Airbnb Listing Analysis

Content

Data Source

- Airbnb price dataset
- Found on Kaggle

AIRBNB PRICE DATASET

Kaggle provides many publicly available datasets across a wide range of domains. We can explore these datasets for our analysis and projects. In this dataset,

- 74111 samples and 29 columns
- including ID, Log_Price, Property_Type, Room_Type, Amemties, Accommodates, Bathrooms, Bed_Type, Cancellation_Policy, Cleaning_Fee, City, Description, First_Review, Last Review, Name, Neighbourhood, Number of Review, Bedrooms, Beds, Zipcode, Instant_Bookable etc

BACKGROUND

In today's fast-paced world, the way we travel and seek accommodations has undergone a remarkable transformation, thanks to platforms like Airbnb. This dynamic marketplace has empowered property owners and travellers, offering a diverse range of lodging options.

PROBLEMS

However, setting the right price for an Airbnb listing is a complex task that poses challenges for both host and travellers

- Setting the right price for an Airbnb listing
- Balancing hosts’ earnings and competitive pricing
- Travellers’ expectations of value for money

PROJECT MOTIVATION

MOTIVATION

In this project, we hope to anaylsis and understand the various factors that influence the pricing of Airbnb listings. Undercover the patterns, trends, and insights that can help hosts optimize their pricing strategies and enable travellers to make informed decisions

Importance of optimizing Airbnb listing prices

Benefits for host

- Maximizing earnings and occipancy rates
- Competing effectively in the marketplace

Benefits for travellers

- Finding affordable and value-for-money accommodations

Data Cleaning

DATA CLEANING

Convert date columns to datetime format

In [8]:
```
# Convert date columns to datetime format
data['first_review'] = pd.to_datetime(data['first_review'])
data['host_since'] = pd.to_datetime(data['host_since'])
data['last_review'] = pd.to_datetime(data['last_review'])
```

Fill missing values with mean

In [9]:
```
# Fill missing values with mean
numeric_columns = ["log_price", "accommodates", "bathrooms", "number_of_reviews", "review_scores_rating", "bedrooms", "beds"
imputer = SimpleImputer(strategy="mean")
data[numeric_columns] = imputer.fit_transform(data[numeric_columns])
```

Remove outliers (rows where log_price is more than 3 standard deviations away from the mean)

In [11]:
```
# Remove outliers
#Remove rows where log_price is more than 3 standard deviations away from the mean
z_scores = (data["log_price"] - data["log_price"].mean()) / data["log_price"].std()
data = data[z_scores.abs() < 3]
```

DATA CLEANING

Cleaned data with sentiment scores

```
In [14]:
nltk.download("vader_lexicon")
sia = SentimentIntensityAnalyzer()

def get_sentiment(review):
    sentiment_scores = sia.polarity_scores(review)
    return sentiment_scores["compound"]

data["sentiment_score"] = data["description"].apply(get_sentiment)

# View the cleaned data with sentiment scores
print(data.head())
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\netis\AppData\Roaming\nltk_data...
```

Label encoder:

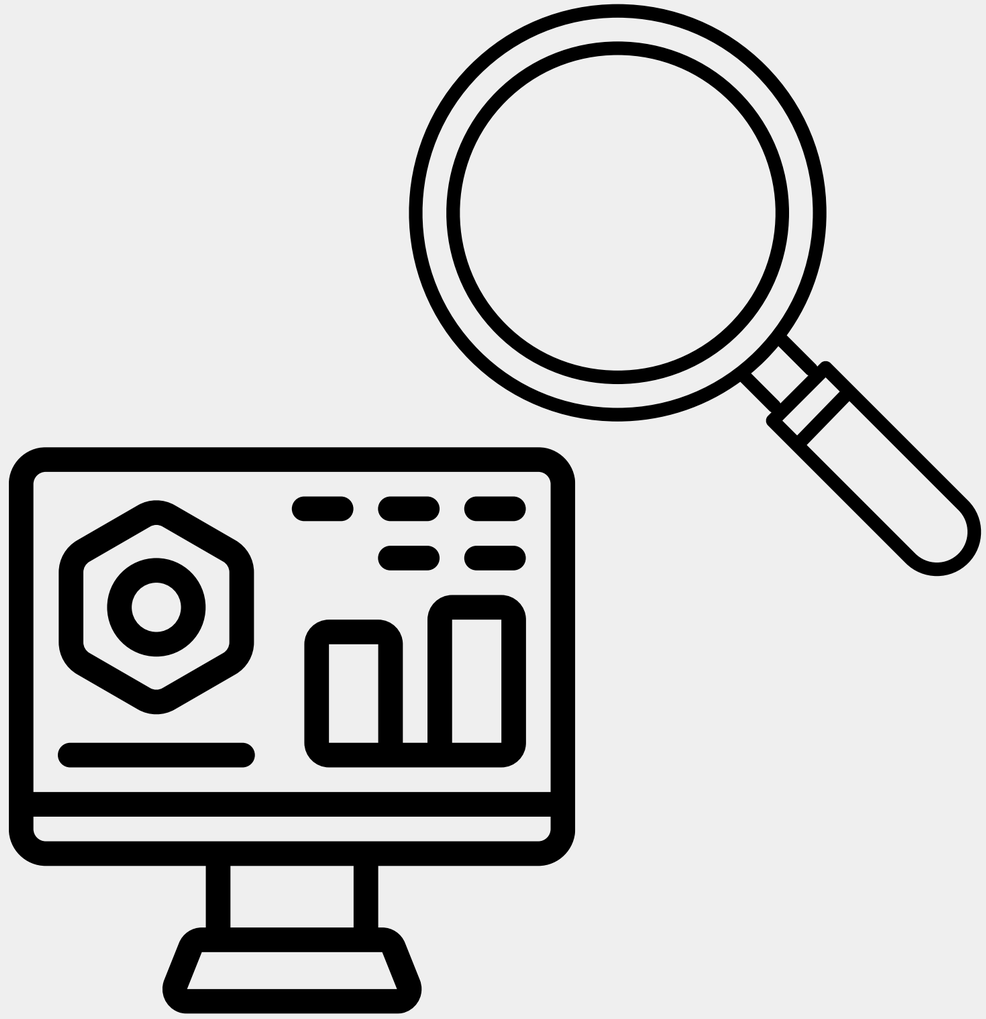
Fitting the Y values into the expected range

