video:

https://drive.google.com/file/d/1F5oyxLjeRN3LDYVZyohvyjlVkG_GGkDv/vie
usp=sharing

Aim

This project aims to build a machine learning model to predict the price of Airbnb listingsbased on various features such as property type, room type, location, amenities, and host characteristics. By analyzing these factors, this project will provide actionable insights to Airbnb hosts to optimize their listing prices.

Project- Part A: Airbnb Price Prediction and Insights

Importing Modules

In [138…

```
# importing requried modules
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score

import warnings
warnings.filterwarnings("ignore")
```

Loading Data

In [140…

```
# load the data set
df = pd.read_csv('airbnb_data.csv')

#to display frist five row
df.head()
```

Out[140…

	id	log_price	property_type	room_type	amenities	accommodates
0	6901257	5.010635	Apartment	Entire home/apt	{"Wireless Internet","Air conditioning",Kitche...	3
1	6304928	5.129899	Apartment	Entire home/apt	{"Wireless Internet","Air conditioning",Kitche...	7
2	7919400	4.976734	Apartment	Entire home/apt	{TV,"Cable TV","Wireless Internet","Air condit...	5
3	13418779	6.620073	House	Entire home/apt	{TV,"Cable TV",Internet,"Wireless Internet",Ki...	4
4	3808709	4.744932	Apartment	Entire home/apt	{TV,Internet,"Wireless Internet","Air conditio...	2

5 rows × 29 columns

Data Exploration and Preprocessing

Data Description

In [142…

```
# display basic information
df.info()

# summary of statistics of the data set
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74111 entries, 0 to 74110
Data columns (total 29 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   id                      74111 non-null  int64  
 1   log_price               74111 non-null  float64
 2   property_type           74111 non-null  object 
 3   room_type               74111 non-null  object 
 4   amenities               74111 non-null  object 
 5   accommodates            74111 non-null  int64  
 6   bathrooms               73911 non-null  float64
 7   bed_type                74111 non-null  object 
 8   cancellation_policy     74111 non-null  object 
 9   cleaning_fee            74111 non-null  bool   
 10  city                    74111 non-null  object 
 11  description             74111 non-null  object 
 12  first_review            58247 non-null  object 
 13  host_has_profile_pic    73923 non-null  object 
 14  host_identity_verified  73923 non-null  object 
 15  host_response_rate      55812 non-null  object 
 16  host_since              73923 non-null  object 
 17  instant_bookable        74111 non-null  object 
 18  last_review             58284 non-null  object 
 19  latitude                74111 non-null  float64
 20  longitude               74111 non-null  float64
 21  name                    74111 non-null  object 
 22  neighbourhood           67239 non-null  object 
 23  number_of_reviews       74111 non-null  int64  
 24  review_scores_rating    57389 non-null  float64
 25  thumbnail_url           65895 non-null  object 
 26  zipcode                 73143 non-null  object 
 27  bedrooms                74020 non-null  float64
 28  beds                    73980 non-null  float64
dtypes: bool(1), float64(7), int64(3), object(18)
memory usage: 15.9+ MB
```

