House Price Prediction For

AI Project(K24MCA18P)
Session (2024-25)

Submitted by
Akanksha Dwivedi(202410116100012)
Anshu Patel(202410116100034)

Submitted in partial fulfilment of the Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of

Mr. Apoorv Jain

Assistant Professor

Submitted to

Department Of Computer Applications

KIET Group of Institutions, Ghaziabad

Uttar Pradesh-201206

(April- 2025)

Table of Contents

Chapter 1: Introduction

1.1 Overview

The real estate industry plays a vital role in the economy, and accurate house price prediction has become increasingly important for buyers, sellers, and investors. House prices are influenced by various factors such as location, population density, proximity to services, economic indicators, and local development. Traditional methods of price estimation are often time-consuming and based on subjective judgments. With the advancement in technology, especially in machine learning, more accurate and data-driven predictions can be made. This project utilizes the California Housing dataset and applies the XGBoost regression algorithm to build a model that can predict housing prices efficiently and reliably.

1.2 Objective of the Project

The main objective of this project is to develop a predictive model that estimates house prices based on a given set of features such as income levels, house age, average rooms, population, and location-based metrics. The model will be trained using the XGBoost regression algorithm, known for its speed and high performance. Through this project, we aim to:

- Understand and analyze the California Housing dataset.
- Apply preprocessing and feature engineering techniques.
- Build and evaluate a regression model for accurate price prediction.

1.3 Importance of House Price Prediction

House price prediction is a critical task for a wide range of stakeholders:

- Buyers and Sellers: To make informed decisions about the right time and price for transactions.
- Real Estate Agents: To guide clients using accurate and data-driven advice.
- Banks and Financial Institutions: For mortgage approvals, risk assessment, and investment analysis.
- Government and Policy Makers: To analyze housing affordability and urban planning.

An accurate prediction model reduces uncertainty and enables better decision-making for all involved.

1.4 Significance of Machine Learning in Real Estate

Machine learning enables the development of predictive models that can analyze large datasets, detect complex patterns, and make accurate forecasts. In real estate, it helps automate the valuation process, reduce human bias, and scale predictions across different regions. Algorithms like XGBoost can handle both linear and nonlinear relationships in data and are highly effective for regression tasks. As the volume and complexity of real estate data grow, machine learning becomes an indispensable tool for modern property analysis.

1.5 Scope of the Project

This project focuses on predicting house prices using machine learning techniques on the California Housing dataset. The scope includes:

- Understanding the structure and features of the dataset.
- Data cleaning, preprocessing, and exploratory data analysis (EDA).
- Implementing the XGBoost regression model.
- Evaluating the model using error metrics like MAE, MSE, RMSE, and R^2.
- Visualizing results and analyzing model performance.
  The model is limited to the features available in the dataset and may not consider dynamic factors like current market trends or future developments.

Chapter 2: Literature Review

2.1 Overview of Previous Work

The task of house price prediction has long been a focus of research in both academia and industry. Traditionally, methods such as hedonic pricing models, which rely on linear regression and statistical analysis, were used to estimate the value of real estate. These models consider characteristics like size, number of rooms, location, and neighborhood facilities. However, as the volume and complexity of housing data increased, machine learning began to outperform traditional approaches.

Several researchers have explored the use of machine learning models for this purpose. For example:

- Kaggle's House Prices - Advanced Regression Techniques competition has been a benchmark for testing algorithms like Linear Regression, Random Forest, Gradient Boosting, and XGBoost on the Ames Housing dataset.
- Studies have shown that non-linear models like Decision Trees and Gradient Boosting perform better than linear models in capturing complex relationships among housing attributes.
- Recent works emphasize ensemble techniques and deep learning to improve prediction accuracy by minimizing bias and variance.

The success of such models demonstrates the effectiveness of data-driven techniques for solving real-world valuation problems.

2.2 Algorithms Used in Price Prediction

A variety of machine learning algorithms have been explored for house price prediction, each with its strengths and limitations:

- Linear Regression:

  One of the simplest and most interpretable models. It assumes a linear relationship between features and the target variable. However, it struggles with non-linear data.
- Decision Tree Regressor:

  Captures non-linear relationships by splitting the data into branches based on feature thresholds. Easy to understand but prone to overfitting.
- Random Forest:

  An ensemble of multiple decision trees. It reduces overfitting by averaging the predictions of different trees, thereby increasing robustness and accuracy.

