**SMART PUBLIC RESTROOM**

**TEAM MEMBER**

**622621121059: VINODHINI.P**

**Phase-4 Submission Document**

**Project Title: Smart Public Restroom**

**Phase 4: Development Part 2**

**Topic: *Continue building the smart public restroom model by feature engineering, model training, and evaluation.***

****

**Smart Public Restroom**

**Introduction:**

* In our increasingly interconnected world, the integration of the Internet of Things (IOT) is revolutionizing the way we interact with the environment around us. From our homes to our cities, IOT technology is transforming everyday spaces into smarter, more efficient, and user-centric environments. One of the remarkable applications of this transformation is the "Smart Public Restroom" model.
* The Smart Public Restroom leverages the power of IOT to enhance the functionality and user experience within these essential facilities. It introduces a new era of convenience, hygiene, and resource management for both the operators and users of public restrooms. By incorporating an array of sensors, connected devices, and data-driven solutions, this model addresses some of the most pressing challenges associated with public restroom management.
* Through the synergy of IOT technology, smart public restrooms offer a host of innovative features. Real-time occupancy monitoring ensures that visitors can quickly identify available facilities, reducing wait times and enhancing convenience. Touch less fixtures and automated sanitization processes minimize the risk of contamination, fostering a cleaner and healthier environment. The efficient management of resources, such as water and electricity, not only reduces operational costs but also promotes sustainability.

This introduction sets the stage for an exploration of the Smart Public Restroom model, demonstrating how IOT technology is reshaping public facilities, making them safer, more efficient, and more responsive to the needs of the public. As we delve deeper into the various components and benefits of this innovative model, it becomes evident that the smart public restroom is a testament to the potential of IOT to improve the quality of our daily lives in ways both practical and profound.

**Given Dataset:**

In the dataset of the smart public restroom model building process consists of main features of the model and its description of working. Here the sample dataset with several features, sensors:

|  |  |  |
| --- | --- | --- |
| Feature | Data Type | Description |
| Timestamp | Date and Time | The time and date at which the data was collected. |
| Stall ID | Integer | The unique identifier of the restroom stall. |
| Occupancy | Boolean | True if the stall is occupied, false if it is empty. |
| Cleanliness | Integer | A measure of the cleanliness of the stall, on a scale of 1 to 10, with 10 being the cleanest. |
| Soap Level | Integer | The percentage of the soap dispenser that remains. |
| Temperature | Integer | The temperature in the restroom stall. |
| Humidity | Integer | The humidity in the restroom stall. |

**Sample Dataset:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Date and time | Toilet ID | Occupancy | Temperature | Humidity | Air quality | Water flow | Paper level | Soap level | Trash level | Other sensors |
| Date and time | 1 | Occupied | 25°C | 50% | Good | 1 L/min | 75% | 50% | 75% | None |
| Date and time | 2 | Unoccupied | 26°C | 55% | Good | 0 L/min | 100% | 100% | 50% | Motion sensor |
| Date and time | 3 | Occupied | 27°C | 60% | Fair | 2 L/min | 50% | 25% | 100% | None |
| Date and time | 4 | Unoccupied | 28°C | 65% | Poor | 0 L/min | 25% | 0% | 100% | CO2 sensor |

**Overview of the Process:**

Building a smart public restroom model using IOT involves a multi-step process to design, implement, and maintain the system effectively. Here is an overview of the five basic steps by feature selection, model training, and evaluation:

**1. Needs Assessment and Planning**:

* **Identify Objectives**: Determine the specific goals of the smart public restroom, such as improved hygiene, resource efficiency, and enhanced user experience.
* **Stakeholder Involvement:** Engage with stakeholders, including facility managers, users, and IOT specialists, to understand their requirements and expectations.
* **Site Evaluation:** Evaluate the existing restroom space, considering layout, infrastructure, and utility connections.
* **Budget and Resources:** Determine the budget and allocate resources for the project.

**2. Sensor Selection and Infrastructure:**

* **Sensor Types:** Choose appropriate IOT sensors and devices for occupancy monitoring, hygiene control, resource management, and user feedback. Examples include motion sensors, water flow sensors, and air quality sensors.
* **Connectivity:** Ensure reliable connectivity through Wi-Fi, Bluetooth, or other IOT communication protocols.
* **Power Supply:** Plan for power sources, including battery-powered sensors or wired connections.
* **Data Integration:** Select a data management system to collect, store, and analyse data from the sensors.

**3. Installation and Integration:**

* **Physical Installation:** Install sensors and devices in the restroom, strategically placing them to capture relevant data.
* **Network Setup:** Configure the sensors to connect to the network and the central management system.
* **Integration:** Ensure seamless integration of sensors with the central control system to enable data collection and real-time monitoring.
* **Testing and Calibration:** Test sensors for accuracy and calibrate as needed to ensure reliable data.

**4. User Interface and Data Visualization:**

* **Dashboard Development:** Create a user-friendly dashboard for facility managers to monitor real-time data.
* **User Feedback System:** Implement a user feedback mechanism, such as kiosks or mobile apps, for restroom users to provide input on cleanliness and overall experience.
* **Alerts and Notifications:** Set up alerts for maintenance staff when issues are detected, and ensure that users receive relevant information, such as stall availability.

**5. Maintenance and Optimization:**

* **Routine Maintenance:** Establish a regular maintenance schedule to ensure sensors and devices operate smoothly.
* **Data Analysis:** Continuously analyse collected data to identify trends and areas for improvement.
* **Upgrades and Expansion:** Consider upgrading or expanding the system as technology evolves or as the restroom's needs change.
* **User Education:** Educate restroom users on the new features and functionality to maximize the benefits of the smart restroom.

Building a smart public restroom using IOT is an on-going process that requires collaboration between various stakeholders and on-going monitoring and optimization to ensure the restroom remains efficient, user-friendly, and meets its objectives.

**PROCEDURE:**

**Feature Selection:**

Feature selection is the process of selecting the most relevant and informative features from a dataset for use in machine learning. It is an important step in the development of a smart public restroom model using IOT, as it can help to improve the accuracy and efficiency of the model.

There are a number of different feature selection methods that can be used, depending on the specific dataset and machine learning algorithm being used. Some common methods include:

* **Correlation analysis:** This method is used to identify features that are highly correlated with each other. Highly correlated features may contain redundant information, so it is often beneficial to remove one or more of these features from the dataset.
* **Information gain:** This method is used to measure the amount of information that each feature provides about the target variable. Features that provide the most information are typically more relevant to the machine learning task, so they are more likely to be included in the selected feature set.
* **Recursive feature elimination (RFE):** This method is used to iteratively remove features from the dataset and evaluate the performance of the machine learning model on each iteration. The features that are removed are the ones that have the least impact on the performance of the model.

