MACHINE LEARNING MODEL FOR OCCUPANCY RATES AND DEMAND IN THE HOSPITALITY INDUSTRY

Final Project Report

1. INTRODUCTION

1.1 Project overview

The project aims to analyze occupancy rate and demand patterns in hospitality industry to provide insights for finding hotel room with efficiency and cost-saving measures. By analyzing historical occupancy data along with other relevant factors such as weather conditions, occupancy patterns, and all, the project seeks to identify trends and patterns that can help customers optimize their model usage.

1.2 Objectives

Occupancy Pattern Identification

The project analyzes historical occupancy consumption data to identify patterns in model usage, such as peak usage times, seasonal variations, and the impact of weather conditions. This information can help customers adjust their model usage to reduce costs.

Cost-Saving Recommendations

Based on the analysis of occupancy consumption patterns and model efficiency, the project provides recommendations to customers on how to reduce theirtime consumption and save costs. This could include tips on adjusting using occupancy model during needed hours, or investing in occupancy rates and demand-efficient hotels.

2. Project Initialization and Planning Phase

2.1 Define Problem Statement

A customer concerned with time efficiency and cost savings wants to optimize the occupancy to reduce costs, posing a challenge lacking the necessary knowledge and tools to monitor and analyze energy usage effectively.

2.2 Project Proposal (Proposed solution)

- The proposed project, "Machine Learning Model for Occupancy Rates and Demand in the Hospitality Industry," aims to leverage machine learning for more accurate solutions.
- Using a comprehensive dataset including occupancy, humidity, humidity ratio, CO2, light, the project seeks to develop a model for optimizing the occupancy.

- This initiative aligns with the Power consumption analysis objective to provide insights for energy efficiency and cost-saving measures and provides recommendations to households on how to reduce their energy consumption and save costs.

2.3 Initial Project Planning

- Initial Project Planning involves outlining key objectives, defining scope, and identifying the occupancy rates and demand.
- It encompasses setting timelines, allocating resources, and determining the overall project strategy.

3. Data Collection and Preprocessing Phase

3.1 Data Collection Plan and Raw Data Sources Identified

- The dataset for " Machine Learning Model for Occupancy Rates and Demand in the Hospitality Industry " is sourced from Kaggle.
- It includes detailed measurements taken over time.
- Date: Date in format dd/mm/yyyy
- Time: time in format hh:mm:ss
- Occupancy: It gives information about occupancy.
- Humidity: Humidity level will be shown in this phase.
- Humidity Ratio: Humidity Ratio is clearly mentioned here.
- CO2: The amount of CO2 will be provided here.

3.2 Data Quality Report

- Data quality is ensured through thorough verification, addressing missing values, and maintaining adherence to ethical guidelines, establishing a reliable foundation for predictive modeling.

3.3 Data Exploration and preprocessing

- Data Exploration involves analyzing the customer demand and training dataset to understand patterns, distributions, and outliers.
- Preprocessing includes handling missing values, scaling, and encoding categorical variables.
- These crucial steps enhance data quality, ensuring the reliability and effectiveness of subsequent analysis.

4. Model Development Phase

4.1 Feature Selection Report

- The Feature Selection Report outlines the rationale behind choosing specific features (e.g., Humidity, Humidity Ratio, CO_2, Light, Occupancy) for the occupancy model.
- It evaluates relevance, importance, and impact on predictive accuracy, ensuring the inclusion of key factors influencing the model's ability.

4.2 Model Selection Report

- The Model Selection Report details the rationale behind choosing Logistic Regression, SVC, Decision Tree Classifier, and K-Neighbors Classifier for occupancy model.
- It considers each model's strengths in handling complex relationships, interpretability, adaptability, and overall predictive performance, ensuring an informed choice aligned with project objectives.

4.3 Initial Model Training Code, Model Validation and Evaluation Report

- The Initial Model Training Code employs selected algorithms on the training and occupancy dataset, setting the foundation for predictive modeling.