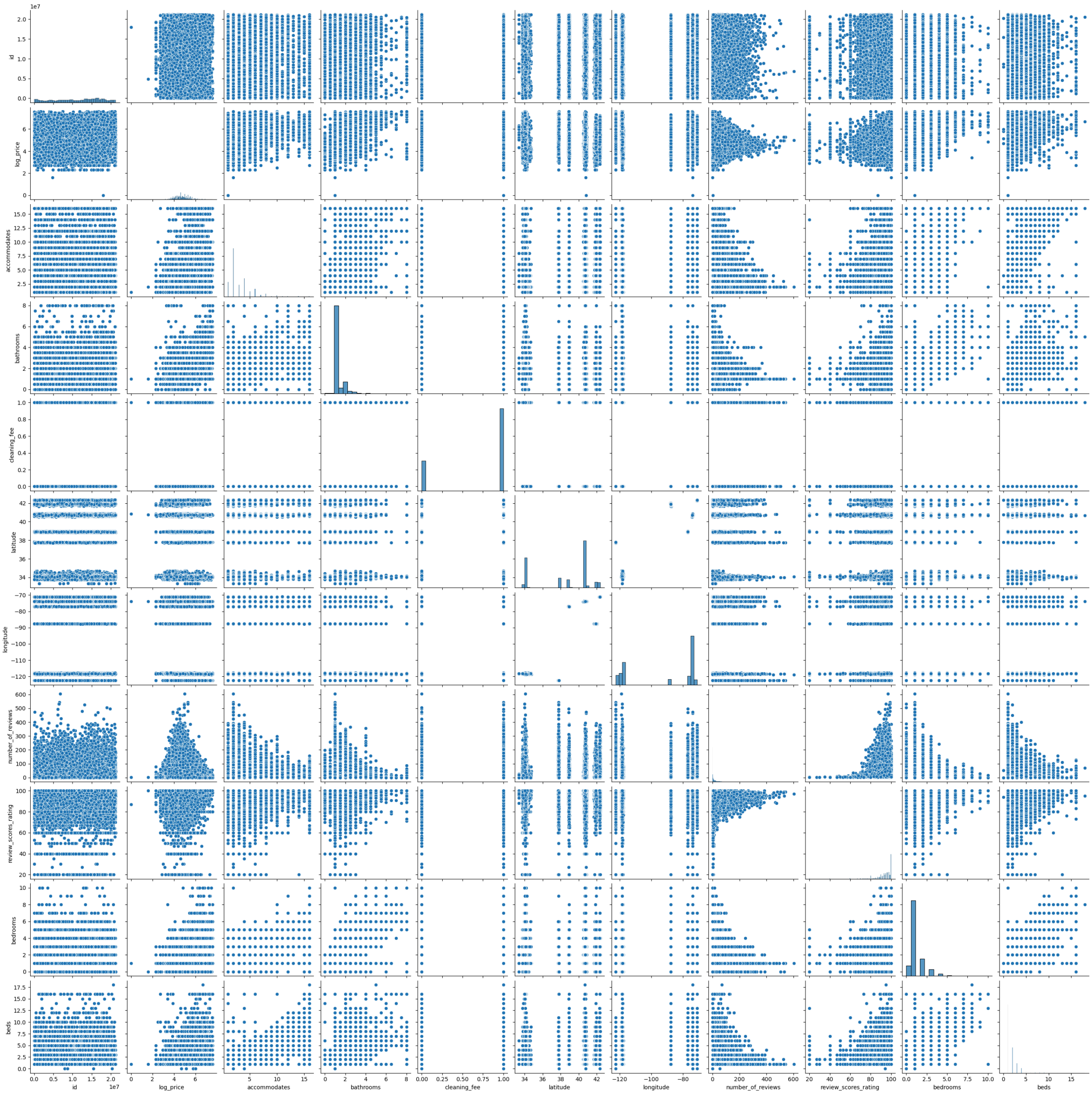
Out[142…

	id	log_price	accommodates	bathrooms	latitude	long
count	7.411100e+04	74111.000000	74111.000000	73911.000000	74111.000000	74111.0
mean	1.126662e+07	4.782069	3.155146	1.235263	38.445958	-92.3
std	6.081735e+06	0.717394	2.153589	0.582044	3.080167	21.7
min	3.440000e+02	0.000000	1.000000	0.000000	33.338905	-122.5
25%	6.261964e+06	4.317488	2.000000	1.000000	34.127908	-118.3
50%	1.225415e+07	4.709530	2.000000	1.000000	40.662138	-76.9
75%	1.640226e+07	5.220356	4.000000	1.000000	40.746096	-73.9
max	2.123090e+07	7.600402	16.000000	8.000000	42.390437	-70.9