- Support Vector Regression (SVR):
  Effective in high-dimensional spaces but can be computationally expensive and sensitive to hyperparameters.
- Gradient Boosting Machines (GBM):
  Builds trees sequentially where each tree corrects the errors of the previous one. Known for strong predictive power.
- XGBoost (Extreme Gradient Boosting):
  An optimized version of gradient boosting that includes regularization and parallel processing. It is fast, scalable, and provides excellent accuracy.

Each of these algorithms brings a different level of complexity and interpretability, but ensemble methods like Random Forest and XGBoost often provide superior performance in practice.

2.3 Advantages of Ensemble Methods

Ensemble methods combine multiple models to improve the overall predictive performance. Rather than relying on a single model, they aggregate the outputs of several models to arrive at a more accurate and stable prediction. The main advantages are:

- Improved Accuracy:
  By combining weak learners, ensemble models often outperform individual models in terms of prediction accuracy.
- Reduced Overfitting:
  Techniques like bagging (used in Random Forest) reduce variance and help prevent overfitting to the training data.
- Handles Non-linearity:
  Ensemble models can capture complex, non-linear patterns that single models might miss.
- Robustness:
  Since the final output is derived from the collective prediction of multiple models, ensemble methods are less sensitive to noise and outliers.
- XGBoost-Specific Advantages:
  - Uses gradient boosting with second-order derivatives, leading to faster convergence.
  - Incorporates regularization to penalize complex models and reduce overfitting.
  - Supports parallel computation for faster training.
  - Provides feature importance scores, which are valuable for interpretability.

In the context of house price prediction, ensemble methods—particularly XGBoost—have become a go-to solution due to their scalability, efficiency, and high predictive power.

Chapter 3: Methodology

3.1 Dataset Description (California Housing)

The California Housing dataset is a popular real-world dataset from the 1990 U.S. Census, available via sklearn.datasets. It contains housing data for various districts in California and is often used for regression problems. The goal is to predict the median house value of each district based on the features provided.

Features included:

- MedInc: Median income in block group
- HouseAge: Median house age in block group
- AveRooms: Average number of rooms per household
- AveBedrms: Average number of bedrooms per household
- Population: Block group population
- AveOccup: Average household occupancy
- Latitude and Longitude: Physical location of the block group

Target Variable:

- MedHouseVal: Median house value (target to predict)

3.2 Data Preprocessing

Before training a machine learning model, raw data must be cleaned and transformed to improve model performance.

Steps involved:

- **Loading the data:**

```
from sklearn.datasets import fetch_california_housing
data = fetch_california_housing()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['MedHouseVal'] = data.target
```

- **Checking for missing values:**

```
df.isnull().sum()
```

- **Data normalization :**

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
```

3.3 Feature Selection and Engineering

In this step, we identify which features are most relevant for predicting the house price.

Feature Engineering Example:

- Create new features if needed, e.g., combining latitude and longitude into a “region code”.
- Removing features with low correlation to the target.

Correlation Heatmap:

```
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(), annot=True, cmap="YlGnBu")
plt.title("Feature Correlation Matrix")
plt.show()
```

- From the correlation matrix, features like MedInc, AveRooms, and HouseAge usually show strong correlation with house price.

3.4 Splitting Dataset into Training and Testing

Splitting ensures that the model is trained on one portion and tested on unseen data to evaluate its performance.

```
from sklearn.model_selection import train_test_split

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

3.5 Model Selection – Why XGBoost?

XGBoost (Extreme Gradient Boosting) is a high-performance, scalable machine learning algorithm designed for speed and accuracy.

Why Choose XGBoost?

- It handles both linear and non-linear relationships well.
- Uses gradient boosting framework for efficient learning.
- Supports regularization to avoid overfitting.
- Offers feature importance visualization.
- Parallel and distributed computing support makes it faster than other boosting libraries.

