SUDIA ARIBIA

PREDICTIVE MODELING FOR CONDO PRICE TRENDS IN SAUDI ARABIA

CONTENT

01 About Us

02 Introduction

03 web scraping

04 Data Overview

05 Data Cleaning

06 Exploratory Data Analysis

07 Machine Learning Model

08 Conclusion

ABOUT US

شموخ علي الحلافي
تسنيم عبدالرحمن الصعيري
منى سعيد فالح
بيان محمد الشمراني

الدكتورة:
حصه محمد القحطاني

INTRODUCTION

Objective: Welcome to our presentation on "Predictive Modeling for Condo Price Trends in Saudi Arabia." Today, we'll delve into how we leveraged Airbnb data to gain valuable insights into condo pricing trends

1. **Overview of the predictive modeling objective.**
2. **Introduction to the Airbnb dataset used in our analysis.**

3.Key steps in data preprocessing and exploration.
4.Highlights from our predictive modeling approach.

5.Evaluation of different models for condo price prediction.
6.implications and potential applications of our findings.

2: WEB SCRAPING

- Explain the data source (Airbnb in Saudi Arabia).
- List and describe relevant columns for condo price prediction (e.g., url, name, header, beds, bedrooms, date_range, price, rating).

```
import requests
from bs4 import BeautifulSoup
```

```
[ ]  def extract_basic_features(listing_html):
         features_dict = {}

         # Extracting basic information
         url_element = listing_html.find('a')
         url = url_element.get('href') if url_element else None
         features_dict['url'] = url

         name_element = listing_html.find("span", {"data-testid": "listing-card-name"})
         name = name_element.text.strip() if name_element else None
         features_dict['name'] = name

         header_element = listing_html.find("div", {"data-testid": "listing-card-title"})
         header = header_element.text.strip() if header_element else None
         features_dict['header'] = header

         # Extracting additional details
         beds_element = listing_html.select_one('.g1qv1ctd .fb4nyux:nth-child(2)')
         beds = beds_element.text.strip() if beds_element else None
         features_dict['beds'] = beds

         bedrooms_element = listing_html.select_one('.g1qv1ctd .fb4nyux:nth-child(3)')
         bedrooms = bedrooms_element.text.strip() if bedrooms_element else None
         features_dict['bedrooms'] = bedrooms

         date_range_element = listing_html.select_one('.g1qv1ctd .fb4nyux:nth-child(4)')
         date_range = date_range_element.text.strip() if date_range_element else None
         features_dict['date_range'] = date_range

         # Extracting pricing information
         price_element = listing_html.select_one('div._1jo4hgw span._tyxjp1')
         price = price_element.text.strip() if price_element else None
         features_dict['price'] = price

         # Extracting rating information
```

Project Initiation

web scraping

BeautifulSoup

DATA OVERVIEW:

The dataset comprises seven columns:

01

url

URL of the condo listing on Airbnb.

