Name : Bhuvnesh Verma

Roll No : 28

Branch : AIML

SEction : A

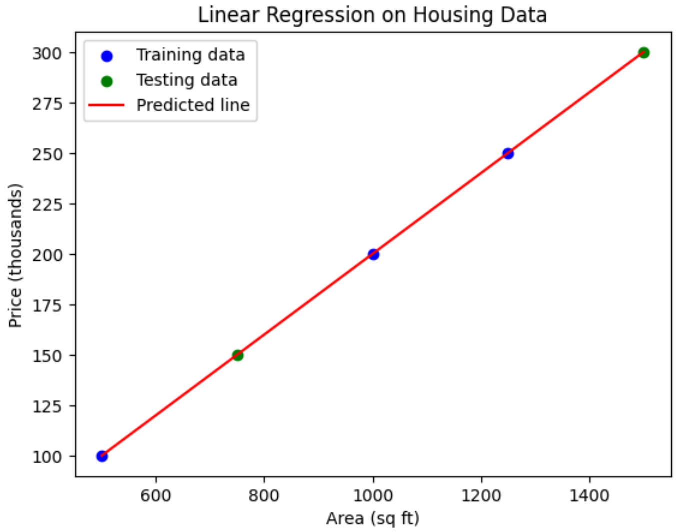
Date : 24 December

Lab : ML [Machine Learning Lab]

Aim : Understand the basics of linear regression using a static dataset and the dataset of Sklearn. Implement it in Python with useful modules such as NumPy, Pandas, and Scikit-learn, and graphically visualize the results using Matplotlib.

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

# Simulated dataset: Housing area vs. price
data = {
    'Area': [500, 750, 1000, 1250, 1500],
    'Price': [100, 150, 200, 250, 300]
}
df = pd.DataFrame(data)

# Separate features (X) and target (y)
X = df[['Area']]  # Independent variable (area)
y = df['Price']   # Dependent variable (price)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)

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

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate the model
print(f"Slope (m): {model.coef_[0]}")
print(f"Intercept (b): {model.intercept_}")
print(f"Test set predictions: {y_pred}")
print(f"Actual test values: {y_test.values}")

# Plot
plt.scatter(X_train, y_train, color='blue', label='Training data')
plt.scatter(X_test, y_test, color='green', label='Testing data')
plt.plot(X, model.predict(X), color='red', label='Predicted line')
plt.xlabel('Area (sq ft)')
plt.ylabel('Price (thousands)')
plt.legend()
plt.title('Linear Regression on Housing Data')
plt.show()

from sklearn.metrics import mean_squared_error, r2_score

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R² Score: {r2}")
```

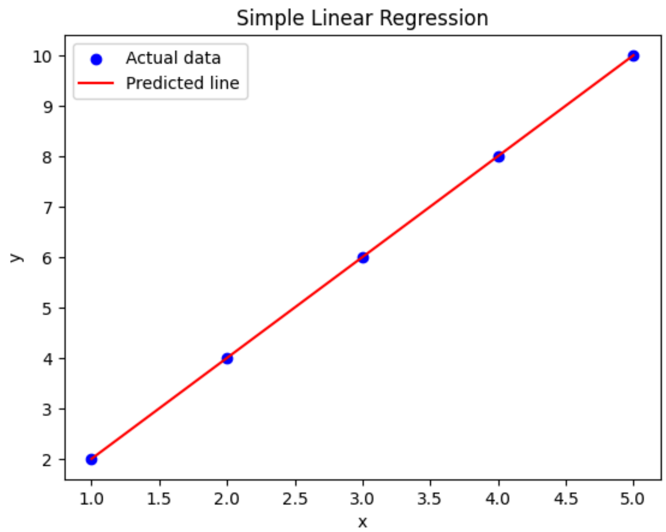
```
Slope (m): 0.19999999999999993
Intercept (b): 8.526512829121202e-14
Test set predictions: [150. 300.]
Actual test values: [150 300]
```




```
Mean Squared Error: 4.0389678347315804e-28
R2 Score: 1.0
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Example data
x = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)  # Independent variable
y = np.array([2, 4, 6, 8, 10])               # Dependent variable

# Create and train the model
model = LinearRegression()
model.fit(x, y)

# Make predictions
y_pred = model.predict(x)

# Display the results
print(f"Slope (m): {model.coef_[0]}")
print(f"Intercept (b): {model.intercept_}")
print(f"Predictions: {y_pred}")

# Plot
plt.scatter(x, y, color='blue', label='Actual data')
plt.plot(x, y_pred, color='red', label='Predicted line')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.title('Simple Linear Regression')
plt.show()
```