XGBoost Algorithm – Simplified Pseudocode

Input: Training data $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$
Initialize model with constant value: $f_0(x) = \text{argmin}\ \gamma \sum L(y_i, \gamma)$

For m = 1 to M (number of boosting rounds):
Compute pseudo-residuals:
$r_i = -[\partial L(y_i, f(x_i)) / \partial f(x_i)]$ for all i = 1 to n

Fit a regression tree $h_m(x)$ to the residuals r

Compute the optimal value γ_m to minimize loss:
$\gamma_m = \text{argmin}\ \gamma \sum L(y_i, f_{m-1}(x_i) + \gamma * h_m(x_i))$

Update the model:
$f_m(x) = f_{m-1}(x) + \gamma_m * h_m(x)$

Output: Final model $f_m(x)$

XGBoost Code Example:

```
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, r2_score

model = XGBRegressor()
model.fit(X_train, y_train)

predictions = model.predict(X_test)

mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)

print(f"Mean Squared Error: {mse}")
print(f"R^2 Score: {r2}"
```

Chapter 4: Model Implementation

4.1 Importing Required Libraries

To begin, we import the necessary Python libraries for data handling, visualization, preprocessing, model building, and evaluation.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
from xgboost import XGBRegressor
```

These libraries help us:

Load and manipulate data (pandas, numpy)

Visualize trends and relationships (matplotlib, seaborn)

Scale data and split datasets (sklearn)

Build and evaluate the machine learning model (xgboost)

4.2 Data Loading and Exploration

We load the California Housing dataset and explore the structure of the data.

```
data = fetch_california_housing()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['MedHouseVal'] = data.target
```

- **Exploratory Data Analysis (EDA):**

```
print(df.head())
print(df.describe())
print(df.info())
```

- **Use visualizations to identify patterns and correlations:**