Analyze the dataset for trends

In [144…

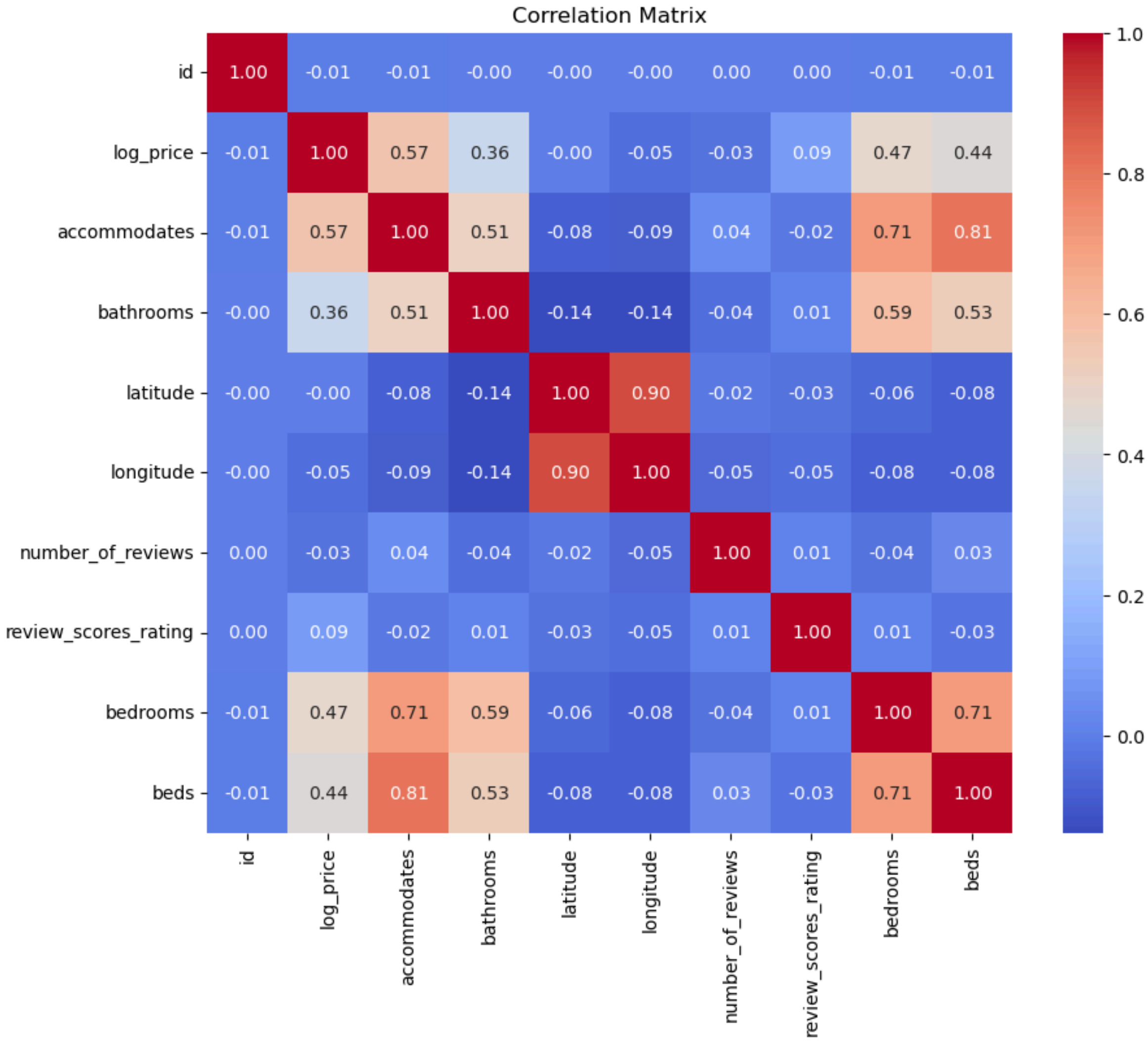
```
#pair plot visualizes relationships between variables.
sns.pairplot(df)#data
plt.show()
```



In [145…

```
# correlation matrix
corr_matrix = df[['id','log_price','accommodates','bathrooms','latitude','longit
                   'review_scores_rating','bedrooms','beds']].corr()

# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```



Missing Values

In [147…

```
#missing values in columns
df.isnull().sum()
```

Out[147…

```
id                           0
log_price                    0
property_type                0
room_type                    0
amenities                    0
accommodates                 0
bathrooms                  200
bed_type                     0
cancellation_policy          0
cleaning_fee                 0
city                         0
description                  0
first_review             15864
host_has_profile_pic       188
host_identity_verified     188
host_response_rate       18299
host_since                 188
instant_bookable             0
last_review              15827
latitude                     0
longitude                    0
name                         0
neighbourhood             6872
number_of_reviews            0
review_scores_rating     16722
thumbnail_url             8216
zipcode                    968
bedrooms                    91
beds                       131
dtype: int64
```

Data cleaning

In [149…

```
# columns left with missing values float64(contineous or Numeric). so, replaced
columns_to_fill = df[['bathrooms','review_scores_rating', 'bedrooms', 'beds']]

# Replace NaN values with mean
for column in columns_to_fill:
    mean = df[column].mean()  # Find  mean value
    df[column].replace(np.NaN, mean, inplace=True)
    df[column]=df[column].astype('int64')
```

In [150…

```
#Columns which are Object(categorical) going to replace 'NaN' with most 'Frequen

columns_to_fill = df[['first_review','host_has_profile_pic', 'host_identity_veri
       'last_review', 'neighbourhood', 'thumbnail_url',
       'zipcode']]

# Replace NaN values with the most freq value
for column in columns_to_fill:
    freq = df[column].value_counts().idxmax()  # Find the most frequent value
    df[column].replace(np.NaN, freq, inplace=True)
```

In [151…

```
# convert Boolean (True or False) to binary form
df['cleaning_fee']=df['cleaning_fee'].astype('int64')

#By using map()
convert_columns = df[['host_has_profile_pic', 'host_identity_verified','instant_
```

```
for column in convert_columns:
    df[column] = df[column].map({'t': 1, 'f': 0})
```

In [152…