02

name

Name of the codo

03

location

the location of the codo

04

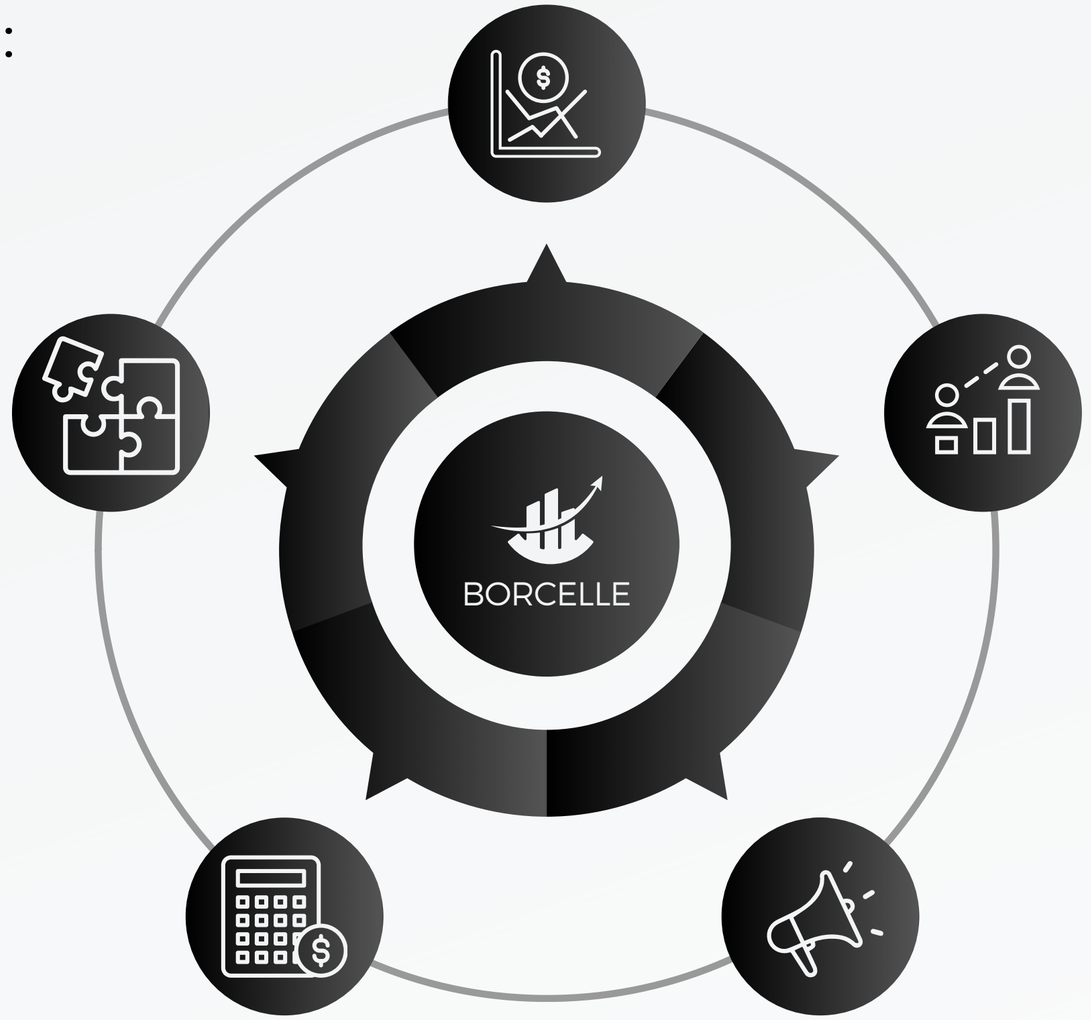
price

the price of the codo

05

rating

the rating of the codo



DATA CLEANING AND PREPROCESSING:

01 Handling Missing Values:

```
data.isnull().sum()
```

```
url              0
name             0
header           0
beds             0
bedrooms         0
date_range       0
price          854
rating          46
dtype: int64
```

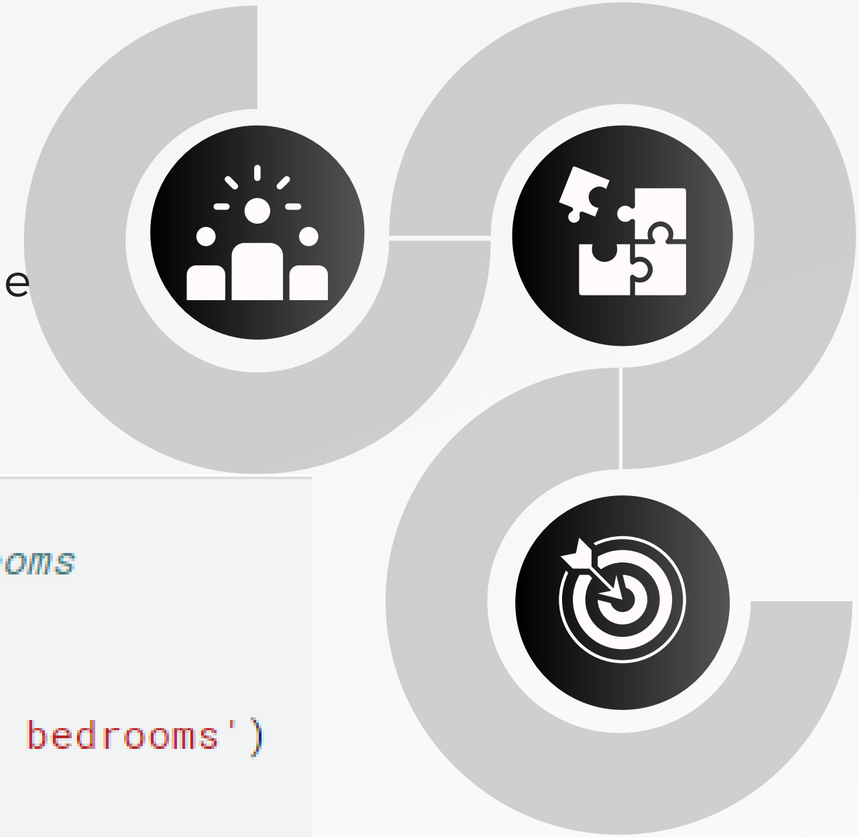
The dataset had missing values in the 'price' and 'rating' columns, which were addressed by dropping rows with null values.

```
data.dropna(inplace=True)
```

DATA CLEANING AND PREPROCESSING:

02 Feature Extraction

New features, 'num_beds' and 'num_bedrooms,' were extracted from the 'bedrooms' column to provide additional insights.