Here are some specific examples of feature selection for a smart public restroom model using IOT:

* **Occupancy:** This feature is essential for predicting the status of the restroom and optimizing the use of resources.
* **Cleanliness:**This feature is important for ensuring that the restroom is clean and hygienic for users.
* **Toilet paper level and soap level:**These features can be used to identify stalls that need to be restocked.
* **Temperature and humidity:** These features can be used to create a more comfortable and pleasant experience for users.
* **Stall type:** This feature could be used to predict which stalls are most likely to be occupied at a given time.
* **Location of the stall:** This feature could be used to predict which stalls are most likely to be used by users.
* **Time of day and day of the week:** These features could be used to predict which stalls are most likely to be occupied during peak hours or on weekends.
* **Weather conditions:** This feature could be used to predict which stalls are more likely to be used during bad weather.
* **Number of people in the restroom:** This feature could be used to predict which stalls are most likely to be occupied if the restroom is crowded.

The specific features that are selected for a smart public restroom model will depend on the specific goals of the model. For example, if the goal of the model is to predict the occupancy of restroom stalls, then the most important features will likely be occupancy, time of day, and day of the week. By carefully selecting the most relevant and informative features for the smart public restroom model, it is possible to improve the accuracy and efficiency of the model. This can lead to a number of benefits, such as improved cleanliness, reduced costs, and an improved user experience.

**Feature Selection Steps:**

Feature selection is a crucial step in building a smart public restroom model using IOT as it helps identify the most relevant and informative data points to improve system efficiency and user experience. Here are the feature selection steps for such a model:

**1. Define Objectives and Use Cases:**

Start by clearly defining the objectives and specific use cases for the smart public restroom. What are the primary goals of the IOT system? For example, is it to enhance hygiene, optimize resource usage, or improve user experience?

**2. Identify Data Sources:**

List all potential data sources within the smart public restroom, such as sensors, devices, and user feedback mechanisms. This includes motion sensors, water flow sensors, air quality sensors, soap dispensers, occupancy detectors, etc.

**3. Collect Initial Data:**

Gather a comprehensive dataset that includes all available data points from the identified sources. This initial dataset will serve as a foundation for feature selection.

**4. Data Pre-processing:**

Clean and pre-process the data to handle missing values, outliers, and data quality issues. Standardize or normalize the data to ensure consistency.

**5. Feature Relevance Assessment:**

Utilize statistical methods, data visualization, and domain knowledge to assess the relevance of each feature in relation to the defined objectives. Consider techniques such as correlation analysis to identify highly correlated features.

**6. Feature Importance Analysis:**

Use machine learning models like decision trees or random forests to determine feature importance scores. These models can indicate which features contribute most to predictive accuracy or model performance.

**7. Dimensionality Reduction:**

If the dataset contains a large number of features, consider dimensionality reduction techniques like Principal Component Analysis (PCA) or feature extraction methods to reduce the number of dimensions while retaining relevant information.

**8. User Feedback and Stakeholder Input:**

Gather input from facility managers, maintenance staff, and restroom users to understand which features and data points are most critical for the smart restroom's effectiveness and user satisfaction.

**9. Feature Selection Algorithms:**

Implement feature selection algorithms, such as Recursive Feature Elimination (RFE), forward selection, or backward elimination. These techniques iteratively evaluate and select the most important features based on a predefined criterion (e.g., information gain or model performance).

**10. Model Validation:**

Continuously validate feature selections using machine learning models and performance metrics. Ensure that the chosen features contribute to the desired outcomes, whether it's occupancy prediction, resource optimization, or user satisfaction.

**11. Iterative Process:**

Feature selection is often an iterative process, and it may need adjustments as the smart restroom system evolves. Periodically reevaluate feature importance and relevance, especially when new sensors or data sources are introduced.

**12. Document the Final Feature Set:**

Document the final set of selected features and the rationale behind their selection. This documentation is valuable for system maintenance and future enhancements.

The feature selection process should be tailored to the specific objectives and requirements of the smart public restroom model, ensuring that the chosen features contribute to the success of the IOT system in meeting its defined goals.

***Feature Selection:***

Once the selected features have been identified, they can be used to train the machine learning model. This is likely to result in a more accurate and efficient model than if all of the features in the dataset were used.

It is important to note that this is just one example of a feature selection program for a smart public restroom using machine learning. There are a number of other feature selection methods that could be used, depending on the specific dataset and machine learning algorithm being used.

**Program:**

* Here is a Python program about smart public restroom feature selection:

import pandas as pd

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import f\_classif

# Load your dataset

data = pd.read\_csv('smart\_restroom\_data.csv') # Replace with your dataset

# Define the features (X) and target variable (y)

X = data.drop(columns=['Target\_Column']) # Replace 'Target\_Column' with the actual target variable

y = data['Target\_Column'] # Replace 'Target\_Column' with the actual target variable

# Feature selection using ANOVA F-statistic

num\_features\_to\_select = 10 # Number of features to select

selector = SelectKBest(score\_func=f\_classif, k=num\_features\_to\_select)

X\_new = selector.fit\_transform(X, y)

# Get the indices of the selected features

selected\_feature\_indices = selector.get\_support(indices=True)

# Create a DataFrame with selected features

selected\_features = X.columns[selected\_feature\_indices]

selected\_data = data[selected\_features]

# Print the selected features

print("Selected Features:")

print(selected\_features)

# Optionally, you can save the selected data to a new CSV file

selected\_data.to\_csv('selected\_smart\_restroom\_data.csv', index=False)

Feature selection in the context of a smart public restroom model using IOT often involves a data analysis and machine learning process to identify the most relevant features. While a comprehensive feature selection program would depend on the specific data and technology stack used, here's a simplified example using Python and the scikit-learn library to demonstrate the concept. In practice, you may need to adapt this program to your specific dataset and requirements.

**In this example:**

***1. Load your dataset:*** Replace 'smart\_restroom\_data.csv' with the path to your dataset.

2. Define your features (X) and the target variable (y). Replace 'Target\_Column' with the actual target variable you are interested in.

3. Use the `SelectKBest` function from scikit-learn with the ANOVA F-statistic (you can choose other scoring functions depending on your use case) to select a specified number of features (in this case, `num\_features\_to\_select`).

4. Get the indices of the selected features using `selector.get\_support(indices=True)`.

5. Create a new DataFrame containing only the selected features.

6. Print and/or save the selected features and data.

This is a basic example, and feature selection methods can be more complex depending on your specific data and requirements. You may need to customize the program further and consider additional techniques like recursive feature elimination, cross-validation, and more advanced machine learning models for feature selection.

***The top 10 most important smart public restroom features in feature selection are:***

The selection of the top 10 features for a smart public restroom model using IOT depends on the specific goals and objectives of the model. Here's a list of potential features that could be considered essential in such a model:

**1. Occupancy Monitoring:**

- Real-time data on the number of occupied stalls.

- Stall occupancy percentage.

**2. Resource Usage:**

- Water consumption data, including water flow and usage patterns for flushing and faucets.

- Electricity consumption for lighting, hand dryers, and other electrical fixtures.

**3. Hygiene Metrics:**

- Hand washing compliance percentage.

- Sanitization alerts, indicating when sanitation supplies are low.

**4. Air Quality:**

- CO2 levels to monitor air freshness and ventilation needs.

- Smell detection data to address any foul smells.

**5. Maintenance Alerts:**

- Sensor malfunction notifications.

