K-Nearest Neighbor Classifier for Room Occupancy Estimation (July 2024)

Pragyan Bhattarai(THA077BEI030)[1], Prashant Raj Bista(THA077BEI032)[1]
bhattaraipragyan.pb@gmail.com , prashant.bista.18@gmail.com
[1]Department of Electronics and Computer Engineering, IOE, Thapathali Campus, Kathmandu 44600, Nepal

ABSTRACT This work presents a thorough analysis of the room occupancy using the K-Nearest Neighbor (KNN) classifier. The KNN algorithm is applied to the dataset "Room Occupancy Estimation", which includes features such as temperature, light, sound, CO2 levels, and PIR (Passive Infrared Readings) from different sensors. Various configurations of the KNN model (including different values of K, distance metrics and weighting schemes) were tested to optimize its performance. The optimal value for K was found to be 3 using Grid Search Cross-Validation technique. The KNN classifier with K=3 and the Manhattan distance metric showed the best performance among other configurations. The accuracy of this model was 99.65%, which is an exceptional result. The Area Under Curve for Receiver Operating Characteristics is obtained to be near-perfect 1.00, indicating the phenomenal performance of the model. The study shows the efficacy of the KNN algorithm for room occupancy estimation and also emphasizes the importance of parameter tuning in enhancing the model performance.

INDEX TERMS Euclidean Distance, K-Nearest Neighbor, Manhattan Distance, Room Occupancy Estimation

I INTRODUCTION

In the context of modern technology, one of the important tasks lies in the optimization and careful use of available resources. Real-time tracking of the number of people in a room can significantly enhance resource efficiency. Monitoring room occupancy can lead to the efficient use of lights, air conditioning, and temperature control. The UCI dataset includes data from multiple sensors, providing insights into the number of people in a room. This data can be used to develop smart systems that adjust environmental controls automatically. Furthermore, it can help in energy conservation efforts and improve overall sustainability. By utilizing such datasets smarter more responsive buildings that adapt to real-time occupancy levels can be developed.

Estimating room occupancy presents several significant challenges due to the variability in environmental conditions. Factors such as temperature, humidity, light intensity, and CO2 levels are subject to constant change influenced by external weather conditions, equipment use, and human activities. The sensors may have calibration errors or inaccuracies, which complicates the task of obtaining reliable occupancy readings. Managing these variations requires sophisticated data prepossessing and robust modeling techniques to ensure the system can handle the complexity and variability inherent in real-world environments.

A PROBLEM STATEMENT

Accurate classification of room occupancy is crucial for optimizing energy efficiency and managing resources effectively. However, without precise data on the number of occupants, smart building systems cannot adjust heating, cooling, and

lighting systems to match actual needs. This leads to energy waste, higher operational costs, and increased environmental impact. There is a need for efficient resource management to be further enhanced through participation in demand response programs. Real-time occupancy data is required to align energy usage with fluctuating supply and demand conditions. Without this data, it is challenging to optimize both cost and energy consumption. Additionally, the lack of accurate occupancy data can result in sub-optimal comfort levels for occupants. It can also hinder the ability to maintain safety protocols by preventing overcrowding in rooms. Solving these issues requires the implementation of systems that can provide accurate, real-time data on room occupancy.

The K-Nearest Neighbors algorithm is a versatile and widely employed machine learning method that classifies data based on the distance between neighboring data points. Unlike other algorithms, KNN does not make any assumptions about the underlying data distribution. It is a non-parametric technique that generates predictions based on the similarity of data points within a given dataset. When the value of 'K' is chosen appropriately, KNN is less sensitive to outliers compared to other algorithms. The KNN algorithm operates by identifying the K nearest neighbors to a given data point using a specified distance metric. The class or value of the data point is then determined by the majority class or value among these K neighbors.

B OBJECTIVES

The main objective is:

- To accurately classify the number of peoples in a room using K-Nearest Neighbour Algorithm.

II RELATED WORK

The paper [1] is the base paper for this dataset. In this paper, authors collect data from multiple low-cost, non-intrusive environmental sensors and investigates machine learning-based occupancy estimation to improve energy efficiency in buildings. The authors deployed a wireless sensor network with 7 nodes in a small office room, collecting data from CO2, temperature, illumination, sound, and motion sensors over 4 days time period. They processed the data, engineered new features like CO2 slope, and applied various supervised learning algorithms including LDA, QDA, SVM, and Random Forest. The study achieved impressive results, with the best performance of 98.4% accuracy and 0.953 F1 score using SVM with RBF kernel on the full feature set. Notably, they demonstrated good performance (93.4% accuracy) even without light sensor data, which they cautioned against over-relying on. The authors also explored dimensionality reduction using PCA, maintaining 92% accuracy with just 4 principal components. While the study was limited to a small room, the authors plan to extend their approach to larger workspace and conduct real-time experiments. This research contributes valuable insights to the field of non-intrusive occupancy detection for smart building management systems.

In the paper [2], authors have compared various machine learning alogirthms for accurately classifying the number of people present in the room. The authors have shown that models such as Linear Discriminant Analysis (LDA), Classification and Regression Trees (CART), Random Forest (RF), and K-Nearest Neighbors (KNN) can achieve high accuracy levels ranging from 95% to 99% for binary occupancy detection. Some advanced approaches like Support Vector Machines (SVM), feed-forward neural networks (FFNNs), and stacking neural networks have also been effective, with accuracies from ranging from 68.87% to 91.18%. Recent developments include the CatBoost algorithm integrated with IoT indoor air quality data, achieving up to 99.85% accuracy, and non-stationary ML algorithms for stream data processing, exceeding 83% accuracy. These advancements highlight the potential of ML techniques to improve occupancy detection, that can lead to significant energy savings and smarter building management.