- The subsequent Model Validation and Evaluation Report rigorously assesses model performance, employing metrics like Accuracy, Weighted avg, Macro avg error to ensure reliability and effectiveness in predicting occupancy demand.

5. Model Optimization and Tuning Phase

Final Model Selection Justification

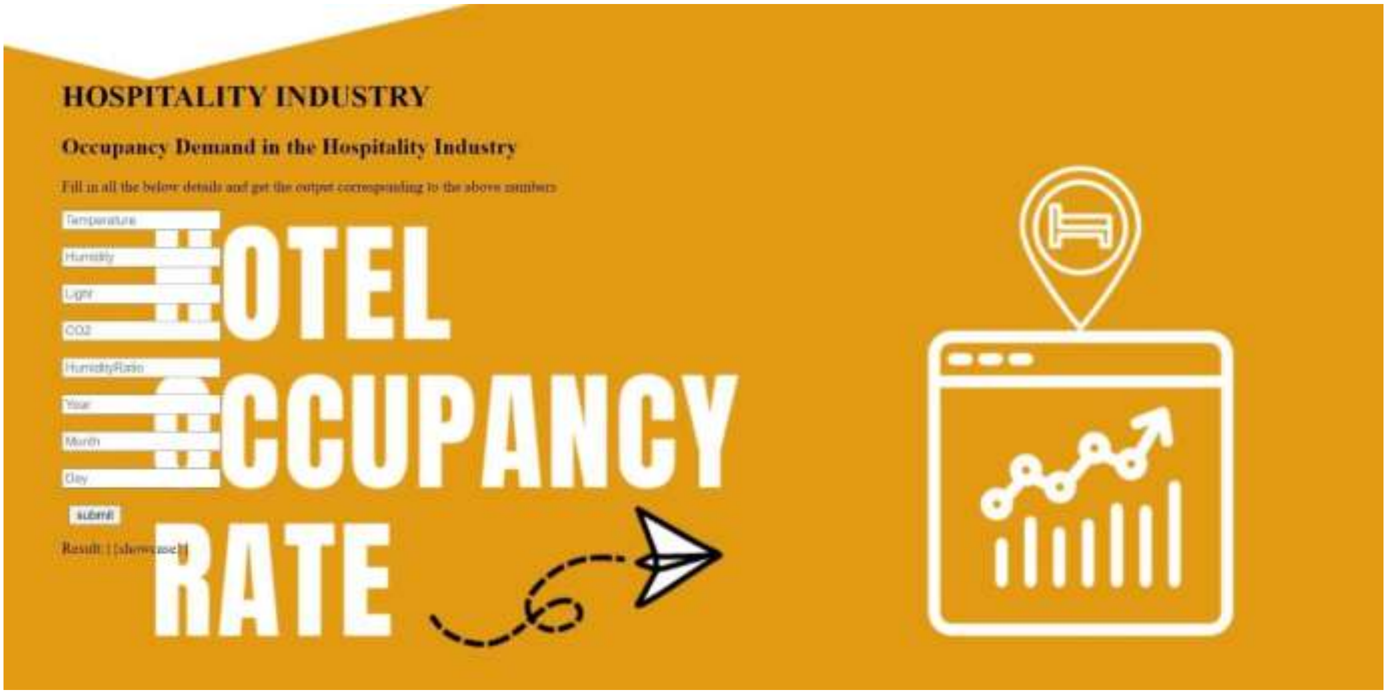
- The K-Neighbors Classifier is the final model chosen because of its best overall performance compared to the other models.
- It captures the Accuracy in the data very well with minimal prediction error. K-Neighbors Classifier can capture complex non-linear relationships.

6. RESULTS

6.1 Output Screenshots

PCA.HTML

HOME PAGE



OUTPUT PAGE

RESULT.HTML

7. Advantages and disadvantages

7. ADVANTAGES AND DISADVANTAGES

Advantages:

1. **Insight into Usage Patterns**: Occupancy analysis provides detailed insights into how time is used within customers, identifying peak usage times, high usage times.

