Sardar Patel Institute of Technology, Mumbai
Department of Electronics and Telecommunication Engineering
T.E.-CSE (2023-2024)
PC 307 - Machine Learning

Name: Deepali Daga **Date:16-02-24**
UID: 2021600012 **Batch: A**

Experiment 3: Multivariable Regression

AIM/OBJECTIVE: Implement Multivariable and polynomial regression technique using open-source software**.**

Outcomes:

- **Explore the Dataset suitable for multivariable regression problem**
- **Explore the pattern from the dataset and apply suitable algorithm**

System Requirements:

Linux OS with Python and libraries or R or windows with MATLAB

Theory: Part I: Multiple Linear Regression

Part II: Polynomial Regression

ALGORITHM:
Step 1: Create a sample dataset with multiple independent variables and one dependent variable (Y).
Step 2: The data is split into training and testing sets using the train_test_split function.
Step3: A linear regression model is created and fitted to the training data.
Step4: Predictions are made on the test set.
Step5: The model is evaluated using metrics like Mean Absolute Error, Mean Squared Error, and Root Mean Squared Error.
Step6: Finally, the coefficients and intercept of the regression equation are printed.

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
import matplotlib.pyplot as plt
import seaborn as sns
```

from sklearn import metrics

dataset link and description:

1. Mutivariabe Regression

price	area	rooms	parking
250000	1500	3	2
300000	1800	4	2
280000	1700	3	1
320000	2000	4	2
350000	2200	5	2
270000	1600	3	1
290000	1750	3	1
310000	1900	4	2
330000	2100	4	2
340000	2150	4	2
260000	1550	3	1
280000	1650	3	1
310000	1950	4	2
325000	2050	4	2
335000	2150	4	2
300000	1850	4	1
280000	1700	3	1
310000	2000	4	2
325000	2100	4	2
345000	2250	5	2

The dataset is about housing prices. Here independent variables are area, rooms and parking. While the dependent variable is price of the house.

2. Polynomial Regression

price	area
250000	1500
300000	1800
280000	1700
320000	2000
350000	2200
270000	1600
290000	1750
310000	1900
330000	2100
340000	2150
260000	1550
280000	1650
310000	1950
325000	2050
335000	2150
300000	1850
280000	1700
310000	2000
325000	2100
345000	2250

The dataset is about the area and price of the house. Price of house should increase as the area increases.

Program / Output and Interpretation: Part 1 : Multivariable Regression

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error


import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Load the dataset
df = pd.read_csv('housing.csv')
```

```
# Initialize MinMaxScaler
scaler = MinMaxScaler()

# Fit and transform the entire dataset (including the target variable)
df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)

# Save the scaled dataset to a new CSV file
df_scaled.to_csv('datafinal.csv', index=False)


# Fit the scaler to the data
scaler.fit(df)

# Get the scaling parameters (minimum and maximum values) for each column
scaling_params = pd.DataFrame({
    'Feature': df.columns,
    'Min': scaler.data_min_,
    'Max': scaler.data_max_
})

print(scaling_params)


df = pd.read_csv('datafinal.csv')
df

# Separate independent and dependent variables
X = df[['area', 'rooms', 'parking']]  # Independent variables
Y = df['price']  # Dependent variable


import numpy as np

# Define the design matrix X
X = np.column_stack((np.ones_like(df['area']), df['area'], df['rooms'], df['parking']))

# Transpose of X
X_transpose = X.T

# Compute X^T * X
X_transpose_X = X_transpose @ X

# Compute the inverse of X^T * X
inverse_X_transpose_X = np.linalg.inv(X_transpose_X)

# Compute the inverse product of X^T * X and X^T
inverse_product = inverse_X_transpose_X @ X_transpose

# Assuming 'Y' is your target variable
Y = df['price']
```

```
# Compute the regression coefficients 'a'
a = inverse_product @ Y

# Compute the predicted Y values
predicted_Y = X @ a

# Print the results
print("Inverse of (X^T X):")
print(inverse_X_transpose_X)

print("\nRegression Coefficients (a):")
print(a)

print("\nPredicted Y values:")
print(predicted_Y)


# 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 the model
model = LinearRegression()
model.fit(X_train, Y_train)

# Make predictions
Y_pred = model.predict(X_test)



# Calculate evaluation metrics
mae = mean_absolute_error(Y_test, Y_pred)
mse = mean_squared_error(Y_test, Y_pred)
rmse = np.sqrt(mse)

print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)

plt.figure(figsize=(10, 5))
plt.plot(np.arange(len(Y_test)), Y_test, marker='o', label='Actual Price')
plt.plot(np.arange(len(Y_test)), Y_pred, marker='x', linestyle='--', label='Predicted Price')
plt.xlabel('Sample')
plt.ylabel('Price')
plt.title('Actual vs. Predicted Housing Prices')
plt.legend()
plt.grid(True)
plt.show()
```