III METHODOLOGY

In this section, the various methodologies that were carried out are described in detail. This includes the steps taken, the techniques employed, and the tools used to achieve the desired outcomes.

Each methodology is explained comprehensively to provide a clear understanding of the processes involved.

A THEORETICAL BACKGROUND

1 K-Nearest Neighbors Algorithm

The K-Nearest Neighbors algorithm is a simple and intuitive machine learning method used for classification. It works by identifying the 'K' closest data points (neighbors) to a new, unseen data point based on a distance metric. The class or value of the new data point is then determined by the majority class. KNN is a non-parametric method, it doesn't make any assumptions about the underlying data distribution, which makes it versatile for different types of data.

2 Distance Metrics

Distance metrics are used to measure the similarity or dissimilarity between data points. Common distance metrics include Euclidean distance, Manhattan distance, Minkowski distance. Euclidean Distance measures the straight-line distance between two points in Euclidean space. It is the most commonly used metric. Manhattan Distance also known as L1 distance, it measures the distance between two points along axes at right angles. Minkowski Distance a generalization of both Euclidean and Manhattan distances, with a parameter that can be adjusted to get either. The choice of distance metric can affect the performance of the K-NN algorithm. The formulas for these metrices can be found in Section A of Appendix.

3 Weighted KNN

In KNN algorithm, different attributes (features) can be given different weights to reflect their importance. Weighting can help improve the accuracy of the algorithm by emphasizing more important features. For example, if certain features are more predictive of the target variable, they can be assigned higher weights, ensuring they have a greater influence on the distance calculation and the resulting classification.

4 Cross-Validation

Cross-validation is a technique used to evaluate the performance of a machine learning model. It involves splitting the data into multiple subsets (folds), training the model on some subsets, and testing it on the remaining ones. This process is repeated several times, and the results are averaged to get a more accurate estimate of the model's performance. Cross-validation helps in assessing how well the model will generalize to new, unseen data, and prevents over-fitting. The cross-validation is performed to find the optimal value for the 'K', which is the number of neighbour to consider while making a classification.

5 ROC Curve

ROC curve stands for the Receiver Operating Characteristics curve. It is the plot of True Positive Rate (TPR) versus False Positive Rate (FPR), y-axis and x-axis respectively. The ROC curve measures the trade-off between TPR and FPR (benefits and costs). It is generally used in balanced datasets and focuses on the overall performance of the model. The origin of the curve is (0,0) and its end is at (1,1). The diagonal line from (0,0) to (1,1) represents the random guessing. The optimal point in the ROC curve is the top left corner, i.e. (0,1) point. The Area Under Curve (AUC) of the ROC curve represents the probability that the classifier ranks a random positive instance higher than a random negative instance. In some cases, the curve can also be misleading, since the high area under the curve might not reflect good performance if the dataset is highly imbalanced.

6 PR Curve

PR Curve stands for Precision-Recall Curve. It is also a graphical representation used to evaluate the performance of a classifier system. The y-axis in the PR curve represents Precision and the x-axis represents Recall. It is generally used in imbalanced datasets. It measures the trade-off between precision and recall. The curve focuses on the performance of the model on the positive class. The optimal point for the PR curve is the top right corner (1,1). Compared to the ROC curve, it is more

informative and focuses on the performance of the minority class.

B SYSTEM BLOCK DIAGRAM

The Figure 15 illustrates the system block diagram of a K-Nearest Neighbors (KNN) algorithm. The input data is preprocessed to ensure proper formatting. The preprocessing includes normalization and handling missing values. After proper preprocessing of the training data, the KNN classifier was trained. When a new data point is introduced in the testing phase, the algorithm calculates the distances between this query point and all training points using a chosen distance metric (e.g., Euclidean, Manhattan). The nearest k points are selected, and their labels are used to predict the query point's label. The predicted label is outputted, and the model's performance is evaluated using metrics such as confusion metrics, ROC curve.

C PERFORMANCE METRICS

1 Accuracy

Accuracy is defined as the ratio of correctly predicted instance (True Positive and True Negative) to the total instances. It indicates the overall efficiency of the model and is best for balanced datasets. The equation for accuracy is given in Equation 5.

2 Precision

The fraction of correctly predicted positive instances (True Positive) to the total of positively predicted instances (True Positive and False Positive) is referred to as Precision. The formula for Precision is given in Equation 6. High precision is important since false positives can be expensive or dangerous. Higher precision means that the model has a lower rate of false positives.

3 True Positive Rate

The percentage of actual positives correctly classified by the model is termed as True Positive Rate (TPR). The formula for True Positive Rate (TPR) is given in Equation 7. A higher value of TPR represents that the model is good at identifying positive cases.

4 False Positive Rate

The percentage of actual negatives incorrectly classified as positives by the model is defined as a False Positive Rate (FPR). The formula for False Positive Rate (FPR) is given in Equation 8. A lower FPR shows that the model makes fewer false positive errors or false alarms.

5 Recall (Sensitivity)

The fraction of correctly predicted positive instances (True Positive) to the total of actual positive instances (True Positive and False Negative) is termed as Recall. The formula for Recall is given in Equation 8. It measures the ability of the model to find all the relevant cases within a dataset. It is essential when the cost of False Negatives is high. Higher recall indicates the model identifies most of the positive instances.

6 F1-Score

The F1 score refers to the harmonic mean of precision and recall. The value of the F1 score lies between 0 and 1. It assesses the balance between precision and recall, especially in imbalanced datasets. The equation for F1 score is given in Equation 9.