2. **Cost Savings**: By understanding usage patterns, customers can implement strategies to reduce consumption during peak hours or adjust usage behaviors.
3. **Smart Decision Making**: Data-driven insights enable informed decision-making regarding appliance upgrades, time-efficient investments, and behavioural changes that optimize hotels use.

Disadvantages:

1. **Cost of Implementation**: Initial costs associated with installing and acquiring advanced occupancy monitoring systems may be prohibitive for some customer.
2. **Privacy Concerns**: Continuous monitoring of model usage raises privacy concerns regarding data collection, storage, and potential misuse of personal information.
3. **Technical Complexity**: Analysing and interpreting energy data requires technical expertise and resources, which may be challenging for customers without access to specialized knowledge or support.

8. CONCLUSION

- In conclusion, the analysis of occupancy rates and demands through advanced monitoring and data analytics presents significant opportunities for optimizing time usage and promoting sustainability. This project has demonstrated the effectiveness of smart monitoring technology coupled with sophisticated data analysis techniques in providing detailed insights into customers demands patterns. By capturing real-time data and applying statistical analysis and machine learning and deep learning algorithms, the project has identified peak usage times, inefficient practices, and opportunities for improvement.

9. FUTURE SCOPE

- **Integration of Machine Learning and Deep Learning**: The integration of **Machine Learning (ML)** and with rates and demand, analysis systems will enable more granular data collection and real-time monitoring.
- **Advanced Data Analytics and AI**: Future advancements in data analytics, machine learning, and artificial intelligence (AI) will enable more sophisticated analysis of time consumption patterns. Predictive analytics models can forecast occupancy demand, identify anomalies, and offer proactive recommendations for optimizing time and cost efficiency based on historical and real-time data.
- **Demand Response Programs**: Increased participation in demand response programs facilitated by time and cost consumption analysis systems will enable customers to adjust their needs.
- **Global Adoption and Standardization**: Increasing global adoption of time and energy analysis technologies will drive economies of scale, reducing costs and improving accessibility for customers worldwide. Standardization of measurement methodologies and data formats will facilitate interoperability and compatibility across different systems and regions.

10. APPENDIX

10.1. SOURCE CODE

INDEX.HTML

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Occupancy Rates And Demand in the Hospatality Industry</title>

   <style>

      body{

         margin: auto;

         padding: 5%;

         background: url('https://www.makcorps.com/blog/wp-
content/uploads/2022/11/hotel-occupancy-rate.png');

         background-repeat: no-repeat;

         background-position: justify;

         background-position-x: center;

         background-size: cover;

      }
```

```
        .page{
            text-align: center;
        }
        p{
          text-size: 10px;


        }
    </style>
</head>
<body>
    <div name="page">
    <h1>HOSPITALITY INDUSTRY</h1>
    <h2>Occupancy Demand in the Hospitality Industry</h2>
    <p>Fill in all the below details and get the output corresponding to the above numbers
</p>
    <form action="/prediction" method="POST">
    <input type="text" name="Temperature" placeholder="Temperature">
    <br><br>
    <input type="text" name="Humidity" placeholder="Humidity">
    <br><br>
    <input type="text" name="Light" placeholder="Light">
    <br><br>
    <input type="text" name="CO2" placeholder="CO2">
    <br><br>
    <input type="text" name="HumidityRatio" placeholder="HumidityRatio">
```

```
    <br><br>
    <input type="text" name="year" placeholder="Year">
    <br><br>
    <input type="text" name="month" placeholder="Month">
    <br><br>
    <input type="text" name="day" placeholder="Day">
    <br><br>
      <button type="submit"> submit</button>
    </div>
    </form>
    <p>Result:{{showcase}}</p>
</body>
</html>
```

APP.PY

```
#importing libraries
from flask import Flask, render_template, request
import pickle
import numpy as np


app = Flask(__name__)


model = pickle.load(open('occupancy.pkl', 'rb'))


@app.route('/')
def home():
```

```
    return render_template("index.html")


