```
# Data manipulation and analysis
import pandas as pd
import numpy as np

# Data visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Machine learning
from sklearn.model_selection import train_test_split
from skle koarn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, r2_score
from sklearn.cluster import KMeans

# Other useful libraries
import os
import sys
import random
```

```
df = pd.read_excel("/content/airbnb hotel.xlsx")  # replace with your file
df.head(5)
```

| | id | NAME | host id | host_identity_verified | host name | neighbourhood group | neighbourhc |
|---|---|---|---|---|---|---|---|
| **0** | 1001254 | Clean & quiet apt home by the park | 80014485718 | unconfirmed | Madaline | Brooklyn | Kensing |
| **1** | 1002102 | Skylit Midtown Castle | 52335172823 | verified | Jenna | Manhattan | Midto |
| **2** | 1002403 | THE VILLAGE OF HARLEM....NEW YORK ! | 78829239556 | NaN | Elise | Manhattan | Harl |
| **3** | 1002755 | NaN | 85098326012 | unconfirmed | Garry | Brooklyn | Clinton |
| **4** | 1003689 | Entire Apt: Spacious Studio/Loft by central park | 92037596077 | verified | Lyndon | Manhattan | East Harl |

5 rows × 26 columns

```
print(df.columns)
```

```
Index(['id', 'NAME', 'host id', 'host_identity_verified', 'host name',
       'neighbourhood group', 'neighbourhood', 'lat', 'long', 'country',
       'country code', 'instant_bookable', 'cancellation_policy', 'room type',
       'Construction year', 'price', 'service fee', 'minimum nights',
       'number of reviews', 'last review', 'reviews per month',
       'review rate number', 'calculated host listings count',
       'availability 365', 'house_rules', 'license'],
      dtype='object')
```

```
import pandas as pd
import numpy as np

from sklearn.model_selection import trai._.est_split
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
# Example: if your dataset is a CSV
# df = pd.read_csv("airbnb.csv")
```

```
features = ["number of reviews", "minimum nights", "calculated host listings count", "availabil
target = "price"

X = df[features]   # Independent variables
y = df[target]     # Dependent variable
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (82079, 4)
X_test shape: (20520, 4)
y_train shape: (82079,)
y_test shape: (20520,)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Drop rows with NaN values in the features or target
df_cleaned = df.dropna(subset=features + [target])

# Re-split the data after dropping rows
X_cleaned = df_cleaned[features]
y_cleaned = df_cleaned[target]

X_train, X_test, y_train, y_test = train_test_split(
    X_cleaned, y_cleaned, test_size=0.2, random_state=42
)

model = LinearRegression()
model.fit(X_train, y_train)
```

▾ LinearRegression

```
LinearRegression()
```

```
y_pred = model.predict(X_test)
print(y_pred[:5])
```

```
[625.23846407 625.23538528 625.25657125 623.76180366 624.91423808]
```

```
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("MAE:", mae)
print("R² Score:", r2)
```

```
MAE: 287.4488378740061
R² Score: -2.0479230028458417e-05
```

```
example = X_test.iloc[0:1]
pred_example = model.predict(example)
```

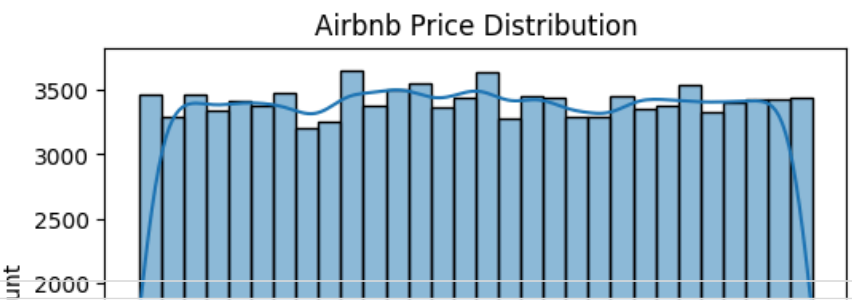
```
print("Example input:", example.to_dict(orient='records')[0])
print("Predicted Price:", pred_example[0])
```

```
Example input: {'number of reviews': 1.0, 'minimum nights': 1.0, 'calculated host listings coun
Predicted Price: 625.2384640714068
```

```
import joblib

joblib.dump(model, "airbnb_price_model.joblib")
```

```
['airbnb_price_model.joblib']
```

```
plt.figure(figsize=(6,4))
sns.histplot(df['price'], bins=30, kde=True)
plt.title("Airbnb Price Distribution")
plt.xlabel("Price")
plt.ylabel("Count")
plt.show()
```




```
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

rf_pred = rf_model.predict(X_test)

rf_mae = mean_absolute_error(y_test, rf_pred)
rf_r2 = r2_score(y_test, rf_pred)

print("Random Forest MAE:", rf_mae)
print("Random Forest R²:", rf_r2)
```

```
Random Forest MAE: 246.21853290547128
Random Forest R²: 0.11930229884939647
```

