



KINGSTON  
ENGINEERING COLLEGE

COLLEGE NAME : Kingston engineering college  
COLLEGE CODE : 5113  
DOMAIN : Artificial intelligence  
PROJECT TITLE : Predicting house prices using machine learning

PROJECT MEMBERS :	NAN MUDHALVAN ID :
A.Tejaswini (leader)	511321104101
A.Vainavi	511321104106
A.Vaishnavi	511321104107
S.Dimple	511321104086
J.Ruchitha	511321104080

## INTRODUCTION

The housing market is a crucial sector with profound impacts on individuals' financial stability and quality of life. Accurate prediction of house prices is vital for both buyers and sellers to make well-informed decisions. Housing prediction in machine learning is a dynamic and pivotal application that leverages data-driven techniques to forecast property prices, offering invaluable insights for both homebuyers and sellers, as well as real estate professionals. By harnessing advanced regression and predictive modeling algorithms, machine learning models can analyze various factors, such as location, size, amenities, and market trends, to make accurate predictions about property values. Using advanced regression techniques like Gradient Boosting or XGBoost can indeed lead to improved prediction accuracy in housing price prediction tasks in machine learning. These techniques are powerful ensemble methods that combine the predictions of multiple weak models (typically decision trees) to create a strong predictive model. Here's how you can approach this:

This predictive power not only aids in making informed decisions but also plays a significant role in optimizing real estate investments and providing a deeper understanding of housing market dynamics. In this context, this article explores the utilization of machine learning techniques for housing prediction, shedding light on the tools and methodologies that enhance the accuracy of these forecasts.

This project employs machine learning techniques to predict house prices using a variety of features, enhancing the understanding of the housing market. It involves using historical data, often a dataset containing information about various factors such as location, size, number of bedrooms, and other features of houses, to build a model that can predict the prices of new, unseen houses.

## Problem Definition and Design Thinking

**Problem Definition:** data preprocessing, feature engineering, model selection, training, and evaluation. The problem is to predict house prices using machine learning techniques. The objective is to develop a model that accurately predicts the prices of houses based on a set of features such as location, square footage, number of bedrooms and bathrooms, and other relevant factors. This project involves

### Design Thinking:

**Data Source:** Choose a dataset containing information about houses, including features like location, square footage, bedrooms, bathrooms, and price.

**Data Preprocessing:** Clean and preprocess the data, handle missing values, and convert categorical features into numerical representations.

**Feature Selection:** Select the most relevant features for predicting house prices.

**Model Selection:** Choose a suitable regression algorithm (e.g., Linear Regression, Random Forest Regressor) for predicting house

### Data Preprocessing:

- **Data Collection:** Gather a dataset that includes relevant features such as location, square footage, number of bedrooms and bathrooms, and any other factors that may influence house prices. This data can be collected from various sources, including real estate websites, government databases, or APIs.
- **Data Cleaning:** Clean the dataset by handling missing values, outliers, and any inconsistencies in the data. This may involve imputing missing values or removing outliers that could negatively impact the model's performance.
- **Feature Scaling:** Normalize or standardize numerical features to ensure they are on a similar scale. This prevents certain features from dominating the model's predictions.

- **Encoding Categorical Variables:** Convert categorical variables (e.g., location) into numerical format using techniques such as one-hot encoding or label encoding.
- **Splitting the Data:** Divide the dataset into training, validation, and test sets to evaluate the model's performance effectively.

### **Feature Engineering:**

- Create new features or transform existing ones to capture meaningful information that may not be evident in the raw data. For example, you can calculate the price per square foot, age of the property, or proximity to key amenities.

### **Model Selection:**

- Choose an appropriate machine learning algorithm for regression tasks. Common choices include linear regression, decision trees, random forests, support vector machines, or more advanced techniques like gradient boosting or neural networks.
- Consider using ensemble methods to combine multiple models for improved performance and robustness.

### **Model Training:**

- Train the selected model(s) on the training dataset using appropriate hyperparameters. Cross-validation can help tune hyperparameters and prevent overfitting.

