Soruce Code:

# === Upload and Load Data ===

from google.colab import files

import io

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

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

import numpy as np

# Prompt user to upload the CSV file

uploaded = files.upload()

filename = next(iter(uploaded))

df = pd.read\_csv(io.BytesIO(uploaded[filename]), encoding="latin-1")

# === Preprocessing ===

df = df.drop\_duplicates()

df = df.dropna()

df.columns = df.columns.str.lower()

df.reset\_index(drop=True, inplace=True)

# Encode categorical variables

categorical\_cols = df.select\_dtypes(include=['object']).columns

label\_encoders = {}

for col in categorical\_cols:

le = LabelEncoder()

df[col] = le.fit\_transform(df[col])

label\_encoders[col] = le

# === Exploratory Data Analysis ===

# 1. Correlation Matrix

plt.figure(figsize=(12, 8))

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

plt.title("Correlation Matrix")

plt.show()

# 2. Distribution of Target Variable (Price)

plt.figure(figsize=(10, 5))

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

plt.title("Distribution of House Prices")

plt.xlabel("Price")

plt.ylabel("Frequency")

plt.show()

# === Features and Target ===

X = df.drop("price", axis=1)

y = df["price"]

# === Feature Scaling ===

scaler = StandardScaler()

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

# === Train-Test Split ===

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X\_scaled, y, test\_size=0.2, random\_state=42

)

# === Model 1: Linear Regression ===

lr\_model = LinearRegression()

lr\_model.fit(X\_train, y\_train)

lr\_pred = lr\_model.predict(X\_test)

print("=== Linear Regression Results ===")

print(f"MAE: {mean\_absolute\_error(y\_test, lr\_pred):.2f}")

print(f"MSE: {mean\_squared\_error(y\_test, lr\_pred):.2f}")

print(f"RMSE: {np.sqrt(mean\_squared\_error(y\_test, lr\_pred)):.2f}")

print(f"R²: {r2\_score(y\_test, lr\_pred):.2f}\n")

# === Model 2: Random Forest Regressor ===

rf\_model = RandomForestRegressor(n\_estimators=100, random\_state=42)

rf\_model.fit(X\_train, y\_train)

rf\_pred = rf\_model.predict(X\_test)

print("=== Random Forest Regressor Results ===")

print(f"MAE: {mean\_absolute\_error(y\_test, rf\_pred):.2f}")

print(f"MSE: {mean\_squared\_error(y\_test, rf\_pred):.2f}")

print(f"RMSE: {np.sqrt(mean\_squared\_error(y\_test, rf\_pred)):.2f}")

print(f"R²: {r2\_score(y\_test, rf\_pred):.2f}\n")

# === Feature Importance from Random Forest ===

feature\_importance = pd.Series(rf\_model.feature\_importances\_, index=X.columns)

plt.figure(figsize=(10, 6))

feature\_importance.sort\_values(ascending=False).head(15).plot(kind='bar')

plt.title("Top 15 Feature Importances")

plt.ylabel("Importance Score")

plt.show()

# === Predict Prices for 5 Sample Inputs ===

sample\_inputs = X\_test[:5]

sample\_preds = rf\_model.predict(sample\_inputs)

print("=== Predicted Prices for 5 Sample Inputs ===")

for idx, price in enumerate(sample\_preds, 1):

print(f"Input {idx}: ₹{price:,.2f}")