Hotel price predictor

This project aimed to create regression models that could predict the hotel’s prices.

The ability to predict these prices with precision has the potential to empower travelers with better budgeting tools and assist hotel businesses in setting competitive rates.

I started the project with web scraping, where data is collected from Booking.com's vast repository of hotel listings for four Italian cities: Florence, Milan, Naples, and Rome. The raw scraped data includes information about the hotel’s name, location, price for one night at specified date 22 February 2024, rating - both expressed in points and in words, room type, guest reviews, distance to the city center, and information if the offer includes free breakfast and free cancellation.
The result of scraping is a dataset containing 4004 entries and 10 columns.

Data

- hotel name – the name of the hotel - categorical data,
- location – the address of the hotel - categorical data,
- price - price for one night at specified date 22 February 2024 in Polish złoty - numerical data,
- rating_score - the hotel's overall score from 1 to 10 - numerical data,
- rating - the hotel's overall score in words - categorical data,
- free_cancelation – information on whether the offer includes a free cancellation or not. If „Free cancellation” appears in the entry that means the offer includes free cancellation. The lack of this information in the entry means that there is no free cancellation for the offer - categorical data,
- breakfast - information on whether the offer includes a free breakfast or not. If „Breakfast included” appears in the entry that means the offer includes free breakfast. The lack of this information in the entry means that there is no free cancellation for the offer categorical data,
- reviews - number of reviews - numerical data,
- room_type - type of room on offer - categorical data,
- distance_to_the_city_center - distance from the hotel to the city center both in meters and kilometers depending on the distance - numerical data.

Data preprocessing

After the data collection phase, I applied data preprocessing to clean, transform, and adapt the data to a format suitable for analysis and the machine learning process.
First I checked if the dataset contained duplicates because because I could not control this during the data scraping process.
After checking and removing duplicates, the dataset contains 3194 entries.
The next steps in data preprocessing were:

- Extracting the city name from the location column. The „location” column contains the address of the hotel. For the data analysis and machine learning process the exact address of the hotel would not be very meaningful, so I decided to extract the city name from the address data and create a new column „City”.

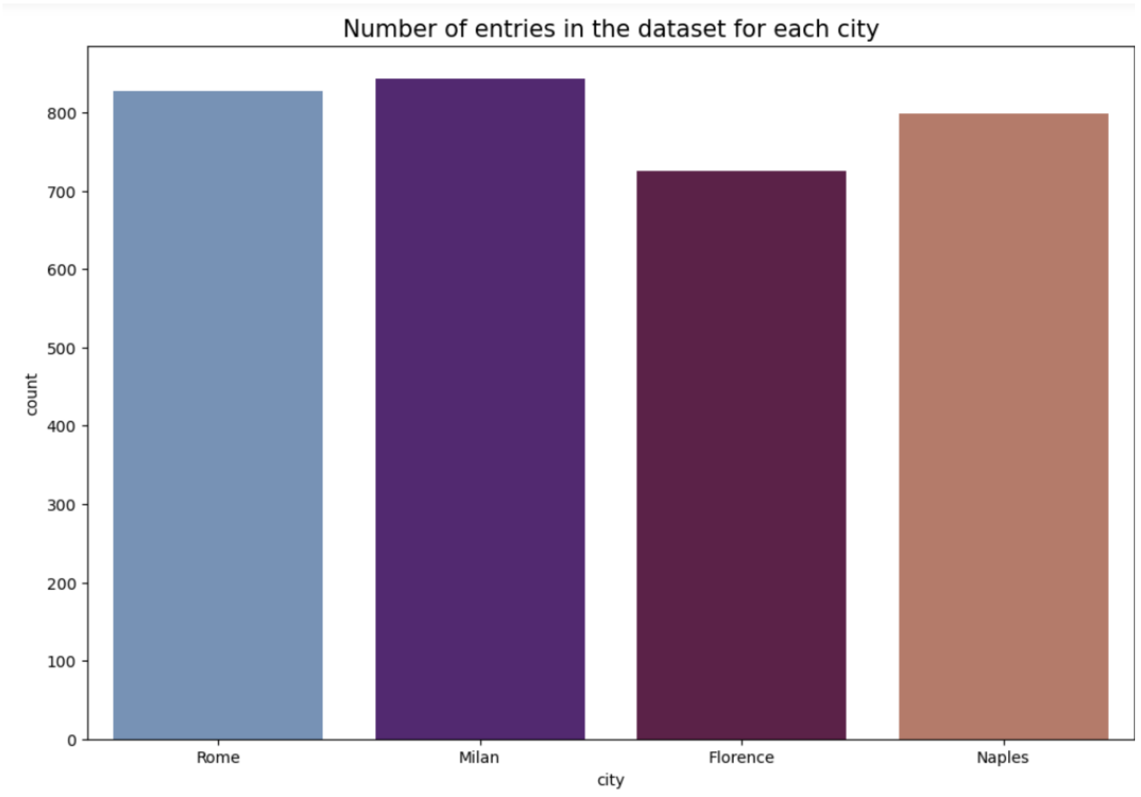
- Converting columns „Free cancellation” and „Breakfast” to boolean. Since these columns contain information about whether the option is included in the offer’s price or not I decided to convert them to boolean type.
- Converting columns „Price” and „Reviews” to numerical.
- Unifying the data in the column „distance_to_the_city_center”. This column contains data expressed in both - meters and kilometers. In order to make the data in the column make sense for the data analysis and machine learning process, I decided to convert kilometers to meters, so all the data in the column would be expressed in meters.
- Categorizing data in „room_type” column. Since the original dataset contained 325 unique types in the "room_type" column (probably when listing, you don't select a room type from a list but can define it yourself) I decided to categorize them into larger groups: "Double room", "Twin room", "Apartment" and "Other" based on the description.
- Filling missing values using k-Nearest Neighbors in the columns „rating_score” and „reviews”.
- Handling outliers in the target „price” column by first detecting them by using Isolation Forest and then replacing detected outliers with the median value for the column.
- Encoding the categorical columns „room_type” and „city” using LabelEncoder.
- Dropping the redundant columns „Hotel name”, „Location” and „Rating”. Since the columns „Hotel name” and „Location” have no meaningful data for the data analysis and machine learning process I decided to drop them. After checking the contents of the „Rating” column, it turned out that there is probably an error on the page, and below the "Good" rating it displays "Rating score" instead of the specific rating. In addition, the column contains the same data as the column „Rating_score” but it is expressed in words, so I decided to remove it from the dataset.

