PREDICTING HOUSE PRICES USING MACHINE LEARNING

Phase 1: Problem Definition and Design Thinking

Problem Definition:

Certainly, here's a problem definition for a project on predicting house prices using machine learning:

Project Title: House Price Prediction Using Machine Learning

Project Description: This project aims to develop a machine learning model that can accurately predict house prices based on various features such as square footage, number of bedrooms, location, and other relevant factors. The goal is to provide homebuyers, sellers, and real estate professionals with a reliable tool to estimate property values, aiding in informed decision-making.

Project Objectives:
1. Build a predictive model that achieves a low mean absolute error (MAE) or root mean squared error (RMSE) for house price predictions.
2. Create a user-friendly interface or API for users to input property details and receive price estimates.
3. Evaluate the model's accuracy and performance through rigorous testing and validation.
4. Provide insights into the key features that influence house prices in the selected housing market.

Scope:
- The project will focus on a specific housing market or region.
- Data will be collected from reliable sources and preprocessed for modeling.
- The model will use regression techniques to predict house prices.
- The project does not involve real estate transactions but aims to provide estimates based on historical data.

Stakeholders:
- Homebuyers and sellers
- Real estate agents and agencies
- Data scientists and analysts
- Local housing authorities (for potential policy insights)

Deliverables:
1. Trained machine learning model for house price prediction.
2. Interactive interface or API for users to make predictions.
3. Documentation on model development and usage.
4. Report on key factors influencing house prices in the chosen market.

Constraints:
- Budget constraints for data acquisition and model development.
- Availability of historical housing data.
- Compliance with data privacy and ethical considerations.

Risks:
- Data quality issues or missing data in the dataset.
- Model overfitting or underfitting.
- External factors like economic changes affecting housing prices.

Timeline:
- The project is expected to be completed within [insert timeline here], with regular progress updates.

Resources:
- Data sources, including historical property data.
- Data scientists and machine learning experts.
- Software and hardware resources for model development.

Budget:
- A budget of [insert budget amount] is allocated for data acquisition, software, and hardware resources.

Methodology/Approach:
- Utilize regression algorithms such as Linear Regression, Decision Trees, and Gradient Boosting.
- Feature engineering to select and preprocess relevant variables.
- Cross-validation and hyperparameter tuning to optimize model performance.

Success Criteria:
- The model achieves a MAE/RMSE within [insert target value].
- Users find the interface/API intuitive and valuable.
- The project provides insights into the local housing market.

Communication Plan:
- Regular project updates to stakeholders.
- Documentation for model usage.
- Final presentation and report on project outcomes.

Approval and Sign-off:
- Project sponsors and stakeholders will review and approve the project plan.

This problem definition provides a comprehensive overview of the project's goals, scope, stakeholders, and constraints, setting the stage for successful house price prediction using machine learning.

Design Thinking:

Design thinking is a creative and user-centric approach to problem-solving. When applied to a project like predicting house prices using machine learning, it can help ensure that the resulting model is not only accurate but also valuable and user-friendly. Here's a simplified design thinking process for this project:

1. **Empathize: Understand User Needs**
   - Conduct interviews or surveys with potential users (homebuyers, sellers, real estate agents) to understand their pain points, goals, and expectations regarding house price predictions.

2. **Define: Frame the Problem**
   - Clearly define the problem statement based on user insights, e.g., "How might we provide accurate and accessible house price estimates to assist in real estate transactions?"

3. **Ideate: Generate Solutions**
   - Brainstorm various approaches to solving the problem. Consider different machine learning algorithms, data sources, and user interface designs.
   - Encourage creative thinking to identify novel solutions.

4. **Prototype: Create a Minimum Viable Product (MVP)**
   - Develop a simplified version of the house price prediction model using a subset of data and a basic user interface.
   - This MVP should be a working model that demonstrates the core functionality.

5. **Test: Gather User Feedback**
   - Invite users to interact with the prototype and collect feedback. Understand how well it meets their needs and where it falls short.
   - Iteratively refine the prototype based on user input.

6. **Refine: Iterate and Improve**
   - Use the feedback from testing to make improvements to the model and user interface.
   - Continue to iterate and refine the solution until it aligns with user expectations.

