PREDICTING HOUSE PRICE USING MACHINE LEARNING

TEAM MEMBER

810621104007: BHAVANI.S

Phase-4 submission document

Project Title: House Price Prediction

Phase 4: Development Part 2

Topic: Continue building the house price prediction model by feature engineering, model training, and evaluation.

HOUSE PRICE PREDICTION

INTRODUCTION:

House price prediction is a critical application of machine learning and data analysis that aims to estimate the market value of residential properties. It is a fundamental task in the real estate industry and has significant implications for homebuyers, sellers, investors, and policymakers. Understanding and accurately predicting house prices can help individuals make informed decisions about buying or selling homes, assess the value of their real estate investments, and provide valuable insights into local housing market trends.

Key components of house price prediction include:

- **Data Collection:**

Gathering relevant data is the first step in house price prediction. This data may include information about the properties (e.g., size, location, features), local market conditions, economic indicators, and historical sales data. Data can be collected from various sources, such as public records, real estate listings, and government databases.

- **Feature Engineering:**

Feature engineering involves selecting and transforming the most relevant attributes or features that influence house prices. This may include factors like the number of bedrooms and bathrooms, square footage, neighborhood characteristics, and property conditions.

- **Data Preprocessing:**

Data preprocessing is essential to clean and prepare the dataset for analysis. This includes handling missing values, removing outliers, and encoding categorical variables. Data normalization or scaling may also be necessary to ensure that all features are on a similar scale.

- **Model Selection:**

Various machine learning and statistical models can be used for house price prediction. Commonly employed models include linear regression, decision trees, random forests, support vector machines, and neural networks. The choice of model depends on the complexity of the problem and the characteristics of the dataset.

- **Model Training:**

The selected model is trained on a portion of the dataset, typically using historical house price data. During training, the model learns the relationships between the input features and the target variable (house prices).

- **Model Evaluation:**

The performance of the model is assessed using various metrics, such as mean absolute error (MAE), mean squared error (MSE), or root mean squared error (RMSE). Cross-validation techniques may also be used to ensure the model's general.

- **Hyperparameter Tuning:**

Models often have hyperparameters that
treated to be optimized for better performance.
Techniques like grid search or random search can help find the best hyperparameter values.

- **Deployment and Prediction:**

Once the model is trained and validated, it can be deployed to make predictions on new, unseen data. These predictions can provide estimates of house prices for properties that are not in the training dataset.

- **Continuous Monitoring and Updates:**

Real estate markets can change over time due to economic conditions, changes in demand, and other need regular factors. Therefore, house price prediction models may updates to maintain their accuracy.

	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
79545.46	5.682861	7.009188	4.09	23086.8	1059034	208 Michael Ferry Apt. 674 Laurabury, NE 37010-5101
79248.64	6.0029	6.730821	3.09	40173.07	1505891	188 Johnson Views Suite 079 Lake Kathleen, CA 48958
61287.07	5.86589	8.512727	5.13	36882.16	1058988	9127 Elizabeth Stravenue Danieltown, WI 06482-3489
63345.24	7.188236	5.586729	3.26	34310.24	1260617	USS Barnett FPO AP 44820
59982.2	5.040555	7.839388	4.23	26354.11	630943.5	USNS Raymond FPO AE 09386
80175.75	4.988408	6.104512	4.04	26748.43	1068138	06039 Jennifer Islands Apt. 443 Tracyport, KS 16077
64698.46	6.025336	8.14776	3.41	60828.25	1502056	4759 Daniel Shoals Suite 442 Nguyenburgh, CO 20247
78394.34	6.98978	6.620478	2.42	36516.36	1573937	972 Joyce Viaduct Lake William, TN 17778-6483
59927.66	5.362126	6.393121	2.3	29387.4	798869.5	USS Gilbert FPO AA 20957

OVERVIEW OF THE PROCESS:

The following is an overview of the process of building a house price prediction model by features selection, model training, and evaluation.

1. **Prepare the data:** This includes cleaning the data, removing outliers , and handling missing values.

