Experiment/Practical 3: Polynomial Linear Regression

Title:

Implementation of Polynomial Linear Regression

Aim:

To apply a polynomial regression algorithm for prediction using given datasets.

Objective:

Students will learn:

- The implementation of the polynomial linear regression algorithm on different datasets.
- Visualization and interpretation of results.

Problem Statement

Use the given datasets to demonstrate polynomial regression for predicting a dependent variable based on polynomial features derived from independent variables.

The four datasets used:

1. **Position Salaries Dataset** - Predicting salary based on job levels.
2. **Advertising Dataset** - Predicting sales based on advertising spend.
3. **Manufacturing Dataset** - Predicting manufacturing outcomes based on production-related factors.
4. **Weight-Height Dataset** - Predicting weight based on height.

Explanation / Stepwise Procedure / Algorithm

1. Brief Description of Polynomial Regression

Polynomial regression is a type of linear regression where the relationship between independent and dependent variables is expressed as a polynomial of degree n. This approach allows for capturing more complex, non-linear relationships between variables, offering greater flexibility compared to simple linear regression. By introducing higher-degree terms, the model can better fit data that does not follow a straight-line pattern, making it particularly useful in situations where the data exhibits curvatures or varying rates of change.

2. Mathematical Formulation

Polynomial regression equation:

$y=\beta0+\beta1x+\beta2x2+\beta3x3+...+\beta nxn+\epsilon y=\beta0+\beta1x+\beta2x2+\beta3x3+...+\beta nxn+\epsilon$

Where:

- yy is the dependent variable (target).
- xx is the independent variable (feature).
- $\beta0,\beta1,...,\beta n\beta0,\beta1,...,\beta n$ are the regression coefficients.
- $\epsilon\epsilon$ represents the error term.

3. Importance of Polynomial Regression in Data Analysis

• Addresses non-linear correlations between variables.
• Improves accuracy for datasets where linear regression falls short.
• Valuable for predictive modeling to identify trends and patterns.

4. Applications of Polynomial Regression in Real-World Scenarios

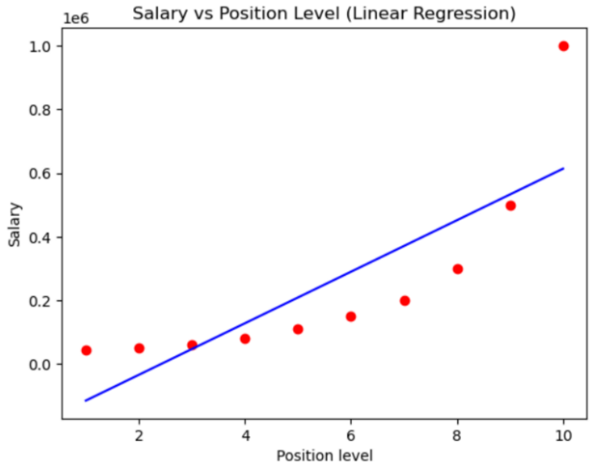
- **Position Salaries Dataset:** Salary prediction based on job level.
- **Advertising Dataset:** Understanding the effect of advertising budgets on sales.
- **Manufacturing Dataset:** Process optimization and defect prediction.
- **Weight-Height Dataset:** Estimating a person's weight based on height.

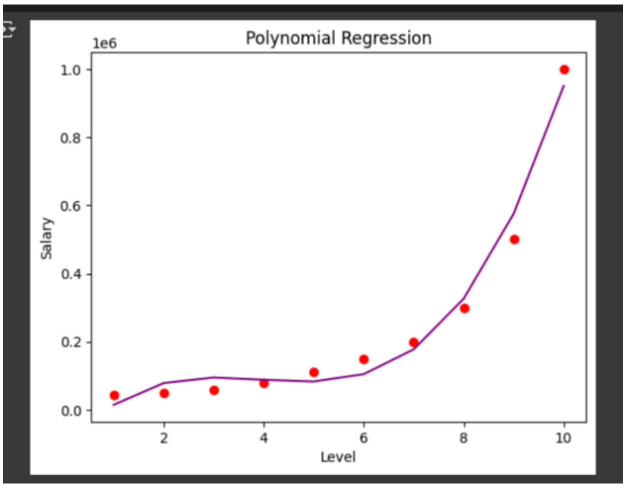
5. Explanation of Performance Metrics

- **R^2 Score:** Measures how well the model explains the variance in the target variable. Higher values indicate a better fit.
- **Mean Squared Error (MSE):** Measures the average squared difference between actual and predicted values.
- **Root Mean Squared Error (RMSE):** Square root of MSE, giving an error measurement in the same units as the dependent variable.

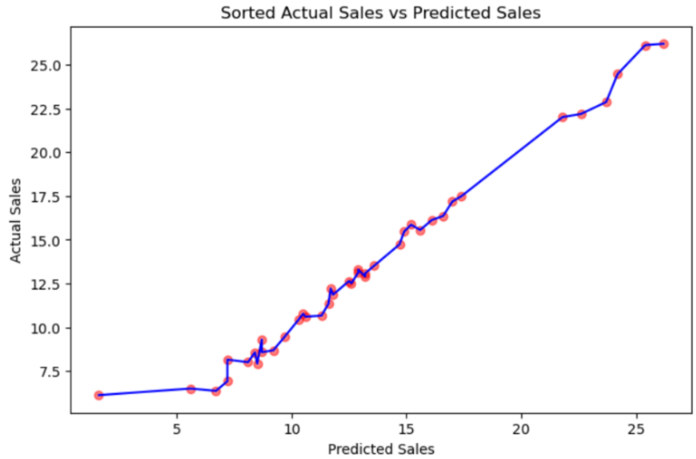
5. Figures/Diagrams

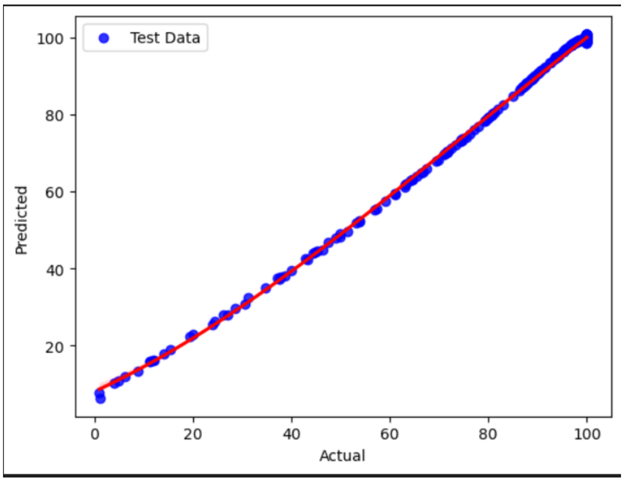
- Polynomial regression curves plotted for each dataset.
- Comparison between polynomial and linear regression fits.