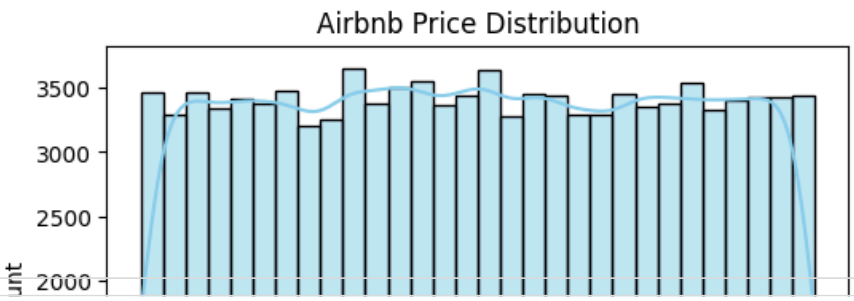
```
importances = rf_model.feature_importances_
feature_importance = pd.DataFrame({'Feature': features, 'Importance': importances})
print(feature_importance.sort_values(by="Importance", ascending=False))
```

```
                        Feature  Importance
3               availability 365    0.414301
0              number of reviews    0.303347
1                 minimum nights    0.160832
2  calculated host listings count    0.121520
```

```
# Save
joblib.dump(rf_model, "airbnb_rf_model.joblib")

# Load back
loaded_model = joblib.load("airbnb_rf_model.joblib")

# Test prediction
print("Reloaded model prediction:", loaded_model.predict(X_test.iloc[0:1]))
```

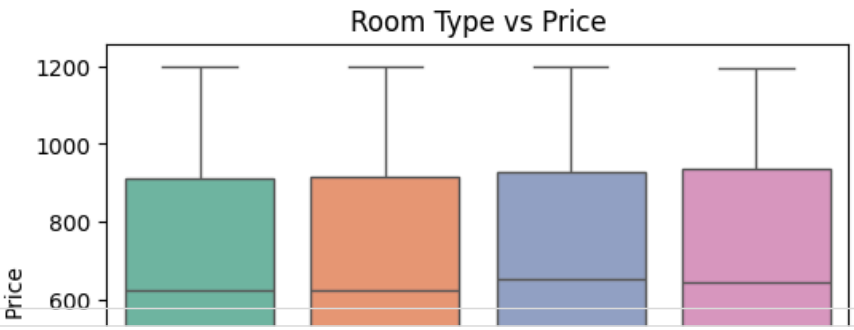
```
Reloaded model prediction: [628.9047088]
```

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(6,4))
sns.histplot(df['price'], bins=30, kde=True, color="skyblue")
plt.title("Airbnb Price Distribution")
plt.xlabel("Price")
plt.ylabel("Count")
plt.show()
```

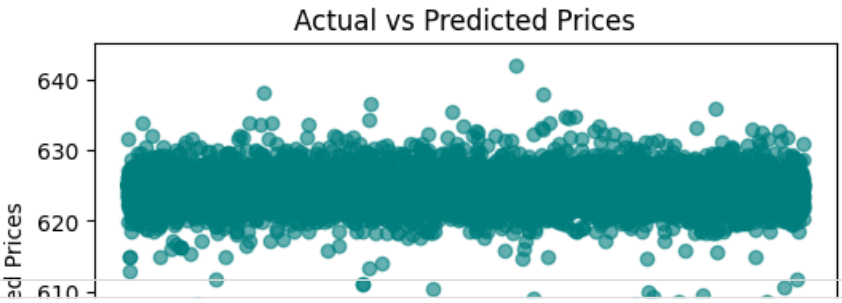



```
plt.figure(figsize=(6,4))
sns.boxplot(x="room type", y="price", data=df, palette="Set2")
plt.title("Room Type vs Price")
plt.xlabel("Room Type")
plt.ylabel("Price")
plt.show()
```