D DATA PRE-PROCESSING PIPELINE

The data first stored in a CSV file was loaded into a data frame with the panda's library. From the loaded dataset, irrelevant columns such as customerId, and row number were dropped. From the dataset rows with null values were checked and if any whole row was dropped. The categorical data were encoded using the label encoder. The continuous data were normalized to zero mean and unit variance. The correlation between features was checked. All features were found to be independent. These data were used to train the Gaussian Naive Bayes. For training of the Categorical Naives Bayes Classifier, all the continuous data were converted to categorical by the process of binning. The data were binned in 5 equal bins. For training the Hybrid Gaussian Naive Bayes, categorical data was fed to Categorical NB, and continuous data was fed to Gaussian NB.

E DATASET DESCRIPTION

The UCI dataset for room occupancy [3] was collected in a 6m x 4.6m room by deploying sensor nodes in a star configuration with the sensor nodes transmitting data to the edge every 30s using wireless transceivers. The data was collected for 4 days in a controlled manner with the occupancy in the room varying between 0 and 3 people. The actual value for the number of people was manually added. The dataset consist of four columns recording the data from four different temperature sensors, four columns containing the data from four different light sensor, four columns containing the data from four different sound sensor, two columns of data for PIR sensor and two columns of data from carbon dioxide sensor. Finally, the last column contains the actual number of people present in the room. The Table 1 summaries all the data attributes present in the dataset. The total dataset comprises 10,129 rows and 19 columns. The dataset consists the following columns.

- **Date:** Date of the record.
- **Time:** Time of the record.
- **S1_Temp to S4_Temp:** Temperature readings from four different sensors.
- **S1_Light to S4_Light:** Light intensity readings from four different sensors.
- **S1_Sound to S4_Sound:** Sound level readings from four different sensors.
- **S5_CO2:** CO2 concentration levels.
- **S5_CO2_Slope:** Rate of change of CO2 levels.
- **S6_PIR & S7_PIR:** Passive infrared sensor readings (motion detection).
- **Room_Occupancy_Count:** Number of occupants in the room.

Room_Occupancy_Count is the target class of the dataset. The room occupancy count varies from 0 to 3. The highest number contributor to the target class is the value 0, which means the room had no one for most of the time period. The time instance at which the number of occupants in the room counts to 0, 1, 2, and 3 is 8228, 459, 748, and 694 respectively. The histogram for this distribution can be visualized in Figure 1.

The temperature sensor data from all 4 temperature sensors (S1_Temp to S2_Temp) have mean values from approximately 25 to 26 degree Celsius. Therefore, the standard deviation is very low, which indicates stable temperature readings. Among all the temperature sensors, correlation ranges from 0.77 to 0.95, indicating that readings across different sensors are strongly correlated.

The light sensors (S1_Light to S4_Light) have readings ranging from 0 to 280 Lux. The readings from the four sensors are highly variable with mean ranging from 13 to 26 Lux. These sensors also have many zero readings. The light sensors have moderate to high correlation, especially between S1_Light and S3_Light ($r = 0.82$) and S1_Light and S2_Light ($r = 0.84$). This indicates that light intensity in different parts of the room tends to change together.

The sound sensors (S1_Sound to S4_Sound) also have positive correlations among each other, but the correlation is not as strong as that among temperature or light sensors. However, S3_Sound and S4_Sound have $r = 0.7$, while other sound sensors have low to moderate correlation among each other. This signifies that the intensity of sound level is also somehow related.

The CO2 sensor (S5_CO2) is placed at the middle of the room during the data collection. The data collected from this sensor has mean around 460 ppm and standard deviation around 200 ppm. The CO2 sensor has strong correlation with temperature sensors (S1_Temp to S2_Temp) with values ranging from 0.65 to 0.87. This suggests that the higher temperature might be associated with higher CO2 levels. Similarly, the CO2 sensor has moderate correlation with light sensors, especially S1_Light (0.6), S2_Light (0.57), and S3_Light (0.65). But, it has low correlation with S4_Light (0.15).

The CO2 Slope (S5_CO2_Slope) has low correlation with other features. It can be interpreted as: the rate of change of CO2 levels is not much linked to other features. The PIR Sensors (S6_PIR and

S7_PIR) are for motion reading. Mostly, the attribute values are 0, indicating little motion inside the room. These PIR Sensors have moderate correlations with sound and light sensors, indicating that the motion detection is somewhat related to changes in sound and light levels.
The Figure 3 shows the heatmap for the correlation between different features. The darker shades indicate weaker correlations and lighter shades indicate stronger correlations. Observing the heatmap, it can be seen that there exists stronger correlation among temperature sensors and between certain light sensors. Further, there exist moderate correlations among sound sensors, and between CO2 sensors and temperature-light sensors. Whether low or high, the correlation among features exists and is positive. So, dimensionality reduction techniques can be helpful to reduce the computational complexity of the model. However, for this dataset, since the number of rows and columns are not so large, the model training can be managed readily even without the need for dimensionality reduction.

F INSTRUMENTATION DETAILS

The experimentation was conducted using VsCode. Python programming language is used to program KNN algorithm. Many Python libraries were utilized for data analysis and visualization such as pandas, numpy, sklearn, matpotlib. Pandas was used for data loading and manipulation. Sklearn and Numpy for data encoding, metrics generation. Matplotlib was use for visualization of results.