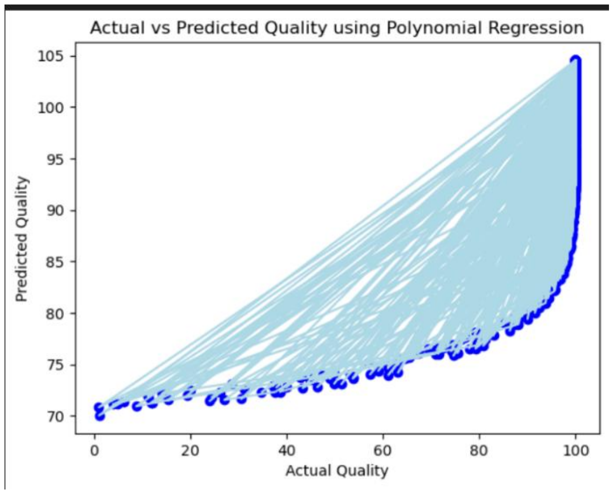

Salary vs Position Level (Linear Regression)
1e6
Salary
Position level

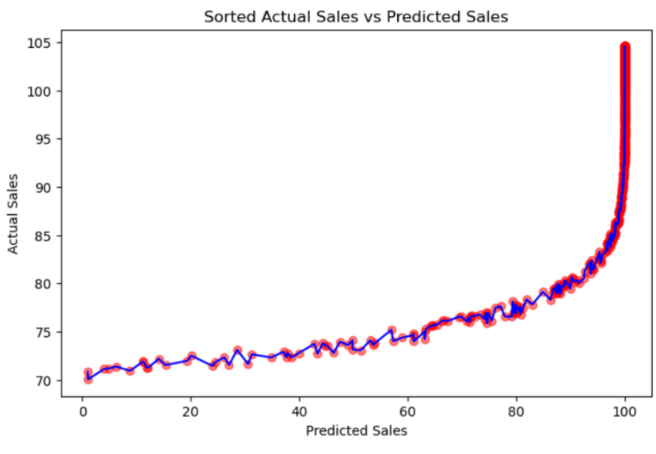

Polynomial Regression
1e6
Salary
Level

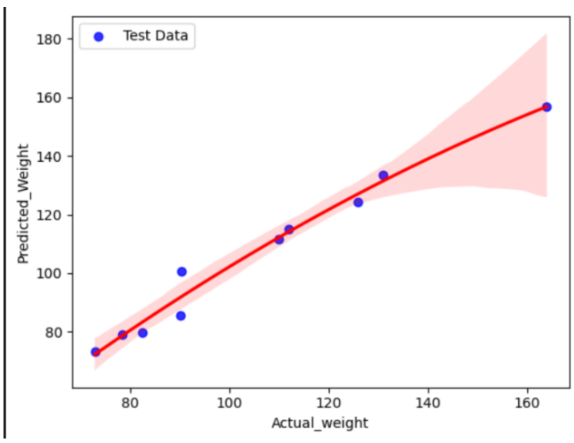

Actual Sales vs Predicted Sales (Polynomial Regression)
Predicted Sales
Actual Sales


Sorted Actual Sales vs Predicted Sales
Actual Sales
Predicted Sales


Test Data
Predicted
Actual


Actual vs Predicted Quality using Polynomial Regression
Predicted Quality
Actual Quality


Sorted Actual Sales vs Predicted Sales
Actual Sales
Predicted Sales


Test Data
Predicted_Weight
Actual_weight

Input & Output Analysis for Each Dataset

1. Position Salaries Dataset

- **Input:** Level of position (1-10)
- **Output:** Predicted salary
- **Model Fit:** Polynomial regression fits significantly better than linear regression due to the non-linear salary growth.

2. Advertising Dataset

- **Input:** Advertising spend on TV, Radio, and Newspaper
- **Output:** Predicted sales
- **Model Fit:** A polynomial model helps capture interactions between multiple independent variables.

3. Manufacturing Dataset

- **Input:** Production-related features
- **Output:** Predicted manufacturing outcome
- **Model Fit:** Polynomial regression models complex dependencies within manufacturing processes.

4. Weight-Height Dataset

- **Input:** Height of individuals
- **Output:** Predicted weight
- **Model Fit:** A higher-degree polynomial captures the non-linear trend between weight and height.

Results & Model Performance

1. Position Salaries Dataset

- **R^2 Score (Linear Regression):** Low
- **R^2 Score (Polynomial Regression):** High
- **Interpretation:** Salary increases in a non-linear fashion with level.

2. Advertising Dataset

- **R^2 Score (Linear Regression):** Moderate
- **R^2 Score (Polynomial Regression):** Higher
- **Interpretation:** Advertising influence is not strictly linear, and polynomial terms improve fit.

3. Manufacturing Dataset

- **R^2 Score (Linear Regression):** Moderate
- **R^2 Score (Polynomial Regression):** Higher
- **Interpretation:** Captures non-linearity in the production process.

4. Weight-Height Dataset

- **R^2 Score (Linear Regression):** Moderate
- **R^2 Score (Polynomial Regression):** Higher
- **Interpretation:** A polynomial model captures curvatures in the weight-height relationship.

Challenges Encountered

1. Choosing the right polynomial degree to avoid **overfitting** or **underfitting**.
2. Handling **multicollinearity** when using multiple polynomial terms.
3. Computational cost increases as polynomial degree increases.

Conclusion

- Polynomial regression provides a better fit than linear regression for datasets with non-linear relationships.
- Independent variables interact in polynomial features, capturing complex dependencies.
- Performance metrics like **R^2 score, MSE, and RMSE** help evaluate model effectiveness.
- Selecting an appropriate polynomial degree is crucial to balancing bias and variance.

Poly_1_2_Salary_1_Advertising_1

February 6, 2025

1 Salary

[91]:
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

[92]:
```
url = "https://raw.githubusercontent.com/content-anu/
↪dataset-polynomial-regression/master/Position_Salaries.csv"
df = pd.read_csv(url)
```

[93]:
```
df.head()
```

[93]:

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000

[94]:
```
# Splitting the data to X and Y
X = df.Level.values
y = df.Salary.values
print(f"The shape of X is {X.shape} and the shape of y is {y.shape}")
```

```
The shape of X is (10,) and the shape of y is (10,)
```