@app.route('/prediction', methods=['POST', 'GET'])
def predict():
    Temperature = float(request.form['Temperature'])
    Humidity = float(request.form['Humidity'])
    Light = float(request.form['Light'])
    CO2 = float(request.form['CO2'])
    HumidityRatio = float(request.form["HumidityRatio"])
    year = int(request.form['year'])
    month = int(request.form['month'])
    day = int(request.form['day'])

    total = [[Temperature, Humidity, Light, CO2, HumidityRatio, year, month, day]]
    y_test=model.predict(total)
    print(y_test)

    if(y_test==[0]):
        ans="It is not Occupied"
    else:
        ans="It is Occupied"
        return render_template("index.html", showcase = ans)


if __name__=="__main__":
    app.run(debug=False)
```

Code Snippets

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
df=pd.read_csv("datatraining.csv",header=0)
df.shape
```

```
(8143, 7)
```

```
df.head()
```

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
1	2015-02-04 17:51:00	23.18	27.2720	426.0	721.25	0.004793	1
2	2015-02-04 17:51:59	23.15	27.2675	429.5	714.00	0.004783	1
3	2015-02-04 17:53:00	23.15	27.2450	426.0	713.50	0.004779	1
4	2015-02-04 17:54:00	23.15	27.2000	426.0	708.25	0.004772	1
5	2015-02-04 17:55:00	23.10	27.2000	426.0	704.50	0.004757	1

```
df.tail()
```

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
8139	2015-02-10 09:29:00	21.05	36.0975	433.0	787.250000	0.005579	1
8140	2015-02-10 09:29:59	21.05	35.9950	433.0	789.500000	0.005563	1
8141	2015-02-10 09:30:59	21.10	36.0950	433.0	798.500000	0.005596	1
8142	2015-02-10 09:32:00	21.10	36.2600	433.0	820.333333	0.005621	1
8143	2015-02-10 09:33:00	21.10	36.2000	447.0	821.000000	0.005612	1

```
df.isnull().any()
```

```
date             False
Temperature      False
Humidity         False
Light            False
CO2              False
HumidityRatio    False
Occupancy        False
dtype: bool
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8143 entries, 1 to 8143
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   date           8143 non-null   object
 1   Temperature    8143 non-null   float64
 2   Humidity       8143 non-null   float64
 3   Light          8143 non-null   float64
 4   CO2            8143 non-null   float64
 5   HumidityRatio  8143 non-null   float64
 6   Occupancy      8143 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 767.0+ KB
```

```
df[["year", "month", "day"]] = df["date"].str.split("-", expand = True)
df.head()
```

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	year	month	day
1	2015-02-04 17:51:00	23.18	27.2720	426.0	721.25	0.004793	1	2015	02	04 17:51:00
2	2015-02-04 17:51:59	23.15	27.2675	429.5	714.00	0.004783	1	2015	02	04 17:51:59
3	2015-02-04 17:53:00	23.15	27.2450	426.0	713.50	0.004779	1	2015	02	04 17:53:00
4	2015-02-04 17:54:00	23.15	27.2000	426.0	708.25	0.004772	1	2015	02	04 17:54:00
5	2015-02-04 17:55:00	23.10	27.2000	426.0	704.50	0.004757	1	2015	02	04 17:55:00

```
df.drop(['date'], axis=1,inplace=True)
df
```

	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	year	month	day
1	23.18	27.2720	426.0	721.250000	0.004793	1	2015	02	04 17:51:00
2	23.15	27.2675	429.5	714.000000	0.004783	1	2015	02	04 17:51:59
3	23.15	27.2450	426.0	713.500000	0.004779	1	2015	02	04 17:53:00
4	23.15	27.2000	426.0	708.250000	0.004772	1	2015	02	04 17:54:00
5	23.10	27.2000	426.0	704.500000	0.004757	1	2015	02	04 17:55:00
...	...	...	...	...	...	...	...	...	...
8139	21.05	36.0975	433.0	787.250000	0.005579	1	2015	02	10 09:29:00
8140	21.05	35.9950	433.0	789.500000	0.005563	1	2015	02	10 09:29:59
8141	21.10	36.0950	433.0	798.500000	0.005596	1	2015	02	10 09:30:59
8142	21.10	36.2600	433.0	820.333333	0.005621	1	2015	02	10 09:32:00
8143	21.10	36.2000	447.0	821.000000	0.005612	1	2015	02	10 09:33:00