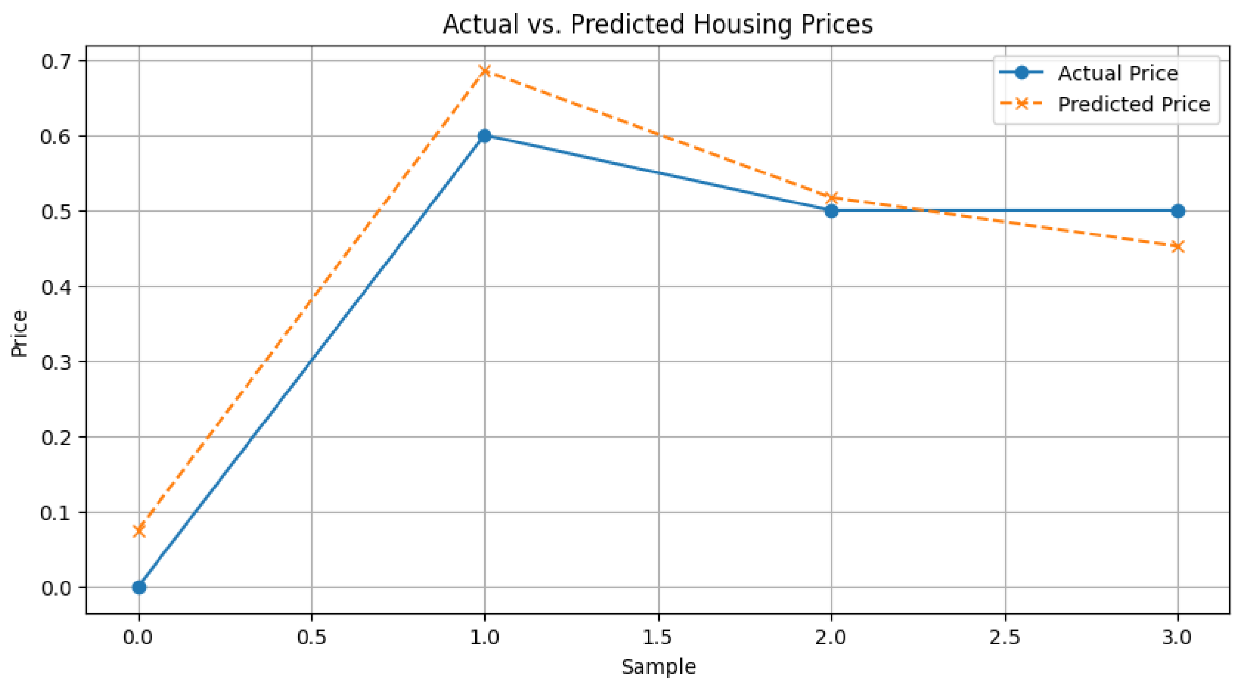
Matrix Operations –

```
Inverse of (X^T X):
[[ 0.24817135 -0.52024639  0.3291219  -0.07055612]
 [-0.52024639  2.70885101 -2.1072341  -0.1926644 ]
 [ 0.3291219  -2.1072341   2.69737164 -0.34431106]
 [-0.07055612 -0.1926644  -0.34431106  0.46428446]]

Regression Coefficients (a):
[ 0.07259126  0.8972544   0.07324047 -0.03169426]

Predicted Y values:
[0.040897   0.436419   0.3118591  0.67568684 0.95157491 0.19222518
 0.37167606 0.55605292 0.79532076 0.85513772 0.13240822 0.25204214
 0.61586988 0.7355038  0.85513772 0.52793021 0.3118591  0.67568684
 0.79532076 1.01139187]
```

Error –

```
Mean Absolute Error: 0.05648584905660388
Mean Squared Error: 0.0038955613652546686
Root Mean Squared Error: 0.06241439046025739
```