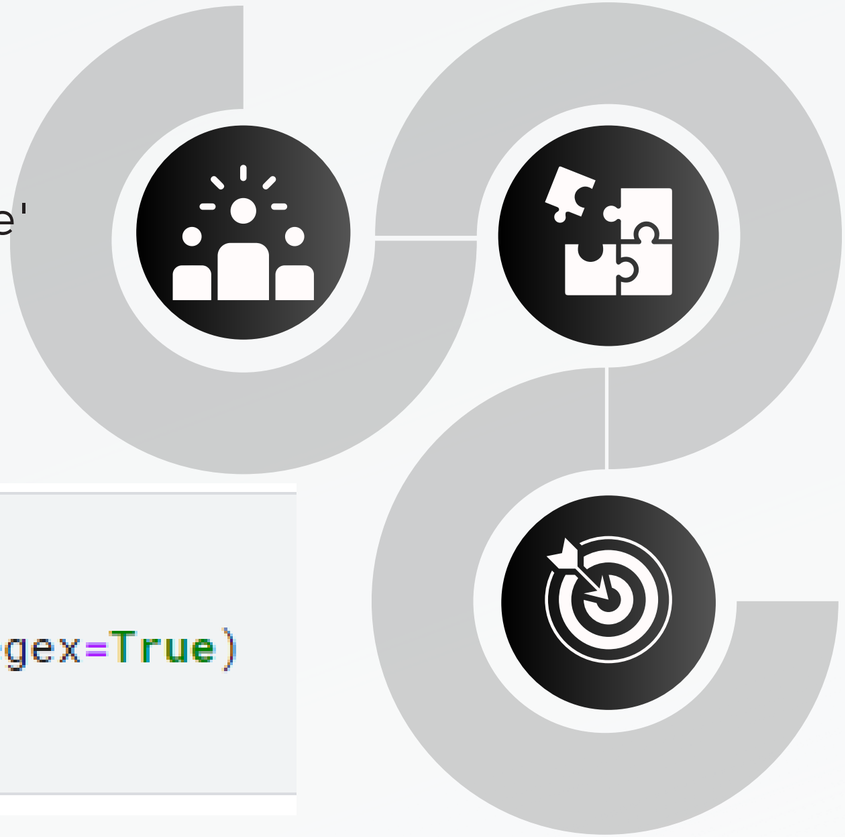


```
# Extract number of beds and number of bedrooms
data[['num_beds', 'num_bedrooms']] =
        data['bedrooms'].str
                .extract('(\d+) beds.*?(\d+) bedrooms')
                .fillna(1)
```

DATA CLEANING AND PREPROCESSING:

02 Fixing Price Values:

Dollar signs were removed from the 'price' column, converting it to a float type for analysis.

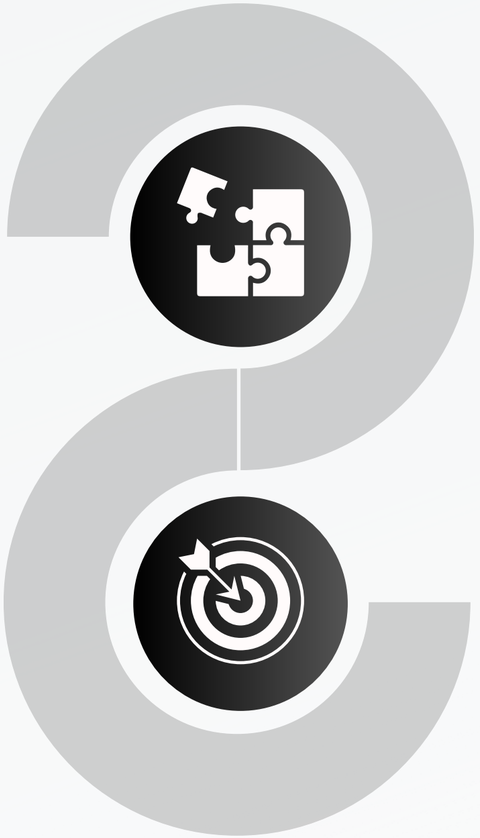


```
data['price'] = data['price']
                .replace('[\$,]', '', regex=True)
                .astype(float)
```

