PHASE 5: PROJECT DOCUMENTATION AND SUBMISSION

Documentation

- Design thinking is a human-centered approach to problem-solving. In the context of predicting house prices using machine learning, we can break down the process into several key phases:

➔ Empathize
➔ Define
➔ Ideate
➔ Prototype
➔ Test

Phase Development:

The development of the predictive house price model can be broken down into several phases, including data collection and preprocessing, model development, and deployment. Here's a high-level overview of these phases:

Data Collection:

Gather historical housing data from various sources, including property listings, public records, and real estate websites.Collect features such as location, square footage, number of bedrooms, amenities, and recent sale prices.

Data Preprocessing:
Clean the data by handling missing values and outliers. Normalize or scale numerical features. Encode categorical features. Split the data into training, validation, and test sets.

Model Development:

Choose and implement machine learning algorithms, such as regression models (e.g., linear regression), ensemble models (e.g., random forests), or deep learning models (e.g., neural networks). Train the model on the training data. Optimize hyperparameters and select the best-performing model.

Evaluation and Validation:

Assess the model's performance using validation data, using appropriate evaluation metrics. Fine-tune the model based on validation results.

Used Data set :

'bedrooms':[2],'bathrooms':[2.5],'sqft_living':[600],'sqft_lot':[600],'floors': [2],'zipcode': [98008]

Model training process:

1.Data Collection.
2.Data Preprocessing.
3.Feature Selection/Engineering.
4.Split Data.
5.Select a Model.
6.Model Training.
7.Model Evaluation.
8.Hyperparameter Tuning.
9.Cross-Validation.
10.Model Deployment.
11.Monitoring and 12.Maintenance.

Data preprocessing steps:

1.Data Collection.
2.Data Cleaning.
3.Feature Selection/Engineering.
4.Data Scaling/Normalization.
5.Categorical Encoding.
6.Train-Test Split.
7.Model Building.
8.Model Evaluation.
9.Hyperparameter Tuning.
10.Prediction.

Machine Learning Algorithms:

Linear Regression: A simple and widely used algorithm for predicting continuous values like house prices.
Decision Trees: These can capture non-linear relationships in the data and are easy to interpret.
Random Forest: An ensemble method based on decision trees that can improve predictive accuracy.

Gradient Boosting (e.g., XGBoost, LightGBM): These algorithms often provide high predictive performance.
Neural Networks: Deep learning models can handle complex data patterns but may require a large amount of data and computational resources.
Model Training Techniques:

Data Preprocessing: This includes handling missing values, encoding categorical variables, and scaling features.
Cross-Validation: Use techniques like k-fold cross-validation to assess model performance.
Hyperparameter Tuning: Optimize hyperparameters for your chosen algorithm, such as learning rate, tree depth, or neural network architecture.
Feature Selection/Engineering: Select relevant features or create new ones that may improve the model's accuracy.
Evaluation Metrics:

Mean Absolute Error (MAE): The average absolute difference between predicted and actual house prices. It provides a straightforward measure of model accuracy.
Mean Squared Error (MSE): The average of the squared differences between predicted and actual prices. MSE penalizes larger errors more heavily.
Root Mean Squared Error (RMSE): The square root of MSE, which is in the same unit as the target variable, making it easier to interpret.
R-squared (R2) Score: Measures the proportion of the variance in the target variable that is predictable by the model. A higher R2 indicates a better model fit.
Median Absolute Error (MedAE): Robust to outliers as it calculates the median of the absolute errors.
Percentage Error: This can provide insights into the relative error, which is useful for some business applications.

SUBMISSION

Predicting House Price using Machine Learning

With the introduction of the power of machine learning in predicting house prices using Python has revolutionized the real estate industry. In this article, we explore the dynamic world of house price prediction using cutting-edge machine-learning techniques. By harnessing the vast potential of data analysis, feature engineering, and model training in Python, we aim to provide a comprehensive guide that equips readers with the tools to make informed decisions in the ever-changing housing market.

Linear regression for house price prediction:

Linear regression is a mainly used technique for the prediction of house prices due to its simplicity and interpretability. It assumes a linear relationship between the independent variables (such as how many bedrooms, number of bathrooms, and square footage) and the dependent variable (house price). By fitting a linear regression model to historical data, we can estimate the coefficients that represent the relationship between the target variable and the features. This enables us to make predictions on new data by multiplying the feature values with their respective coefficients and summing them up. Linear regression provides insights into the impact of each feature on the house price, enabling us to understand the significance of different factors and make informed decisions in the real estate market.

House price prediction using machine learning

Machine learning involves training a computer to recognize patterns and make predictions based on data. In the case of house price prediction, we can use historical data on various features of a house, such as its location, size, and amenities, to train a machine-learning model. Once the model is

trained, it can analyse new data on a given house and make a prediction of its market value.

House price prediction using machine learning(Linear regression model)

Follow the steps given below to perform the prediction of house prices using machine learning −

We have used Kaggle kc_house_data dataset.

Import the required libraries and modules, including pandas for data manipulation, scikit-learn for machine learning algorithms, and LinearRegression for the linear regression model.

Loading the required dataset with pd.read_csv and select the features we want to use for prediction (e.g., bedrooms, bathrooms, sqft_living, sqft_lot, floors, and zip code), as well as the target variable (price).

Split the data into a training set and a test set using the train_test_split function, with 80% of the data used for training and 20% for testing.

Create an instance of the linear regression model using LinearRegression(). We then perform the model training by calling the function fit() with the training data.

Once the model is trained, we make predictions for the test data set using predict and store the results in y_pred.
To evaluate the performance of the model, we calculate the R^2 score using the score for the test set.