7. **Develop: Build the Full-Fledged Solution**
   - Once the prototype meets user needs and expectations, proceed to develop the complete house price prediction model and user interface.
   - Utilize machine learning best practices for data preprocessing, model training, and validation.

8. **Test Again: Validate the Full Solution**
   - Conduct thorough testing of the final model and interface to ensure accuracy, reliability, and usability.
   - Address any issues or bugs that arise during this phase.

9. **Implement: Deploy the Solution**
   - Launch the house price prediction tool, making it accessible to users.
   - Ensure that it's integrated seamlessly into the real estate market or platform.

10. **Evaluate and Improve: Monitor User Feedback**
    - Continuously gather user feedback and monitor the performance of the model in real-world scenarios.
    - Make updates and improvements as necessary to maintain accuracy and user satisfaction.

11. **Scale: Expand Usage**
    - If successful, consider scaling the solution to cover larger geographic areas or additional markets.

Throughout the design thinking process, keep the end-users at the center of your decision-making. By empathizing with their needs, iterating based on their feedback, and continuously improving the solution, you can create a house price prediction model that not only predicts accurately but also provides a positive and valuable user experience.

Solving the problem of predicting house prices using machine learning involves several steps. Here's a simplified guide to get you started:

1. **Data Collection**:
   - Gather a dataset of historical housing prices. You can find such datasets on websites like Kaggle, or you may need to collect data from local sources.
   - Include relevant features like the number of bedrooms, bathrooms, square footage, location, and any other factors that influence house prices.

2. **Data Preprocessing**:
   - Handle missing data: Determine how to deal with missing values, either by imputing them or removing rows with missing data.
   - Feature engineering: Create new features or transform existing ones if it improves the model's performance.

3. **Data Splitting**:
   - Split your dataset into training and testing sets. A common split is 80% for training and 20% for testing.

4. **Model Selection**:

- Choose a regression algorithm suitable for predicting continuous values. Common choices include Linear Regression, Decision Trees, Random Forests, or Gradient Boosting.

5. **Model Training**:
   - Train your selected model on the training dataset using the features to predict house prices.

6. **Model Evaluation**:
   - Use evaluation metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), or R-squared to assess the model's performance on the testing dataset.
   - You may need to fine-tune hyperparameters to improve model accuracy.

7. **Model Deployment**:
   - Once you are satisfied with the model's performance, you can deploy it. Deployment methods vary depending on your application, but it could involve creating a web application or API to take user inputs and provide predictions.

8. **Continuous Monitoring and Maintenance**:
   - Regularly update your model with new data to ensure it stays accurate over time.
   - Monitor for model drift or changes in data distribution that could affect predictions.

9. **User Interface:**
   - Create a user-friendly interface for users to input property details and receive price predictions. This could be a web app, mobile app, or even a simple form.

10. **Feedback Loop**:
    - Encourage users to provide feedback on the predictions and use this feedback to make improvements to the model.

Remember that predicting house prices is a complex task influenced by various factors, and the accuracy of your model will depend on the quality and quantity of data, feature selection, and the choice of the machine learning algorithm. It's also essential to consider ethical and legal implications, especially when dealing with sensitive data.

11.Data Source

A good data source for house price prediction using machine learning should be Accurate, Complete, Covering the geographic area of interest, Accessible.