IV RESULTS

The highest accuracy for the KNN model was obtained using K value as 3 and distance metric as Manhattan Distance. The obtained accuracy using these parameters is 99.65%. Using the default parameters (K=5 and distance metric as Euclidean distance), 99.01% accuracy was obtained. To improve the accuracy, the optimal value of 'K' was searched using the GridSearchCV module of ScikitLearn (a Machine Learning library for Python). The optimal value of K was found to be 3. This resulted in the increase of accuracy from 99.01% (default, K=5) to 99.11% (K=3). Then, the distance metric was set to Cosine distance. But, doing this, caused the accuracy to drop to 99.06%. This showed that the cosine distance is not a suitable distance metric for this dataset. Further, the distance parameter was changed to Manhattan distance. This resulted in an increase in the accuracy of the model to 99.65%, which is the best accuracy obtained throughout the experimentation. The three algorithms provided by the ScikitLearn library to compute the nearest neighbors are BallTree, KDTree and Brute-Force algorithm. However, switching between the algorithms for K=3 didn't result in any change in accuracy and the accuracy was noted to be 99.11%.
For the KNN classifier with optimal value of K (=3), the Area Under Curve for Receiver Operating Characteristics (for all classes) is obtained to be almost 1.00. Likewise, the Area Under Curve for Precision-Recall Curve (KNN with K=3) achieved an area of 1.00 for class 0, 0.99 for class 1, 0.98 for class 2 and 0.97 for class 3.
For the best KNN classifier in this case (with K=3 and the Manhattan Distance metric), the Area Under Curve for Receiver Operating Characteristics (for all classes) is obtained to be near-perfect 1.00, indicating the phenomenal performance of the model. Similarly, the Area Under Curve for Precision-Recall Curve (KNN with K=3 and the Manhattan Distance metric) achieved an area of 1.00 for class (0 and 1) and 0.99 for class (2 and 3).

V DISCUSSION AND ANALYSIS

The K-Nearest Neighbor (KNN) algorithm has produced satisfactory results for the dataset (Room Occupancy Estimation), proving itself as a suitable model for this dataset.

A Optimal value of K

The hyperparameter selection was assisted by GridSearchCV which helped find out K=3 as the optimal value for K. The plot in Figure 9 shows the cross-validation accuracy for different values of K. The Accuracy versus Number of Neighbors taken into account (K) is plotted with the value of K ranging from 1 to 30 with step size 2. The maximum accuracy was obtained at K=3 (99.11%). But

as the number of neighbors taken into consideration (K) was increased, the accuracy of the respective KNN model started decreasing. This is due to the effect of data points that are responsible for majority voting for an unknown data instance, even though they might be farther from the unknown data instance itself. The optimal K value as 3 suggests that including more than 3 neighbors for majority voting does not provide better results.

B KNN with default configuration

The K-Nearest Neighbor classifier produced an accuracy of 99.01% with default configuration, which is a good result. The default configuration has a K value of 5, Euclidean distance (p=2) as distance metric and uniform weights. The confusion matrix (K=5) in Figure 2 shows the model with default configuration was able to predict True label 0 as 0 (1214 times), 0 as 1 (1 time), 0 as 2 (1 time) and 1 as 3 (3 times); True label 1 as 1 (101 times), 1 as 0 (0 times), 1 as 2 (0 times) and 1 as 3 (2 times); True label 2 as 2 (151 times), 2 as 0 (0 times), 2 as 1 (0 times) and 2 as 3 (13 times); and True label 3 as 3 (133 times), 3 as 0 (4 times), 3 as 1 (1 time) and 3 as 2 (2 times). Specifically for the class label 2, the model predicts it as 3 for 13 times, which significantly reduces the performance of the model. From this, it can be clearly seen that the classifier is able to perform well for the majority class (label 0) but sometimes struggles with minority classes.

C KNN with K=3

As the optimal value for K was found to be 3, the accuracy of the model was increased slightly to 99.11%. Rest of the parameters like distance and weights were set as default (distance=Euclidean and weights=uniform). The confusion matrix (K=3) in Figure 5 shows the model was able to predict: True label 0 as 0 (1615 times), 0 as 1 (0 times), 0 as 2 (0 times), and 0 as 2 (0 times); True label 1 as 1 (102 times), 1 as 0 (1 time), 1 as 2 (0 times), and 1 as 3 (0 times); True label 2 as 2 (157 times), 2 as 0 (0 times), 2 as 1 (2 times), and 2 as 3 (5 times); and True label 3 as 3 (134 times), 3 as 0 (0 times), 3 as 1 (0 times), and 3 as 2 (6 times). Although the performance got slightly better, it still misclassified the minority class.

D KNN with K=3 and metric=Cosine

The value of K set to 3 and the metric taken as cosine could only get the accuracy of 99.06%. This value is better than that with default configuration (K=5) of KNN, but got worse than that with K=3 and metric as Euclidean. The confusion matrix (K=3 & Metric=Cosine) in Figure 4 shows the model was able to predict: True label 0 as 0 (1615 times), 0 as 1 (0 times), 0 as 2 (1 time), and 0 as 3 (3 times); True label 1 as 1 (102 times), 1 as 0 (1 time), 1 as 2 (0 times), and 1 as 3 (0 times); True label 2 as 2 (157 times), 2 as 0 (0 times), 2 as 1 (0 times), and 2 as 3 (7 times); and True label 3 as 3 (133 times), 3 as 0 (1 time), 3 as 1 (0 times), and 3 as 2 (6 times). This configuration of the model shows a slight variation compared to other configurations, but still shows a similar pattern of misclassification like the other ones. Thus, the cosine distance does not provide any significant advantage over the Euclidean distance in this case.

E KNN with K=3 and metric=Manhattan

The KNN classifier with K value of 3 and metric as Manhattan distance provided the best accuracy among all the configurations. The accuracy was obtained as 99.65%, which is a significant improvement over the model with K value as 3 and metric as Euclidean distance. The confusion matrix (K=3 & Metric=Manhattan) in Figure 7 shows the model was able to predict: True label 0 as 0 (1618 times), 0 as 1 (0 times), 0 as 2 (0 times), and 0 as 3 (1 time); True label 1 as 1 (103 times), 1 as 0 (0 times), 1 as 2 (0 times), and 1 as 3 (0 times); True label 2 as 2 (161 times), 2 as 0 (0 times), 2 as 1 (0 times), and 2 as 3 (3 times); and True label 3 as 3 (137 times), 3 as 0 (0 times), 3 as 1 (0 times), and 3 as 2 (2 times). All these results are evident to say that the model is able to improve its performance for minority class as well, since it predicts the minority classes more accurately than any other KNN models experimented.