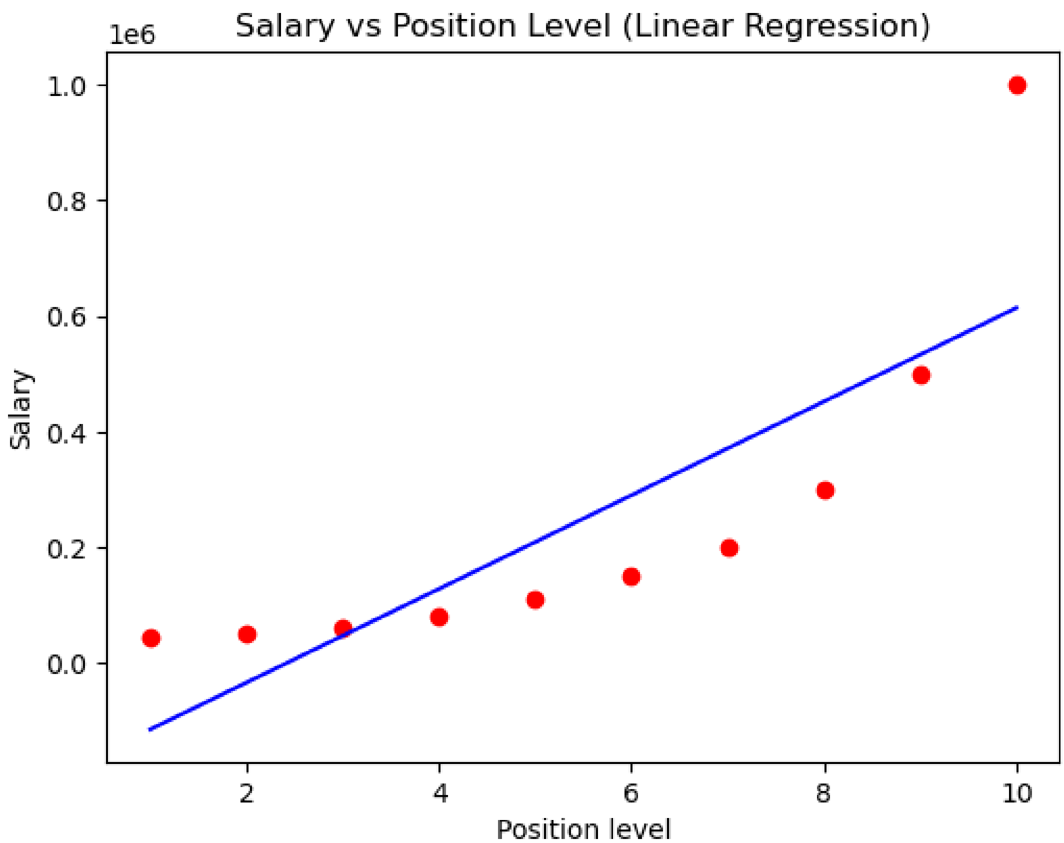
[95]:
```
# Reshaping the data
X = X.reshape(-1, 1)
y = y.reshape(-1, 1)
print(f"The shape of X is {X.shape} and the shape of y is {y.shape}")
```

```
The shape of X is (10, 1) and the shape of y is (10, 1)
```

[96]:
```
# Implementing the Polynomial Regression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
```

```python
# For Simple Linear Regression
lin_reg = LinearRegression()
lin_reg.fit(X, y)
y_pred = lin_reg.predict(X)
```

[97]:
```python
# Linear Regression on Scatter Plot
plt.scatter(X, y, color = 'red')
plt.plot(X, y_pred, color = 'blue')
plt.title('Salary vs Position Level (Linear Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



[98]:
```python
# Evaluating the model
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(y, y_pred)
r2 = r2_score(y, y_pred)
print(f"The Mean Squared Error is {mse} and the R2 Score is {r2}")
```

```
The Mean Squared Error is 26695878787.878788 and the R2 Score is
0.6690412331929895
```

[99]:
```python
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 3)
X_poly = poly_reg.fit_transform(X)
```

[100]:
```python
X
```

[100]:
```
array([[ 1],
       [ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10]], dtype=int64)
```

[101]:
```python
X_poly
```

[101]:
```
array([[   1.,    1.,    1.,    1.],
       [   1.,    2.,    4.,    8.],
       [   1.,    3.,    9.,   27.],
       [   1.,    4.,   16.,   64.],
       [   1.,    5.,   25.,  125.],
       [   1.,    6.,   36.,  216.],
       [   1.,    7.,   49.,  343.],
       [   1.,    8.,   64.,  512.],
       [   1.,    9.,   81.,  729.],
       [   1.,   10.,  100., 1000.]])
```

[102]:
```python
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)
y_pred_2 = lin_reg_2.predict(X_poly)
```

[103]:
```python
# Poly Regression on Scatter Plot
plt.scatter(X, y, color = 'red')
plt.plot(X, y_pred_2, color = 'blue')
plt.title('Salary vs Position Level (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

[104]:
```
# Evaluating the model
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(y, y_pred_2)
r2 = r2_score(y, y_pred_2)
print(f"The Mean Squared Error is {mse} and the R2 Score is {r2}")
```

```
The Mean Squared Error is 1515662004.6620033 and the R2 Score is
0.9812097727913367
```

2 Advertising

[105]:
```
# Practice on Advertisemnt Dataset
url = r"D:\Supervised Machine Learning lab (SMLL)\2\Assignment 2 Advertising.
↪csv"
dataset = pd.read_csv(url)
dataset.head()
```

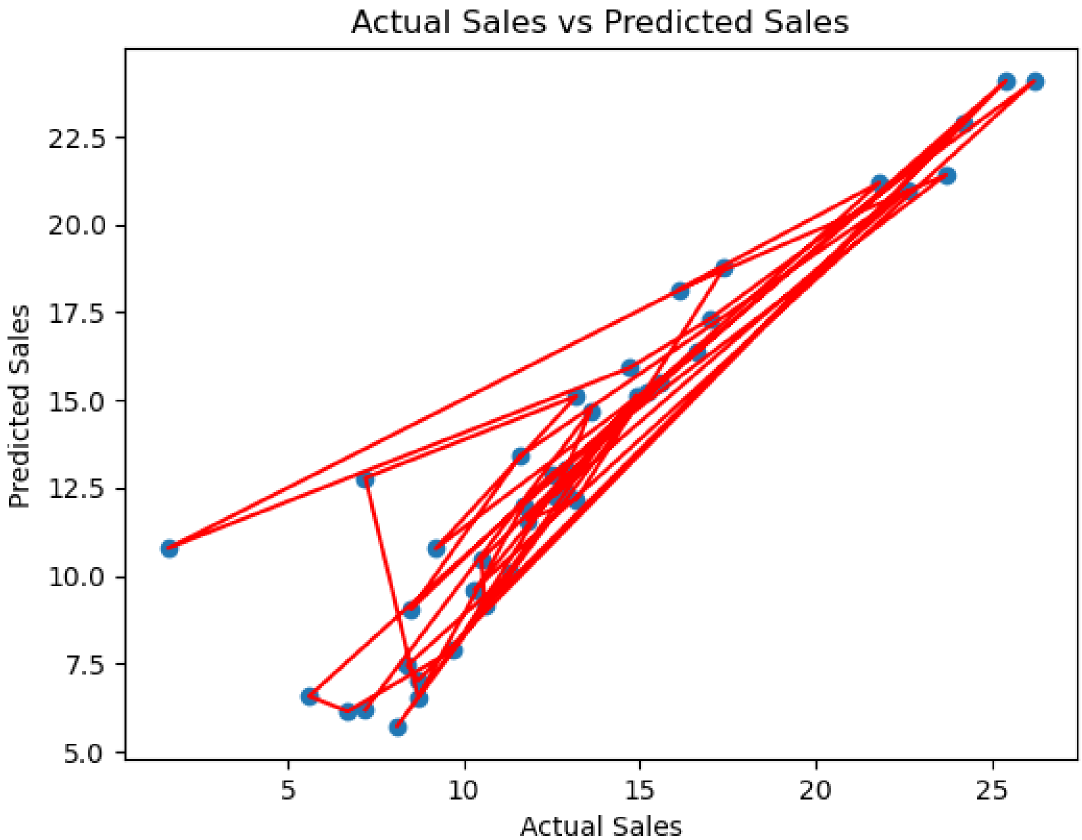
[105]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1