DATA CLEANING AND PREPROCESSING:

02 **City Mapping:** The 'header' column was mapped to 'city' for better analysis.

```
# City names mapping
mapping = { 'Apartment in Riyadh': "Riyadh", 'Apartment in Jeddah': "Jeddah", 'Condo in Riyadh': "Riyadh",
           'Apartment in Makkah': "Makkah", 'Condo in Jeddah': "Jeddah", 'Home in Riyadh': "Riyadh",
           'Apartment in King Abdullah Economic City': "King Abdullah", 'Apartment in الرياض': "Riyadh",
           'Hotel in Makkah': "Makkah", 'Apartment in  Muhammadiyah, Jeddah': "Jeddah",
           'Apartment in Mecca': "Makkah", 'Chalet in Riyadh': "Riyadh", 'Condo in Makkah': "Makkah",
           'Room in Riyadh': "Riyadh", 'Home in Jeddah': "Jeddah", 'Villa in Riyadh': "Riyadh",
           'Hut in Jeddah': "Jeddah", 'Cabin in Jeddah': "Jeddah", 'Hut in Taif': "others",
           'Apartment in Diriyah': "others", 'Guest suite in Riyadh': "Riyadh", 'Loft in Jeddah': "Jeddah",
           'Condo in Al Khobar': "others", 'Chalet in Duba': "others", 'Chalet in Mecca': "Makkah",
           'Vacation home in Riyadh': "Riyadh", 'Tiny home in Ragal Almaa': "others", 'Loft in Riyadh City': "Riyadh",
           'Place to stay in Riyadh': "Riyadh", 'Farm stay in Jalajil': "others", 'Barn in Riyadh': "Riyadh",
           'Ranch in AlUla': "others" , 'Cabin in At Taif': "others", 'Villa in Jeddah': "Jeddah",
           'Guesthouse in Makkah': "Makkah"}

# Apply mapping to the 'city' column
data['city'] = data['header'].map(mapping)
```



EXPLORATORY DATA ANALYSIS (EDA):

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis vulputate nulla at ante rhoncus, vel efficitur felis condimentum. Proin odio odio.

Project Initiation

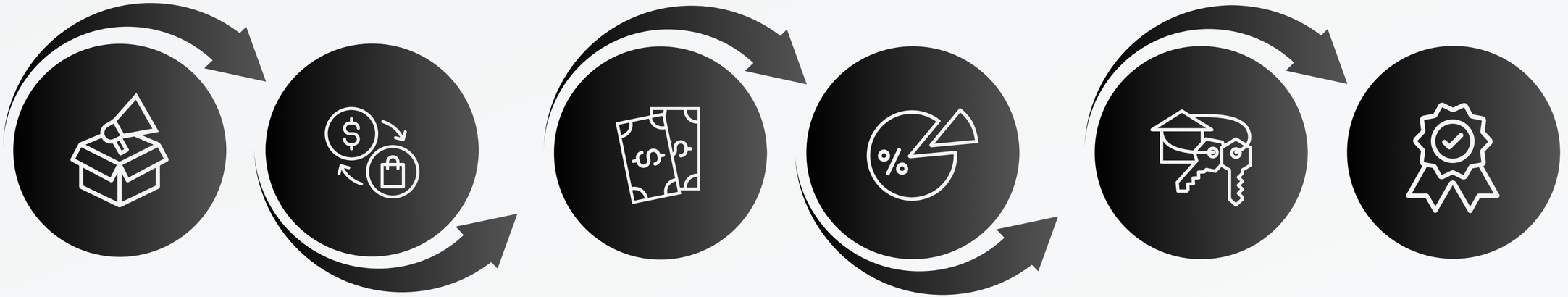
Review and Editing

Presentation and Sharing

Exploratory Data Analysis (EDA):