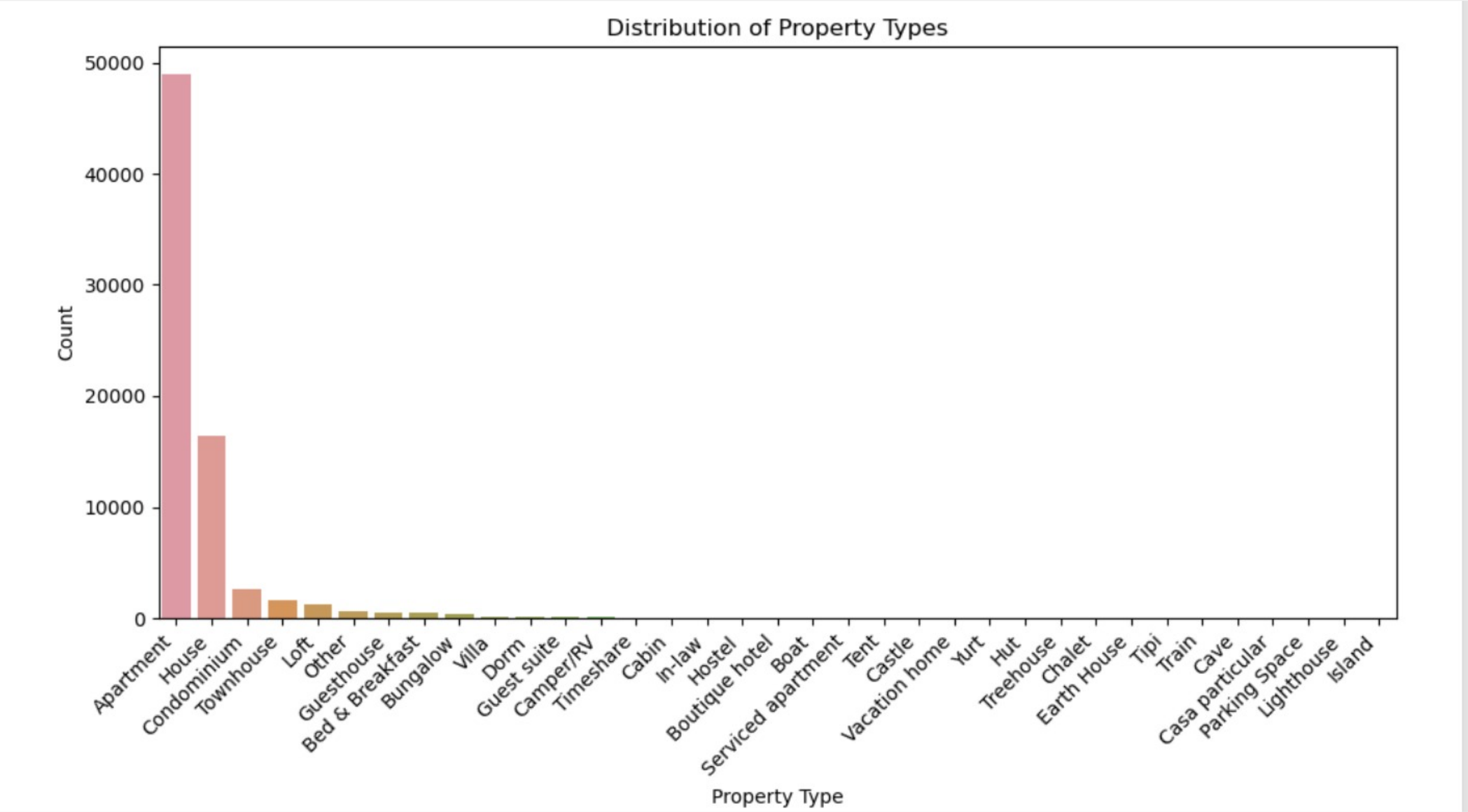
Fit transform:

object->string

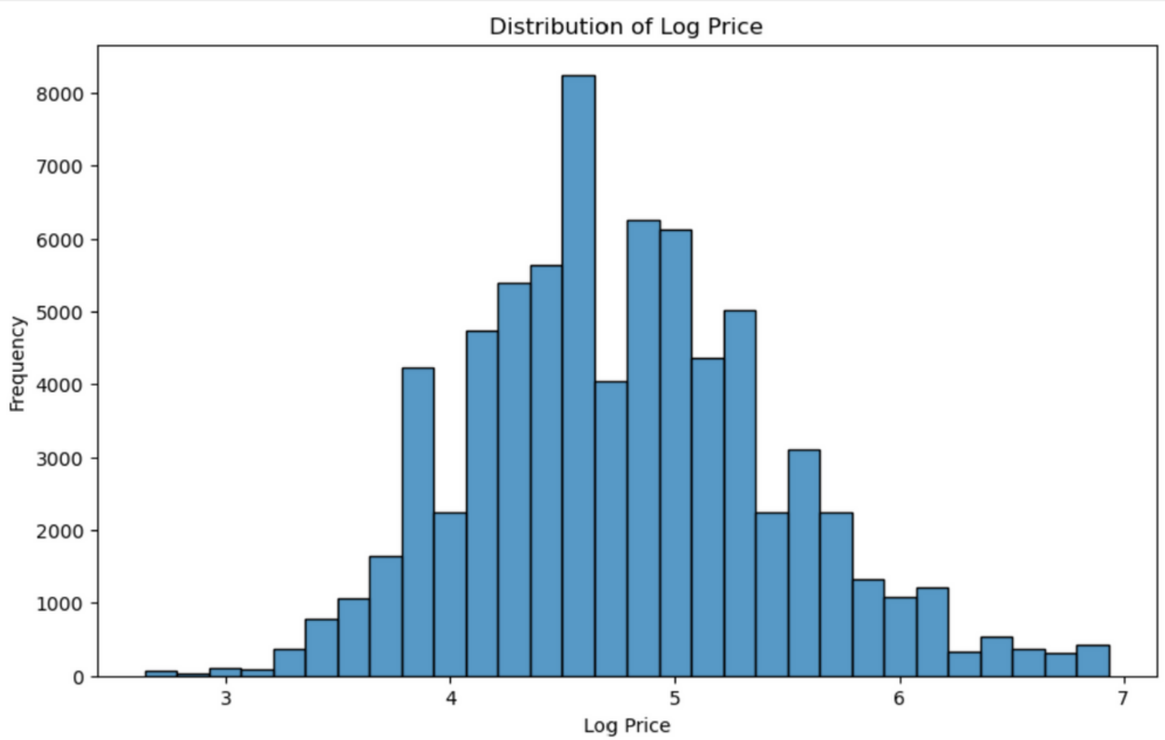
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y_train = le.fit_transform(y_train)
df["property_type"] = le.fit_transform(df["property_type"])
df["room_type"] = le.fit_transform(df["room_type"])
df["amenities"] = le.fit_transform(df["amenities"])
df["bed_type"] = le.fit_transform(df["bed_type"])
df["cancellation_policy"] = le.fit_transform(df["cancellation_policy"])
df["city"] = le.fit_transform(df["city"])
df["description"] = le.fit_transform(df["description"])
df["first_review"] = le.fit_transform(df["first_review"])
df["host_has_profile_pic"] = le.fit_transform(df["host_has_profile_pic"])
df["host_identity_verified"] = le.fit_transform(df["host_identity_verified"])
