|  |  |
| --- | --- |
| Student ID | Student Name |
| 2022170629 | ميسون حلمي عبدالرحمن علي شعبان |
| 2022170162 | رؤي محمد لطفي محمد |
| 2022170447 | منة ياسر حامد مصطفى |
| 2022170340 | مايا سامح احمد سمير |
| 2022170051 | أروى ابوزليمة محمد نور |

**Guest Satisfaction Prediction**

Team ID: SC\_9

[ **Feature Engineering** 11](#_Toc198462850)

[1) Explanation for some features: 11](#_Toc198462851)

[ **Filling Missing Values** 15](#_Toc198462852)

[1) Normal Cases: 15](#_Toc198462853)

[ **Encoding :** 17](#_Toc198462854)

[ **Milestone 1 Feature selection:** 19](#_Toc198462855)

[ **Visualizations** 22](#_Toc198462856)

[ host\_response\_rate 22](#_Toc198462857)

[ host\_listings\_count 22](#_Toc198462858)

[22](#_Toc198462859)

[ accommodates 23](#_Toc198462860)

[ bathrooms 23](#_Toc198462861)

[ bedrooms 24](#_Toc198462862)

[24](#_Toc198462863)

[ beds 24](#_Toc198462864)

[ nightly\_price 25](#_Toc198462865)

[ price\_per\_stay 25](#_Toc198462866)

[ security\_deposit 26](#_Toc198462867)

[- cleaning\_fee 26](#_Toc198462868)

[- guests\_included 27](#_Toc198462869)

[ extra\_people 27](#_Toc198462870)

[ minimuma\_nights 28](#_Toc198462871)

[ maximum\_nights 28](#_Toc198462872)

[ number\_of\_stays 29](#_Toc198462873)

[ number\_of\_reviews 29](#_Toc198462874)

[ review\_scores\_rating 30](#_Toc198462875)

[ Handling outliers 33](#_Toc198462876)

[ **Text Analysis & Preprocessing** 35](#_Toc198462877)

[ **Regression Model Trials with Different Selected Features** 37](#_Toc198462878)

[ **Some Graphs for Each Model Performance** 39](#_Toc198462879)

[39](#_Toc198462880)

[ **Model Performance Conclusion** 40](#_Toc198462881)

[ **Classification Model Trial with Different Selected Features** 42](#_Toc198462882)

[ **Some Graphs for Each Model Performance** 45](#_Toc198462883)

[ **Conclusion** 46](#_Toc198462884)

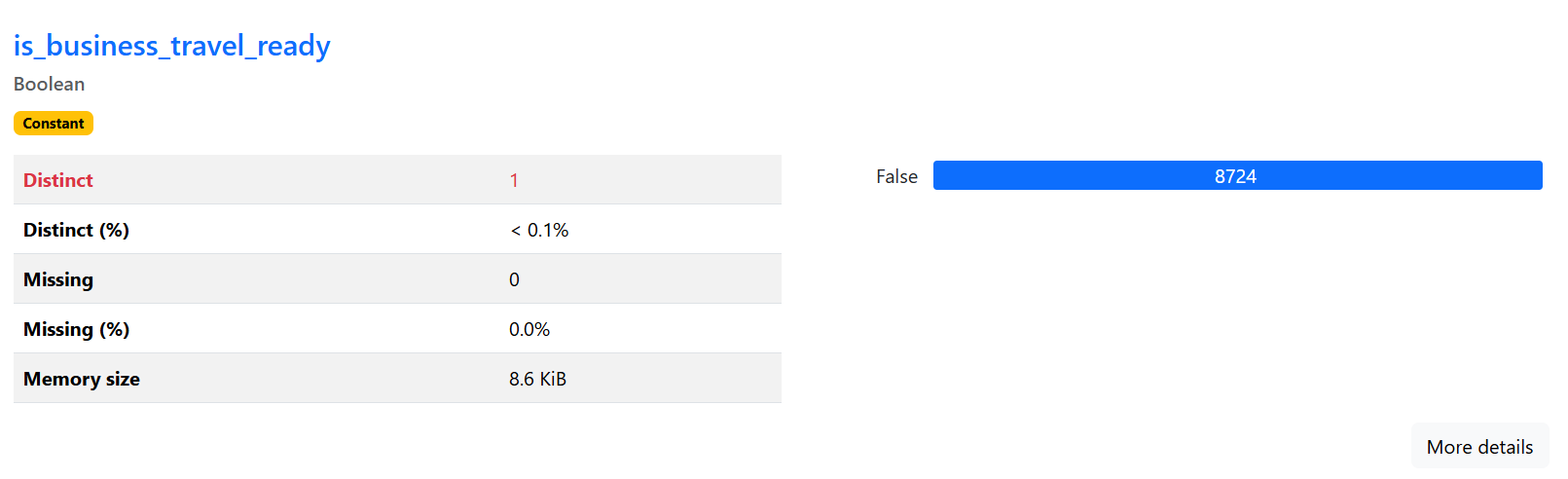
# 

* **Observations**

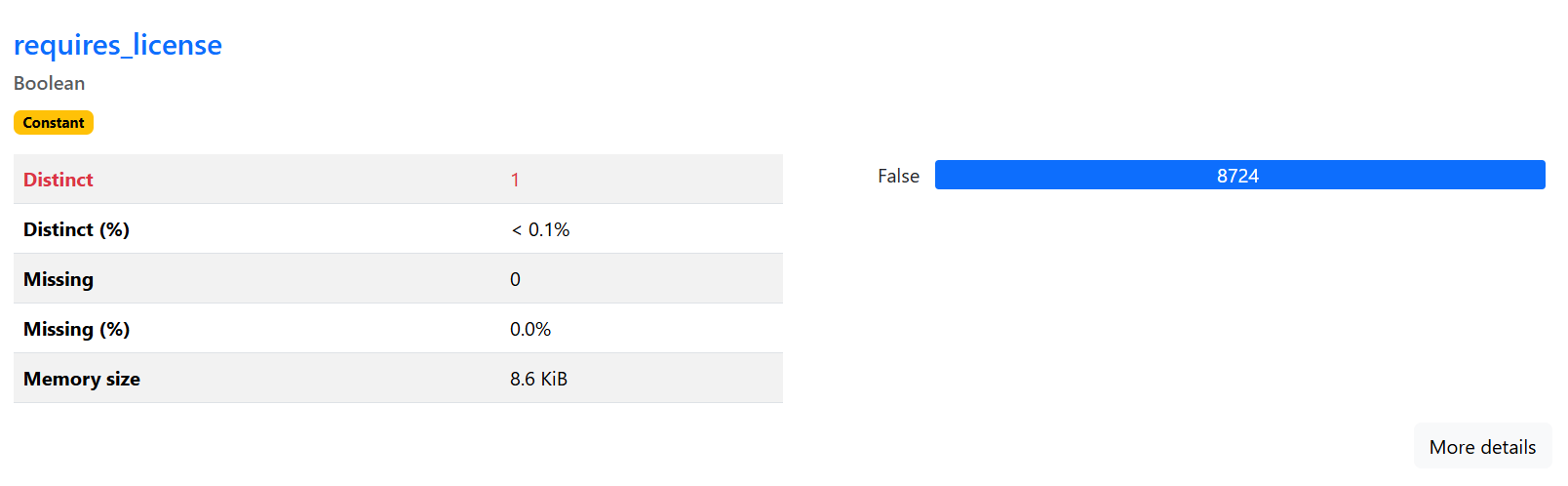
A group of text boxes

AI-generated content may be incorrect.According to our EDA We have observed the following:

* + Column name: is\_business\_travel\_ready

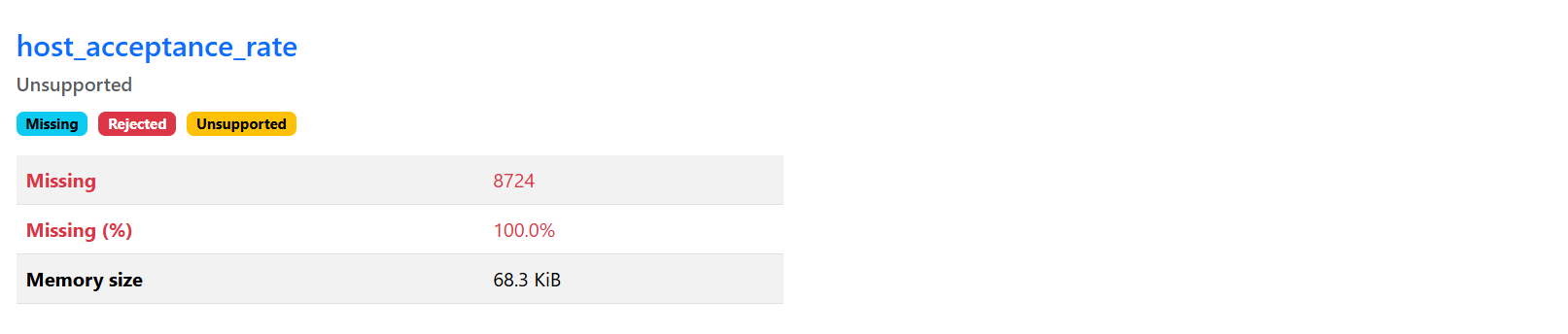
Observation: all false values so it’s not a differentiating feature

* + Column name: requires\_license

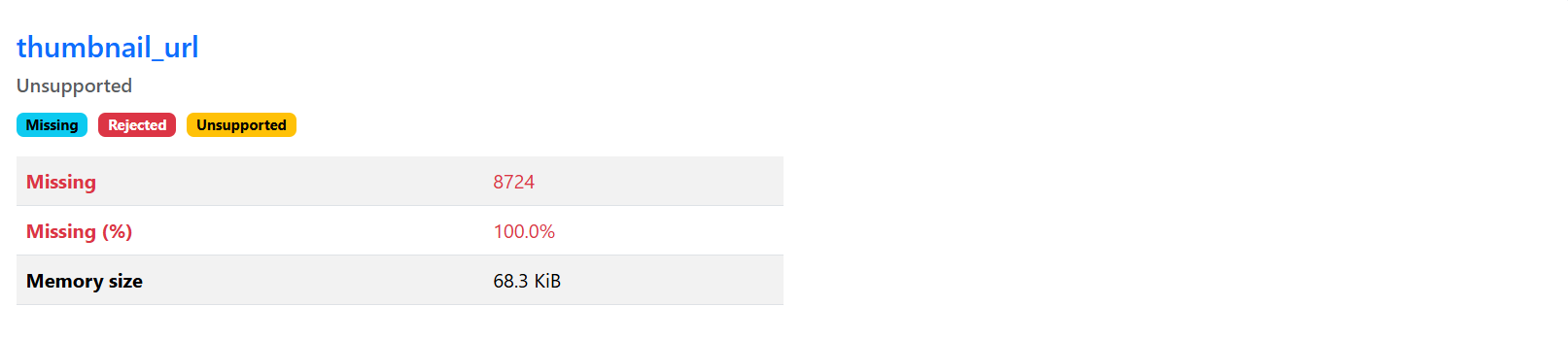
Observation: all false values so it’s not a differentiating feature