8143 rows × 9 columns

```
[ ]  df.dtypes
```

```
Temperature      float64
Humidity         float64
Light            float64
CO2              float64
HumidityRatio    float64
Occupancy          int64
year              object
month             object
day               object
dtype: object
```

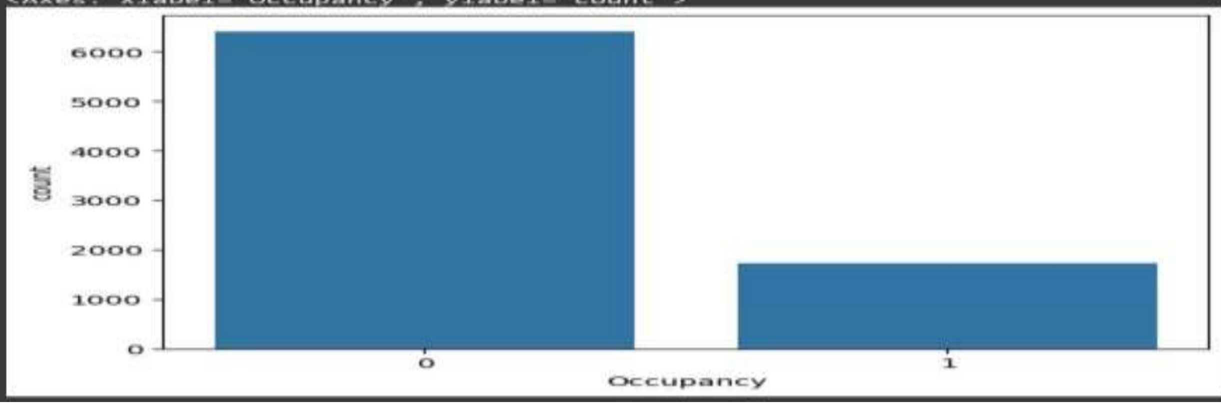
```
df.isnull().sum()
```

```
Temperature      0
Humidity         0
Light            0
CO2              0
HumidityRatio    0
Occupancy        0
year             0
month            0
day              0
dtype: int64
```

