**Problem Statement**

A real estate company wishes to analyze the prices of properties based on various factors such as area, number of rooms, bathrooms, bedrooms, etc. Create a multiple linear regression model which is capable of predicting the sale price of houses based on multiple factors and evaluate the accuracy of this model.

**List of Activities**

**Activity 1:** Analyzing the Dataset

**Activity 2:** Data Preparation

**Activity 3:** Train-Test Split

**Activity 4:** Model Training

**Activity 5:** Model Prediction and Evaluation

#### Activity 1: Analyzing the Dataset

* Create a Pandas DataFrame for **Housing** dataset using the below link. This dataset consists of following columns:

| **Field** | **Description** |
| --- | --- |
| price | Sale price of a house in INR |
| area | Total size of a property in square feet |
| bedrooms | Number of bedrooms |
| bathrooms | Number of bathrooms |
| Store  ys | Number of storeys excluding basement |
| mainroad | yes, if the house faces a main road |
| livingroom | yes, if the house has a separate living room or a drawing room for guests |
| basement | yes, if the house has a basement |
| hotwaterheating | yes, if the house uses gas for hot water heating |
| airconditioning | yes, if there is central air conditioning |
| parking | number of cars that can be parked |
| prefarea | yes, if the house is located in the preferred neighborhood of the city |

**Dataset Link:** house-prices.csv

* Print the first five rows of the dataset. Check for null values and treat them accordingly.

**# Import modules**

import pandas as pd  
import numpy as np  
from sklearn .model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score,mean\_absolute\_error

**# Load the dataset**

df=pd.read\_csv('https://raw.githubusercontent.com/mariya-sherin/DSML/main/house-prices%20-%20house-prices%20-%20house-prices%20-%20house-prices.csv')

**# Print first five rows using head() function**

print(df.head())

**Output:**

C:\Users\USER\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\USER\PycharmProjects\pythonProject\MultipleLinearRegression.py

price area bedrooms ... parking prefarea furnishingstatus

0 13300000 7420 4 ... 2 yes furnished

1 12250000 8960 4 ... 3 no furnished

2 12250000 9960 3 ... 2 yes semi-furnished

3 12215000 7500 4 ... 3 yes furnished

4 11410000 7420 4 ... 2 no furnished

[5 rows x 13 columns]

**# Check if there are any null values. If any column has null values, treat them accordingly**

print(df.isnull().sum())

**Output:**

price 0

area 0

bedrooms 0

bathrooms 0

stories 0

mainroad 0

guestroom 0

basement 0

hotwaterheating 0

airconditioning 0

parking 0

prefarea 0

furnishingstatus 0

dtype: int64

#### Activity 2: Data Preparation

This dataset contains many columns having categorical data i.e. values 'Yes' or 'No'. However, for linear regression, we need numerical data. So you need to convert all 'Yes' and 'No' values to 1s and 0s, where

* 1 means 'Yes'
* 0 means 'No'

Similarly, replace

* unfurnished with 0
* semi-furnished with 1
* furnished with 2

**Hint:** To replace all 'Yes' values with 1 and 'No' values with 0, use replace() function of the DataFrame object.

For ex: df.replace(to\_replace="yes", value=1, inplace=True) ⇒ replaces the "yes" values in all columns with 1. If you need to make changes inplace, use inplace boolean argument.

**# Replace all non-numeric values with numeric values.**

df.replace({'yes': 1, 'no': 0},inplace=True)  
print(df.head())  
df.replace({'unfurnished':0,'semi-furnished':1,'furnished':2},inplace=True)  
print(df.head())

**Output:**

price area bedrooms ... parking prefarea furnishingstatus

0 13300000 7420 4 ... 2 1 furnished

1 12250000 8960 4 ... 3 0 furnished

2 12250000 9960 3 ... 2 1 semi-furnished

3 12215000 7500 4 ... 3 1 furnished

4 11410000 7420 4 ... 2 0 furnished

[5 rows x 13 columns]

price area bedrooms ... parking prefarea furnishingstatus

0 13300000 7420 4 ... 2 1 2

1 12250000 8960 4 ... 3 0 2

2 12250000 9960 3 ... 2 1 1

3 12215000 7500 4 ... 3 1 2

4 11410000 7420 4 ... 2 0 2

[5 rows x 13 columns]

#### Activity 3: Train-Test Split

You need to predict the house prices based on several factors. Thus, price is the target variable and other columns except price will be feature variables.

Split the dataset into training set and test set such that the training set contains 67% of the instances and the remaining instances will become the test set.

**# Split the DataFrame into the training and test sets.**

X=df.drop('price',axis=1)  
y=df['price']  
X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.33)

#### Activity 4: Model Training

Implement multiple linear regression using sklearn module in the following way:

1. Reshape the target variable array into two-dimensional arrays by using reshape(-1, 1) function of the numpy module.
2. Deploy the model by importing the LinearRegression class and create an object of this class.
3. Call the fit() function on the LinearRegression object.

**# Create two-dimensional NumPy arrays for the target variable**

y\_train\_reshaped=y\_train.values.reshape(-1,1)  
y\_test\_reshaped=y\_test.values.reshape(-1,1)  
X\_train\_reshaped=X\_train.values.reshape(-1,1)  
X\_test\_reshaped=X\_test.values.reshape(-1,1)

**# Build linear regression model**

sklearn\_lin\_reg=LinearRegression()  
sklearn\_lin\_reg.fit(X\_train,y\_train)

**# Print the value of the intercept**

print(model.intercept\_)

**Output:**

[-552333.87235215]

**# Print the names of the features along with the values of their corresponding coefficients.**

print(list(zip(X.columns,model.coef\_)))

**Output:**

[('area', np.float64(295.36240293000276)), ('bedrooms', np.float64(119265.05589863146)), ('bathrooms', np.float64(1090058.5463947817)), ('stories', np.float64(445653.3870109009)), ('mainroad', np.float64(413202.3034623168)), ('guestroom', np.float64(177910.90295019385)), ('basement', np.float64(386436.9910702257)), ('hotwaterheating', np.float64(709497.6817044108)), ('airconditioning', np.float64(726725.6712548047)), ('parking', np.float64(233834.78754633997)), ('prefarea', np.float64(600572.8229410385)), ('furnishingstatus', np.float64(187558.59412958377))]

#### Activity 5: Model Prediction and Evaluation

Predict the values for both training and test sets by calling the predict() function on the LinearRegression object. Also, calculate the R2, MSE, RMSE and MAE values to evaluate the accuracy of your model.

y\_train\_pred=sklearn\_lin\_reg.predict(X\_train)  
y\_test\_pred=sklearn\_lin\_reg.predict(X\_test)  
  
print("r2\_score of test:",r2\_score(y\_test\_reshaped,y\_test\_pred))  
print("r2\_score of train:",r2\_score(y\_train\_reshaped,y\_train\_pred))  
print("mean\_squared\_error of test:",mean\_squared\_error(y\_test\_reshaped,y\_test\_pred))  
print("mean\_squared\_error of train:",mean\_squared\_error(y\_train\_reshaped,y\_train\_pred))  
print("mean\_absolute\_error of test:",np.sqrt(mean\_squared\_error(y\_test\_reshaped,y\_test\_pred)))  
print("mean\_absolute\_error of train:",np.sqrt(mean\_squared\_error(y\_train\_reshaped,y\_train\_pred)))

**Output:**

r2\_score of test: 0.6651073106746996

r2\_score of train: 0.6818402341669547

mean\_squared\_error of test: 1147357675837.3013

mean\_squared\_error of train: 1121359833777.8613

mean\_absolute\_error of test: 1071147.8309912696

mean\_absolute\_error of train: 1058942.7906066792

Process finished with exit code 0