Report On

Hotel Data Analysis

Submitted in partial fulfillment of the requirements of the Course project in Semester V of Third Year Computer Engineering

by
Vipul Bhoir (Roll No. 07)
Mrudul Chaudhari (Roll No. 12)
Abhinav Desai (Roll No. 14)

Supervisor
Prof. Swati Varma

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

(2023-24)

Vidyavardhini's College of Engineering & Technology
Department of Computer Engineering

CERTIFICATE

This is to certify that the project entitled “Hotel Data Analysis” is a bonafide work of "Vipul Bhoir (Roll No. 07), Mrudul Chaudhari (Roll No. 12), Abhinav Desai (Roll No. 14)" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester V of Third Year Computer Engineering.

Prof. Swati Varma **Supervisor**	Dr. Megha Trivedi Head of Department

Contents **Pg. No**

1. ABSTRACT

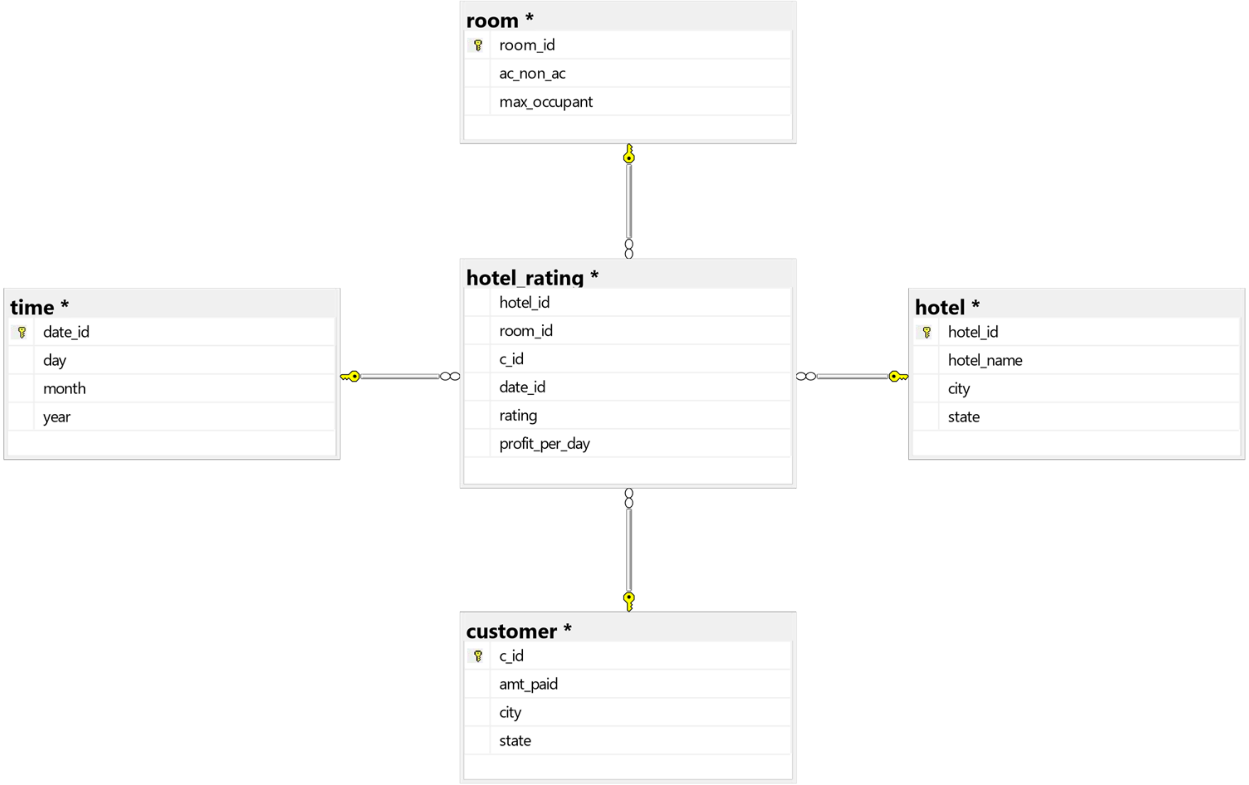
The hospitality industry relies heavily on data-driven decision-making to enhance guest experiences and improve operational efficiency. This project aims to conduct a comprehensive data analysis for a hotel, with the goal of optimizing its performance. The analysis involves examining various aspects of hotel operations, including occupancy rates, revenue management, customer reviews, and operational expenses. This project leverages data analysis, statistical modeling, and machine learning techniques to provide valuable insights that can be used to make informed decisions in the highly competitive hospitality industry. By optimizing operations and enhancing the guest experience, the hotel can gain a competitive edge and achieve better financial performance.

2. PROBLEM STATEMENT

The hospitality industry is highly competitive, and hotels strive to provide exceptional service while optimizing their operations to maximize revenue. To achieve this, hotels need to make data-driven decisions that improve their efficiency and guest satisfaction. This project aims to address this challenge by leveraging data analysis to provide actionable insights for a hotel's management. Develop a predictive model that forecasts room demand patterns, allowing the hotel to optimize pricing strategies, staff allocation, and inventory management. Analyze historical pricing data, competitor rates, and market trends to recommend dynamic pricing strategies that maximize revenue without compromising occupancy. Analyze historical pricing data, competitor rates, and market trends to recommend dynamic pricing strategies that maximize revenue without compromising occupancy.hotel data analysis project aims to provide actionable insights that empower the hotel to make informed decisions, optimize its operations, and ultimately improve guest satisfaction and financial performance.

3. BLOCK DIGRAM:

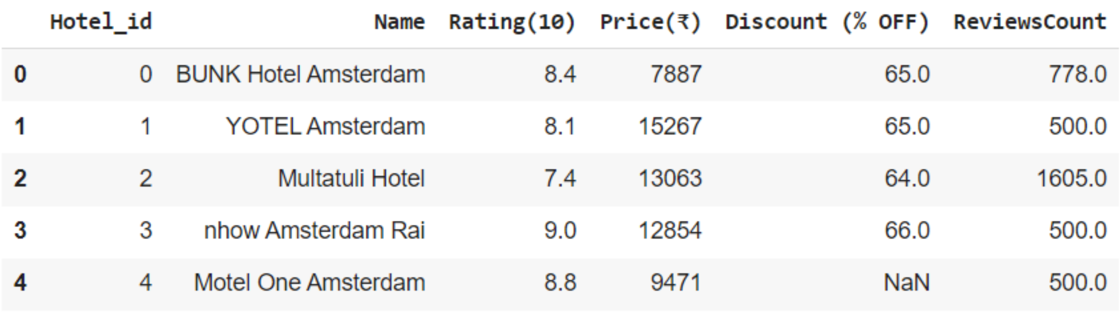
Star Schema:



4. CODE & OUTPUT:

4.1 Data Visualization

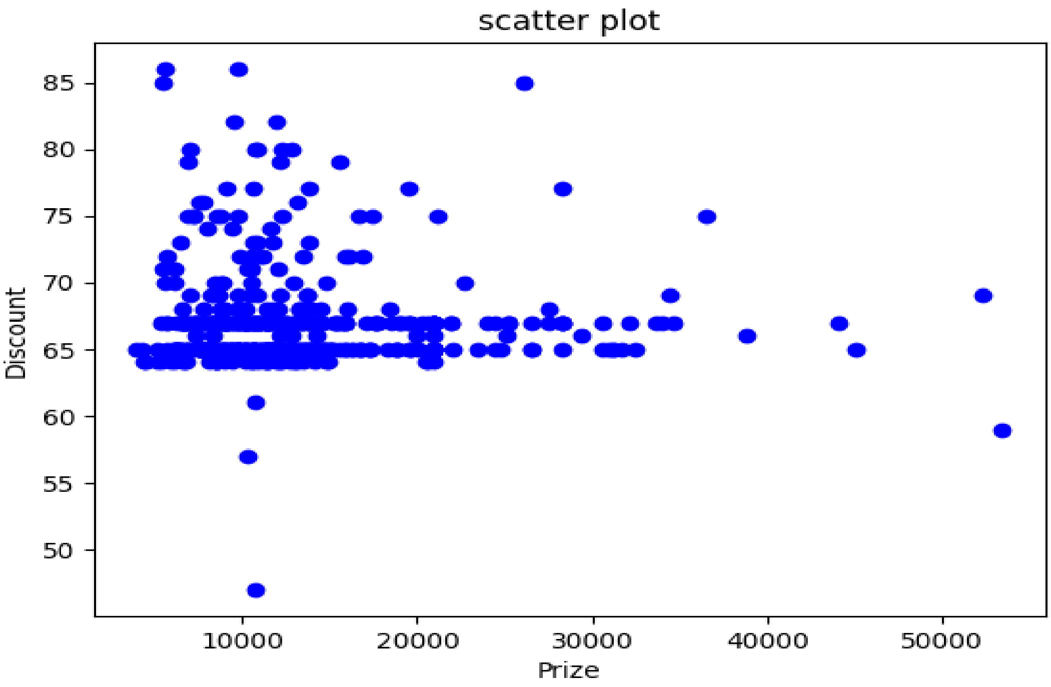
```
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
dataset=pd.read_excel("dwm1.xlsx")
```