```
Slope (m): 2.0
Intercept (b): 0.0
Predictions: [ 2.  4.  6.  8. 10.]
```




```
import pandas as pd
import numpy as np
```

```python
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
# Add another feature to the dataset
data = {
    'Area': [500, 750, 1000, 1250, 1500],
    'Bedrooms': [1, 2, 3, 4, 5],
    'Price': [100, 150, 200, 250, 300]
}
df = pd.DataFrame(data)

# Separate features (X) and target (y)
X = df[['Area', 'Bedrooms']]  # Multiple features
y = df['Price']

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.4, random_state=42)

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

# Predict and evaluate
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error after Standardization: {mse}")
print(f"R² Score after Standardization: {r2}")
```

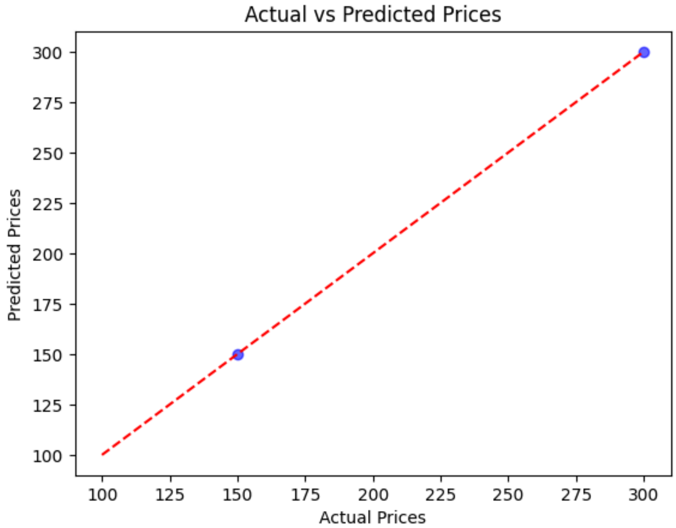
```
Mean Squared Error after Standardization: 8.077935669463161e-27
R² Score after Standardization: 1.0
```

```python
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import pandas as pd

# Dataset with two features
data = {
    'Area': [500, 750, 1000, 1250, 1500],
    'Bedrooms': [1, 2, 3, 4, 5],
    'Price': [100, 150, 200, 250, 300]
}
df = pd.DataFrame(data)

# Separate features and target
X = df[['Area', 'Bedrooms']]
y = df['Price']

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split dataset (increase test size)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.4, random_state=42)

# Check data sizes
print(f"Training samples: {len(X_train)}, Testing samples: {len(X_test)}")

# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Predictions and metrics
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R² Score: {r2}")
```

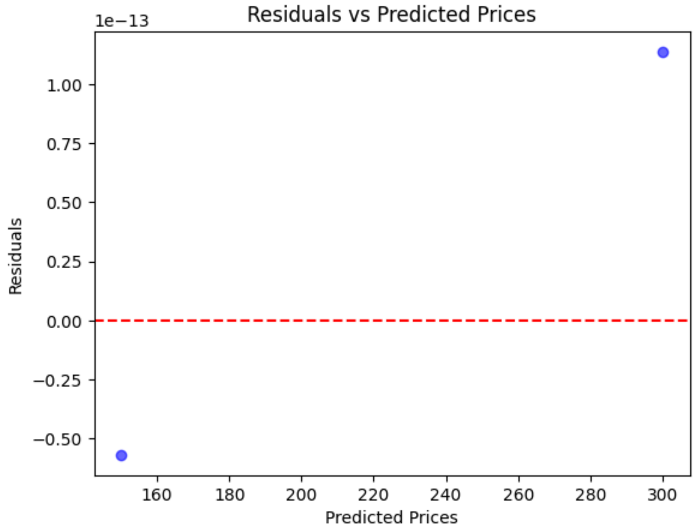
```
Training samples: 3, Testing samples: 2
Mean Squared Error: 8.077935669463161e-27
R² Score: 1.0
```