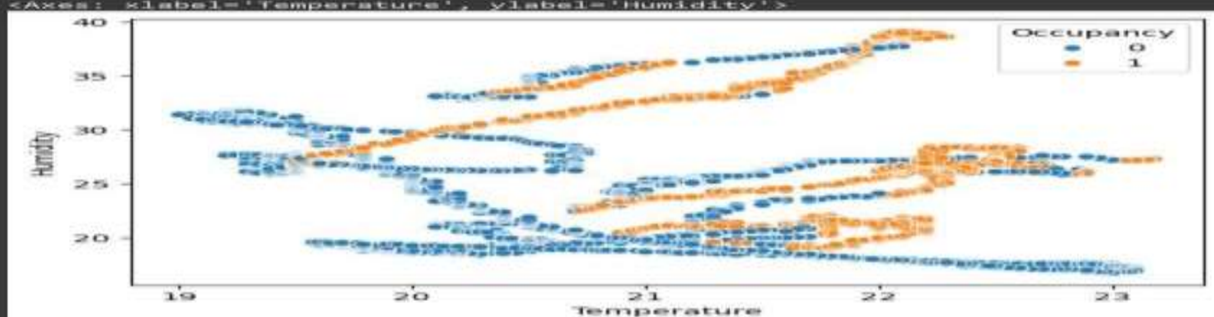
```
df.describe()
```

	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
count	8143.000000	8143.000000	8143.000000	8143.000000	8143.000000	8143.000000
mean	20.619084	25.731507	119.519375	606.546243	0.003863	0.212330
std	1.016916	5.531211	194.755805	314.320877	0.000852	0.408982
min	19.000000	16.745000	0.000000	412.750000	0.002674	0.000000
25%	19.700000	20.200000	0.000000	439.000000	0.003078	0.000000
50%	20.390000	26.222500	0.000000	453.500000	0.003801	0.000000
75%	21.390000	30.533333	256.375000	638.833333	0.004352	0.000000
max	23.180000	39.117500	1546.333333	2028.500000	0.006476	1.000000

```
sns.countplot(x='Occupancy',data=df)
<Axes: xlabel='Occupancy', ylabel='count'>
```




```
sns.scatterplot(x='Temperature', y='Humidity', hue='Occupancy', data=df)
<Axes: xlabel='Temperature', ylabel='Humidity'>
```



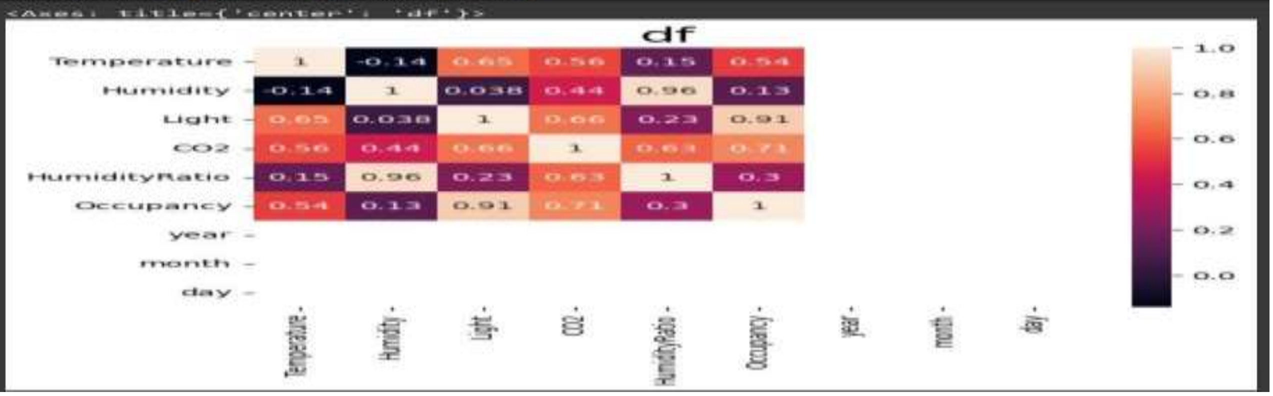

```
[22] df['day'] = pd.to_numeric(df['day'], errors='coerce')
```

Insert code cell below
Ctrl+M B

	...ure	Humidity	Light	CO2	HumidityRatio	Occupancy	year	month	day
1	23.18	27.2720	426.0	721.250000	0.004793	1	2015	02	NaN
2	23.15	27.2675	429.5	714.000000	0.004783	1	2015	02	NaN
3	23.15	27.2450	426.0	713.500000	0.004779	1	2015	02	NaN
4	23.15	27.2000	426.0	708.250000	0.004772	1	2015	02	NaN
5	23.10	27.2000	426.0	704.500000	0.004757	1	2015	02	NaN
...	...	...	...	...	...	...	...	...	...
8139	21.05	36.0975	433.0	787.250000	0.005579	1	2015	02	NaN
8140	21.05	35.9950	433.0	789.500000	0.005563	1	2015	02	NaN
8141	21.10	36.0950	433.0	798.500000	0.005596	1	2015	02	NaN
8142	21.10	36.2600	433.0	820.333333	0.005621	1	2015	02	NaN
8143	21.10	36.2000	447.0	821.000000	0.005612	1	2015	02	NaN