df["host_response_rate"] = le.fit_transform(df["host_response_rate"])
df["host_since"] = le.fit_transform(df["host_since"])
df["instant_bookable"] = le.fit_transform(df["instant_bookable"])
df["last_review"] = le.fit_transform(df["last_review"])
df["name"] = le.fit_transform(df["name"])
df["neighbourhood"] = le.fit_transform(df["neighbourhood"])
df["thumbnail_url"] = le.fit_transform(df["thumbnail_url"])
df["zipcode"] = le.fit_transform(df["zipcode"])
```

Exploratory Data Analysis





Most Common Property Types

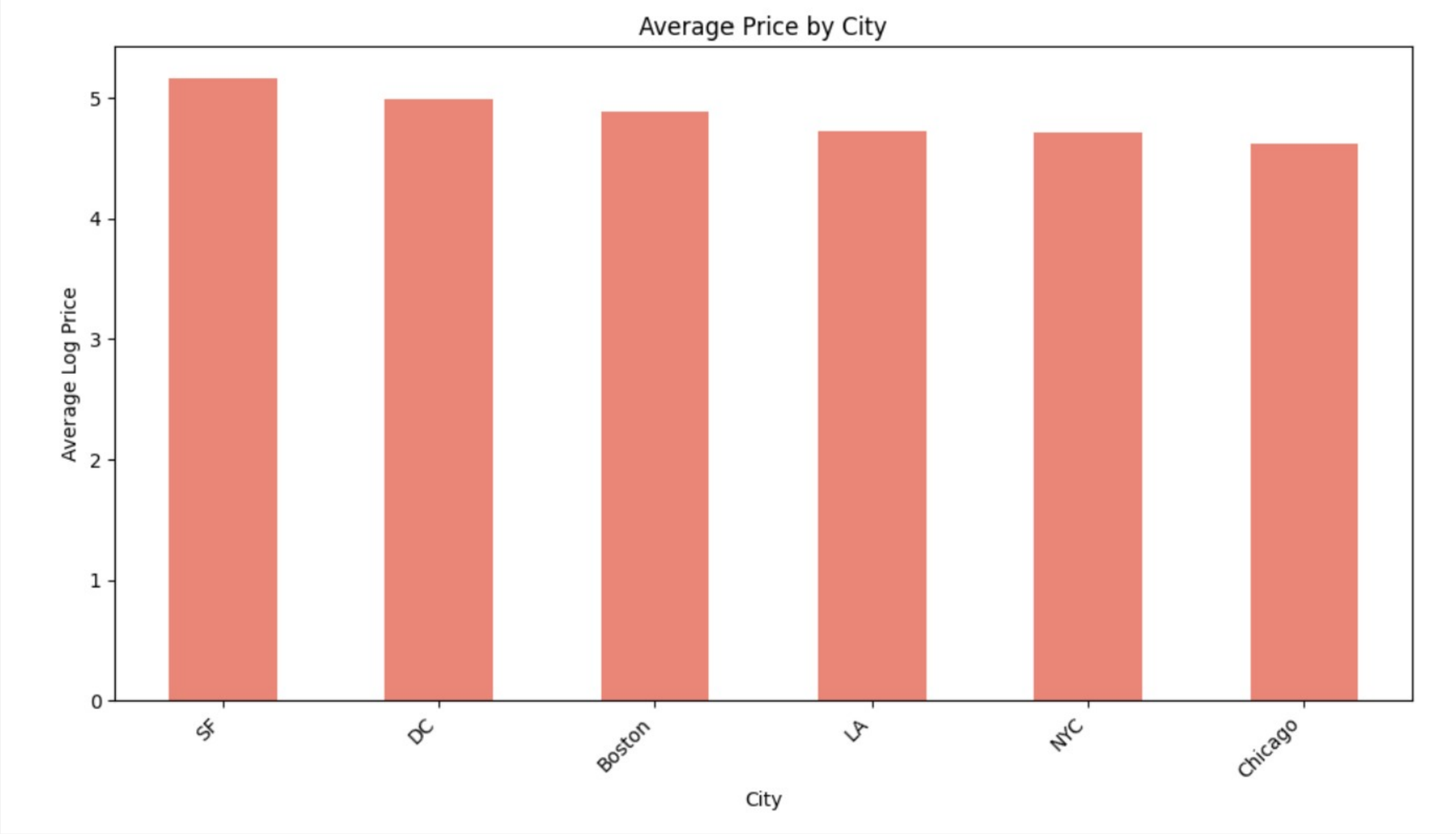


Graphs explaining relationship between log price and popularity

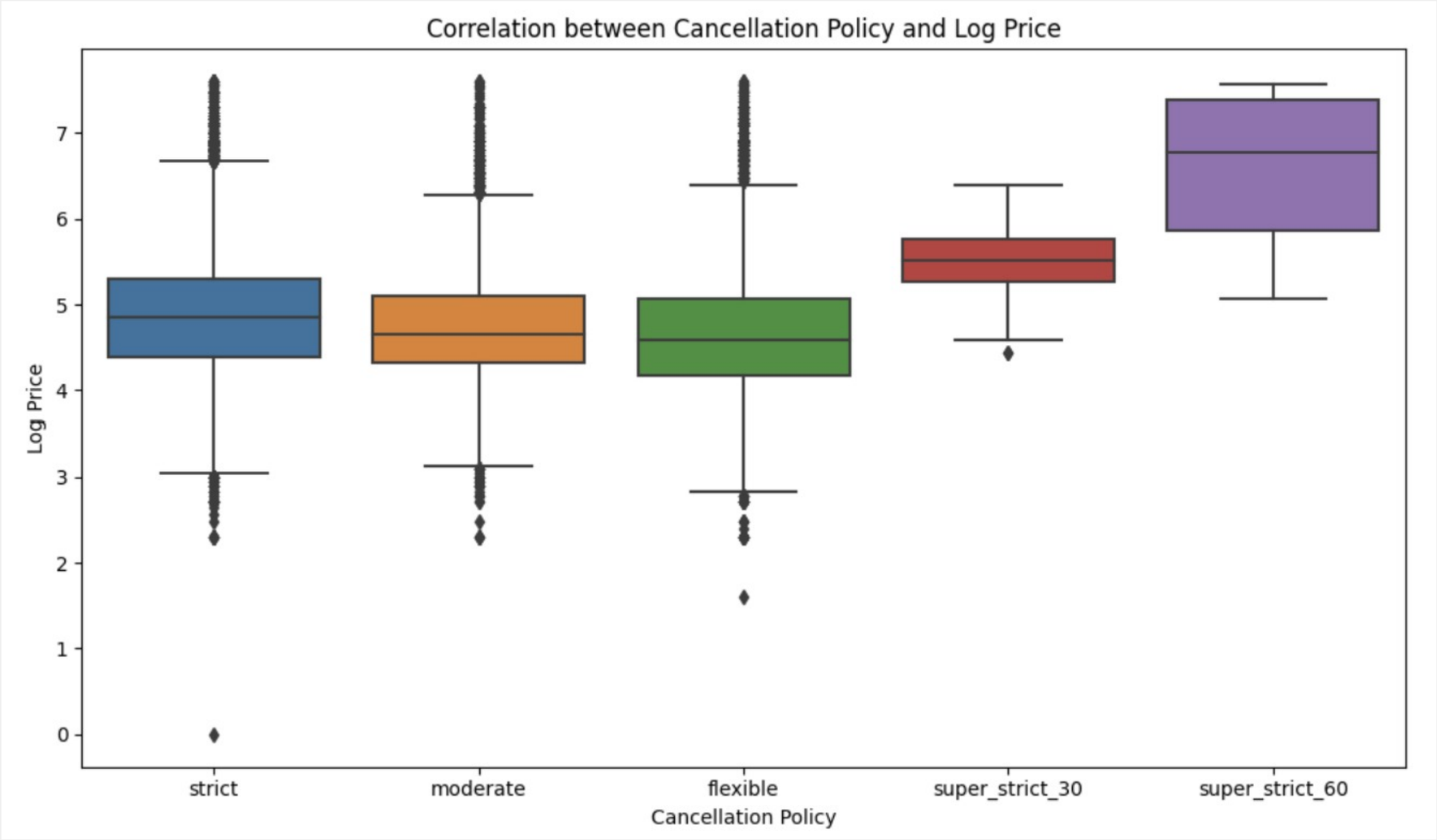
