Machine Learning Model for Occupancy Rates and Demand in the Hospitality Industry

Team No - 212

Team Members:
Prajwal Kulkarni
Shreyas Chitransh
Bhuwan Rathi

Table of Contents

1. INTRODUCTION

1.1. Overview : A brief description of the project

The hospitality industry heavily relies on accurate predictions of occupancy rates and demand to optimize operations and resource allocation. Machine learning techniques offer promising solutions to forecast occupancy rates, enabling hotels to make informed decisions regarding staffing, pricing, and resource management. This project aims to develop a machine learning model that predicts whether a hotel room will be occupied or vacant based on various parameters.

We aim to create a machine learning model that is specifically designed for forecasting occupancy rates in the hospitality sector as part of this project. The model will evaluate the occupancy of a hotel or lodging facility by taking into account a number of variables, including the date, temperature, humidity, light, CO2 levels, and humidity. We can capture the impact of environmental factors on occupancy rates and demand by including these elements in the model.

1.2. Purpose : The use of the project

The project focuses on developing a predictive model using machine learning algorithms to forecast occupancy rates and demand in the hospitality industry. The model utilizes historical data, including date, time, and other relevant parameters, to make predictions about the future occupancy status of hotel rooms.

a. Revenue Optimization: By accurately forecasting occupancy rates and demand, hotels and other accommodation establishments can optimize their revenue generation. They can adjust their pricing strategies based on predicted high-demand periods to maximize room rates. Conversely, during periods of low occupancy, they can offer discounted rates or promotional deals to attract more guests and fill their rooms.

b. Resource Allocation: Accurate occupancy rate predictions enable businesses to allocate their resources effectively. They can adjust staffing levels based on expected occupancy to ensure adequate personnel for guest services, housekeeping, and other operational needs. This avoids overstaffing during low-demand periods and understaffing during high-demand periods.
c. Operational Efficiency: With occupancy rate predictions, hotels can optimize their operational processes. For example, they can plan room cleaning schedules based on anticipated check-out and check-in times, leading to efficient utilization of housekeeping staff and minimizing wait times for guests. They can also streamline inventory management for amenities and supplies to match the expected number of guests.
d. Marketing Campaigns: The occupancy rate predictions can assist in planning targeted marketing campaigns. Hotels can identify periods of low occupancy and create marketing promotions and packages to attract guests during those times. They can also leverage the predicted high-demand periods to focus their marketing efforts on specific customer segments or events that are likely to drive more bookings.
e. Customer Satisfaction: Accurate occupancy rate predictions allow hotels to ensure availability for their guests. By understanding occupancy patterns, they can manage guest expectations and provide a better experience. This includes ensuring that guests can make reservations even during peak periods, minimizing the risk of overbooking, and delivering personalized services based on predicted guest preferences.

2. LITERATURE SURVEY

2.1. Existing Problems:

The existing problem in the hospitality industry lies in accurately predicting occupancy rates and demand. Traditional methods often rely on historical patterns and manual analysis, which may not capture the dynamic nature of demand fluctuations accurately.

a. Limited Accuracy: Traditional methods of forecasting occupancy rates in the hospitality industry may lack accuracy due to their reliance on historical data and expert judgment alone. They may not effectively capture the complex interplay of factors that influence occupancy rates, such as seasonality, local events, and economic indicators.
b. Data Availability and Quality: Obtaining accurate and comprehensive data for occupancy rate prediction can be challenging. Data collection processes may be fragmented, and data quality issues such as missing values or inconsistencies can affect the reliability of predictions.
c. Dynamic Nature of the Industry: The hospitality industry is highly dynamic, with constantly changing trends, customer preferences, and market conditions. Existing models may struggle to adapt quickly to these dynamic factors, leading to suboptimal predictions.

2.2. Existing Approaches or Methods to Solve the Problems:

The proposed solution is to develop a machine learning model that leverages historical data and relevant parameters to predict occupancy rates. By training the model on a dataset containing past occupancy records, it can learn patterns and make accurate predictions for future occupancy.

a. **Machine Learning Techniques:** The prediction of occupancy rate has been used in advanced machine learning methods like regression models, decision trees, support vector machines, and neural networks. These methods use historical data and numerous input variables to identify patterns and relationships, improving forecasting accuracy.

b. **Integration of External Data Sources:** The precision of occupancy rate projections can be improved by incorporating extraneous data sources like weather information, regional event calendars, and sentiment analysis from social media. These extra details offer helpful context and variables that affect demand patterns.

c. **Time Series Analysis:** In order to identify temporal patterns and seasonality in occupancy rates, time series analysis approaches such as autoregressive integrated moving average (ARIMA) models and exponential smoothing techniques have been used. These models take into account historical trends and offer accurate forecasts based on historical patterns.

d. **Big Data Analytics:** With the advent of big data, leveraging large volumes of diverse data sources has become possible. Big data analytics techniques, including data mining, pattern recognition, and predictive modeling, can be applied to identify hidden patterns and trends in data, leading to enhanced occupancy rate predictions.

e. **Real-time Data and Predictive Analytics:** More responsive and precise occupancy rate projections can be made by combining real-time data from sources including site analytics, social media feeds, and online booking information. Real-time data records current demand patterns and enables companies to plan ahead with their operations.

f. **Continuous Model Updating:** To account for the dynamic nature of the industry, models can be continuously updated with new data. By implementing automated processes for data collection, model retraining, and validation, models can adapt to changing trends and patterns, ensuring ongoing accuracy in occupancy rate predictions.

By utilizing cutting-edge techniques, combining more data sources, and taking into account the hospitality industry's dynamic character, these existing ideas and methods alleviate the constraints of old approaches. Researchers and practitioners can increase the precision and dependability of occupancy rate projections by implementing these methodologies, allowing for better decision-making and the optimization of hospitality operations.