```
#checking missing values
df.isnull().sum()
```

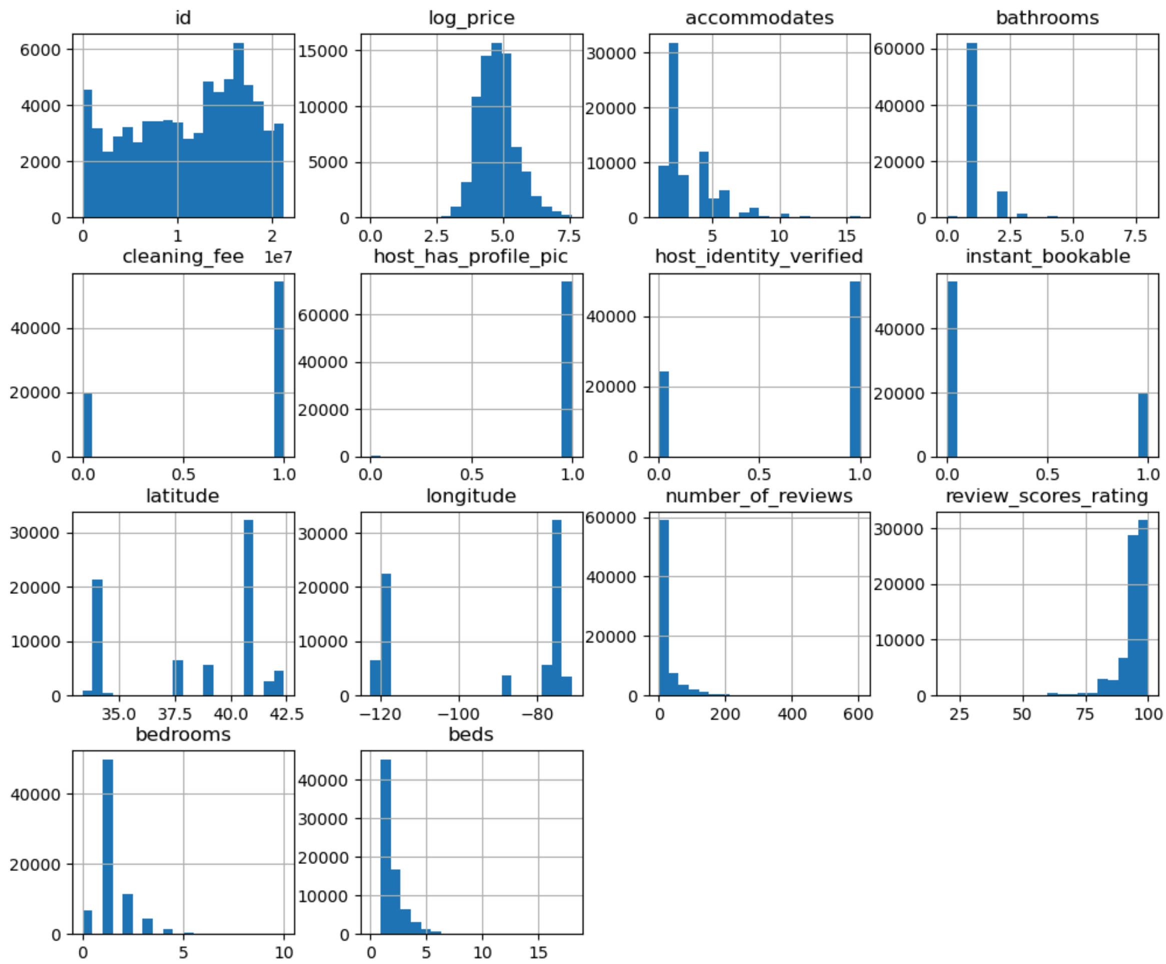
Out[152…