```
/tmp/ipython-input-172952402.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign

  sns.boxplot(x="room type", y="price", data=df, palette="Set2")
```



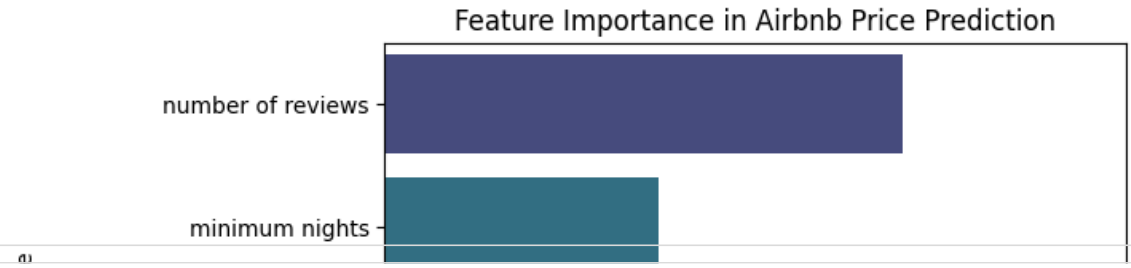

```
plt.figure(figsize=(6,4))
plt.scatter(y_test, y_pred, alpha=0.6, color="teal")
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted Prices")
plt.show()
```




```
importances = rf_model.feature_importances_
feature_names = features

plt.figure(figsize=(6,4))
sns.barplot(x=importances, y=feature_names, palette="viridis")
plt.title("Feature Importance in Airbnb Price Prediction")
plt.xlabel("Importance")
```

```
plt.ylabel("Feature")
plt.show()
```

```
/tmp/ipython-input-2714944744.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign

  sns.barplot(x=importances, y=feature_names, palette="viridis")
```

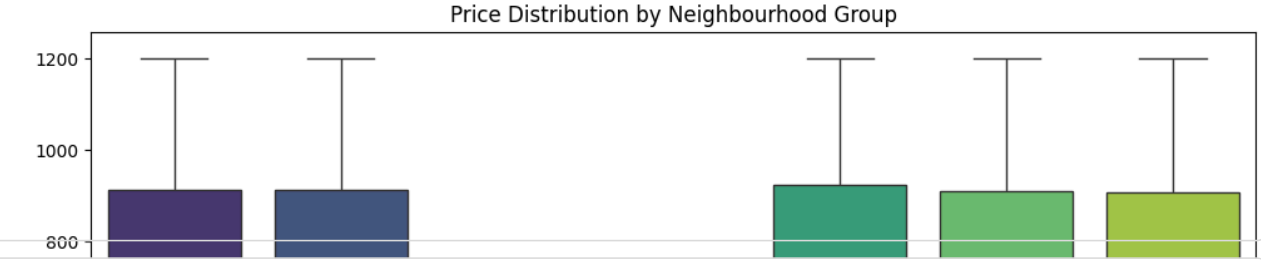



```
# Group by neighbourhood_group and room_type, then calculate mean of reviews
avg_reviews = df.groupby(['neighbourhood group', 'room type'])['reviews per month'].mean().res