### **Model Evaluation:**

- Assess the model's performance on the validation dataset using relevant evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or R-squared ( $R^2$ ) to measure prediction accuracy.
- Use visualization techniques like scatter plots or residual plots to understand how well the model captures the underlying patterns in the data.

# INNOVATION

## **Splitting the Data:**

Split your dataset into training and testing sets to evaluate the model's performance accurately. A common split ratio is 80/20 or 70/30.

## **Choosing the Algorithm:**

You mentioned two popular algorithms: Gradient Boosting and XGBoost. These algorithms are robust and often perform very well in regression tasks. You can choose one of them or even try both to see which one works better for your specific dataset.

## **Hyperparameter Tuning:**

Tune the hyperparameters of the chosen algorithm(s) to optimize their performance. This can involve using techniques like grid search or randomized search.

## **Training the Model:**

Train your chosen model(s) on the training data using the optimized hyperparameters.

## **Model Evaluation:**

Evaluate your model's performance using appropriate regression metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R<sup>2</sup>) on the testing data.

## **Feature Importance:**

For interpretability and insights, analyze feature importance scores provided by Gradient Boosting or XGBoost. This can help you understand which features have the most impact on predictions

## **DEVELOPMENT PART 1:**

### **Data Loading:**

•First, you need to obtain the housing dataset. This dataset should contain information about various features of houses (e.g., square footage.etc) and their corresponding sale prices.

•Depending on the format of your dataset (e.g., CSV, Excel.etc) we can use libraries like Pandas in Python to load the data into a data structure that can be easily manipulated.

### **Data Exploration:**

•Once the dataset is loaded, it's essential to explore and understand its structure. Check for features, their data types, and identify the target variable.

•You can use Pandas for basic data exploration commands like `info()`, `describe()`, and `head()`.

### **Feature Selection and Engineering:**

- Create new features or transform existing ones if they can provide additional insights.
- Choose the relevant features that are likely to have a significant impact on house prices.

### **Model Building:**

- After data preprocessing, you can proceed to select a machine learning model for predicting house prices. Common choices include linear regression, decision trees, random forests, or more advanced models like neural networks.

### **Model Training:**

Train your chosen model on the training data. The model learns the relationships between the features and the target variable (house prices).

### **Model Evaluation:**

Evaluate the model's performance using appropriate evaluation metrics, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or R-squared.

### **Prediction:**

Finally, you can use the trained model to make predictions on new or unseen data. This allows you to estimate house prices based on the model's learned patterns..

## **DEVELOPMENT PART 2:**

### **Feature Selection:**

Feature selection is a crucial step in building a predictive model, as it helps identify the most relevant features while reducing noise and complexity. Here are some common techniques for feature selection

### **Correlation Analysis:**

Identify which features are highly correlated with the target variable (house prices) and keep those with strong correlations.

### **Feature Importance:**

Utilize algorithms like Random Forest or Gradient Boosting to assess the importance of each feature in predicting house prices. Features with higher importance can be retained.

### **Recursive Feature Elimination (RFE):**



Train models with different subsets of features and iteratively eliminate the least important ones until you achieve the desired number of features.

### **Domain Knowledge:**

Consider domain expertise to determine which features are likely to have a significant impact on house prices.

### **Univariate Feature Selection:**

features based on Use statistical tests like chi-squared or ANOVA to select features based on their relationship with the target variable.

### **Model Training:**

With feature selection completed, you can proceed to train your predictive model. Here are the steps:

#### **Data Split:**

Divide your dataset into a training set and a validation set. This allows you to train your model on one portion of the data and evaluate its performance on another to avoid overfitting.

#### **Choose Model:**

Select an appropriate machine learning algorithm for regression. Common choices include Linear Regression, Decision Trees, Random Forest, Gradient Boosting, and Neural Networks.

#### **Feature Engineering:**

Prepare your data by encoding categorical variables, scaling numerical features, and handling missing values.

#### **Model Training:**

Fit the selected model to the training data. Make sure to use the features selected in the previous step.