April

July



January

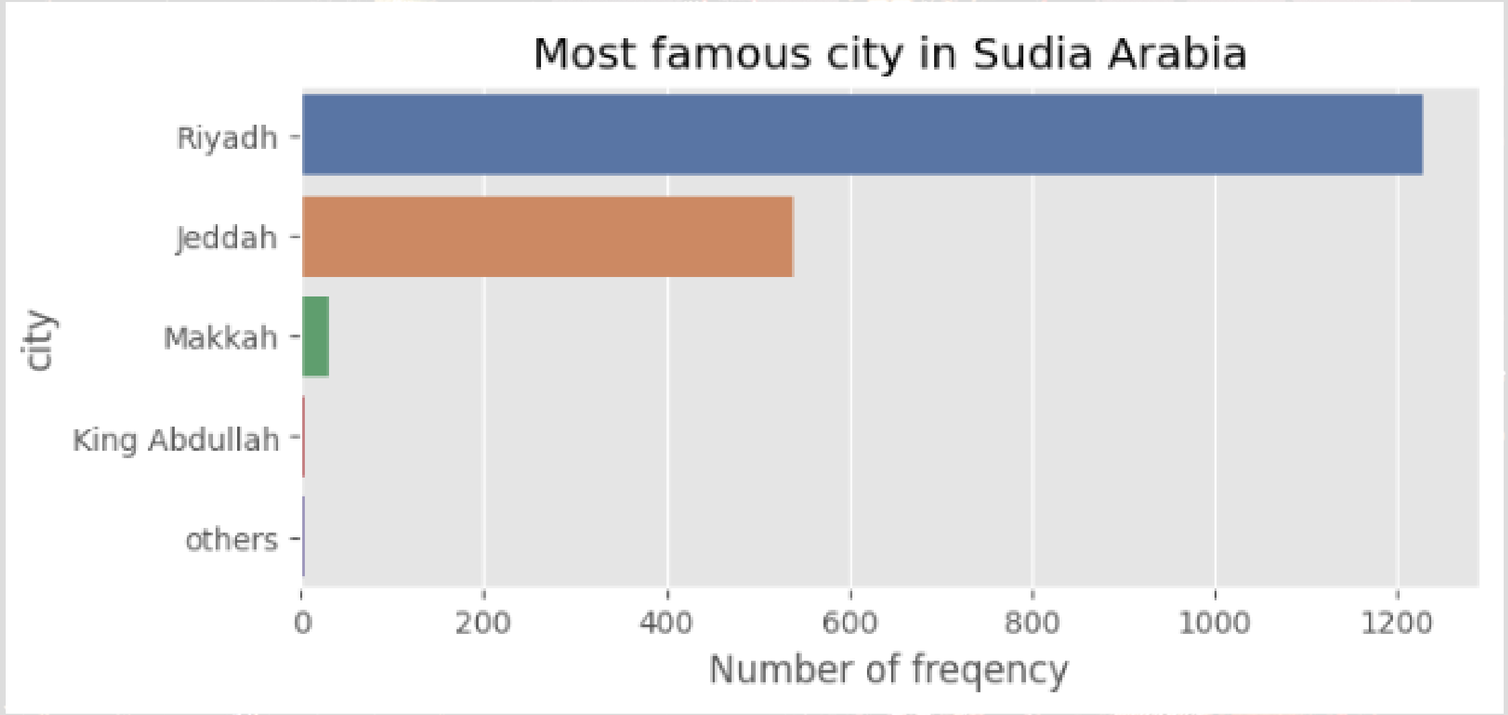
March

May

EXPLORATORY DATA ANALYSIS (EDA):

City Distribution:

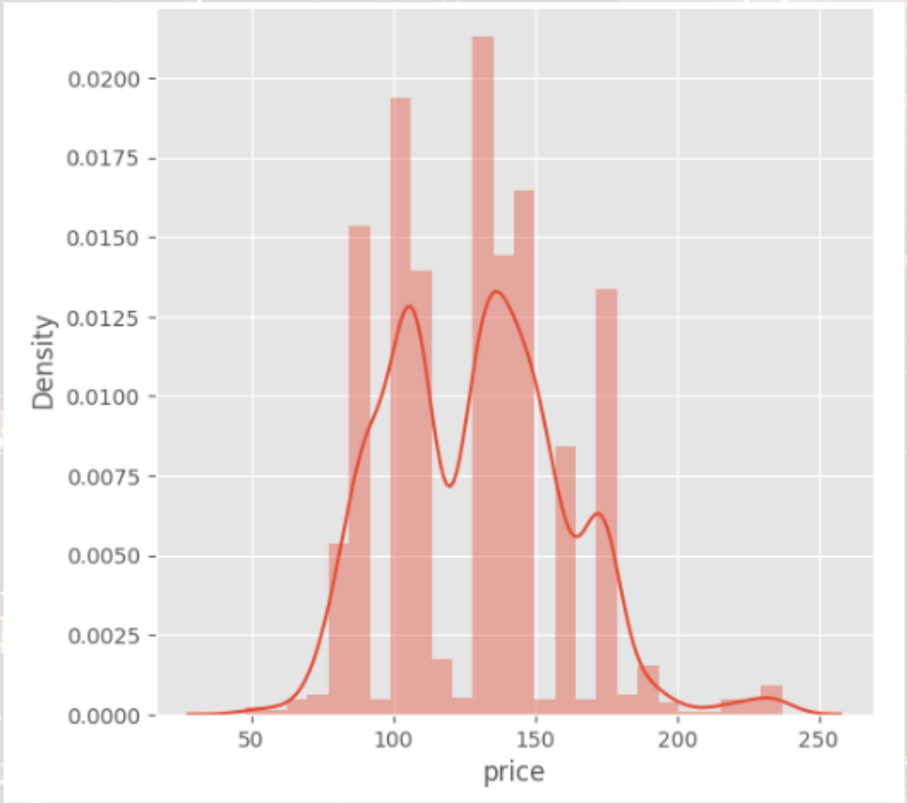
Riyadh emerged as the most frequent city in the dataset, influencing the overall distribution.



