Artificial Intelligence

CA3 REPORT

Setareh Dehghanfard 810002005

CA description

In this CA, we aim to estimate the price of hotels in Europe based on some of their features. We are treating the problem as a classification problem, which means that we will only predict if a price is more than a threshold or if it is less than that. We will use several methods to make this model. All outputs can be viewed in the notebook and are not repeated here.

ExploratoryDataAnalysis

Here the goal is to get to know the data better.

Here is the overall structure of the dataset:

```
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 7322 entries, 0 to 7321

Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   name                  7322 non-null   object
 1   location              7322 non-null   object
 2   price                 7322 non-null   object
 3   rating                7129 non-null   float64
 4   quality               7169 non-null   object
 5   review                7169 non-null   object
 6   bed                   7299 non-null   object
```

```
 7   size                   2454 non-null   object
 8   distance_from_center   7322 non-null   float64
 9   room_type              7322 non-null   object
 10  nights                 7322 non-null   object
 11  adults                 7322 non-null   object
 12  free_cancellation      583 non-null    object
dtypes: float64(2), object(11)
memory usage: 743.8+ KB
```

Which shows us the name of all the features, what type each one is and how much non-null data each feature has.

For further preparation, the following steps were done:

- Used `describe()` and `info()` to summarize the data.
- Addressed missing values and outliers in the dataset.
- Plotting the number of unique values for each feature
- Histograms for continuous variables like `price` and `rating`.
- Scatter plots to explore relationships between `the target and the features`
- Correlation heatmaps to identify the role of each feature in prediction, seems that the number of nights plays a great role in predicting the target.
- Quality descriptions aligned closely with user ratings and prices.

Data Preprocessing and Feature Engineering

First the numerical part of some features were extracted and replaced. Some categorical features were also replaced by numbers. Such as free_cancalation and quality.for the location feature, since there is a big variety to the values and can be too complicated for the model to learn, I only kept the city part, and replaced it with numbers for each city.

Next was removing some columns, as said before, from location I only kept the city, which means i removed the column location. Also the name of the hotel does not contribute much to the price in most cases, so that was removed as well, I also removed room_type since it included a big variety of unique values.

There are several ways to replace the NAN values. The pandas library offers a lot of functions to do that. For this dataset I decided on using the median value of each feature to replace the NANs. Other methods could be using average or the mode of each feature or column.

Next the prices were classified into two classes based on the median value for the price.

Model Development, Training and Evaluation

I first used a MinMaxScaler() to normalize some of the columns. Based on the values of each column, I decoded on normalizing the features review, size and distance_from_center. Optimization process works more efficiently when the features are on a similar scale. Without normalization, features with larger numerical ranges may dominate the learning process, which is why normalization is important here. Normalizing could be done on different scales.

Next the data is splitted to train and test, also X and y where separated. Y being the target or price, and X being the features excluding the price.

Here is the accuracy for each model using optimized parameters, other metrics were also calculated which can be viewed in the notebook.

Naive Bayes	Decision Tree	Random Forest	Adaptive Boosting n_estimator = 50	Adaptive Boosting n_estimator = 100	Adaptive Boosting n_estimator = 200	XG Boost	Boosting from Scratch
0.8293	1.0	0.9986	0.9931	0.9972	1.0	1.0	0.8566

Comparison with Library Implementation

The custom boosting implementation builds the SAMME algorithm step by step, calculating weights, errors, and predictions manually. This allows flexibility for experiments but is slower, more complex, and prone to mistakes. In contrast, Scikit-learn's `AdaBoostClassifier` is fast, reliable, and easy to use, as it handles all calculations and

edge cases automatically. The custom method is useful for learning or research, while Scikit-learn is better for practical, efficient applications.