Visualization –



Program / Output and Interpretation: Part 2 : Polynomial Regression

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd


dataset = pd.read_csv('example.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
print(X.shape)
print(y.shape)




from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)





from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4)
X_poly = poly_reg.fit_transform(X)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)




plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg.predict(X), color = 'blue')
plt.title('House Price Prediction (Linear Regression)')
plt.xlabel('Area')
plt.ylabel('Price')
plt.show()
```

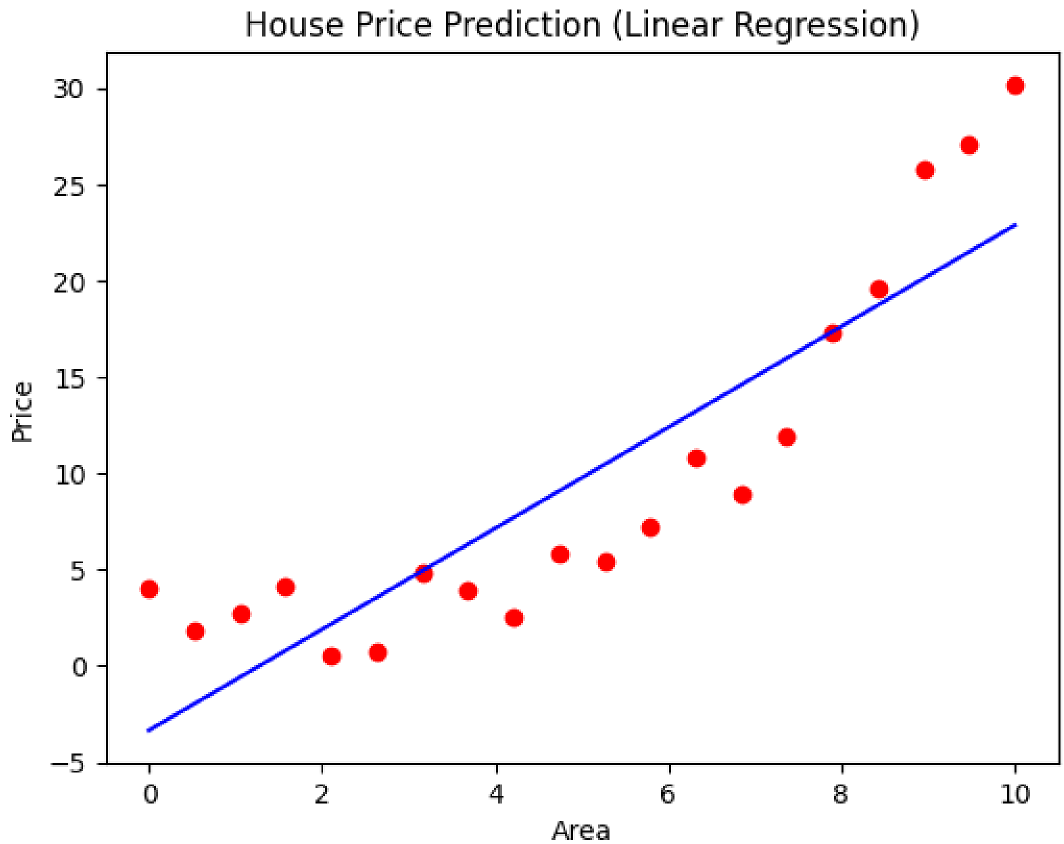
```
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg_2.predict(poly_reg.fit_transform(X)), color = 'blue')
plt.title('House Price prediction (Polynomial Regression)')
plt.xlabel('Area')
plt.ylabel('Price')
plt.show()
```

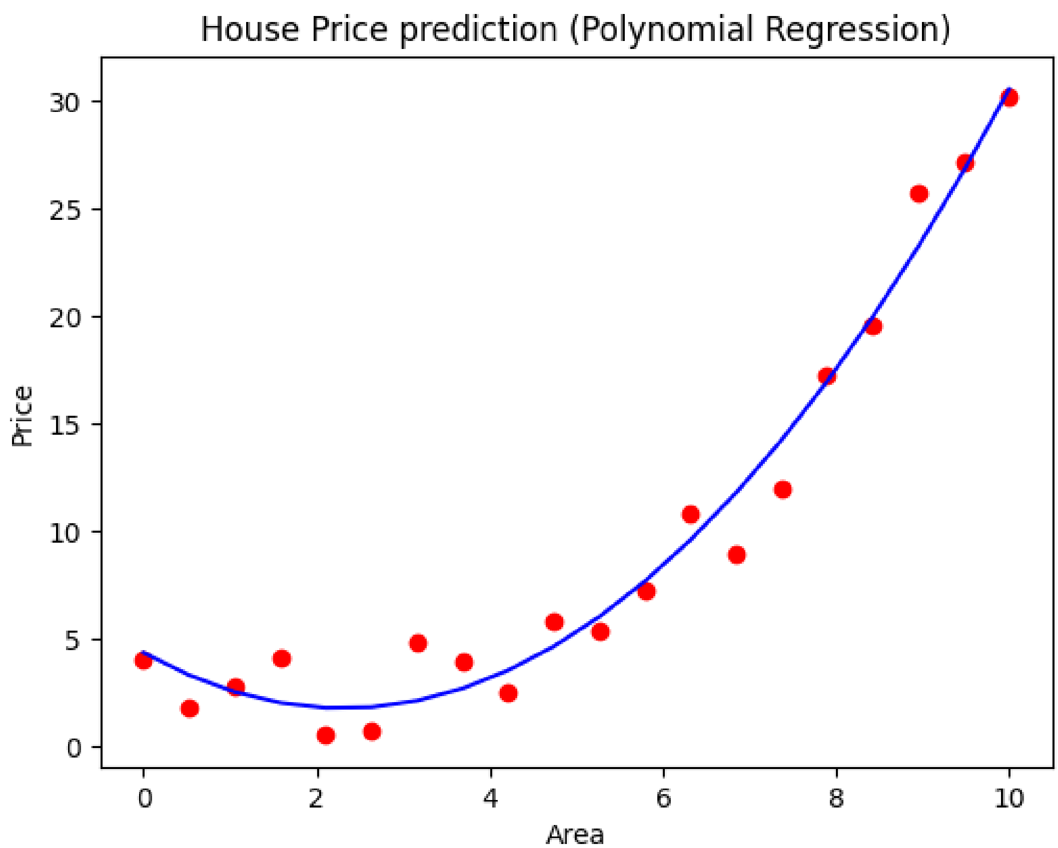
```
X_grid = np.arange(min(X), max(X), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, lin_reg_2.predict(poly_reg.fit_transform(X_grid)), color = 'blue')
plt.title('House price prediction (Polynomial Regression)')
plt.xlabel('Area')
plt.ylabel('Price')
plt.show()
```

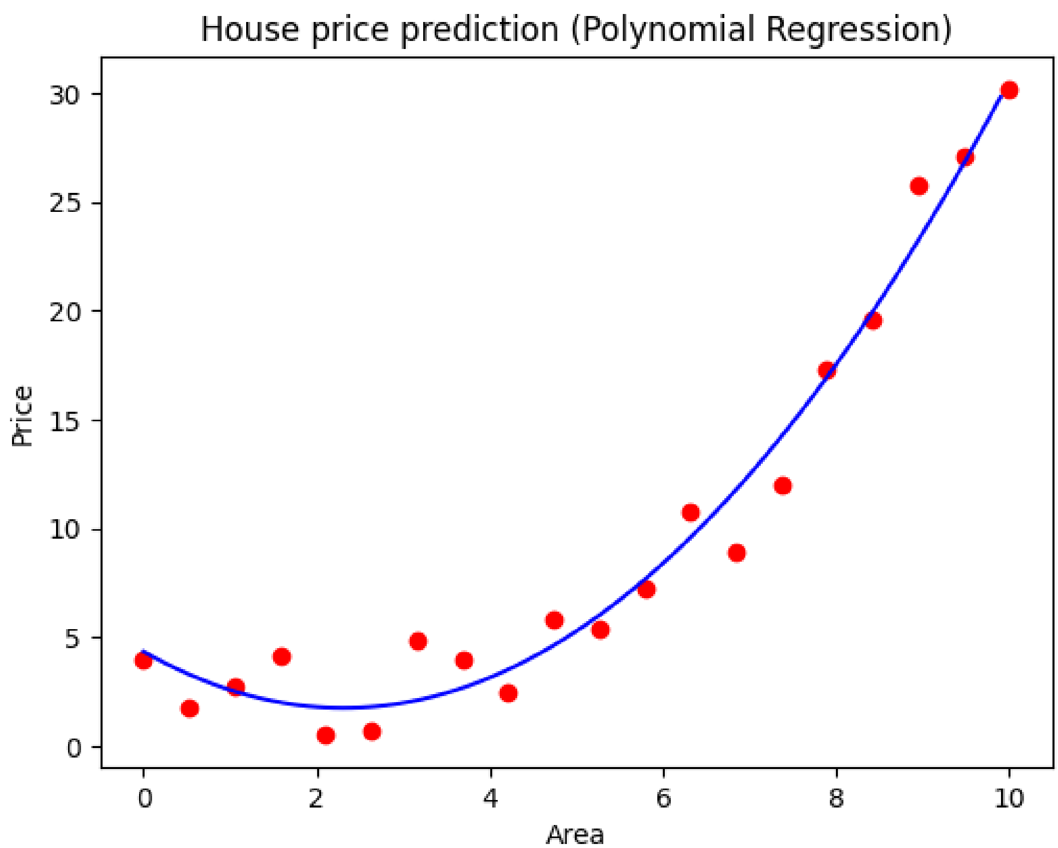
```
lin_reg.predict([[105]])
```