EXPLORATORY DATA ANALYSIS (EDA):

Price Distribution:

The distribution of house pricing indicated a variety of price ranges across the dataset.

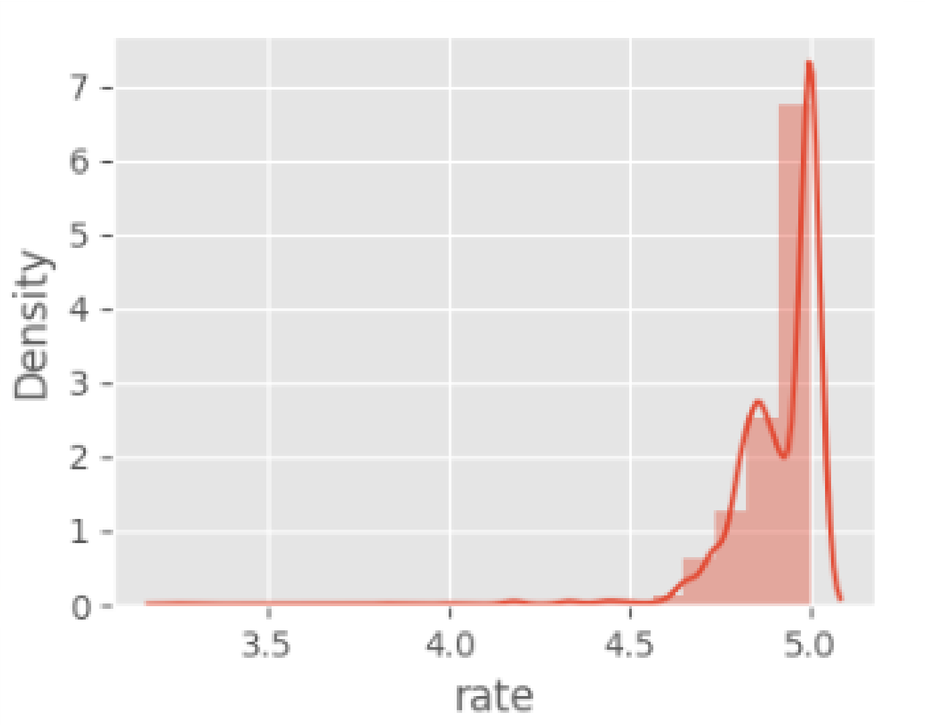



```
plt.figure(figsize=(6,6))
sns.distplot(df4['price'])
plt.show()
```

EXPLORATORY DATA ANALYSIS (EDA):

Rating Distribution:

The majority of places had ratings between 4.5 and 5, with lower ratings being rare.

