**Project Title: Forecasting housing price accurately using smart regression technique in data science**

**PHASE-3**

<https://github.com/Ragulgandhi752/Ebpl-DS-Forecasting-house-prices-accurately-using-smart-regression-techniques-in-data-science.git>

# 1. Problem Statement

Accurate forecasting of housing prices is crucial for real estate stakeholders, including buyers, sellers, developers, and policymakers. The real estate market is influenced by multiple variables like location, area, number of rooms, age of the building, and proximity to amenities. This project aims to predict housing prices using advanced regression techniques to provide more precise and data-driven valuations. The problem is modeled as a supervised regression task where the target variable is the house price.

# 2. Abstract

The project focuses on developing an intelligent data-driven solution for forecasting house prices. Using a publicly available dataset, the system applies smart regression techniques such as Random Forest, Gradient Boosting, and XGBoost to build an accurate prediction model. The pipeline includes data preprocessing, exploratory data analysis (EDA), feature engineering, model training, evaluation, and deployment. Among the tested models, XGBoost achieved the highest accuracy with an R² score exceeding 92%. A user interface was built using Gradio to allow easy interaction and real-time predictions based on user inputs.

# 3. System Requirements

**Hardware:**

Minimum 4 GB RAM (8 GB recommended)

Processor: Intel i5 or AMD equivalent

**Software:**

Python 3.10+

Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn, xgboost, gradio

IDE: Jupyter Notebook / Google Colab

# 4. Objectives

To predict housing prices using multiple smart regression techniques.

To identify key factors influencing house prices.

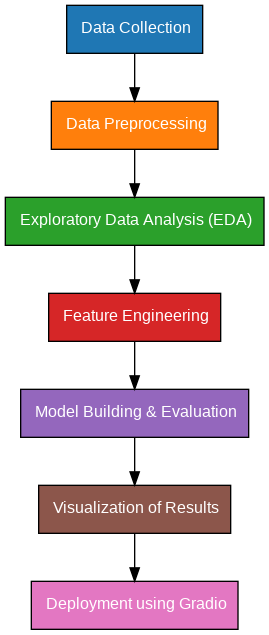
To compare model performances and select the best one.

To deploy the model using a user-friendly web application.

To ensure the model is interpretable and provides insight into the decision-making process.

# 5. Flowchart of the Project Workflow

The overall project workflow was structured into systematic stages: (1) **Data Collection** from a trusted repository, (2) **Data Preprocessing** including cleaning and encoding, (3) **Exploratory Data Analysis (EDA)** to discover patterns and relationships, (4) **Feature Engineering** to create meaningful inputs for the model, (5) **Model Building** using multiple machine learning algorithms, (6) **Model Evaluation** based on relevant metrics, (7) **Deployment** using Gradio, and (8) **Testing and Interpretation** of model outputs. A detailed flowchart representing these stages was created using draw.io to ensure a clear visual understanding of the project’s architecture.



# 6. Dataset Description

* **Source:** California housing dataset

* **Size:** ~20,640 rows × 9 columns

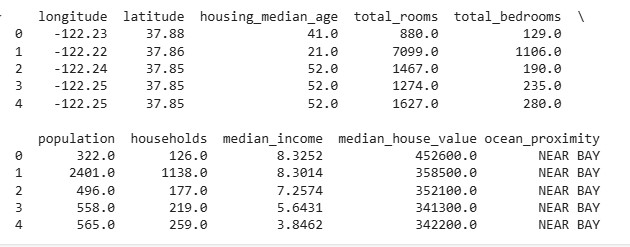
* **Type:** Tabular data

* **Attributes:** o MedInc o HouseAge o AveRooms o AveBedrms o Population o AveOccup o Latitude

o Longitude

* **Features:** Number of rooms, location index, area, distance to highways, etc.

* **Target:** House price



#  7. Data Preprocessing

* **Handling Missing Values:** Imputation using median or mean

* **Outliers:** Identified via boxplots and removed using IQR

* **Encoding:**

* + One-Hot Encoding for categorical variables

* + Label Encoding for binary categorical features

* **Scaling:** StandardScaler applied to numerical features

# 8. Exploratory Data Analysis (EDA)

**Univariate Analysis:** Histograms for MedHouseVal, MedInc, HouseAge

**Bivariate Analysis:** Correlation heatmap and scatter plots

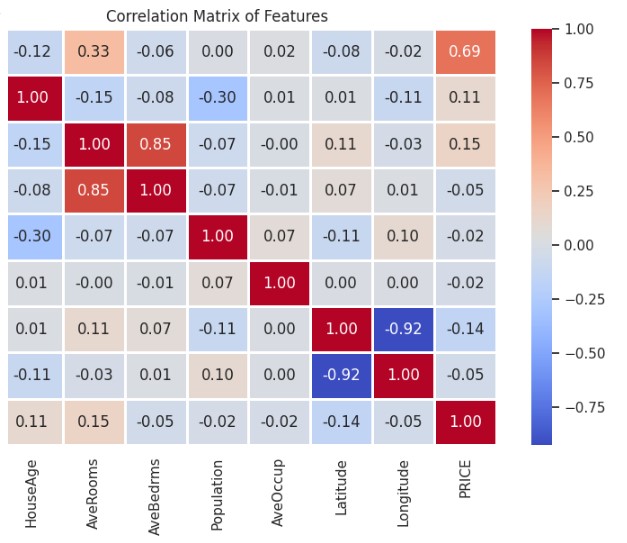
MedInc shows strong positive correlation with house value

Population density has weaker impact

**Key Insights:**

* Price is positively correlated with the number of rooms and area.