2. **Perform feature selection:** This can be done using a variety of methods, such as correlation analysis, information gain, and recursive feature elimination.
3. **Train the model:** There are many different machine learning algorithms that can be used for house price prediction. Some popular choices includes linear regression, random forests, and gradient boosting machines.
4. **Evaluate the model:** This can be done by calculating the mean squared error(MSE) or the root mean squared error(RMSE) of the model's predictions on the held-out test set.
5. **Deploy the model:** Once the model has been evaluated and found to be performing well, it can be deployed to prediction so that it can be used to predict the house prices of news houses.

PROCEDURE:

Feature selection:

Feature selection is an important step in building a machine learning model, as it helps improve model performance, reduce overfitting, and enhance interpretability. Here's a general procedure for feature selection:

1. **Define the Goal:**

   - Start by defining your goal or objective. What are you trying to achieve with feature selection? Are you looking to improve model performance, reduce model complexity, or gain better insights into the data?
   -

2. **Data Collection and Preprocessing:**

   - Gather and preprocess your data. This includes handling missing values, encoding categorical variables, and standardizing or normalizing numerical features.

3. **Explore the Data:**

- Perform exploratory data analysis (EDA) to gain insights into your dataset. Visualizations, summary statistics, and correlation analyses can help identify potentially important features.
-

4. **Select a Feature Selection Method:**

- Choose one or more feature selection methods based on your problem and dataset. Common methods include correlation analysis, feature importance from models, mutual information, recursive feature elimination (RFE), and L1 regularization (Lasso).

5. **Correlation Analysis:**

- Calculate the correlation between each feature and the target variable. Select the features with the highest correlations.

6. **Feature Importance from Models:**

- Train a model on your data and use feature importance scores (e.g., feature_importances_ attribute in scikit-learn for decision trees or random forests). Select the most important features.

7. **Mutual Information:**

- Calculate mutual information between features and the target variable to measure the information provided by each feature. Select features with high mutual information values.

8. **Recursive Feature Elimination (RFE):**

- Start with all features, train a model, and iteratively remove the least important feature. Stop when model performance stabilizes.

9. **L1 Regularization (Lasso):**

- Use L1 regularization with linear models to encourage sparsity in the feature set. The model will automatically select the most important features by shrinking less important feature coefficients to zero.

10. **Cross-Validation:**

- Employ cross-validation techniques to assess how well different sets of features perform. This helps ensure that the feature selection process generalizes to new data.

11. **Evaluate Model Performance:**

- After each round of feature selection, evaluate the model's performance using appropriate metrics (e.g., mean squared error for regression tasks or accuracy for classification tasks).

12. **Iterate and Refine:**

- Feature selection is often an iterative process. Revisit and adjust your feature selection method or criteria as needed based on the results and performance of your model.

13. **Documentation:**

- Keep a record of the selected features and the reasoning behind their selection. This documentation is essential for reproducibility and model interpretation.

14. **Test on New Data:**

- Finally, evaluate your model's performance on new, unseen data to ensure that the selected features generalize well.

FEATURES SELECTION:

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

# Load your dataset (replace 'data.csv' with your data file)
data = pd.read_csv('data.csv')

# Define your target variable (house prices) and features
X = data.drop('HousePrice', axis=1)  # Features
y = data['HousePrice']  # Target variable

# 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)
```

```
# Standardize the features (important for Lasso)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train a Lasso regression model
alpha = 0.01  # Adjust the alpha (regularization strength) as needed
lasso = Lasso(alpha=alpha)
lasso.fit(X_train, y_train)

# Get feature importance scores from the Lasso model
feature_importance = np.abs(lasso.coef_)

# Select features with non-zero coefficients
selected_features = X.columns[feature_importance > 0]

# Reconstruct the feature matrix with selected features
```

```
X_train_selected = X_train[:, feature_importance >_0]

X_test_selected = X_test[:, feature_importance > 0]


# Train a machine learning model using the selected features (e.g., regression or other models)

# Replace 'YourModel' with the specific model you want to use (e.g., RandomForestRegressor, GradientBoostingRegressor, etc.)

from sklearn.ensemble import YourModel


model = YourModel()

model.fit(X_train_selected, y_train)


# Make predictions on the test set

y_pred = model.predict(X_test_selected)


# Evaluate the model's performance

mse = mean_squared_error(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
```

```
# Output the selected features
print('Selected Features:', selected_features)
```

MODEL TRAINING:

```
# 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, r2_score

# Load your dataset (replace 'data.csv' with your data file)
data = pd.read_csv('data.csv')

# Define your target variable (house prices) and features
X = data.drop('HousePrice', axis=1)  # Features
y = data['HousePrice']  # Target variable

# 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)

# Train a Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

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

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

# Output the model's performance metrics
print(f'Mean Squared Error: {mse}')
print(f'R-squared (R2) Score: {r2}')

# Sample output:
```

Mean Squared Error: 1234567.89

R-squared (R2) Score: 0.75

LINEAR REGRESSION:

```
# 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, r2_score

# Sample data (replace this with your actual dataset)
data = {
    'Size': [1400, 1600, 1700, 1875, 1100, 1550, 2350, 2450, 1425, 1700],
    'Bedrooms': [3, 3, 2, 3, 2, 2, 4, 4, 3, 3],
    'Bathrooms': [2, 2, 2, 2, 1, 2, 3, 3, 2, 2],
    'Price': [245000, 312000, 279000, 308000, 199000, 219000, 405000, 324000, 319000, 255000]
}
```

```
# Create a DataFrame
df = pd.DataFrame(data)

# Define your features (X) and target variable (y)
X = df[['Size', 'Bedrooms', 'Bathrooms']]
y = df['Price']

# 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)

# Train a Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

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

```
# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Output the model's performance metrics
print("Linear Regression Model Performance:")
print(f'Mean Squared Error (MSE): {mse:.2f}')
print(f'R-squared (R2) Score: {r2:.2f}')

# Sample output:
# Linear Regression Model Performance:
# Mean Squared Error (MSE): 66432875.50
# R-squared (R2) Score: 0.73
```

RIDGE REGRESSION:

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

# Sample data (replace this with your actual dataset)
data = {
    'Size': [1400, 1600, 1700, 1875, 1100, 1550, 2350, 2450, 1425, 1700],
    'Bedrooms': [3, 3, 2, 3, 2, 2, 4, 4, 3, 3],
    'Bathrooms': [2, 2, 2, 2, 1, 2, 3, 3, 2, 2],
    'Price': [245000, 312000, 279000, 308000, 199000, 219000, 405000, 324000, 319000, 255000]
}
```

```
# Create a DataFrame
df = pd.DataFrame(data)

# Define your features (X) and target variable (y)
X = df[['Size', 'Bedrooms', 'Bathrooms']]
y = df['Price']

# 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)

# Train a Ridge Regression model
alpha = 1.0  # You can adjust the regularization strength
ridge = Ridge(alpha=alpha)
ridge.fit(X_train, y_train)

# Make predictions on the test set
y_pred = ridge.predict(X_test)
```

```
# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Output the model's performance metrics
print("Ridge Regression Model Performance:")
print(f'Mean Squared Error (MSE): {mse:.2f}')
print(f'R-squared (R2) Score: {r2:.2f}')

# Sample output:
# Ridge Regression Model Performance:
# Mean Squared Error (MSE): 10741863.78
# R-squared (R2) Score: 0.79
```

RANDOM FOREST REGRESSION, XGBoost REGRESSION, POLYNOMIAL REGRESSION: # Import necessary libraries

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Sample data (replace this with your actual dataset)
data = {
    'Size': [1400, 1600, 1700, 1875, 1100, 1550, 2350, 2450, 1425, 1700],
    'Bedrooms': [3, 3, 2, 3, 2, 2, 4, 4, 3, 3],
    'Bathrooms': [2, 2, 2, 2, 1, 2, 3, 3, 2, 2],
    'Price': [245000, 312000, 279000, 308000, 199000, 219000, 405000, 324000, 319000, 255000]
```

```
}

# Create a DataFrame
df = pd.DataFrame(data)

# Define your features (X) and target variable (y)
X = df[['Size', 'Bedrooms', 'Bathrooms']]
y = df['Price']

# 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)

# Train a Random Forest Regression model
rf_model = RandomForestRegressor(n_estimators=100,
random_state=42)
rf_model.fit(X_train, y_train)

# Train an XGBoost Regression model
```

```
xgb_model = XGBRegressor(n_estimators=100,
learning_rate=0.1, max_depth=3, random_state=42)
xgb_model.fit(X_train, y_train)

# Train a Polynomial Regression model (degree=2)
poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
poly_model = LinearRegression()
poly_model.fit(X_train_poly, y_train)

# Make predictions on the test set for each model
y_pred_rf = rf_model.predict(X_test)
y_pred_xgb = xgb_model.predict(X_test)
X_test_poly = poly.transform(X_test)
y_pred_poly = poly_model.predict(X_test_poly)

# Evaluate the models' performance
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)
```

```
mse_xgb = mean_squared_error(y_test, y_pred_xgb)
r2_xgb = r2_score(y_test, y_pred_xgb)
mse_poly = mean_squared_error(y_test, y_pred_poly)
r2_poly = r2_score(y_test, y_pred_poly)

# Output the models' performance metrics
print("Random Forest Regression Model Performance:")
print(f'Mean Squared Error (MSE): {mse_rf:.2f}')
print(f'R-squared (R2) Score: {r2_rf:.2f}')
print("\nXGBoost Regression Model Performance:")
print(f'Mean Squared Error (MSE): {mse_xgb:.2f}')
print(f'R-squared (R2) Score: {r2_xgb:.2f}')
print("\nPolynomial Regression Model Performance:")
print(f'Mean Squared Error (MSE): {mse_poly:.2f}')
print(f'R-squared (R2) Score: {r2_poly:.2f}')

# Sample output:
# Random Forest Regression Model Performance:
# Mean Squared Error (MSE): 3274859.90
```

R-squared (R2) Score: 0.89

#

XGBoost Regression Model Performance:

Mean Squared Error (MSE): 3505483.57

R-squared (R2) Score: 0.88

#

Polynomial Regression Model Performance:

Mean Squared Error (MSE): 5217539.49

R-squared (R2) Score: 0.82

FEATURE ENGINEERING:

Feature engineering is a crucial step in house price prediction to create new features or transform existing ones, making them more informative for the machine learning model. Effective feature engineering can significantly improve the model's predictive accuracy. Here are some feature engineering techniques you can use for house price prediction:

1. **Area-Related Features:**

   - Combine or extract new features from existing area-related features like the total square footage of the house, the size of the living area, or the size of the lot. For example, create a feature for the ratio of the living area to the total area.

2. **Age and Renovation:**

   - Create features that represent the age of the property. You can also add a binary feature to indicate whether the property has been recently renovated.

3. **Categorical Features:**

- Convert categorical variables like "neighborhood" or "property type" into one-hot encoding or label encoding to make them suitable for machine learning models.

4. **Proximity Features:**

- Create features that represent the proximity of the property to important locations, such as schools, hospitals, parks, public transportation, and city centers.

5. **Price Per Square Foot:**

- Calculate the price per square foot, which can provide insights into the relative value of different properties.

6. **Composite Features:**

- Combine multiple features to create new ones. For example, you can create a "quality score" by combining ratings for different property features like the kitchen, bathrooms, and exterior.

7. **Historical Trends:**

- Include features that capture historical data, such as the average house price in the neighborhood over the past few years or the inflation-adjusted price changes.
-

8. **Seasonal and Temporal Features:**

- Account for seasonal variations by adding features that indicate the time of year the property is listed or sold. Temporal features like day of the week or month can also be useful.

9. **Exterior Features**:

- Create features that reflect the quality or condition of the property's exterior, landscaping, or curb appeal.

10. **Interaction Terms:**

- Add interaction terms between features. For example, you can multiply the number of bedrooms and bathrooms to create a "bed-bath" interaction feature.

11. **Engineering Binary Flags:**

- Create binary flags to represent specific property characteristics, such as whether the property has a

swimming pool, a fireplace, a garage, or energy-efficient appliances.

12. **Logarithmic Transformation:**

- Apply logarithmic transformations to features that have a skewed distribution, such as the price, to make the data more normally distributed.

13. **Density Features:**

- Calculate population density or the density of certain amenities in the neighborhood, which can be informative.

14. **Composite Metrics:**

- Calculate composite metrics like walk ability scores or school quality indexes by combining relevant neighborhood attributes.

15. **Economic Indicators:**

- Incorporate economic indicators like local job market data, interest rates, or inflation rates that can impact house .