

## **PHASE 4 ASSIGNMENT**

**PROJECT TITLE:** Feature selection, Model training, Evaluation of an dataset.

**PROBLEM DEFINITION:** 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 data preprocessing, feature engineering, model selection, training, and evaluation.

**GITHUB LINK:**

<https://github.com/Dharanimuthusamy4/predicting-house-price-using-machine-learning.git>

<https://github.com/Dharanimuthusamy4/Innovation.git>

**DOCUMENT:**

**Building the project by Feature selection, Model training, Evaluation of an dataset.**

**DATASET LINK ON: Predicting House Prices**

<https://www.kaggle.com/datasets/vedavyasv/usa-housing>

Creating a house price prediction model involves several key steps, including feature selection, model training, and evaluation. Here's a step-by-step guide to help you build such a model:

1. **Data Collection and Preparation:**

- Gather a dataset that includes information about houses and their sale prices. Common features might include square footage, number of bedrooms and bathrooms, location, etc.
- Preprocess the data by handling missing values, encoding categorical variables (e.g., one-hot encoding), and scaling numerical features if necessary.

## **2. Feature Selection:**

- Feature selection is crucial for building an effective model. You want to choose the most relevant features to predict house prices. There are various methods for feature selection, such as:
  - Correlation analysis: Identify features that have a strong correlation with the target variable (e.g., using a correlation matrix).
  - Recursive Feature Elimination (RFE): Use techniques like RFE to iteratively remove the least important features.
  - Feature importance from tree-based models: If you plan to use decision tree-based models (e.g., Random Forest), you can use feature importance scores.

## **3. Data Splitting:**

- Split the dataset into a training set and a testing set (e.g., 80% for training and 20% for testing) to evaluate the model's performance.

## **4. Model Selection:**

- Choose a machine learning model suitable for regression tasks. Some common choices include Linear Regression, Decision Trees, Random Forest, Support Vector Machines, and Gradient Boosting models (e.g., XGBoost).

## **5. Model Training:**

- Fit the selected model to the training data using the features you've chosen.
- Tune hyperparameters, if necessary, using techniques like cross-validation or grid search.

## **6. Model Evaluation:**

- Use the testing dataset to evaluate your model. Common evaluation metrics for regression problems include:

- Mean Absolute Error (MAE): The average absolute difference between predicted and actual prices.
- Mean Squared Error (MSE): The average of the squared differences between predicted and actual prices.
- Root Mean Squared Error (RMSE): The square root of MSE, providing a measure in the original unit of the target variable.
- R-squared (R2) score: A measure of how well the model explains the variance in the target variable.

#### 7. Visualization and Interpretation:

- Visualize the model's predictions against the actual prices to understand how well it performs. You can use scatter plots or residual plots for this purpose.
- Interpret the model's coefficients or feature importances to understand the impact of each feature on house prices.

#### 8. Fine-Tuning and Iteration:

- Based on the evaluation results and interpretation, you may need to make adjustments to your model. This could involve adding or removing features, changing the model, or fine-tuning hyperparameters.

#### 9. Deployment:

- Once you are satisfied with the model's performance, you can deploy it in a real-world application where it can make predictions on new, unseen data.

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LinearRegression
```

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

```
# Split the data
```

```
X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.2, random_state=42)
```

```
# Create and train the model
```

```
model = LinearRegression()
```

```
model.fit(X_train, y_train)

# Make predictions

y_pred = model.predict(X_test)

# 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)
```

#### **10. Monitoring and Maintenance:**

- Continuously monitor the model's performance and update it as necessary to ensure it remains accurate and relevant.

**SUBMITTED BY,**

**STUDENT REG NO:** 711221104012

**NAAN MUDHALVAN:** au711221104012