```
id                        0
log_price                 0
property_type             0
room_type                 0
amenities                 0
accommodates              0
bathrooms                 0
bed_type                  0
cancellation_policy       0
cleaning_fee              0
city                      0
description               0
first_review              0
host_has_profile_pic      0
host_identity_verified    0
host_response_rate        0
host_since                0
instant_bookable          0
last_review               0
latitude                  0
longitude                 0
name                      0
neighbourhood             0
number_of_reviews         0
review_scores_rating      0
thumbnail_url             0
zipcode                   0
bedrooms                  0
beds                      0
dtype: int64
```

Outliers

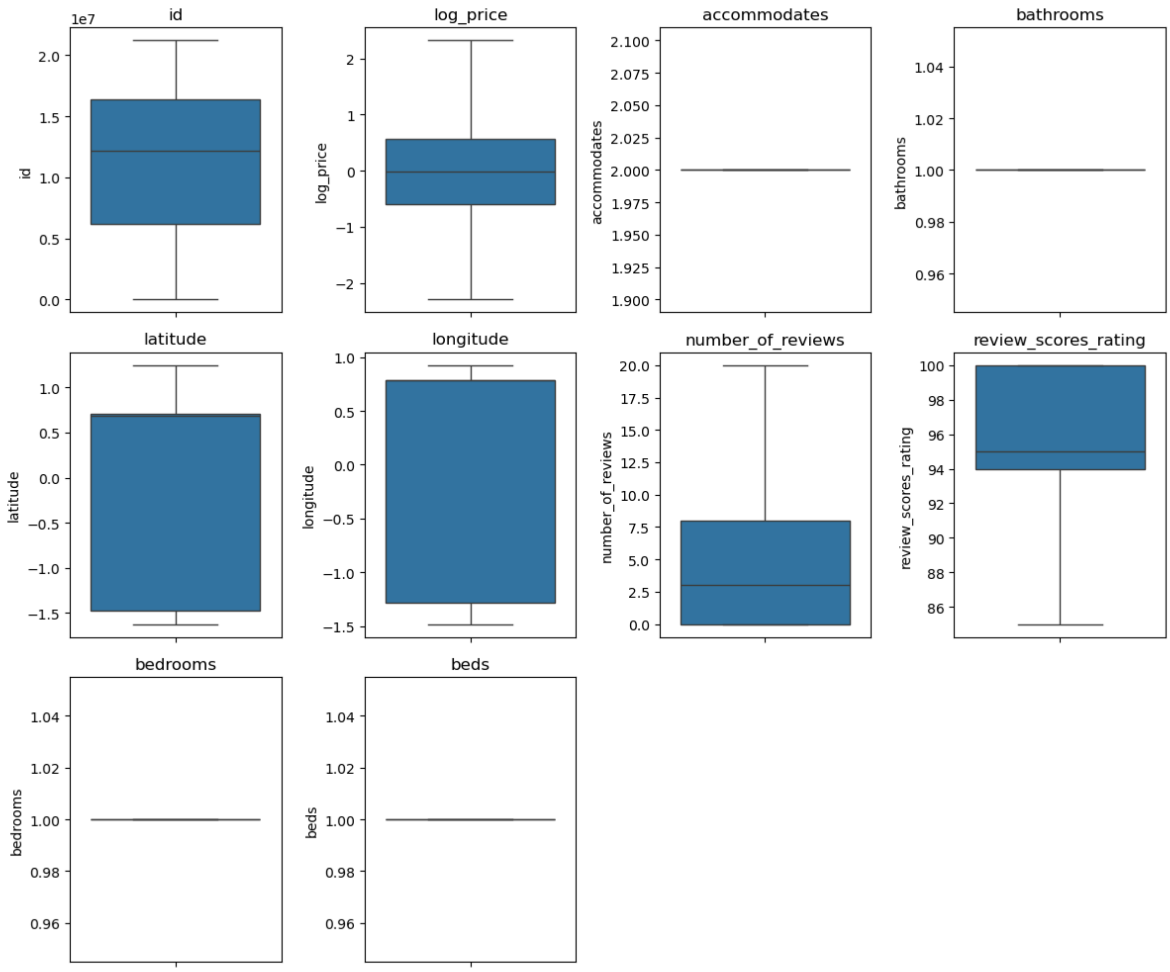
In [154…

```
# histogram for features.
df.hist(figsize=(12,10),bins=20)
plt.show()
```



In [216…

```
# Box plot for outliers.
plt.figure(figsize=(12, 10))
for i,column in enumerate(df[['id','log_price','accommodates','bathrooms','latit
                              'review_scores_rating','bedrooms','beds']]):
    plt.subplot(3,4,i+1)
    sns.boxplot(df[column])
    plt.title(column)
plt.tight_layout()
plt.show()
```



Remove outliers using IQR method

In [214...

```
# Remove outliers using IQR method
column = df[['id','log_price','accommodates','bathrooms','latitude','longitude',
                         'review_scores_rating','bedrooms','beds']]
for col in column:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    df = df[~((df[col] < (Q1 - 1.5 * IQR)) | (df[col] > (Q3 + 1.5 * IQR)))]
```

Feature Engineering

In [159...

```
#converting amenities to numeric by length
df['amenities'] = df['amenities'].apply(lambda x: len(x.split(',')))
```

Transformations.

In [161...

```
#Encoding
df = pd.get_dummies(df, columns=['room_type'], dtype=int)
```

In [162...

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['host_response_rate'] = le.fit_transform(df['host_response_rate'])
```

In [163…
```
le = LabelEncoder()
df['property_type'] = le.fit_transform(df['property_type'])
```

