**Step 1: Upload CSV File**

from google.colab import files

uploaded = files.upload()

**Step 2: Load and Preview Dataset**

import pandas as pd

df = pd.read\_csv("House Price Prediction Dataset.csv")

df.head()

**Step 3: Import Plot Libraries**

import matplotlib.pyplot as plt

import seaborn as sns

# Optional: for better-looking graphs

sns.set(style="whitegrid")

plt.rcParams["figure.figsize"] = (10, 6)

**Step 4: Data Visualizations**

**1.Correlation Heatmap (Numerical Features)**

numerical\_cols = ["Area", "Bedrooms", "Bathrooms", "Floors", "YearBuilt", "Price"]

corr\_matrix = df[numerical\_cols].corr()

sns.heatmap(corr\_matrix, annot=True, cmap="coolwarm", fmt=".2f")

plt.title("Correlation Heatmap")

plt.show()

**2. Average House Price by Location**

avg\_price\_by\_location = df.groupby("Location")["Price"].mean().sort\_values(ascending=False)

avg\_price\_by\_location.plot(kind="bar", color="skyblue")

plt.title("Average House Price by Location")

plt.ylabel("Average Price")

plt.xlabel("Location")

plt.xticks(rotation=45)

plt.show()

**3. Boxplot: Price vs Bedrooms**

sns.boxplot(x="Bedrooms", y="Price", data=df, palette="Set3")

plt.title("House Price Distribution by Number of Bedrooms")

plt.show()

**Step 5: Preprocessing + Model + Prediction**

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

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

import numpy as np

# Define features and target

X = df.drop(columns=["Id", "Price"])

y = df["Price"]

# Categorical & numeric columns

categorical\_cols = ["Location", "Condition", "Garage"]

preprocessor = ColumnTransformer([

("cat", OneHotEncoder(drop="first"), categorical\_cols)

], remainder="passthrough")

# Pipeline

model = Pipeline([

("preprocessor", preprocessor),

("regressor", RandomForestRegressor(random\_state=42))

])

# Split data

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

# Train

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

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

# Evaluate

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

r2 = r2\_score(y\_test, y\_pred)

print("🏁 Model Performance:")

print(f"RMSE: ₹{rmse:,.0f}")

print(f"R² Score: {r2:.3f}")

**Step 6: Graph - Actual vs Predicted Prices**

results = pd.DataFrame({"Actual": y\_test.values, "Predicted": y\_pred})

sns.scatterplot(x="Actual", y="Predicted", data=results, color="purple")

plt.plot([results.Actual.min(), results.Actual.max()],

[results.Actual.min(), results.Actual.max()],

'r--', label="Perfect Prediction")

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

plt.legend()

plt.show()