# *Source code*

```
import pandas as pd
 import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt from sklearn.model_selection
 import train_test_split from sklearn.linear_model
 import LinearRegression from sklearn.ensemble
 import RandomForestRegressor
from sklearn.metrics
import mean_absolute_error, mean_squared_error, r2_score
import xgboost as xgb
import gradio as gr

# 1. Data Loading & Preprocessing (Using California Housing Dataset)
from sklearn.datasets import fetch_california_housing
 # Import California housing dataset

# Load California housing dataset california = fetch_california_housing()
data = pd.DataFrame(california.data, columns=california.feature_names)
data['PRICE'] = california.target  # Target variable is 'PRICE'

# 2. EDA (Exploratory Data Analysis)
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt='.2f', linewidths=1)
plt.title('Correlation Matrix of Features')
plt.show()

# Pairplot to visualize relationships sns.pairplot(data, diag_kind='kde') plt.show()

# 3. Data Preprocessing
# Splitting the dataset into features (X) and target variable (y)
X = data.drop('PRICE', axis=1)
 y = data['PRICE']

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Modeling: Linear Regression, Random Forest Regressor, and XGBoost Regressor

# Linear Regression Model lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

# Random Forest Regressor Model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# XGBoost Regressor Model
xgb_model = xgb.XGBRegressor(n_estimators=100, learning_rate=0.05,
random_state=42) xgb_model.fit(X_train, y_train)

# 5. Model Evaluation (Mean Absolute Error, Mean Squared Error, and R^2 score) def
evaluate_model(model, X_test, y_test):
   y_pred = model.predict(X_test)
  mae = mean_absolute_error(y_test, y_pred)
```

```
    mse = mean_squared_error(y_test, y_pred)

   r2 = r2_score(y_test, y_pred)

  return mae, mse, r2


# Evaluating all models

 models = {  "Linear Regression": lr_model,

   "Random Forest": rf_model,

   "XGBoost": xgb_model

}


for name, model in models.items():

   mae, mse, r2 = evaluate_model(model, X_test, y_test)

  print(f"Model: {name}")

  print(f"Mean Absolute Error: {mae:.2f}")

  print(f"Mean Squared Error: {mse:.2f}")

 print(f"R^2 Score: {r2:.2f}\n")


# 6. Visualization of Model Performance

# Comparing predicted vs actual for XGBoost (best model)

y_pred = xgb_model.predict(X_test)


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

plt.scatter(y_test, y_pred)

plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red', linewidth=2)
plt.xlabel('Actual Prices')

 plt.ylabel('Predicted Prices')

plt.title('Actual vs Predicted Housing Prices (XGBoost)')

plt.show()
```

```
# 7. Gradio UI for Model Deployment


def predict_price(features):     # Convert features to a DataFrame

    features = np.array(features).reshape(1, -1)


    # Predict using XGBoost model (best performing model)

 predicted_price = xgb_model.predict(features)

  return predicted_price[0]

# Create Gradio Interface for prediction

# The number of features in California housing dataset is different from Boston

# We need to update the number of sliders based on the new dataset # Changed
gr.inputs.Slider to gr.Slider and gr.outputs.Textbox to gr.Textbox

 inputs = [gr.Slider(minimum=data[col].min(), maximum=data[col].max(),

value=data[col].mean(), label=col) for col in california.feature_names]

output = gr.Textbox(label="Predicted Housing Price")


gr.Interface(fn=predict_price, inputs=inputs, outputs=output, live=True).launch()
```