Dataset Link: **https://www.kaggle.com/datasets/vedavyasv/usa-housing/code**

Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
79545.45857	5.682861322	7.009188143	4.09	23086.8005	1059033.56	208
79248.64245	6.002899808	6.730821019	3.09	40173.07217	1505890.91	188
61287.06718	5.86588984	8.51272743	5.13	36882.1594	1058987.99	9127
63345.24005	7.188236095	5.586728665	3.26	34310.24283	1260616.81	USS
59982.19723	5.040554523	7.839387785	4.23	26354.10947	630943.489	USNS
80175.75416	4.988407758	6.104512439	4.04	26748.42842	1068138.07	06039
64698.46343	6.025335907	8.147759585	3.41	60828.24909	1502055.82	4759
78394.33928	6.989779748	6.620477995	2.42	36516.35897	1573936.56	972 Joyce
59927.66081	5.36212557	6.393120981	2.3	29387.396	798869.533	USS
81885.92718	4.42367179	8.167688003	6.1	40149.96575	1545154.81	Unit 9446
80527.47208	8.093512681	5.0427468	4.1	47224.35984	1707045.72	6368
50593.6955	4.496512793	7.467627404	4.49	34343.99189	663732.397	911
39033.80924	7.671755373	7.250029317	3.1	39220.36147	1042814.1	209
73163.66344	6.919534825	5.993187901	2.27	32326.12314	1291331.52	829
69391.38018	5.344776177	8.406417715	4.37	35521.29403	1402818.21	PSC 5330,
73091.86675	5.443156467	8.517512711	4.01	23929.52405	1306674.66	2278
79706.96306	5.067889591	8.219771123	3.12	39717.81358	1556786.6	064
61929.07702	4.788550242	5.097009554	4.3	24595.9015	528485.247	5498
63508.1943	5.94716514	7.187773835	5.12	35719.65305	1019425.94	Unit 7424
62085.2764	5.739410844	7.091808104	5.49	44922.1067	1030591.43	19696
86294.99909	6.62745694	8.011897853	4.07	47560.77534	2146925.34	030 Larry
60835.08998	5.551221592	6.517175038	2.1	45574.74166	929247.6	USNS
64490.65027	4.21032287	5.478087731	4.31	40358.96011	718887.232	95198
60697.35154	6.170484091	7.150536572	6.34	28140.96709	743999.819	9003 Jay
59748.85549	5.339339881	7.748681606	4.23	27809.98654	895737.133	24282

12.Data Preprocessing

Data preprocessing is the critical first step in any machine learning project. It involves cleaning the data, removing outliers, and handling missing values to prepare the dataset for model training. In the context of the house price prediction project, let's elaborate on the specific steps:

a) Duplicate Removal:

Duplicate rows can introduce bias into the model. We will identify and remove duplicates, typically by sorting the dataset based on a unique identifier (e.g., property ID) and then eliminating consecutive rows with the same identifier.

b) Handling Missing Values:

Missing data is common and needs to be addressed. We will utilize suitable methods such as:

 Mean Imputation: Replace missing values with the mean of the feature for the remaining rows. This is appropriate for numerical features.

 Median Imputation: If data contains outliers, median imputation can be more robust as it is less sensitive to extreme values.

c) Categorical Variable Encoding:

Categorical variables, such as property type or location, need to be converted into numerical form so that machine learning models can process them. Two common approaches include:

 One-Hot Encoding: Create binary columns for each category, representing the presence or absence of that category.

 Label Encoding: Assign a unique integer to each category, preserving the ordinal relationship if applicable.

d) Data Normalization:

To ensure that all features are on a consistent scale, normalization techniques can be

applied. This includes:
 Standardization: Scaling features to have a mean of 0 and a standard deviation of 1.
 Min-Max Scaling: Scaling features to a specified range (e.g., 0 to 1)
PYHON PROGRAM:

```
# Import necessary libraries
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
# Load the dataset (replace 'house_data.csv' with your dataset file)
data = pd.read_csv('E:/USA_Housing.csv')
# Display the first few rows of the dataset to get an overview
print("Dataset Preview:")
print(data.head())
# Data Preprocessing
# 1. Handle Missing Values
# Let's fill missing values in numeric columns with the mean and in categorical columns
with the most frequent value.
numeric_cols = data.select_dtypes(include='number').columns
categorical_cols = data.select_dtypes(exclude='number').columns
imputer_numeric = SimpleImputer(strategy='mean')
imputer_categorical = SimpleImputer(strategy='most_frequent')
data[numeric_cols] = imputer_numeric.fit_transform(data[numeric_cols])
data[categorical_cols] = imputer_categorical.fit_transform(data[categorical_cols])
# 2. Convert Categorical Features to Numerical
# We'll use Label Encoding for simplicity here. You can also use one-hot encoding for
nominal categorical features.
label_encoder = LabelEncoder()
for col in categorical_cols:
data[col] = label_encoder.fit_transform(data[col])
# 3. Split Data into Features (X) and Target (y)
X = data.drop(columns=['Price']) # Features
y = data['Price'] # Target
# 4. Normalize the Data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Split data into training and testing sets (adjust test_size as needed)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)
# Display the preprocessed data
print("\nPreprocessed Data:")
print(X_train[:5]) # Display first 5 rows of preprocessed features
print(y_train[:5]) # Display first 5 rows of target values
```