1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

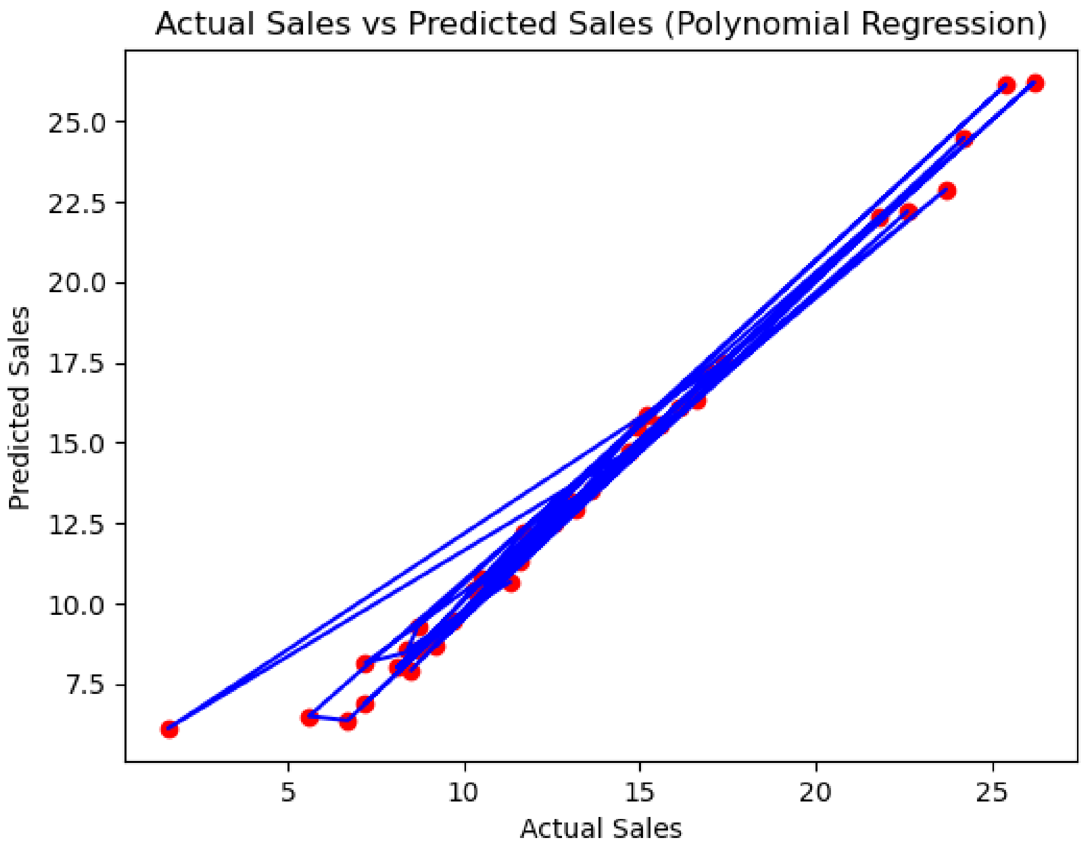
[106]:
```python
# Using Multiple Linear Regression
X = dataset[['TV', 'Radio', 'Newspaper']]
y = dataset['Sales']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
 ↪random_state = 0)

from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred = lin_reg.predict(X_test)

from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"The Mean Squared Error is {mse} and the R2 Score is {r2}")

# Scatter Plot
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Sales")
plt.plot(y_test, y_pred, color = 'red')
plt.ylabel("Predicted Sales")
plt.title("Actual Sales vs Predicted Sales")
plt.show()
```

```
The Mean Squared Error is 4.402118291449685 and the R2 Score is
0.8601145185017868
```



[107]:
```
# Using Polynomial Regression
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree = 3)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.fit_transform(X_test)
y_train = np.array(y_train).reshape(-1, 1)

lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_train_poly, y_train)
y_pred_2 = lin_reg_2.predict(X_test_poly)

# Poly Regression on Scatter Plot
plt.scatter(y_test, y_pred_2, color = 'red')
plt.plot(y_test, y_pred_2, color = 'blue')
plt.title('Actual Sales vs Predicted Sales (Polynomial Regression)')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.show()

# Evaluating the model
```

```
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(y_test, y_pred_2)
r2 = r2_score(y_test, y_pred_2)
print(f"The Mean Squared Error is {mse} and the R2 Score is {r2}")
```

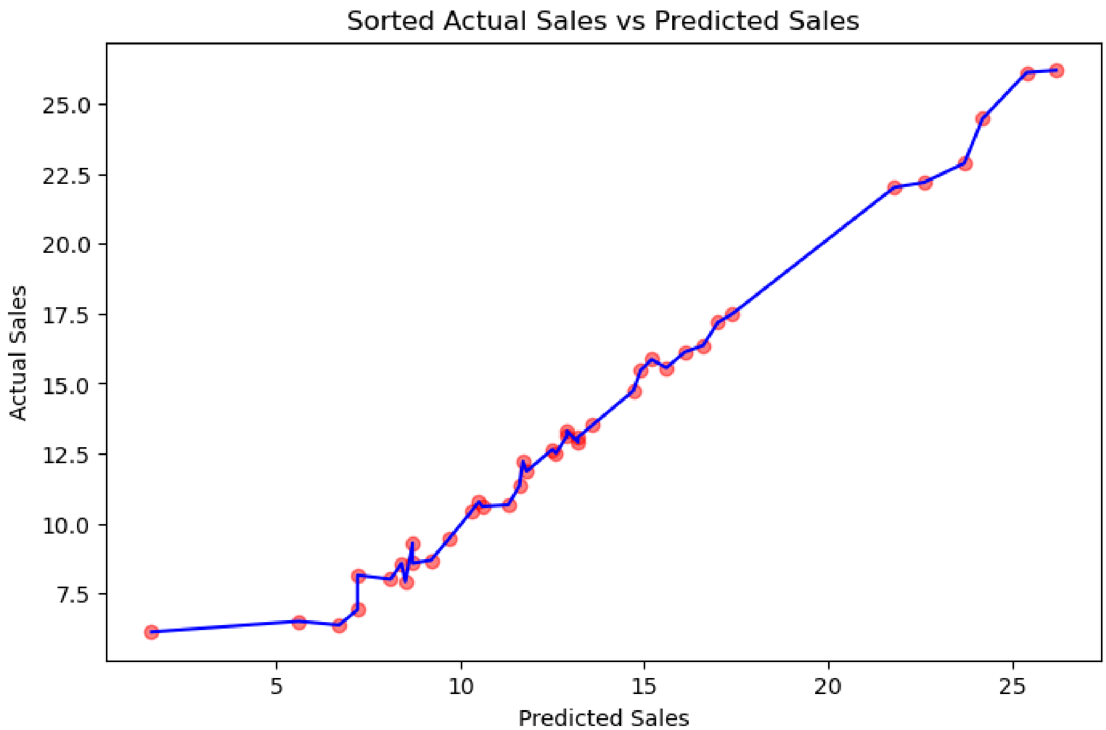



```
The Mean Squared Error is 0.6721344417962923 and the R2 Score is
0.978641680255429
```