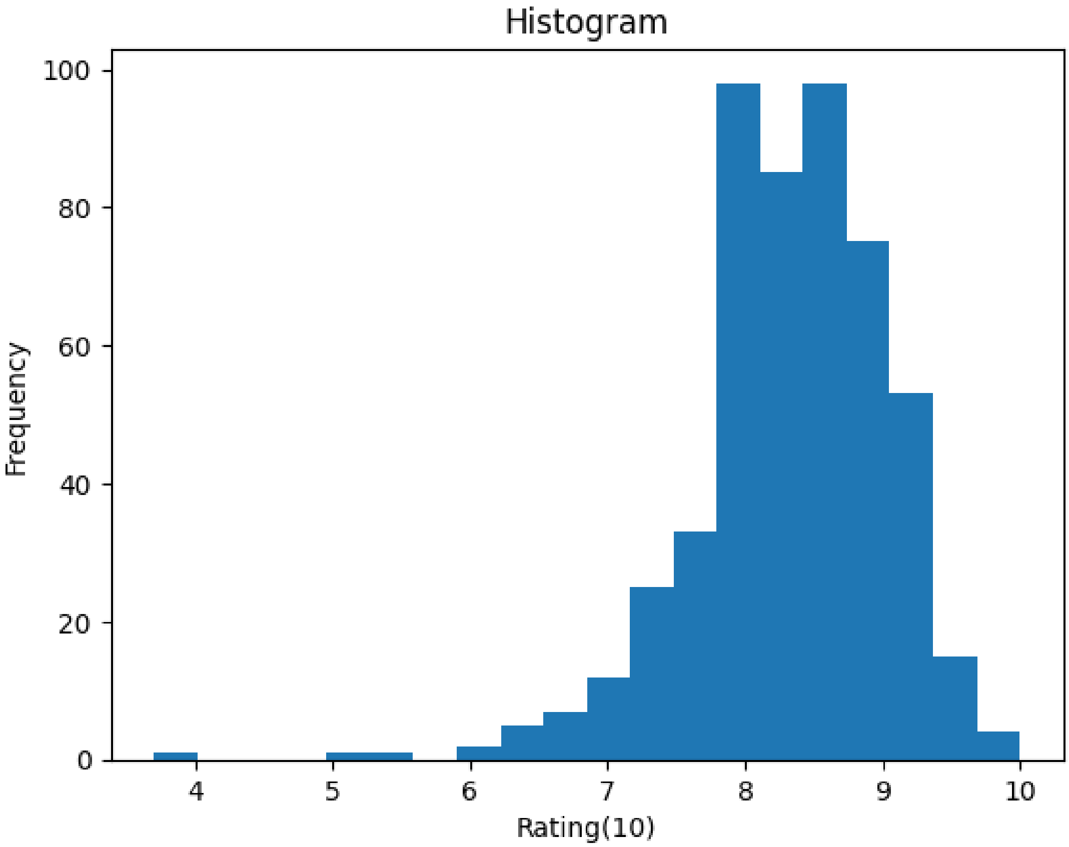
	Hotel_id	Name	Rating(10)	Price(₹)	Discount (% OFF)	ReviewsCount
0	0	BUNK Hotel Amsterdam	8.4	7887	65.0	778.0
1	1	YOTEL Amsterdam	8.1	15267	65.0	500.0
2	2	Multatuli Hotel	7.4	13063	64.0	1605.0
3	3	nhow Amsterdam Rai	9.0	12854	66.0	500.0
4	4	Motel One Amsterdam	8.8	9471	NaN	500.0

```
dataset.shape
(525, 6)
```

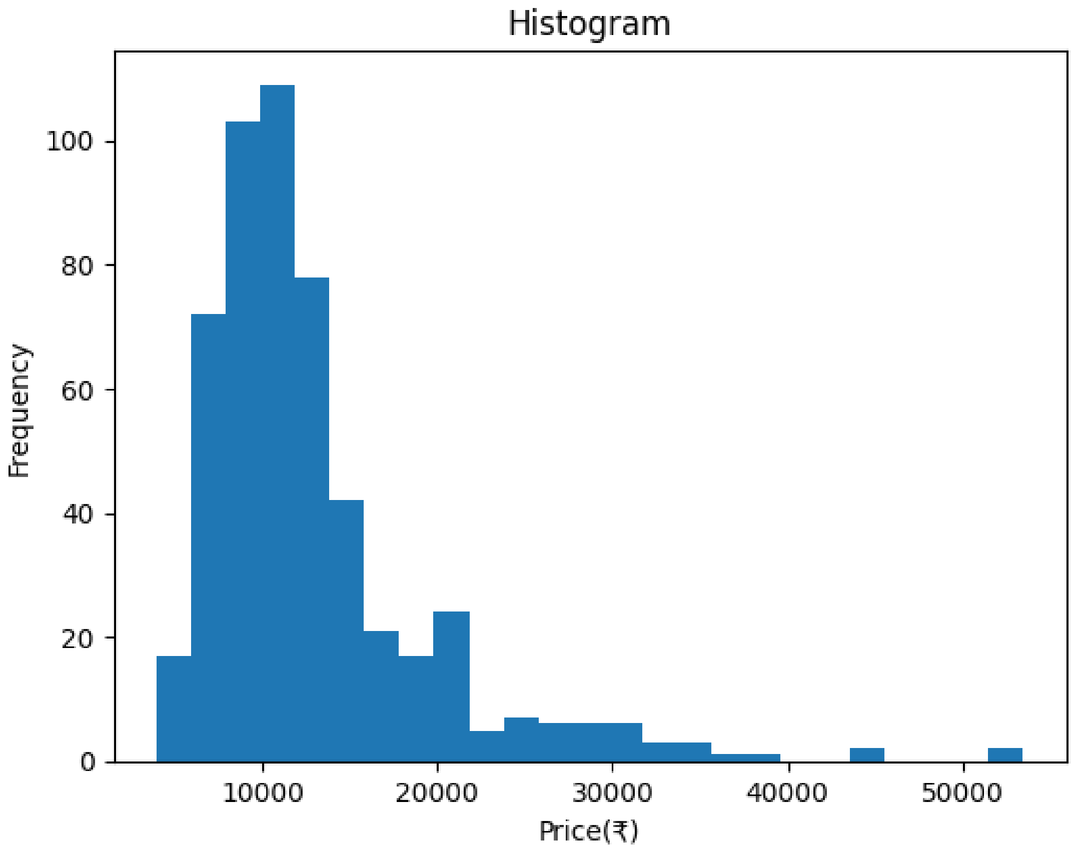
```
plt.scatter(dataset['Price(₹)'],dataset['Discount (% OFF)'],
color='blue')
plt.title("scatter plot")
plt.xlabel("Prize")
plt.ylabel("Discount")
plt.show()
```



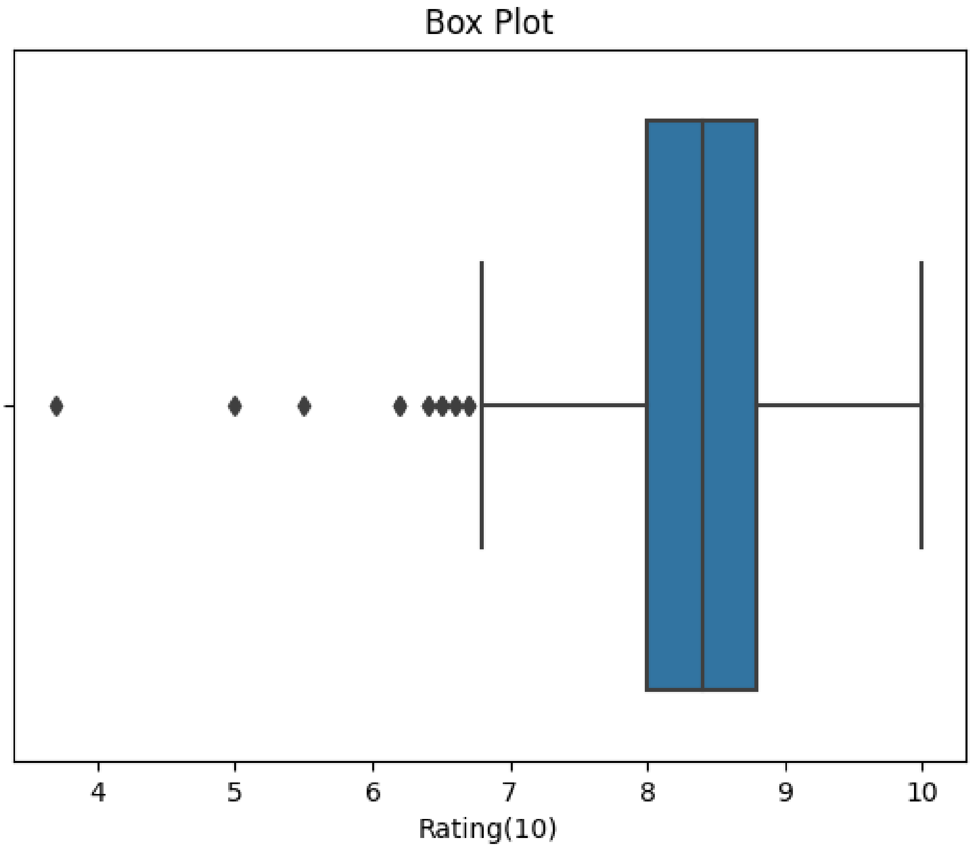
```
plt.hist(dataset['Rating(10)'], bins=20)
plt.title("Histogram")
plt.xlabel("Rating(10)")
plt.ylabel("Frequency")
plt.show()
```