print(avg_reviews)
```

```
   neighbourhood group        room type  reviews per month
0                Bronx  Entire home/apt           2.054692
1                Bronx     Private room           1.572678
2                Bronx      Shared room           1.262889
3             Brooklyn  Entire home/apt           1.368625
4             Brooklyn       Hotel room           0.456250
5             Brooklyn     Private room           1.251082
6             Brooklyn      Shared room           1.056677
7            Manhattan  Entire home/apt           1.108236
8            Manhattan       Hotel room           3.214600
9            Manhattan     Private room           1.464696
10           Manhattan      Shared room           1.769502
11              Queens  Entire home/apt           1.890681
12              Queens       Hotel room           0.602500
13              Queens     Private room           1.805396
14              Queens      Shared room           1.409135
15       Staten Island  Entire home/apt           2.017182
16       Staten Island     Private room           1.417664
17       Staten Island      Shared room           0.880000
18             brookln     Private room           1.340000
19            manhatan     Private room           2.120000
```

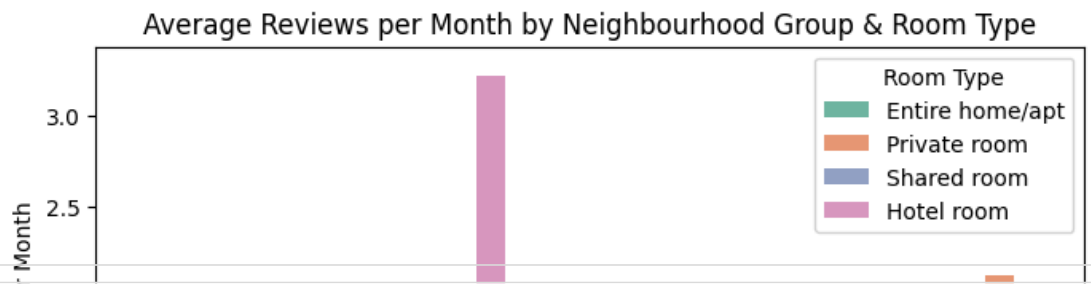
```
plt.figure(figsize=(12, 6))
sns.boxplot(x="neighbourhood group", y="price", data=df, palette="viridis")
plt.title("Price Distribution by Neighbourhood Group")
plt.xlabel("Neighbourhood Group")
plt.ylabel("Price")
plt.show()
```

```
/tmp/ipython-input-1090576545.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign

  sns.boxplot(x="neighbourhood group", y="price", data=df, palette="viridis")
```



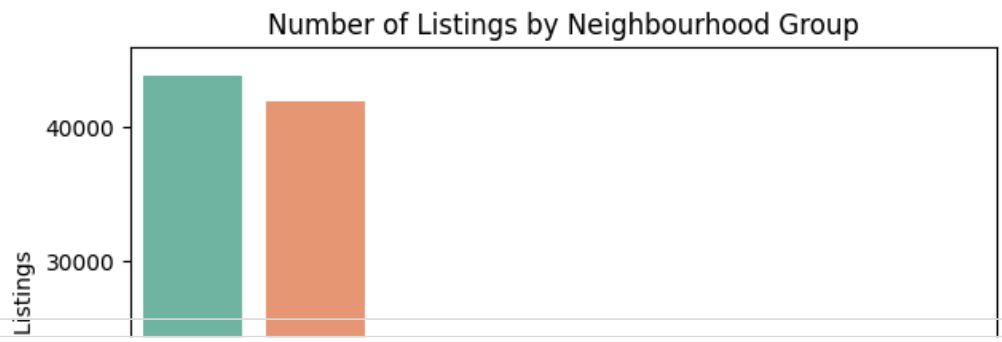

```
plt.figure(figsize=(8,5))
sns.barplot(x="neighbourhood group",
            y="reviews per month",
            hue="room type",
            data=avg_reviews,
```

```
            palette="Set2")

plt.title("Average Reviews per Month by Neighbourhood Group & Room Type")
plt.xlabel("Neighbourhood Group")
plt.ylabel("Average Reviews per Month")
plt.legend(title="Room Type")
plt.show()
```

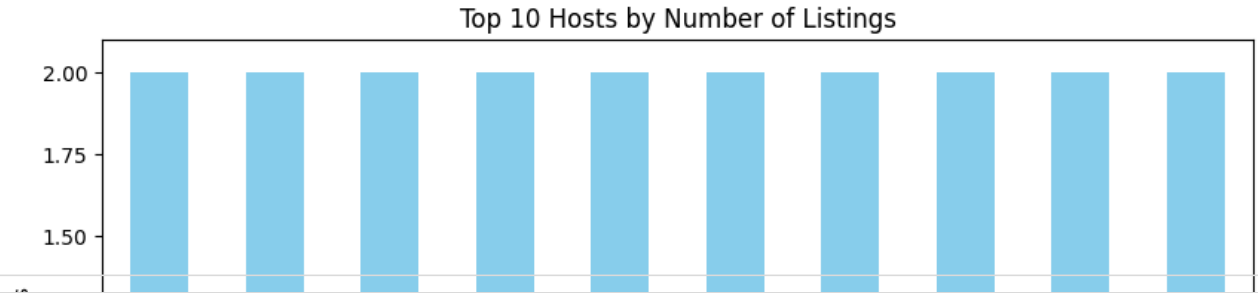



```
plt.figure(figsize=(7,5))
sns.countplot(
    x="neighbourhood group",
    data=df,
    palette="Set2",
    order=df['neighbourhood group'].value_counts().index
)

plt.title("Number of Listings by Neighbourhood Group")
plt.xlabel("Neighbourhood Group")
plt.ylabel("Number of Listings")
plt.show()
```