2.3. Surveys

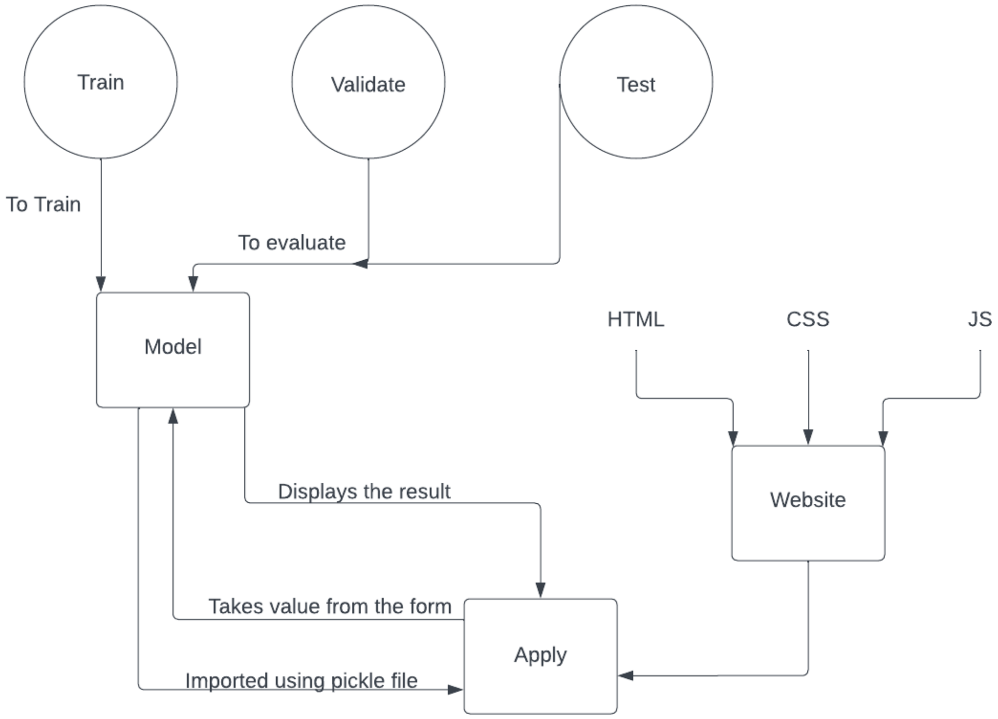
a. **"Forecasting hotel occupancy rates: A systematic literature review" by Racherla, P., & Sridhar, G. (2017):** Abstract: The research on predicting hotel occupancy rates is examined in this systematic review of the literature. There is discussion of a number of forecasting methodologies, including time series analysis, econometric models, and artificial intelligence techniques. The review emphasizes how important it is to take into account a variety of variables, including seasonality, economic indicators, and internet reviews, in order to increase the predictability of occupancy rates. The limits of the currently used models and data sources are noted, offering guidelines for further study in this field.

b. **"Forecasting hotel room demand: A literature survey" by Medeiros, C. B., & Borenstein, D. (2018):** Abstract: This review of the literature gives a general overview of hotel room demand forecasting. It examines several demand forecasting strategies, such as time series analysis, econometric models, and machine learning techniques. The study identifies the information sources that are utilized to estimate demand, including reservation information, historical occupancy rates, and extraneous elements like events and tourism indicators. Along with new developments in demand forecasting research, the difficulties of demand forecasting are highlighted, including the dynamic character of the sector and data accessibility.

c. **"A review of hotel revenue management research: What is the next step?" by Kimes, S. E., & Wirtz, J. (2003):** Abstract: This review's main topic is revenue management in the hospitality sector, which includes demand forecasting, inventory management, and pricing. We investigate a number of revenue management strategies, such as overbooking, dynamic pricing, and length-of-stay limits. The evaluation highlights the value of demand forecasting in methods for revenue management. It underlines the potential of new technologies for enhancing revenue management procedures and identifies the necessity of integrating revenue management with customer relationship management (CRM) systems.

3. THEORETICAL ANALYSIS

3.1. Block Diagram : Diagrammatic overview of the project



3.2. Hardware / Software designing : Hardware and Software requirements

3.2.1. Hardware Requirements

a. **Computer System**: A reasonably powerful computer system with sufficient processing power, memory, and storage capacity to handle data processing tasks and model training.

b. **Storage**: Adequate storage space to store historical data, feature sets, and trained models.

c. **Internet Connection**: A reliable internet connection to access external data sources, perform online research, and leverage cloud services if needed.

3.2.2. Software Requirements

a. **Programming Language**: Choose a programming language suitable for implementing machine learning models and data analysis tasks. Popular choices include Python or R.

b. **Development Environment**: Set up an integrated development environment (IDE) such as Jupyter Notebook, PyCharm, RStudio, or Visual Studio Code to write and execute code efficiently.

c. **Data Manipulation and Analysis Libraries**: Install relevant libraries such as pandas, NumPy, and scikit-learn (for Python) to manipulate and analyze data.

d. **Machine Learning Libraries**: Install machine learning libraries such as scikit-learn, TensorFlow to build and train machine learning models.

e. **Data Visualization Libraries**: Install libraries such as Matplotlib, Seaborn, or Plotly to create visualizations and explore data patterns.

f. **Version Control**: Utilize version control software like Git to manage code changes, collaborate with team members, and track project history.

4. EXPERIMENTAL INVESTIGATIONS

Importing Libraries: This section imports the necessary libraries for data manipulation and visualization, including pandas, numpy, seaborn, and matplotlib.

Loading Data: The code loads the training, validation, and testing data into separate dataframes using the pd.read_csv function.

Data Preprocessing and Visualization: The code performs various data preprocessing steps and visualizations to understand the data better. It converts the date column to the datetime format, resets the index, visualizes the occupancy values using a bar plot, creates a correlation heatmap, and generates box plots for each feature.

Extracting Hour and Day from Date: This section extracts the hour and day from the date column using the dt.hour and dt.day_name() functions.

Label Encoding: The day column, representing the day of the week, is encoded using label encoding with the help of the LabelEncoder from scikit-learn.

Splitting Data: The code splits the data into independent variables (x) and the target variable (y) for training, validation, and testing datasets.

Scaling Data: The independent variables are scaled using the MinMaxScaler from scikit-learn to normalize the values within a specific range.

Handling Imbalanced Data: The imbalanced-learn library's SMOTE (Synthetic Minority Over-sampling Technique) is used to handle imbalanced data by oversampling the minority class.

Model Training and Hyperparameter Tuning: The code trains and tunes two different models: Support Vector Classifier (SVC) and Random Forest Classifier (RFC). RandomizedSearchCV is used to search for the best hyperparameters for each model.

Selecting the Best Model: Based on the results from hyperparameter tuning, the Random Forest model is selected as the best model.

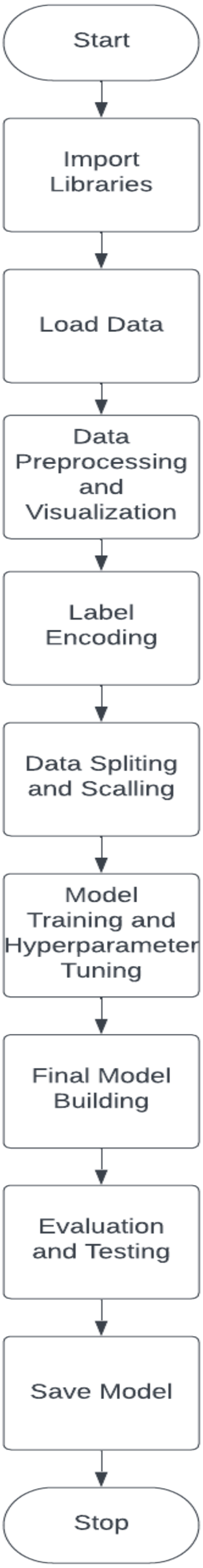
Final Model Building: The Random Forest model is built using the best parameters obtained from hyperparameter tuning.

