

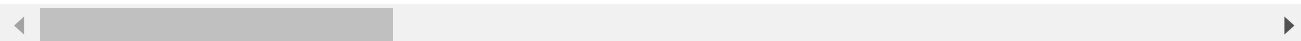
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
datafile = pd.read_csv("Airbnb_data - airbnb_data.csv")
datafile.head()
```



	id	log_price	property_type	room_type	amenities	accommodates	b
0	6901257	5.010635	Apartment	Entire home/apt	{"Wireless Internet","Air conditioning",Kitch...	3	
1	6304928	5.129899	Apartment	Entire home/apt	{"Wireless Internet","Air conditioning",Kitch...	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



```
datafile.info()
```



```
<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

```

```
datafile.describe()
```



	id	log_price	accommodates	bathrooms	latitude	longitude
count	7.411100e+04	74111.000000	74111.000000	73911.000000	74111.000000	74111.000000
mean	1.126662e+07	4.782069	3.155146	1.235263	38.445958	-92.39752
std	6.081735e+06	0.717394	2.153589	0.582044	3.080167	21.70532
min	3.440000e+02	0.000000	1.000000	0.000000	33.338905	-122.51150
25%	6.261964e+06	4.317488	2.000000	1.000000	34.127908	-118.34237
50%	1.225415e+07	4.709530	2.000000	1.000000	40.662138	-76.99696
75%	1.640226e+07	5.220356	4.000000	1.000000	40.746096	-73.95466
max	2.123090e+07	7.600402	16.000000	8.000000	42.390437	-70.98504

```

#replacing the missing values with na
datafile.fillna(datafile.mean(),inplace=True)
datafile['new_feature'] = datafile['existing_feature1'] / datafile['existing_feature2']

#scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
datafile[['feature1', 'feature2']] = scaler.fit_transform(datafile[['feature1', 'feature2']])

#training model test division
from sklearn.model_selection import train_test_split

X = datafile.drop('price', axis=1)
y = datafile['price']

```

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

# Developing the model
from sklearn.linear_model import LinearRegression

# Create a model instance
model = LinearRegression()

# Fit the model
model.fit(X_train, y_train)

# Make predictions
predictions = model.predict(X_test)

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

print('RMSE:', mean_squared_error(y_test, predictions, squared=False))
print('MAE:', mean_absolute_error(y_test, predictions))
print('R²:', r2_score(y_test, predictions))

# Visualization
plt.scatter(y_test, predictions)
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual Prices vs Predicted Prices')
plt.show()
```