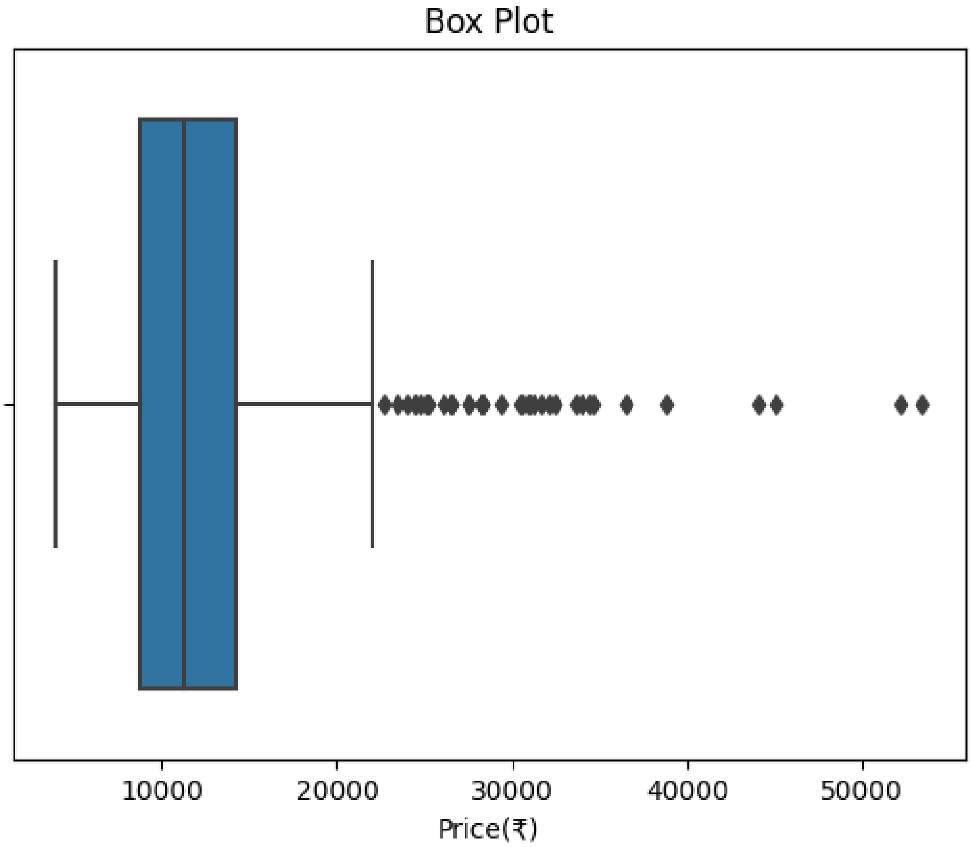
```
plt.hist(dataset['Price(₹)'], bins=25)
plt.title("Histogram")
plt.xlabel("Price(₹)")
plt.ylabel("Frequency")
plt.show()
```



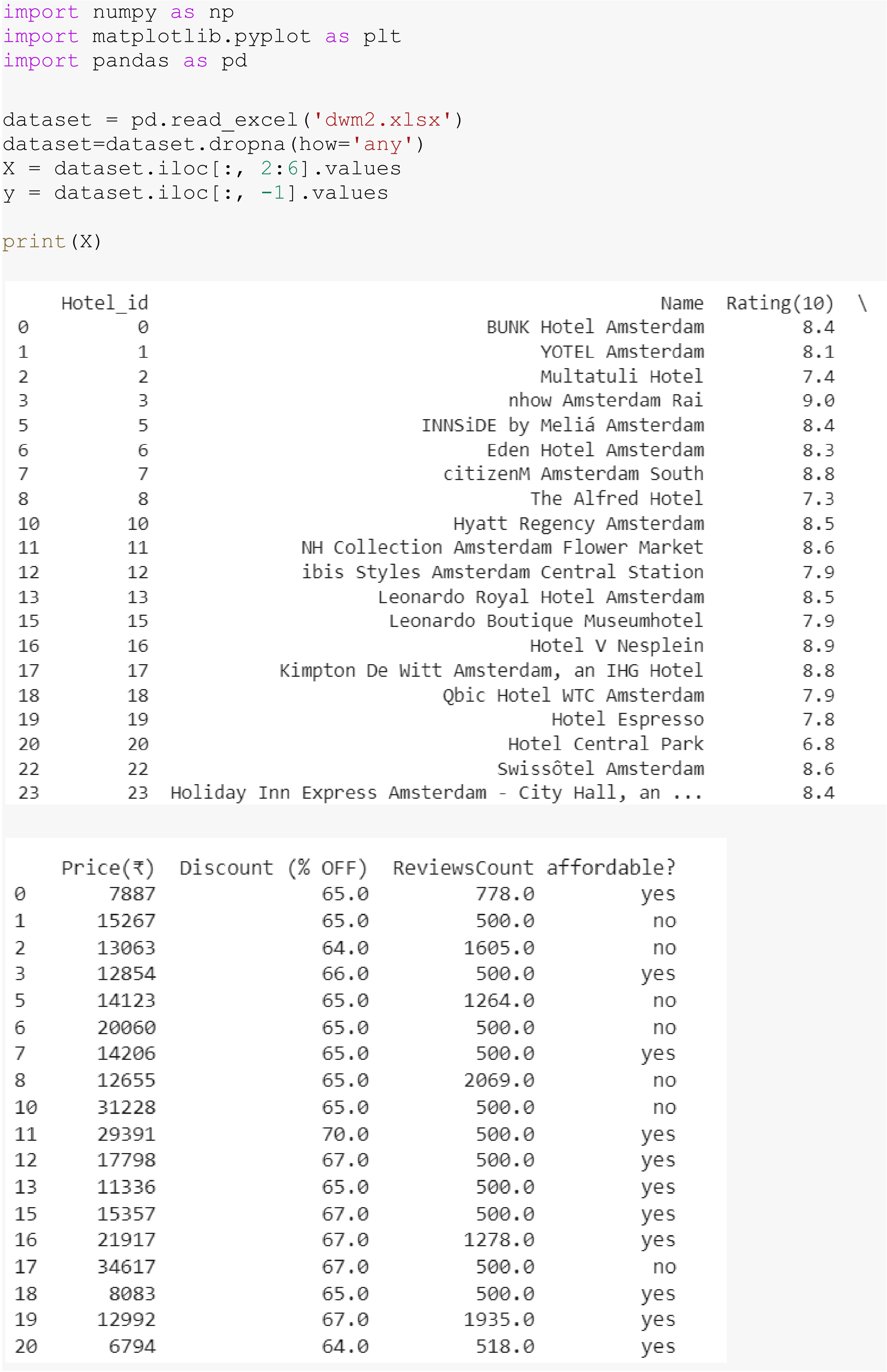
```
sb.boxplot(x="Rating(10)", data=dataset)
plt.title("Box Plot")
```



```
sb.boxplot(x="Price(₹)", data=dataset)
plt.title("Box Plot")
```