```
lin_reg_2.predict(poly_reg.fit_transform([[105]]))
```

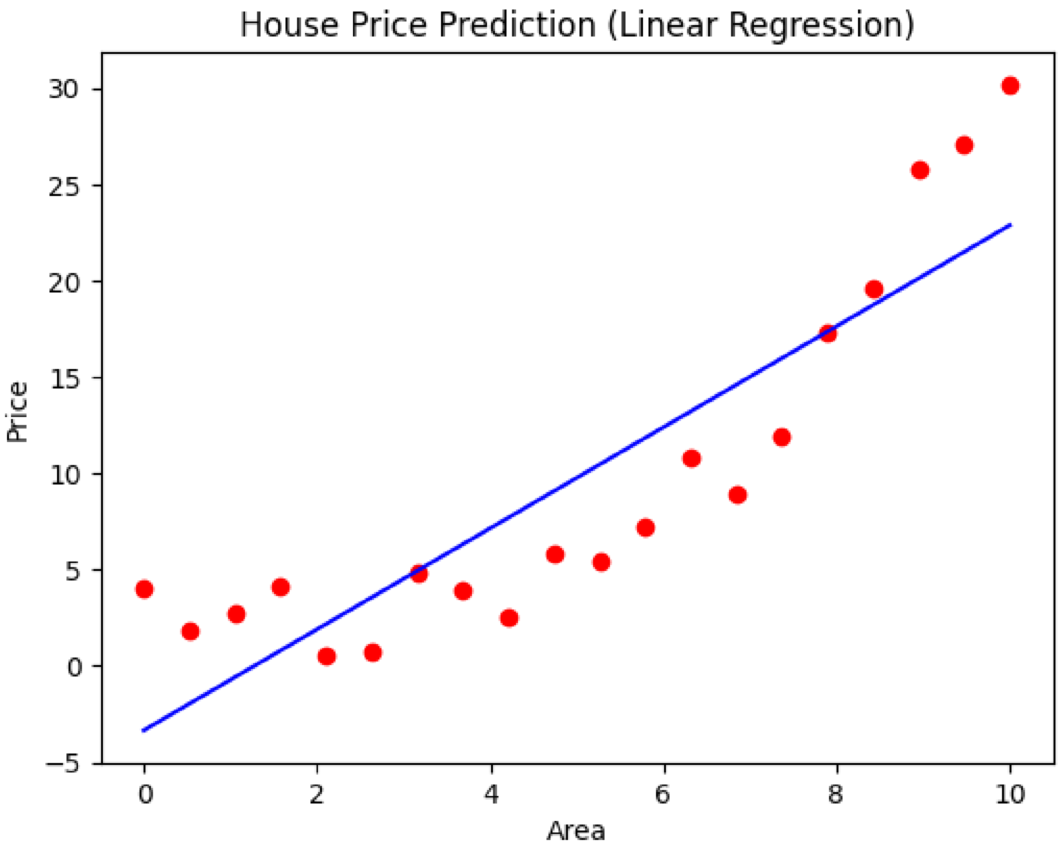
Visualizations –

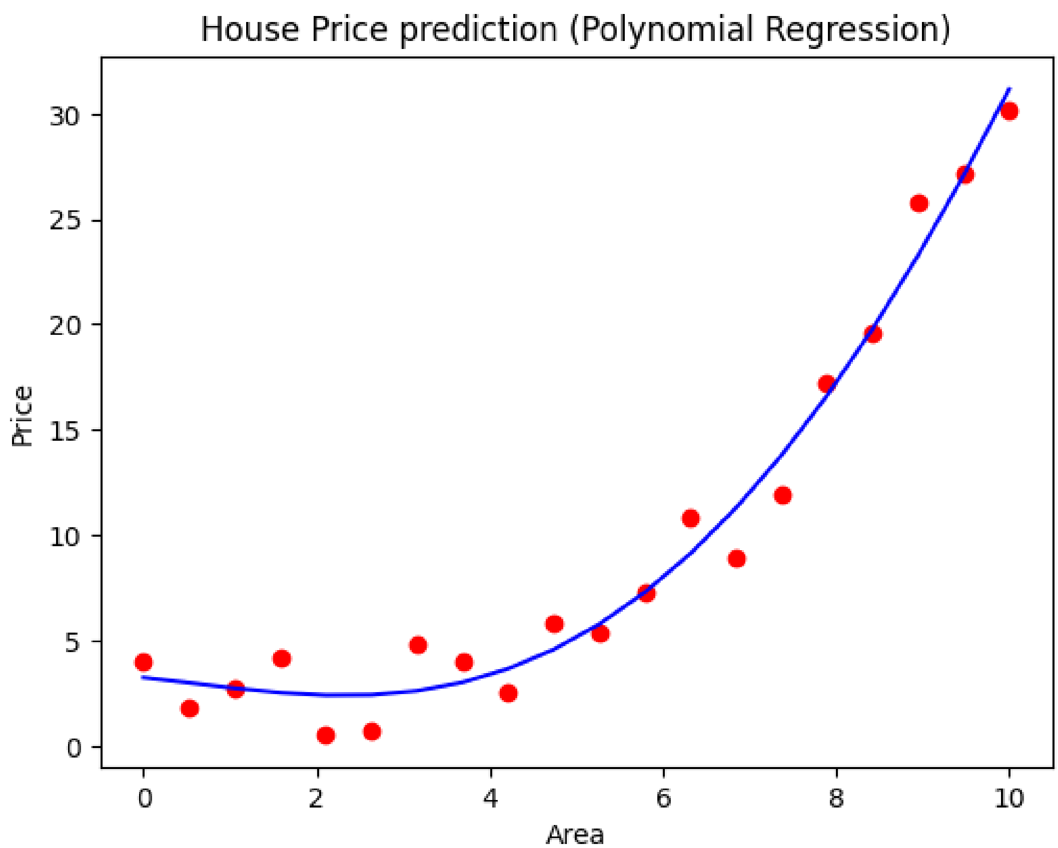

House Price Prediction (Linear Regression)
Price
Area
30
25
20
15
10
5
0
−5
0
2
4
6
8
10


House Price prediction (Polynomial Regression)
Price
30
25
20
15
10
5
0
0
2
4
6
8
10
Area

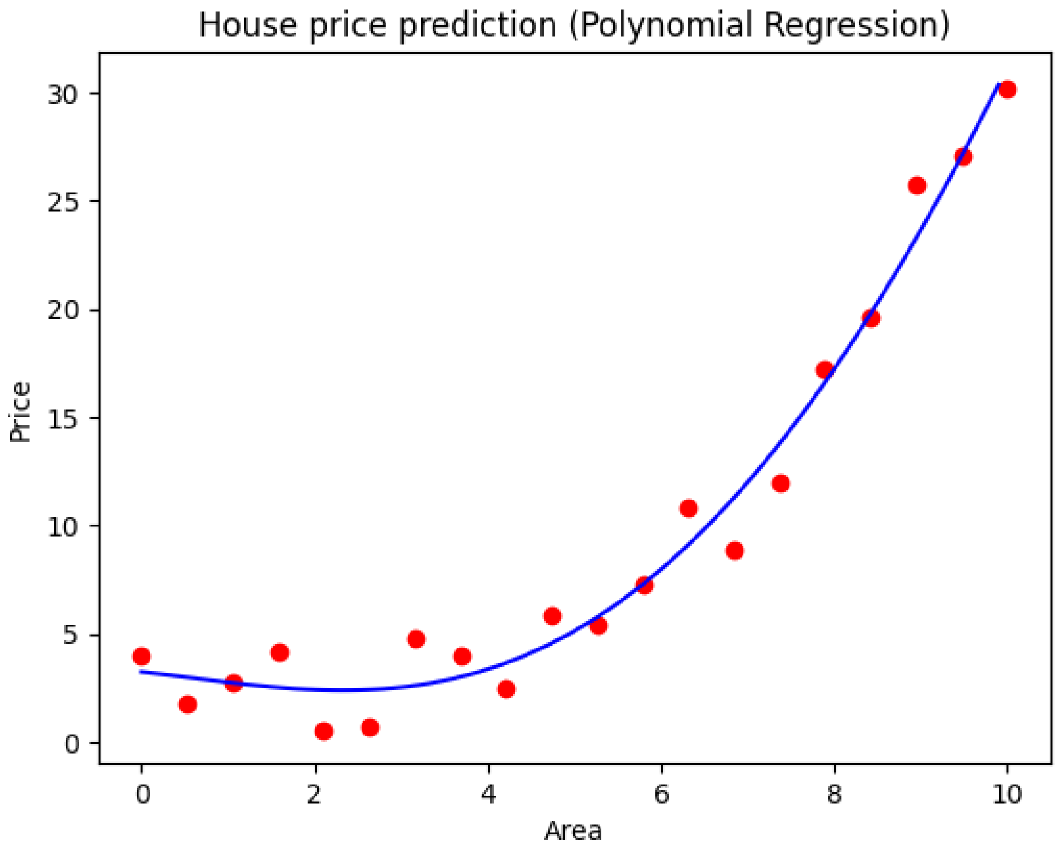

House price prediction (Polynomial Regression)
Price
Area
30
25
20
15
10
5
0
0
2
4
6
8
10

Degree – 4


House Price Prediction (Linear Regression)
Price
Area
30
25
20
15
10
5
0
−5
0
2
4
6
8
10



House Price prediction (Polynomial Regression)
Price
Area
0
5
10
15
20
25
30
0
2
4
6
8
10



Conclusion:

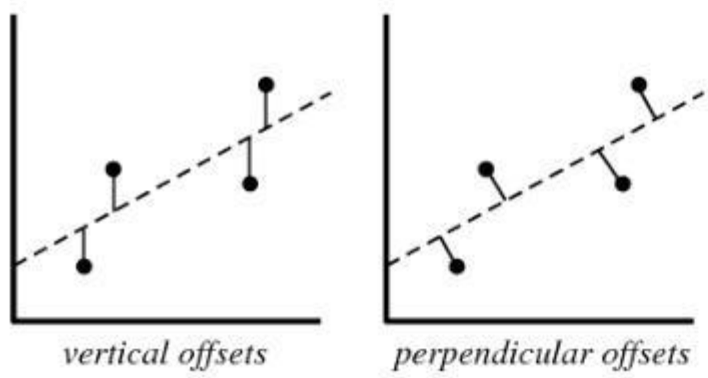
In this experiment, a dataset comprising prices of housing with features such as area, number of rooms , parking and the dependent variable price was analyzed using multi-regression and polynomial regression