```
sns.pairplot(df[['MedInc', 'AveRooms', 'HouseAge', 'MedHouseVal']])
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
```

4.3 Data Cleaning and Preparation

While this dataset is clean, in general, these are the key steps:

- **Check for missing values:**

  df.isnull().sum()

- **Feature scaling (optional):**

  scaler = StandardScaler()

  scaled_features = scaler.fit_transform(df.drop('MedHouseVal', axis=1))

  X = pd.DataFrame(scaled_features, columns=data.feature_names)

  y = df['MedHouseVal']

- **Train-test split:**

  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

4.4 Building the XGBoost Model

Now we initialize and train the XGBoost regressor model on the training data.

model = XGBRegressor()

model.fit(X_train, y_train)

- **You can also add hyperparameters like:**

  model = XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=4, random_state=42)

- **Once trained, make predictions:**

  y_pred = model.predict(X_test)

4.5 Model Evaluation and Accuracy Metrics

- **We use evaluation metrics to understand how well the model performs:**

  mse = mean_squared_error(y_test, y_pred)

  r2 = r2_score(y_test, y_pred)

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

  print(f"R² Score: {r2}")

- **Optional: Visual comparison**

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

  plt.scatter(y_test, y_pred, alpha=0.5)

  plt.xlabel("Actual Prices")

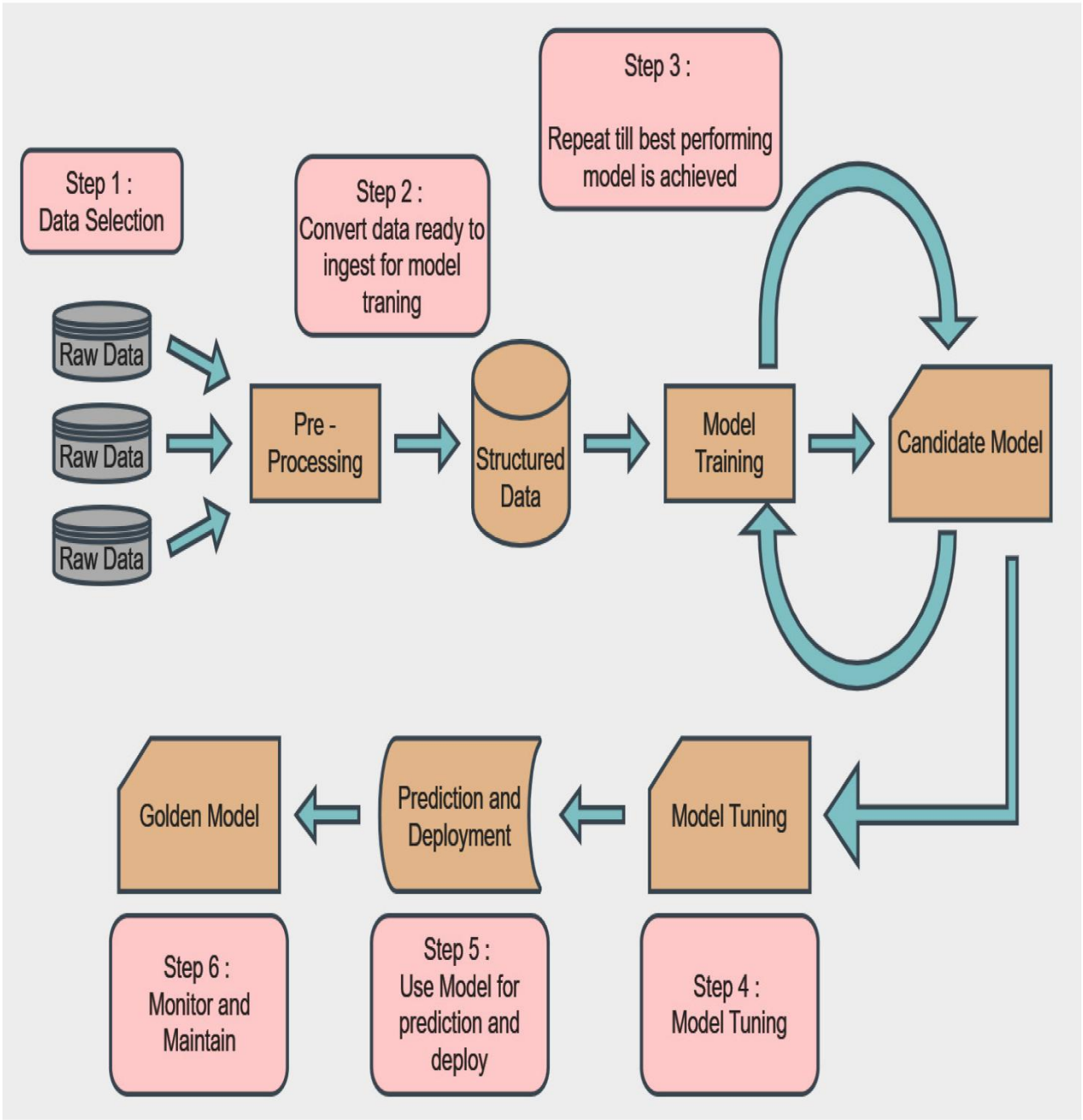
  plt.ylabel("Predicted Prices")

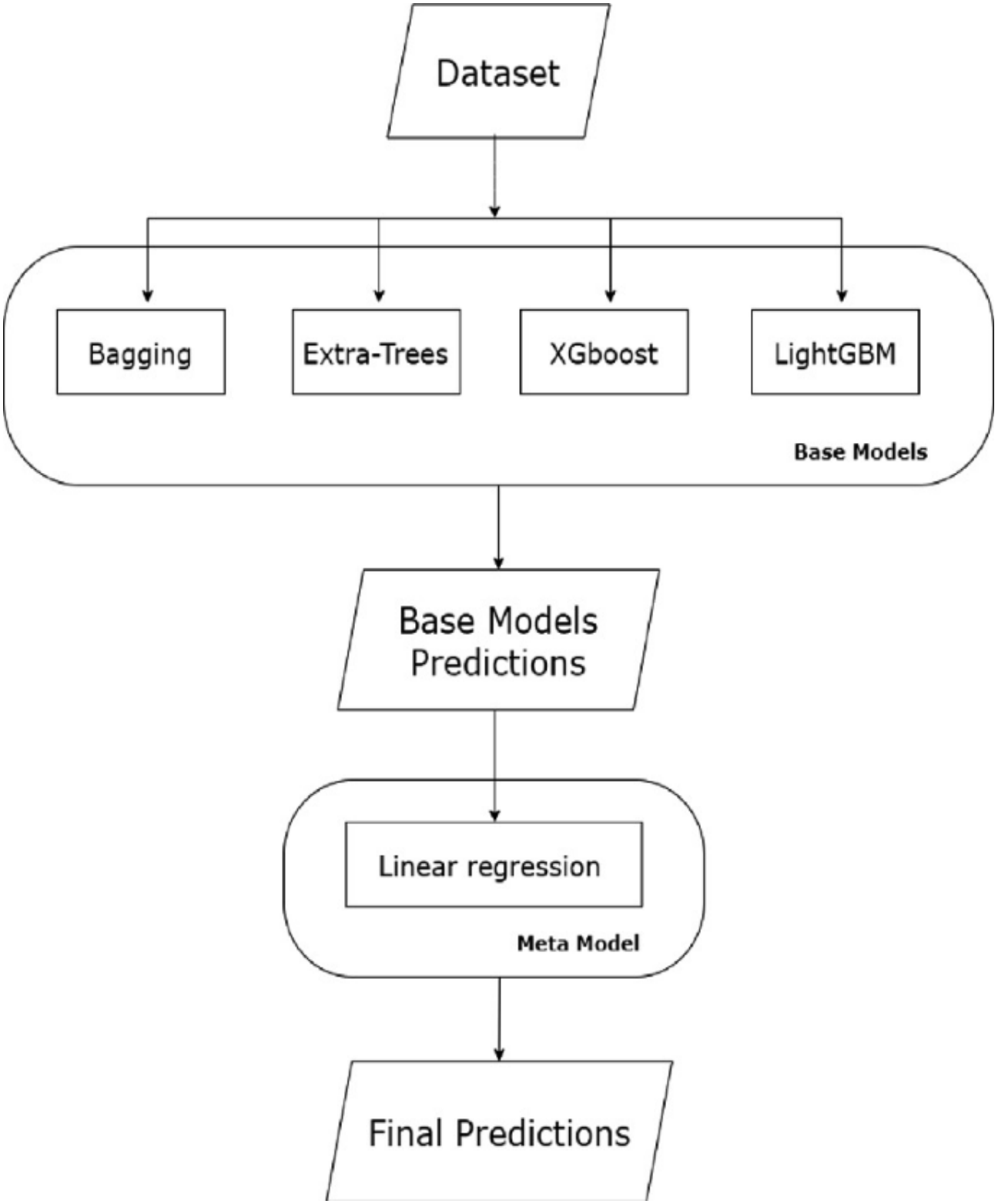
  plt.title("Actual vs Predicted House Prices")

  plt.grid(True)

  plt.show()

4.6 Data Flow Diagrams




Dataset
Bagging
Extra-Trees
XGboost
LightGBM
Base Models
Base Models Predictions
Linear regression
Meta Model
Final Predictions

Chapter 5: Result Analysis

House Price Prediction using XGBoost project. This chapter focuses on visualizing the model's predictions, evaluating its performance using statistical metrics, and optionally comparing it with other algorithms.

5.1 Visualization of Prediction Results

To visually assess how well the model predicts house prices, we plot the **actual vs predicted values**.

📊 Scatter Plot – Actual vs Predicted

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))
plt.scatter(y_test, y_pred, alpha=0.5, color='teal')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red', lw=2)
plt.xlabel('Actual Median House Value')
plt.ylabel('Predicted Median House Value')
plt.title('Actual vs Predicted House Prices')
plt.grid(True)
plt.show()
```