4.2 Data Preprocessing



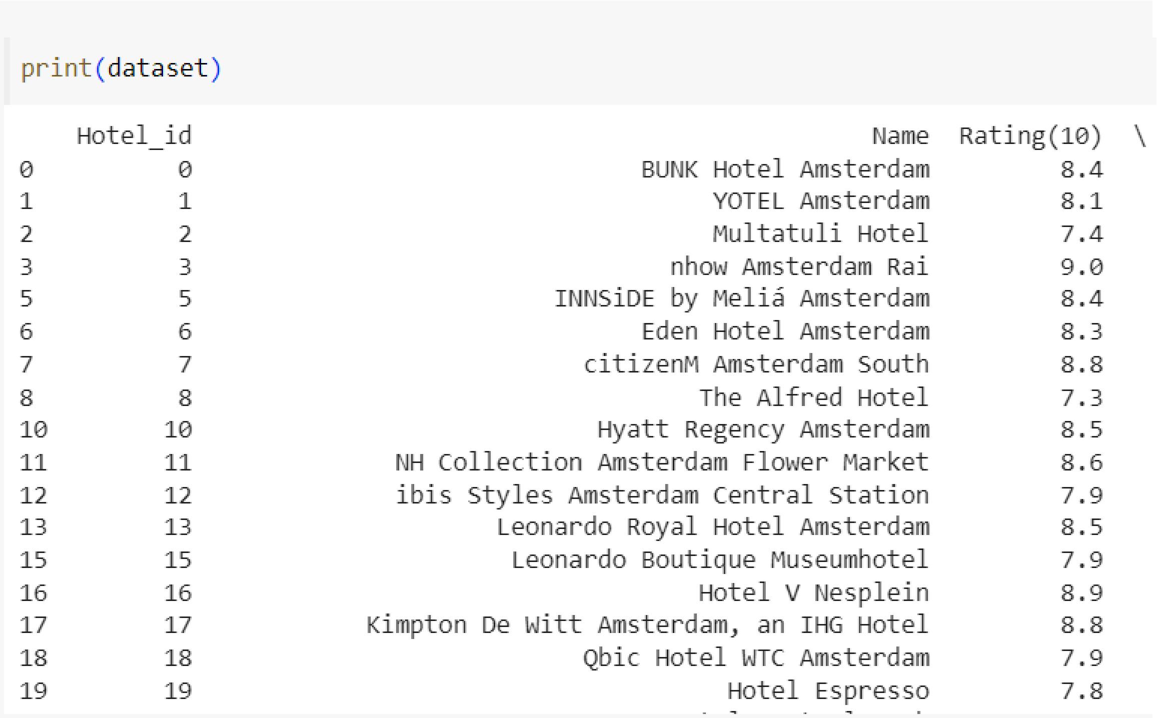
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd


dataset = pd.read_excel('dwm2.xlsx')
dataset=dataset.dropna(how='any')
X = dataset.iloc[:, 2:6].values
y = dataset.iloc[:, -1].values

print(X)
```

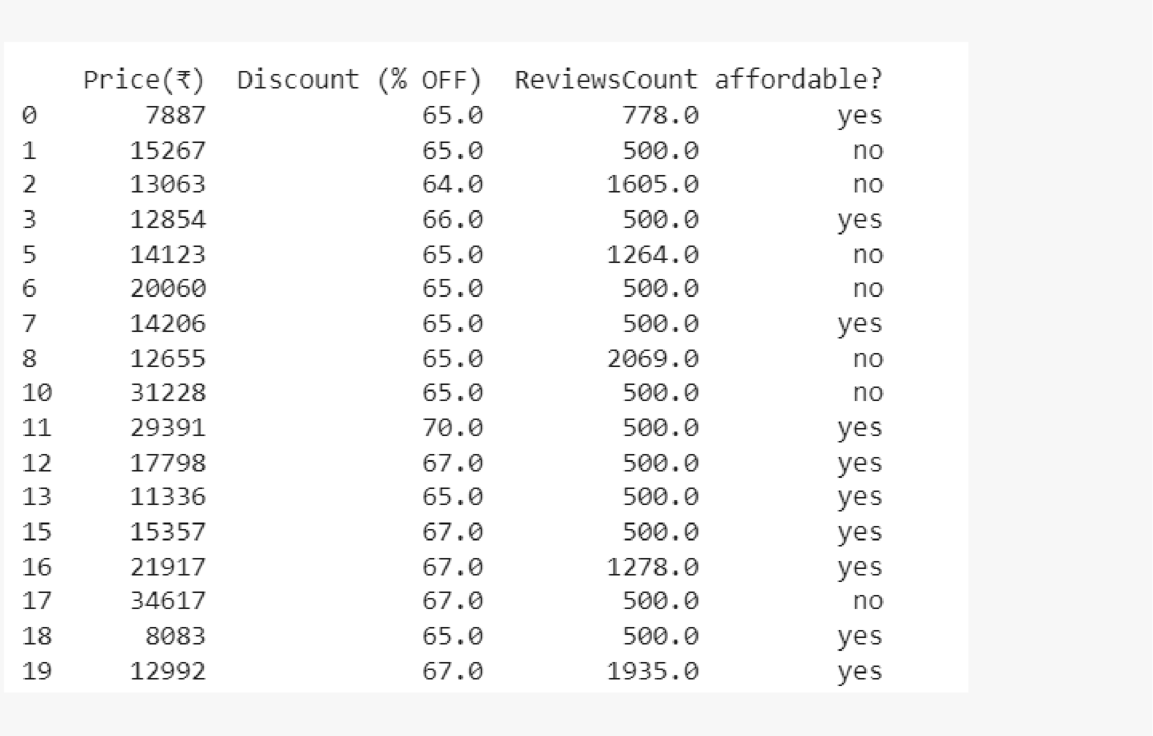
```
    Hotel_id                                              Name  Rating(10)  \
0          0                              BUNK Hotel Amsterdam         8.4   
1          1                                   YOTEL Amsterdam         8.1   
2          2                                   Multatuli Hotel         7.4   
3          3                                nhow Amsterdam Rai         9.0   
5          5                        INNSiDE by Meliá Amsterdam         8.4   
6          6                              Eden Hotel Amsterdam         8.3   
7          7                          citizenM Amsterdam South         8.8   
8          8                                  The Alfred Hotel         7.3   
10        10                           Hyatt Regency Amsterdam         8.5   
11        11             NH Collection Amsterdam Flower Market         8.6   
12        12             ibis Styles Amsterdam Central Station         7.9   
13        13                    Leonardo Royal Hotel Amsterdam         8.5   
15        15                     Leonardo Boutique Museumhotel         7.9   
16        16                                  Hotel V Nesplein         8.9   
17        17           Kimpton De Witt Amsterdam, an IHG Hotel         8.8   
18        18                          Qbic Hotel WTC Amsterdam         7.9   
19        19                                    Hotel Espresso         7.8   
20        20                                Hotel Central Park         6.8   
22        22                               Swissôtel Amsterdam         8.6   
23        23  Holiday Inn Express Amsterdam - City Hall, an ...         8.4   
```

```
    Price(₹)  Discount (% OFF)  ReviewsCount affordable?
0       7887              65.0         778.0        yes
1      15267              65.0         500.0         no
2      13063              64.0        1605.0         no
3      12854              66.0         500.0        yes
5      14123              65.0        1264.0         no
6      20060              65.0         500.0         no
7      14206              65.0         500.0        yes
8      12655              65.0        2069.0         no
10     31228              65.0         500.0         no
11     29391              70.0         500.0        yes
12     17798              67.0         500.0        yes
13     11336              65.0         500.0        yes
15     15357              67.0         500.0        yes
16     21917              67.0        1278.0        yes
17     34617              67.0         500.0         no
18      8083              65.0         500.0        yes
19     12992              67.0        1935.0        yes
20      6794              64.0         518.0        yes
```

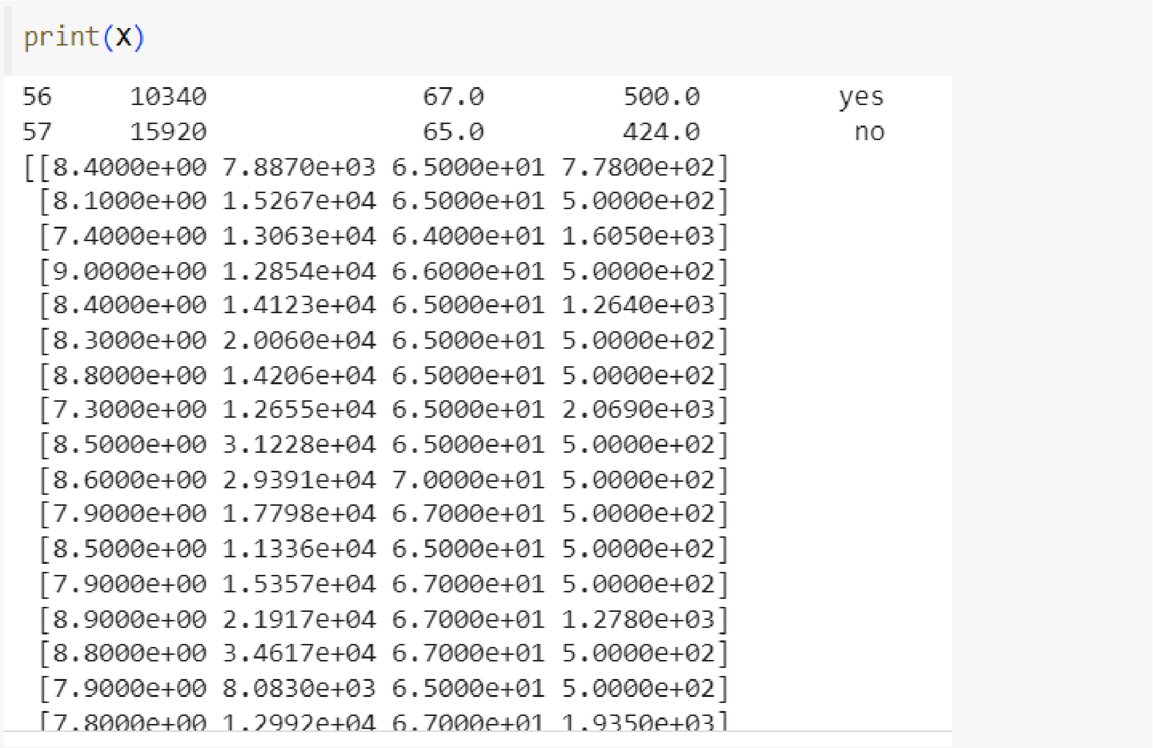