OUTPUT:

Dataset Preview:

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

Preprocessed Data:

[[-0.19105816 -0.13226994 -0.13969293 0.12047677 -0.83757985 -1.00562872]

[-1.39450169 0.42786736 0.79541275 -0.55212509 1.15729018 1.61946754]

[-0.35137865 0.46394489 1.70199509 0.03133676 -0.32671213 1.63886651]

[-0.13944143 0.1104872 0.22289331 -0.75471601 -0.90401197 -1.54810704]

[0.62516685 2.20969666 0.42984356 -0.45488144 0.12566216 0.98830821]]

4227 1.094880e+06

4676 1.300389e+06

800 1.382172e+06

3671 1.027428e+06

4193 1.562887e+06

Name: Price, dtype: float64

3.Feature selection :

Feature Selection is the process of identifying and selecting the most relevant features from a dataset for a given machine learning task. The goal of feature selection is to improve the performance of the machine learning model by reducing the number of features and eliminating irrelevant or redundant features.

There are a variety of feature selection techniques. Some of the most common techniques include:

 Correlation-based feature selection: This technique selects features based on their correlation with the target variable. Features with high correlation with the target variable are more likely to be relevant for predicting the target variable, so they are selected.

 Information gain-based feature selection: This technique selects features based on their information gain. Information gain measures how much information a feature

provides about the target variable. Features with high information gain are more likely to be relevant for predicting the target variable, so they are selected.
 Recursive feature elimination (RFE): This technique starts with all features and then recursively removes the least important feature until a desired number of features are remaining. The importance of a feature is measured using a variety of methods, such as cross-validation or the coefficient of determination (R-squared).
 Model selection is the process of choosing a suitable machine learning algorithm for a given machine learning task. The goal of model selection is to find an algorithm that is both accurate and efficient.
There are a variety of machine learning algorithms that can be used for house price prediction. Some of the most common algorithms include:
 Linear regression: Linear regression is a simple but effective algorithm for house price prediction. Linear regression models the relationship between the house price and the features using a linear function.
 Random forest regressor: Random forest regressor is a more complex algorithm that builds a multitude of decision trees to predict the house price. Random forest regressors are typically more accurate than linear regression models, but they can be more computationally expensive to train.
 Gradient boosting regressor: Gradient boosting regressor is another complex algorithm that builds a sequence of decision trees to predict the house price. Gradient boosting regressors are typically more accurate than random forest regressors, but they can be even more computationally expensive to train.

4. Model Selection:

Choose machine learning algorithms suitable for regression tasks. Common models for predicting house prices include linear regression, decision trees, random forests, gradient boosting, and neural networks
(e.g., deep learning models).
Experiment with multiple algorithms to determine which one provides the best performance for your specific dataset. You can also consider ensemble methods.
PYTHON PROGRAM:

```
# Import necessary libraries
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
selector = SelectKBest(score_func=f_regression, k=k)
X_train_selected = selector.fit_transform(X_train, y_train)
# Model Selection
# Let's choose both Linear Regression and Random Forest Regressor for comparison.
linear_reg_model = LinearRegression()
random_forest_model = RandomForestRegressor(n_estimators=100, random_state=42)
# Train the models on the selected features
linear_reg_model.fit(X_train_selected, y_train)
```

```
random_forest_model.fit(X_train_selected, y_train)
# Evaluate the models on the test set
X_test_selected = selector.transform(X_test)
# Make predictions
linear_reg_predictions = linear_reg_model.predict(X_test_selected)
random_forest_predictions = random_forest_model.predict(X_test_selected)
# Evaluate model performance
def evaluate_model(predictions, model_name):
mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
print(f"{model_name} Model Evaluation:")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2) Score: {r2}\n")
# Performance Measure
elr_mse = mean_squared_error(y_test, pred)
elr_rmse = np.sqrt(lr_mse)
elr_r2 = r2_score(y_test, pred)
# Show Measures
result = '''
MSE : {}
RMSE : {}
R^2 : {}
'''.format(lr_mse, lr_rmse, lr_r2)
print(result)
# Model Comparision
names.append("elr")
mses.append(elr_mse)
rmses.append(elr_rmse)
r2s.append(elr_r2)
evaluate_model(linear_reg_predictions, "Linear Regression")
evaluate_model(random_forest_predictions, "Random Forest Regressor")
```