All of the above actions were first performed on the training dataset and later on the test dataset. If the action required the use of a model (filling in missing values by KNNImputer, label encoding by LabelEncoder, handling outliers by Isolation Forest) the model was fitted only on the training dataset and then only the transform was performed on the test dataset.

Data analysis

For the following analysis, I used the dataset after data processing.

Let's start by checking the contents of the „city" column. I will check if we have more or less the same amount of data from each city or maybe for some cities we have much more data.

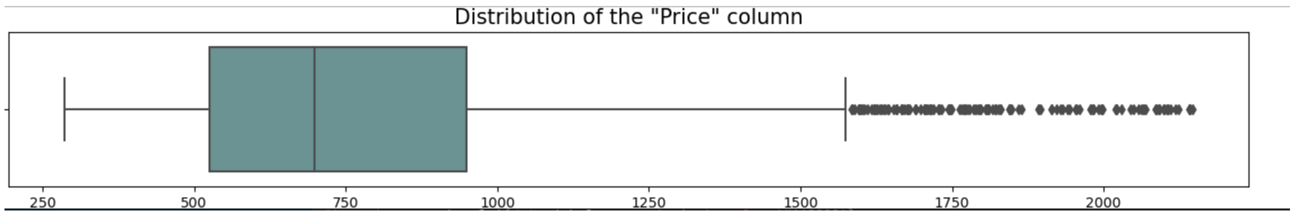


Almost every city has a similar amount of data about 800 entries in the dataset, only Florence has fewer entries than the other cities. You can see the exact values in the table below.