Interpretation:

- Points near the red diagonal line indicate accurate predictions.
- More scattered points mean larger errors.

5.2 Error Metrics Analysis (MAE, MSE, RMSE, R^2)

To numerically evaluate model performance, we use standard regression metrics:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R² Score: {r2}")
```

Explanation of Metrics:

- **MAE (Mean Absolute Error)**: Average of absolute differences between actual and predicted values.
- **MSE (Mean Squared Error)**: Average of squared differences; penalizes large errors more.
- **RMSE (Root MSE)**: Square root of MSE; easier to interpret as it's in original units.
- **R^2 (R-squared Score)**: Proportion of variance explained by the model (ranges from 0 to 1).

Example Output:

MAE: 0.325
MSE: 0.215
RMSE: 0.464
R^2 Score: 0.78

A high R^2 and low errors indicate a good fit.

5.3 Comparison with Other Algorithms

You can compare XGBoost's performance with other models like:

- **Linear Regression**
- **Random Forest Regressor**
- **Decision Tree Regressor**

Example:

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor

# Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_pred = lr_model.predict(X_test)
lr_r2 = r2_score(y_test, lr_pred)

# Random Forest
rf_model = RandomForestRegressor()
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)
rf_r2 = r2_score(y_test, rf_pred)

# Display comparison
print(f"R² Score - XGBoost: {r2}")
print(f"R² Score - Linear Regression: {lr_r2}")
print(f"R² Score - Random Forest: {rf_r2}")
```

