Predicting House Prices with Linear Regression

The objective of this project is to build a predictive model using linear regression to estimate a numerical outcome based on a dataset with relevant features. Linear regression is a fundamental machine learning algorithm, and this project provides hands-on experience in developing, evaluating, and interpreting a predictive model.

Import Library

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt
```

+ Code + Text

```
import seaborn as sns
```

Import Dataset

```
hp=pd.read_csv('/Housing.csv')
```

Displaying first 10 rows

```
hp.head(10)
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotw
0	13300000	7420	4	2	3	yes	no	no	
1	12250000	8960	4	4	4	yes	no	no	
2	12250000	9960	3	2	2	yes	no	yes	
3	12215000	7500	4	2	2	yes	no	yes	
4	11410000	7420	4	1	2	yes	yes	yes	
5	10850000	7500	3	3	1	yes	no	yes	
6	10150000	8580	4	3	4	yes	no	no	
7	10150000	16200	5	3	2	yes	no	no	
8	9870000	8100	4	1	2	yes	yes	yes	
9	9800000	5750	3	2	4	yes	yes	no	

Next steps: View recommended plots

Data Preparation

Checking for any missing values

```
hp.isnull().sum()
```

```
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
```

Now handling the missing values for the numeric columns only

Now selecting numeric columns

```
numeric_hp=hp.select_dtypes(include=['number'])
```

Filling missing values in numeric columns

```
hp[numeric_hp.columns]=numeric_hp.fillna(numeric_hp.median())
```

Ensuring correct data types

```
hp.dtypes
```

```
price              int64
area               int64
bedrooms           int64
bathrooms          int64
stories            int64
mainroad          object
guestroom         object
basement          object
hotwaterheating   object
airconditioning   object
parking            int64
prefarea          object
furnishingstatus  object
dtype: object
```

Feature Selection

Correlation Matrix to see the relationship betweeen features and the target variable 'price'

convert relevant columns to numeric type before calculating correlation

converting to numeric , replace non-numeric with Not a Number

```
hp_numeric=hp.apply(pd.to_numeric, errors='coerce')
```

Fill Not a Number values with median

```
hp_numeric.fillna(hp_numeric.median(), inplace=True)
```

```
corr_matrix=hp_numeric.corr()
```

```
corr_matrix['price'].sort_values(ascending=False)
```

```
price             1.000000
area              0.535997
bathrooms         0.517545
stories           0.420712
parking           0.384394
bedrooms          0.366494
mainroad               NaN
guestroom              NaN
basement               NaN
hotwaterheating        NaN
airconditioning        NaN
prefarea               NaN
furnishingstatus       NaN
Name: price, dtype: float64
```

Now selecting features based on correlation

```
features=['bathrooms','bedrooms','area','stories','mainroad','guestroom','basement','hotwaterheating','airconditioning','parking','prefa
```

```
x=hp[features]
y=hp['price']
```

Now converting categorial variables to dummy variables

```
X=pd.get_dummies(x,drop_first=True)
```

MODEL TRAINING

Splitting data into training and testing

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LinearRegression
```

```
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

Training the model

```
model=LinearRegression()
```

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

```
▾ LinearRegression
LinearRegression()
```

Model Evaluate

```
from sklearn.metrics import mean_squared_error,r2_score
```

Making Predictions

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

Now calculating the Mean Squared Error and R-squared

```
mse= mean_squared_error(y_test,y_pred)
```

```
r2=r2_score(y_test,y_pred)
```

```
f"Mean Squared Error: {mse}"
```

```
'Mean Squared Error: 1754318687330.7283'
```

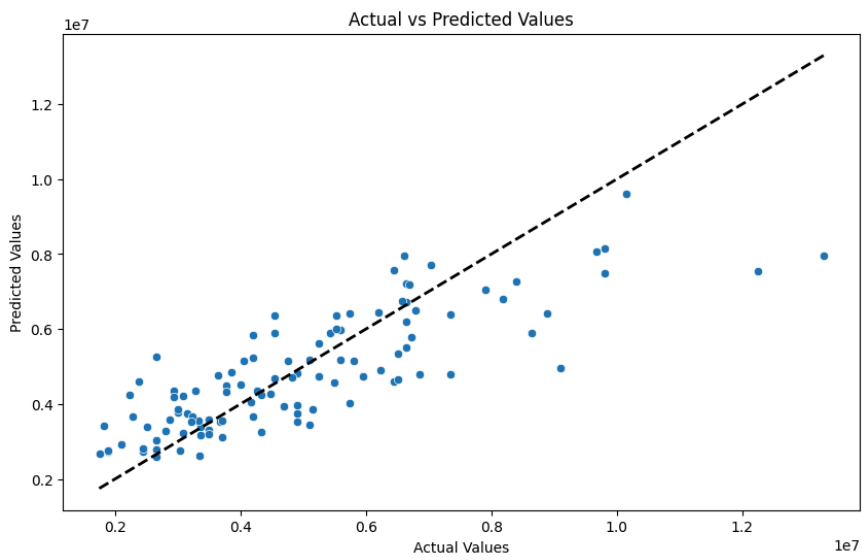
```
f"R-squared: {r2}"
```

```
'R-squared: 0.6529242642153057'
```

Now Visualization

Scatter plot of predicted values and actual values

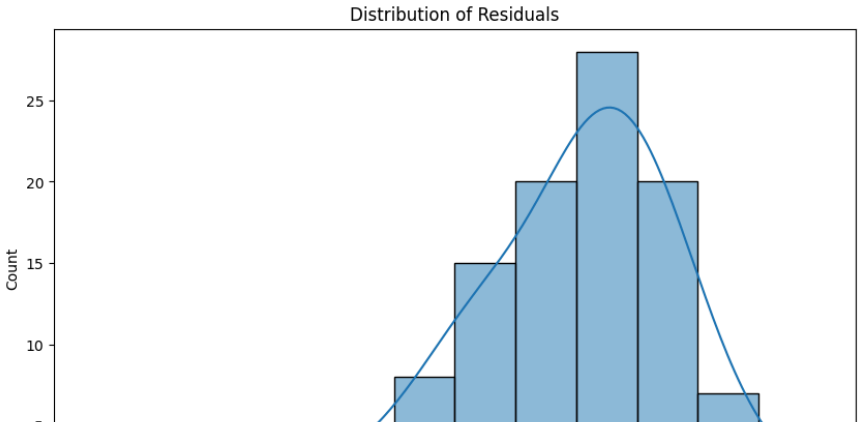
```
plt.figure(figsize=(10,6))
sns.scatterplot(x=y_test,y=y_pred)
plt.plot([min(y_test),max(y_test)],[min(y_test),max(y_test)],'k--',lw=2)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values')
plt.show()
```



Distribution Plot of residuals

```
residuals=y_pred-y_test
```

```
plt.figure(figsize=(10,6))
sns.histplot(residuals,kde=True)
plt.xlabel('Residuals')
plt.ylabel('Count')
plt.title('Distribution of Residuals')
plt.show()
```


Distribution of Residuals
Count
25
20
15
10