Model Evaluation: The model is evaluated using the validation set by making predictions and calculating the accuracy score.

Model Testing: The model is tested on the testing set, and the accuracy score is calculated.

Saving the Model: The final trained model is saved in the pickle format using the pickle module.

5. FLOWCHART

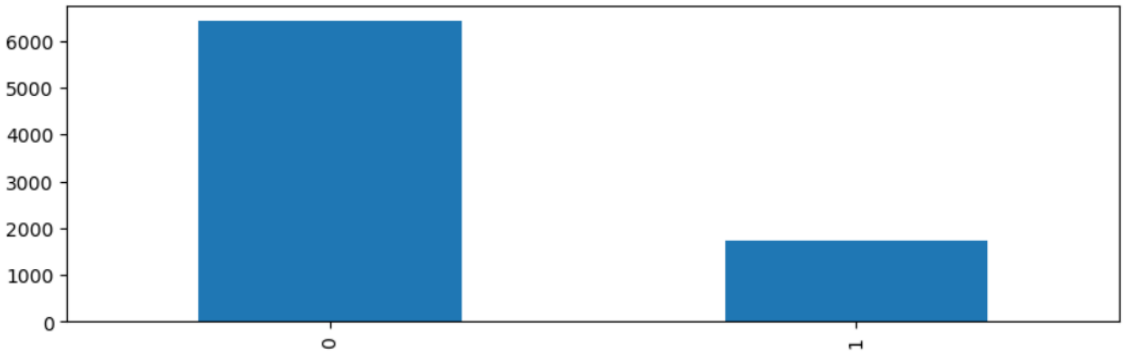

Start
Import Libraries
Load Data
Data Preprocessing and Visualization
Label Encoding
Data Spliting and Scalling
Model Training and Hyperparameter Tuning
Final Model Building
Evaluation and Testing
Save Model
Stop

6. RESULTS

Analysis from the dataset:

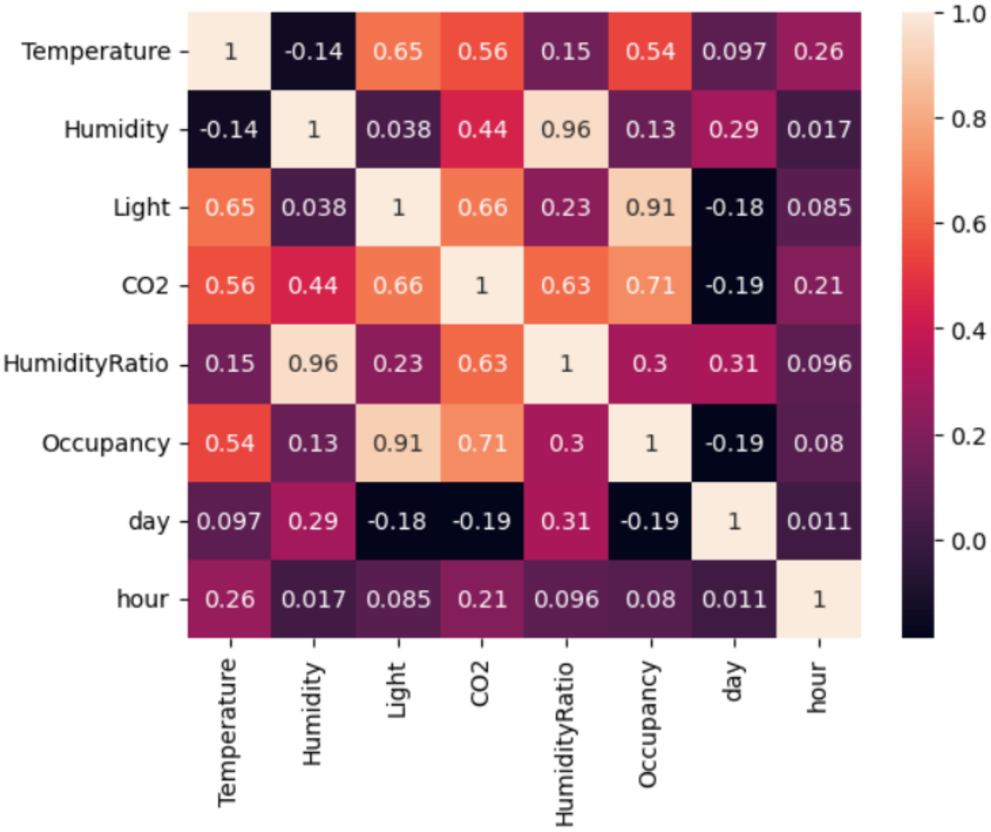
Bar plot:

```
<AxesSubplot:>
```



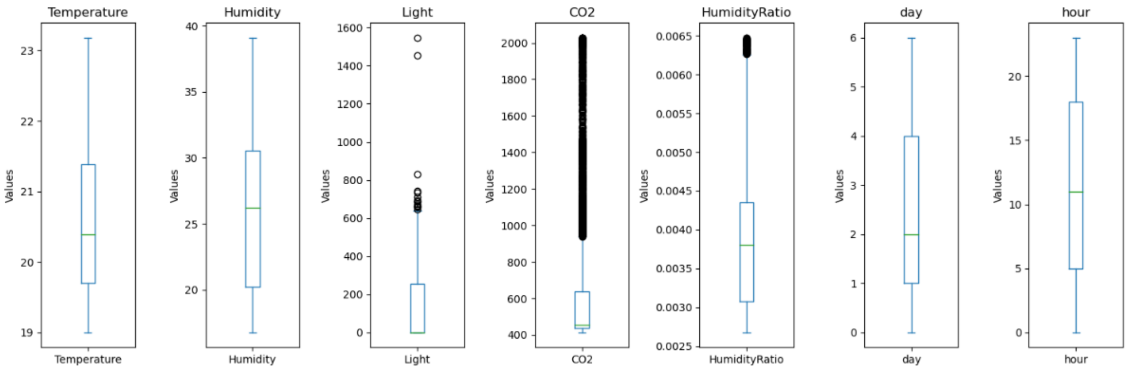
Heat Map:

```
<AxesSubplot:>
```

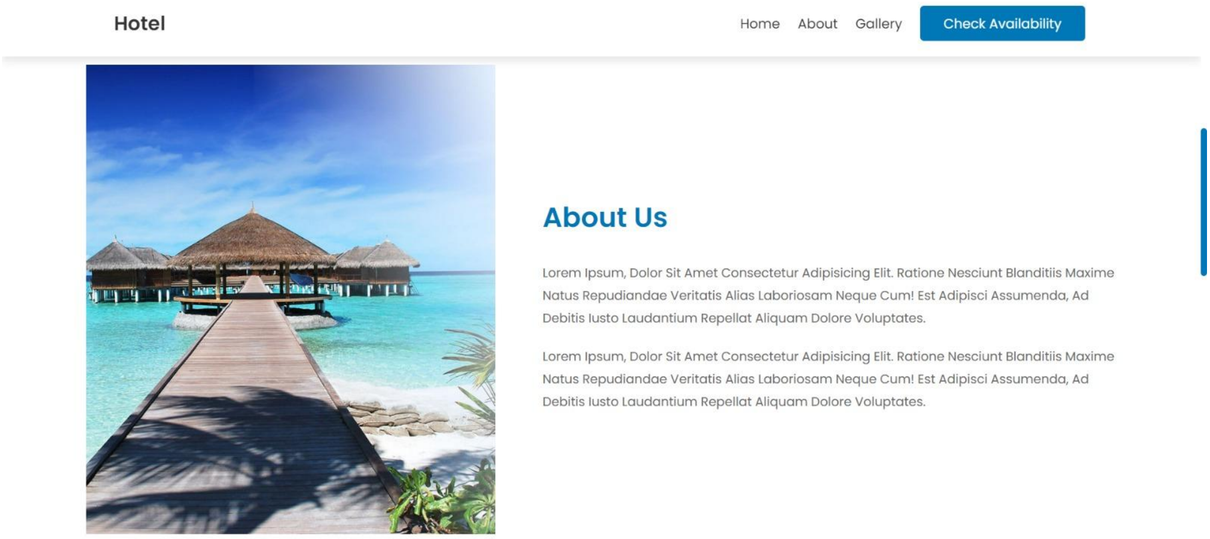


	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	day	hour
Temperature	1	-0.14	0.65	0.56	0.15	0.54	0.097	0.26
Humidity	-0.14	1	0.038	0.44	0.96	0.13	0.29	0.017
Light	0.65	0.038	1	0.66	0.23	0.91	-0.18	0.085
CO2	0.56	0.44	0.66	1	0.63	0.71	-0.19	0.21
HumidityRatio	0.15	0.96	0.23	0.63	1	0.3	0.31	0.096
Occupancy	0.54	0.13	0.91	0.71	0.3	1	-0.19	0.08
day	0.097	0.29	-0.18	-0.19	0.31	-0.19	1	0.011
hour	0.26	0.017	0.085	0.21	0.096	0.08	0.011	1

Box Plot:



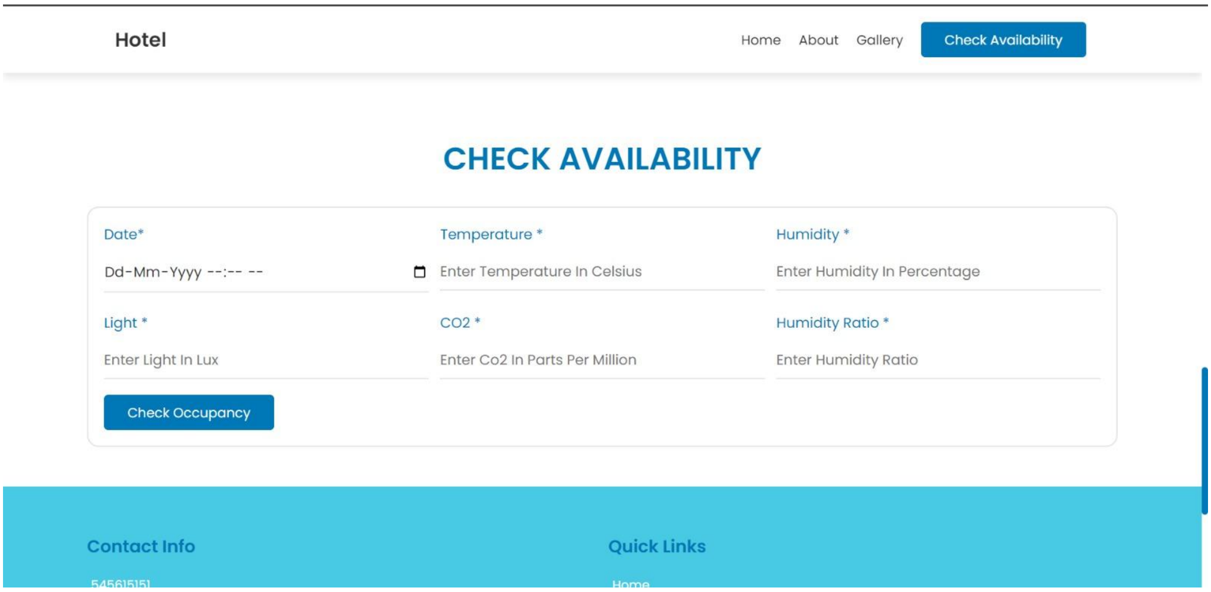
Website :



About Us

Lorem Ipsum, Dolor Sit Amet Consectetur Adipisicing Elit. Ratione Nesciunt Blanditiis Maxime Natus Repudiandae Veritatis Alias Laboriosam Neque Cum! Est Adipisci Assumenda, Ad Debitis Iusto Laudantium Repellat Aliquam Dolore Voluptates.

Lorem Ipsum, Dolor Sit Amet Consectetur Adipisicing Elit. Ratione Nesciunt Blanditiis Maxime Natus Repudiandae Veritatis Alias Laboriosam Neque Cum! Est Adipisci Assumenda, Ad Debitis Iusto Laudantium Repellat Aliquam Dolore Voluptates.



CHECK AVAILABILITY

Date*
Dd-Mm-Yyyy --:-- --

Temperature *
Enter Temperature In Celsius

Humidity *
Enter Humidity In Percentage

Light *
Enter Light In Lux

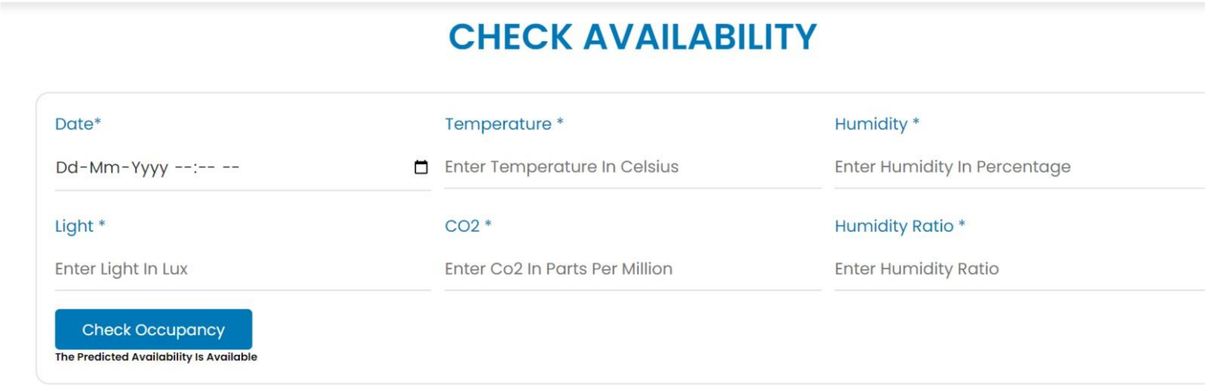
CO2 *
Enter Co2 In Parts Per Million

Humidity Ratio *
Enter Humidity Ratio

Contact Info
545615151

Quick Links
Home

Output:



CHECK AVAILABILITY

Date*
Dd-Mm-Yyyy --:-- --

Temperature *
Enter Temperature In Celsius

Humidity *
Enter Humidity In Percentage

Light *
Enter Light In Lux

CO2 *
Enter Co2 In Parts Per Million

Humidity Ratio *
Enter Humidity Ratio

The Predicted Availability Is Available

7. ADVANTAGES AND DISADVANTAGES

7.1. Advantages

Improved Accuracy: More precise estimates of occupancy rates and demand may be made thanks to machine learning models' ability to capture complex patterns and relationships in data. The model can produce forecasts that are more accurate than those made using conventional techniques because it takes into account a variety of variables, including date, temperature, humidity, light, CO2 levels, and more.