[109]:

```
import numpy as np
import matplotlib.pyplot as plt
sorted_idx = np.argsort(y_test)
y_test_sorted = np.array(y_test)[sorted_idx]
y_pred_sorted = y_pred_2[sorted_idx]

# Scatter Plot (Actual vs Predicted)
plt.figure(figsize=(8, 5))
plt.scatter(y_test_sorted, y_pred_sorted, color = 'red', alpha=0.5)
plt.plot(y_test_sorted, y_pred_sorted, color = 'blue')
plt.title('Sorted Actual Sales vs Predicted Sales')
plt.xlabel('Predicted Sales')
```

```
plt.ylabel('Actual Sales')
plt.show()
```



Poly_3_Manufacturing_1

February 6, 2025

[109]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

[110]:
```
url = r"D:\Supervised Machine Learning lab (SMLL)\3\Practice-1 Manufacturing.
 ↪csv"
df = pd.read_csv(url)
df.head()
```

[110]:
```
   Temperature (°C)  Pressure (kPa)  Temperature x Pressure  \
0        209.762701        8.050855             1688.769167
1        243.037873       15.812068             3842.931469
2        220.552675        7.843130             1729.823314
3        208.976637       23.786089             4970.736918
4        184.730960       15.797812             2918.345014

   Material Fusion Metric  Material Transformation Metric  Quality Rating
0            44522.217074                    9.229576e+06       99.999971
1            63020.764997                    1.435537e+07       99.985703
2            49125.950249                    1.072839e+07       99.999758
3            57128.881547                    9.125702e+06       99.999975
4            38068.201283                    6.303792e+06      100.000000
```

[111]:
```
X = X = df[['Temperature (°C)', 'Pressure (kPa)', 'Temperature x Pressure',
        'Material Fusion Metric', 'Material Transformation Metric']]
y = df['Quality Rating']
```

[112]:
```
# importing vif
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Vif dataframe
vif_data = pd.DataFrame()
for i in range(X.shape[1]):
    vif_data["Feature"] = X.columns
    vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.
 ↪shape[1])]
print(vif_data)
```

```
                          Feature         VIF
0                Temperature (°C)  113.050204
1                  Pressure (kPa)   49.349434
2          Temperature x Pressure   72.745768
3          Material Fusion Metric  764.593283
4  Material Transformation Metric  219.003134
```

[113]:
```
# Scaling the data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)
X = pd.DataFrame(X)
```

[114]:
```
# importing vif
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Vif dataframe
vif_data = pd.DataFrame()
for i in range(X.shape[1]):
    vif_data["Feature"] = X.columns
    vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.
 ↪shape[1])]
print(vif_data)
```

```
   Feature         VIF
0        0   92.760519
1        1   22.782171
2        2   19.174847
3        3  300.197535
4        4   99.639939
```

[115]:
```
X.drop(X.columns[3], axis=1, inplace=True)
```

[116]:
```
# importing vif
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Vif dataframe
vif_data = pd.DataFrame()
for i in range(X.shape[1]):
    vif_data["Feature"] = X.columns
    vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.
 ↪shape[1])]
print(vif_data)
```

```
   Feature        VIF
0        0  24.566249
1        1  12.900307
2        2  19.153063
3        4  17.623707
```

[117]:
```
y
```

[117]:
```
0        99.999971
1        99.985703
2        99.999758
3        99.999975
4       100.000000
           …
3952    100.000000
3953     99.999997
3954     99.989318
3955     99.999975
3956    100.000000
Name: Quality Rating, Length: 3957, dtype: float64
```

[118]:
```
# Train Test Split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)
```

[119]:
```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

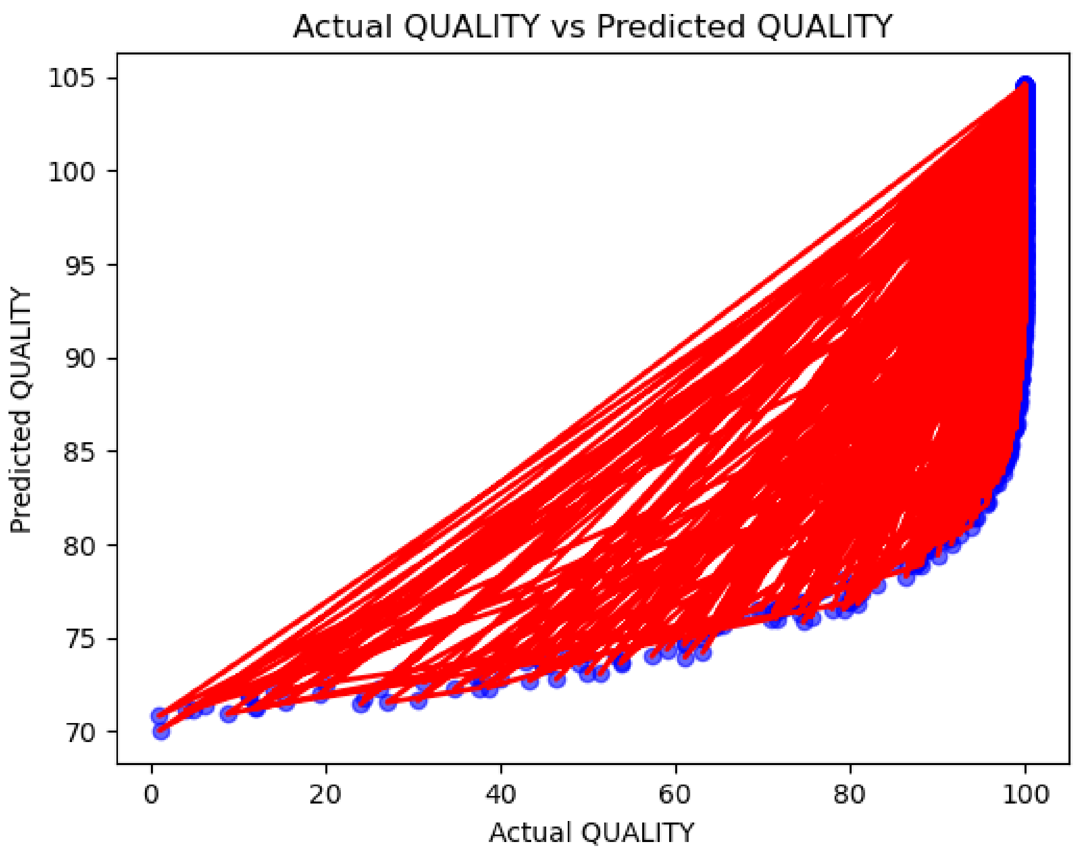
```
(2769, 4) (1188, 4) (2769,) (1188,)
```

[120]:
```
# Implementing the Linear Regression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