```python
import matplotlib.pyplot as plt

# Plot actual vs predicted values
plt.scatter(y_test, y_pred, color='blue', alpha=0.6)
plt.plot([y.min(), y.max()], [y.min(), y.max()], color='red', linestyle='--')  # Reference line
```

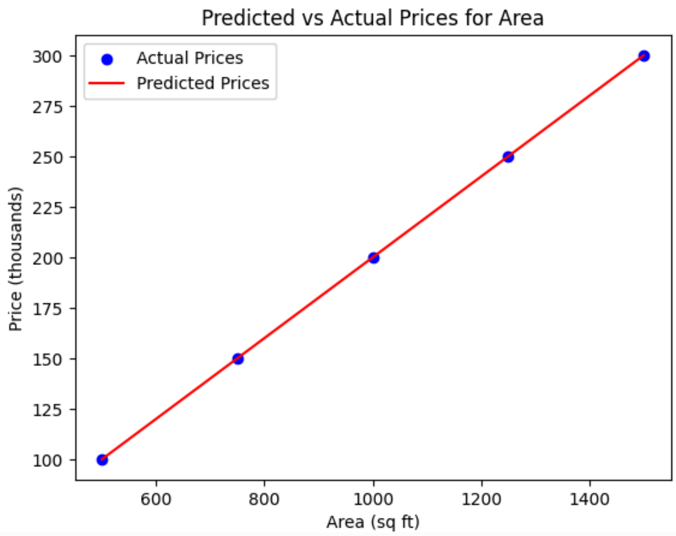
```
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual vs Predicted Prices')
plt.show()
```




```
# Residuals
residuals = y_test - y_pred

# Plot residuals
plt.scatter(y_pred, residuals, color='blue', alpha=0.6)
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel('Predicted Prices')
plt.ylabel('Residuals')
plt.title('Residuals vs Predicted Prices')
plt.show()
```

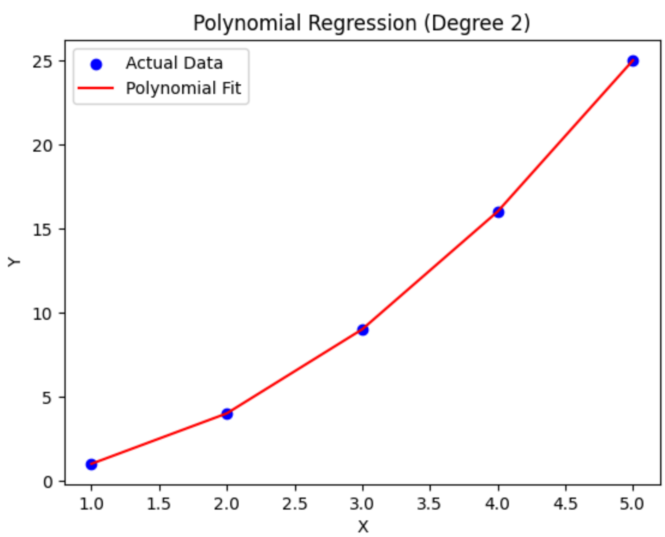



```
# Add predictions to the original dataframe
df['Predicted Price'] = model.predict(scaler.transform(df[['Area', 'Bedrooms']]))

# Plot Area vs Price
plt.scatter(df['Area'], df['Price'], color='blue', label='Actual Prices')
plt.plot(df['Area'], df['Predicted Price'], color='red', label='Predicted Prices')
plt.xlabel('Area (sq ft)')
plt.ylabel('Price (thousands)')
plt.title('Predicted vs Actual Prices for Area')
plt.legend()
plt.show()
```




```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import numpy as np

# Simulated non-linear data
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)  # Independent variable
y = np.array([1, 4, 9, 16, 25])              # Quadratic relationship: y = x^2

# Transform data to include polynomial features (degree 2)
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

# Train polynomial regression model
model = LinearRegression()
model.fit(X_poly, y)

# Predict
y_pred = model.predict(X_poly)

# Evaluate
mse = mean_squared_error(y, y_pred)
r2 = r2_score(y, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R² Score: {r2}")

# Plot
plt.scatter(X, y, color='blue', label='Actual Data')
plt.plot(X, y_pred, color='red', label='Polynomial Fit')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Polynomial Regression (Degree 2)')
plt.legend()
plt.show()
```

```
Mean Squared Error: 1.5146129380243426e-29
R² Score: 1.0
```




```
# Polynomial regression on housing data
```

```
# Polynomial regression on housing data
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X_scaled)

# Split data
X_train_poly, X_test_poly, y_train, y_test = train_test_split(X_poly, y, test_size=0.4, random_state=42)

# Train model
model_poly = LinearRegression()
model_poly.fit(X_train_poly, y_train)

# Predict
y_pred_poly = model_poly.predict(X_test_poly)

# Evaluate
mse_poly = mean_squared_error(y_test, y_pred_poly)
r2_poly = r2_score(y_test, y_pred_poly)

print(f"Polynomial Regression (Degree 2) MSE: {mse_poly}")
print(f"Polynomial Regression (Degree 2) R² Score: {r2_poly}")
```

```
Polynomial Regression (Degree 2) MSE: 0.0
Polynomial Regression (Degree 2) R² Score: 1.0
```

Result and Conclusion: Successfully implemented the linear regression model. Gained an understanding of the logic behind polynomial regression, mean squared error, and R-squared.