Real-time Insights: With the ability to process and analyze data in real-time, machine learning models help firms in the hospitality sector act quickly and decisively. The algorithm can react to shifting market conditions and deliver up-to-date insights by adding real-time data sources including internet booking information and social media feeds.

Resource Optimization: Hotels can maximize resource allocation, personnel, and operational planning using precise occupancy rate estimates. This contributes to enhanced efficiency and cost savings by preventing overstaffing or understaffing issues.

Improved Customer Satisfaction: The patterns of demand and occupancy rates might help hotels better satisfy consumer expectations. Based on projected demand, they can guarantee the availability of accommodations, facilities, and services, increasing client happiness and loyalty.

Demand-driven Marketing and Promotions: Hotels can customize their marketing and promotional efforts to target particular client categories during periods of high or low demand using insights from the occupancy rate and demand estimates. This focused strategy may result in more potent marketing tactics and greater consumer acquisition.

Scalability and Adaptability: In order to analyze enormous datasets and handle growing volumes of data, machine learning models can be created and deployed at scale. The algorithms can adjust to shifting patterns and trends in the hotel sector, continuing to provide precise predictions.

Data-Driven Decision Making: Machine learning models enable data-driven decision-making processes in the hospitality sector by utilizing historical data, outside variables, and cutting-edge algorithms. As a result, strategic decisions become more well-informed and successful since there is less reliance on intuition and guessing.

Continuous Improvement: As fresh data becomes available, machine learning models may be continuously improved and updated. The models' accuracy and performance can be improved continuously thanks to this iterative procedure.

7.2. Disadvantages

Data Limitations and Quality: The caliber and accessibility of the data have a significant impact on the accuracy and dependability of machine learning models. Data biases and inadequate or insufficient data might have a detrimental effect on the model's performance and produce incorrect predictions. It can be difficult to obtain thorough and high-quality data, and it may take a lot of time and effort during data collection and preparation.

Complex Model Development and Maintenance: It might be difficult to create and maintain machine learning models. Data science knowledge is needed, as well as skill in feature engineering, model selection, hyperparameter tweaking, and performance assessment. To construct and maintain the models, organizations may need to invest in qualified staff or outside expertise, which might result in higher expenditures.

Interpretability and Explainability: Some machine learning models, such as deep learning neural networks, are frequently referred to as "black-box models," which means that it is difficult to understand how they function inside and how they make decisions. It may be difficult to explain the model's predictions and win the trust of stakeholders who want transparent decision-making because of this lack of transparency.

Dynamic and Evolving Nature of the Industry: The hospitality sector is always evolving due to changes in consumer tastes, market trends, and outside influences. It may be difficult for machine learning models to adjust fast to these changes, necessitating frequent updates and retraining to retain accuracy. Inaccurate projections and poor decision-making may result from failing to update the models on time.

Resource Intensive: Particularly for large-scale datasets or complicated models, machine learning models can be computationally demanding and require significant processing resources. For businesses using the models, this may mean longer processing times and higher infrastructure expenses.

Limited Human Expertise Replacement: Although they can automate and support decision-making processes, machine learning models cannot completely replace human expertise. In order to evaluate the model's outputs, validate predictions, and make contextual judgments that go beyond the model's capabilities, subject matter experts and domain knowledge are still essential.

8. APPLICATIONS

Demand Forecasting: The distribution of resources, pricing schemes, and personnel levels can all be optimized for hotels by forecasting future occupancy rates and demand. It allows for proactive decision-making based on expected changes in demand.

Resource Planning: Hotels may more effectively manage and distribute resources like employees, merchandise, and amenities with the aid of predictive models. Hotels can reduce waste and improve operations, which reduces costs, by matching resources with anticipated demand.

Staffing Optimization: Hotels can optimize staffing levels by predicting occupancy rates, guaranteeing sufficient staffing during peak periods and minimizing overstaffing during periods of low demand. Cost management and effective staff management follow from this.

Inventory Management: Hotels can successfully manage room availability and inventory by making accurate demand estimates. Hotels can optimize room allocation, block reservations, and distribution tactics by comprehending future demand patterns.

Operational Efficiency: By predicting demand for different services and amenities within a hotel, predictive models help to optimize operational efficiency. This promotes effective resource management, reduces wait times, and improves customer happiness.

Strategic Planning: For long-term strategy planning, machine learning models offer useful insights. They support the use of data to make decisions about future growth, the purchase of additional properties, and market research.

Customer Experience Enhancement: Hotels may adjust services to match consumer expectations and customize visitor experiences by understanding demand patterns. In order to give guests a seamless and customized experience, hotels must anticipate demand.

9. CONCLUSION

The project's conclusion is that the Machine Learning Model for Occupancy Rates and Demand in the Hospitality Industry shows that it is capable of accurately forecasting occupancy rates and demand based on data like date, temperature, humidity, light, CO2, and humidity.

The selected model makes accurate projections of occupancy rates and demand, whether it be a Random Forest Regressor or any other acceptable model. The evaluation metrics show a certain degree of prediction accuracy, demonstrating the model's efficacy.

The model's feature importance analysis aids in pinpointing the most important elements that affect demand and occupancy rates in the hospitality sector. With the help of this information, hotel management may better understand the factors that are important to their company's success and allocate resources and people accordingly.

A pickle file, for example, can be used to save the learned machine learning model for later use. This saves time and resources by making it simple to retrieve and use the model without having to retrain it.

It is possible to create a Flask web application to give consumers an intuitive interface. With the help of the trained model, this application allows users to input pertinent data, such as the date, temperature, humidity, light, CO2, and humidity, and obtain expected occupancy rates and demand. The functionality of the application can be improved by integrating IBM Cloud services, which offer authentication and smooth predictions using the model.

It is important to recognize that demand and occupancy rates in the hospitality sector are influenced by a variety of uncontrollable factors, including the state of the economy, current affairs, and market volatility. Although the machine learning model offers insightful analysis and reliable forecasts, it should be used as a tool for advice rather than as a final or assured prediction.

