PREDICTING HOUSE PRICE USING MACHINE LEARNING

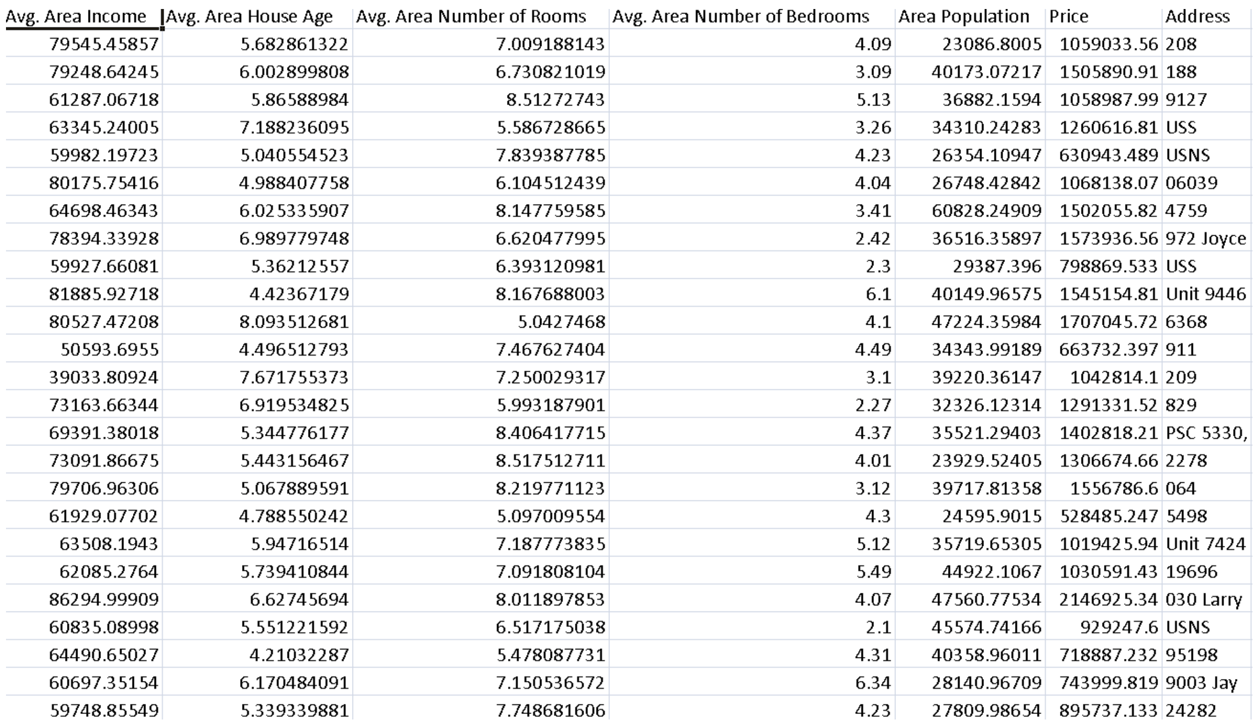
Batch member
510521205049 : S.Tharun

Phase 1 submission document

Project: House Price Prediction
Phase 1: Data Preprocessing and Feature Selection.

1.Data Source

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



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

2.Data Preprocessing

- The first step in any machine learning project is to preprocess the data. This involves cleaning the data, removing outliers, and handling missing values.
- To preprocess the house price data, we would first need to identify and remove any duplicate records. We could do this by using a unique identifier for each house, such as the property ID or the address.
- Next, we would need to check for and correct any errors in the data. For example, we might need to correct any typos in the addresses or zip codes. We might also need to correct any errors in the data types, such as ensuring that all of the numerical features are stored as numerics.
- Once the data has been cleaned, we need to handle any missing values. We could do this by dropping the records with missing values, or by imputing the missing values with the mean or median value of the feature. However, it is important to note that dropping records can reduce the size of the dataset, which can impact the performance of the machine learning model.

Now, we categorize the features depending on their datatype (int, float, object) and then calculate the number of them.

PYTHON 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:

Once the data has been preprocessed, we need to engineer the features. This involves transforming the existing features into new features that are more informative for the machine learning model.

For the house price data, we might want to engineer features such as:

- The age of the house
- The square footage of the house
- The number of bedrooms and bathrooms
- The quality of the school district
- The crime rate in the neighborhood
- The distance to amenities such as shops and restaurants
- The type of property (e.g., single-family home, townhouse, condo)

We could also engineer features such as the price per square foot and the price per bedroom. These features can be more informative than the raw features, as they take into account the size of the house.

4.Model Selection:

Random Forest Regressor for predicting house prices. Random Forest Regressors are ensemble learning algorithms that combine the predictions of multiple decision trees to produce a more accurate prediction. They are well-suited for regression tasks because they are able to learn complex relationships between the input features and the target variable.

Advantage:

- They are able to learn complex relationships between the input features and the target variable, even if the relationships are non-linear.
- They are relatively robust to outliers and noise in the data.
- They are easy to train and tune.

Disadvantage:

- They can be computationally expensive to train on large datasets.
- They can be difficult to interpret, as it can be difficult to understand which features are most important to the model's predictions.

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:

To train a selected model using preprocessed data for house price prediction using machine learning, you can follow these steps:

1.Import the required libraries. This will include libraries for machine learning, data processing, and visualization.

2.Load the preprocessed data. This will typically involve loading the data from a CSV file or database.

3.Split the data into training and test sets. The training set will be used to train the model, and the test set will be used to evaluate the performance of the trained model.

4.Initialize the selected model. This will involve setting the hyperparameters of the model.

5.Train the model. This will involve feeding the training data to the model and allowing the model to learn the relationships between the input features and the target variable.

6.Evaluate the trained model on the test set. This will involve feeding the test data to the model and calculating the model's performance metrics, such as the mean squared error or R-squared score.

6.Evaluation:

There are a number of metrics that can be used to evaluate the performance of a machine learning model for house price prediction. Some of the most common metrics include:

- Mean Absolute Error (MAE): The MAE is the average of the absolute differences between the predicted prices and the actual prices. It is a good measure of the overall accuracy of the model.
- Root Mean Squared Error (RMSE): The RMSE is the square root of the average squared differences between the predicted prices and the actual prices. It is a good measure of the magnitude of the errors.
- R-squared: The R-squared score is a measure of how well the model explains the variation in the target variable. It ranges from 0 to 1, with higher values indicating a better fit.

Conclusion

The Phase 1 submission for the house price prediction project should include a detailed description of the data preprocessing and feature engineering steps that were taken. It should also include a preliminary analysis of the data. The Phase 1 submission should not include any trained machine learning models.

This is because the data preprocessing and feature engineering steps can have a significant impact on the performance of the machine learning model. It is important to finalize these steps before training and evaluating the model.