In [164…
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df[['log_price', 'latitude', 'longitude']] = scaler.fit_transform(df[['log_price
```

In [165…
```
df.columns
```

Out[165…
```
Index(['id', 'log_price', 'property_type', 'amenities', 'accommodates',
       'bathrooms', 'bed_type', 'cancellation_policy', 'cleaning_fee', 'city',
       'description', 'first_review', 'host_has_profile_pic',
       'host_identity_verified', 'host_response_rate', 'host_since',
       'instant_bookable', 'last_review', 'latitude', 'longitude', 'name',
       'neighbourhood', 'number_of_reviews', 'review_scores_rating',
       'thumbnail_url', 'zipcode', 'bedrooms', 'beds',
       'room_type_Entire home/apt', 'room_type_Private room',
       'room_type_Shared room'],
      dtype='object')
```

Model Development

Build a regression model to predict listing prices.

In [218…
```
#define multiple predicator
x_multi = df[['property_type','amenities','accommodates','bathrooms','cleaning_f
    'host_response_rate','instant_bookable','latitude','longitude','number_of_re
    'bedrooms','beds','room_type_Entire home/apt','room_type_Private room','room
y_multi = df['log_price']
```

In [220…
```
# Split the dataset into training and testing sets
x_train_multi,x_test_multi,y_train_multi,y_test_multi = train_test_split(x_multi
                                                                         test_siz
```

In [222…
```
#create and traun the model
multi_model = LinearRegression()
multi_model.fit(x_train_multi,y_train_multi)
```

Out[222…
```
▾ LinearRegression
LinearRegression()
```

In [224…
```
#predication
y_pred_multi = multi_model.predict(x_test_multi)

#Evaluate the model
print(f"Mean Squared Error: {mean_squared_error(y_test_multi,y_pred_multi)}")
print(f"R^ Score: {r2_score(y_test_multi,y_pred_multi)}")
```

```
Mean Squared Error: 0.5339284962033279
R^ Score: 0.31575991152572835
```

In [226…
```
# Display the coefficients of the model
coefficients = pd.DataFrame(multi_model.coef_.flatten(), x_multi.columns, column
```

