1.upload the dataset

from google.colab import files

uploaded = files.upload()

2.Load the dataset

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

file\_path = "housing\_price\_dataset.csv"  # Updated file\_path to the uploaded file name

df = pd.read\_csv(file\_path)

df.head()

3.Data exploration

print(df.info())

print(df.describe())

4.Check for missing values and duplicates

print("Missing values:\n", df.isnull().sum())

print("Duplicate rows:", df.duplicated().sum())

5.Visualize a feww features

sns.histplot(df['Price'], kde=True)

plt.title('Distribution of House Prices')

plt.show()

sns.scatterplot(x='SquareFeet', y='Price', data=df)

plt.title('Price vs Square Feet')

plt.show()

6.Identify target and features

y = df['Price']

X = df.drop(['Price'], axis=1)

7.Convert categorical columns to numerical

X = pd.get\_dummies(X, drop\_first=True)

8.One-hot encoding

X = pd.get\_dummies(X, drop\_first=True)

9.Feature scaling

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

10.Train-test split

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

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

11.Model building

from sklearn.linear\_model import LinearRegression, Ridge, Lasso

from sklearn.ensemble import RandomForestRegressor

from xgboost import XGBRegressor

models = {

    'Linear Regression': LinearRegression(),

    'Ridge Regression': Ridge(),

    'Lasso Regression': Lasso(),

    'Random Forest': RandomForestRegressor(),

    'XGBoost': XGBRegressor()

}

12.Evaluation

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

for name, model in models.items():

    model.fit(X\_train, y\_train)

    y\_pred = model.predict(X\_test)

    print(f"\n{name} Evaluation:")

    print("MAE:", mean\_absolute\_error(y\_test, y\_pred))

    print("RMSE:", np.sqrt(mean\_squared\_error(y\_test, y\_pred)))

    print("R^2 Score:", r2\_score(y\_test, y\_pred))

13.Make predictions from new input

sample\_input = X\_test[0].reshape(1, -1)

predicted\_price = models['XGBoost'].predict(sample\_input)

print("\nPredicted Price for Sample Input:", predicted\_price[0])

16.Deployment -building an interactive app

!pip install gradio

import gradio as gr

17.Create a prediction function

def predict\_price(SquareFeet, Bedrooms, Bathrooms, YearBuilt, Neighborhood):

    df\_input = pd.DataFrame({

        'SquareFeet': [SquareFeet],

        'Bedrooms': [Bedrooms],

        'Bathrooms': [Bathrooms],

        'YearBuilt': [YearBuilt],

        'Neighborhood': [Neighborhood]

    })

    df\_encoded = pd.get\_dummies(df\_input)

    df\_encoded = df\_encoded.reindex(columns=X.columns, fill\_value=0)

    scaled\_input = scaler.transform(df\_encoded)

    pred = models['XGBoost'].predict(scaled\_input)

    return f"Predicted House Price: ${pred[0]:,.2f}"

18.Create the gradio interface

interface = gr.Interface(

    fn=predict\_price,

    inputs=[

        gr.Number(label="Square Feet"),

        gr.Number(label="Bedrooms"),

        gr.Number(label="Bathrooms"),

        gr.Number(label="Year Built"),

        gr.Radio(["Urban", "Suburb", "Rural"], label="Neighborhood")

    ],

    outputs="text"

)

interface.launch()