### **Hyperparameter Tuning:**

Optimize the hyperparameters of the model to achieve the best performance. This can be done using techniques like grid search or random search.

### **Cross-Validation:**

Perform k-fold cross-validation to assess the model's performance and ensure it generalizes well to unseen data.

### **Model Evaluation:**

Evaluate the model on the validation set using appropriate metrics, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE). This step helps you determine how well your model is performing.

## **EVALUATION:**

In the evaluation phase, you assess how well your house price prediction model performs. Key steps include:

### **Model Metrics:**

Calculate and interpret the chosen evaluation metrics to understand the model's performance. You might also consider visualizations like residual plots.

### **Overfitting and Underfitting:**

Check for signs of overfitting or underfitting and make necessary adjustments to your model.

**Model Interpretability:** Depending on the model chosen, you may want to interpret the model's coefficients or feature importance to gain insights into the driving factors behind house prices.

### **Model Comparison:**

If you've experimented with multiple algorithms or model variants, compare their performance to choose the best one.

### **Deployment Considerations:**

Consider how the model will be used in a real-world application. Think about deployment options, scalability, and maintenance

### **Documentation:**

Document the entire process, including the features selected, model details, and evaluation results. This documentation will be valuable for future reference and collaboration.

## **OVERVIEW:**

### **Import Necessary Libraries:**

- Import the required libraries, including Pandas for data manipulation, NumPy for numerical operations, and various modules from Scikit-Learn for machine learning tasks.

### **Load the California Housing Dataset:**

- The code uses the `fetch_california_housing` function from Scikit-Learn to load the California housing dataset. It sets the `as_frame` parameter to `True`, which creates a DataFrame for the dataset. The dataset contains information about housing in various regions of California.

### **Split the Data:**

- The dataset is split into training and testing sets using the `train_test_split` function from Scikit-Learn. The data is divided into features (X) and target (y) with an 80-20 train-test split ratio.

### **pFeature Selection:**

The code creates a list of selected features, which, in this example, includes all the columns from the dataset. However, you can customize this list to select specific features if needed. Feature selection is an essential step in machine learning to decide which attributes are relevant for the model.

### **Create and Train the Model:**

. A linear regression model is created using the `LinearRegression` class from Scikit-Learn. The model is then trained using the training data (`X_train`) and target values (`y_train`). Linear regression is used for modeling the relationship between the features and the target variable.

### **Make Predictions:**

. The trained model is used to make predictions on the test set (`X_test`) by calling the `predict` method. The predicted values are stored in the `y_pred` variable.

### **Evaluate the Model:**

.Two evaluation metrics are computed to assess the performance of the linear regression model:

**Mean Squared Error (MSE):** It quantifies the average squared difference between the actual target values (`y_test`) and the predicted values (`y_pred`). A lower MSE indicates better model performance.

**R-squared (R2):** Also known as the coefficient of determination, it measures the proportion of the variance in the target variable that is predictable from the features. R2 values range from 0 to 1, with higher values indicating a better fit. It is often used as a goodness-of-fit measure.

## **Print the Results:**

.The code prints the calculated Mean Squared Error (MSE) and squared (R2) values to the console.

## **PROGRAM:**

```
import pandas as pd

# Load the dataset

data = pd.read_csv("C:\\Users\\Aneeta
Tejaswini\\Downloads\\USA_Housing.csv") # Replace with your
dataset file path

data = data.dropna() # Example: Removing rows with missing data

data = pd.get_dummies(data, columns=["Address"]) #
print(data.columns)


# If the column exists, apply one-hot encoding
if "Address" in data.columns:

    data = pd.get_dummies(data, columns=["Address"])
else:

    print("The 'address' column does not exist in the DataFrame.")
```

## **OUTPUT**

The 'address' column does not exist in the DataFrame.