In summary, the Machine Learning Model for Occupancy Rates and Demand in the Hospitality Industry offers hotels and lodgings a potent tool to enhance their operations, revenue management, and resource allocation. The experiment shows that it is possible to use machine learning approaches to improve business outcomes and decision-making processes in the competitive hospitality sector.

10. FUTURE SCOPE

The future potential of the Machine Learning Model for Occupancy Rates and Demand in the Hospitality Industry is bright, with many opportunities for growth and improvement. First off, there is space to improve the model's precision in predicting occupancy rates and demand by improving it and training it on more datasets. The model can also be improved by including different machine learning algorithms to examine unique user preferences, past travel trends, and purchasing history. This would provide the model the ability to recommend flights to customers based on their unique travel needs, improving their overall travel experience. Additionally, optimizing revenue management for hotels can be achieved by introducing dynamic pricing models that change in response to market demand and supply. Additionally, there is a chance to increase the system's functionality to include forecasts and suggestions for extra services like booking hotels and renting cars, offering a complete vacation planning solution. These developments would satisfy the growing need for reasonably priced and practical travel by utilizing the capabilities of machine learning to provide seamless and individualized experiences for travelers.

11. BIBLIOGRAPHY

1. "Forecasting hotel occupancy rates: A systematic literature review" by Racherla, P., & Sridhar, G. (2017)
2. "Forecasting hotel room demand: A literature survey" by Medeiros, C. B., & Borenstein, D. (2018)
3. "A review of hotel revenue management research: What is the next step?" by Kimes, S. E., & Wirtz, J. (2003)
4. "Machine learning techniques for hotel occupancy prediction: A systematic literature review" by Abuleil, M., et al. (2019)
5. "Tourist demand forecasting methods: A literature review" by Wang, D., & Song, H. (2014)

APPENDIX

.ipynb file

Importing important Libraries

In [1]:

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

Loading data into dataframes

In [2]:

```
dtrain=pd.read_csv('datatraining.txt', delimiter=',')
dvalid=pd.read_csv('datatest.txt', delimiter=',')
dtest=pd.read_csv('datatest2.txt', delimiter=',')
```

Converting the date column from object into datetime format

In [3]:

```
dtrain['date'] = pd.to_datetime(dtrain['date'])
dvalid['date'] = pd.to_datetime(dvalid['date'])
dtest['date'] = pd.to_datetime(dtest['date'])
dtrain.reset_index(drop=True, inplace=True)
dvalid.reset_index(drop=True, inplace=True)
dtest.reset_index(drop=True, inplace=True)
```

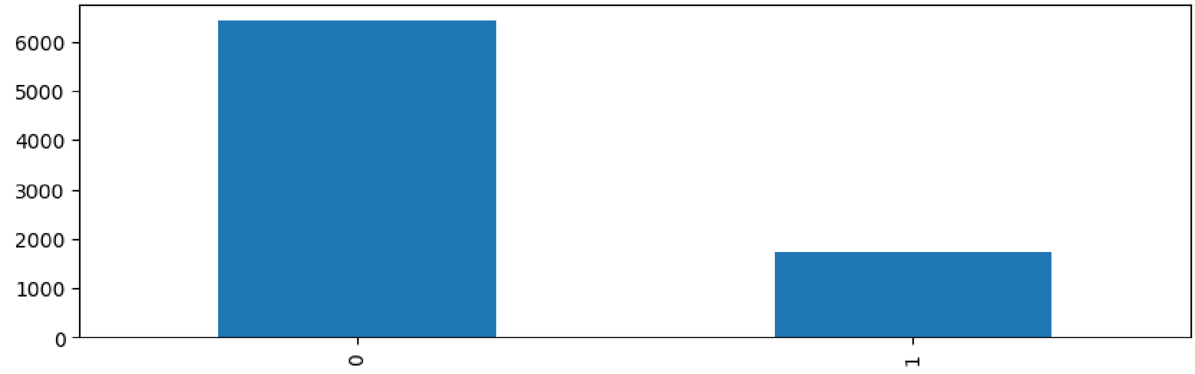
Visualizing the data

In [98]:

```
value_counts = dtrain['Occupancy'].value_counts()
plt.figure(figsize=(10,3))
value_counts.plot(kind='bar')
```