techniques. The multi-regression model provided insights into the linear relationships between Area, Rooms and Price, yielding interpretable coefficients. The extension to polynomial regression introduced complexity, capturing potential non-linear patterns in the data. The results were visualized by Scatter plots.

Test your skill:

1. How a Multi variable regression technique different from linear regression

   Multi-variable regression is a type of linear regression that involves more than one independent variable. In contrast, simple linear regression deals with only a single independent variable. While both models aim to capture the linear relationship between independent and dependent variables, multi-variable regression extends this by considering multiple predictors simultaneously. It allows for a more comprehensive analysis of the impact of multiple factors on the dependent variable, providing a more nuanced and realistic representation of complex relationships.

2. Which of the following offsets, do we use in case of least square line fit? Suppose horizontal axis is independent variable and vertical axis is dependent variable. Justify.



A. Vertical offset
B. Perpendicular offset
C. Both but depend on situation
D. None of above

Vertical offset in least squares regression quantifies the difference between observed data points and their predicted values by the fitted line. Minimizing these offsets optimizes the line's fit by reducing the sum of squared deviations, ensuring a model that accurately represents the relationship between variables

3. Why do we use multivariable regression models?

Multivariable regression is used to capture multiple influences on a dependent variable, control for confounding, improve predictive power, provide flexibility for non-linear relationships, and offer a more realistic modeling approach.