import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

# Load the dataset

try:

    data = pd.read\_csv('/content/data.csv')

except FileNotFoundError:

    print("Error: '/content/data.csv' not found. Please upload the file to your Colab environment.")

    exit()

# Handle missing values (replace with mean for numerical features)

numerical\_cols = data.select\_dtypes(include=['number']).columns

for col in numerical\_cols:

    data[col].fillna(data[col].mean(), inplace=True)

# Feature Selection

features = ['bedrooms', 'bathrooms', 'sqft\_living', 'floors', 'condition', 'sqft\_above', 'sqft\_basement', 'yr\_built']

target = 'price'

X = data[features]

y = data[target]

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Predict

y\_pred = model.predict(X\_test)

# Evaluate

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = mse\*\*0.5

# print(f"Root Mean Squared Error: {rmse:.2f}")

# Get input from user

input\_values = []

for feature in features:

    value = float(input(f"Enter value for {feature}: "))

    input\_values.append(value)

# Create DataFrame with the user input

new\_house\_features = pd.DataFrame([input\_values], columns=features)

# Predict using the trained model

predicted\_price = model.predict(new\_house\_features)

# Output the result

print(f"\nPredicted Price for new house: ${predicted\_price[0]:,.2f}")

usd\_to\_inr = 83.54

# Print predicted price in INR

print(f"\nPredicted Price for new house: ₹{predicted\_price[0] \* usd\_to\_inr:,.2f}")

# # Predict new house

# new\_house\_features = pd.DataFrame([[3, 2, 1600, 1, 4, 1500, 100, 1995]], columns=features)

# predicted\_price = model.predict(new\_house\_features)

# print(f"Predicted Price for new house: ${predicted\_price[0]:,.2f}")

# =========================

# 📈 Visualization Section

# =========================

# 1. Feature vs Price scatter plots

plt.figure(figsize=(15, 10))

for i, feature in enumerate(features[:6]):

    plt.subplot(2, 3, i + 1)

    sns.scatterplot(x=data[feature], y=data['price'], alpha=0.5)

    plt.title(f'{feature} vs Price')

    plt.xlabel(feature)

    plt.ylabel('Price')

plt.tight\_layout()

plt.show()

# 2. Actual vs Predicted Prices

plt.figure(figsize=(8, 6))

sns.scatterplot(x=y\_test, y=y\_pred, alpha=0.6)

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], '--r')

plt.xlabel('Actual Price')

plt.ylabel('Predicted Price')

plt.title('Actual vs Predicted House Prices')

plt.show()

# 3. Residuals plot

residuals = y\_test - y\_pred

plt.figure(figsize=(8, 6))

sns.histplot(residuals, kde=True, bins=30)

plt.title('Residuals Distribution')

plt.xlabel('Prediction Error')

plt.ylabel('Frequency')

plt.show()