* + Column name: host\_acceptance\_rate

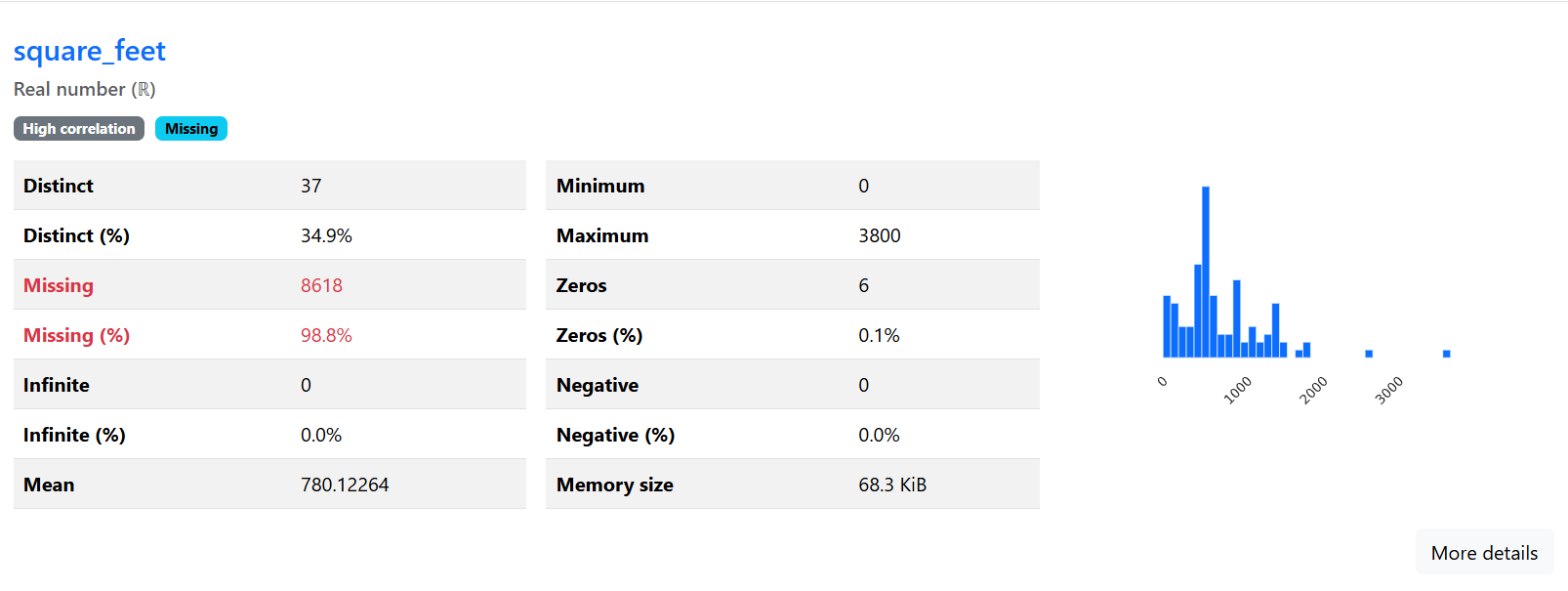
Observation: empty



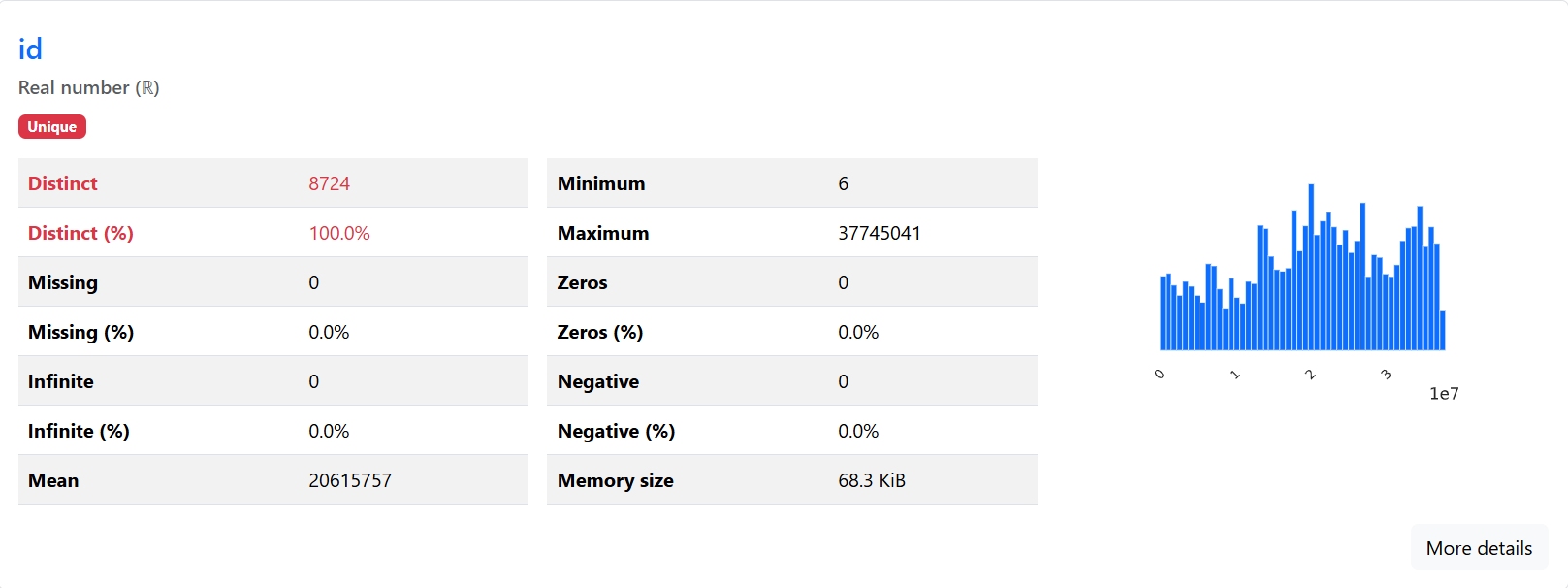
* + Column name: thumbnail\_url

Observation: empty

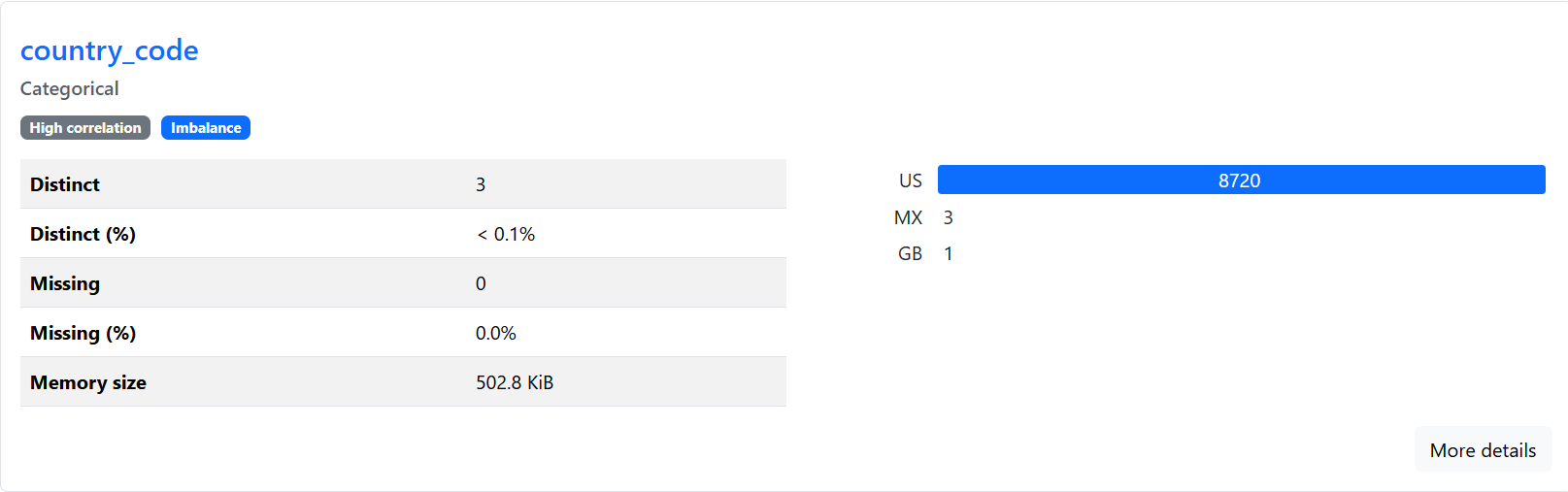
* + Column name: square\_feet

Observation: 98% empty

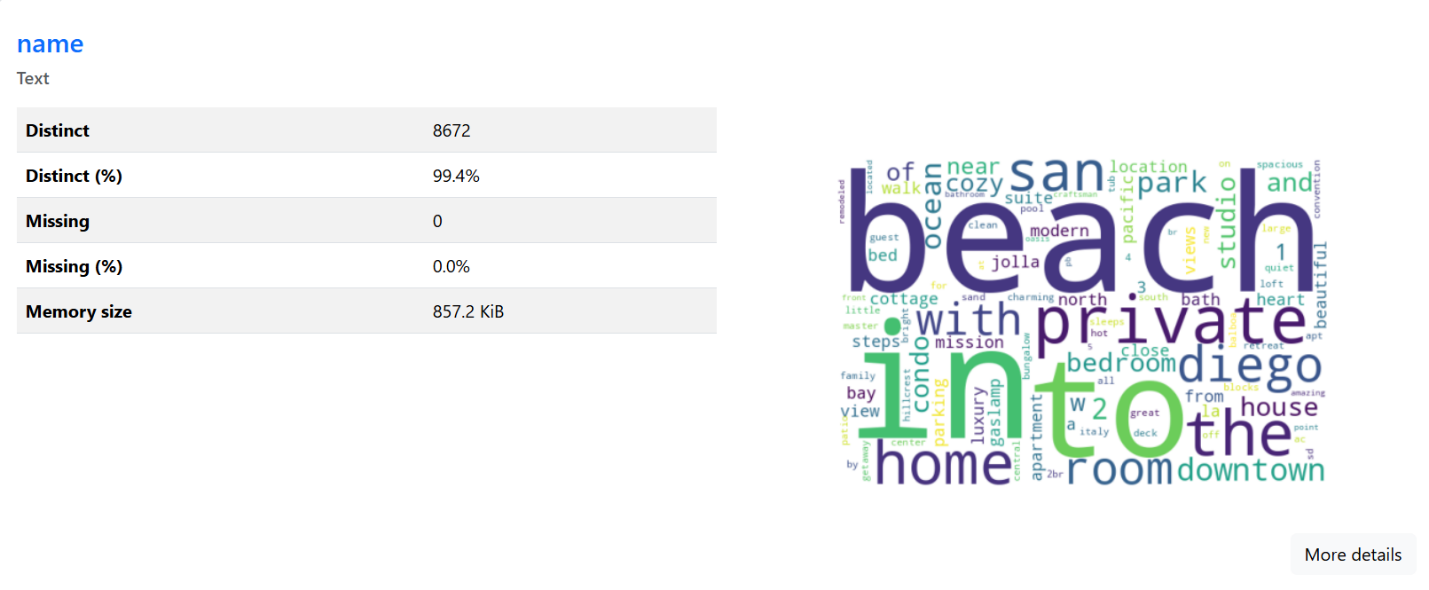
* + Column name: ID

Observation: distinct

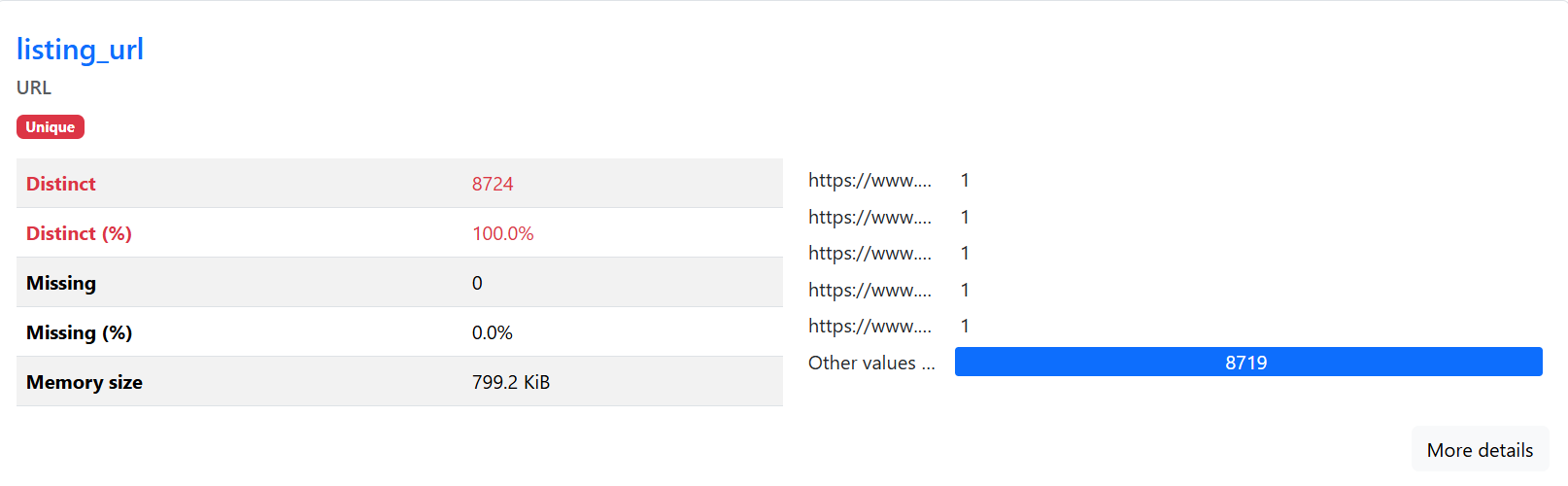
* + Column name: country\_code

Observation: all values are the same (US)

* + Column name: name

Observation: almost all values are distinct values

* + Column name: listing\_url

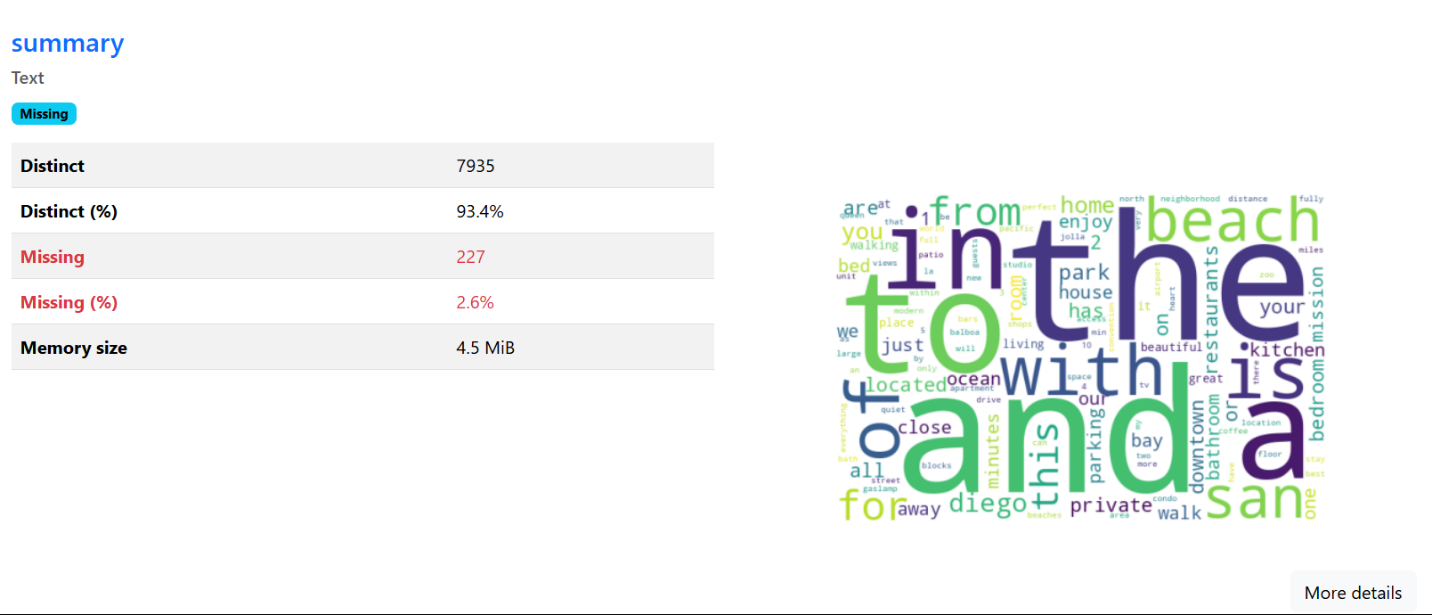
Observation: distinct and we couldn’t find new data to fetch

* + Column name: host\_url

Observation: distinct and we couldn’t find new data to fetch

* + Column name: summary

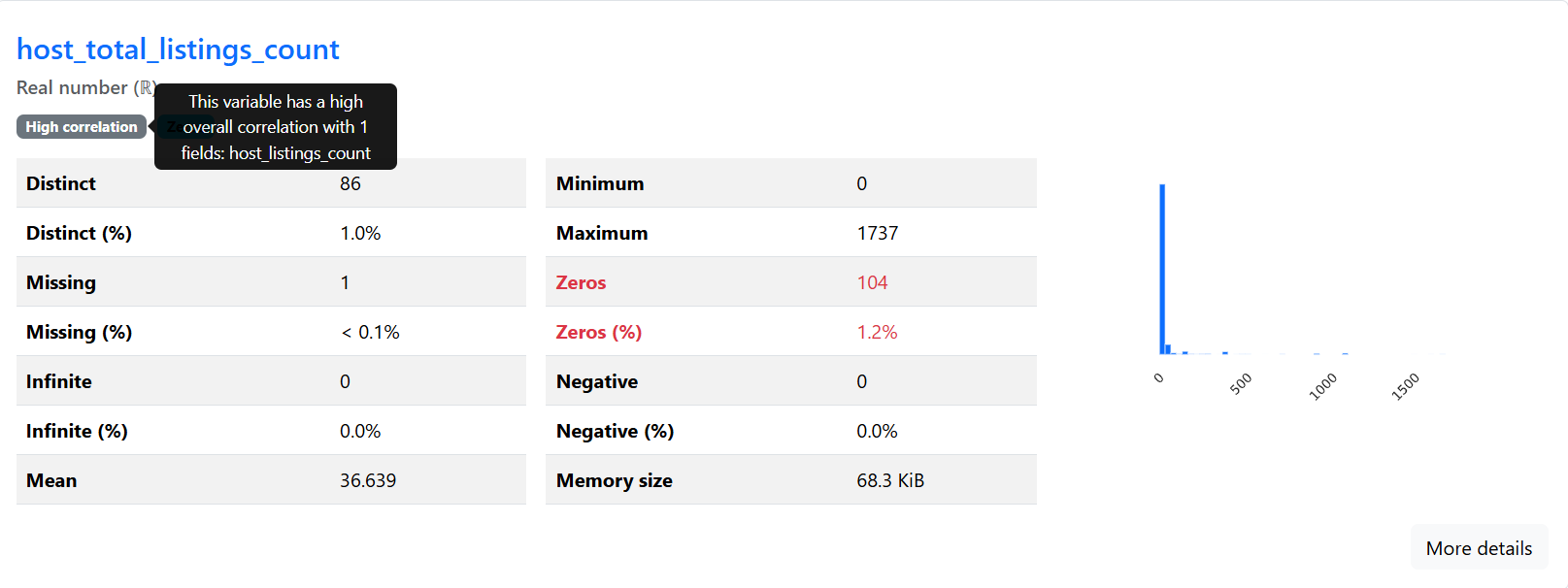
Observation: has 93% distinct values & 2.6% missing values,