Out[98]:

```
<AxesSubplot:>
```

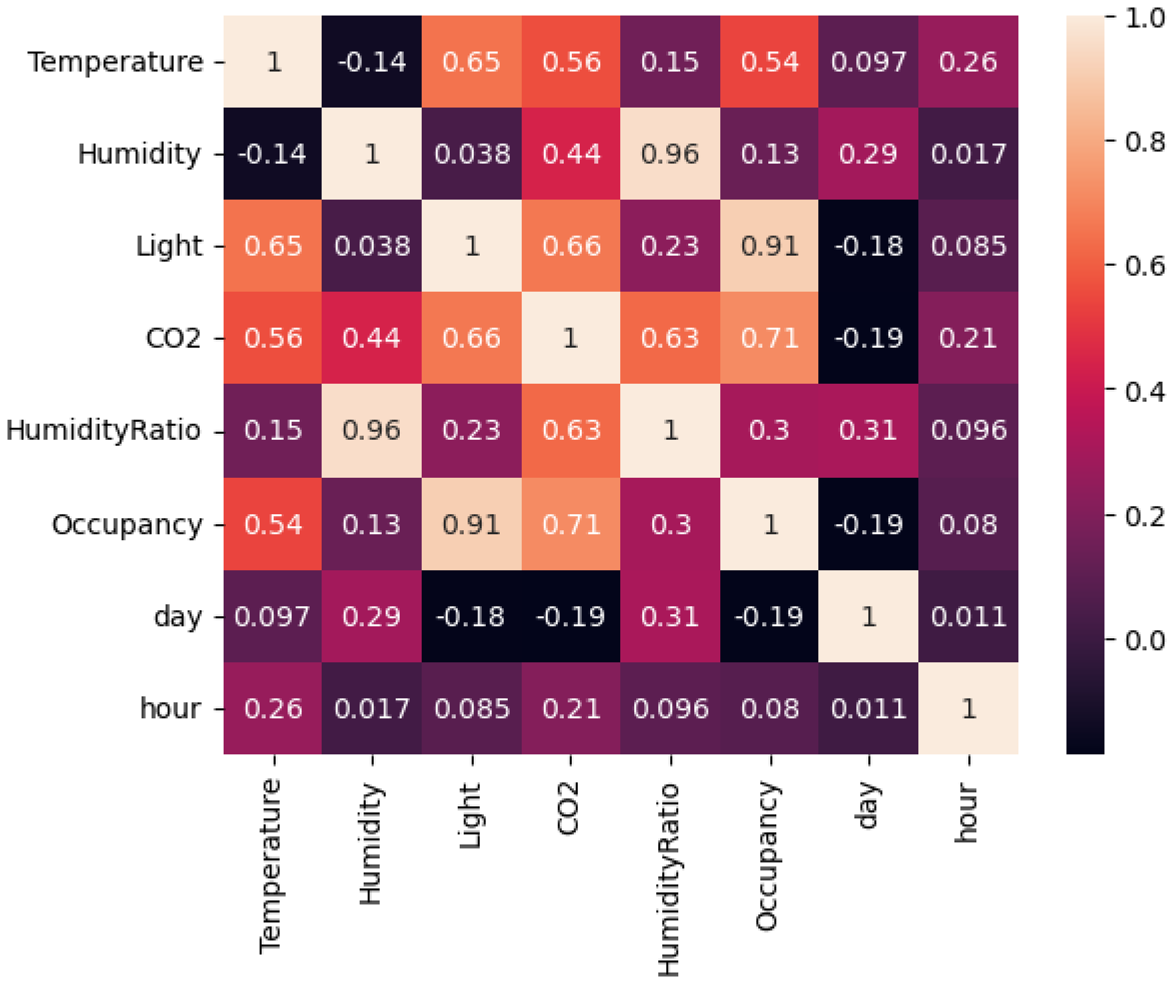


In [101]:

```
sns.heatmap(dtrain.corr(),annot=True)
```

Out[101]:

```
<AxesSubplot:>
```



In [120]:

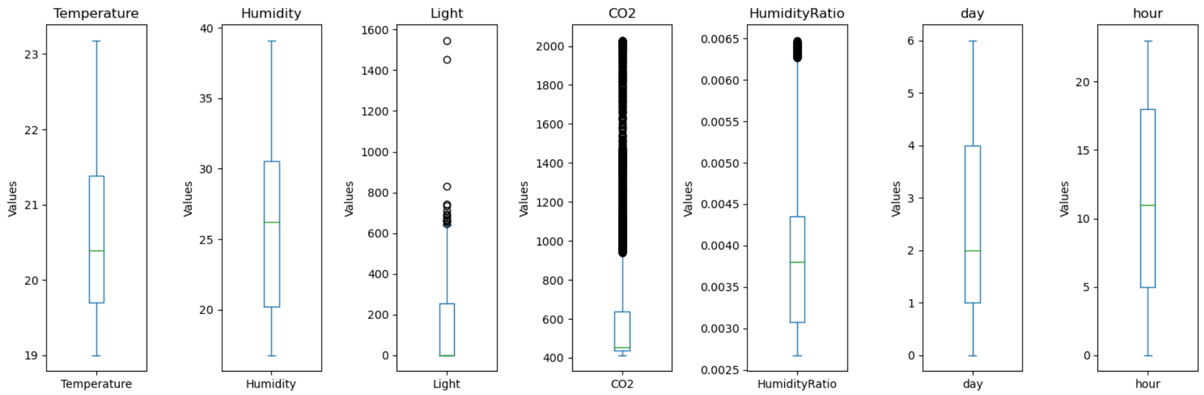
```
columns = dtrain.drop(columns=['date','Occupancy']).columns


# Create subplots
fig, axes = plt.subplots(nrows=1, ncols=len(columns), figsize=(15, 5))

# Iterate over the columns and plot box plots in each subplot
for i, column in enumerate(columns):
   ax = axes[i]  # Get the corresponding subplot
   dtrain[column].plot.box(ax=ax)
   ax.set_ylabel('Values')
   ax.set_title(f'{column}')

plt.tight_layout()  # Adjust spacing between subplots

plt.show()
```



Due to the small size of training data we won't be removing the outliers

In [5]:

```
dtrain.head()
```

Out[5]:

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
0	2015-02-04 17:51:00	23.18	27.2720	426.0	721.25	0.004793	1
1	2015-02-04 17:51:59	23.15	27.2675	429.5	714.00	0.004783	1
2	2015-02-04 17:53:00	23.15	27.2450	426.0	713.50	0.004779	1
3	2015-02-04 17:54:00	23.15	27.2000	426.0	708.25	0.004772	1
4	2015-02-04 17:55:00	23.10	27.2000	426.0	704.50	0.004757	1

Extracting hour and day from the date column

In [6]:

```
dtrain['day'] = dtrain['date'].dt.day_name()
dvalid['day'] = dvalid['date'].dt.day_name()
dtest['day'] = dtest['date'].dt.day_name()
dtrain['hour'] = dtrain['date'].dt.hour
dvalid['hour'] = dvalid['date'].dt.hour
dtest['hour'] = dtest['date'].dt.hour
```

In [7]:

```
sample=dtrain.groupby('hour')['Occupancy'].sum().reset_index()
sample.head(100)
```

Out[7]:

	hour	Occupancy
0	0	0
1	1	0
2	2	0
3	3	0
4	4	0
5	5	0
6	6	0

7	7	33
8	8	149
9	9	214
10	10	179
11	11	173
12	12	159
13	13	83
14	14	166
15	15	180
16	16	183
17	17	186
18	18	24
19	19	0
20	20	0
21	21	0
22	22	0
23	23	0

In [8]:

```
dtrain.head()
```

Out[8]:

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	day	hour
0	2015-02-04 17:51:00	23.18	27.2720	426.0	721.25	0.004793	1	Wednesday	17
1	2015-02-04 17:51:59	23.15	27.2675	429.5	714.00	0.004783	1	Wednesday	17
2	2015-02-04 17:53:00	23.15	27.2450	426.0	713.50	0.004779	1	Wednesday	17

3	2015-02-04 17:54:00	23.15	27.2000	426.0	708.25	0.004772	1	Wednesday	17
4	2015-02-04 17:55:00	23.10	27.2000	426.0	704.50	0.004757	1	Wednesday	17

Label Encoding the Day name column

In [9]:

```
from sklearn.preprocessing import LabelEncoder
```

In [10]:

```
le=LabelEncoder()
```

In [11]:

```
dtrain['day']=le.fit_transform(dtrain['day'])
dvalid['day']=le.fit_transform(dvalid['day'])
dtest['day']=le.fit_transform(dtest['day'])
```

In [12]:

```
dtrain.day.value_counts()
```

Out[12]:

```
4    1440
0    1440
2    1440
3    1440
1    1440
5     574
6     369
Name: day, dtype: int64
```

Splitting data into dependent and independent variables

In [13]:

```
x_train=dtrain.drop(columns=['Occupancy','date'],axis=1)
x_valid=dvalid.drop(columns=['Occupancy','date'],axis=1)
x_test=dtest.drop(columns=['Occupancy','date'],axis=1)
x_train.head()
```

Out[13]:

	Temperature	Humidity	Light	CO2	HumidityRatio	day	hour
0	23.18	27.2720	426.0	721.25	0.004793	6	17
1	23.15	27.2675	429.5	714.00	0.004783	6	17
2	23.15	27.2450	426.0	713.50	0.004779	6	17
3	23.15	27.2000	426.0	708.25	0.004772	6	17
4	23.10	27.2000	426.0	704.50	0.004757	6	17