Number of reviews tends to be stable and the shape of both graphs are similar-looking

More graphs related to log-price

The price of coastal areas is slightly higher than inland areas.



More graphs related to log-price



According to the level of cancellation policy, the log price range will have a slightly difference.

Except for “super strict 30 and 60”.

XGBoost & Ridge regularization

WHY WE USE XGBOOST & RIDGE REGULARIZATION

Improved model performance : Both techniques can enhance the predictive performance of machine learning models.

XGBoost's : Combining weak models often leads to improved accuracy and handling of complex relationships in the data.

Ridge regularization : Helps to reduce the impact of noisy features and prevents overfitting, resulting in more stable and reliable predictions.

WHY WE USE XGBOOST & RIDGE REGULARIZATION

Feature selection and importance

XGBoost's : Provides built-in feature importance measures, which help identify the most essential features in the dataset. This enables feature selection and can guide data-driven decision-making.

Ridge regularization : Indirectly performs feature selection by shrinking less significant coefficients towards zero, effectively reducing the influence of irrelevant features.

WHY WE USE XGBOOST & RIDGE REGULARIZATION

Robustness to noise and outliers

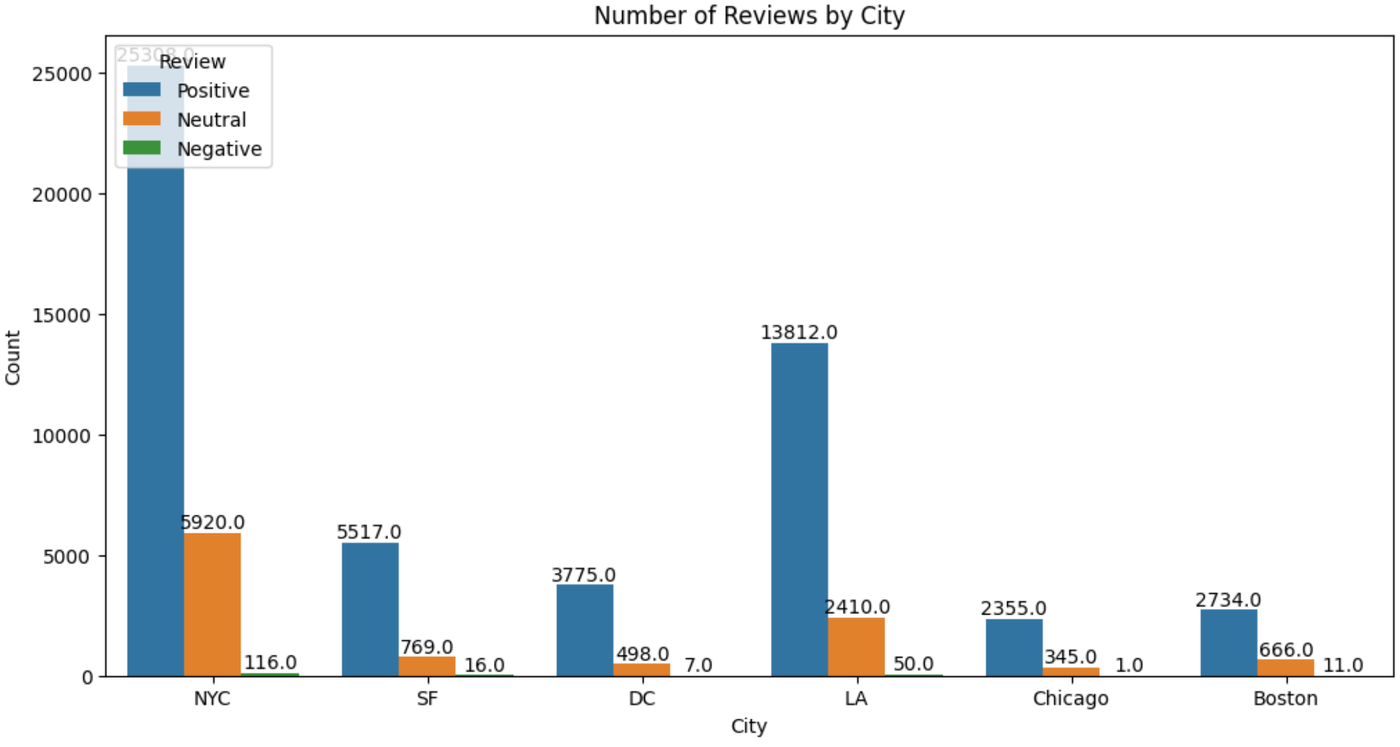