Also, it almost has the same data as description and space

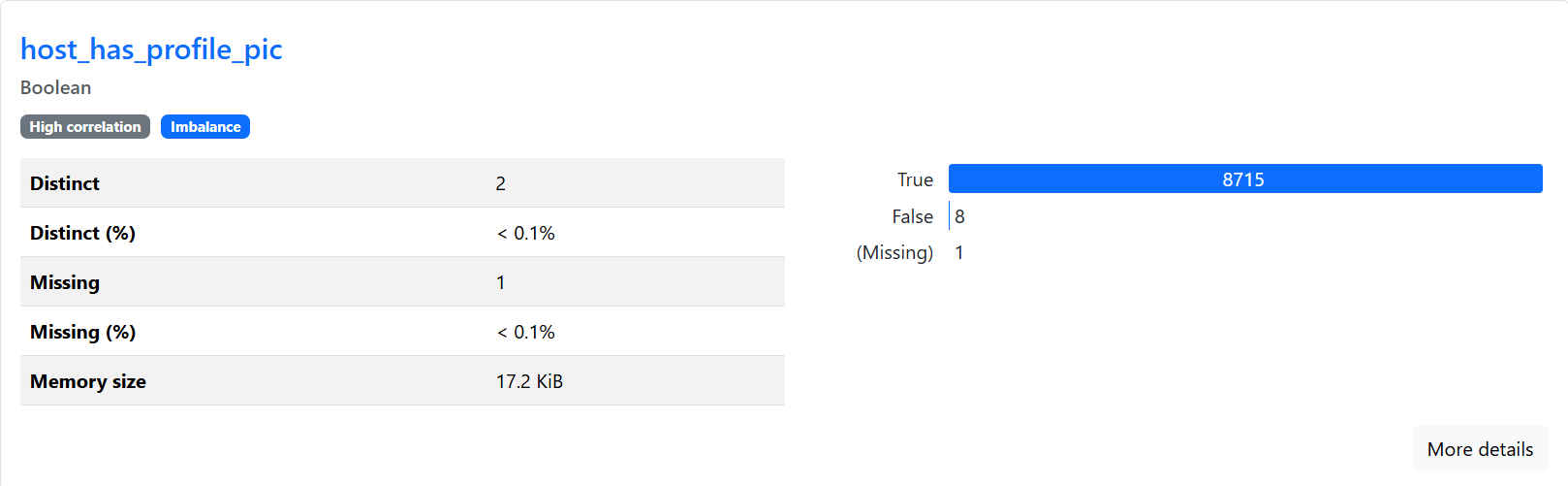
* + Column name: space
* A close-up of a text

  AI-generated content may be incorrect.Observation: has 94% distinct values & 18% missing values, it Also, it almost has the same data as description and summary

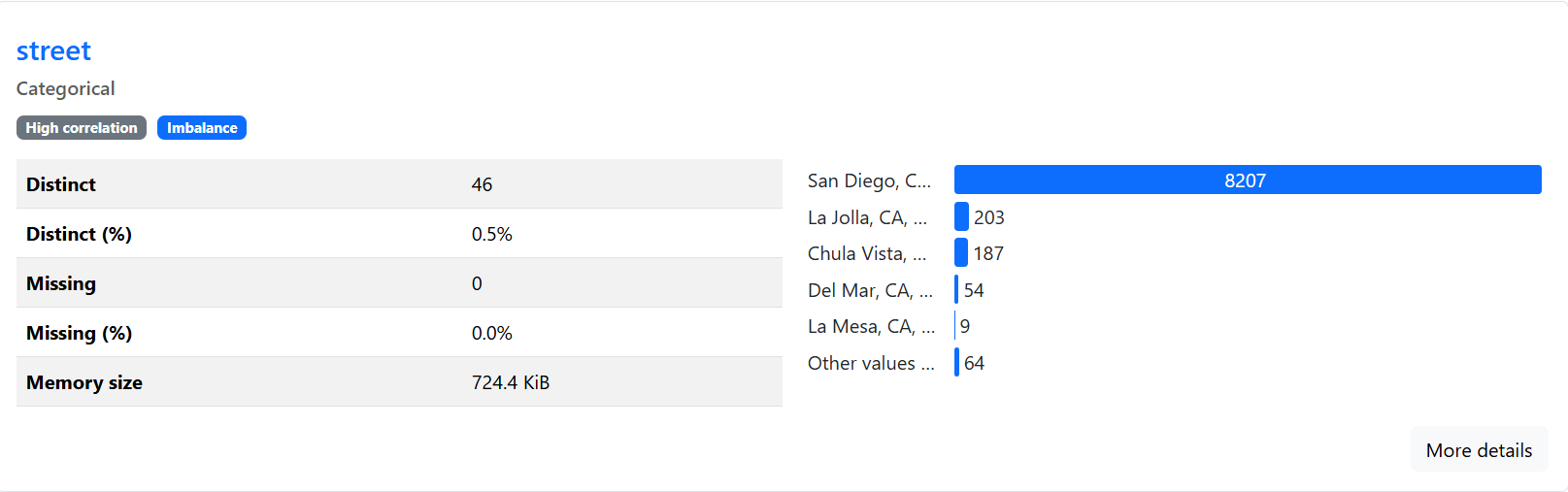
* + Column name: host\_total\_listings\_count

Observation: correlation between it and host\_listings\_count is 1

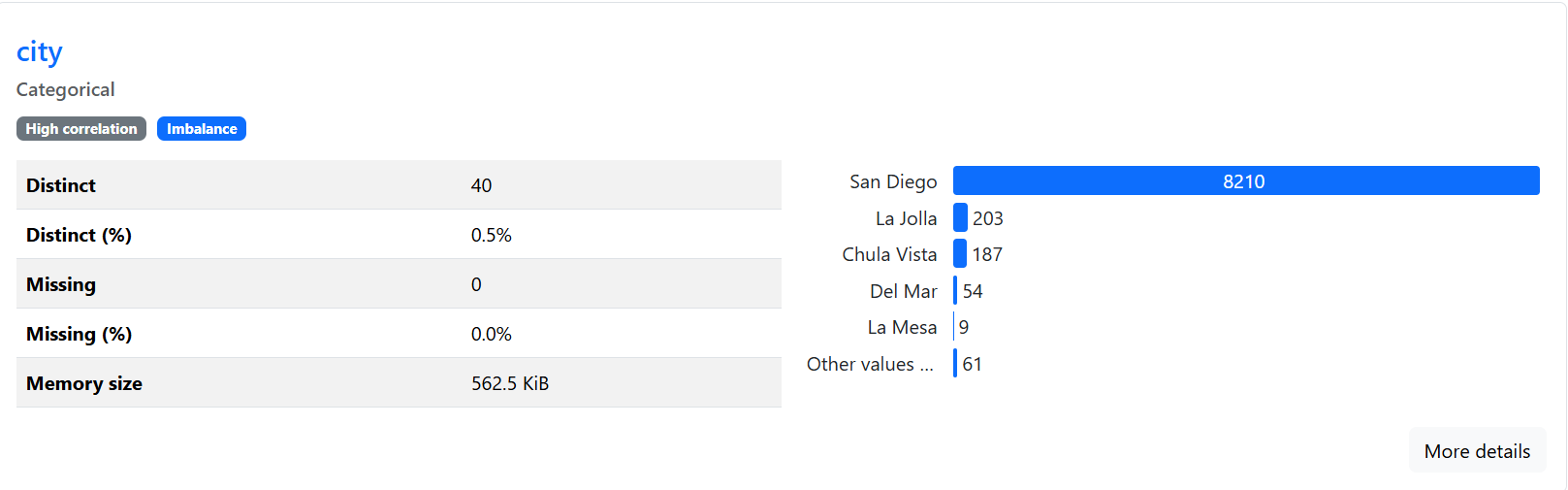
* + Column name: host\_has\_profile\_pic