Demonstrate how to predict the price of a new house by creating a new dataframe new_house with the features of the house. We pass this dataframe to the model's prediction function to obtain the predicted price.

Example (Algorithm)

from sklearn.model_selection import train_test_split

```
from sklearn.linear_model import LinearRegression
import pandas as pdd

# Loading the dataset
data_h = pdd.read_csv('kc_house_data.csv')

# Selecting the features and target variable
Features1 = ['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'zipcode']
target = 'price'
X1 = data_h[features1]
y1 = data_h[target]

# We will perform the data splitting into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.2, random_state=42)

# instance of the Linear Regression model creation
model = LinearRegression()

# Training the model
model.fit(X_train, y_train)

# Making predictions on the test set
y_pred = model.predict(X_test)

# Evaluating the model
score = model.score(X_test, y_test)
print("Model R^2 Score:", score)
# Predicting the price of a new house
new_house = pdd.DataFrame({'bedrooms': [2], 'bathrooms': [2.5], 'sqft_living': [600], 'sqft_lot': [600], 'floors': [2], 'zipcode': [98008]})
predicted_price = model.predict(new_house)
print("Predicted Price:", predicted_price[0]
```

Output:

Model R^2 Score: 0.5152176902631012
Predicted Price: 121215.61449578404

Predicting house prices using machine learning typically involves using a regression algorithm. Below is a simple example using Python and the scikit-learn library to build a linear regression model to predict house prices.

***Program**:*

1. Import libraries

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

2. Load the dataset

Load your dataset into a Pandas DataFrame. You can typically findhouse price datasets in CSV format, but you can adapt this code to otherformats as needed.

```
#Load the dataset
data = pd.read_csv('your_dataset.csv')
```

3. Define the features (X) and target (y)

```
X = data[['feature1', 'feature2', ...]]
```

4. Replace with the feature columns

```
y = data['target_column']  # Replace with the target column
```

5. 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)
```

6. Create a Linear Regression model

```
model = LinearRegression()
```

7. Fit the model on the training data

```
model.fit(X_train, y_train)
```

8. Make predictions on the test data

```
y_pred = model.predict(X_test)
```

9. Calculate the Mean Squared Error to evaluate the model

```
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
```

10. Now use the trained model to predict house prices

```
new_data = np.array([[value1, value2, ...]])
```

11. Replace with your new data

```
predicted_price = model.predict(new_data)
print(f"Predicted House Price: {predicted_price[0]}")
```

Make sure to replace 'our_dataset.csv' with the path to our dataset file, 'feature1', 'feature2', etc. with the actual feature columns we have, and 'target_column' with the actual target column (house prices). We can also customize the features and dataset as needed.

Start building the house price prediction model by loading and preprocessing the dataset.

To load and preprocess a housing dataset for the given data frame, we would typically need access to the dataset.

1. Load the Housing Dataset: First, we need to load the housing dataset. We can use libraries like Pandas in Python to read the dataset from a CSV file or another data source.

2. Select Data: Since we have a specific set of data we want to use, filter the dataset to include only the rows that match your criteria. In our case, we want to select rows with 'bedrooms' equal to 2, 'bathrooms' equal to 2.5, 'sqft_living' equal to 600, 'sqft_lot' equal to 600, 'floors' equal to 2, and 'zipcode' equal to 98008.

3. Preprocess Data: Once we selected the relevant rows, we can perform preprocessing steps such as handling missing values, scaling, and encoding categorical features. Since our provided data seems to be numeric, we may not need extensive preprocessing.

Here's a Python code to give an idea of how this might look with a sample dataset:

```
import pandas as pd

# Load the housing dataset (replace 'dataset.csv' with our dataset file)
housing_data = pd.read_csv('dataset.csv')

# Filter the data based on our criteria
selected_data = housing_data[(housing_data['bedrooms'] == 2) &
                             (housing_data['bathrooms'] == 2.5) &
                             (housing_data['sqft_living'] == 600) &
                             (housing_data['sqft_lot'] == 600) &
                             (housing_data['floors'] == 2) &
                             (housing_data['zipcode'] == 98008)]

# Now 'selected_data' contains the rows that match our criteria
```

A code to predict the price of a new house having the data frame {'bedrooms':[2],'bathrooms':[2.5],'sqft_living':[600],'sqft_lot':[600],'floors': [2],'zipcode': [98008]}). We pass this data frame to the model's prediction function to obtain the predicted price.

Assuming we have a linear regression model, we can use it to predict the house price for the given data frame({'bedrooms': [2], 'bathrooms': [2.5], 'sqft_living': [600], 'sqft_lot': [600], 'floors': [2], 'zipcode': [98008]}):

```
import pandas as pd
from sklearn.linear_model import LinearRegression

# Create a DataFrame with the new house's features
new_house_data = pd.DataFrame({'bedrooms': [2], 'bathrooms': [2.5], 'sqft_living': [600], 'sqft_lot': [600], 'floors': [2], 'zipcode': [98008]})

# Load your trained model (you should replace this with your actual model)
# For demonstration purposes, we'll create a simple linear regression model
model = LinearRegression()

# Load your pre-trained model here
# model = load_your_model()

# Predict the price for the new house
predicted_price = model.predict(new_house_data)

# Print the predicted price
print("Predicted Price:", predicted_price)
```

Conclusion

In conclusion, using machine learning in Python is a powerful tool for predicting house prices. By gathering and cleaning data, visualizing patterns, and training and evaluating our models, we can make informed decisions in the dynamic world of real estate.

By leveraging advanced algorithms and data analysis, we can make accurate predictions and inform decision-making processes. This approach empowers buyers, sellers, and investors to make informed choices in a dynamic and competitive market, ultimately maximizing their opportunities and outcomes.