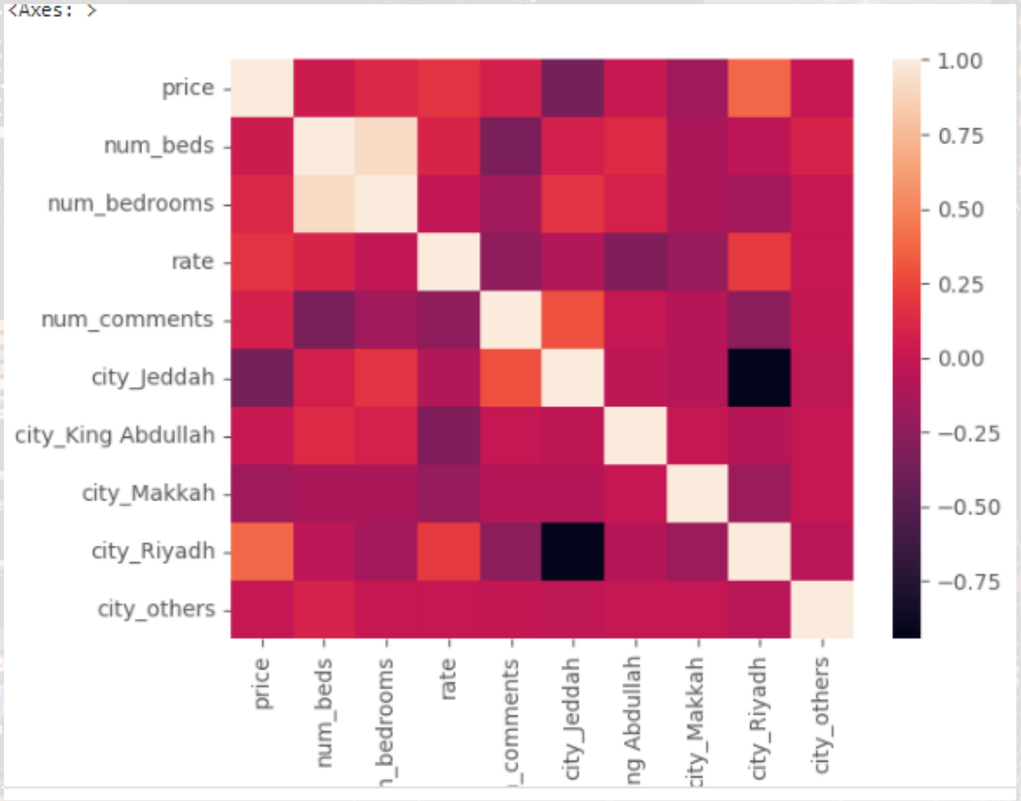



```
plt.figure(figsize=(4,3))
rating = df4['rate']
sns.distplot(rating,bins=20)
```

EXPLORATORY DATA ANALYSIS (EDA):

Correlation Heatmap:

1. A heatmap was used to visualize the correlation between different features.

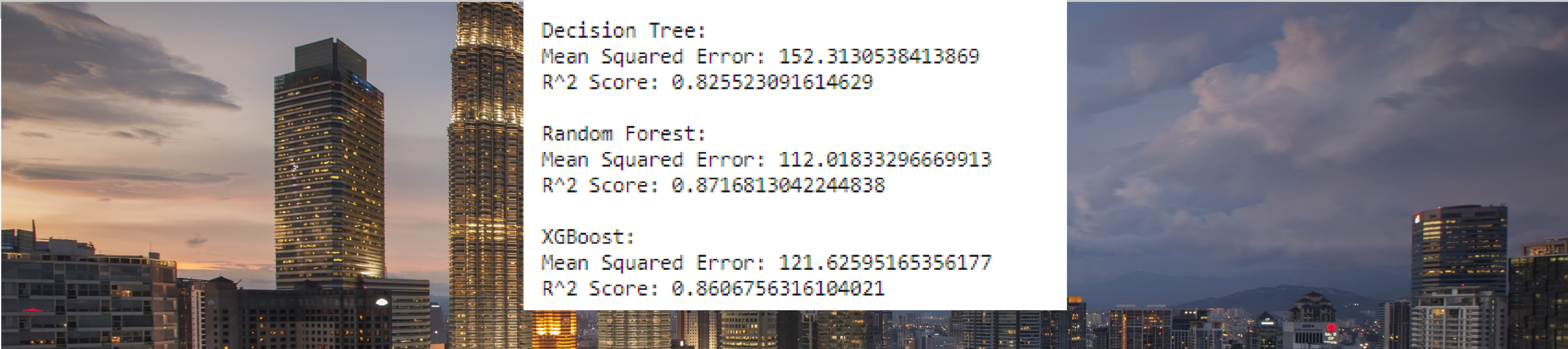



```
sns.heatmap(df5.corr())
```

MACHINE LEARNING MODELS:

The dataset was split into training and testing sets for machine learning model evaluation. Several regression models, including Linear Regression, K-Nearest Neighbors, Decision Tree, Random Forest, and XGBoost, were applied to predict departmental prices.

```
Linear Regression:
Mean Squared Error: 572.9087369873265
R^2 Score: 0.34372437098785846

K-Nearest Neighbors:
Mean Squared Error: 169.96782369146004
R^2 Score: 0.8052992855519931

Decision Tree:
Mean Squared Error: 152.3130538413869
R^2 Score: 0.825523091614629

Random Forest:
Mean Squared Error: 112.01833296669913
R^2 Score: 0.8716813042244838

XGBoost:
Mean Squared Error: 121.62595165356177
R^2 Score: 0.8606756316104021
```