Observation: all true values

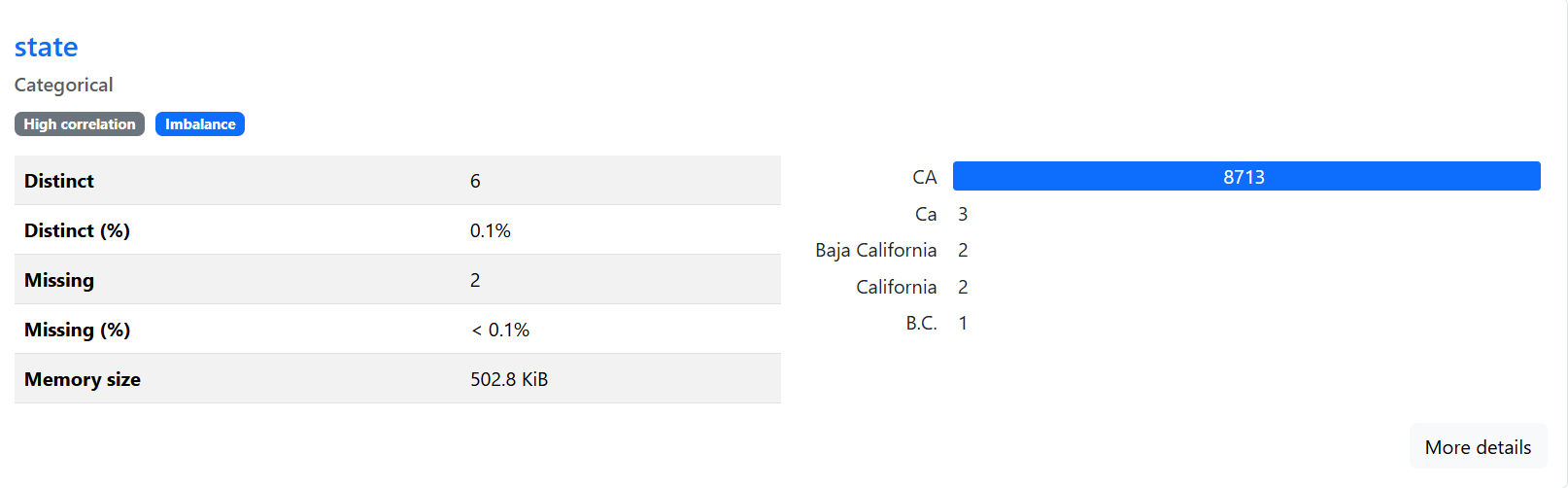
* + Column name: street

Observation: 99% is San Diego, CA, United States

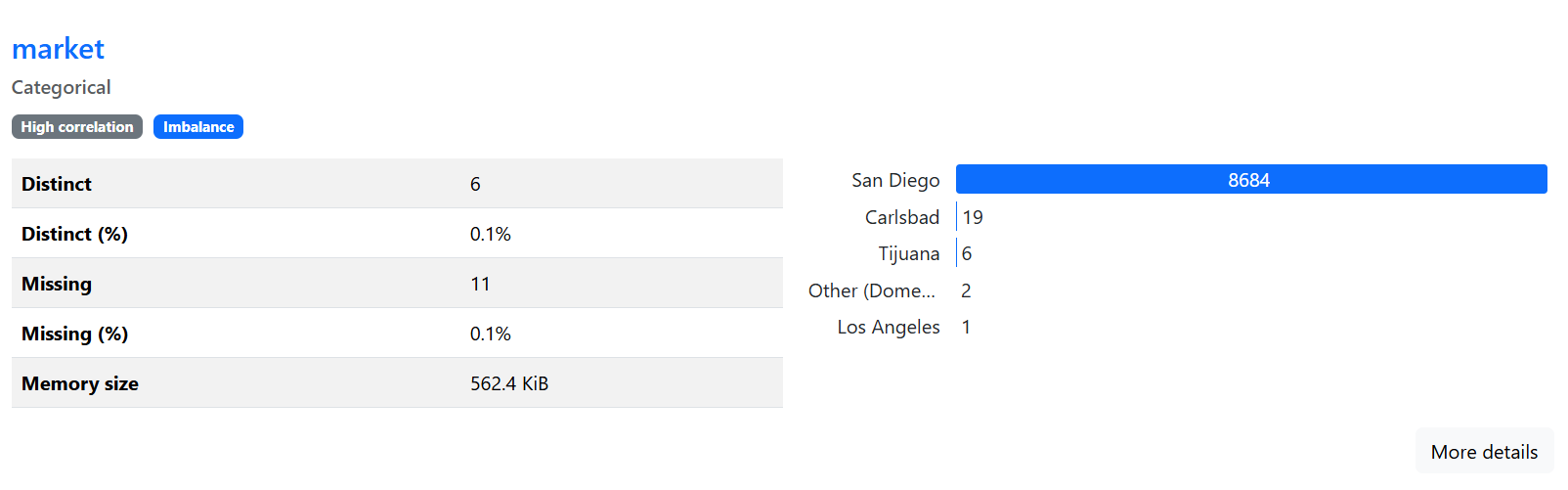
* + Column name: city

Observation: 98% is San Diego

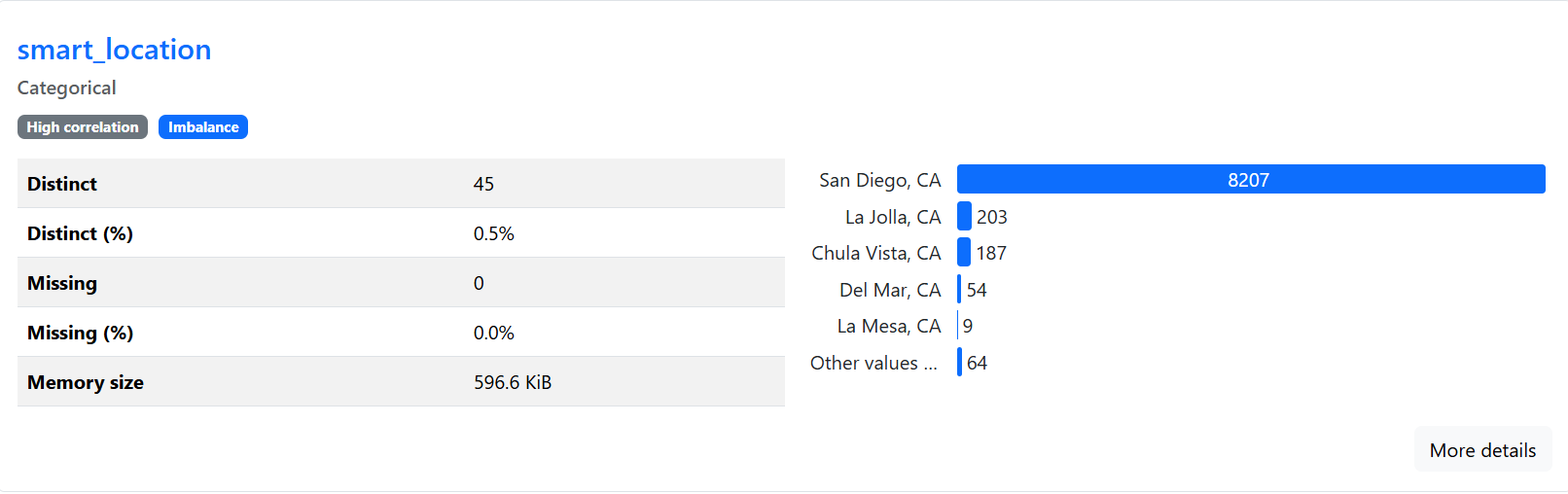
* + Column name: state

Observation: 99% is CA

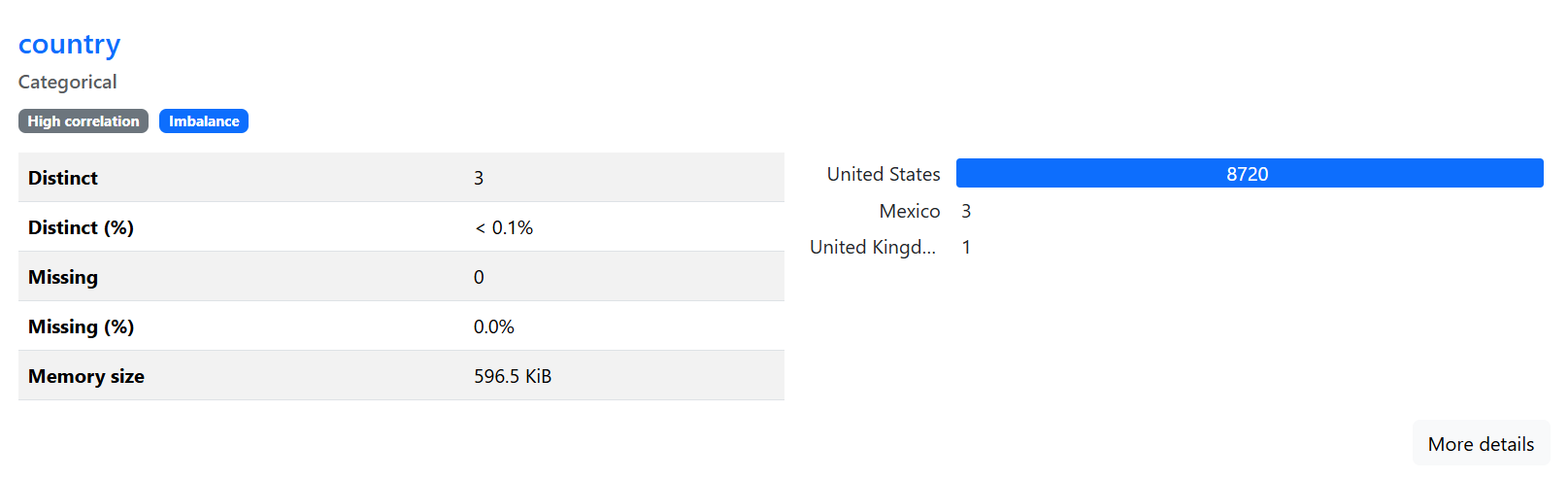
* + Column name: market

Observation: 99% is San Diego

* + Column name: smart\_location

Observation: 99% is San Diego, CA

* + Column name: country

Observation: 99% is United states

* + Column name: longitude

Observation: they are almost identical as the difference is in very small decimal

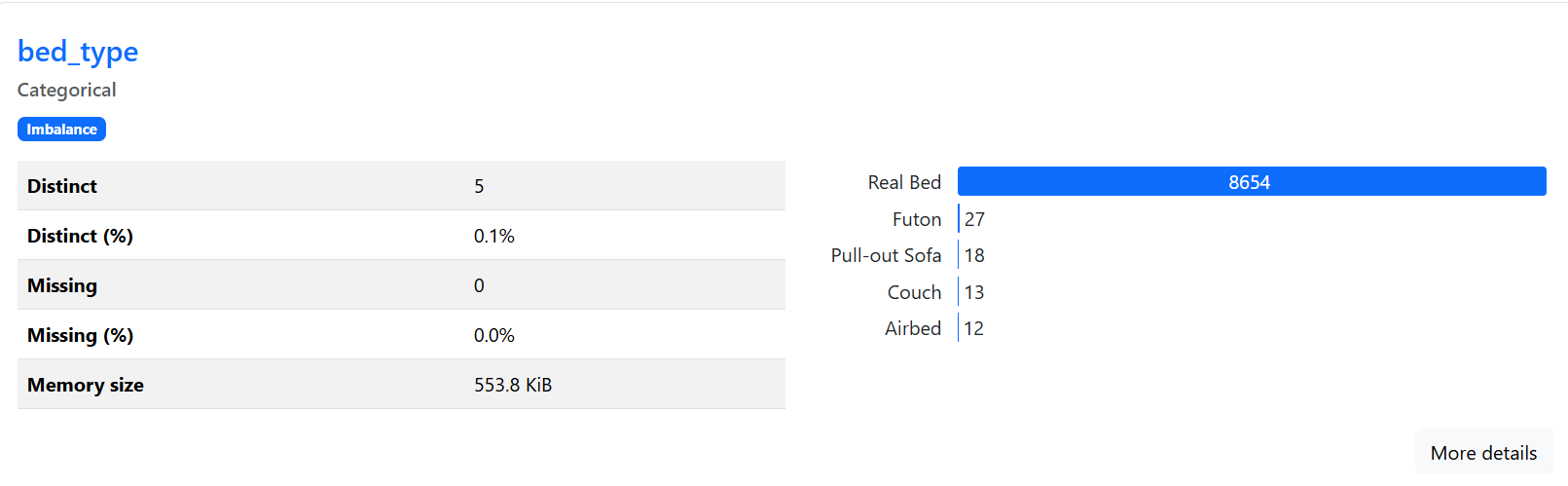


* + Column name: latitude

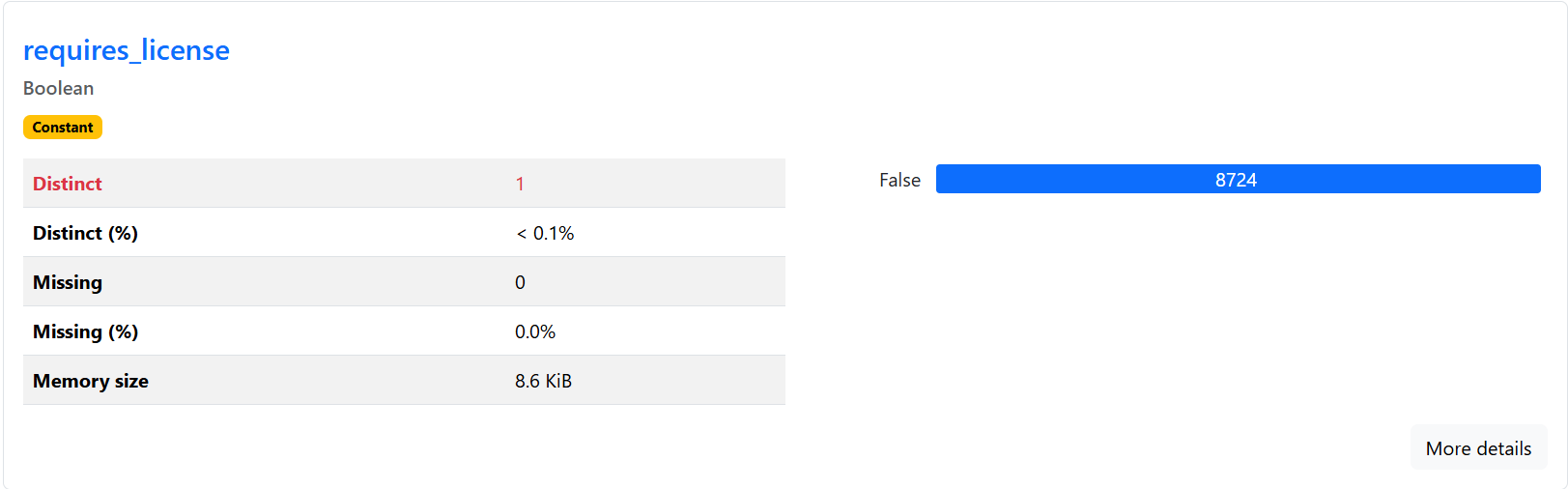
Observation: they are almost identical as the difference is in very small decimal



* + Column name: bed\_type

Observation: 99% is Real bed

* + Column name: requires\_license

Observation: 100% is false values

* + Column name: require\_guest\_profile\_picture

Observation: 97% is false

* + Column name: require\_guest\_phone\_verification

A screenshot of a computer

AI-generated content may be incorrect.Observation: 97% is false

* + Column name: neighborhood

Observation: we used neighborhood cleansed instead, as it has no missing values

* + Column name: is\_business\_travel\_ready

Observation: all false values

* + Column name: host\_name

Observation: we can use the host\_Id instead

# **Feature Engineering**

# Explanation for some features:

* **Host Response Power***:* We combine the host's response rate with the number of listings they manage. A host who responds quickly and operates multiple listings is typically more professional and experienced.
* **Host Commitment**: This feature measures a host's dedication over time. A host who has been active for a long period but has relatively few stays may indicate lower engagement.
* **Bedroom Quality**: This reflects the balance between the number of bedrooms and bathrooms. A property with too few bathrooms compared to bedrooms might make guests uncomfortable.
* **Space per Guest**: By looking at the ratio of accommodation to the number of beds, we estimate how much personal space each guest has.
* **Essential Amenities**: We tracked the availability of crucial amenities such as Wi-Fi, air conditioning, and a kitchen. Missing any of these essentials can seriously impact a guest’s stay and, in turn, their review score.
* **Review Consistency**: A property with many reviews but fluctuating scores signals inconsistency. Stable, high-quality performance is key to maintaining strong reviews, and this feature captures that trend.
* **Price Value**: Guests naturally assess whether the experience matches the price they paid. Listings that seem overpriced relative to what they offer are more likely to receive lower scores.

1. Basic information about selected features:

* From the first\_review and last\_reviewwe extracted the duration in days

***df['review\_duration\_days'] =***

***(df['last\_review'] - df['first\_review']).dt.days***

We have also extracted the duration in months, so the numbers are not confusing for the model.

***df['review\_duration\_months'] = df['review\_duration\_days'] / 30***

From this we observed that the **review\_duration\_months** is the most suitable to use, as its average is reasonable.

* From the host\_Since column we have extracted the host duration in days, months, and years from today.

***df['host\_duration\_days'] =***

***(pd.to\_datetime("today") - df['host\_since']).dt.days***

***df['host\_duration\_months'] = df['host\_duration\_days'] / 30***

***df['host\_duration\_years'] = df['host\_duration\_days'] / 365.25***

From this we observed that the **host\_duration\_years** is the most suitable to use as its average is reasonable.

* Also, from the host\_since column we have extracted the host\_duration relative to the oldest host.

***oldest\_host\_date = df['host\_since'].min()***

***df['host\_relative\_age\_days'] =***

***(df['host\_since'] - oldest\_host\_date).dt.days***

***df['host\_relative\_age\_months'] = df['host\_relative\_age\_days'] / 30***

***df['host\_relative\_age\_years'] =***

***df['host\_relative\_age\_days'] / 365.25***

In this case as well, **host\_relative\_age\_years** offered the most meaningful and interpretable representation.

* From host\_response\_rate and host\_listings\_count we have extracted host\_response\_power

***df['host\_response\_power'] =***

***df['host\_response\_rate'] \* np.log1p(df['host\_listings\_count'])***

* From bedrooms and bathrooms, we have extracted bedroom\_quality

***df['bedroom\_quality'] = df['bedrooms'] / (df['bathrooms'] + 0.5)***

* From accommodates and beds we have extracted space\_per\_guest

***df['space\_per\_guest'] = df['accommodates'] / (df['beds'] + 0.1)***

* From host\_duration\_days we have extracted seasonal\_demand

***df['seasonal\_demand'] =***

***np.sin(2 \* np.pi \* (df['host\_duration\_days'] % 365) / 365)***

* From review\_duration\_days we have extracted recent\_review\_boost

***df['recent\_review\_boost'] =***

***np.where(df['review\_duration\_days'] < 30, 1, 0.5)***

* From has\_wifi, has\_air\_conditioning and has\_kitchen we have extracted essential\_amenities.

***df['essential\_amenities'] =***

***df[['has\_wifi', 'has\_air\_conditioning', 'has\_kitchen']].sum(axis=1)***

* From has\_hot\_tub, has\_hot\_tub, has\_pool, has\_gym we have extracted luxury\_amenities.

***df['luxury\_amenities'] =***

***df[['has\_hot\_tub', 'has\_pool', 'has\_gym']].sum(axis=1)***

* From number\_of\_reviews and review\_scores\_rating we have extracted review\_consistency.

**df['review\_consistency'] =**

**df['number\_of\_reviews'] / (df['review\_scores\_rating'] + 1)**

* From review\_scores\_rating and number\_of\_reviews we have extracted positive\_momentum.

***df['positive\_momentum'] =***

***df['review\_scores\_rating'] \* np.log1p(df['number\_of\_reviews'])***

* From nightly\_price and accommodates we have extracted price\_value

***df['price\_value'] = df['nightly\_price'] / df['accommodates']***

# **Filling Missing Values**

# Normal Cases:

* Filling text columns like **( 'description', 'transit', 'access', 'interaction', 'house\_rules')** with ‘No information provided’
* Filling categorical columns like (**'host\_location', 'host\_is\_superhost', 'host\_identity\_verified')** with mode.

1. Special Cases:
   * Fill host\_neighbourhood using neighbourhood\_cleansed using group by as they have a higher percentage of missing values.

***df['host\_neighbourhood'] = df.groupby('neighbourhood\_cleansed')['host\_neighbourhood']***

***.transform(lambda x: x.fillna(x.mode()[0] if not x.mode().empty***

***else df['host\_neighbourhood'].mode()[0]))***

* Fill host\_response\_time using host\_is\_superhost and host\_response\_rate using group by as they have a higher percentage of missing values.

***df['host\_response\_time'] = df.groupby(['host\_is\_superhost', 'host\_response\_rate'])['host\_response\_time']***

***.transform(lambda x: x.fillna(x.mode()[0] if not x.mode().empty***

***else df['host\_response\_time'].mode()[0]))***

* Fill neighborhood\_overview with neighbourhood\_cleansed using group by as they have a higher percentage of missing values.

***df['neighborhood\_overview'] = df.groupby('neighbourhood\_cleansed')***

***['neighborhood\_overview'].transform( lambda x:x.fillna(x.mode()[0]***

***if not x.mode().empty***

***else "No overviewavailable"))***

* If any of them are still missing fill with overall Mode
* Fill 'host\_about’ with this formula.

***df['host\_about'] = df.apply(***

***lambda row: (***

***"is a host offering a {row['room\_type']} in {row['neighbourhood\_cleansed']}. "***

***f"Guests usually enjoy a {row['property\_type']} with great hospitality."***

***if pd.isnull(row['host\_about']) else row['host\_about']***

***), axis=1***

***)***

* Fill notes with this formula.

***df['notes'] = df.apply(***

***lambda row: (***

***f"This {row['room\_type']} in {row['neighbourhood\_cleansed']} offers a comfortable stay. "***

***f"Arrival instructions and local tips will be shared after booking."***

***if pd.isnull(row['notes']) else row['notes']***

***), axis=1***

***)***

* provides a contextual default based on the existing data, helping to maintain meaningful descriptions and improve the quality of the dataset.

# **Encoding :**

1. Binary Replaced Columns:

***df = df.replace({***

***'require\_guest\_phone\_verification': {'t': 1, 'f': 0},***

***'require\_guest\_profile\_picture': {'t': 1, 'f': 0},***

***'instant\_bookable': {'t': 1, 'f': 0},***

***'is\_location\_exact': {'t': 1, 'f': 0},***

***'host\_identity\_verified': {'t': 1, 'f': 0},***

***'host\_has\_profile\_pic': {'t': 1, 'f': 0},***

***'is\_business\_travel\_ready': {'t': 1, 'f': 0},***

***'host\_is\_superhost': {'t': 1, 'f': 0},***

***'requires\_license': {'t': 1, 'f': 0}***

***})***

This step replaces boolean-like string values ('t' for true and 'f' for false) with numerical equivalents (1 for true and 0 for false).

1. Label Encoded Categorical Columns:

***label\_cols = ['cancellation\_policy\_cleaned', 'room\_type\_cleaned', 'property\_type\_cleaned', 'host\_neighbourhood\_cleaned']***

Label encoding is applied to categorical columnsto convert their unique categories into integer labels.

For example, if room\_type\_cleaned contains values like "Entire home/apt", "Private room", and "Shared room", they will be encoded as 0, 1, and 2, respectively.

This transformation is necessary to apply **Nominal relationship** as machine learning algorithms cannot process text directly.

1. Embedding Layer for Host Neighbourhood:

An embedding layer is created to represent the host neighborhoods in a dense vector space. This is particularly useful in deep learning models where categorical variables need to be represented in a meaningful way.

* Parameters : input\_dim: Total number of unique neighborhoods (num\_neighborhoods). output\_dim: Dimensionality of the embedding vectors (set to 8 in this case).

The host\_neighbourhood\_cleaned column is converted to a TensorFlow tensor (neighbourhood\_tensor) and passed through the embedding layer to generate dense embeddings.

1. Mapping Ordinal Categories:

***mapping = {***

***'within an hour': 1,***

***'within a few hours': 2,***

***'within a day': 3,***

***'a few days or more': 4***

***}***

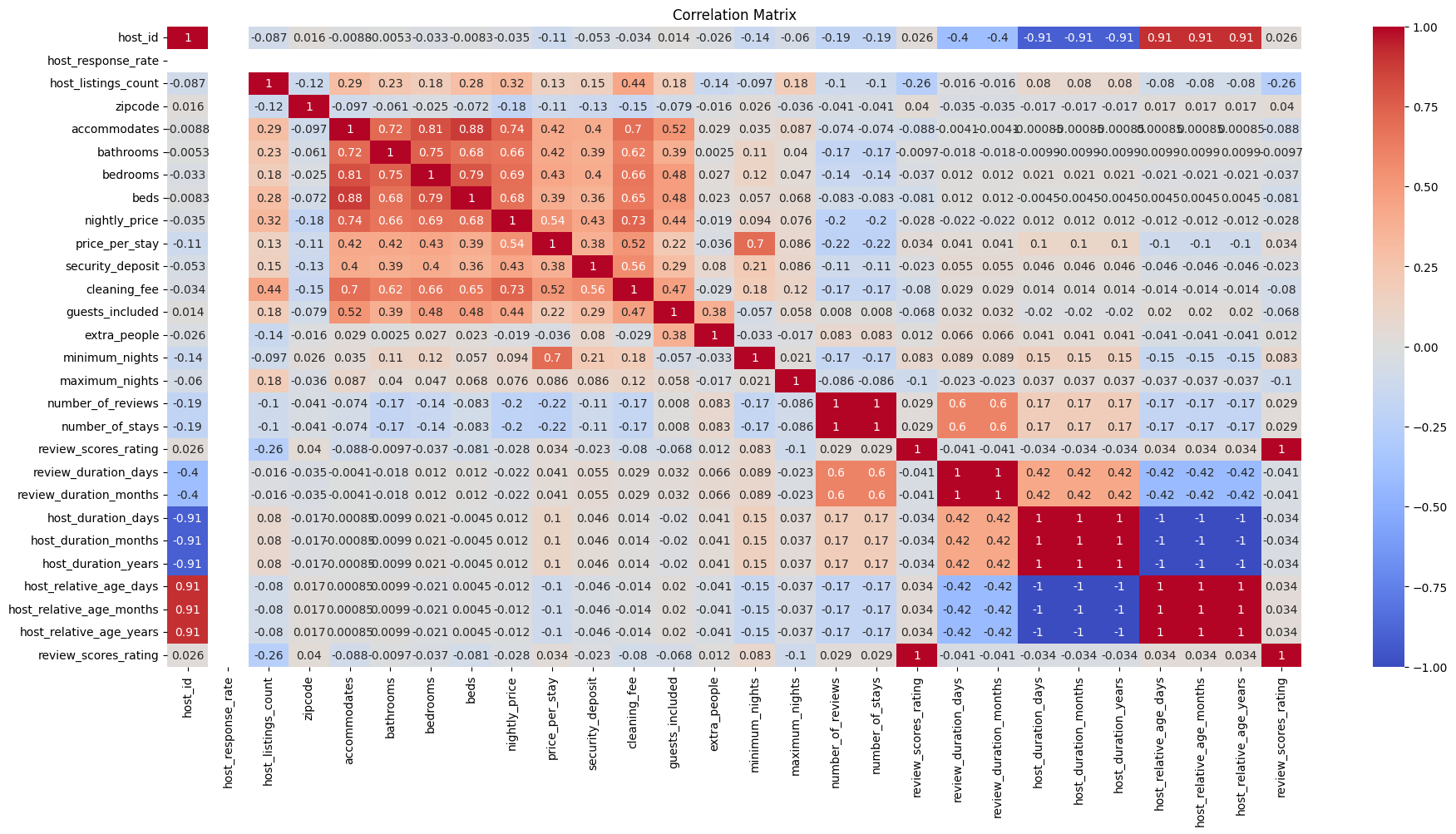
***df['host\_response\_time'] = df['host\_response\_time'].map(mapping)***

This mapping preserves the **ordinal relationship** between the categories, ensuring that the numerical representation reflects their logical order.

# **Milestone 1 Feature selection:**

1. Correlation Analysis :

* To identify relationships between numerical features and the target variable (review\_scores\_rating)
* Features with a correlation greater than 0.1 with the target variable are selected for further analysis.
* **Selected columns: ['host\_listings\_count', 'accommodates', 'beds', 'cleaning\_fee', 'minimum\_nights', 'maximum\_nights', 'review\_scores\_rating'].**



1. Missing Value Imputation (ANOVA):

Missing values in numerical columns are imputed using the KNN algorithm, which estimates missing values based on the nearest neighbors.

1. primary selection:

***X = df[Numerical\_cols]***

***y = df['review\_scores\_rating']***

***selector = SelectKBest(score\_func=mutual\_info\_regression, k=10)***

***X\_new = selector.fit\_transform(X, y)***

***selected\_features = X.columns[selector.get\_support()]***

***print("Selected Top 10 Features:")***

***for feature in selected\_features:***

***print("-", feature)***

SelectKBest with mutual\_info\_regression is used to select the top ten features based on their relevance to the target variable.

1. Text Feature Extraction:

Textual columns are vectorized using TfidfVectorizer to convert them into numerical representations.

The resulting sparse matrices are combined into a single feature matrix.

1. Feature Selection for Text Features:

Identify the most important text-based features using ANOVA F-test.

SelectKBest with f\_classif is applied to select the top 100 text features.

The importance of each feature is evaluated using F-scores and p-values.

Output :

Source columns contributing the most selected features are identified.

Example: host\_about\_cleaned, house\_rules\_cleaned, etc.

1. Interaction Features:

Create new features by combining existing numerical and binary features to capture interactions.

***listing\_features\_df['accommodates\_per\_bed'] = listing\_features\_df['accommodates'] / (listing\_features\_df['beds'] + 1e-9)***

***listing\_features\_df['has\_kitchen\_and\_iron'] = (listing\_features\_df['has\_kitchen'] == 1) &***

***(listing\_features\_df['has\_iron'] == 1)***

***listing\_features\_df['has\_kitchen\_and\_laptop\_workspace'] = (listing\_features\_df['has\_kitchen'] == 1) & (listing\_features\_df['has\_laptop\_friendly\_workspace'] == 1)***

1. Polynomial Features:

Generate polynomial combinations of numerical features to capture non-linear relationships.

Explanation :

Polynomial features up to degree 2 are generated for numerical columns.

These features are added to the main DataFrame for further modeling.

1. Encoding Categorical Variables:

Categorical columns are encoded using LabelEncoder to prepare them for machine learning models.

1. Aggregating Features:

{ Group-level statistics (e.g., average cleaning fee per property type) are calculated and added as new features. }

***avg\_cleaning\_fee\_by\_property\_type = listing\_features\_df.groupby('property\_type\_cleaned')***

***['cleaning\_fee'].transform('mean')***

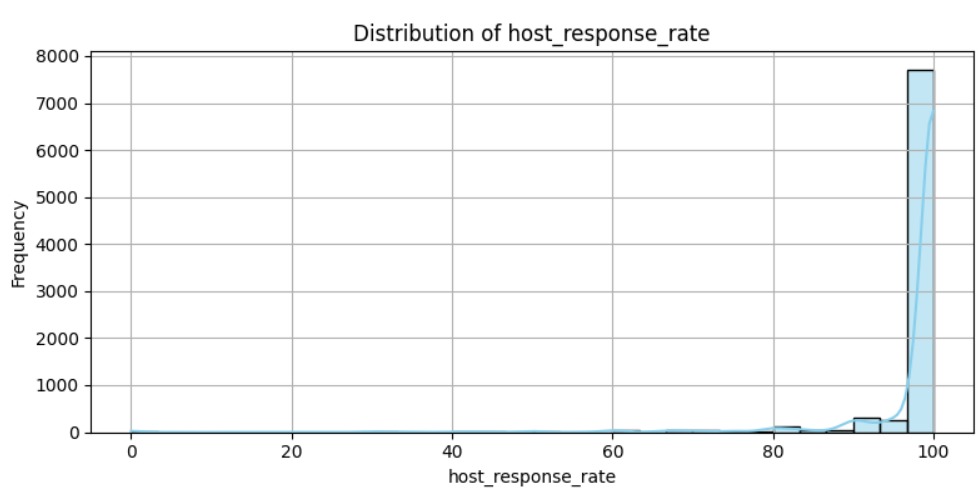
***listing\_features\_df['avg\_cleaning\_fee\_per\_property\_type'] = avg\_cleaning\_fee\_by\_property\_type.iloc[:, 0****]*

# **Visualizations**

1. Distributions:

### host\_response\_rate

**Distribution:** Mostly centered near 100%.  
**Outliers:** Some hosts have lower response rates (~0–40%).  
**Skewness:** Left-skewed (more high values).  
**Interpretation:** Most hosts respond quickly.



### 

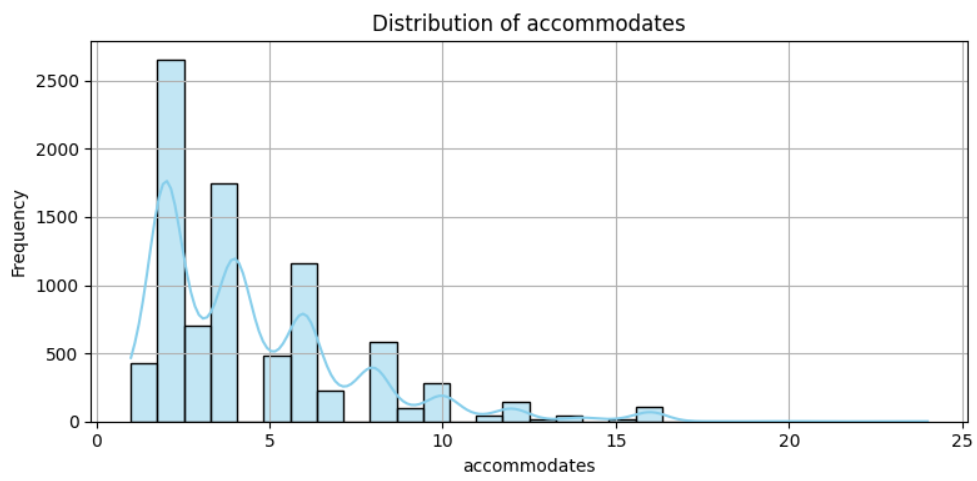
### host\_listings\_count

**Distribution:**  Right-skewed (most hosts have few listings).  
**Outliers:** Hosts with a very high number of listings (superhosts)  
**Skewness:** Right-skewed.  
**Interpretation:** Majority of hosts manage few listings, but a small number manage many.

### 

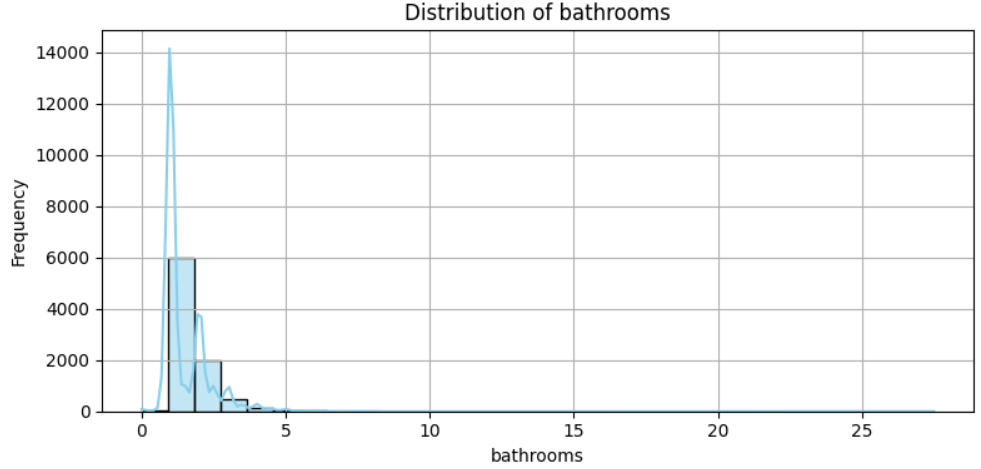
### accommodates

**Distribution:**  Peaks around 2–4 guests.  
**Outliers:** Listings for very large groups (>10 guests).  
**Skewness:** Right-skewed.  
**Interpretation:** Most listings accommodate small groups; large group accommodations are rare but could be premium offering



### bathrooms

**Distribution:**  Peaks at 1 bathroom.  
**Outliers:** Listings with many bathrooms (> 4).  
**Skewness:** Right-skewed.  
**Interpretation:** Typical listings have 1–2 bathrooms; luxury listings have more.



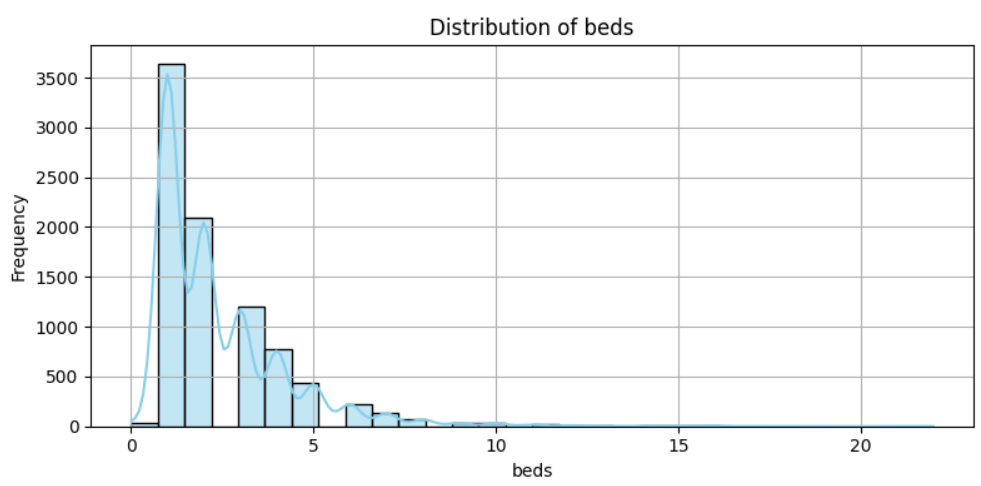
### bedrooms

**Distribution:**  Peaks at 1–2 bedrooms.  
**Outliers:** Listings with >5 bedrooms.  
**Skewness:** Right-skewed.  
**Interpretation:** Most properties are small apartments or houses; big listings are rare and more likely to be luxury.

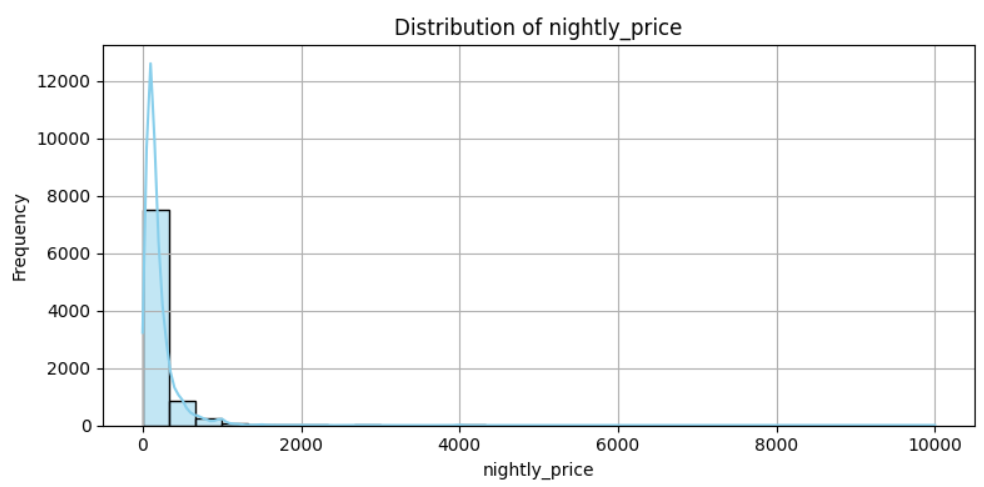
### 

### 

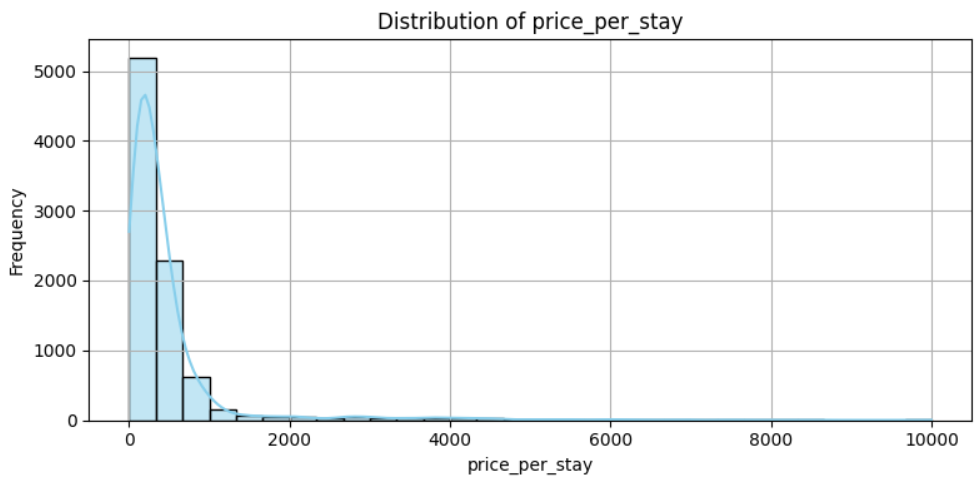
### beds

**Distribution:**  Centered around 1–3 beds.  
**Outliers:** Listings with > 5 beds.  
**Skewness:** Right-skewed.  
**Interpretation:** Listings generally offer 1–3 beds; large-capacity listings are less frequent.

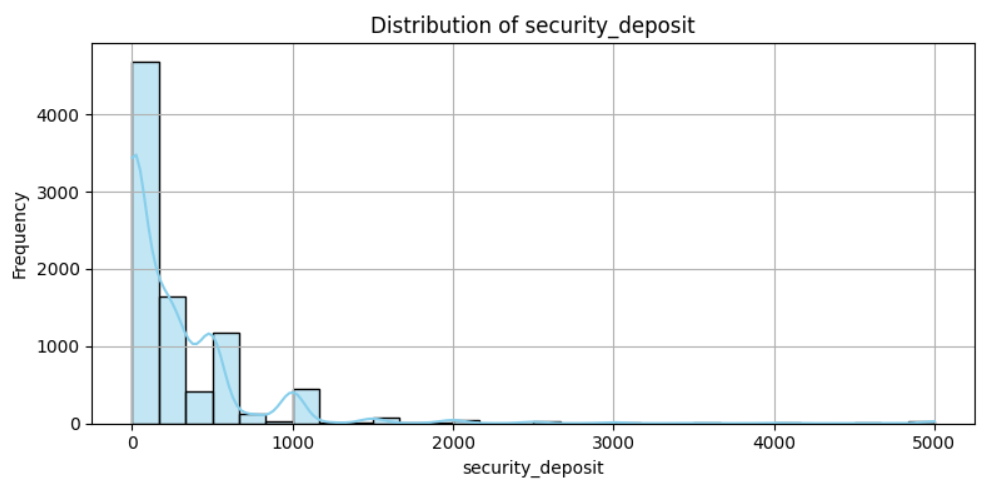
### nightly\_price

**Distribution:**  Highly right-skewed.  
**Outliers:** Listings priced extremely high per night.  
**Skewness:** Right-skewed.  
**Interpretation:** Most properties are affordable, but luxury properties (outliers) are priced much higher.

### price\_per\_stay

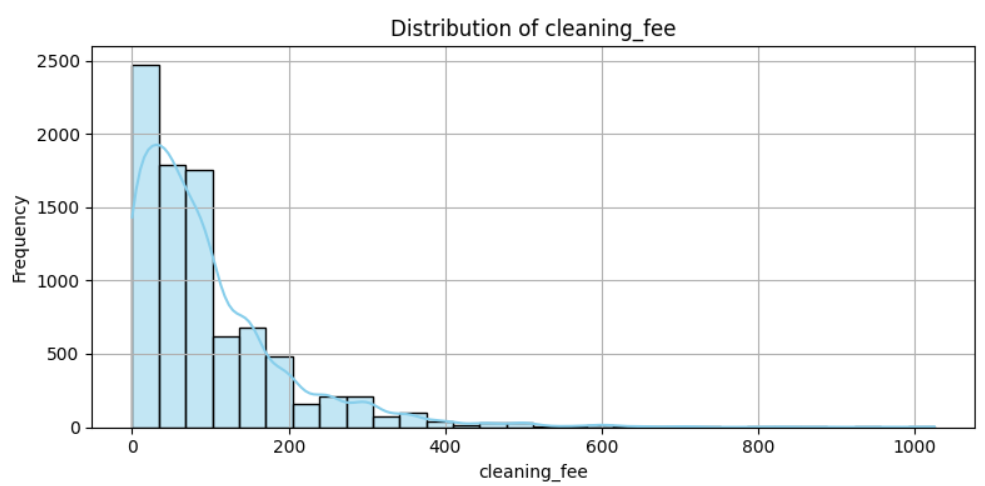
**Distribution:**  Right-skewed.  
**Outliers:** Extremely high total stay prices.  
**Skewness:** Right-skewed.  
**Interpretation:** Reflects nightly price multiplied by the stay duration. Highlights how longer stays or luxury accommodations inflate total prices.

### security\_deposit

**Distribution:**  Mostly near zero, with some higher values.  
**Outliers:** Listings requiring very high deposits.  
**Skewness:** Strong right-skewed  
**Interpretation:** Most listings require little or no deposit; some high-end listings ask for larger security deposits.

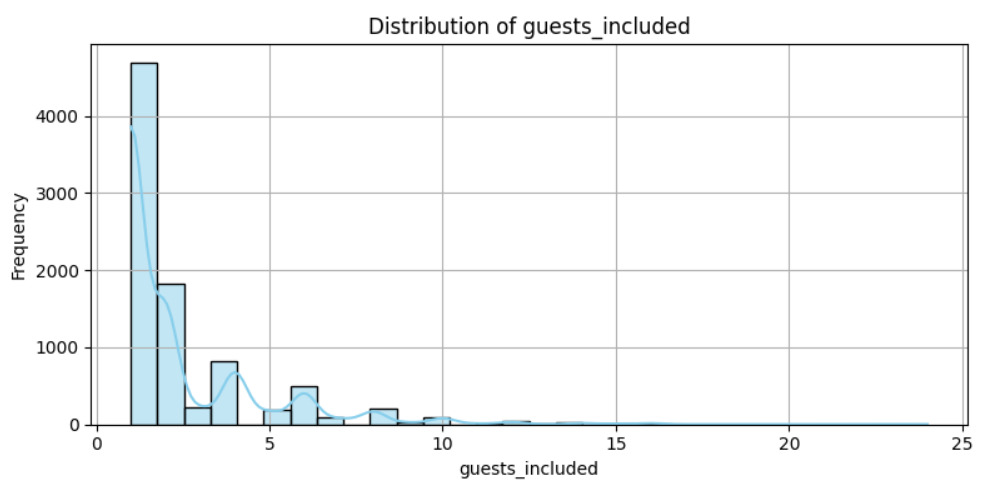
### cleaning\_fee

**Distribution:**  Right-skewed.  
**Outliers:** Some listings have very high cleaning fees.  
**Skewness:** Right-skewed.  
**Interpretation:** Most cleaning fees are modest; luxury listings or those with complex properties may charge high fees.

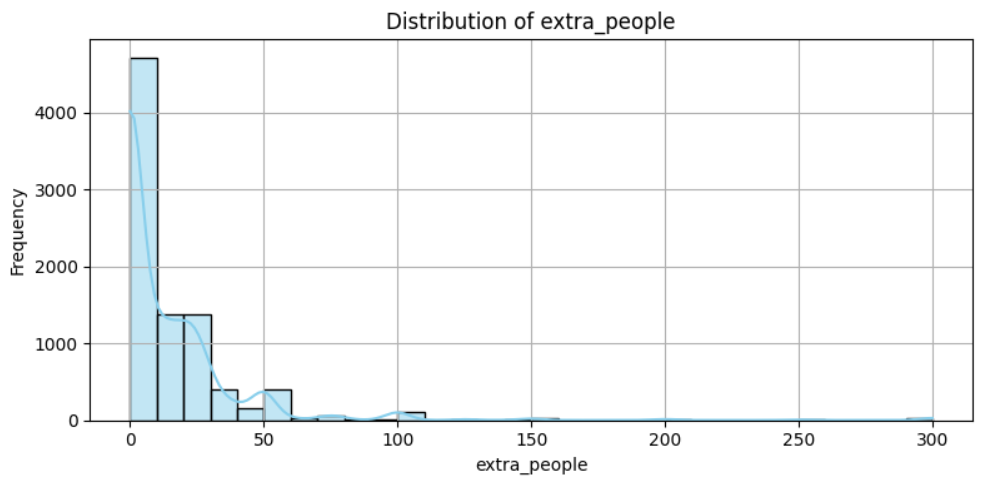


### guests\_included

**Distribution:**  Peak at 1–2 guests.  
**Outliers:** Listings allowing many guests included at base price.  
**Skewness:** Right-skewed.  
**Interpretation:** Most listings include a few guests before additional charges apply

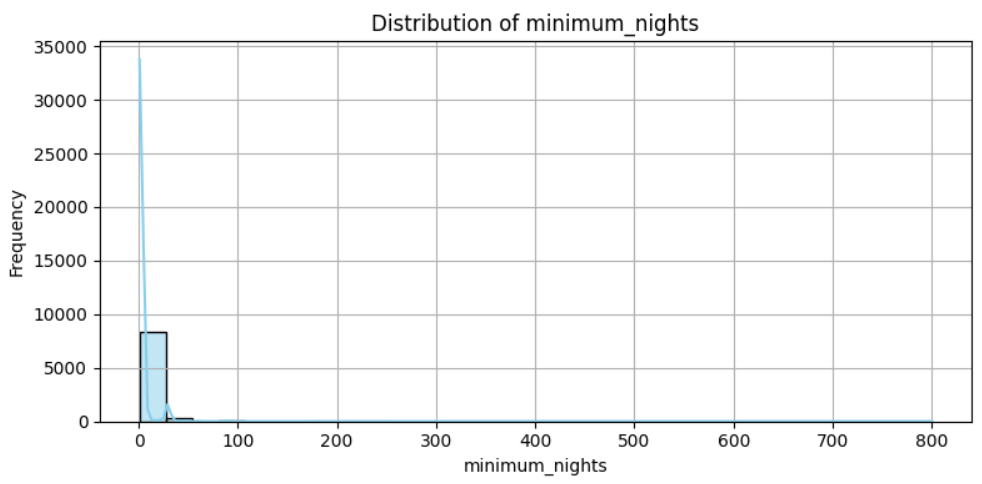


### extra\_people

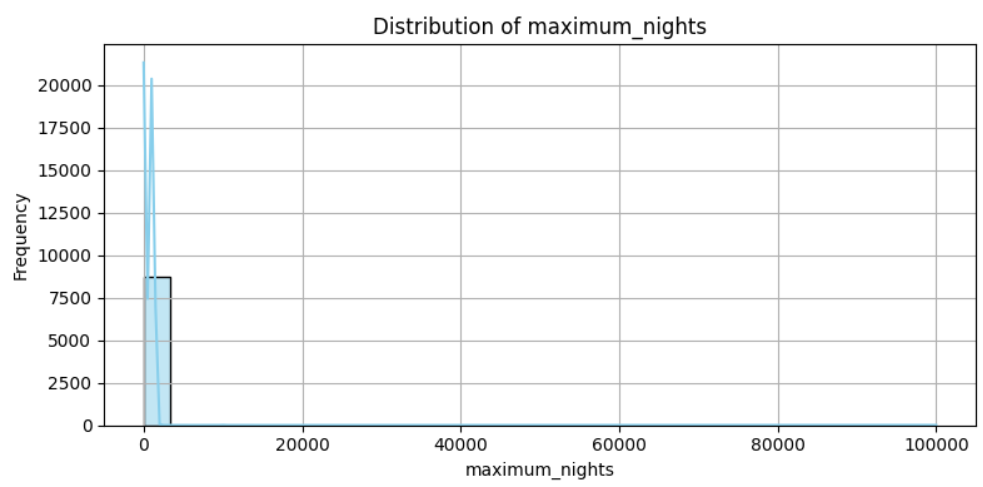
**Distribution:**  Most listings charge little to nothing extra; right-skewed.  
**Outliers:**  Some listings charge very high extra fees per guest.  
**Skewness:** Right-skewed.  
**Interpretation:** Extra people fees are generally low but can be significant for larger groups.

### minimum\_nights

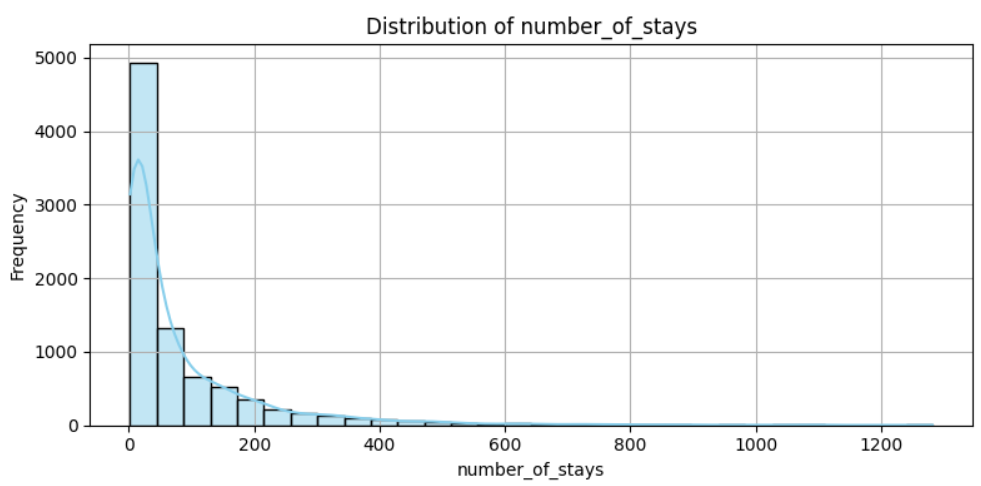
**Distribution:**  Peaks at 1 night minimum.  
**Outliers:**  Peaks at 1 night minimum.  
**Skewness:** Right-skewed.  
**Interpretation:** Most listings allow short stays; some require longer minimums, often for regulatory or preference reasons.



### maximum\_nights

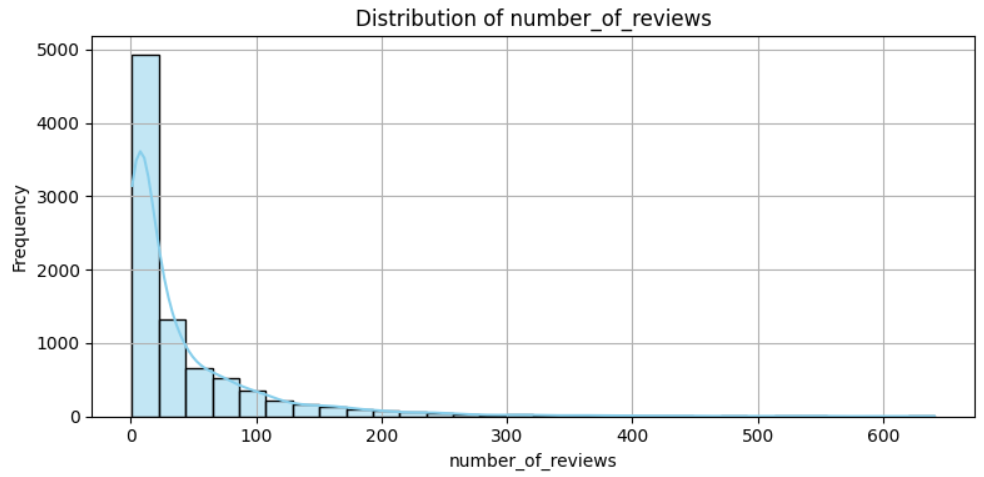
**Distribution:**  Very wide spread; many very high maximum values.  
**Outliers:**  Maximum nights set to 365+, 9999+, etc.  
**Skewness:** Extremely right-skewed.  
**Interpretation:** Some listings set very high maximums as defaults; usually not meaningful.

### number\_of\_stays

**Distribution:**  Similar to reviews, peaks at lower numbers.  
**Outliers:**  Listings with very high numbers of stays.  
**Skewness:** Right-skewed.  
**Interpretation:** Only a few listings have very high turnover.

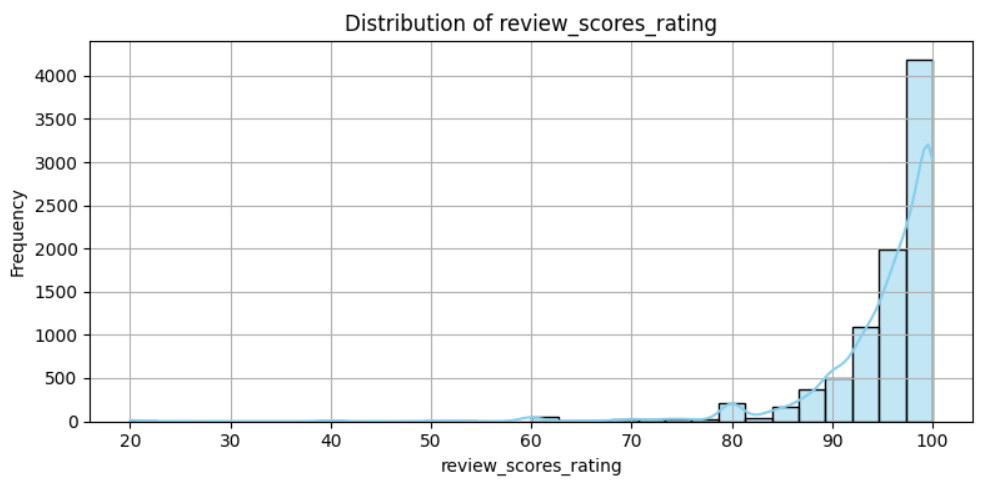
### number\_of\_reviews

**Distribution:**  Peaks at low review counts.  
**Outliers:**  Listings with hundreds of reviews.  
**Skewness:** Right-skewed.  
**Interpretation:** Most listings have few reviews; some very popular listings have accumulated hundreds.



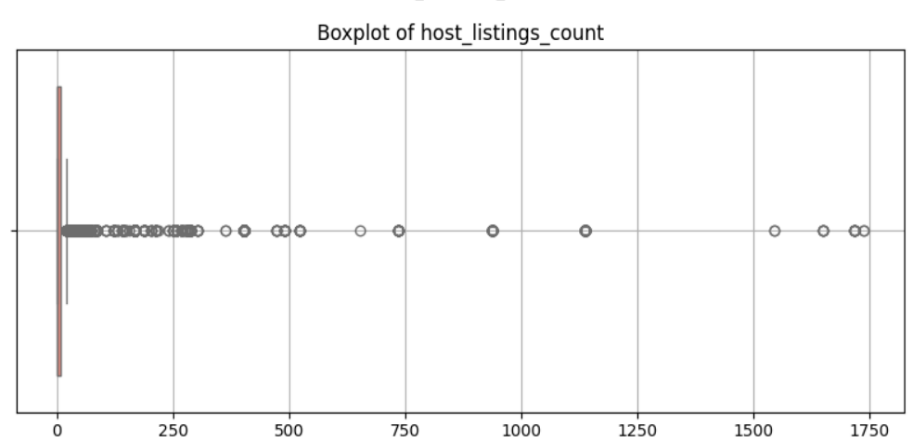
### review\_scores\_rating

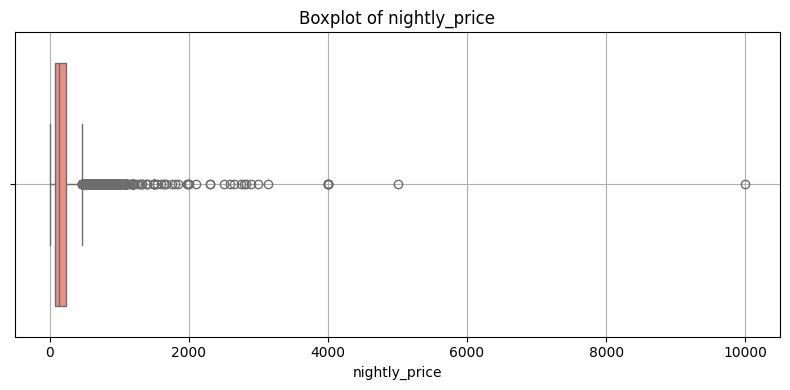
**Distribution:**  Heavily clustered around 4–5 stars (80–100).  
**Outliers:**  Very low scores (rare).  
**Skewness:** Left-skewed (most ratings are high).  
**Interpretation:** Airbnb reviews are biased positively; users mostly leave high ratings.

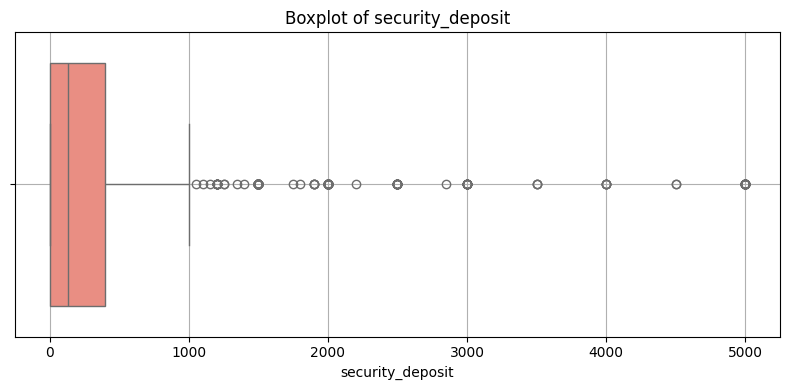


1. Boxplots :

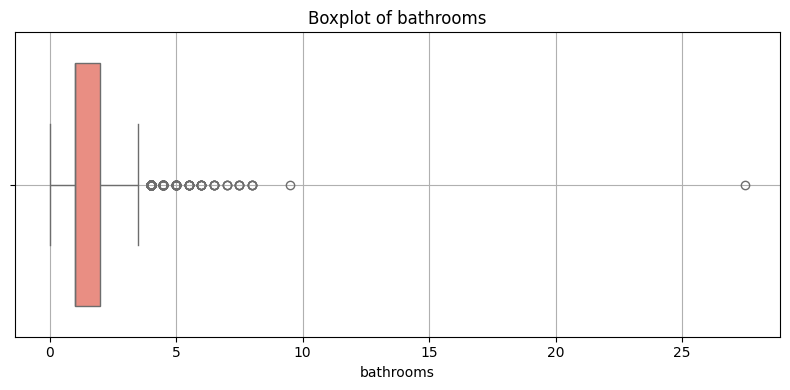
* Features such as **host\_listings\_count, nightly\_price,** and **security\_deposit** showed significant presence of outliers, reflecting natural variability in hosts and listing types.

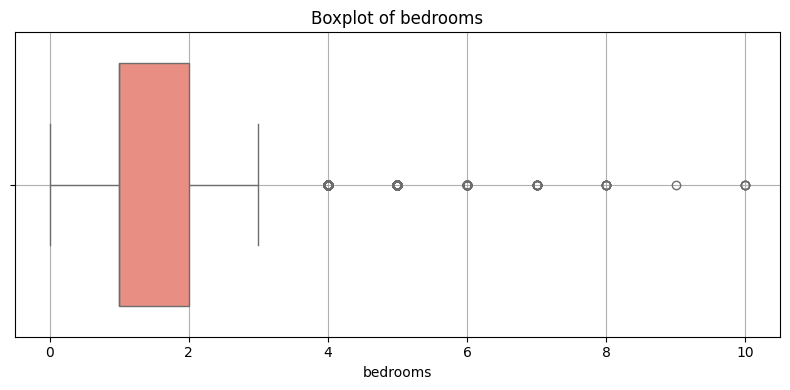


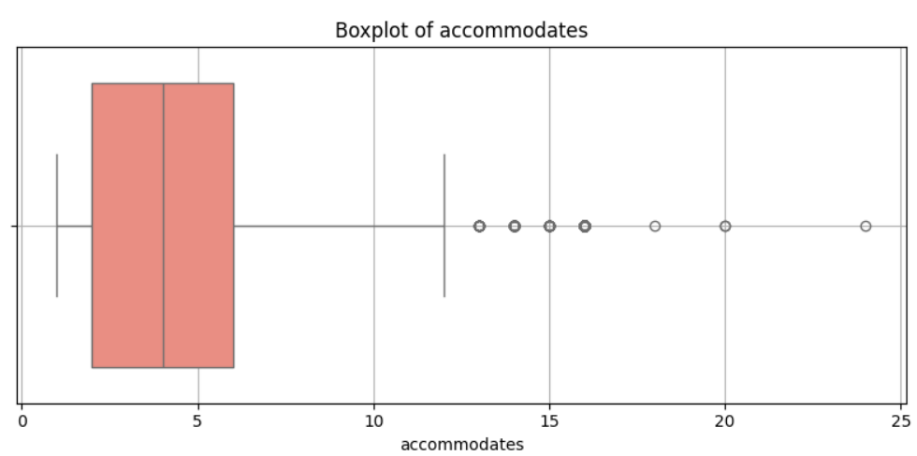




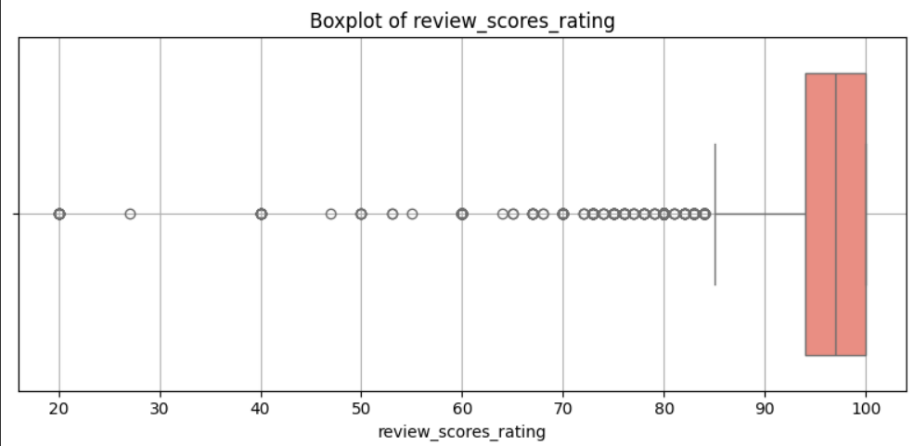
* Several features, including **accommodates, bedrooms,** and **bathrooms**, exhibited slight right-skewness, indicating a concentration of smaller properties with a few larger ones.







* Fields like **review\_scores\_rating** showed high clustering toward positive values, confirming user bias toward higher ratings.



# 

# Handling outliers

* From the previous visualizations we can see that there are outliers in a lot of columns.
* We used the Interquartile Range (IQR) method to detect and cap outliers across numerical features. Capping outliers, instead of removing them, allowed us to maintain the integrity of the dataset while reducing the skewing effect of extreme values.

|  |  |  |
| --- | --- | --- |
| **Column** | **Type of Outliers** | **Comments** |
| host\_response\_rate | Low values (close to 0%) | Some hosts barely respond. |
| host\_listings\_count | Very high values (100+ listings) | Power users like companies. |
| accommodates | Very large values (>10 guests) | Big properties are rare. |
| bathrooms | Listings with 4+ bathrooms | Luxury/villa properties. |
| bedrooms | Listings with many bedrooms (>5) | Large homes or villas. |
| beds | Listings offering many beds | Dormitory-style listings. |
| nightly\_price | Very expensive nightly rates | Luxury or very expensive listings. |
| price\_per\_stay | Extremely high total prices | Long stays + high price. |
| security\_deposit | Very large deposits | Luxury properties. |
| cleaning\_fee | Very high cleaning fees | Big villas or luxury properties. |
| minimum\_nights | Listings requiring long minimum stays | 30 days+, very rare. |
| maximum\_nights | Extremely high values (9999 days etc.) | Defaults or rare configurations. |
| number\_of\_reviews | Properties with hundreds of reviews | Very popular listings. |
| number\_of\_stays | Same as above, high booking counts. |  |

# **Text Analysis & Preprocessing**

1. **Amenities Parsing and Categorization:**
   * Each listing had a messy amenities field.
   * We cleaned this field by removing curly brackets and quotes, splitting it into a simple list.
   * Then, we mapped each amenity to logical categories like "Basic Necessities," "Comfort/Convenience," "Technology," etc.
   * For example, if a listing had "Wi-Fi" and "Heating," we categorized them under "Technology" and "Basic Necessities."
2. **Creating Binary Features for Amenities:**

* We have created new features that indicate whether each amenity is available or not.
* If a listing has “Wi-Fi,” a new column, has\_wifi will have a 1. Otherwise, it will be 0.
* This turned the messy text data into clean, machine-readable columns.

1. **Text Column Identification and Cleaning:**

* We scanned the entire dataset to find real text columns, not numbers.

1. **We filtered based on:**

* Enough text length
* Enough variation
* Not being numeric

1. **Then cleaned these text fields:**

* Lowercased all words
* Removed punctuation
* Removed stop words like "the" and "is"
* Lemmatized words (e.g., "running" becomes "run")

1. **Dropping Unnecessary Columns:**

* We removed the original messy text columns after cleaning them.
* Exception: host\_response\_time was preserved to introducing null values after encoding

1. **Analysis of the Importance of Text Fields:**

* We calculated how important each cleaned text field is for predicting review\_scores\_rating.
* We used: 1. TF-IDF to vectorize words

1. Variance and correlation with the target to measure influence
2. Higher scores mean the text is more informative for predicting good or bad reviews.
3. **A graph with different colored bars

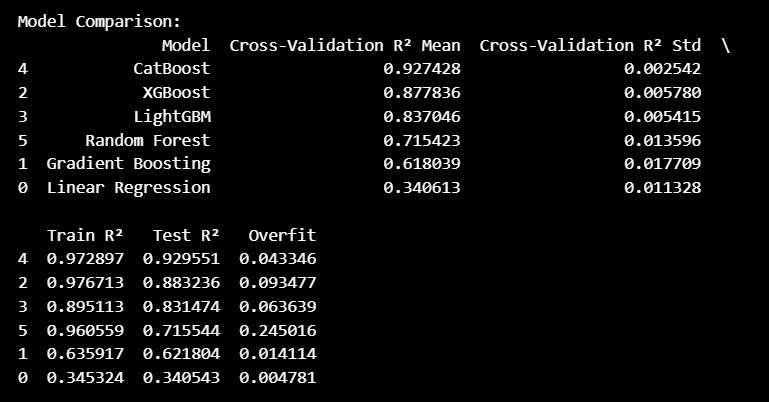
   AI-generated content may be incorrect.Visualizing Importance Scores:**

1. **Host Location Analysis**

* We checked how many hosts exactly listed their location as "San Diego, California, United States."
* We cleaned the location text first.

# **Regression Model Trials with Different Selected Features**

* For this set of features, we selected variables such as **[host\_response\_rate, host\_is\_superhost, host\_listings\_count, accommodates, bathrooms, bedrooms, beds, number\_of\_reviews, number\_of\_stays, review\_duration\_days, host\_duration\_days, instant\_bookable, and host\_identity\_verified]**.These features were chosen based on their strong correlation with the target variable (review\_scores\_rating).
* We prioritized variables that demonstrated the clearest and most consistent relationships with guest satisfaction, aiming to capture the key factors that directly impact review scores.
* We use these features **[host\_response\_power, host\_commitment, bedroom\_quality, space\_per\_guest, essential\_amenities, review\_consistency]** from feature engineering to make model train better and prevent overfitting.
* The features **room\_type\_cleaned** and **cancellation\_policy\_cleaned** were selected based on **ANOVA** tests, as they demonstrated the most significant impact on review scores.
* We also used a Pipeline to automate preprocessing and modeling. Numerical features were imputed with the median and scaled, while categorical features were imputed with the most frequent value and one-hot encoded. We trained a Gradient Boosting model with early stopping to prevent overfitting. The model’s performance was evaluated using 5-fold cross-validation and a train-test split.We used that because the train-test R^2 was too low and by using Pipeline we improved the model accuracy.
* The following results were obtained by training and evaluating the selected features using various models, such as **Linear Regression, Gradient Boosting, XGBoost, LightGBM, CatBoost, and Random Forest**.



# **Some Graphs for Each Regression Model Performance**

A graph with red squares

AI-generated content may be incorrect.A graph with green squares

AI-generated content may be incorrect.A screenshot of a graph

AI-generated content may be incorrect.

# A group of text boxes AI-generated content may be incorrect.

# **Model Performance Conclusion**

After evaluating all models through cross-validation and comparing their training and testing R² scores, a few key insights emerged:

* **CatBoost** delivered the best overall performance, achieving the highest cross-validation R² score (0.8867) and the strongest test R² (0.8922) with minimal overfitting. This suggests it generalizes very well to unseen data.
* **XGBoost** and **LightGBM** also performed strongly, with high R² values on both training and testing sets.
* **Random Forest** performed decently, but the overfitting was much higher (Train R²: 0.9605 vs. Test R²: 0.7159), meaning it captured the training patterns very well but struggled more on new data.
* **Linear Regression** had moderate performances, with low overfitting but relatively low R² scores, meaning they may not be capturing the complexity of the relationships well enough.
* **In summary:  
  CatBoost** was the best model, closely followed by **XGBoost** and **LightGBM**. These tree-based ensemble models handle the complexity of the dataset better than linear or simpler models, while maintaining a good balance between bias and variance.

# **Classification Feature Selection**

* **Chi-square**

We selected categorical features and applied the Chi-square test to evaluate their dependency with the target variable. After encoding both features (via one-hot encoding) and the target (via label encoding), we ranked the features by their Chi-square scores. The top features were selected for further analysis/modeling based on their statistical significance.

**Selected features** = ['cancellation\_policy', 'host\_about', 'host\_is\_superhost', 'house\_rules', 'neighbourhood\_cleansed', 'property\_type', 'transit']

* **ANOVA**

We applied the ANOVA F-test to identify numerical features that have a statistically significant relationship with the target variable. After cleaning the data, imputing missing values, and removing constant columns, we computed F-statistics and p-values for each feature. Features were then ranked, and the top 10 were selected. Those with p-values less than 0.05 were considered statistically significant and are prioritized for inclusion in the final model.

**Selected features** = [ 'host\_listings\_count', 'number\_of\_stays', 'number\_of\_reviews', 'has\_lock\_on\_bedroom\_door', 'has\_hot\_water', 'maximum\_nights','minimum\_nights','has\_laptop\_friendly\_workspace',

'accommodates', 'has\_air\_conditioning']

# **Classification Model Trial with Different Selected Features**

* **Selected Features:**

numeric\_features = [

'host\_response\_rate', 'host\_is\_superhost', 'accommodates',

'bathrooms', 'bedrooms', 'beds', 'nightly\_price',

'cleaning\_fee', 'number\_of\_reviews', 'minimum\_nights',

'host\_listings\_count', 'number\_of\_stays', 'security\_deposit',

'extra\_people', 'maximum\_nights', 'guests\_included'

]

categorical\_features = [

'neighbourhood\_cleansed', 'instant\_bookable',

'cancellation\_policy', 'property\_type', 'room\_type'

]

amenity\_features = [

'has\_wifi', 'has\_air\_conditioning', 'has\_kitchen',

'has\_heating', 'has\_tv', 'has\_free\_parking\_on\_premises',

'has\_iron', 'has\_laptop\_friendly\_workspace'

]

* **Models and Hyper parameters**

| **Model** | **Hyperparameters** | **Train Acc** | **CV Acc** | **Test Acc** | **Train Time (s)** | **Test Time (s)** |
| --- | --- | --- | --- | --- | --- | --- |
| **Random Forest** | max\_depth=20 | 0.9943 | 0.5971 | 0.6126 | 5.50 | 0.07 |
|  | min\_samples\_split=10 | 0.9069 | 0.5979 | 0.6166 | 4.12 | 0.08 |
|  | n\_estimators=300 | 0.9990 | 0.5958 | 0.6080 | 22.09 | 0.21 |
| **XGBoost** | learning\_rate=0.1 | 0.7444 | 0.5908 | 0.6201 | 0.79 | 0.04 |
|  | max\_depth=3 | 0.6694 | 0.5903 | 0.6189 | 0.38 | 0.03 |
|  | learning\_rate=0.01 | 0.6429 | 0.5928 | 0.6069 | 1.01 | 0.04 |
| **Gradient Boosting** | n\_estimators=100 | 0.6474 | 0.5985 | 0.6086 | 5.52 | 0.03 |
|  | max\_depth=3 | 0.6474 | 0.5985 | 0.6086 | 6.23 | 0.05 |
|  | n\_estimators=200 | 0.6756 | 0.5923 | 0.6052 | 11.42 | 0.07 |
| **Logistic Regression** | C=0.1 | 0.5872 | 0.5743 | 0.5782 | 0.51 | 0.02 |
|  | max\_iter=100 | 0.5934 | 0.5727 | 0.5765 | 0.49 | 0.02 |
|  | solver=lbfgs | 0.5934 | 0.5727 | 0.5765 | 1.13 | 0.02 |
| **SVM** | C=1.0 | 0.6275 | 0.5918 | 0.5960 | 39.45 | 1.51 |
|  | kernel=rbf | 0.6275 | 0.5918 | 0.5960 | 39.91 | 1.52 |
|  | gamma=scale | 0.6275 | 0.5918 | 0.5960 | 39.51 | 1.90 |

* **Observations**

Among the evaluated classification models**, Random Forest, XGBoost, and Gradient Boosting** consistently delivered the highest cross-validation and test accuracies, demonstrating their effectiveness on this dataset.

**Random Forest** achieved **the best balance of accuracy and training time**, with hyperparameters such as max\_depth=20 and min\_samples\_split=10 improving generalization without overfitting.

**XGBoost** showed competitive accuracy, benefiting from careful tuning of learning\_rate and max\_depth, though it had generally faster training times than Random Forest.

**Gradient Boosting** also performed well, with optimal n\_estimators and max\_depth values enhancing predictive performance.

On the other hand**, Logistic Regression and SVM models**, while faster to train and test, **lagged in accuracy**, indicating that their simpler decision boundaries might not capture the complexity of the data as effectively.

* **Conclusion**

Overall, ensemble tree-based methods are preferred for this task due to their superior accuracy and reasonable computational cost. Fine-tuning hyperparameters notably improved performance across models, underscoring the importance of systematic hyperparameter search in classification problems.

**The best model is XGBoost.**

Why XGBoost?

* High predictive accuracy
* Efficient handling of missing data
* Robustness to overfitting due to regularization
* Fast training with parallel processing
* Works well on structured/tabular data

# **Some Graphs for Each Classification Model Performance**

A screenshot of a graph

AI-generated content may be incorrect.

A green and blue screen with white text

AI-generated content may be incorrect.

# **Conclusion**

Ensemble tree-based models outperformed simpler models in both classification and regression tasks.

* **XGBoost** was the best Classification model
* Balanced accuracy, speed, and overfitting control
* **CatBoost** was the best Regression model
* Models like Logistic Regression, SVM, and Linear Regression performed faster but could not match the predictive power of ensemble methods.

.