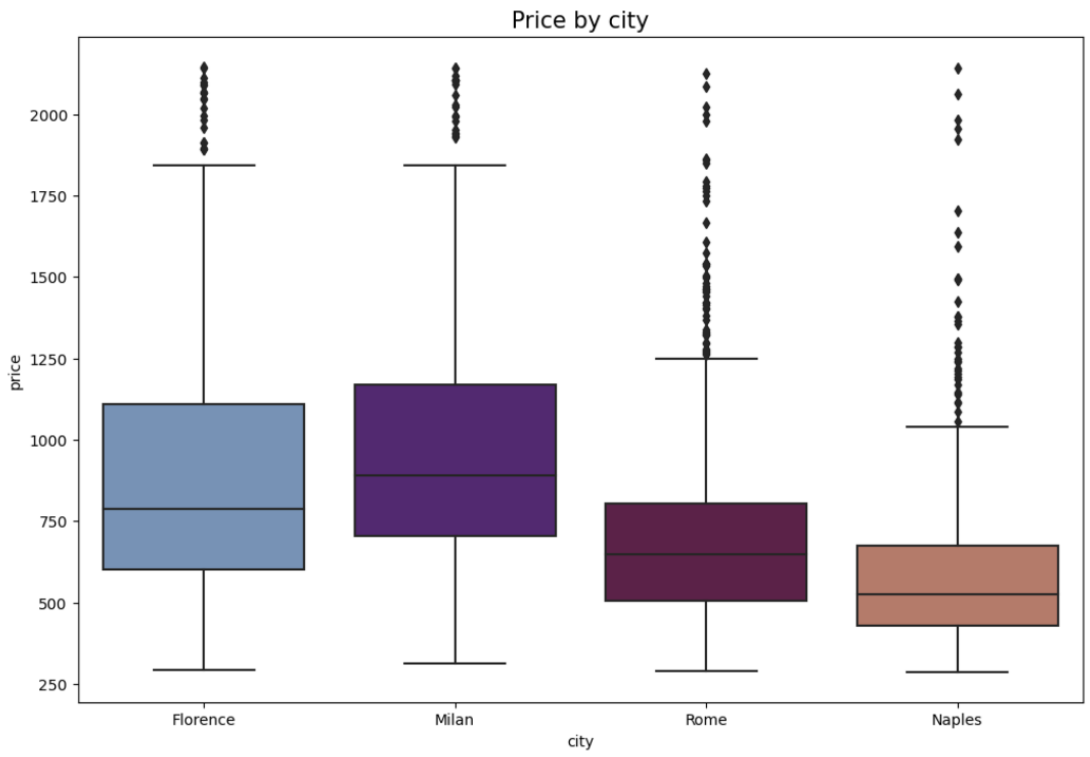
City	Number of entries in the dataset
Milan	843
Rome	827
Naples	799
Florence	725

In the machine learning process, I would like to predict price. Let's check the „price" column's distribution.

On the above boxplot, we can see that most values are in the range of about 520 to 950 with the median value about 700.