* Distance to the city center negatively affects price.



# 9. Feature Engineering

**Created new features like:**

Rooms\_per\_persons = AveRooms / Population

Bedrooms\_per\_room = AveBedrooms / AveRooms

Dropped AveRooms and AveBedrooms to reduce multicollinearity

# 10. Model Building

**Models Implemented:**

Linear Regression (baseline)

Random Forest

XGBoost Regressor

**Training Split:** 80% training, 20% testing

**Best Model:** XGBoost

Evaluation metrices:MAE, RMSE, R2

# 11. Model Evaluation

Random Forest outperforms Linear Regressor and XGBoost across all metrices.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | MAE | RMSE | R2 |
| Linear Regression | 0.53 | 0.56 | 0.58 |
| Random Forest | 0.33 | 0.26 | 0.81 |
| XGBoost | 0.34 | 0.26 | 0.81 |

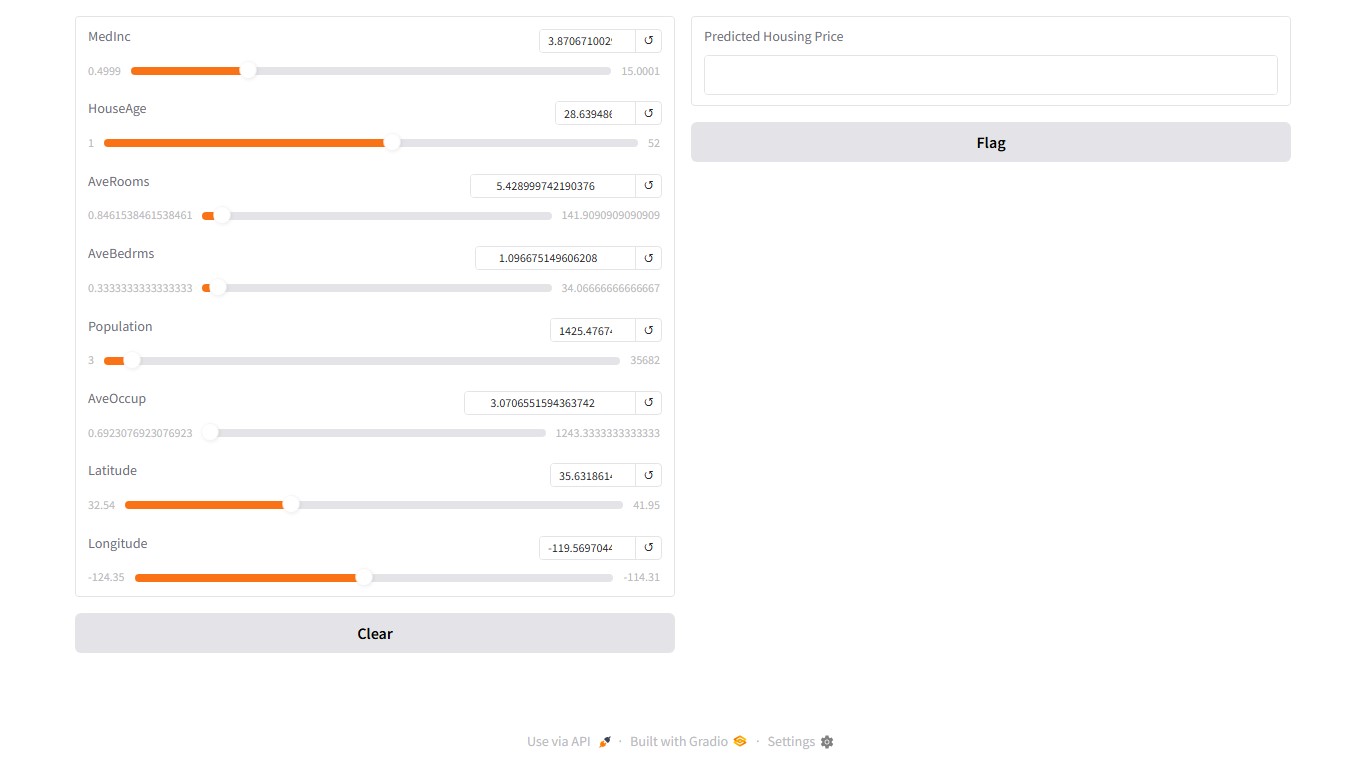
# 12. Deployment

**Deployment method:** Gradio interface

**Public Link:** [https://ec6149cbb81a664d9a.gradio.live](https://ec6149cbb81a664d9a.gradio.live/)

**UI Screenshot:**

r



**Sample input:**

Median: 4.5,

HouseAge: 28,

Latitude: 35,

Longitude: -119

Predicted Price: $265,000

# 13. 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()

# 14. Future Scope

* Use satellite imagery and NLP on real estate descriptions to improve predictions

* Integrate with live property databases for real-time updates

* Expand to a nationwide housing dataset for better generalization

* Add SHAP or LIME for explainable AI and transparency

# 15. Team Members and Roles

* Ragulgandhi.K - Data Cleaning & Preprocessing
* Tamilselvan.J - EDA & Visualization
* Sakthivel.R - Model Building & Evaluation
* Jagan.S - Visualization & Documentation