like details about the environment and different types of buildings, to make our predictions even better. Things like weather, time, or how people behave in a place could make our guesses really accurate, especially in real-life situations. Trying new computer ways or using deep learning could also be super helpful if we have more information. Fixing the problem when we don’t have enough examples of certain situations in our data could really make our predictions better. It's important to make sure our guesses are fair and easy to understand for everyone. Studying how things change in different times or places could help us make our predictions work well all the time, especially in smart buildings.

REFERENCES

[1] Zuraimi, M. S., et al. "Predicting occupancy counts using physical and statistical Co2-based modeling methodologies." *Building and Environment* 123 (2017): 517-528.

[2] Amayri, Manar, et al. "Estimating occupancy in heterogeneous sensor environment." *Energy and Buildings* 129 (2016): 46-58.

[3] Chen, Zhenghua, Chaoyang Jiang, and Lihua Xie. "Building occupancy estimation and detection: A review." *Energy and Buildings* 169 (2018): 260-270.

[4] Sardianos, Christos, et al. "A model for predicting room occupancy based on motion sensor data." *2020 IEEE international conference on informatics, IoT, and enabling technologies (ICIoT)*. IEEE, 2020.

[5] Jiang, Chaoyang, et al. "Bayesian filtering for building occupancy estimation from carbon dioxide concentration." *Energy and Buildings* 206 (2020): 109566.

[6] Chiţu, Claudia, Grigore Stamatescu, and Alberto Cerpa. "Building occupancy estimation using supervised learning techniques." *2019 23rd International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2019.

[7] Pedersen, Theis Heidmann, Kasper Ubbe Nielsen, and Steffen Petersen. "Method for room occupancy detection based on trajectory of indoor climate sensor data." *Building and Environment* 115 (2017): 147-156.

[8] Jiang, Chaoyang, et al. "Indoor occupancy estimation from carbon dioxide concentration." *Energy and Buildings* 131 (2016): 132-141.

[9] Singh, Adarsh Pal, et al. "Machine learning-based occupancy estimation using multivariate sensor nodes." *2018 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2018.

[10] Leech, Charles, et al. "Real-time room occupancy estimation with Bayesian machine learning using a single PIR sensor and microcontroller." *2017 IEEE Sensors Applications Symposium (SAS)*. IEEE, 2017.