## PROGRAM

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

data["Price"] = scaler.fit_transform(data["Price"].values.reshape(-1,
1))

from sklearn.model_selection import train_test_split

X = data.drop("Price", axis=1) # Features
y = data["Price"] # Target variable

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

from sklearn.linear_model import LinearRegression

model = LinearRegression()

from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score

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)

print("Mean Absolute Error:", mae)

print("Mean Squared Error:", mse)

print("R-squared:", r2)model.fit(X_train, y_train)
```

**OUTPUT:**

Mean Absolute Error: 0.2290742914490079

Mean Squared Error: 0.08093326122903559

R-squared: 0.9179914220230679

**PROGRAM:**

```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load the California housing dataset
data = fetch_california_housing(as_frame=True)
X, y = data.data, data.target

# 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)

# Feature Selection (You can use more advanced feature selection
techniques)
```

# In this example, we'll use all features, but you can select specific ones.

```
selected_features = list(X.columns)
```

# Create and train the model

```
model = LinearRegression()
```

```
model.fit(X_train[selected_features], y_train)
```

# Make predictions on the test set

```
y_pred = model.predict(X_test[selected_features])
```

# Evaluate the model

```
mse = mean_squared_error(y_test, y_pred)
```

```
r2 = r2_score(y_test, y_pred)
```

```
print("Mean Squared Error:", mse)
```

```
print("R-squared:", r2)
```

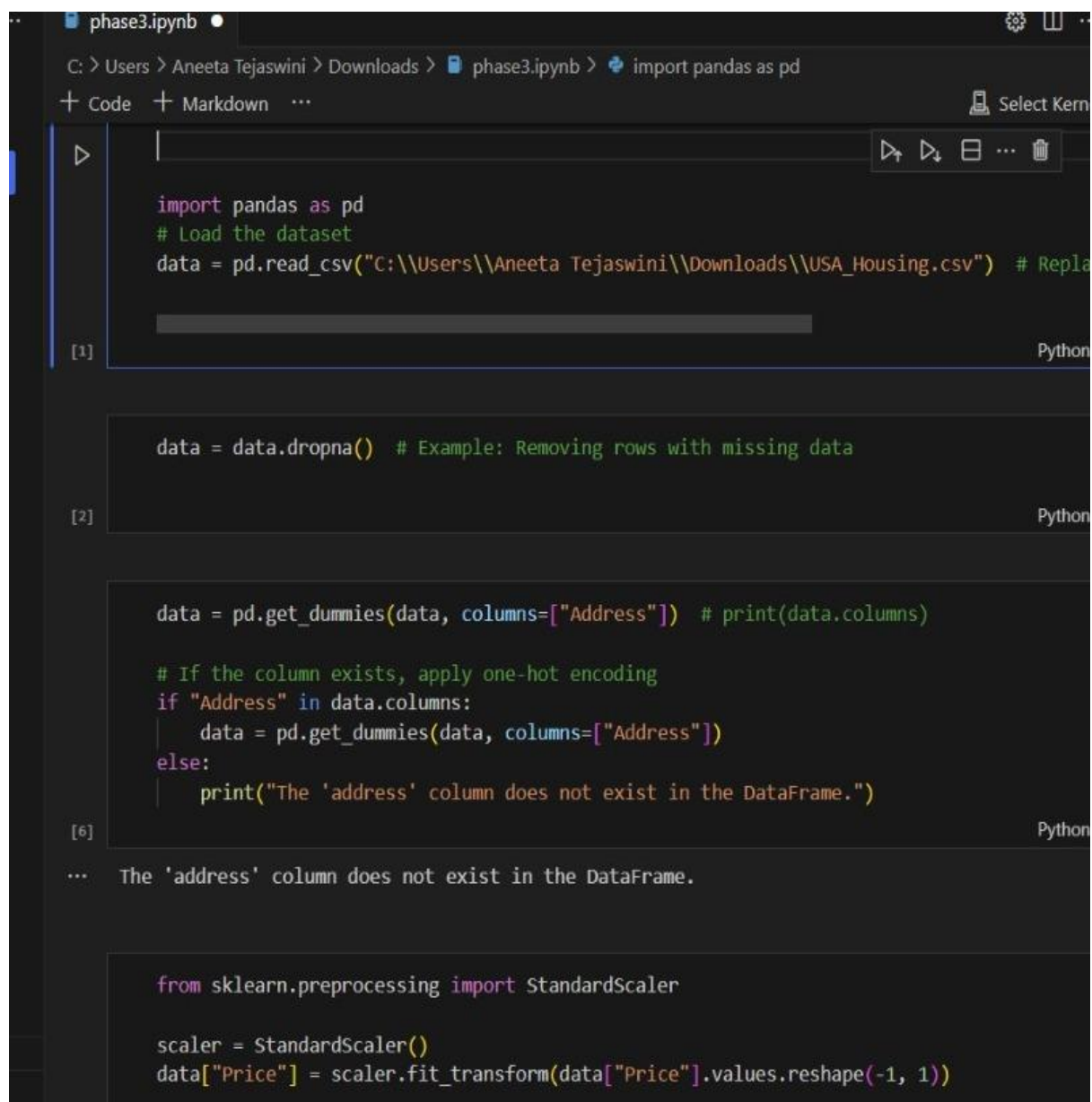
### **OUTPUT:**

Mean Squared Error: 0.5558915986952437

R-squared: 0.5757877060324512



## SCREENSHOT 1:



The screenshot shows a Jupyter Notebook interface with a dark theme. The file name is 'phase3.ipynb'. The path in the breadcrumb is 'C:\> Users > Aneeta Tejaswini > Downloads > phase3.ipynb'. The code is written in Python and is organized into several cells. The first cell contains code to import pandas and load a CSV file. The second cell contains code to drop rows with missing data. The third cell contains code to create dummy variables for the 'Address' column. The fourth cell shows the output of the previous cell, which is a message indicating that the 'address' column does not exist in the DataFrame. The fifth cell contains code to import StandardScaler from sklearn.preprocessing and fit it to the 'Price' column.