```
print(dataset)
```

```
    Hotel_id                                               Name  Rating(10)  \
0          0                               BUNK Hotel Amsterdam         8.4
1          1                                    YOTEL Amsterdam         8.1
2          2                                    Multatuli Hotel         7.4
3          3                                 nhow Amsterdam Rai         9.0
5          5                         INNSiDE by Meliá Amsterdam         8.4
6          6                               Eden Hotel Amsterdam         8.3
7          7                           citizenM Amsterdam South         8.8
8          8                                   The Alfred Hotel         7.3
10        10                            Hyatt Regency Amsterdam         8.5
11        11              NH Collection Amsterdam Flower Market         8.6
12        12              ibis Styles Amsterdam Central Station         7.9
13        13                     Leonardo Royal Hotel Amsterdam         8.5
15        15                      Leonardo Boutique Museumhotel         7.9
16        16                                   Hotel V Nesplein         8.9
17        17            Kimpton De Witt Amsterdam, an IHG Hotel         8.8
18        18                           Qbic Hotel WTC Amsterdam         7.9
19        19                                     Hotel Espresso         7.8
```

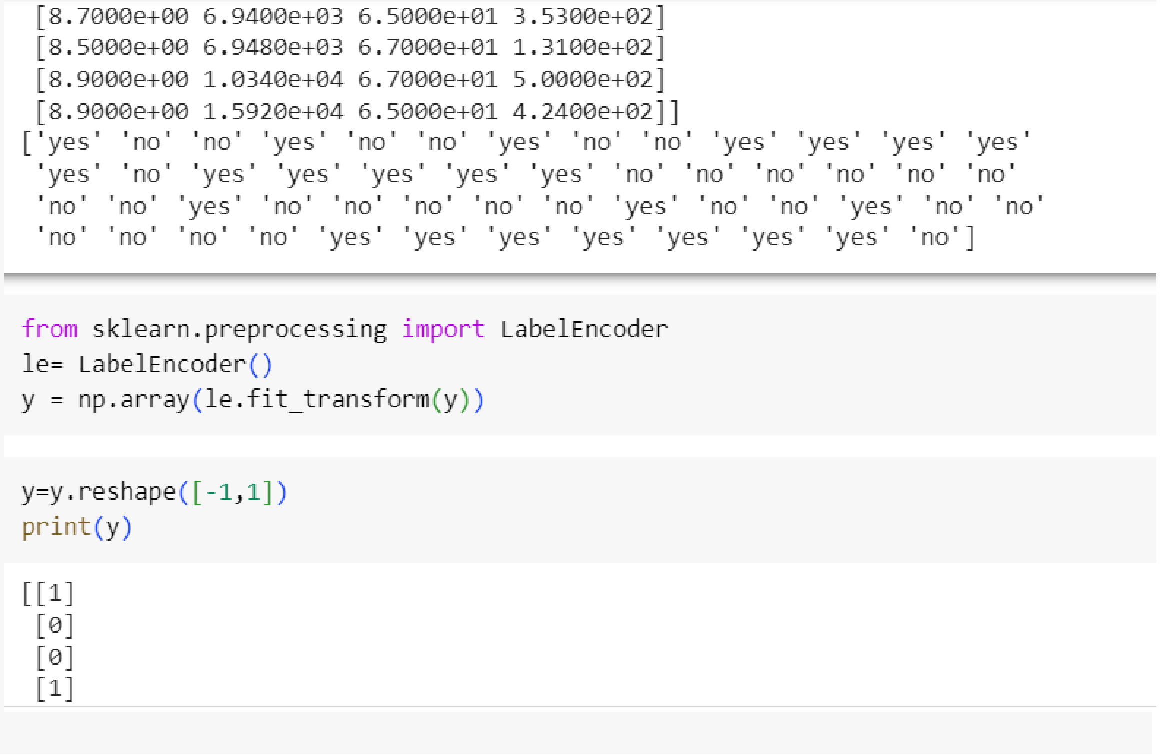


```
    Price(₹)  Discount (% OFF)  ReviewsCount affordable?
0       7887              65.0         778.0         yes
1      15267              65.0         500.0          no
2      13063              64.0        1605.0          no
3      12854              66.0         500.0         yes
5      14123              65.0        1264.0          no
6      20060              65.0         500.0          no
7      14206              65.0         500.0         yes
8      12655              65.0        2069.0          no
10     31228              65.0         500.0          no
11     29391              70.0         500.0         yes
12     17798              67.0         500.0         yes
13     11336              65.0         500.0         yes
15     15357              67.0         500.0         yes
16     21917              67.0        1278.0         yes
17     34617              67.0         500.0          no
18      8083              65.0         500.0         yes
19     12992              67.0        1935.0         yes
```



```
print(X)
```

```
56      10340                67.0          500.0          yes
57      15920                65.0          424.0           no
[[8.4000e+00 7.8870e+03 6.5000e+01 7.7800e+02]
 [8.1000e+00 1.5267e+04 6.5000e+01 5.0000e+02]
 [7.4000e+00 1.3063e+04 6.4000e+01 1.6050e+03]
 [9.0000e+00 1.2854e+04 6.6000e+01 5.0000e+02]
 [8.4000e+00 1.4123e+04 6.5000e+01 1.2640e+03]
 [8.3000e+00 2.0060e+04 6.5000e+01 5.0000e+02]
 [8.8000e+00 1.4206e+04 6.5000e+01 5.0000e+02]
 [7.3000e+00 1.2655e+04 6.5000e+01 2.0690e+03]
 [8.5000e+00 3.1228e+04 6.5000e+01 5.0000e+02]
 [8.6000e+00 2.9391e+04 7.0000e+01 5.0000e+02]
 [7.9000e+00 1.7798e+04 6.7000e+01 5.0000e+02]
 [8.5000e+00 1.1336e+04 6.5000e+01 5.0000e+02]
 [7.9000e+00 1.5357e+04 6.7000e+01 5.0000e+02]
 [8.9000e+00 2.1917e+04 6.7000e+01 1.2780e+03]
 [8.8000e+00 3.4617e+04 6.7000e+01 5.0000e+02]
 [7.9000e+00 8.0830e+03 6.5000e+01 5.0000e+02]
 [7.8000e+00 1.2992e+04 6.7000e+01 1.9350e+03]
```

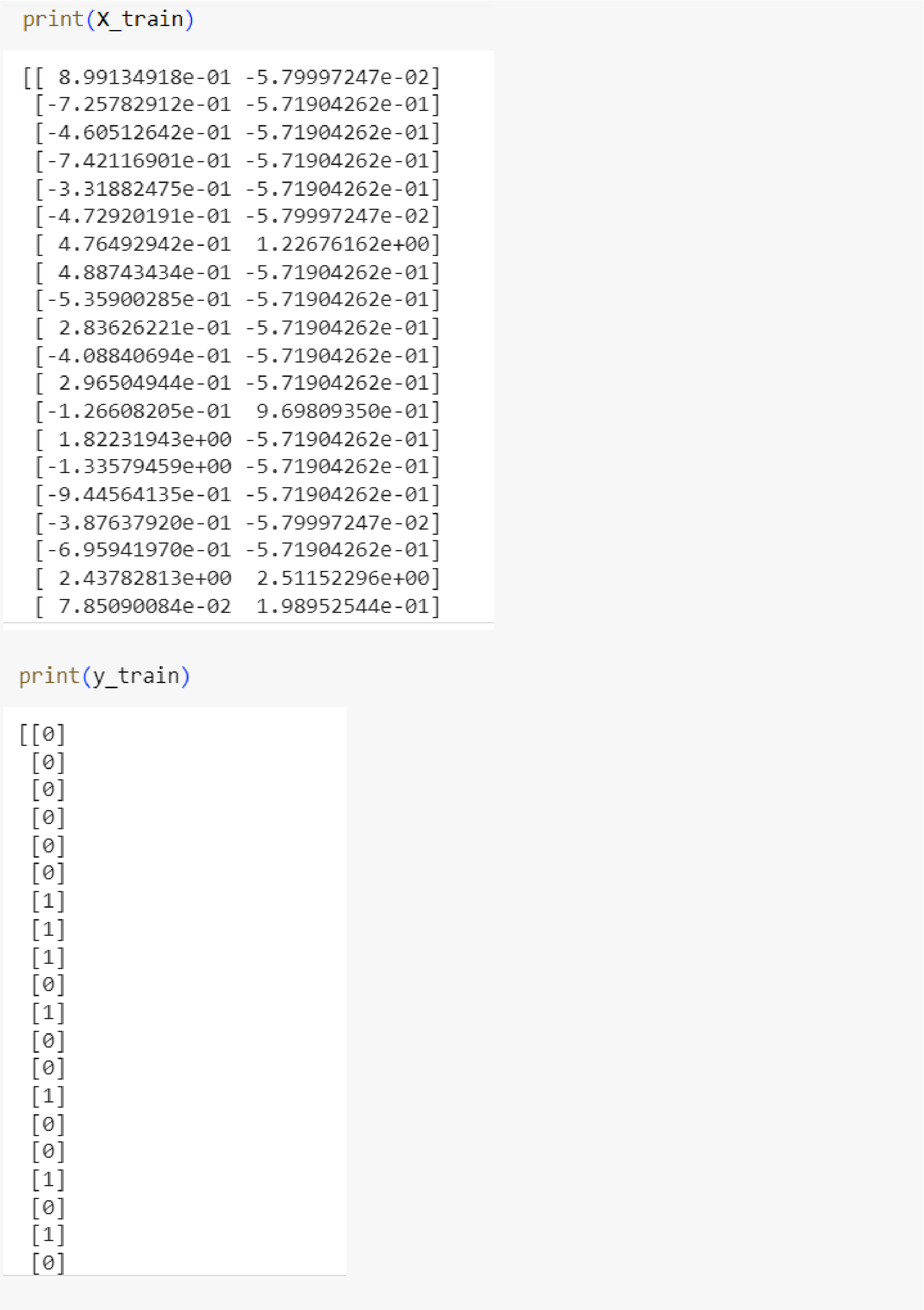


