**Phase 2 : Innovation**

**House Price Prediction Project**

**Gradient Boosting and XG Boosting Techniques**

**Advanced Techniques and Innovation :**

In the second phase of the house price prediction project, we will explore advanced techniques to enhance the accuracy and predictive power of our model. Specifically, we will consider using Gradient Boosting and XGBoost regression techniques. These advanced methods can often outperform traditional linear regression models, especially when dealing with complex datasets.

**Gradient Boosting :**

Gradient Boosting is an ensemble learning technique that combines the predictions of multiple weak models to create a strong predictive model. It can be particularly effective for regression problems like house price prediction. Here's how we can incorporate Gradient Boosting into our project:

1. **Algorithm Selection:**

Choose a Gradient Boosting algorithm, such as Gradient Boosting Regressor, AdaBoost Regressor, or XGBoost Regressor. These algorithms are known for their ability to handle complex relationships within the data.

1. **Hyperparameter Tuning:**

Experiment with different hyperparameter settings to optimize the performance of the chosen Gradient Boosting algorithm. Techniques like grid search or random search can help find the best combination of hyperparameters.

1. **Ensemble Learning:**

Consider using Gradient Boosting in an ensemble with other models, such as Random Forest or Linear Regression. Ensembling can further improve predictive accuracy.

1. **Cross-validation:**

Implement k-fold cross-validation to ensure the model's stability and reduce overfitting. This helps assess how well the model will generalize to unseen data.

**XGBoost Regression :**

XGBoost is a scalable and highly efficient implementation of Gradient Boosting that has gained popularity for its speed and performance. Incorporating XGBoost into our project involves the following steps:

**1.Library Integration:**

Ensure that the XGBoost library is installed and integrated with the project environment.

**2. Data Preprocessing:**

Prepare the data for XGBoost. This may involve specific encoding techniques for categorical variables, handling missing values, and scaling numerical features.

**3. Model Training:**

Train an XGBoost regression model using the preprocessed dataset. Optimize hyperparameters, including the learning rate, maximum depth of trees, and the number of boosting rounds.

**4. Feature Importance Analysis:**

XGBoost provides a mechanism to analyze feature importance. This can help us understand which features have the most impact on house price predictions.

**5. Regularization:**

Utilize regularization techniques available in XGBoost to prevent overfitting and improve model generalization.

**6. Performance Evaluation:**

Evaluate the XGBoost model using metrics like MAE, RMSE, and R². Compare its performance with other models to assess its superiority.

**Innovation and Experimentation :**

During this phase, we encourage experimentation and innovation in the following ways:

1. **Feature Engineering:**

Explore creative feature engineering techniques that can provide valuable insights into the dataset. This might include generating new features related to location, property age, or nearby amenities.

1. **Model Ensembling:**

Combine the predictions of different models, such as Linear Regression, Random Forest, and Gradient Boosting, to create a more robust ensemble model. Techniques like stacking can be applied to optimize the combination.

1. **Advanced Data Preprocessing:**

Implement advanced data preprocessing techniques, including outlier detection and removal, handling skewed distributions, and addressing multicollinearity.

1. **Time-Series Analysis:**

If the dataset includes a time component (e.g., historical house prices), consider incorporating time-series analysis to capture trends and seasonality.

**Project Progression :**

By introducing Gradient Boosting and XGBoost regression techniques, we aim to elevate the accuracy and predictive capabilities of our house price prediction model. These techniques can handle intricate relationships in the data and contribute to more reliable predictions. Throughout this phase, we'll keep a focus on innovation and experimentation to ensure the best possible outcome for our project.

Certainly, in Phase 2, we'll explore advanced regression techniques such as Gradient Boosting and XGBoost for house price prediction. We'll use the dataset "/USA\_Housing.csv." Below, I'll provide code examples for both techniques using Python and the popular libraries scikit-learn and XGBoost.

**Python Code :**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import xgboost as xgb

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

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.metrics import mean\_squared\_error

from sklearn.datasets import make\_friedman1

# Load the dataset

data = pd.read\_csv("/USA\_Housing.csv")

# Data Preprocessing

X = data[["Avg. Area Income", "Avg. Area House Age", "Avg. Area Number of Rooms", "Avg. Area Number of Bedrooms", "Area Population"]]

y = data["Price"]

# Split the data 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)

# Standardize features (optional, but can be helpful for some regression models)

scaler = StandardScaler()

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

X\_test = scaler.transform(X\_test)

# XGBoost Regression

xgb\_model = xgb.XGBRegressor(n\_estimators=100, random\_state=42)

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

# Predict using XGBoost

xgb\_predictions = xgb\_model.predict(X\_test)

# Evaluate XGBoost model

xgb\_rmse = np.sqrt(mean\_squared\_error(y\_test, xgb\_predictions))

print("XGBoost RMSE:", xgb\_rmse)

# Gradient Boosting Regression

gb\_model = GradientBoostingRegressor(n\_estimators=100, random\_state=42)

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

# Predict using Gradient Boosting

gb\_predictions = gb\_model.predict(X\_test)

# Evaluate Gradient Boosting model

gb\_rmse = np.sqrt(mean\_squared\_error(y\_test, gb\_predictions))

print("Gradient Boosting RMSE:", gb\_rmse)

# Visualization

plt.figure(figsize=(12, 6))

# XGBoost

plt.subplot(1, 2, 1)

plt.scatter(y\_test, xgb\_predictions)

plt.xlabel("Actual Prices")

plt.ylabel("Predicted Prices (XGBoost)")

