Problem Statement:

Problem Statement: Create a Machine Learning-based House Price Prediction System that predicts the market price of houses based on various features like size, location, amenities, and more. The system aims to provide accurate price estimates for real estate properties to assist buyers, sellers, and real estate professionals in making informed decisions.

Design Thinking Process:

Empathize: Understand the needs and concerns of homebuyers, sellers, and real estate professionals. Gather insights into the factors that influence house prices and market trends.

Define: Define the problem by identifying key features that impact house prices. Set clear objectives for the system, including accurate price estimation and usability.

Ideate: Brainstorm potential solutions and strategies for predicting house prices. Consider factors such as property size, location, number of bedrooms, amenities, and market demand.

Prototype: Develop a prototype of the Machine Learning-based House Price Prediction System that can analyze data and provide accurate price estimates.

Test: Test the prototype on real estate data and validate its accuracy and usability. Gather feedback from users and real estate experts for further improvements.

Implement: Implement the Machine Learning-based system, collaborate with real estate agencies, and integrate it into their online platforms for real-time price estimation.

Iterate: Continuously refine the system based on user feedback and changing real estate market conditions.

Phases of Development:

Data Collection: Gather real estate data, including property features (e.g., size, location, amenities), historical price data, and market trends.

Data Preprocessing: Clean the data, handle missing values, perform feature engineering, and transform data into a suitable format for machine learning.

Feature Selection: Choose relevant features that significantly influence house prices, considering factors like correlation with prices and market trends.

Model Development: Select a machine learning algorithm suitable for regression tasks (e.g., Linear Regression, Gradient Boosting) to predict house prices.

Model Training: Train the machine learning model on historical real estate data to learn the relationships between features and house prices.

Model Evaluation: Assess the model's performance using appropriate evaluation metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2).

Deployment: Deploy the Machine Learning-based House Price Prediction System for real-time price estimation on real estate websites and platforms.

Dataset Description:

Data Sources: Real estate databases, property listings, historical sales data, market statistics, geographic information systems (GIS), and external data sources.

Data Preprocessing:

Handle missing values, convert data types, perform feature engineering, and encode categorical variables.

Normalize and standardize features.

Remove outliers and noise from the data.

Model Training:

Split data into training and testing sets.

Train the selected machine learning model on historical real estate data.

Evaluation Metrics:

Use regression evaluation metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2) to measure the model's accuracy in predicting house prices.

Innovative Techniques or Approaches: Implement geographic information systems (GIS) data for spatial analysis and pricing based on location.

Submisson :

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score

# Step 1: Load the Dataset
data = pd.read_csv("house_prices.csv")

# Step 2: Data Preprocessing
# Your data preprocessing code here

# Step 3: Data Splitting
X = data.drop("Price", axis=1)
y = data["Price"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Step 4: Model Training
model = RandomForestRegressor(n_estimators=100,
random_state=42)
model.fit(X_train, y_train)

# Step 5: Model Evaluation
```

```
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(f"Mean Absolute Error: {mae}")
print(f"Mean Squared Error: {mse}")
print(f"R-squared (R2) Score: {r2}")
```