- Waste bin levels to optimize maintenance schedules.

**6. User Feedback:**

- User ratings for cleanliness and overall experience.

- User comments regarding restroom conditions.

**7. Supply Levels:**

- Toilet paper inventory levels.

- Soap and sanitizer inventory levels.

**8. Accessibility Metrics:**

- Voice command usage data.

- Assessment of non-slip flooring conditions.

**9. Security and Privacy Compliance:**

- Security breach alerts.

- User privacy compliance monitoring.

**10. Resource Efficiency:**

- Calculations of water and energy savings due to occupancy-based control systems.

These features are selected to cover a wide range of aspects related to a smart public restroom's functionality, user experience, hygiene, resource management, and sustainability. The actual selection of features will depend on the specific goals of the model and the available IoT sensors and devices in the restroom. It's important to continuously monitor and adapt feature selection based on evolving needs and technologies.

**Model training:**

* Choose a machine learning algorithm: There are a number of different machine learning algorithms that can be used for smart public restroom, such as Support Vector Machines (SVMs), Random Forests, Neural Networks, Predicting Occupancy, Detecting anomalies, Identifying Trends, Long-Short Term Memory (LSTM), KNN, decision trees are covered.
* Regression algorithms can also be used to predict when equipment in a public restroom is likely to fail. This information could then be used to schedule preventive maintenance tasks, avoiding costly and disruptive breakdowns. For example, a regression algorithm could be used to predict when a toilet is likely to clog based on historical data, such as the number of times it has been used and the type of waste that has been flushed down it. The facility manager could then schedule a maintenance worker to clean the toilet before it clogs, avoiding the need to close the restroom for repairs.
* To write an algorithm for a smart public restroom using IOT, you will need to:

1. Collect data from the IOT devices in the restroom, such as occupancy sensors, temperature sensors, and motion sensors.
2. Pre-process the data to clean it and prepare it for the respective algorithm.
3. Train the algorithm on the pre-processed data.
4. Deploy the trained algorithm to the smart public restroom.

**Machine Learning Models:**

* **Support Vector Machines (SVMs):**

 SVMs are a type of supervised learning algorithm that can be used for both classification and regression tasks. SVMs are often used for anomaly detection, as they can be trained to identify patterns in data that deviate from the norm.

Here is a pseudo code for a simple SVM algorithm for a smart public restroom:

**Program:**

import numpy as np

class SVM:

def \_\_init\_\_(self, C=1.0, kernel='linear'):

self.C = C

self.kernel = kernel

self.support\_vectors = None

self.bias = None

def fit(self, X, y):

# Preprocess the data

X\_scaled = (X - np.mean(X)) / np.std(X)

# Train the SVM algorithm

self.support\_vectors, self.bias = train\_svm(X\_scaled, y, self.C, self.kernel)

def predict(self, X):

# Scale the new data

X\_scaled = (X - np.mean(X)) / np.std(X)

# Predict the labels for the new data

predictions = svm\_predict(self.support\_vectors, self.bias, X\_scaled)

return predictions

def train\_svm(X, y, C=1.0, kernel='linear'):

# Train the SVM algorithm using a library such as scikit-learn

svm = SVC(C=C, kernel=kernel)

svm.fit(X, y)

# Return the support vectors and bias

return svm.support\_vectors\_, svm.intercept\_

def svm\_predict(support\_vectors, bias, X):

# Predict the labels for the new data

predictions = np.dot(support\_vectors, X.T) + bias

# Return the predicted labels

return predictions

* **Random Forests:**

Random forests are an ensemble learning algorithm that combines the predictions of multiple decision trees to produce a more accurate prediction. Random forests are often used for classification tasks, such as predicting whether a restroom is likely to be occupied or not.

Here is a pseudocode for a simple random forests algorithm for a smart public restroom:

**Program:**

import numpy as np

class RandomForest:

def \_\_init\_\_(self, n\_estimators=10, max\_depth=None, min\_samples\_split=2, min\_samples\_leaf=1, bootstrap=True):

self.n\_estimators = n\_estimators

self.max\_depth = max\_depth

self.min\_samples\_split = min\_samples\_split

self.min\_samples\_leaf = min\_samples\_leaf

self.bootstrap = bootstrap

self.trees = None

def fit(self, X, y):

# Preprocess the data

X\_scaled = (X - np.mean(X)) / np.std(X)

# Train the random forests algorithm

self.trees = []

for i in range(self.n\_estimators):

# Create a new decision tree

tree = DecisionTreeClassifier(max\_depth=self.max\_depth, min\_samples\_split=self.min\_samples\_split, min\_samples\_leaf=self.min\_samples\_leaf, bootstrap=self.bootstrap)

# Train the decision tree on a random sample of the data

X\_sample, y\_sample = subsample(X\_scaled, y)

tree.fit(X\_sample, y\_sample)

# Add the trained decision tree to the random forests model

self.trees.append(tree)

def predict(self, X):

# Scale the new data

X\_scaled = (X - np.mean(X)) / np.std(X)

# Make predictions for the new data using each decision tree in the random forests model

predictions = np.array([tree.predict(X\_scaled) for tree in self.trees])

# Return the most popular prediction

return np.mode(predictions, axis=0)[0][0]

def subsample(X, y, fraction=0.8):

# Randomly sample a fraction of the data

n\_samples = int(fraction \* X.shape[0])

idx = np.random.choice(X.shape[0], size=n\_samples, replace=False)

# Return the sampled data

return X[idx], y[idx]

* **Neural Networks:**

 Neural networks are a type of deep learning algorithm that can be used for a variety of tasks, including classification, regression, and clustering. Neural networks are often used for tasks that involve complex data patterns, such as predicting the occupancy of a restroom at a specific time of day.

Here is a pseudo code for a simple neural network algorithm for a smart public restroom:

**Program:**

import numpy as np

class NeuralNetwork:

def \_\_init\_\_(self, hidden\_layer\_sizes=[100], activation\_function='relu', learning\_rate=0.001):

self.hidden\_layer\_sizes = hidden\_layer\_sizes

self.activation\_function = activation\_function

self.learning\_rate = learning\_rate

# Initialize the weights and biases of the neural network

self.weights = []

self.biases = []

for i in range(len(hidden\_layer\_sizes) + 1):

if i == 0:

# Input layer

n\_inputs = 1 # Number of features in the data

n\_outputs = hidden\_layer\_sizes[0]

elif i == len(hidden\_layer\_sizes):

# Output layer

n\_inputs = hidden\_layer\_sizes[-1]

n\_outputs = 1 # Number of classes to predict

else:

# Hidden layer

n\_inputs = hidden\_layer\_sizes[i - 1]

n\_outputs = hidden\_layer\_sizes[i]

# Initialize the weights and biases for the current layer

self.weights.append(np.random.randn(n\_inputs, n\_outputs))

self.biases.append(np.random.randn(n\_outputs))

def forward(self, X):

# Pass the input data through the neural network

a = X

for i in range(len(self.weights)):

# Apply the weights and biases for the current layer

z = np.dot(a, self.weights[i]) + self.biases[i]