```python
print(coefficients)
```

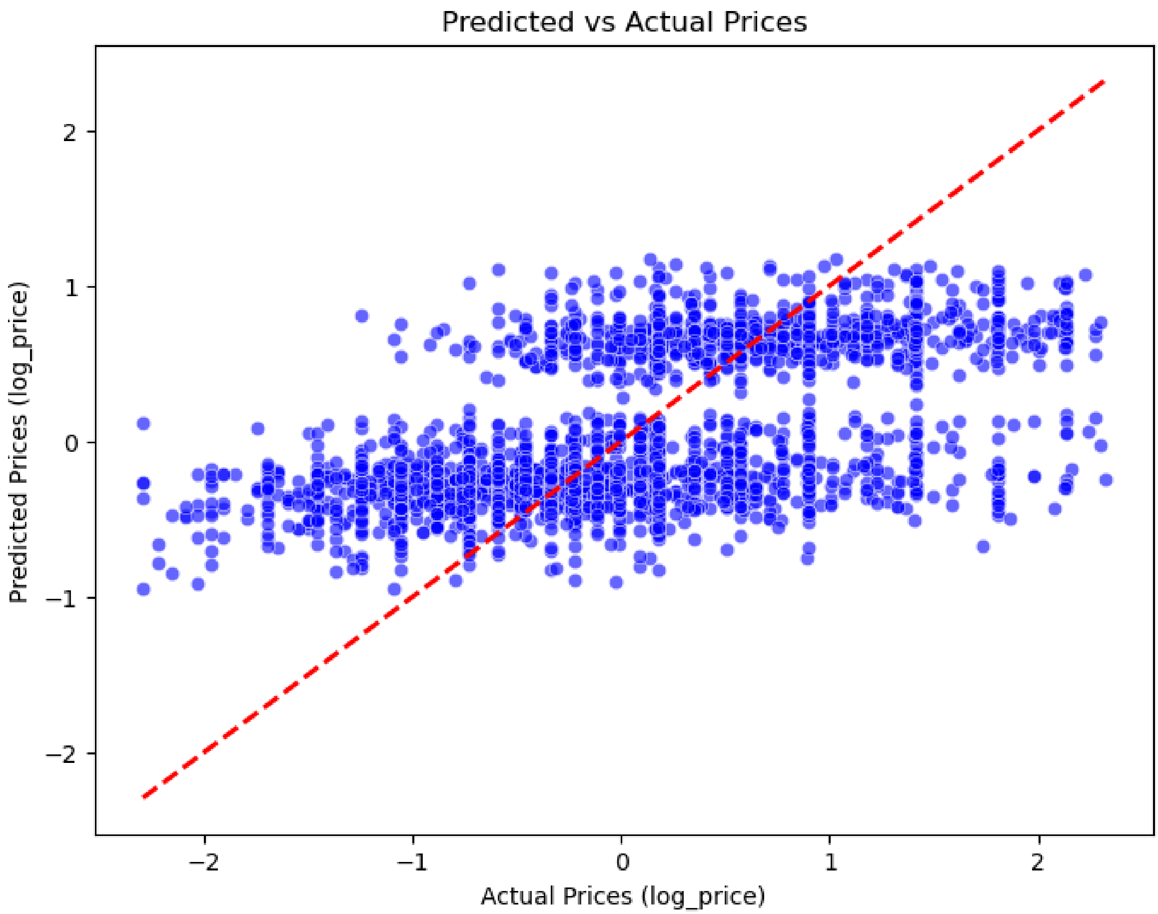
```
                           Coefficient
property_type              -1.263407e-03
amenities                   9.978617e-03
accommodates                1.110223e-16
bathrooms                  -3.885781e-16
cleaning_fee               -2.234336e-02
host_has_profile_pic       -4.732447e-02
host_identity_verified      2.380934e-03
host_response_rate         -6.856752e-04
instant_bookable           -1.160430e-01
latitude                    2.748607e-01
longitude                  -2.446982e-01
number_of_reviews          -6.448629e-03
review_scores_rating        4.702135e-03
bedrooms                    0.000000e+00
beds                        0.000000e+00
room_type_Entire home/apt   7.586729e-01
room_type_Private room     -1.895214e-01
room_type_Shared room      -5.691515e-01
number_of_reviews          -6.448629e-03
```

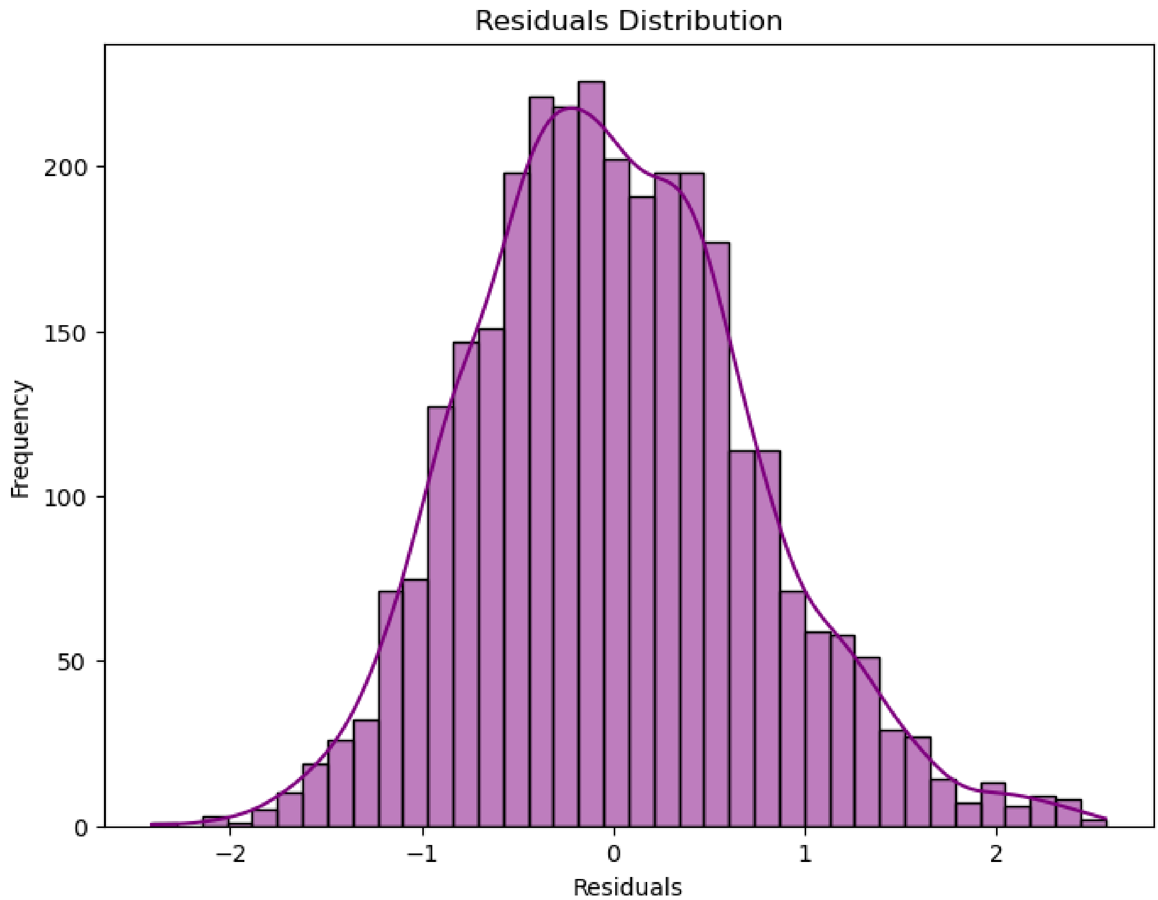
In [228…

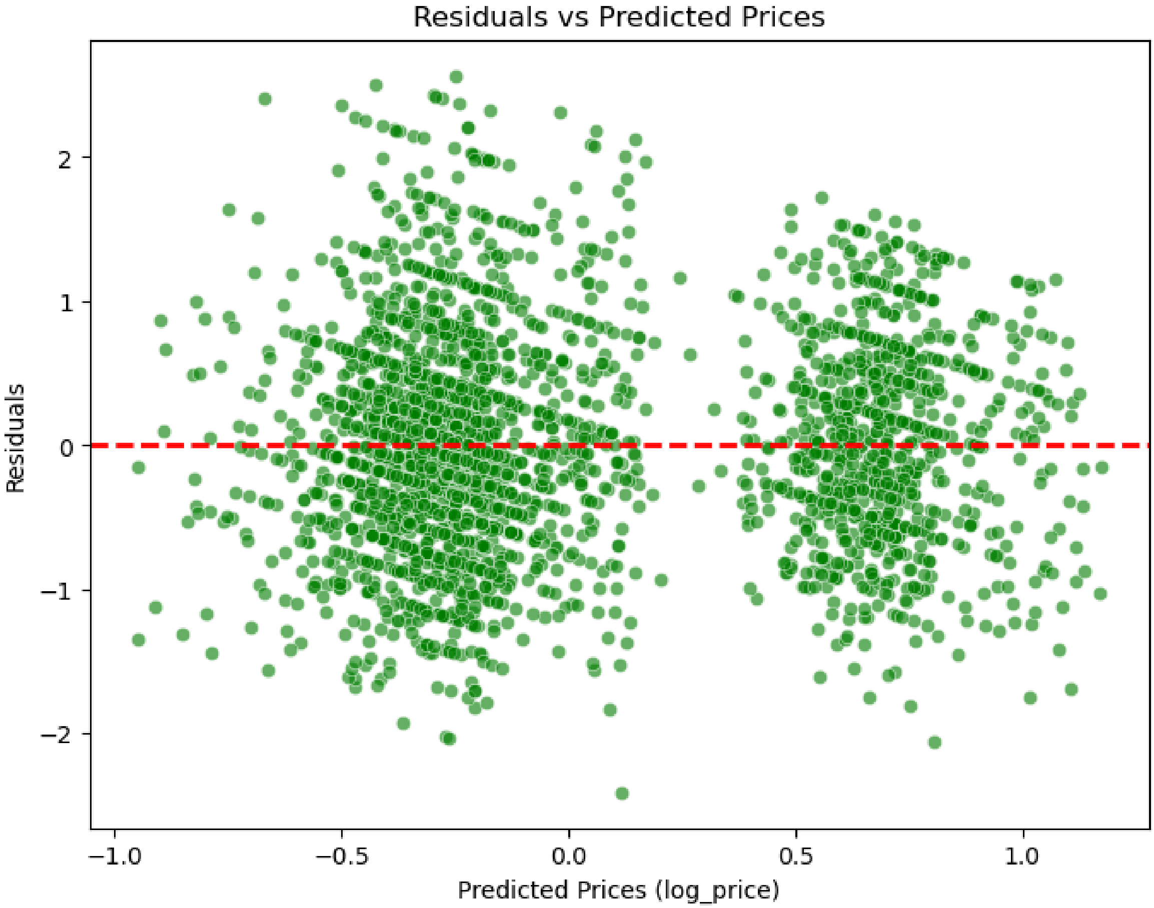
```python
# Predicted vs. Actual Plot
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test_multi, y=y_pred_multi, alpha=0.6, color='blue')