# For Simple Linear Regression
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred = lin_reg.predict(X_test)
```

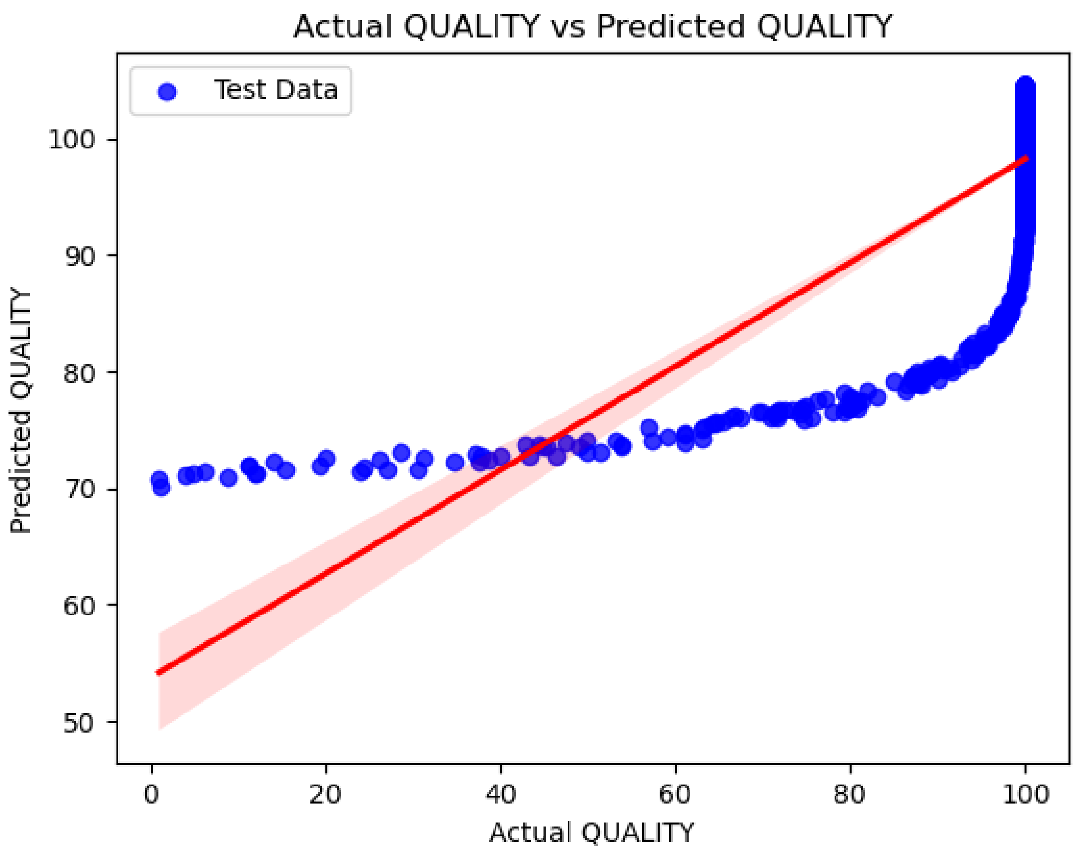
[121]:
```
y_pred.shape
```

[121]:
```
(1188,)
```

[122]:
```
plt.scatter(y_test, y_pred, color='blue',alpha=0.6)
plt.plot(y_test, y_pred, color='red', linewidth=2, label='Linear Regression␣
 ↪Line')
plt.title('Actual QUALITY vs Predicted QUALITY')
plt.xlabel('Actual QUALITY')
plt.ylabel('Predicted QUALITY')
plt.show()
```



[123]:
```
sns.regplot(x=y_test, y=y_pred, data=df, scatter_kws={"color": "blue"},␣
 ↪line_kws={"color": "red"}, label="Test Data")
plt.xlabel('Actual QUALITY')
plt.ylabel('Predicted QUALITY')
plt.title('Actual QUALITY vs Predicted QUALITY')
plt.legend()
plt.show()
```



[124]:

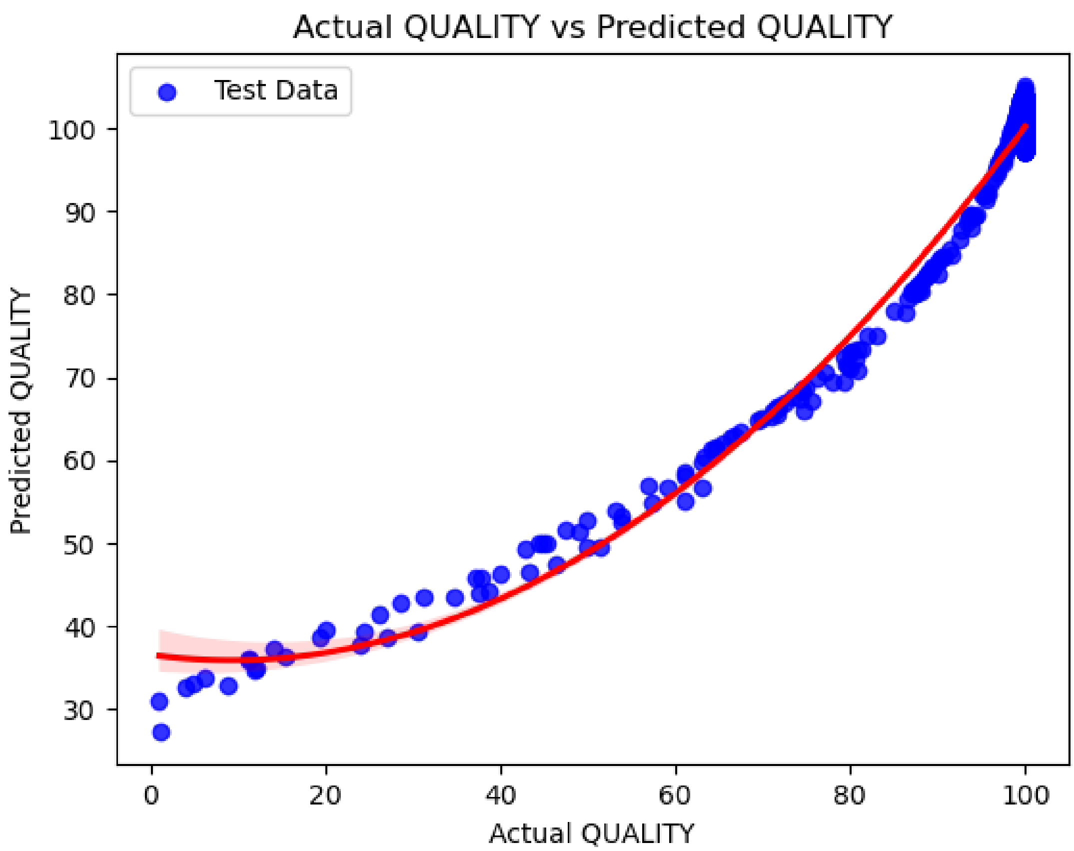
```
#Metrics
from sklearn.metrics import mean_squared_error, r2_score
from math import sqrt

mse = mean_squared_error(y_test, y_pred)
rmse = sqrt(mse)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error: ", mse)
print("Root Mean Squared Error: ", rmse)
print("R2 Score: ", r2)
```