OUTPUT:

```
Linear Regression Model Evaluation:
Mean Squared Error (MSE): 10089009300.893988
R-squared (R2) Score: 0.9179971706834331
Random Forest Regressor Model Evaluation:
Mean Squared Error (MSE): 14463028828.265167
R-squared (R2) Score: 0.8824454166872736
MSE : 10141766848.330585
RMSE : 100706.33966305491
R^2 : 0.913302484308253
```

5.Model Training :

The task involves training a selected machine learning model using preprocessed dat
a and subsequently evaluating the model's performance using key metrics such as Mean Ab

solute Error (MAE), Root Mean Squared Error (RMSE), and R-squared.

6.Evaluation Metrics:

Mean Absolute Error (MAE):

MAE measures the average absolute difference between the predicted values and the actual target values. It provides insight into the average magnitude of errors made by the model.

Root Mean Squared Error (RMSE):

RMSE is another common metric that calculates the square root of the average of squared differences between predicted and actual values. It provides information about the typical magnitude of errors and gives higher penalties to larger errors.

S-squared (R2):

T-R-squared quantifies the proportion of the variance in the target variable that is explained by the model. It ranges from 0 to 1, where a higher value indicates a better fit. It is often used to assess how well the model captures the variation in the data.

Feature Engineering

Feature engineering is the process of creating new features or transforming existing ones to provide more meaningful information to the machine learning model. For the house price prediction project, we can consider the following feature engineering techniques:

1. Age of the House: Create a new feature that represents the age of each house by subtracting the year built from the current year. This feature can be informative as older houses may have different pricing dynamics compared to newer ones.
2. Square Footage per Bedroom: Calculate a new feature by dividing the total square footage of a property by the number of bedrooms. This metric can provide insights into the spaciousness of bedrooms, which can be a key factor in house pricing.
3. Bathrooms per Square Foot: Compute a new feature by dividing the number of bathrooms by the total square footage. This can capture the luxury level of bathrooms relative to the property's size.
4. School District Quality: Integrate external data from a third-party API to assess the quality of the school district in which each house is located. Properties situated in neighborhoods with better school districts often command higher prices.

Phase 1 Deliverables:

For Phase 1, the following deliverables are expected:

1. Clean and Preprocessed Dataset:

Provide a dataset where duplicates have been removed, missing values have been addressed, and categorical variables have been encoded. This dataset should be ready for use in machine learning models.

2. List of Engineered Features:

Document all the newly created features along with their descriptions and how they were calculated. This list should make it clear how each engineered feature adds value to the predictive model.

3. Data Preprocessing and Feature Engineering Report:

Create a detailed report that outlines the steps taken during data preprocessing and feature engineering. This report should cover the methods used, challenges encountered, and

the rationale behind the decisions made at each step.

Next Steps

In Phase 2 of the project, we will proceed with the following tasks:

 Model Selection and Training: Select an appropriate machine learning algorithm(s) for house price prediction, train the chosen model(s) on the preprocessed data, and fine-tune hyperparameters if necessary.

 Model Evaluation: Evaluate the model's performance using various evaluation metrics on a held-out test dataset to assess its accuracy and generalization ability.

Additional Considerations

In addition to the core data preprocessing and feature engineering steps, it's important to keep in mind the following considerations:

1. Machine Learning Algorithm Choice: The choice of machine learning algorithm(s) should be based on the dataset's characteristics and the desired performance metrics. Options include linear regression, random forests, gradient boosting, and neural networks.

2. Cross-Validation: Implement cross-validation to assess the model's performance without overfitting. Cross-validation involves splitting the data into folds, training on one subset, and evaluating on the others to obtain a more robust estimate of performance.

3. Interpreting Model Results: After training the model, focus on interpreting the results. Understand which features are most influential for predicting house prices, and explore how the model's predictions change as feature values vary. Visualization and feature importance analysis can aid in interpretation.

CONCLUSION:

In Phase 1, we have established a clear understanding of our goal: to predict house prices using machine learning. We outlined a structured approach that includes data source selection, data preprocessing, feature selection, model selection, model training, and evaluation. This sets the stage for our project's successful execution in subsequent phases