```
 [8.7000e+00 6.9400e+03 6.5000e+01 3.5300e+02]
 [8.5000e+00 6.9480e+03 6.7000e+01 1.3100e+02]
 [8.9000e+00 1.0340e+04 6.7000e+01 5.0000e+02]
 [8.9000e+00 1.5920e+04 6.5000e+01 4.2400e+02]]
['yes' 'no' 'no' 'yes' 'no' 'no' 'yes' 'no' 'no' 'yes' 'yes' 'yes' 'yes'
 'yes' 'no' 'yes' 'yes' 'yes' 'yes' 'yes' 'no' 'no' 'no' 'no' 'no' 'no'
 'no' 'no' 'yes' 'no' 'no' 'no' 'no' 'no' 'yes' 'no' 'no' 'yes' 'no' 'no'
 'no' 'no' 'no' 'no' 'yes' 'yes' 'yes' 'yes' 'yes' 'yes' 'yes' 'no']
```

```
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
y = np.array(le.fit_transform(y))
```

```
y=y.reshape([-1,1])
print(y)
```

```
[[1]
 [0]
 [0]
 [1]
```

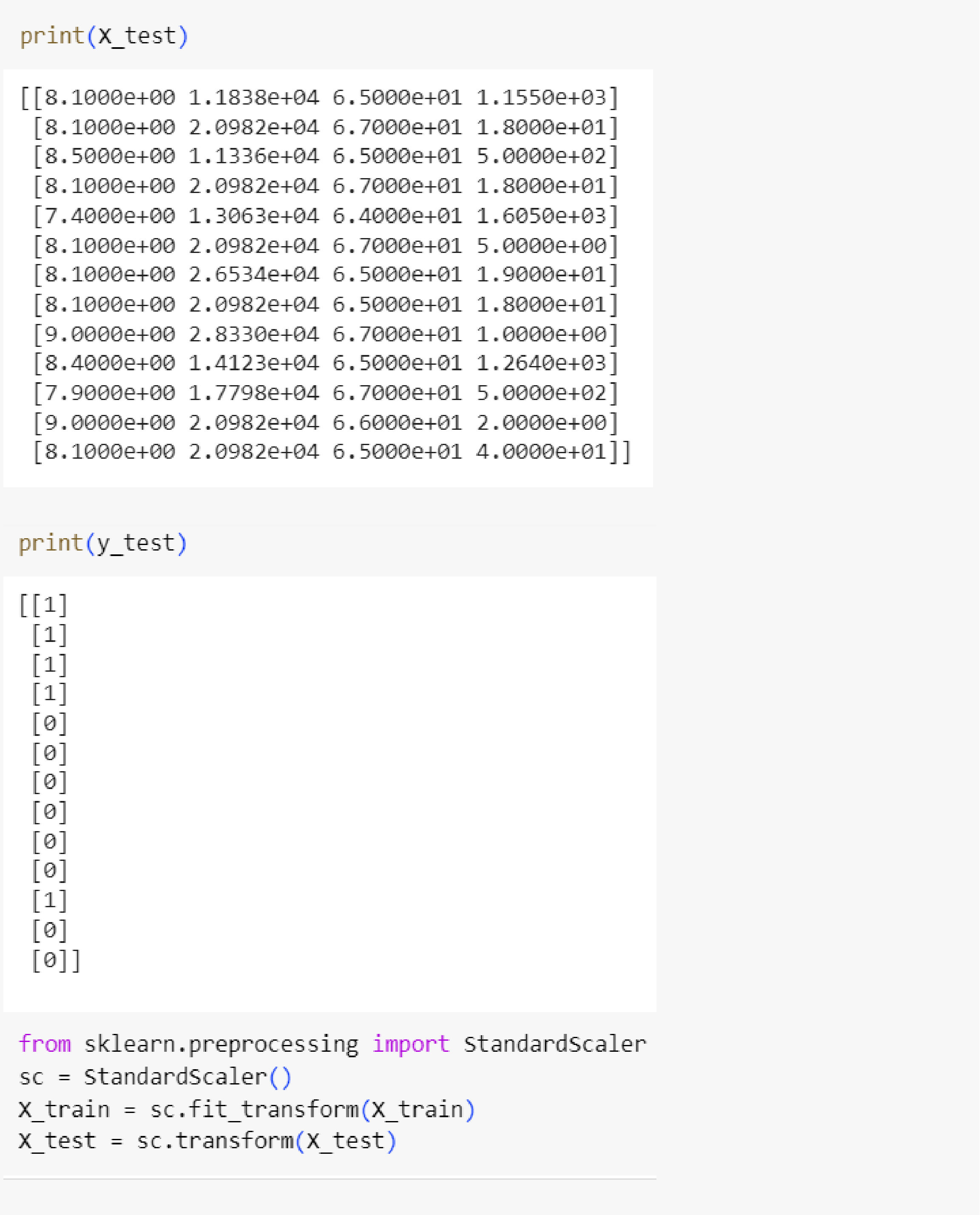


```
print(X_train)
```

```
[[ 8.99134918e-01 -5.79997247e-02]
 [-7.25782912e-01 -5.71904262e-01]
 [-4.60512642e-01 -5.71904262e-01]
 [-7.42116901e-01 -5.71904262e-01]
 [-3.31882475e-01 -5.71904262e-01]
 [-4.72920191e-01 -5.79997247e-02]
 [ 4.76492942e-01  1.22676162e+00]
 [ 4.88743434e-01 -5.71904262e-01]
 [-5.35900285e-01 -5.71904262e-01]
 [ 2.83626221e-01 -5.71904262e-01]
 [-4.08840694e-01 -5.71904262e-01]
 [ 2.96504944e-01 -5.71904262e-01]
 [-1.26608205e-01  9.69809350e-01]
 [ 1.82231943e+00 -5.71904262e-01]
 [-1.33579459e+00 -5.71904262e-01]
 [-9.44564135e-01 -5.71904262e-01]
 [-3.87637920e-01 -5.79997247e-02]
 [-6.95941970e-01 -5.71904262e-01]
 [ 2.43782813e+00  2.51152296e+00]
 [ 7.85090084e-02  1.98952544e-01]
```

```
print(y_train)
```

```
[[0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [1]
 [1]
 [1]
 [0]
 [1]
 [0]
 [0]
 [1]
 [0]
 [0]
 [1]
 [0]
 [1]
 [0]
```



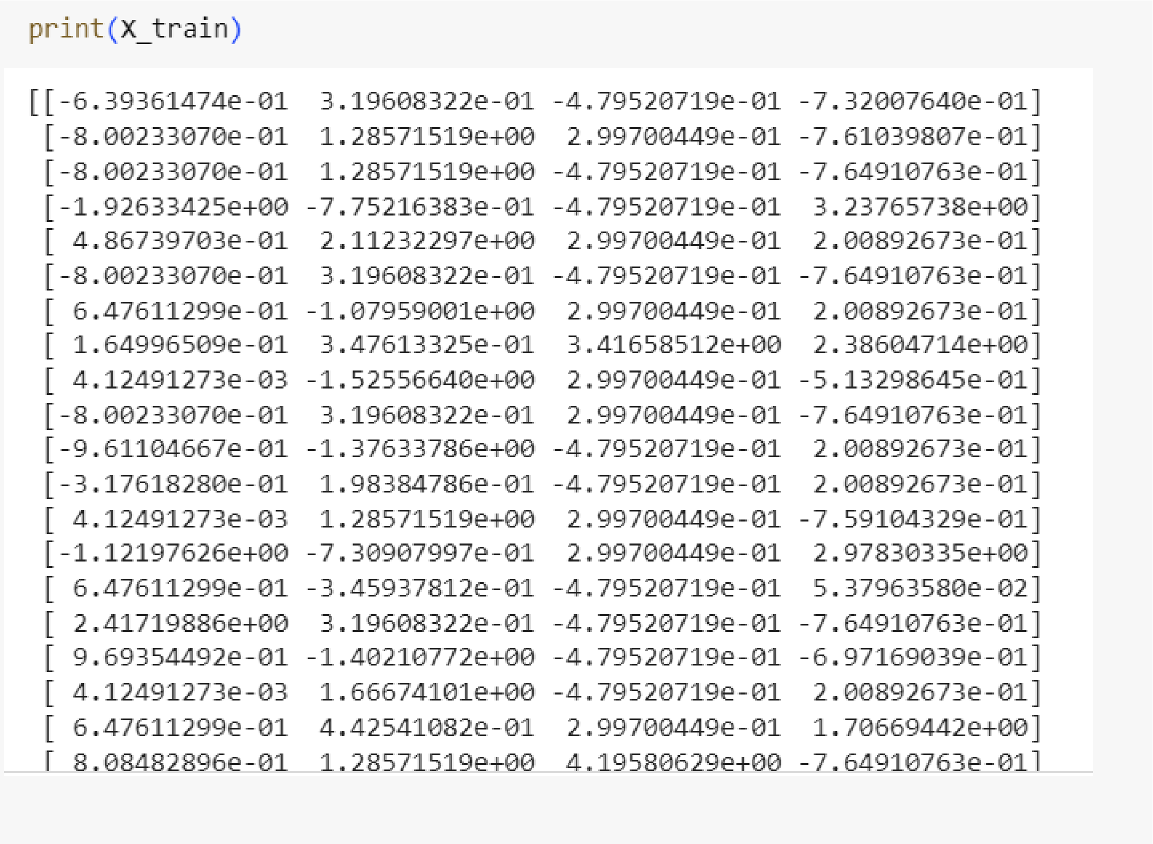
```
print(X_test)
```

```
[[8.1000e+00 1.1838e+04 6.5000e+01 1.1550e+03]
 [8.1000e+00 2.0982e+04 6.7000e+01 1.8000e+01]
 [8.5000e+00 1.1336e+04 6.5000e+01 5.0000e+02]
 [8.1000e+00 2.0982e+04 6.7000e+01 1.8000e+01]
 [7.4000e+00 1.3063e+04 6.4000e+01 1.6050e+03]
 [8.1000e+00 2.0982e+04 6.7000e+01 5.0000e+00]
 [8.1000e+00 2.6534e+04 6.5000e+01 1.9000e+01]
 [8.1000e+00 2.0982e+04 6.5000e+01 1.8000e+01]
 [9.0000e+00 2.8330e+04 6.7000e+01 1.0000e+00]
 [8.4000e+00 1.4123e+04 6.5000e+01 1.2640e+03]
 [7.9000e+00 1.7798e+04 6.7000e+01 5.0000e+02]
 [9.0000e+00 2.0982e+04 6.6000e+01 2.0000e+00]
 [8.1000e+00 2.0982e+04 6.5000e+01 4.0000e+01]]
```

```
print(y_test)
```

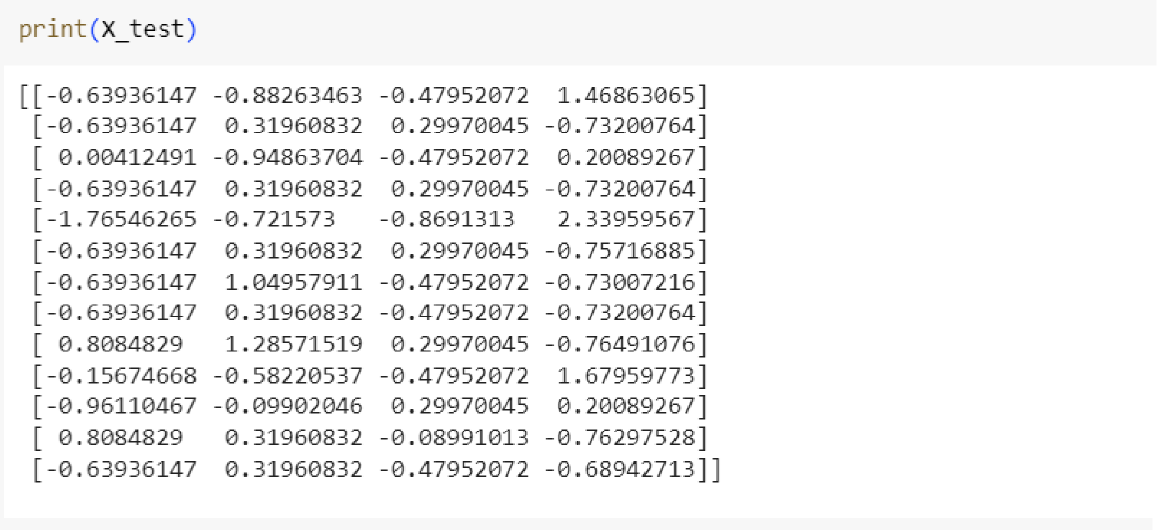
```
[[1]
 [1]
 [1]
 [1]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [1]
 [0]
 [0]]
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```



```
print(X_train)
```

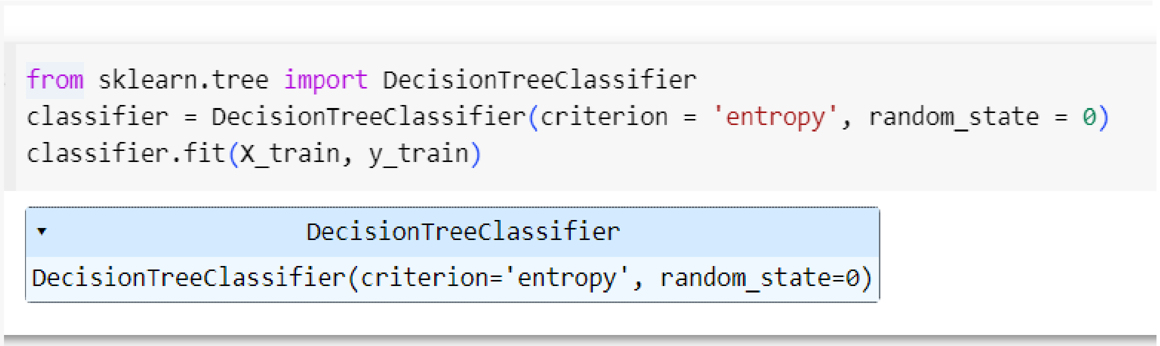
```
[[-6.39361474e-01  3.19608322e-01 -4.79520719e-01 -7.32007640e-01]
 [-8.00233070e-01  1.28571519e+00  2.99700449e-01 -7.61039807e-01]
 [-8.00233070e-01  1.28571519e+00 -4.79520719e-01 -7.64910763e-01]
 [-1.92633425e+00 -7.75216383e-01 -4.79520719e-01  3.23765738e+00]
 [ 4.86739703e-01  2.11232297e+00  2.99700449e-01  2.00892673e-01]
 [-8.00233070e-01  3.19608322e-01 -4.79520719e-01 -7.64910763e-01]
 [ 6.47611299e-01 -1.07959001e+00  2.99700449e-01  2.00892673e-01]
 [ 1.64996509e-01  3.47613325e-01  3.41658512e+00  2.38604714e+00]
 [ 4.12491273e-03 -1.52556640e+00  2.99700449e-01 -5.13298645e-01]
 [-8.00233070e-01  3.19608322e-01  2.99700449e-01 -7.64910763e-01]
 [-9.61104667e-01 -1.37633786e+00 -4.79520719e-01  2.00892673e-01]
 [-3.17618280e-01  1.98384786e-01 -4.79520719e-01  2.00892673e-01]
 [ 4.12491273e-03  1.28571519e+00  2.99700449e-01 -7.59104329e-01]
 [-1.12197626e+00 -7.30907997e-01  2.99700449e-01  2.97830335e+00]
 [ 6.47611299e-01 -3.45937812e-01 -4.79520719e-01  5.37963580e-02]
 [ 2.41719886e+00  3.19608322e-01 -4.79520719e-01 -7.64910763e-01]
 [ 9.69354492e-01 -1.40210772e+00 -4.79520719e-01 -6.97169039e-01]
 [ 4.12491273e-03  1.66674101e+00 -4.79520719e-01  2.00892673e-01]
 [ 6.47611299e-01  4.42541082e-01  2.99700449e-01  1.70669442e+00]
 [ 8.08482896e-01  1.28571519e+00  4.19580629e+00 -7.64910763e-01]
```



```
print(X_test)
```

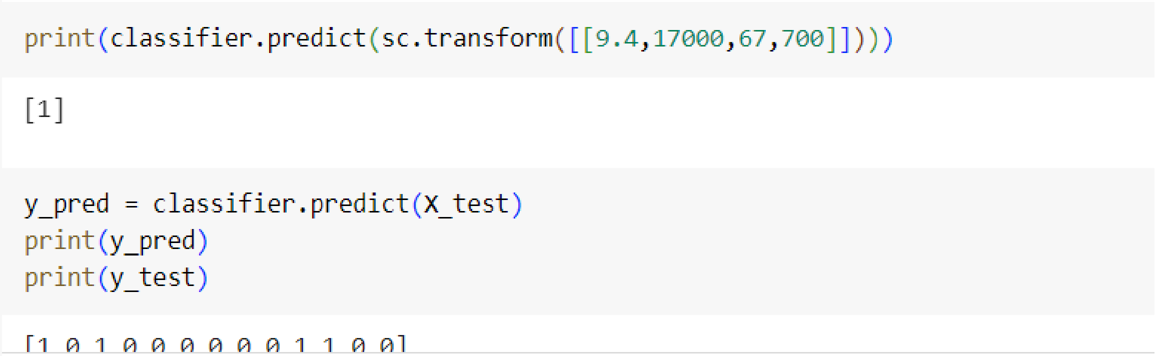
```
[[-0.63936147 -0.88263463 -0.47952072  1.46863065]
 [-0.63936147  0.31960832  0.29970045 -0.73200764]
 [ 0.00412491 -0.94863704 -0.47952072  0.20089267]
 [-0.63936147  0.31960832  0.29970045 -0.73200764]
 [-1.76546265 -0.721573   -0.8691313   2.33959567]
 [-0.63936147  0.31960832  0.29970045 -0.75716885]
 [-0.63936147  1.04957911 -0.47952072 -0.73007216]
 [-0.63936147  0.31960832 -0.47952072 -0.73200764]
 [ 0.8084829   1.28571519  0.29970045 -0.76491076]
 [-0.15674668 -0.58220537 -0.47952072  1.67959773]
 [-0.96110467 -0.09902046  0.29970045  0.20089267]
 [ 0.8084829   0.31960832 -0.08991013 -0.76297528]
 [-0.63936147  0.31960832 -0.47952072 -0.68942713]]
```

4.3 Decision Tree Algorithm



```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

```
▾                DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

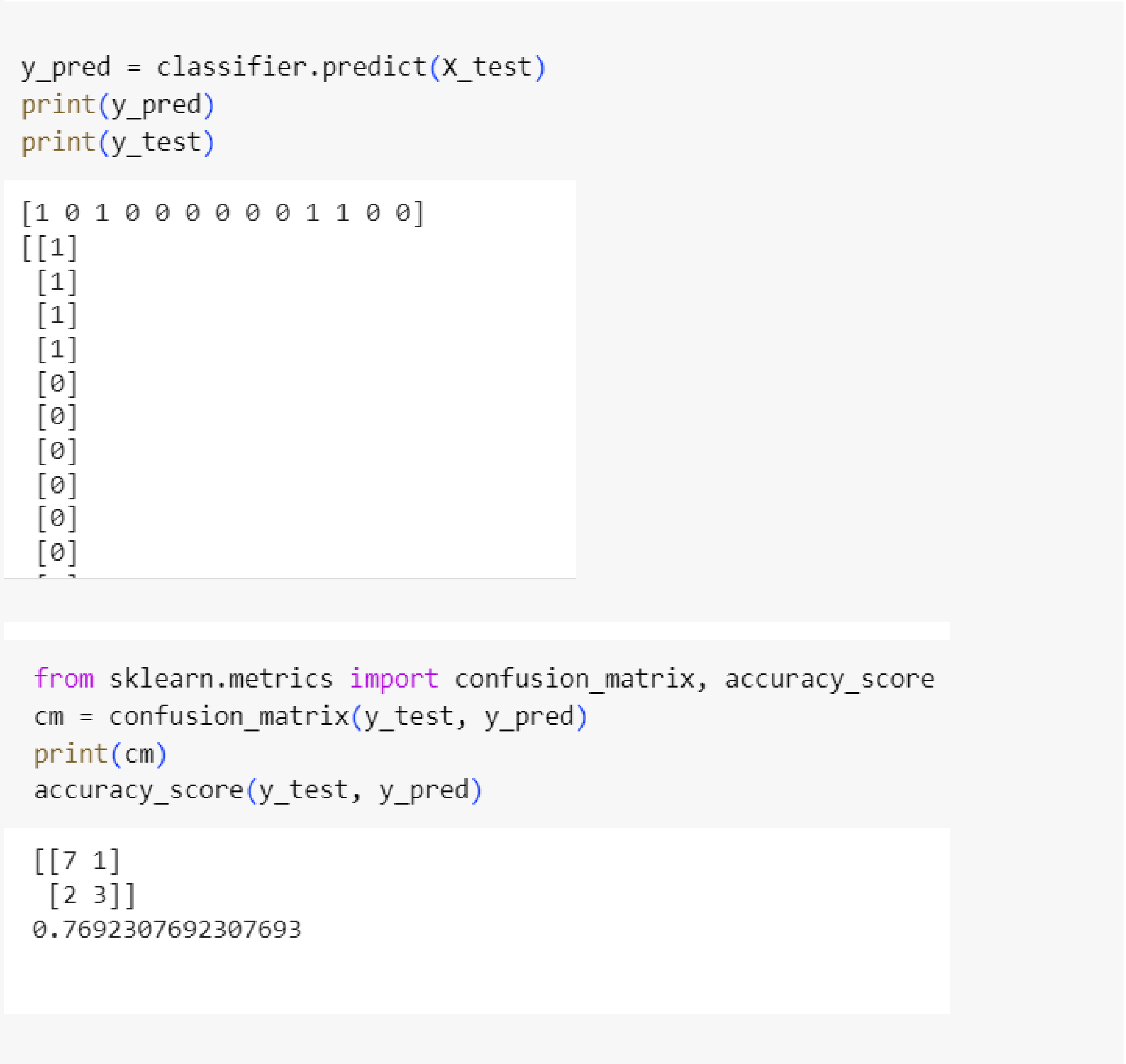


```
print(classifier.predict(sc.transform([[9.4,17000,67,700]])))
```

```
[1]
```

```
y_pred = classifier.predict(X_test)
print(y_pred)
print(y_test)
```

```
[1 0 1 0 0 0 0 0 0 1 1 0 0]
```



```
y_pred = classifier.predict(X_test)
print(y_pred)
print(y_test)
```

```
[1 0 1 0 0 0 0 0 0 1 1 0 0]
[[1]
 [1]
 [1]
 [1]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[7 1]
 [2 3]]
0.7692307692307693
```

4.4 Clustering

CONCLUSION:

From the above analysis technique, we can conclude that,

Preprocessing is a critical step in data warehousing and data mining that involves preparing and cleaning data to make it suitable for analysis. Data in its raw form often contains noise, inconsistencies, missing values, and other issues that can impede the effectiveness of data analysis and data mining. Preprocessing aims to address these issues and create a clean, structured dataset that can be used for meaningful analysis and mining.

Data visualization plays a crucial role in the fields of data warehousing and data mining, as it helps in making sense of large and complex datasets, uncovering patterns, and gaining insights from the data. Data visualization helps transform raw data into actionable insights, making it a critical component in the decision-making process and in gaining a deeper understanding of data for both business intelligence and data mining applications.

In our case, we have trained and tested the hotel dataset to predict the the hotel is Affordable or not by giving certain parameters like Rating, Reviews, Price, and Discount. The accuracy of the trained dataset is 77%.

REFERENCES:

https://www.researchgate.net/publication/373013644_Data_Analysis_End-to-End_on_Hotel_Management_Project

https://nycdatascience.com/blog/r/hotel-bookings-data-analysis/

https://jovian.com/shivanshusingla27/hotel-booking-data-analysis-eda-project