F KNN with K=3, metric=Manhattan and weights=Distance

Alike the previous KNN model (with K value of 3, metric as Manhattan distance and weights set as

uniform), the modified KNN model with weights changed to distance also provided the same accuracy of 99.65%. The confusion matrix (K=3 & Metric=Manhattan & Weights=Distance) in Figure 6 shows the model was able to predict: True label 0 as 0 (1618 times), 0 as 1 (0 times), 0 as 2 (0 times), and 0 as 3 (1 time); True label 1 as 1 (103 times), 1 as 0 (0 times), 1 as 2 (0 times), and 1 as 3 (0 times); True label 2 as 2 (161 times), 2 as 0 (0 times), 2 as 1 (0 times), and 2 as 3 (3 times); and True label 3 as 3 (137 times), 3 as 0 (0 times), 3 as 1 (0 times), and 3 as 2 (2 times). Thus, the confusion matrix shows no difference between those two models. This means, in this dataset, weighting by distance does not significantly change the performance of the model compared to that by uniform weights.

G KNN with different algorithms

Scikit-learn enables the use of three different algorithms for K-Nearest Neighbors which include Ball Tree, KD Tree and Brute-Force algorithm. However, by default, the KNN classifier tries to automatically decide the most suitable algorithm for classification. For experimentation purpose, the parameter for 'algorithm' was set to 'ball_tree', 'kd_tree' and 'brute' one by one with K value set constant at 3. For each of these algorithms, there was no significant difference in the accuracy of the model. This shows that the 'algorithm' parameter has no much influence on the model training for this dataset. Also, the default setting 'auto' automatically finds the most appropriate algorithm based on the values passed to the KNN classifier. Hence, it becomes not so necessary to choose the algorithm to compute the nearest neighbors.

H ROC and PR

The Area Under Curve (AUC) for Receiver Operating Characteristic (ROC) Curve in Figure13 (for K=3 and metric=Manhattan) has area 1.00 (for all classes) which indicates almost perfect classification. The curves are nearly perfect, which suggests the model has a high true positive rate and a low false positive rate for all classes. The Precision-Recall (PR) curve in Figure 11(for K=3 and metric=Manhattan) is also nearly perfect showing that the model has high precision and recall for all classes. For the PR curve, classes 0 and 1 have near perfect AUC (1.00), while classes 2 and 3 have slightly lower AUC (0.99).
From the observations of the experiments with different parameters of the KNN classifiers model, it can be seen that the number of the Nearest Neighbor chosen (K) has the most significant impact on the model itself. Increasing the value of K (more than 3) for the dataset being used, is unlikely to improve the overall performance of the model. Likewise, the choice of the distance metric (Manhattan, Euclidean, Cosine) has a slight impact on the model performance. In this case, the Manhattan distance is the best performing one with the value of K as 3, and the Cosine distance metric seems to have no advantage over the Manhattan or Euclidean distance. Changing the parameters: 'weights' and 'algorithm' has no much impact on the performance of the model, as the accuracy obtained by tweaking these parameters is the same as others. The ROC curve and PR curve indicate exceptional classification performance for all four classes.

VI CONCLUSION

Looking at the accuracy obtained by the model, the K-Nearest Neighbor classifier stands as a suitable algorithm for this problem. However, for getting the best result using KNN algorithm, careful tuning of the parameters (especially K and distance metrics in this case) is necessary. The best performance was achieved using the K value of 3 with Manhattan distance metric. These systems can benefit from accurate occupancy detection to optimize energy consumption, enhance occupant comfort, and improve overall building efficiency. This model configuration could better classify the minority classes. However, the other models were good enough to be utilized for the classification problem. The problem of the misclassification of the minority class using other model configurations might be associated with the class imbalance problem. The dataset contains imbalanced target classes with label 0 being the majority class. This imbalance leads to the classifier's bias towards predicting the majority class. Thus, techniques like SMOTE (Synthetic Minority Over-

Sampling Technique) can be taken into account to address the issue of class imbalance. Additionally, the techniques of dimensionality reduction like PCA (Principal Component Analysis) can be utilized to reduce the computational complexity.

VII Appendix

A MATHEMATICAL FORMULAE

The main idea behind K-NN is to find the distance between the points whose class is to be known and the 'k' nearest neighbours. The formula for L_p distance between two points two points $X1$ and $X2$ can be calculated as.

$$||X1 - X2||_p = (\sum_{i=1}^{n} |x_1, i - x_2, i|^p)^{1/p} \quad (1)$$

where $X1$ and $X2$ are the data points, x_1, i and x_2, i are the i^th attributes in the $X1$ and $X2$. The Equation A is called as Minkwoski distance which is the general case of all the distance metrices.