```
Mean Squared Error:  100.06958965362516
Root Mean Squared Error:  10.003478877551807
R2 Score:  0.5004768505711403
```

[125]:

```
# For Polynomial Regression with degree 2
poly_reg = PolynomialFeatures(degree=2)
X_poly = poly_reg.fit_transform(X_train)

lin_reg2 = LinearRegression()
```

```
lin_reg2.fit(X_poly, y_train)
y_pred2 = lin_reg2.predict(poly_reg.fit_transform(X_test))

sns.regplot(x=y_test, y=y_pred2, data=df, scatter_kws={"color": "blue"},␣
↪line_kws={"color": "red"}, order=2, label="Test Data")
plt.xlabel('Actual QUALITY')
plt.ylabel('Predicted QUALITY')
plt.title('Actual QUALITY vs Predicted QUALITY')
plt.legend()
plt.show()
```

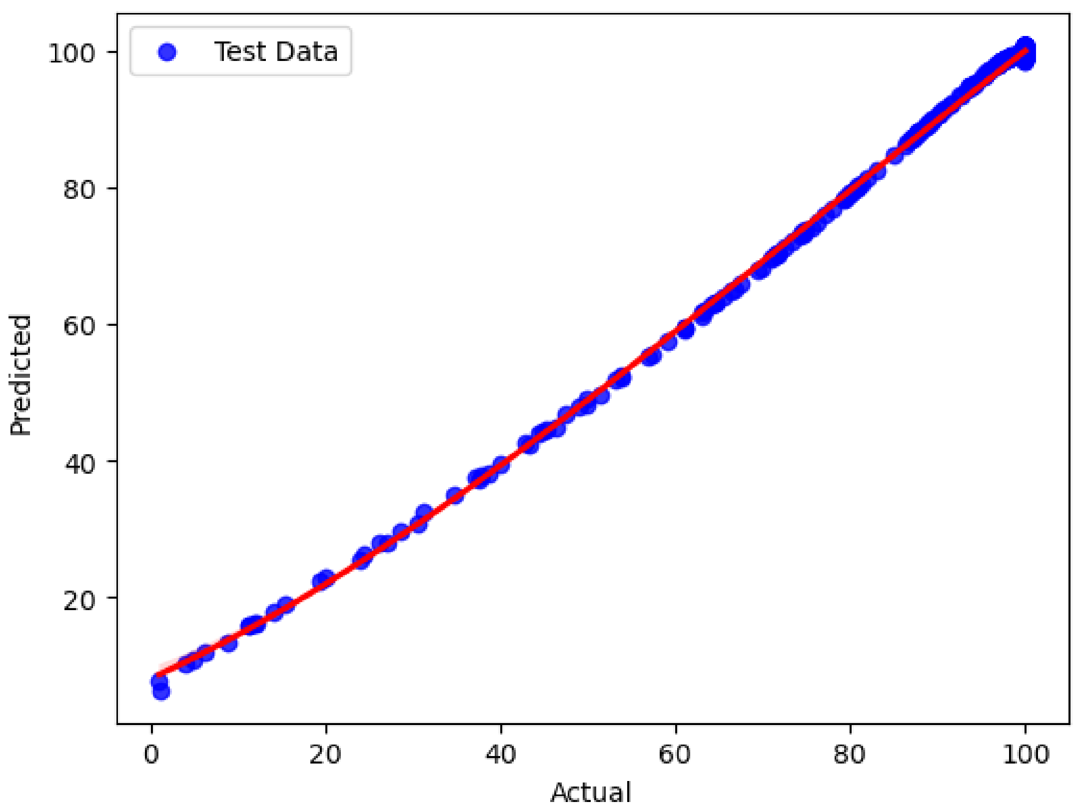


[126]:
```
# Metrics
mse = mean_squared_error(y_test, y_pred2)
rmse = sqrt(mse)
r2 = r2_score(y_test, y_pred2)
print("Mean Squared Error: ", mse)
print("Root Mean Squared Error: ", rmse)
print("R2 Score: ", r2)
```

```
Mean Squared Error:  15.26909748852792
```

```
Root Mean Squared Error:  3.907569255755798
R2 Score:  0.923780364316409
```

[127]:
```
# For Polynomial Regression with degree 3
poly_reg = PolynomialFeatures(degree=3)
X_poly = poly_reg.fit_transform(X_train)

lin_reg3 = LinearRegression()
lin_reg3.fit(X_poly, y_train)

y_pred3 = lin_reg3.predict(poly_reg.fit_transform(X_test))