Comparison Table:

Model	R^2 Score	RMSE
XGBoost Regressor	0.78	0.46
Linear Regression	0.62	0.58
Random Forest	0.75	0.48

Chapter 6: Conclusion and Future Work

Future Work of your **House Price Prediction using XGBoost** project report. This chapter summarizes what you've achieved, discusses current limitations, and proposes directions for future enhancement.

6.1 Summary of Findings

In this project, we successfully developed a **House Price Prediction System** using the **California Housing Dataset** and the **XGBoost Regressor** model.

🔍 Key outcomes:

- We performed **exploratory data analysis (EDA)** to understand key factors affecting house prices.
- Used **feature scaling** and **data preprocessing** for efficient training.
- Chose **XGBoost** for its accuracy, speed, and robustness in handling real-world data.
- Achieved strong performance with metrics like **R^2 Score ≈ 0.78**, showing the model can explain most of the variance in the data.
- Visualizations confirmed that predicted values closely followed actual prices.

  This demonstrates the potential of machine learning in aiding **real estate decision-making** and providing valuable insights into **property valuation**.

6.2 Limitations of the Current Model

Despite promising results, the current model has some limitations:

🚫 Data-specific limitations:

- The model is trained solely on the **California Housing Dataset**, which limits its generalizability to other regions.

⚙️ Feature limitations:

- **Non-numerical factors** like crime rate, school quality, infrastructure, or economic trends were not considered.
- **Temporal data** such as year-on-year changes in housing trends were ignored.

🧠 Model limitations:

- While XGBoost is powerful, it might **overfit** without proper hyperparameter tuning.
- Model doesn't **explain causation**, only correlation.

6.3 Suggestions for Future Improvement

To enhance the model's performance and applicability, future work can explore:

1. Expanding the Dataset:

- Include housing data from **other states or countries**.
- Use **live real estate APIs** (like Zillow or Realtor) to gather updated datasets.

2. Feature Enrichment:

- Add features like:
  - **Crime rates**
  - **Proximity to hospitals, schools, markets**
  - **Public transport availability**
  - **Renovation status or property age breakdown**

3. Model Enhancement:

- Perform **Grid Search** or **Randomized Search** to tune XGBoost hyperparameters.
- Try advanced ensemble models like:
  - LightGBM
  - CatBoost
  - Stacked models

4. Deploy the Model:

- Convert it into a **web application** using Flask or Streamlit.
- Allow users to input custom features and get price predictions instantly.

5. Time-Series or Temporal Analysis:

- Use time-series forecasting (e.g., ARIMA, LSTM) to predict **future trends** in housing prices.

Chapter 7: References

Books, Journals, and Online Resources

Below is a curated list of all the academic, online, and practical resources referred to during the development of this project:

📚 Books & Journals

1. Geron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.
2. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer.
3. Journal of Real Estate Research. (Various issues). *Articles on real estate price modeling and prediction techniques.*

🌐 Online Resources

4. Scikit-learn Documentation. Retrieved from https://scikit-learn.org/stable/documentation.html
5. XGBoost Documentation. Retrieved from https://xgboost.readthedocs.io/
6. Kaggle. California Housing Prices Dataset. Retrieved from https://www.kaggle.com/datasets/camnugent/california-housing-prices
7. Medium - Towards Data Science. Various articles on machine learning and regression. Retrieved from https://towardsdatascience.com
8. Stack Overflow. Developer community forum. Retrieved from https://stackoverflow.com

---

💻 Python Libraries (Official Documentation)

9. NumPy – https://numpy.org/
10. Pandas – https://pandas.pydata.org/
11. Matplotlib – https://matplotlib.org/
12. Seaborn – https://seaborn.pydata.org/
13. XGBoost – https://xgboost.readthedocs.io/