Both XGBoost and Ridge regularization techniques can improve the robustness of models by reducing the impact of noisy data points and outliers. XGBoost's ensemble approach and Ridge regularization's penalty on large coefficients help to mitigate the influence of extreme values, leading to more robust and stable predictions.

Review Ratings

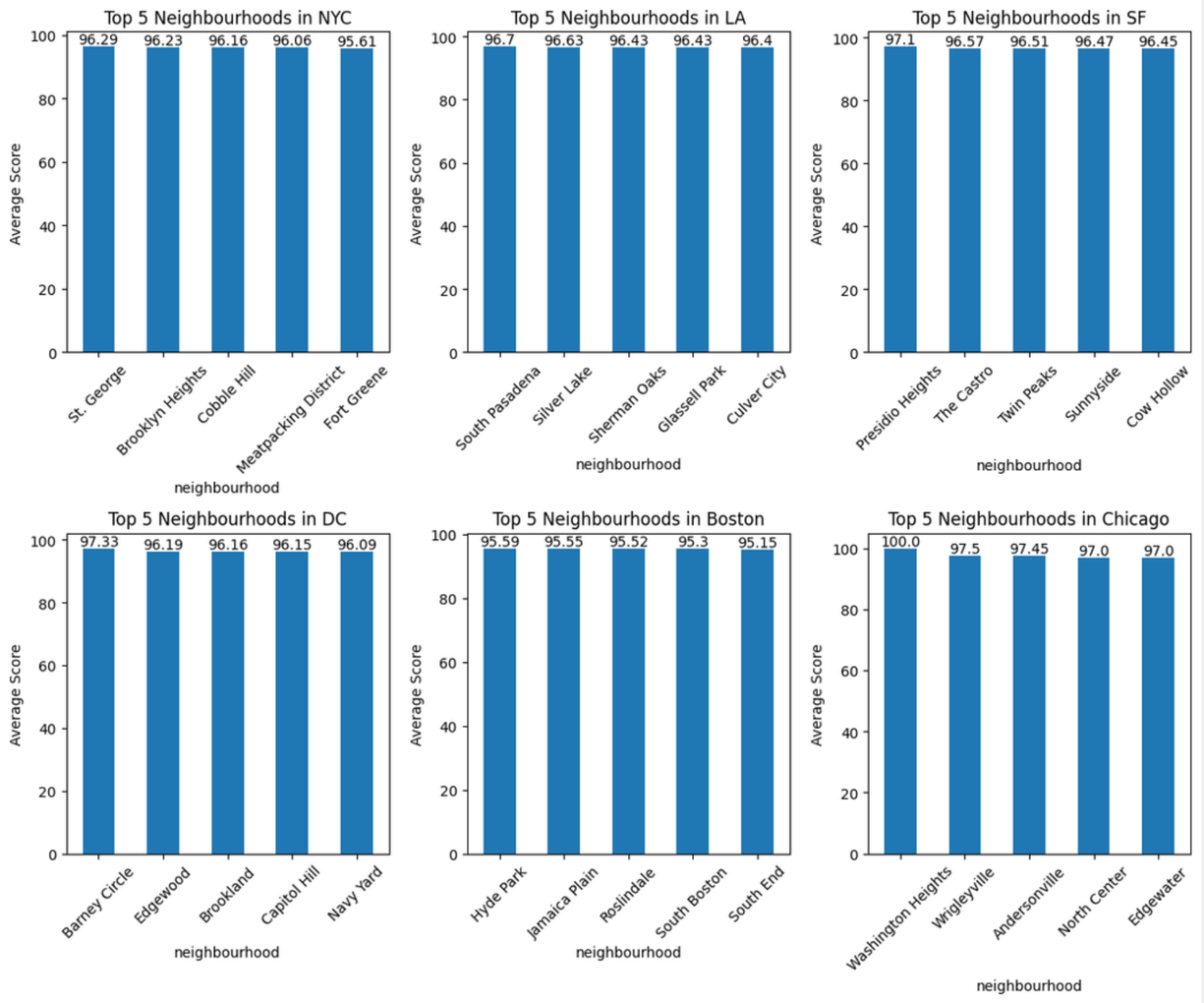
Data Outlook

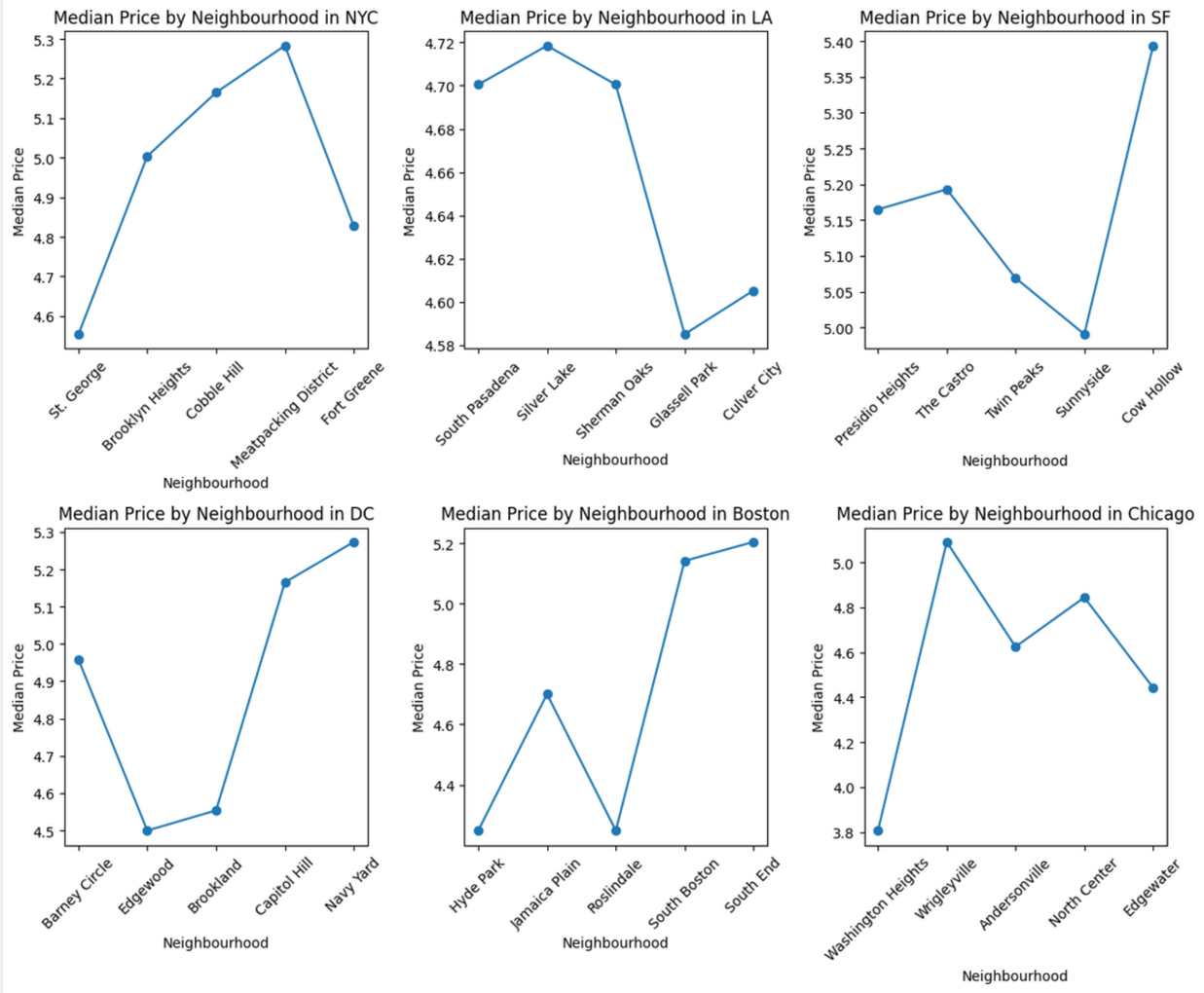
- Data contains Airbnb listings from the following states
  - San Francisco
  - Los Angeles
  - Chicago
  - Boston
  - New York City
  - Chicago

- These are some of the most densely populated states in USA, which is perfect for potential Airbnb listers.



Top 5 Neighbourhoods Per City

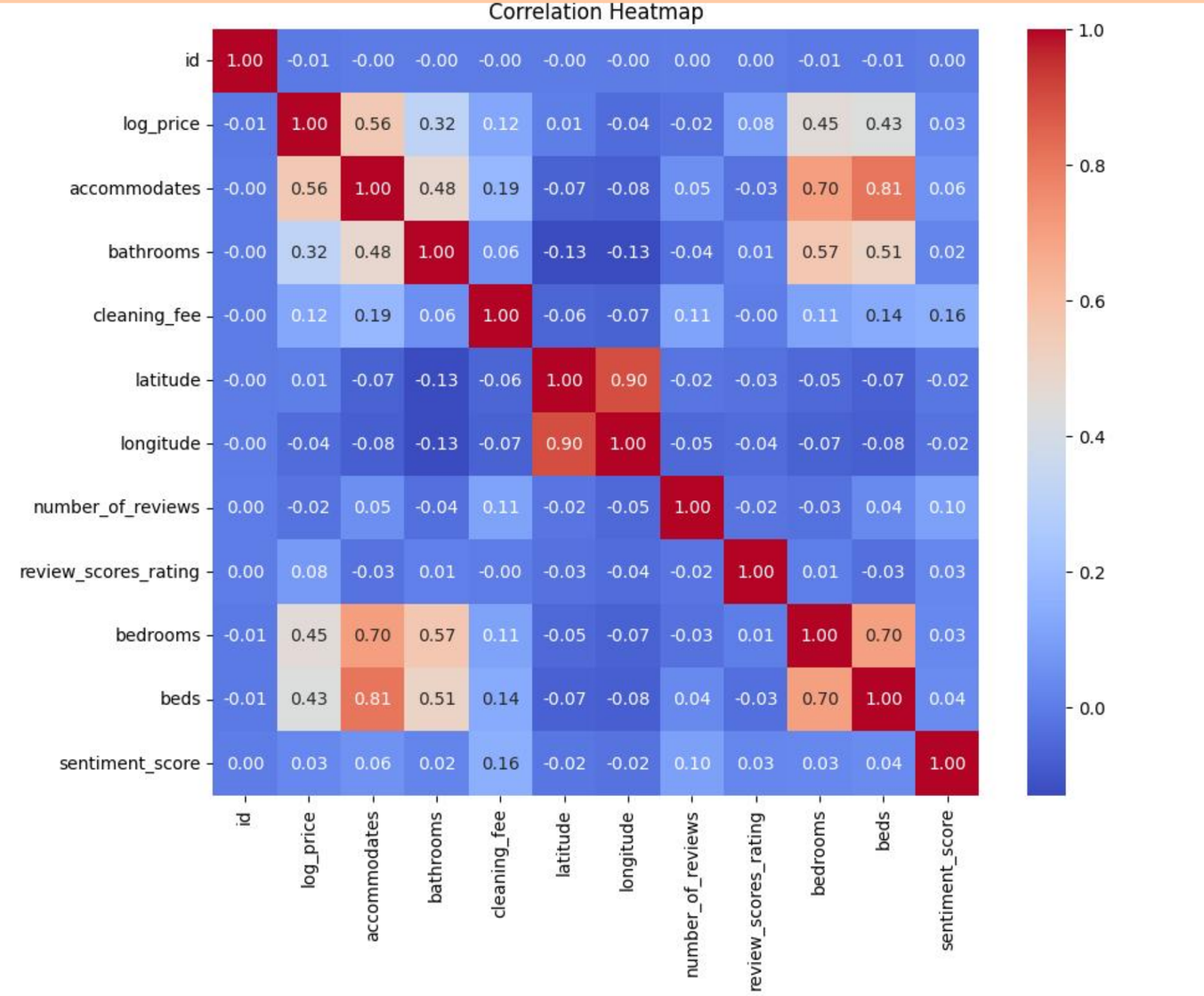




Result

(From result of XGBoost model)

Variation of log price among different factors



Log price highly correlate with accomodates (0.56)

Log price correlate with beds (0.43) and bedrooms (0.45)

Log price slightly correlate with rating (0.08) and sentiment score (0.03)

Log price least correlate with longtitudes (-0.04) and number of reviews (-0.02)

HOW TO BETTER PRICE THEIR AIRBNB

Focus on **personal factors** instead of **environmental factors**

Investigate more on the needs of the users

- accomodates
- bedroom types
- beds provided

Avoid spending/cut cost on the external factors

- number of reviews
- review scores rating
- sentiment score

Q&A Session

Thanks for listening!

[redacted]