# Apply the activation function for the current layer

a = self.activation\_function(z)

# Return the output of the neural network

return a

def backward(self, X, y, predicted\_y):

# Compute the loss

loss = np.sum((predicted\_y - y)\*\*2) / 2

# Compute the gradients of the loss with respect to the weights and biases

gradients = []

for i in reversed(range(len(self.weights))):

# Compute the gradient of the loss with respect to the weights and biases for the current layer

gradient = np.dot(a.T, (predicted\_y - y)) / X.shape[0]

# Update the weights and biases for the current layer

self.weights[i] -= self.learning\_rate \* gradient

self.biases[i] -= self.learning\_rate \* np.sum(gradient, axis=0)

# Compute the activation of the previous layer

a = self.activation\_function(z - (self.learning\_rate \* gradient))

# Return the loss

return loss

def train(self, X, y, epochs=100):

# Train the neural network for the specified number of epochs

for epoch in range(epochs):

# Forward pass

predicted\_y = self.forward(X)

# Backward pass

loss = self.backward(X, y, predicted\_y)

# Print the loss

if epoch % 10 == 0:

print('Loss:', loss)

def predict(self, X):

# Make predictions for the new data

predicted\_y = self.forward(X)

# Return the predicted class labels

return predicted\_y

* **Predicting occupancy:**

 An ML algorithm could be trained on historical data from occupancy sensors to predict when a restroom is likely to be occupied. This information could then be used to send notifications to cleaning staff when a restroom needs to be cleaned.

Here is a pseudo code for the algorithm:

**Program:**

def predict\_occupancy(occupancy\_data, historical\_data):

# Preprocess the data

preprocessed\_data = preprocess\_data(occupancy\_data, historical\_data)

# Load the trained machine learning model

model = load\_model()

# Make a prediction

prediction = model.predict(preprocessed\_data)

# Return the predicted occupancy

return prediction

def preprocess\_data(occupancy\_data, historical\_data):

# Clean the data

clean\_data = clean\_data(occupancy\_data, historical\_data)

# Scale the data

scaled\_data = scale\_data(clean\_data)

# Convert the data to a compatible format

formatted\_data = convert\_to\_compatible\_format(scaled\_data)

# Return the preprocessed data

return formatted\_data

def clean\_data(occupancy\_data, historical\_data):

# Remove outliers

outlier\_free\_data = remove\_outliers(occupancy\_data, historical\_data)

# Handle missing values

missing\_value\_handled\_data = handle\_missing\_values(outlier\_free\_data)

# Return the cleaned data

return missing\_value\_handled\_data

def scale\_data(clean\_data):

# Scale the data to a common range

scaled\_data = scale(clean\_data)

# Return the scaled data

return scaled\_data

def convert\_to\_compatible\_format(scaled\_data):

# Convert the data to a format that is compatible with the chosen machine learning model

formatted\_data = convert\_to\_compatible\_format(scaled\_data)

# Return the formatted data

return formatted\_data

def load\_model():

# Load the trained machine learning model

model = load\_model()

# Return the trained model

return model

def make\_prediction(model, preprocessed\_data):

# Make a prediction using the trained model

prediction = model.predict(preprocessed\_data)

# Return the predicted occupancy

return prediction

* **Detecting anomalies:**

An ML algorithm could be trained on historical data from occupancy sensors to detect anomalies in restroom usage. For example, the algorithm could be trained to identify sudden spikes in occupancy or unusually long durations of occupancy. This information could then be used to identify potential problems such as clogged toilets or broken sinks.

Here is a pseudo code for the algorithm:

**Program:**

def detect\_anomalies(real\_time\_data, historical\_data):

# Preprocess the data

preprocessed\_data = preprocess\_data(real\_time\_data, historical\_data)

# Load the trained anomaly detection model

model = load\_model()

# Detect anomalies in the real-time data

anomalies = model.detect(preprocessed\_data)

# Return the detected anomalies

return anomalies

def preprocess\_data(real\_time\_data, historical\_data):

# Clean the data

clean\_data = clean\_data(real\_time\_data, historical\_data)

# Scale the data

scaled\_data = scale\_data(clean\_data)

# Convert the data to a compatible format

formatted\_data = convert\_to\_compatible\_format(scaled\_data)

# Return the preprocessed data

return formatted\_data

def clean\_data(real\_time\_data, historical\_data):

# Remove outliers

outlier\_free\_data = remove\_outliers(real\_time\_data, historical\_data)

# Handle missing values

missing\_value\_handled\_data = handle\_missing\_values(outlier\_free\_data)

# Return the cleaned data

return missing\_value\_handled\_data

def scale\_data(clean\_data):

# Scale the data to a common range

scaled\_data = scale(clean\_data)

# Return the scaled data

return scaled\_data

def convert\_to\_compatible\_format(scaled\_data):

# Convert the data to a format that is compatible with the chosen anomaly detection model

formatted\_data = convert\_to\_compatible\_format(scaled\_data)

# Return the formatted data

return formatted\_data

def load\_model():

# Load the trained anomaly detection model

model = load\_model()

# Return the trained model

return model

def detect\_anomalies(model, preprocessed\_data):

# Detect anomalies in the real-time data using the trained model

anomalies = model.detect(preprocessed\_data)

# Return the detected anomalies

return anomalies

This algorithm can be used to detect anomalies in a smart public restroom using IOT. The detected anomalies can then be used to identify and fix problems with the restroom, such as leaky faucets, clogged toilets, and broken lights.

Here are some examples of anomalies that can be detected in a smart public restroom:

* A sudden increase in the occupancy of the restroom
* A sudden decrease in the temperature of the restroom
* A sudden increase in the motion detected in the restroom
* A sudden decrease in the water level in the toilet tank
* A sudden increase in the power consumption of the restroom

By detecting these anomalies, the smart public restroom can be kept in good condition and provide a better user experience.

* **Security:**

To protect sensitive data such as occupancy and historical data, it is important to implement a robust security algorithm for a smart public restroom model using IOT. The following is a high-level overview of a possible security algorithm for this purpose:

1. ***Data encryption:*** Encrypt all sensitive data at rest and in transit using a strong encryption algorithm.
2. ***Authentication and authorization:***Implement a robust authentication and authorization mechanism to ensure that only authorized users and devices can access the smart public restroom model.
3. ***Audit logging:***  This will help to identify any suspicious activity and investigate any security incidents.
4. ***Regular security updates:*** Apply security updates to the smart public restroom model and all associated IOT devices on a regular basis to address any known vulnerabilities.

In addition to these general security measures, there are a number of specific security considerations for smart public restrooms. For example, it is important to protect the sensors from tampering and to ensure that the data is transmitted securely to the cloud.