```
import pandas as pd
# Load the dataset
data = pd.read_csv("C:\\Users\\Aneeta Tejaswini\\Downloads\\USA_Housing.csv") # Repla

[1] Python

data = data.dropna() # Example: Removing rows with missing data

[2] Python

data = pd.get_dummies(data, columns=["Address"]) # print(data.columns)

# If the column exists, apply one-hot encoding
if "Address" in data.columns:
    data = pd.get_dummies(data, columns=["Address"])
else:
    print("The 'address' column does not exist in the DataFrame.")

[6] Python

... The 'address' column does not exist in the DataFrame.

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
data["Price"] = scaler.fit_transform(data["Price"].values.reshape(-1, 1))
```

## SCREENSHOT 2:

```
In [2]: #Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load the California housing dataset
data = fetch_california_housing(as_frame=True)
X, y = data.data, data.target

# 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)

# Feature Selection (You can use more advanced feature selection techniques)
# In this example, we'll use all features, but you can select specific ones.
selected_features = list(X.columns)

# Create and train the model
model = LinearRegression()
model.fit(X_train[selected_features], y_train)

# Make predictions on the test set
y_pred = model.predict(X_test[selected_features])
```

## SCREENSHOT 3:

```
# Load the California housing dataset
data = fetch_california_housing(as_frame=True)
X, y = data.data, data.target

# 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)

# Feature Selection (You can use more advanced feature selection techniques)
# In this example, we'll use all features, but you can select specific ones.
selected_features = list(X.columns)

# Create and train the model
model = LinearRegression()
model.fit(X_train[selected_features], y_train)

# Make predictions on the test set
y_pred = model.predict(X_test[selected_features])

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

Mean Squared Error: 0.5558915986952437

R-squared: 0.5757877060324512

## CONCLUSION:

- Consider any limitations or challenges encountered during the project, such as data quality issues or the need for more extensive feature engineering.
- If the model meets the desired performance criteria, it can be deployed for real-world predictions. If not, further iterations, feature engineering, or model tuning may be necessary.

utilizing machine learning for predicting house prices can be a valuable asset in the real estate industry. It offers the potential for more accurate and data-driven pricing, benefiting both buyers and sellers. However, it is essential to keep in mind that while machine learning models can provide valuable insights, they should be used in conjunction with human expertise and an understanding of the broader market context.

THANK YOU