8143 rows × 9 columns

```
plt.title("df", y=1,size=20)
sns.heatmap(df.corr(),annot=True)
<Axes: title={'center': 'df'}>
```



```python
from sklearn.preprocessing import LabelEncoder
label_encoder_x=LabelEncoder()
df['year']=label_encoder_x.fit_transform(df['year'])
df['month']=label_encoder_x.fit_transform(df['month'])
df['day']=label_encoder_x.fit_transform(df['day'])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8143 entries, 1 to 8143
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Temperature     8143 non-null   float64
 1   Humidity        8143 non-null   float64
 2   Light           8143 non-null   float64
 3   CO2             8143 non-null   float64
 4   HumidityRatio   8143 non-null   float64
 5   Occupancy       8143 non-null   int64
 6   year            8143 non-null   int64
 7   month           8143 non-null   int64
 8   day             8143 non-null   int64
dtypes: float64(5), int64(4)
memory usage: 894.2 KB
```

```
[26] df
```

	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	year	month	day
1	23.18	27.2720	426.0	721.250000	0.004793	1	0	0	0
2	23.15	27.2675	429.5	714.000000	0.004783	1	0	0	0
3	23.15	27.2450	426.0	713.500000	0.004779	1	0	0	0
4	23.15	27.2000	426.0	708.250000	0.004772	1	0	0	0
5	23.10	27.2000	426.0	704.500000	0.004757	1	0	0	0
...	...	...	...	...	...	...	...	...	...
8139	21.05	36.0975	433.0	787.250000	0.005579	1	0	0	0
8140	21.05	35.9950	433.0	789.500000	0.005563	1	0	0	0
8141	21.10	36.0950	433.0	798.500000	0.005596	1	0	0	0
8142	21.10	36.2600	433.0	820.333333	0.005621	1	0	0	0
8143	21.10	36.2000	447.0	821.000000	0.005612	1	0	0	0

8143 rows × 9 columns

Next steps: Generate code with df View recommended plots

```
[27] y=df.iloc[:,5:6].values
     x = df.drop('Occupancy', axis=1)
```

```
[ ] x.shape
(8143, 8)

[ ] y.shape
(8143, 1)

[ ] from sklearn.model_selection import train_test_split
    x_train,x_test,y_train,y_test=train_test_split(x, y, test_size=0.2,random_state=0)
    x_test.shape
(1629, 8)

[ ] x_train.shape
(6514, 8)

[ ] y_train.shape
(6514, 1)

[ ] y_test.shape
(1629, 1)
```

```
[ ] from sklearn.preprocessing import StandardScaler
    sc_x=StandardScaler()
    x_train=sc_x.fit_transform(x_train)
    x_test=sc_x.transform(x_test)
```

```
[ ] from sklearn.linear_model import LogisticRegression
    lr = LogisticRegression()
    lr.fit(x_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataC
  y = column_or_1d(y, warn=True)
▾ LogisticRegression
LogisticRegression()
```

```
[ ] from sklearn.metrics import accuracy_score, classification_report
    y_pred = lr.predict(x_test)
    print(accuracy_score(y_test, y_pred))
    print(classification_report(y_test, y_pred))
```

```
0.9864947820748926
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      1264
           1       0.96      0.98      0.97       365

    accuracy                           0.99      1629
   macro avg       0.98      0.99      0.98      1629
weighted avg       0.99      0.99      0.99      1629
```

```
[ ] from sklearn.tree import DecisionTreeClassifier
    classifier = DecisionTreeClassifier(random_state = 0)
    classifier.fit(x_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

```
[ ] ypred=classifier.predict(x_test)
    from sklearn.metrics import accuracy_score, classification_report
    print(accuracy_score(y_test, ypred))
    print(classification_report(y_test, ypred))
```

```
0.9920196439533456
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      1264
           1       0.98      0.98      0.98       365

    accuracy                           0.99      1629
   macro avg       0.99      0.99      0.99      1629
weighted avg       0.99      0.99      0.99      1629
```

```
[ ] from sklearn.svm import SVC
    sv=SVC()
    sv.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py
  y = column_or_1d(y, warn=True)
SVC
SVC()
```