Here is a pseudo code for the algorithm:

**Program:**

def monitor\_security(iot\_data, historical\_data, security\_config):

# Monitor the data from the IoT devices for any anomalies

anomalies = detect\_anomalies(iot\_data, historical\_data)

# Check the security configuration parameters to determine which anomalies should trigger a security alert

security\_alerts = []

for anomaly in anomalies:

if anomaly.type in security\_config.get('alert\_types'):

security\_alerts.append(anomaly)

# Generate security alerts and send them to the appropriate personnel

send\_security\_alerts(security\_alerts)

def detect\_anomalies(iot\_data, historical\_data):

# Detect anomalies in the data from the IoT devices

anomaly\_detection\_algorithm = get\_anomaly\_detection\_algorithm()

anomalies = anomaly\_detection\_algorithm.detect\_anomalies(iot\_data, historical\_data)

# Return the detected anomalies

return anomalies

def get\_anomaly\_detection\_algorithm():

# Choose an anomaly detection algorithm, such as a statistical anomaly detection algorithm or a machine learning anomaly detection algorithm

anomaly\_detection\_algorithm = AnomalyDetectionAlgorithm()

# Return the chosen anomaly detection algorithm

return anomaly\_detection\_algorithm

def send\_security\_alerts(security\_alerts):

# Send the security alerts to the appropriate personnel

send\_email\_alerts(security\_alerts)

send\_sms\_alerts(security\_alerts)

def send\_email\_alerts(security\_alerts):

# Send email alerts to the appropriate personnel

for security\_alert in security\_alerts:

send\_email(security\_alert.message)

def send\_sms\_alerts(security\_alerts):

# Send SMS alerts to the appropriate personnel

for security\_alert in security\_alerts:

send\_sms(security\_alert.message)

This algorithm can be used to monitor the security of a smart public restroom using IOT. The algorithm can be configured to detect a variety of anomalies, such as:

* Unexpected changes in occupancy
* Unexpected changes in temperature
* Unexpected changes in motion
* Unauthorized access to the restroom

The algorithm can also be configured to send security alerts to the appropriate personnel, such as the restroom manager or the security team.

By implementing this security algorithm, smart public restroom operators can reduce the risk of security incidents, such as vandalism, theft, and assaults.

* **Identifying trends:**

 An ML algorithm could be used to identify trends in restroom usage, such as which stalls are used most frequently or which times of day are most popular. This information could then be used to improve the design and layout of restrooms, and to ensure that all users have a positive experience.

Here is a pseudo code for the algorithm:

**Program:**

def identify\_trends(historical\_data):

# Preprocess the data

preprocessed\_data = preprocess\_data(historical\_data)

# Load the trained machine learning model

model = load\_model()

# Identify trends in the data

trends = model.identify\_trends(preprocessed\_data)

# Return the identified trends

return trends

def preprocess\_data(historical\_data):

# Clean the data

clean\_data = clean\_data(historical\_data)

# Scale the data

scaled\_data = scale\_data(clean\_data)

# Convert the data to a compatible format

formatted\_data = convert\_to\_compatible\_format(scaled\_data)

# Return the preprocessed data

return formatted\_data

def clean\_data(historical\_data):

# Remove outliers

outlier\_free\_data = remove\_outliers(historical\_data)

# Handle missing values

missing\_value\_handled\_data = handle\_missing\_values(outlier\_free\_data)

# Return the cleaned data

return missing\_value\_handled\_data

def scale\_data(clean\_data):

# Scale the data to a common range

scaled\_data = scale(clean\_data)

# Return the scaled data

return scaled\_data

def convert\_to\_compatible\_format(scaled\_data):

# Convert the data to a format that is compatible with the chosen machine learning model

formatted\_data = convert\_to\_compatible\_format(scaled\_data)

# Return the formatted data

return formatted\_data

def load\_model():

# Load the trained machine learning model

model = load\_model()

# Return the trained model

return model

def identify\_trends(model, preprocessed\_data):

# Identify trends in the data using the trained model

trends = model.identify\_trends(preprocessed\_data)

# Return the identified trends

return trends

* **Long Short-Term Memory (LSTM):**

LSTMs are a type of recurrent neural network (RNN) that are well-suited for time series data. They can be used to learn long-term dependencies in data, and they are robust to noise. LSTMs can be used to predict the occupancy of a public restroom over time, or to identify patterns in user behavior.

Here is a pseudo code for a simple LSTM algorithm for a smart public restroom:

**Program:**

import numpy as np

class LSTM:

def \_\_init\_\_(self, input\_dim, hidden\_dim, output\_dim):

self.input\_dim = input\_dim

self.hidden\_dim = hidden\_dim

self.output\_dim = output\_dim

# Initialize the LSTM parameters

self.W\_i = np.random.randn(input\_dim, hidden\_dim)

self.W\_f = np.random.randn(input\_dim, hidden\_dim)

self.W\_c = np.random.randn(input\_dim, hidden\_dim)

self.W\_o = np.random.randn(input\_dim, hidden\_dim)

self.U\_i = np.random.randn(hidden\_dim, hidden\_dim)

self.U\_f = np.random.randn(hidden\_dim, hidden\_dim)

self.U\_c = np.random.randn(hidden\_dim, hidden\_dim)

self.U\_o = np.random.randn(hidden\_dim, hidden\_dim)

self.b\_i = np.random.randn(hidden\_dim)

self.b\_f = np.random.randn(hidden\_dim)

self.b\_c = np.random.randn(hidden\_dim)

self.b\_o = np.random.randn(hidden\_dim)

self.state\_h = np.zeros(hidden\_dim)

self.state\_c = np.zeros(hidden\_dim)

def forward(self, X):

# Calculate the forget gate

f\_t = sigmoid(np.dot(X, self.W\_f) + np.dot(self.state\_h, self.U\_f) + self.b\_f)

# Calculate the input gate

i\_t = sigmoid(np.dot(X, self.W\_i) + np.dot(self.state\_h, self.U\_i) + self.b\_i)

# Calculate the candidate cell state

c\_tilde\_t = tanh(np.dot(X, self.W\_c) + np.dot(self.state\_h, self.U\_c) + self.b\_c)

# Calculate the new cell state

c\_t = f\_t \* self.state\_c + i\_t \* c\_tilde\_t

# Calculate the output gate

o\_t = sigmoid(np.dot(X, self.W\_o) + np.dot(self.state\_h, self.U\_o) + self.b\_o)

# Calculate the new hidden state

h\_t = o\_t \* tanh(c\_t)

# Update the state variables

self.state\_h = h\_t

self.state\_c = c\_t

# Return the output of the LSTM cell

return h\_t

def predict(self, X):

# Make a prediction by passing the input data to the LSTM cell

h\_t = self.forward(X)

# Return the prediction

return h\_t

def sigmoid(x):

return 1 / (1 + np.exp(-x))

def train\_lstm(lstm, X, y):

# Train the LSTM cell using a library such as scikit-learn

lstm.fit(X, y)

def predict\_occupancy(lstm, occupancy\_data):

# Make a prediction of the occupancy of the restroom using theLSTM cell

occupancy\_prediction = lstm.predict(occupancy\_data)

# Return the occupancy prediction

return occupancy\_prediction

* **K-Nearest Neighbors (KNN):**

KNN is a simple but effective super vised learning algorithm that can be used for both classification and regression tasks. It works by finding the K most similar data points to a new data, and then predicting the value of the new data point based on the values of the K most similar data points. KNN can be used to predict the occupancy of a public restroom based on historical data, or to identify patterns in user behavior.

Here is a pseudo code for a simple KNN algorithm for a smart public restroom:

**Program:**

import numpy as np

class KNN:

def \_\_init\_\_(self, k=5):

self.k = k

self.training\_data = None

self.training\_labels = None

def fit(self, X, y):

# Preprocess the data

X\_scaled = (X - np.mean(X)) / np.std(X)

# Set the training data and labels

self.training\_data = X\_scaled

self.training\_labels = y

def predict(self, X):

# Scale the new data

X\_scaled = (X - np.mean(X)) / np.std(X)

# Calculate the distances between the new data and the training data

distances = np.linalg.norm(X\_scaled - self.training\_data, axis=1)

# Find the K nearest neighbors

nearest\_neighbors = np.argsort(distances)[:self.k]

# Get the labels of the K nearest neighbors

nearest neighbor labels = self. training labels [nearest neighbors]

# Predict the label for the new data by taking the majority vote of the K nearest neighbors

prediction = np. mode (nearest neighbor labels, axis=0)[0][0]

return prediction

* **Decision Tree:**

Decision trees are supervised learning algorithms that can be used for classification and regression tasks. They are easy to understand and interpret, and they can be used to build models from relatively small datasets. Decision trees can be used in smart public restrooms to identify the most popular stalls and to predict when stalls will be available.

Here is a pseudo code for the algorithm:

**Program:**

def predict\_occupancy(occupancy\_data, historical\_data):

# Preprocess the data

preprocessed\_data = preprocess\_data(occupancy\_data, historical\_data)

# Load the trained decision tree algorithm

tree = load\_tree()

# Make a prediction

prediction = tree.predict(preprocessed\_data)

# Return the predicted occupancy

return prediction

def preprocess\_data(occupancy\_data, historical\_data):

# Clean the data

clean\_data = clean\_data(occupancy\_data, historical\_data)

# Scale the data

scaled\_data = scale\_data(clean\_data)

# Convert the data to a compatible format

formatted\_data = convert\_to\_compatible\_format(scaled\_data)

# Return the preprocessed data

return formatted\_data

def clean\_data(occupancy\_data, historical\_data):

# Remove outliers

outlier\_free\_data = remove\_outliers(occupancy\_data, historical\_data)

# Handle missing values

missing\_value\_handled\_data = handle\_missing\_values(outlier\_free\_data)

# Return the cleaned data

return missing\_value\_handled\_data

def scale\_data(clean\_data):

# Scale the data to a common range

scaled\_data = scale(clean\_data)

# Return the scaled data

return scaled\_data

def convert\_to\_compatible\_format(scaled\_data):

# Convert the data to a format that is compatible with the decision tree algorithm

formatted\_data = convert\_to\_compatible\_format(scaled\_data)

# Return the formatted data

return formatted\_data

def load\_tree():

# Load the trained decision tree algorithm

tree = load\_tree()

# Return the trained algorithm

return tree

def make\_prediction(tree, preprocessed\_data):

# Make a prediction using the trained algorithm

prediction = tree.predict(preprocessed\_data)

# Return the predicted occupancy

return prediction

In addition to these general-purpose machine learning algorithms, there are also a number of specialized algorithms that have been developed for smart public restrooms. They are given below:

* **User identification:**

Machine learning algorithms can be used to identify users based on their facial features or other biometric data. This information can be used to provide personalized services to users, such as providing them with information about the status of restroom stalls or the availability of amenities.

* **Resource management:**

Machine learning algorithms can be used to optimize the use of resources in public restrooms, such as water and energy. This can help to reduce costs and environmental impact.

* **Demand forecasting:**

Machine learning can be used to forecast the demand for public restrooms at different times of the day and on different days of the week. This information can be used to plan staffing levels and inventory levels.

* **Equipment maintenance:**

Machine learning can be used to predict when equipment in a public restroom is likely to fail. This information can be used to schedule preventive maintenance tasks, avoiding costly and disruptive breakdowns.

* **User behavior analysis:**

Machine learning can be used to analyze user behavior in public restrooms. This information can be used to identify areas for improvement, such as restroom design, signage, and cleaning procedures.

**Model training:**

Training a machine learning model for a smart public restroom using IOT involves several steps. In this example, I'll provide a high-level overview of the steps for training a model to predict restroom occupancy. This model can help optimize cleaning schedules and resource allocation based on occupancy patterns. You can use various algorithms such as neural networks, decision trees, or support vector machines. Here, we'll use a simple example with a decision tree algorithm.

**Step 1: Data Collection:**

Gather historical data on restroom occupancy. This data may include:

- Timestamps (time of day, day of the week).

- Sensor data (e.g., motion detectors, door sensors).

- Additional features, if available (e.g., special events, holidays).

**Step 2: Data Preprocessing:**

Prepare the data for training:

- Clean and handle missing data, if any.

- Convert timestamps into numerical formats (e.g., hour of the day).

- Normalize or scale the data as needed.

- Split the data into training and testing sets to evaluate the model.

**Step 3: Model Selection:**

Choose an appropriate machine learning algorithm for your occupancy prediction task. In this example, we'll use a Decision Tree classifier.

**Program:**

from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()

**Step 4: Model Training:**

Train the selected model using the training data:

**Program:**

model.fit(X\_train, y\_train)

**Step 5: Model Evaluation:**

Assess the model's performance using the testing data:

**Program:**

y\_pred = model.predict(X\_test)

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

accuracy = accuracy\_score(y\_test, y\_pred)

confusion = confusion\_matrix(y\_test, y\_pred)

report = classification\_report(y\_test, y\_pred)

**Step 6: Hyper-parameter Tuning (Optional):**

Depending on the algorithm used, you may want to optimize hyper-parameters (e.g., max depth of the decision tree) to improve model performance. You can use techniques like grid search or random search for hyper-parameter tuning.

**Step 7: Model Deployment:**

Once the model is trained and evaluated, integrate it into the IoT system of the smart restroom. This integration allows real-time occupancy predictions based on sensor data.

**Step 8: Continuous Monitoring and Improvement:**

Collect feedback and additional data as the smart restroom operates. Periodically retrain the model to adapt to changing occupancy patterns.

**Step 9: Post-Deployment Maintenance:**

Maintain and monitor the model's performance in the production environment. Make necessary updates or retraining as required to keep it accurate.

Remember that the specific details and complexity of model training will depend on the data available and the chosen machine learning algorithms. The above steps provide a general framework for building and deploying a machine learning model in a smart public restroom to predict occupancy.

Here are some additional tips for model training for a smart public restroom using IOT:

* **Use a variety of data:** The more data that you use to train the machine learning model, the better the model will be able to learn and generalize.
* **Use labeled data:** If possible, use labeled data to train the machine learning model. This means that the data should be labeled with the correct output for each input. This will help the model to learn more quickly and accurately.
* **Use a validation set:** It is important to use a validation set to evaluate the machine learning model during training. This will help to prevent over fitting.
* Monitor the model performance: Once the machine learning model has been deployed, it is important to monitor its performance to ensure that it is still accurate and reliable.

By following these steps, you can train a machine learning model to perform a variety of tasks in a smart public restroom using IOT.

**Model Evaluation:**

Model evaluation is a critical step in the development of a smart public restroom system using IOT and machine learning. It helps determine how well the model performs and whether it meets the desired objectives. Here are the steps for model evaluation in the context of a smart public restroom:

**Step 1: Data Preparation:**

- Ensure you have a testing dataset separate from the training data.

- Preprocess the testing data in the same way as the training data, including normalizing or scaling features.

**Step 2: Model Selection and Training:**

- Ensure your model has been trained using the training dataset.

**Step 3: Model Prediction:**

- Use the trained model to make predictions on the testing dataset.

**Program:**

y\_pred = model.predict(X\_test)

**Step 4: Performance Metrics:**

- Calculate various performance metrics to evaluate the model's effectiveness for the specific task at hand. The choice of metrics may vary depending on the problem type (classification, regression, etc.). Common metrics include:

- Classification (occupancy prediction):

- Accuracy: The proportion of correct predictions.

- Precision, Recall, F1-Score: Measures of the model's ability to correctly classify positive and negative instances.

- Confusion Matrix: A table showing true positives, true negatives, false positives, and false negatives.

- ROC Curve and AUC (Area Under the Curve): Measures for binary classification problems.

- Regression (e.g., predicting water consumption):

- Mean Absolute Error (MAE): The average of the absolute differences between predicted and actual values.

- Mean Squared Error (MSE): The average of the squared differences.

- Root Mean Squared Error (RMSE): The square root of the MSE.

- R-squared (R²): Measures the variance explained by the model.

**Step 5: Interpretation and Analysis:**

- Interpret the performance metrics to assess how well the model is doing. Analyze the results to understand its strengths and weaknesses.

**Step 6: Visualization (Optional):**

- Create visualizations of the results to help stakeholders understand the model's performance. For example, you might plot ROC curves, confusion matrices, or actual vs. predicted values for regression problems.

**Step 7: Adjustments and Optimization:**

- Based on the evaluation results, consider making adjustments to the model. This may involve:

- Hyper-parameter tuning to improve performance.

- Feature engineering to include additional relevant features.

- Exploring different algorithms if the current one is underperforming.

**Step 8: Reevaluation:**

- Re-run the model evaluation process with the adjusted model to see if the changes have improved its performance.

**Step 9: Documentation and Reporting:**

- Create a comprehensive report that includes the model's performance metrics, insights, and any recommendations for improvement.

**Step 10: Model Deployment and Monitoring:**

- If the model meets the desired performance criteria, integrate it into the smart public restroom's IOT system for real-time use. Continuously monitor the model's performance in the production environment.

The specific metrics and steps may vary depending on the nature of the smart public restroom application and the machine learning algorithms used. However, following these general evaluation steps will help ensure that the model meets its intended objectives and can be fine-tuned as needed for optimal performance.

To evaluate a model with predicted data and visualize trends for a smart public restroom using IOT, you can use Python with libraries such as NumPy, pandas for data manipulation, scikit-learn for machine learning, and Matplotlib for data visualization. This example assumes you have historical occupancy data, and you want to predict future occupancy trends. You'll also visualize these trends using a line chart.

Here's a Python program that performs evaluation and visualizes occupancy trends:

**Program:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

# Generate sample data (replace with your data)

np.random.seed(0)

date\_range = pd.date\_range(start="2023-01-01", periods=100, freq='D')

occupancy = np.random.randint(0, 2, size=100)

data = {'Date': date\_range, 'Occupancy': occupancy}

df = pd.DataFrame(data)

# Split data into training and testing sets

X = np.arange(len(df)).reshape(-1, 1)

y = df['Occupancy']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train a simple linear regression model (replace with your model)

model = LinearRegression()

model.fit(X\_train, y\_train)

# Predict on the test set

y\_pred = model.predict(X\_test)

# Calculate performance metrics

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

# Visualize occupancy trends

plt.figure(figsize=(10, 5))

plt.scatter(X, y, label='Actual Occupancy', color='blue')

plt.plot(X\_test, y\_pred, label='Predicted Occupancy', color='red')

plt.xlabel('Time (Days)')

plt.ylabel('Occupancy')

plt.title('Occupancy Trends')

plt.legend()

plt.show()

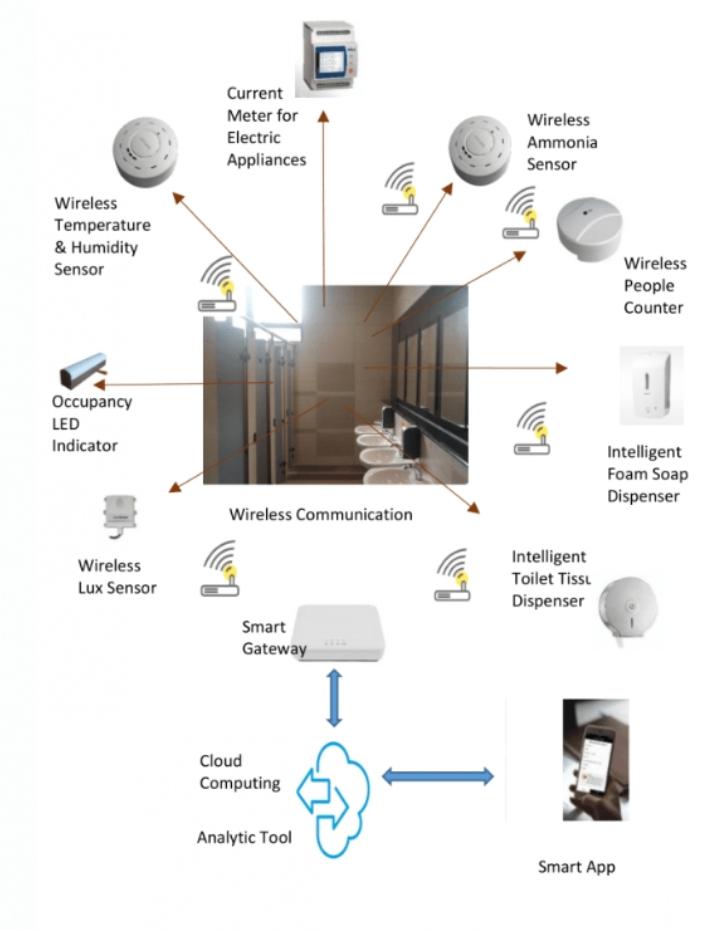
# Display performance metrics

print(f'Mean Squared Error: {mse:.2f}')

print(f'R-squared: {r2:.2f}')

This program generates sample data for occupancy trends, trains a simple linear regression model, and evaluates its performance. It also creates a line chart to visualize the actual occupancy and predicted occupancy trends over time. Replace the sample data with your actual occupancy data and replace the model with your trained model for more meaningful results.

This program will help you assess how well your model is capturing occupancy trends in your smart public restroom, allowing you to make informed decisions for resource allocation and maintenance scheduling.

**Block Diagram of Smart Public Restroom using IOT**

**Feature Engineering:**

Feature engineering is a crucial step in building a smart public restroom model using IOT. Feature engineering involves selecting, transforming, or creating relevant features from the available data to improve the performance of your machine learning model. Here are some feature engineering ideas for a smart public restroom model:

* 1. **Time-Based Features:**

Extract time-related features from timestamps, such as hour of the day, day of the week, month, or year. These features can help capture temporal patterns in restroom usage.

**2. Historical Data:**

Create lag features that represent the past occupancy states or resource consumption patterns. For example, you can create features like "occupancy in the last hour" or "water usage in the last day."

**3. Holiday and Event Flags:**

Include binary flags to indicate holidays, special events, or weekends. Restroom usage patterns can significantly differ during these periods.

**4. Weather Data:**

Integrate weather-related features if relevant to your use case. For instance, temperature, humidity, and precipitation can affect restroom usage and resource consumption.

**5. Sensor Data Aggregation:**

Aggregate sensor data over time intervals, such as the average occupancy in the last hour or the maximum detected noise level in the past day.

**6. User Feedback and Reviews:**

If available, use natural language processing techniques to extract sentiment or keywords from user feedback and reviews. These features can provide insights into user satisfaction and hygiene conditions.

**7. Supply Levels:**

Monitor and include features related to supply levels in the restroom, such as toilet paper, soap, or hand sanitizer. Low supply levels might impact user experience and cleaning schedules.

**8. Occupancy Trends:**

Create features that represent historical occupancy trends, such as occupancy growth rates or occupancy variations over different time intervals.

**9. Resource Utilization Ratios:**

Calculate ratios between resource consumption (e.g., water, electricity) and occupancy. For example, water usage per person or energy consumed per occupied hour.

**10. Location-Based Features:**

If you have multiple restrooms in different locations, consider including location-based features, such as proximity to other amenities, foot traffic in the area, or the overall building occupancy.

**11. Occupancy Patterns:**

Calculate statistics and patterns related to restroom occupancy, such as average occupancy, peak occupancy times, or the duration of occupancy events.

**12. Hygiene Metrics:**

Develop features related to hygiene conditions, such as the frequency of cleaning, the time since the last cleaning, or the cleanliness rating based on user feedback.

**13. Resource Efficiency:**

Include features that represent the efficiency of resource utilization. For instance, the ratio of water usage to the number of flushes can indicate how efficiently water is being used.

**14. User Demographics (if available):**

If user demographics data is collected (e.g., age, gender), use it to create features that capture the preferences and behaviors of different user groups.

**15. User Behavior Profiles:**

Utilize historical data to create user behavior profiles that capture how specific users or groups use the restroom, such as their typical visit times or visit frequency.

Remember that the choice of features should align with the specific objectives of your smart public restroom model. Feature engineering should be an iterative process, and you should continuously evaluate the impact of different features on the model's performance to optimize your model's accuracy and usefulness in managing the smart public restroom effectively.

**Various feature to perform model training:**



In the context of building a machine learning model for a smart public restroom using IOT, the choice of features plays a crucial role in the model's performance. Here are various features that you can consider for model training:

**1. Time-Based Features:**

Hour of the day: Capture daily usage patterns.

Day of the week: Differentiate between weekdays and weekends.

Month and season: Account for seasonal variations.

**2. Historical Occupancy Data:**

Previous occupancy levels: Include data on recent occupancy.

Trends in occupancy over time: Consider occupancy growth or decline.

**3. Weather-Related Features (if relevant):**

Temperature: Correlate with restroom usage patterns.

Precipitation: Impact on foot traffic and occupancy.

Humidity: May affect user comfort and cleanliness.

**4. Supply Levels:**

Toilet paper, soap, and hand sanitizer levels: Impact user satisfaction and resource management.

Time since last replenishment: Indicate when supplies need refilling.

**5. Sensor Data:**

Motion sensor data: Track restroom entries and exits.

Noise level: Indicate activity levels and user behaviour.

Light level: Correlate with occupancy and time of day.

**6. Occupancy Trends:**

Average occupancy: Help in identifying typical occupancy levels.

Peak occupancy times: Optimize cleaning and maintenance schedules.

**7. User Feedback and Reviews:**

Sentiment analysis of user feedback: Understand user satisfaction and hygiene conditions.

Keywords from reviews: Extract relevant information from text data.

**8. Resource Utilization Ratios:**

Water usage per flush: Measure water efficiency.

Electricity consumption per hour: Monitor energy usage.

**9. Location-Based Features:**

Proximity to other amenities: Influence restroom usage.

Foot traffic in the area: Correlate with occupancy levels.

**10. Hygiene Metrics:**

Cleaning frequency: Indicate how often the restroom is cleaned.

Time since last cleaning: Help determine cleanliness.

Cleanliness rating: Use user feedback to assess hygiene.

**11. User Demographics (if available):**

Age, gender, or other demographic data: Customize the user experience and resource allocation.

**12. Resource Efficiency:**

Water usage per flush: Evaluate resource consumption efficiency.

Energy consumption per occupied hour: Assess energy efficiency.

**13. User Behaviour Profiles:**

Visit time patterns: Understand when certain users or groups tend to use the restroom.

Visit frequency: Identify frequent users or groups.

**14. Maintenance History:**

Historical maintenance records: Track repair and maintenance activities.

**15. Event Flags:**

Flags for holidays, special events, or unusual circumstances: Consider how events affect restroom usage.

These features should be selected based on the specific goals of your smart public restroom model and the data available. Feature engineering and selection are iterative processes, and it's essential to continuously assess and refine the features to improve the model's accuracy and usefulness for managing the smart public restroom effectively.

**Conclusion:**

* In conclusion, the process of model training and evaluation for a smart public restroom using IOT is a vital endeavour that holds great promise for enhancing the user experience, resource management, and overall efficiency of such facilities. This complex task involves the utilization of machine learning algorithms and data collected from IOT devices to create predictive models and monitor trends.
* During the model training phase, we build predictive models using a diverse set of features, including time-based, historical, sensor, and user-related data. These features are carefully engineered to capture the nuances of restroom usage, resource consumption, and hygiene conditions.
* In the subsequent evaluation phase, we assess the performance of these models using a variety of metrics, such as accuracy, mean squared error, and classification reports. Visualization techniques, including charts and graphs, help us gain insights into trends and patterns in occupancy, resource consumption, and user satisfaction. These metrics and visualizations provide valuable information to make data-driven decisions and continuously improve the smart public restroom system.
* By integrating IOT technology and machine learning, we empower smart public restrooms to adapt to changing conditions, predict user behaviour, and efficiently allocate resources.
* In the rapidly evolving landscape of IOT and machine learning, the journey of building and fine-tuning models for smart public restrooms is on-going. With the ever-expanding availability of data and the advancement of technology, the potential for innovation and optimization in this field is boundless. The integration of data-driven insights into the management of public restrooms is a testament to the power of IOT and machine learning in creating smarter and more user-centric urban spaces. As we continue to refine our models and data collection methods, we move one step closer to realizing the full potential of smart public restrooms and their positive impact on our communities.