If the value of p is equal 2, the euclidean distance is obtained between $X1$ and $X2$.

$$||X1 - X2||_2 = \sqrt{(\sum_{i=1}^{n} |x_1, i - x_2, i|^2)} \quad (2)$$

When the value of p is set equal to 1, Manhattan distance or Taxi-cab distance is obtained.

$$||X1 - X2||_1 = (\sum_{i=1}^{n} |x_1, i - x_2, i|^2) \quad (3)$$

When calculating the distance between the data point some features may be more discriminative than other features. Such features can be scaled by using weights.

$$D(X1, X2) = (\sum_{i=1}^{n} \sum_{k} W_k |x_{1,i,k} - x_{2,i,k}|^p)^{1/p} \quad (4)$$

Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

where:

- TP = True Positives
- TN = True Negatives
- FP = False Positives
- FN = False Negatives

Precision

$$\text{Precision} = \frac{TP}{TP + FP} \quad (6)$$

where:

- TP = True Positives
- FP = False Positives

True Positive Rate (TPR) or Recall

$$\text{TPR (Recall)} = \frac{TP}{TP + FN} \quad (7)$$

where:

- TP = True Positives
- FN = False Negatives

False Positive Rate (FPR)

$$\text{FPR} = \frac{FP}{FP + TN} \quad (8)$$

where:

- FP = False Positives
- TN = True Negatives

F1 Score

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

where:

- Precision $= \frac{TP}{TP+FP}$
- Recall $= \frac{TP}{TP+FN}$

B GRAPHS

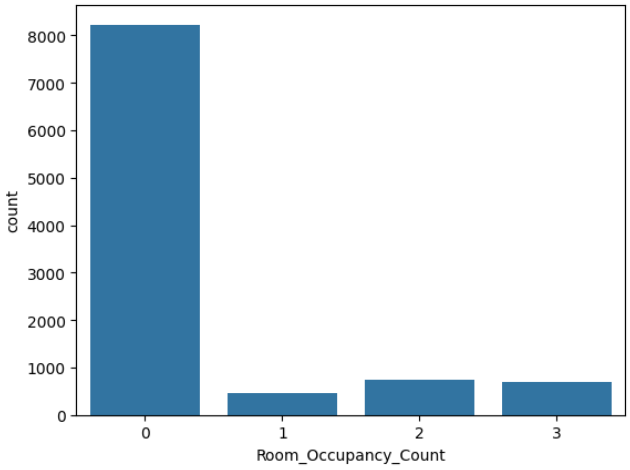

Figure 1: Countplot of Room Occupancy Count

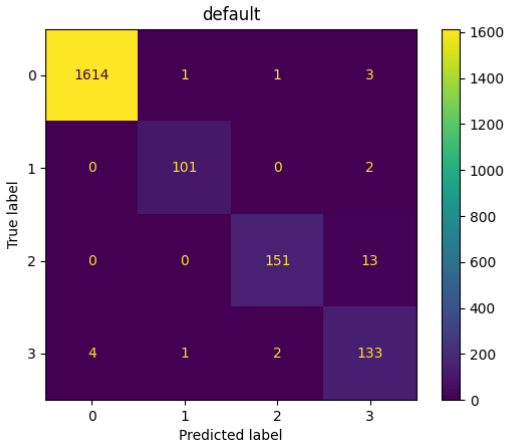

Figure 2: Default Confusion Matrix

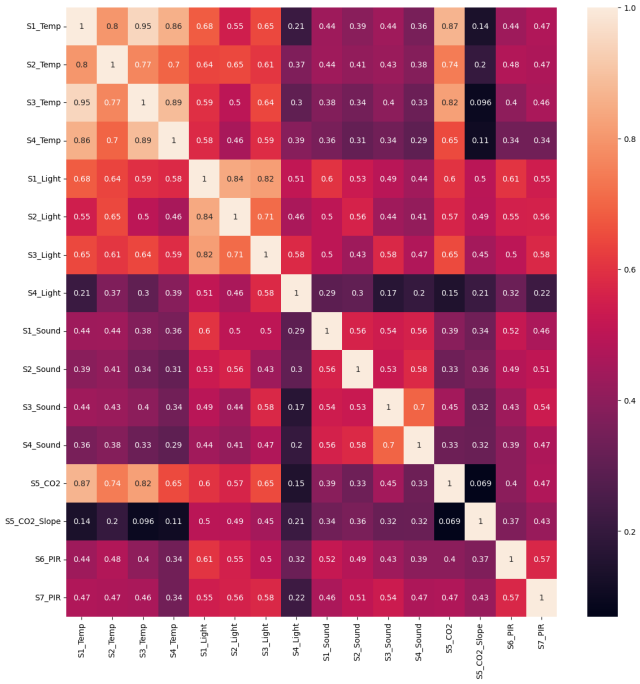

Figure 3: Heatmap of Features

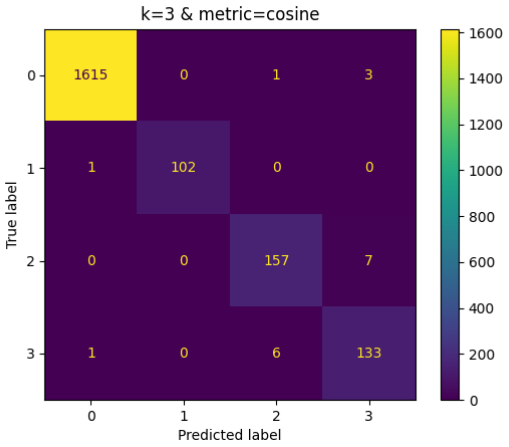

Figure 4: $k = 3$ & metric=cosine Confusion Matrix

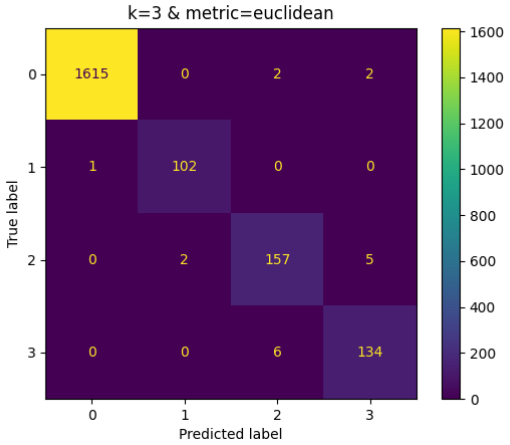

Figure 5: $k = 3$ & metric=euclidean Confusion Matrix

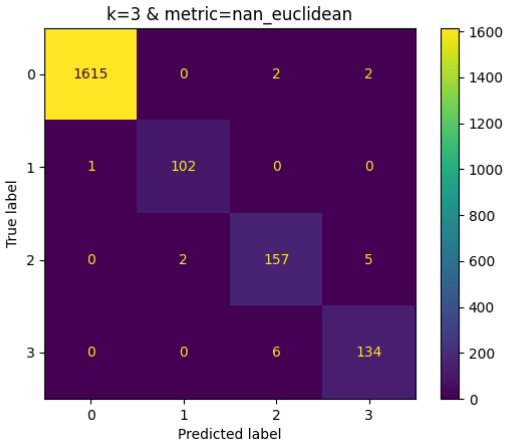

Figure 8: $k = 3$ & metric=nan_euclidean Confusion Matrix

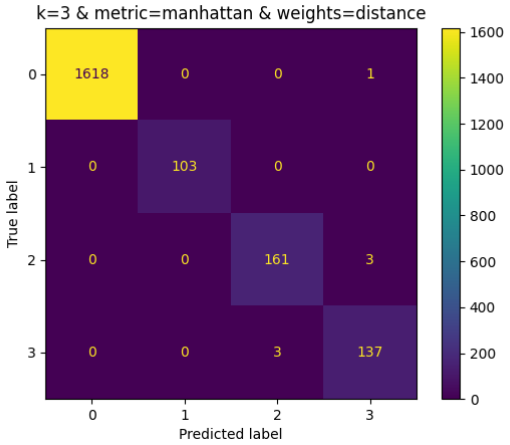

Figure 6: $k = 3$ & metric=Manhattan Confusion Matrix

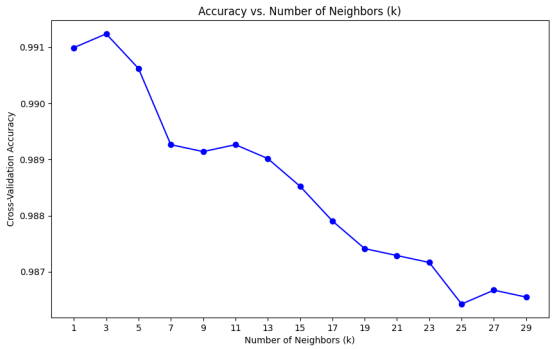

Figure 9: k vs Accuracy

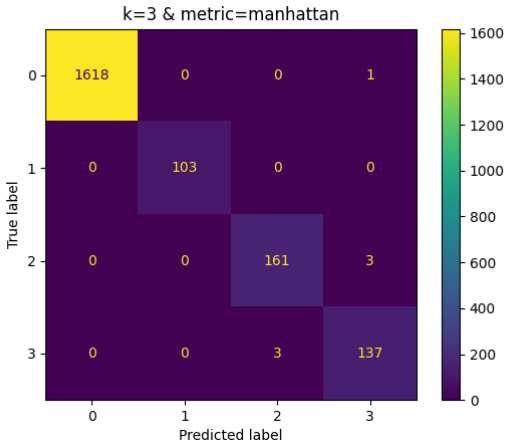

Figure 7: $k = 3$ & metric=manhattan Confusion Matrix

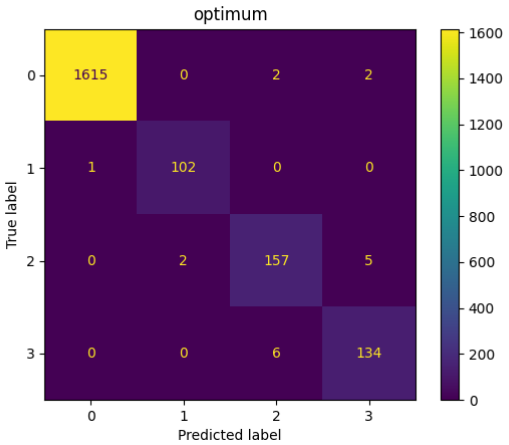

Figure 10: Optimum Confusion Matrix

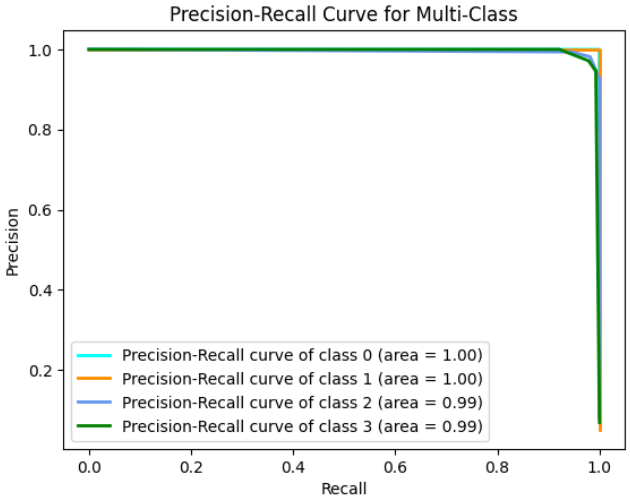

Figure 11: PR for Manhattan

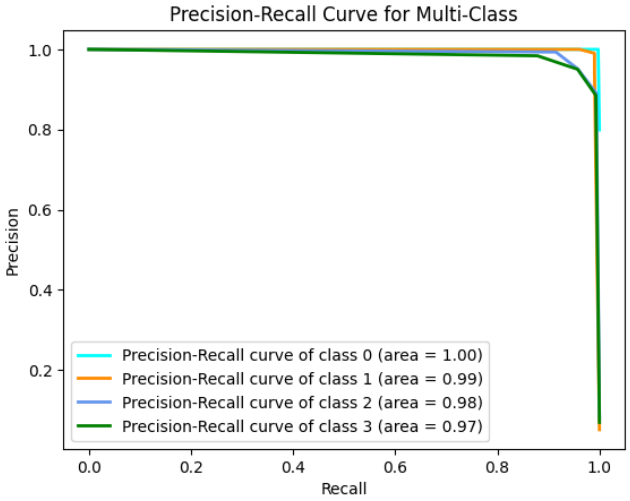

Figure 12: PR for Optimum

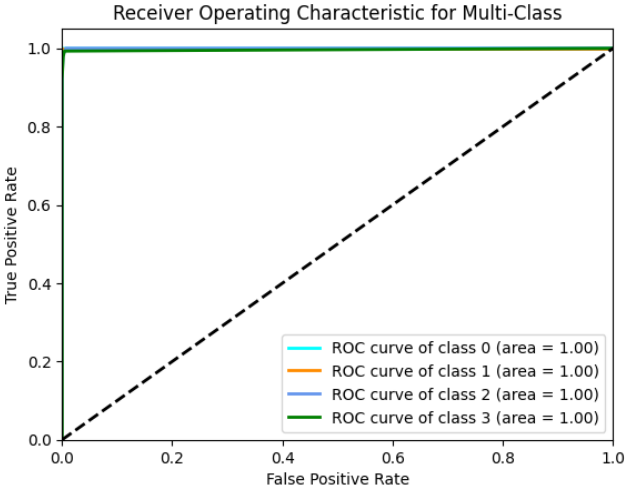

Figure 13: ROC for Manhattan

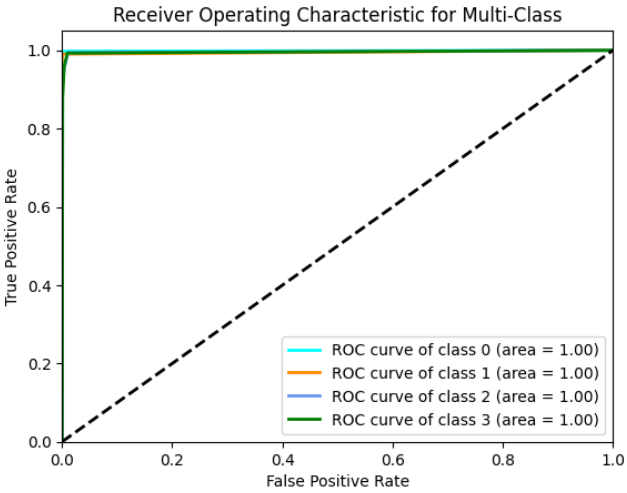

Figure 14: ROC for Optimum

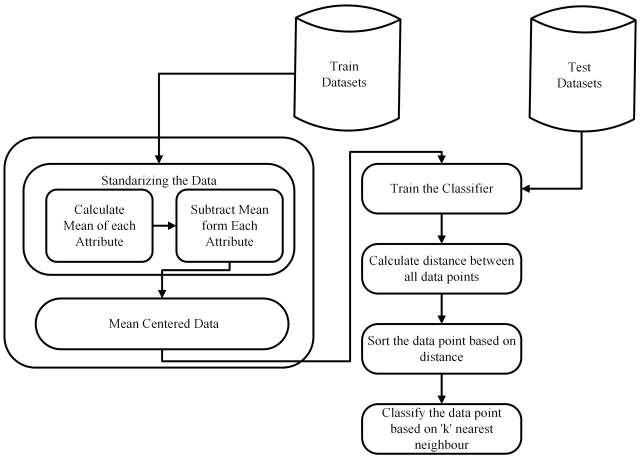

Figure 15: System Block Diagram

C Table

Table 1: Dataset Description

Variable Name	Role	Type
Date	Feature	Date
Time	Feature	Date
S1_Temp	Feature	Continuous
S2_Temp	Feature	Continuous
S3_Temp	Feature	Continuous
S4_Temp	Feature	Continuous
S1_Light	Feature	Integer
S2_Light	Feature	Integer
S3_Light	Feature	Integer
S4_Light	Feature	Integer
S1_Sound	Feature	Continuous
S2_Sound	Feature	Continuous
S3_Sound	Feature	Continuous
S4_Sound	Feature	Continuous
S5_CO2	Feature	Integer
S5_CO2_Slope	Feature	Continuous
S6_PIR	Feature	Binary
S7_PIR	Feature	Integer
Occupancy_Count	Target	Integer

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," *2018 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:56336847

[2] Y. Ibrahim, U. Y. Bagaye, and A. I. Muhammad, "Machine learning for accurate office room occupancy detection using multi-sensor data," *Engineering Proceedings*, vol. 58, no. 1, 2023. [Online]. Available: https://www.mdpi.com/2673-4591/58/1/67

[3] A. P. Singh and S. Chaudhari, "Room Occupancy Estimation," UCI Machine Learning Repository, 2023, DOI: https://doi.org/10.24432/C5P605.

Pragyan Bhattrai is currently pursuing Bachelor's degree in Electronics, Communication and Information engineering in IOE, Thapathali Campus. He is deeply passionate about working with data. His journey began with learning the basics of Python and gaining an understanding of data science. Since then, he has been getting acquainted with data visualization techniques. He is eager to further strengthen his statistical knowledge and venture into the field of machine learning.

Prashant Raj Bista is currently pursuing Bachelor's degree in Electronics, Communication and Information engineering in IOE, Thapathali Campus.He is fascinated by open-source software's and open-source community. He is currently learning about machine learning and its application on making human life easier. His hobby includes reading novels, exploring machines and web development.