```
[ ] ypred2=sv.predict(x_test)
    from sklearn.metrics import accuracy_score, classification_report
    print(accuracy_score(y_test, ypred2))
    print(classification_report(y_test, ypred2))
```

```
0.9901780233271946
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      1264
           1       0.96      1.00      0.98       365

    accuracy                           0.99      1629
   macro avg       0.98      0.99      0.99      1629
weighted avg       0.99      0.99      0.99      1629
```

```
from sklearn.neighbors import KNeighborsClassifier
Kn=KNeighborsClassifier()
Kn.fit(x_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/_classification.py:215: DataConversionWarni
  return self._fit(X, y)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

```
ypred3=Kn.predict(x_test)
from sklearn.metrics import accuracy_score, classification_report
print(accuracy_score(y_test, ypred3))
print(classification_report(y_test, ypred3))
```

```
0.9883364027010436
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1264
           1       0.97      0.98      0.97       365

    accuracy                           0.99      1629
   macro avg       0.98      0.98      0.98      1629
weighted avg       0.99      0.99      0.99      1629
```

```
classifier.predict([[23.18, 27.2720,  426.0,  721.250000, 0.004793,0,0,0]])
```

```
array([1])
```

```python
import pickle
pickle.dump(classifier,open('occupancy.pkl','wb'))
```

.HTML:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Occupancy Rates And Demand in the Hospatality Industry</title>
    <style>         body{margin:     auto;
padding: 5%;               background:
url('https://www.makcorps.com/blog/wpcontent/uploads/2022/11/hotel-
occupancy-rate.png');              background-repeat: no-repeat;
background-position: justify;              background-position-x:
center;              background-size: cover;
        }         .page{
text-align: center;
        }         p{
text-size: 10px;

        }
    </style>
</head>
<body>
    <div name="page">
    <h1>HOSPITALITY INDUSTRY</h1>
    <h2>Occupancy Demand in the Hospitality Industry</h2>
    <p>Fill in all the below details and get the output corresponding to the above numbers
</p>
    <form action="/prediction" method="POST">
    <input type="text" name="Temperature" placeholder="Temperature">      <br><br>
    <input type="text" name="Humidity" placeholder="Humidity">      <br><br>
    <input type="text" name="Light" placeholder="Light">
```

```
    <br><br>
    <input type="text" name="CO2" placeholder="CO2">
<br><br>
    <input type="text" name="HumidityRatio" placeholder="HumidityRatio">
<br><br>
    <input type="text" name="year" placeholder="Year">
    <br><br>
    <input type="text" name="month" placeholder="Month">
    <br><br>
    <input type="text" name="day" placeholder="Day">
<br><br>
      <button type="submit"> submit</button>
    </div>
    </form>
    <p>Result:{{showcase}}</p>
</body>
</html>
```

App.py:

```
#importing libraries from flask import Flask,
render_template, request  import pickle import
numpy as np
 app =
Flask(__name__)
 model = pickle.load(open('occupancy.pkl',
'rb'))

@app.route('/') def
home():
    return render_template("index.html")

@app.route('/prediction', methods=['POST', 'GET'])
def predict():
    Temperature = float(request.form['Temperature'])
    Humidity = float(request.form['Humidity'])
    Light = float(request.form['Light'])
    CO2 = float(request.form['CO2'])
    HumidityRatio =
float(request.form["HumidityRatio"])      year =
int(request.form['year'])      month =
int(request.form['month'])      day =
int(request.form['day'])
         total = [[Temperature, Humidity, Light, CO2, HumidityRatio, year, month,
day]]
```

```
    y_test=model.predict(total)
print(y_test)

if(y_test==[0]):
        ans="It is not Occupied"
else:
        ans="It is Occupied"          return
render_template("index.html", showcase = ans)
 if
_name_=="_main_":
app.run(debug=False)
```

10.2 GitHub and project Demo link:

Github link:

https://github.com/abhishek369915/mini-project-occupancy-rate

Project Demo link:

https://drive.google.com/drive/folders/1pcK5hBaWP95J4GXwurZX1qMORbVtb8bK