plt.plot([y_test_multi.min(), y_test_multi.max()], [y_test_multi.min(), y_test_m
         color='red', linestyle='--', linewidth=2)
plt.title('Predicted vs Actual Prices')
plt.xlabel('Actual Prices (log_price)')
plt.ylabel('Predicted Prices (log_price)')
plt.show()

# Residuals Plot
residuals = y_test_multi - y_pred_multi
plt.figure(figsize=(8, 6))
sns.histplot(residuals, kde=True, color='purple')
plt.title('Residuals Distribution')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.show()

# Residuals vs. Predicted Plot
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_pred_multi, y=residuals, alpha=0.6, color='green')
plt.axhline(0, color='red', linestyle='--', linewidth=2)
plt.title('Residuals vs Predicted Prices')
plt.xlabel('Predicted Prices (log_price)')
plt.ylabel('Residuals')
plt.show()
```


Predicted vs Actual Prices
Predicted Prices (log_price)
Actual Prices (log_price)
2
1
0
−1
−2
−2
−1
0
1
2


Residuals Distribution
Frequency
Residuals
200
150
100
50
0
−2
−1
0
1
2


Residuals vs Predicted Prices
Residuals
Predicted Prices (log_price)
2
1
0
−1
−2
−1.0
−0.5
0.0
0.5
1.0

In [ ]:

In [ ]: