Project Summary

This project is about analysing and predicting house prices by a linear regression model

Project details:

Author: K.Santhosh

Date:6/3/2025

Version:1

problem statement:Build a machine learning model to predict house prices.

importing datasets

```
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 r2_score,mean_absolute_error
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import GridSearchCV

df =pd.read_csv('/content/Housing.csv')
```

Exploring Dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   price             545 non-null    int64 
 1   area              545 non-null    int64 
 2   bedrooms          545 non-null    int64 
 3   bathrooms         545 non-null    int64 
 4   stories           545 non-null    int64 
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64 
```

```
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

```
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 545,\n  \"fields\": [\n    {\n      \"column\": \"price\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 1870439,\n        \"min\": 1750000,\n        \"max\": 13300000,\n        \"num_unique_values\": 219,\n        \"samples\": [\n          3773000,\n          5285000,\n          1820000\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"area\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 2170,\n        \"min\": 1650,\n        \"max\": 16200,\n        \"num_unique_values\": 284,\n        \"samples\": [\n          6000,\n          2684,\n          5360\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"bedrooms\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 1,\n        \"max\": 6,\n        \"num_unique_values\": 6,\n        \"samples\": [\n          4,\n          3,\n          1\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"bathrooms\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 1,\n        \"max\": 4,\n        \"num_unique_values\": 4,\n        \"samples\": [\n          4,\n          3,\n          2\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"stories\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 1,\n        \"max\": 4,\n        \"num_unique_values\": 4,\n        \"samples\": [\n          4,\n          1,\n          3\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"mainroad\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n        \"samples\": [\n          \"no\",\n          \"yes\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"guestroom\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n        \"samples\": [\n          \"yes\",\n          \"no\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"basement\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n        \"samples\": [\n          \"yes\",\n          \"no\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"hotwaterheating\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
```

```
\"num_unique_values\": 2,\n        \"samples\": [\n          \"yes\",\
n          \"no\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"airconditioning\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"no\",\n          \"yes\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"parking\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 0,\n
\"min\": 0,\n        \"max\": 3,\n        \"num_unique_values\": 4,\n
\"samples\": [\n          3,\n          1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"prefarea\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
2,\n        \"samples\": [\n          \"no\",\n          \"yes\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"furnishingstatus\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"furnished\",\n          \"semi-furnished\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```


Encoding categorical variables

```
categorical_cols = ['mainroad', 'guestroom', 'basement',
'hotwaterheating',
                    'airconditioning', 'prefarea', 'furnishingstatus']

df_encoded = df.copy()
label_encoders = {}

for col in categorical_cols:
    le = LabelEncoder()
    df_encoded[col] = le.fit_transform(df_encoded[col])
    label_encoders[col] = le  # Store encoders for future use
missing_values = df_encoded.isnull().sum()
df_encoded.head(), missing_values
```

```
(      price  area  bedrooms  bathrooms  ...  airconditioning  parking
prefarea  furnishingstatus
 0  13300000  7420         4          2  ...                1        2
1                 0
 1  12250000  8960         4          4  ...                1        3
0                 0
 2  12250000  9960         3          2  ...                0        2
1                 1
 3  12215000  7500         4          2  ...                1        3
```

```
1                  0
 4  11410000  7420         4          1  ...                 1        2
0                  0

 [5 rows x 13 columns],
 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)
```

splitting the dataset

```
X = df_encoded.drop(columns=['price'])
y = df_encoded['price']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

X_train.shape, X_test.shape
```

```
((436, 12), (109, 12))
```

Training linear regression model

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

```
LinearRegression()
```

```
y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

mae, r2
```

```
(979679.6912959901, 0.6494754192267803)
```

Hyperparameter tuning

```
param_grid = {
    'fit_intercept': [True, False]
}

grid_search = GridSearchCV(LinearRegression(), param_grid, cv=5,
scoring='r2', n_jobs=-1)
grid_search.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, estimator=LinearRegression(), n_jobs=-1,
             param_grid={'fit_intercept': [True, False]},
scoring='r2')
```

```
best_model = grid_search.best_estimator_
best_params = grid_search.best_params_
best_r2 = grid_search.best_score_

best_params, best_r2
```

```
({'fit_intercept': True}, 0.6469452898841016)
```