MACHINE LEARNING MODELS:

Model Evaluation:

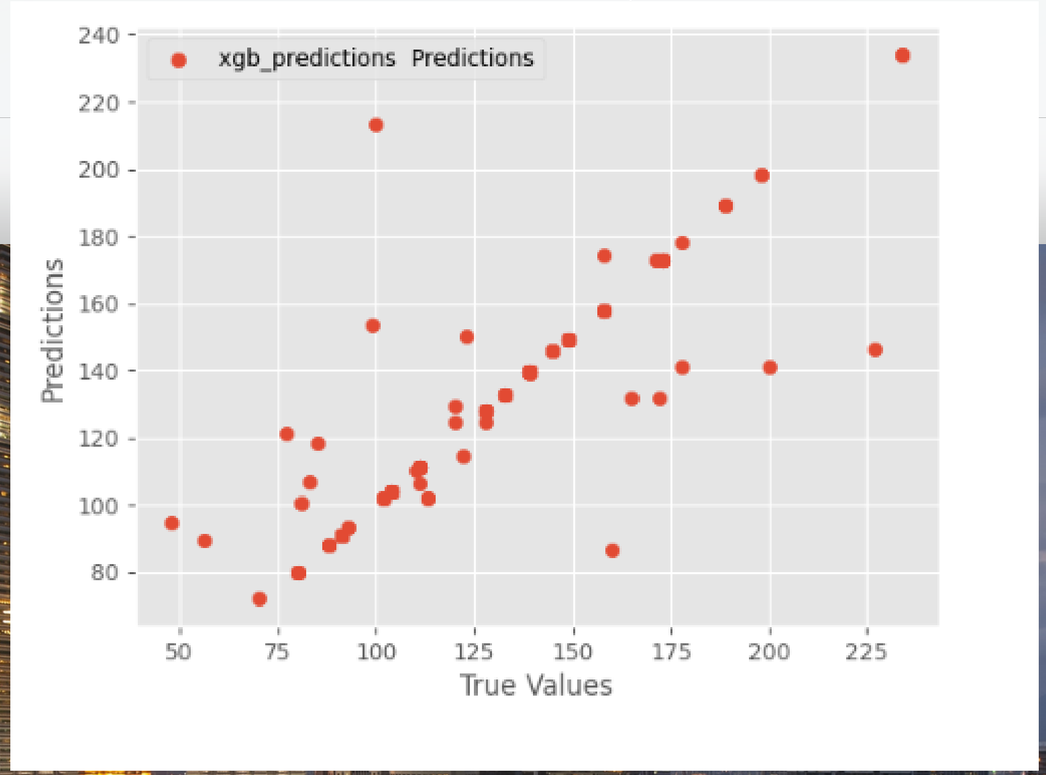
- Linear Regression: MSE = 572.91, R^2 = 0.34
- K-Nearest Neighbors: MSE = 169.97, R^2 = 0.81
- Decision Tree: MSE = 154.61, R^2 = 0.82
- Random Forest: MSE = 118.55, R^2 = 0.86
- XGBoost: MSE = 121.63, R^2 = 0.86

```
lr_model.fit(X_train, y_train)
lr_predictions = lr_model.predict(X_test)

# Logistic Regression (for demonstration purposes; not suitable for regression)
# Note: Logistic Regression is typically used for classification problems
# For regression, consider other algorithms like Linear Regression
logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)
logistic_predictions = logistic_model.predict(X_test)

# K-Nearest Neighbors
knn_model = KNeighborsRegressor()
knn_model.fit(X_train, y_train)
knn_predictions = knn_model.predict(X_test)

# Decision Tree
dt_model = DecisionTreeRegressor()
dt_model.fit(X_train, y_train)
dt_predictions = dt_model.predict(X_test)

# Random Forest
rf_model = RandomForestRegressor()
```

MACHINE LEARNING MODELS:

Model Evaluation:

- Linear Regression: MSE = 572.91, R^2 = 0.34
- K-Nearest Neighbors: MSE = 169.97, R^2 = 0.81
- Decision Tree: MSE = 154.61, R^2 = 0.82
- Random Forest: MSE = 118.55, R^2 = 0.86
- XGBoost: MSE = 121.63, R^2 = 0.86



ANY QUESTIONS