```
/tmp/ipython-input-4106682205.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign

  sns.countplot(
```




```
# Calculate the top 10 hosts by number of listings
top_hosts = df['host id'].value_counts().nlargest(10)

plt.figure(figsize=(10,6))
top_hosts.plot(kind="bar", color="skyblue")

plt.title("Top 10 Hosts by Number of Listings")
plt.xlabel("Host ID")
plt.ylabel("Number of Listings")
plt.xticks(rotation=45)
plt.show()
```



```
# Check for missing values
missing_values = df.isnull().sum()
print("Missing values in each column:")
print(missing_values[missing_values > 0])

# --- Examples of handling missing values ---

# Example 1: Imputing missing values in a numerical column with the median
# (Choose a relevant numerical column with missing values, e.g., 'reviews per month' if it has
if 'reviews per month' in df.columns and df['reviews per month'].isnull().any():
    df['reviews per month'].fillna(df['reviews per month'].median(), inplace=True)
    print("\nMissing values in 'reviews per month' after median imputation:", df['reviews per

# Example 2: Imputing missing values in a categorical column with the mode
# (Choose a relevant categorical column with missing values, e.g., 'host_identity_verified')
if 'host_identity_verified' in df.columns and df['host_identity_verified'].isnull().any():
    df['host_identity_verified'].fillna(df['host_identity_verified'].mode()[0], inplace=True)
    print("Missing values in 'host_identity_verified' after mode imputation:", df['host_identi

# Example 3: Dropping columns with a high percentage of missing values (e.g., > 50%)
# You can adjust the threshold as needed
threshold = len(df) * 0.5
df_dropped_cols = df.dropna(axis=1, thresh=threshold)
print("\nShape of DataFrame after dropping columns with > 50% missing values:", df_dropped_col

# Example 4: Dropping rows with missing values in specific columns
# (You've already done this for features and target in a previous step, but here's a general e
# df_dropped_rows = df.dropna(subset=['column1', 'column2'])
# print("\nShape of DataFrame after dropping rows with missing values in specific columns:", d
```

```
Missing values in each column:
NAME                              270
host_identity_verified            289
host name                         408
neighbourhood group                29
neighbourhood                      16
lat                                 8
long                                8
country                           532
country code                      131
instant_bookable                  105
cancellation_policy                76
Construction year                 214
price                             247
service fee                       273
minimum nights                    409
number of reviews                 183
last review                     15893
reviews per month               15879
review rate number                326
calculated host listings count    319
availability 365                  448
house_rules                     54843
license                        102597
dtype: int64

Missing values in 'reviews per month' after median imputation: 0
Missing values in 'host_identity_verified' after mode imputation: 0

Shape of DataFrame after dropping columns with > 50% missing values: (102599, 24)
/tmp/ipython-input-1853974400.py:11: FutureWarning: A value is trying to be set on a copy of a
The behavior will change in pandas 3.0. This inplace method will never work because the interme

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value


  df['reviews per month'].fillna(df['reviews per month'].median(), inplace=True)
/tmp/ipython-input-1853974400.py:17: FutureWarning: A value is trying to be set on a copy of a
The behavior will change in pandas 3.0. This inplace method will never work because the interme

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value
```

```
df['host_identity_verified'].fillna(df['host_identity_verified'].mode()[0], inplace=True)
```

```
print(df.columns)
```

```
Index(['id', 'NAME', 'host id', 'host_identity_verified', 'host name',
       'neighbourhood group', 'neighbourhood', 'lat', 'long', 'country',
       'country code', 'instant_bookable', 'cancellation_policy', 'room type',
       'Construction year', 'price', 'service fee', 'minimum nights',
       'number of reviews', 'last review', 'reviews per month',
       'review rate number', 'calculated host listings count',
       'availability 365', 'house_rules', 'license'],
      dtype='object')
```

```
# Count of listings by Room Type
plt.figure(figsize=(8, 5))
sns.countplot(x="room type", data=df, palette="Set2")
plt.title("Number of Listings by Room Type")
plt.xlabel("Room Type")
plt.ylabel("Number of Listings")
plt.show()
```

```
/tmp/ipython-input-4066182852.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign

  sns.countplot(x="room type", data=df, palette="Set2")
```