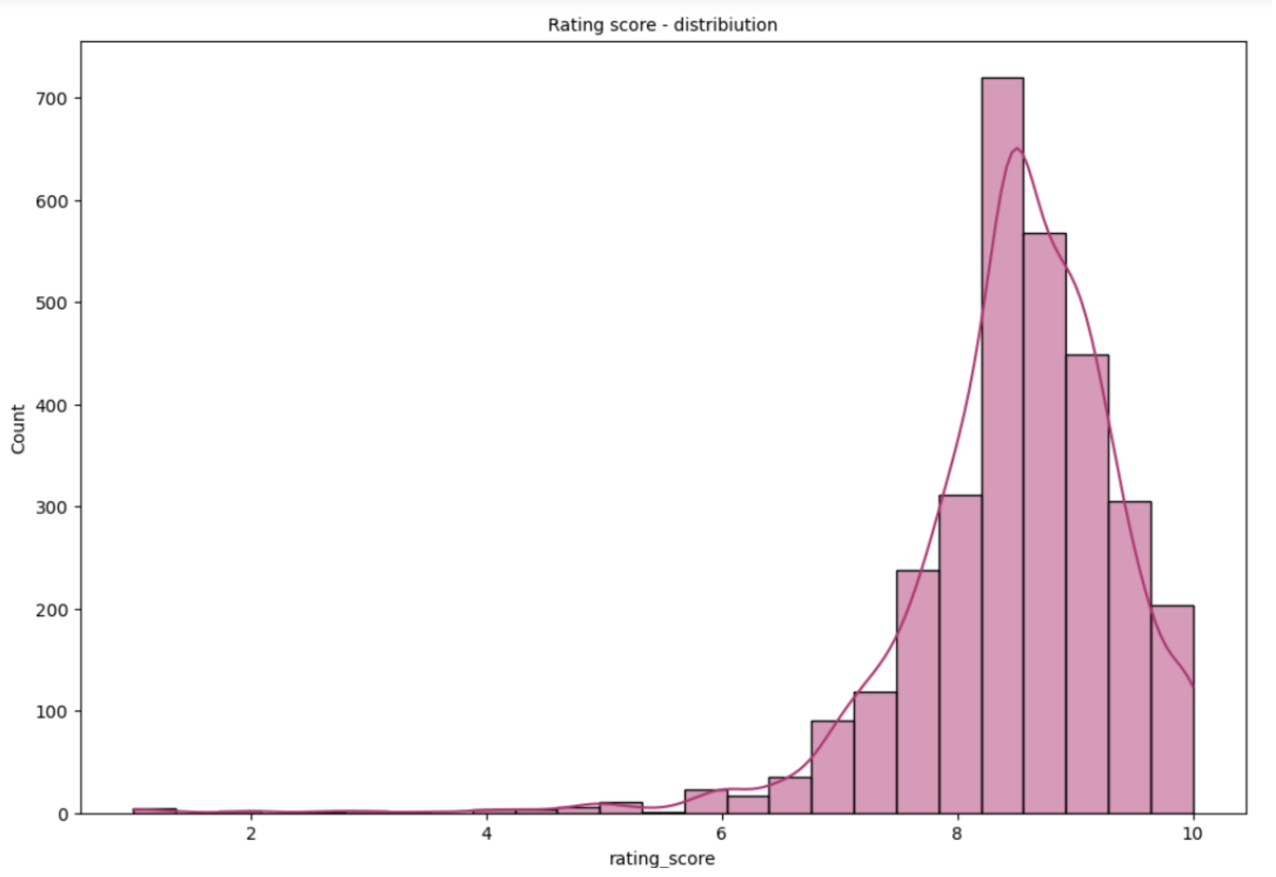


The below plot shows the „Price” distribiution in each city.



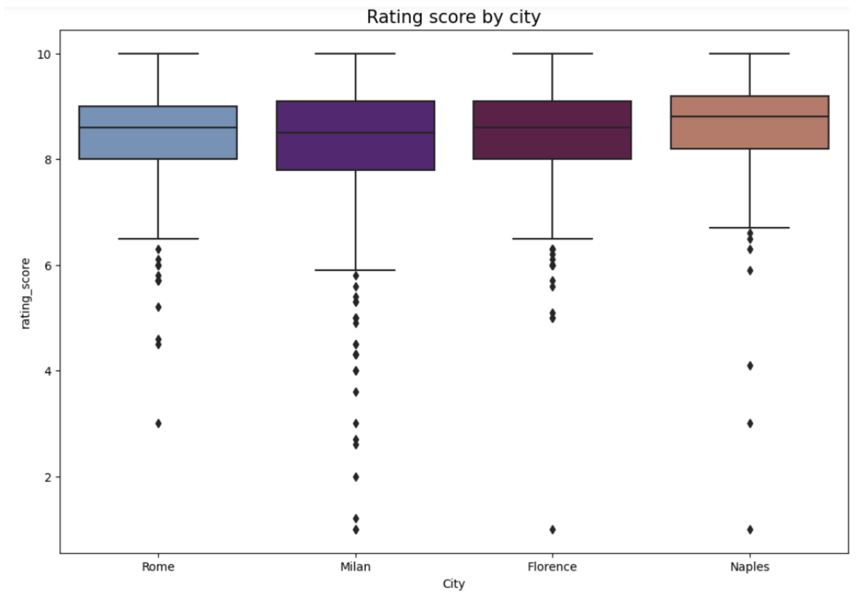
From the plot above, we can deduce that the most expensive offers (both the price range and their median) are in Milan, followed by Florence, then Rome, and the cheapest offers are in Naples. For Rome and Naples where prices are lower, we can see many more outliers than for Milan and Florence.

Let’s check the rating score distribution by plotting a histogram for this column.



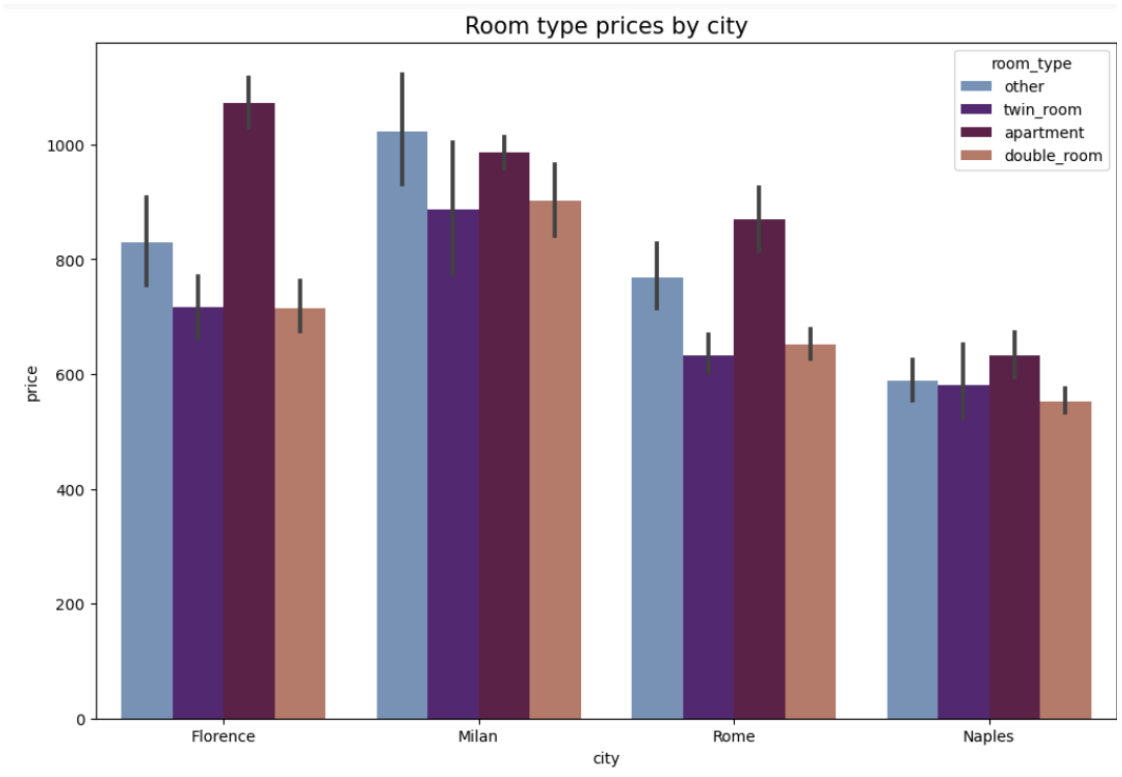
From the histogram above we can infer that the hotels in the dataset usually have a good rating. Most of the values are between 8 and 9, which in my opinion is a very high score if the scale is from 1 to 10. We can see that below 6 there are already relatively few values. About 200 hotels in the dataset received the highest rating: 10.

Now I would like to take a look at the rating values by city.



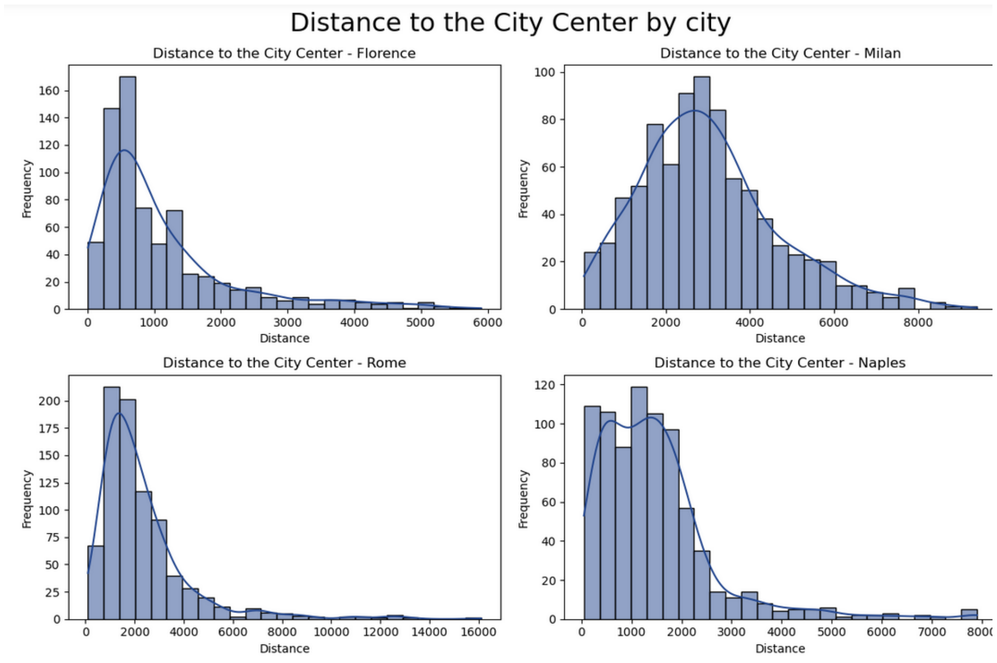
Comparing the boxplots above we can infer that the values for the „rating_score” for each city look pretty similar. The range of the most frequent values is between 8 to 9 for each city. Each city has hotels with the highest 10 rating, but only Rome doesn’t have the lowest „1” rating. The lowest rating for Rome is „3”. We can observe many outliers in „Rating_score” for Milan, more than for other cities. Naples has the fewest outliers in the „Rating_score” column.

Now let's check what the prices of the different types of rooms in the cities look like.

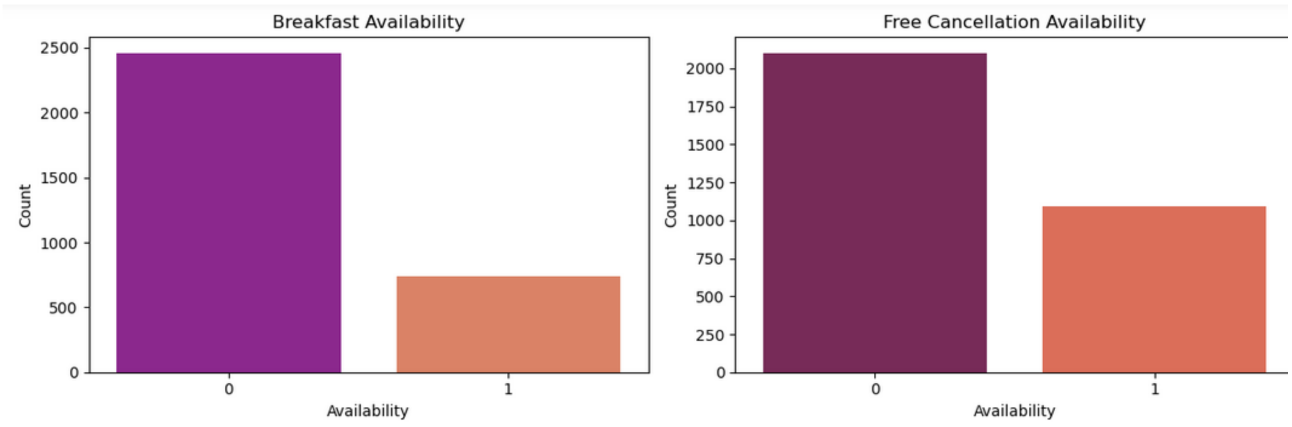


In all cities except Milan, the most expensive accommodation type is apartments. In Milan, the most expensive is the "Other" type, and this type includes all the types of rooms that did not qualify for groups. In Rome and Milan, a double room is more expensive than a twin room, in Florence the prices of these types are similar, but in Naples, a twin room is more expensive than a double room.

Checking the data in the "distance_to_the_city_center" column for each city, by plotting a histogram for the variable for each city, we can infer that in Florence, Naples, and Rome most accommodations are distributed close to the city center - up to 1,000 meters away in Florence and 2,000 meters away in Naples and Rome. In Milan, most of the hotels are within 4,000 meters of the city center.

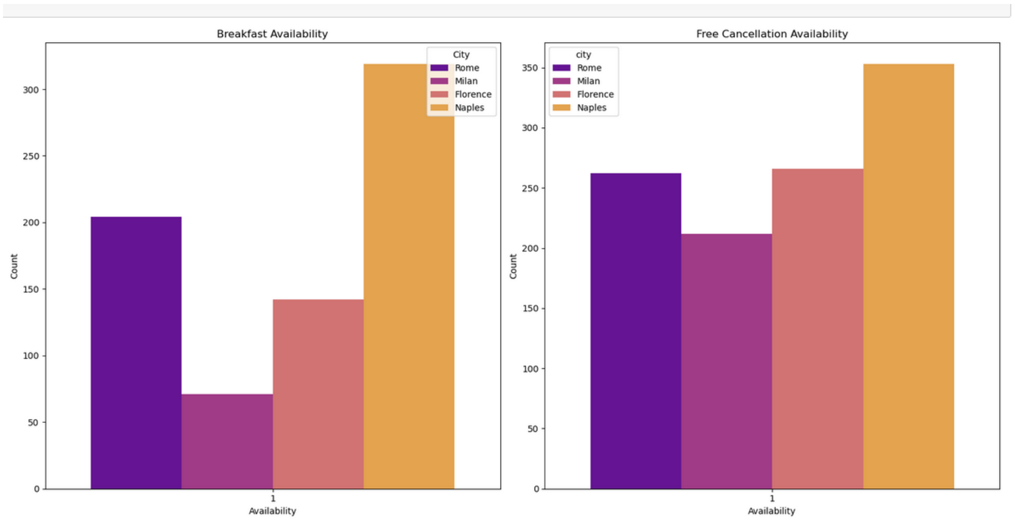


Additional benefits such as free breakfast or free cancellation if our plans change are a nice addition. Let's see how many hotels in the dataset offer these options.



The bar plots above show that hotels are more likely to not offer free breakfast than to offer it. Only about 700 (and that's less than 1/4) listings include free breakfast. For the free cancellation, it's a bit better, about 1000 - 1/3 of the hotels allow this option.

In the end, I decided to see if more hotels offer additional amenities in any of the cities.



The above countplots show that the largest number of hotels that offer free breakfast or free cancellation is in Naples, followed by Rome. The fewest offers with free breakfast or free cancellation are in Milan.

Machine learning Process

I created three types of machine learning models for my prediction. This method helped me to find the best solution for the problem. In this section, I will briefly talk about the machine-learning methods I used. TO optimize hyperparameters in the models I used the Optuna library. This helped me make these models work better and predict hotel prices more accurately.

Optuna library

Optuna is an open-source Python library for automated hyperparameter optimization. It provides a flexible and efficient framework for tuning the hyperparameters of machine learning models and other optimization tasks. Optuna employs a variety of optimization algorithms, including Bayesian optimization and various sampling methods, to efficiently search for the best combination of hyperparameters for a given objective function.

Random Forest Regressor

Random forest method uses decision trees which are one of the most popular decision algorithms. The decision tree consists of the so-called roots and branches leading to the following vertices (nodes) where some decisions are made. The last nodes are called leaves and there, instead of making a decision, the algorithm returns classification information. Random Forest is an ensemble learning method because the decision is made based on voting of the trees. Each of these trees is created on a random subset of the training data. Therefore, each tree makes a decision based on the different data.

XGBoost (Extreme Gradient Boosting)

XGBoost is an ensemble learning method that uses the power of decision trees. It creates a collection of decision trees, with each tree correcting the errors of the previous one. These trees work together to make a final prediction through a weighted voting system, resulting in accurate and robust predictions. XGBoost is known for its efficiency and effectiveness in various applications.

K-Nearest Neighbors (KNN) Regressor

K-Nearest Neighbors (KNN) is an intuitive machine learning technique used for regression tasks. In KNN regression, instead of creating a mathematical model, predictions are made based on the similarity between a data point and its neighboring data points. The "K" in KNN represents the number of nearest neighbors considered for prediction.

To make a prediction for a new data point, KNN looks at the K closest data points from the training set. It calculates an average or weighted average of their target values to predict the target value for the new data point. KNN is particularly useful when there is no clear underlying mathematical relationship between input features and the target variable. It's a flexible and simple method that can be effective for various regression problems.

Results

I used three metrics to compare the models results: Median Absolute Error, Mean Absoulte Error, Mean absolute percentage error

Model	Median Absolute Error	Mean Absolute Error	Mean absolute percentage error
Random Forest Regressor	160.95	215.69	0.30
XGBoost	137.83	228.95	0.26
K-Nearest Neighbors	172.5	228.64	0.33

In summary, when comparing the three models:

- **XGBoost** achieved the lowest Median Absolute Error (137.83), indicating that it is the most robust in predicting the median values.
- **XGBoost** also had the lowest the lowest Mean Absolute Percentage Error (0.26), suggesting that it provided the most accurate percentage-wise predictions on average.
- **Random Forest Regressor** had a relatively lower Mean Absolute Error (215.69) compared to KNN and XGBoost, making it better in terms of average absolute prediction error.

In this comparison, XGBoost stands out as the best performing model.