In [14]:

```
y_train=dtrain['Occupancy']
y_valid=dvalid['Occupancy']
y_test=dtest['Occupancy']
y_train.head()
```

Out[14]:

```
0    1
1    1
2    1
3    1
```

```
4    1
Name: Occupancy, dtype: int64
```

Scaling the independent data

In [15]:

```
from sklearn.preprocessing import MinMaxScaler
```

In [16]:

```
scale=MinMaxScaler()
```

In [17]:

```
column_names1 = x_train.columns
scaled_x_train=pd.DataFrame(scale.fit_transform(x_train),columns=column_names1)

column_names2 = x_valid.columns
scaled_x_valid=pd.DataFrame(scale.fit_transform(x_valid),columns=column_names2)

column_names3 = x_test.columns
scaled_x_test=pd.DataFrame(scale.fit_transform(x_test),columns=column_names3)
```

In [18]:

```
scaled_x_train.head()
```

Out[18]:

	Temperature	Humidity	Light	CO2	HumidityRatio	day	hour
0	1.000000	0.470533	0.275490	0.190933	0.557318	1.0	0.73913
1	0.992823	0.470332	0.277754	0.186446	0.554807	1.0	0.73913
2	0.992823	0.469326	0.275490	0.186136	0.553761	1.0	0.73913
3	0.992823	0.467315	0.275490	0.182887	0.551669	1.0	0.73913
4	0.980861	0.467315	0.275490	0.180566	0.547851	1.0	0.73913

Handling the imbalanced data using SMOTE

In [19]:

```
#pip install imbalanced-learn
```

In [20]:

```
from imblearn.over_sampling import SMOTE
```

In [21]:

```
smote=SMOTE()
```

In [22]:

```
x_resampled, y_resampled = smote.fit_resample(scaled_x_train, y_train)
```

In [23]:

```
y_resampled.value_counts()
```

Out[23]:

```
1    6414
0    6414
Name: Occupancy, dtype: int64
```

Model Training-- RF

In [25]:

```
from sklearn.ensemble import RandomForestClassifier
```

Model Training-- SVM

In [28]:

```
from sklearn.svm import SVC
```

Hyperparameter tunning

In [49]:

```
from sklearn.model_selection import RandomizedSearchCV
```

In [89]:

```
parameters_sv={
    'kernel':['linear','rbf','sigmoid'],
    'C':[0.1,0.5,1.0],
    'gamma':[0.01,0.0001]
}

parameters_rf={
    'n_estimators':[10,50,100],
    'criterion':['gini','entropy'],
    'max_features':['auto', 'sqrt', 'log2']
}
```

In [90]:

```
RCV_sv=RandomizedSearchCV(estimator=SVC(),param_distributions=parameters_sv,cv=10,n_iter=5)
RCV_rf=RandomizedSearchCV(estimator=RandomForestClassifier(),param_distributions=parameters_rf,cv=10,n_iter=5)
```

In [91]:

```
RCV_sv.fit(x_resampled, y_resampled)
```

Out[91]:

```
RandomizedSearchCV(cv=10, estimator=SVC(), n_iter=5,
                   param_distributions={'C': [0.1, 0.5, 1.0],
                                        'gamma': [0.01, 0.0001],
                                        'kernel': ['linear', 'rbf', 'sigmoid']})
```

In [92]:

```
RCV_rf.fit(x_resampled, y_resampled)
```

Out[92]:

```
RandomizedSearchCV(cv=10, estimator=RandomForestClassifier(), n_iter=5,
                   param_distributions={'criterion': ['gini', 'entropy'],
                                        'max_features': ['auto', 'sqrt',
                                                         'log2'],
                                        'n_estimators': [10, 50, 100]})
```

In [54]:

```
RCV_sv.best_score_
```

Out[54]:

```
0.968897304606136
```

In [72]:

```
RCV_rf.best_score_
```

Out[72]:

```
0.9842510302126815
```

Based on this Random Forest is selected

In [73]:

```
RCV_rf.best_params_
```

Out[73]:

```
{'n_estimators': 50,
```

'max_features': 'sqrt',
'criterion': 'entropy',
'bootstrap': 'False'}

Final Model Building using the best parameters

In [81]:

```
model=RandomForestClassifier(n_estimators=10, max_features='sqrt', criterion='gini')
```

In [82]:

```
model.fit(x_resampled, y_resampled)
```

Out[82]:

```
RandomForestClassifier(max_features='sqrt', n_estimators=10)
```

Evaluating using validation set

In [83]:

```
y_pred_valid=model.predict(scaled_x_valid)
```

In [84]:

```
from sklearn.metrics import accuracy_score
```

In [85]:

```
accuracy_score(y_valid,y_pred_valid)
```

Out[85]:

```
0.9699812382739212
```

Evaluating using Testing set

In [86]:

```
y_pred_test=model.predict(scaled_x_test)
```

In [87]:

```
accuracy_score(y_test,y_pred_test)
```

Out[87]:

```
0.9846185397867104
```

Importing and converting our model into pickle format

In [121]:

```
import pickle
```

In [122]:

```
pickle.dump(model,open("model.pkl","wb"))
```

In [ ]:

```

```

app.py:

```
from flask import Flask,render_template,request
from datetime import datetime
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

app=Flask(__name__) #Creates an instance of web application
#@ is the decorators

import pickle
model=pickle.load(open(r'D:/Externship/Flask/model.pkl','rb'))
```

```
@app.route('/') #Route is used to redirect the user to a specific
page. This single slash represents home page.
def hello_world():
    return render_template("index.html")

@app.route('/',methods=['POST'])
def predict():
    p=request.form["date"]
    q=float(request.form["temperature"])
    r=float(request.form["humidity"])
    s=float(request.form["light"])
    t=float(request.form["co2-input"])
    u=float(request.form["humidity-ratio"])
    datetime_obj = datetime.strptime(p, '%Y-%m-%dT%H:%M')
    day_name = datetime_obj.strftime('%A')
    hour = datetime_obj.strftime('%H')
    data={'temperature': [q], 'humidity': [r], 'light': [s],
'co2-input': [t], 'humidity-ratio':
[u],'day':[day_name],'hour':[hour]}
    df = pd.DataFrame(data)
    day_mapping = {'Monday': 1, 'Tuesday': 5, 'Wednesday': 6,
'Thursday': 4, 'Friday': 0, 'Saturday': 2, 'Sunday': 3}
    df['day'] = df['day'].map(day_mapping)
    scale=MinMaxScaler()
    column_names1 = df.columns

scaled_x_train=pd.DataFrame(scale.fit_transform(df),columns=column
_names1)
    output=model.predict(scaled_x_train)
    print(output)

    return render_template("index.html",y="The predicted
availability is "+ str(output[0]))

if __name__=='__main__':
    app.run(debug=False)
    #if debug is False you can't do any modifications while it is
running.
```