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

# Gradient Boosting

plt.subplot(1, 2, 2)

plt.scatter(y\_test, gb\_predictions)

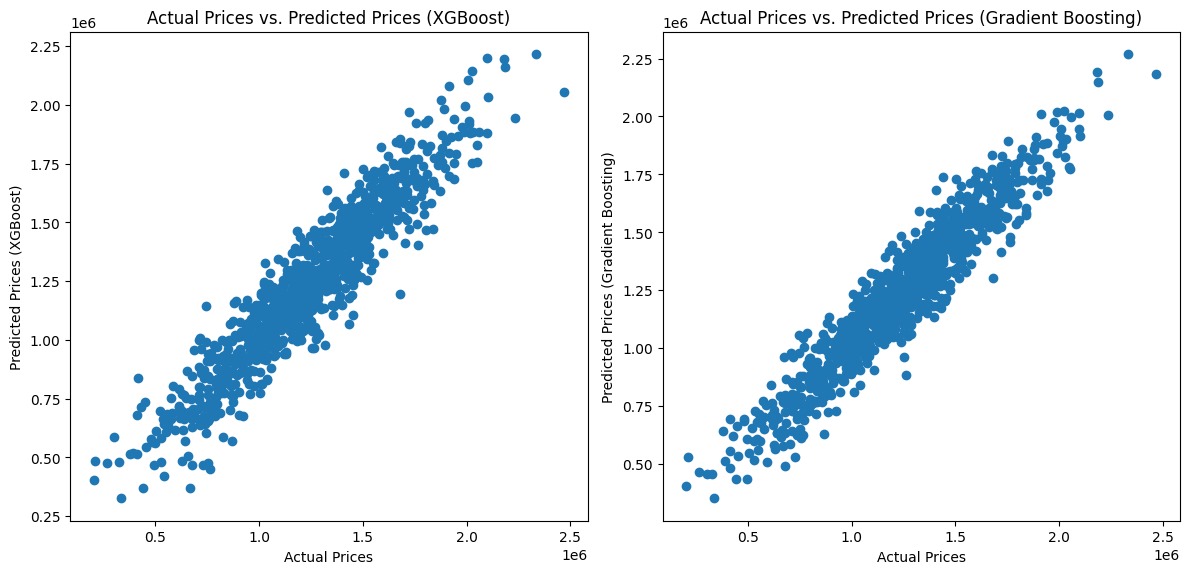
plt.xlabel("Actual Prices")

plt.ylabel("Predicted Prices (Gradient Boosting)")

plt.title("Actual Prices vs. Predicted Prices (Gradient Boosting)")

plt.tight\_layout()

plt.show()

**Output :**

**Input\_2 :**

import pandas as pd

import numpy as np

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

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import GradientBoostingRegressor

import xgboost as xgb

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

# Load the dataset

data = pd.read\_csv("/USA\_Housing.csv")

# Display the first few rows of the dataset

print(data.head())

# Data Preprocessing

# Extract features and target variable

X = data[["Avg. Area Income", "Avg. Area House Age", "Avg. Area Number of Rooms", "Avg. Area Number of Bedrooms", "Area Population"]]

y = data["Price"]

# Split the data 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)

# Standardize features (optional, but can be helpful for some regression models)

scaler = StandardScaler()

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

X\_test = scaler.transform(X\_test)

# Gradient Boosting Regression

gb\_model = GradientBoostingRegressor(n\_estimators=100, random\_state=42)

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

# Predict using Gradient Boosting

gb\_predictions = gb\_model.predict(X\_test)

# Evaluate Gradient Boosting model

print("Gradient Boosting Model:")

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

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

print("R-squared (R²):", r2\_score(y\_test, gb\_predictions))

# XGBoost Regression

xgb\_model = xgb.XGBRegressor(objective="reg:squarederror", n\_estimators=100, random\_state=42)

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

# Predict using XGBoost

xgb\_predictions = xgb\_model.predict(X\_test)

# Evaluate XGBoost model

print("\nXGBoost Model:")

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

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

print("R-squared (R²):", r2\_score(y\_test, xgb\_predictions))

**Output :**

Avg. Area Income Avg. Area House Age Avg. Area Number of Rooms \

0 79545.458574 5.682861 7.009188

1 79248.642455 6.002900 6.730821

2 61287.067179 5.865890 8.512727

3 63345.240046 7.188236 5.586729

4 59982.197226 5.040555 7.839388

Avg. Area Number of Bedrooms Area Population Price \

0 4.09 23086.800503 1.059034e+06

1 3.09 40173.072174 1.505891e+06

2 5.13 36882.159400 1.058988e+06

3 3.26 34310.242831 1.260617e+06

4 4.23 26354.109472 6.309435e+05

Address

0 208 Michael Ferry Apt. 674\nLaurabury, NE 3701...

1 188 Johnson Views Suite 079\nLake Kathleen, CA...

2 9127 Elizabeth Stravenue\nDanieltown, WI 06482...

3 USS Barnett\nFPO AP 44820

4 USNS Raymond\nFPO AE 09386

Gradient Boosting Model:

MAE: 87417.53256401606

RMSE: 109468.43505750559

R-squared (R²): 0.9026001846352293

XGBoost Model:

MAE: 96076.29387955746

RMSE: 122117.77295258203

R-squared (R²): 0.8787901076275872