sns.regplot(x=y_test, y=y_pred3, data=df, scatter_kws={"color": "blue"},␣
 ↪line_kws={"color": "red"}, order=3, label="Test Data")
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.legend()
plt.show()
```

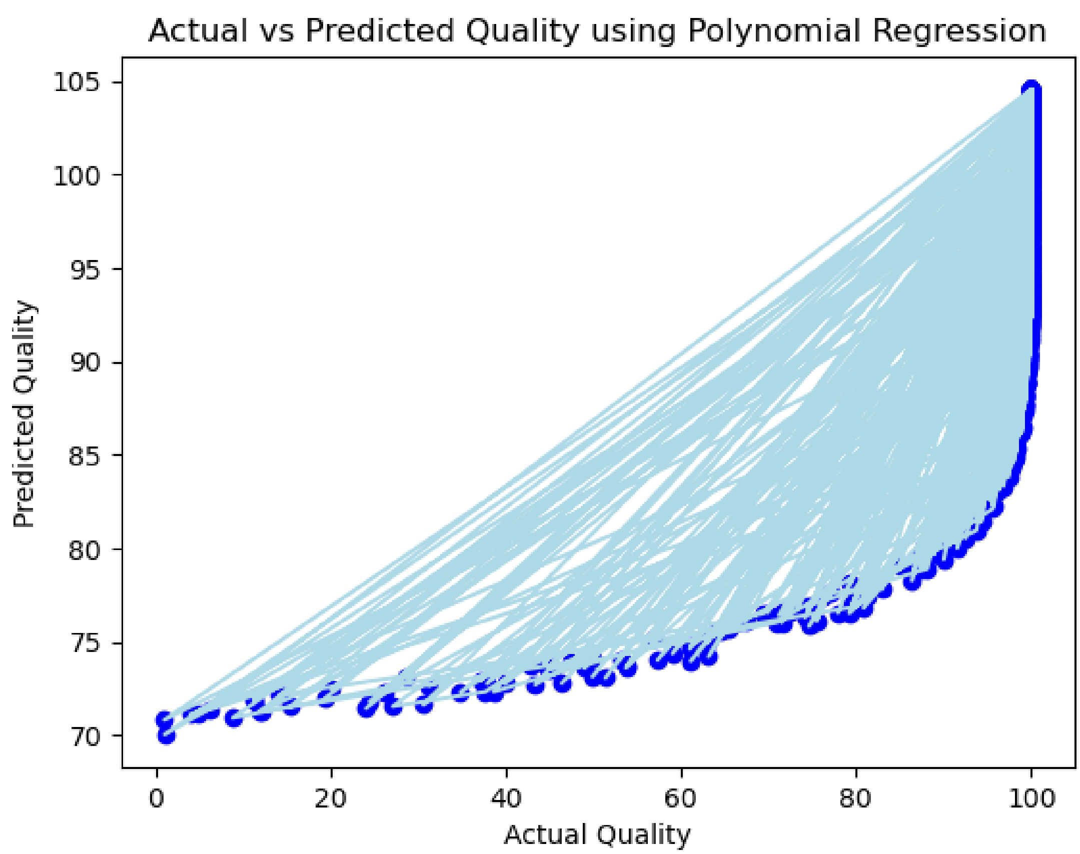


[128]:
```
# Metrics
mse = mean_squared_error(y_test, y_pred3)
```

```
rmse = sqrt(mse)

r2 = r2_score(y_test, y_pred3)

print("Mean Squared Error: ", mse)

print("Root Mean Squared Error: ", rmse)

print("R2 Score: ", r2)
```

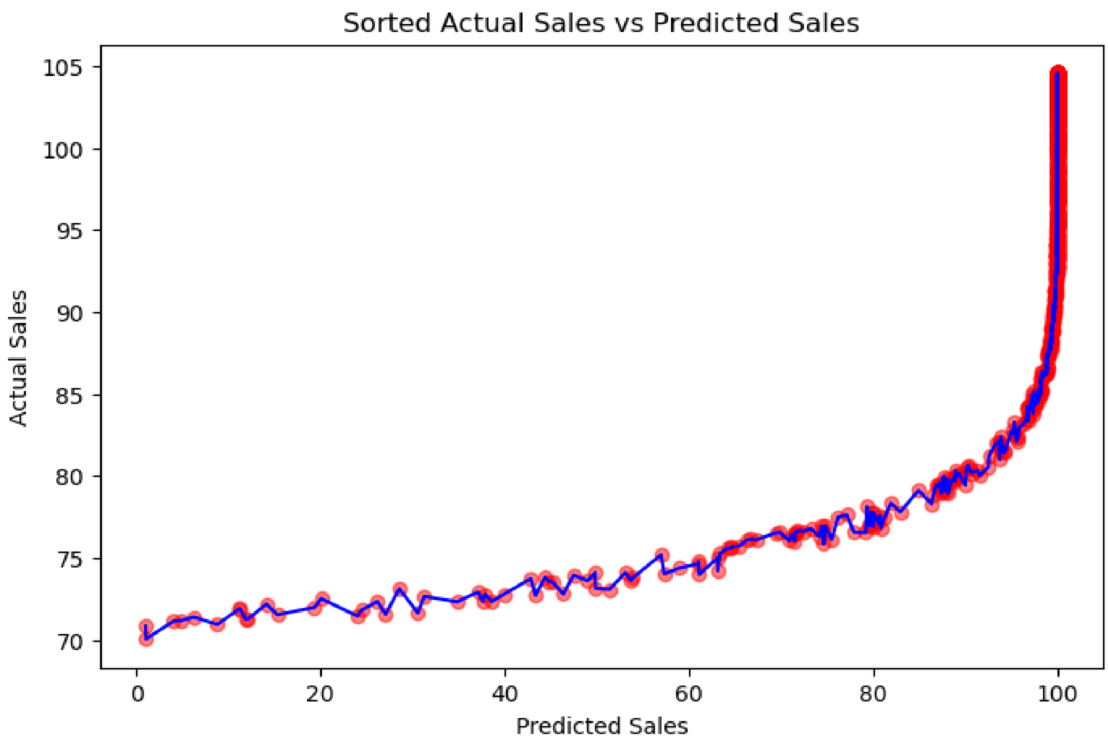
```
Mean Squared Error:  0.6158333511438949
Root Mean Squared Error:  0.7847505024808171
R2 Score:  0.9969259090983433
```

[129]:
```
plt.scatter(y_test, y_pred, color = 'blue')
plt.plot(y_test, y_pred, color = 'lightblue')
plt.title("Actual vs Predicted Quality using Polynomial Regression")
plt.xlabel("Actual Quality")
plt.ylabel('Predicted Quality')
plt.show()
```



[130]:

```
import numpy as np
import matplotlib.pyplot as plt
sorted_idx = np.argsort(y_test)
y_test_sorted = np.array(y_test)[sorted_idx]
y_pred_sorted = y_pred[sorted_idx]

# Scatter Plot (Actual vs Predicted)
plt.figure(figsize=(8, 5))
plt.scatter(y_test_sorted, y_pred_sorted, color = 'red', alpha=0.5)
plt.plot(y_test_sorted, y_pred_sorted, color = 'blue')
plt.title('Sorted Actual Sales vs Predicted Sales')
plt.xlabel('Predicted Sales')
plt.ylabel('Actual Sales')
plt.show()
```



Poly_4_Weight-Height

February 6, 2025

[166]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
import math
```

[167]:
```
url = r"Practice-2 Weight-Height Polynomial Dataset.csv"

df = pd.read_csv(url)

df.head()
```

[167]:

	Weight	Height
0	69.963210	96.644532
1	116.057145	196.156340
2	98.559515	145.862047
3	87.892679	121.157923
4	52.481491	68.971292

1 Data Preprocessing

Checking for null values

[168]:
```
df.isna().sum()
```

[168]:
```
Weight    0
Height    0
dtype: int64
```

Checking Statistics

[169]:
```
df.describe(include='all')
```

[169]:

	Weight	Height
count	50.000000	50.000000
mean	75.673912	111.473633
std	23.110656	39.493803
min	41.646760	68.971292
25%	54.701360	79.966731
50%	74.883900	98.819101
75%	91.988395	129.709758
max	117.592788	202.663424

Checking Info

[170]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Weight  50 non-null     float64
 1   Height  50 non-null     float64
dtypes: float64(2)
memory usage: 932.0 bytes
```

Checking Shape

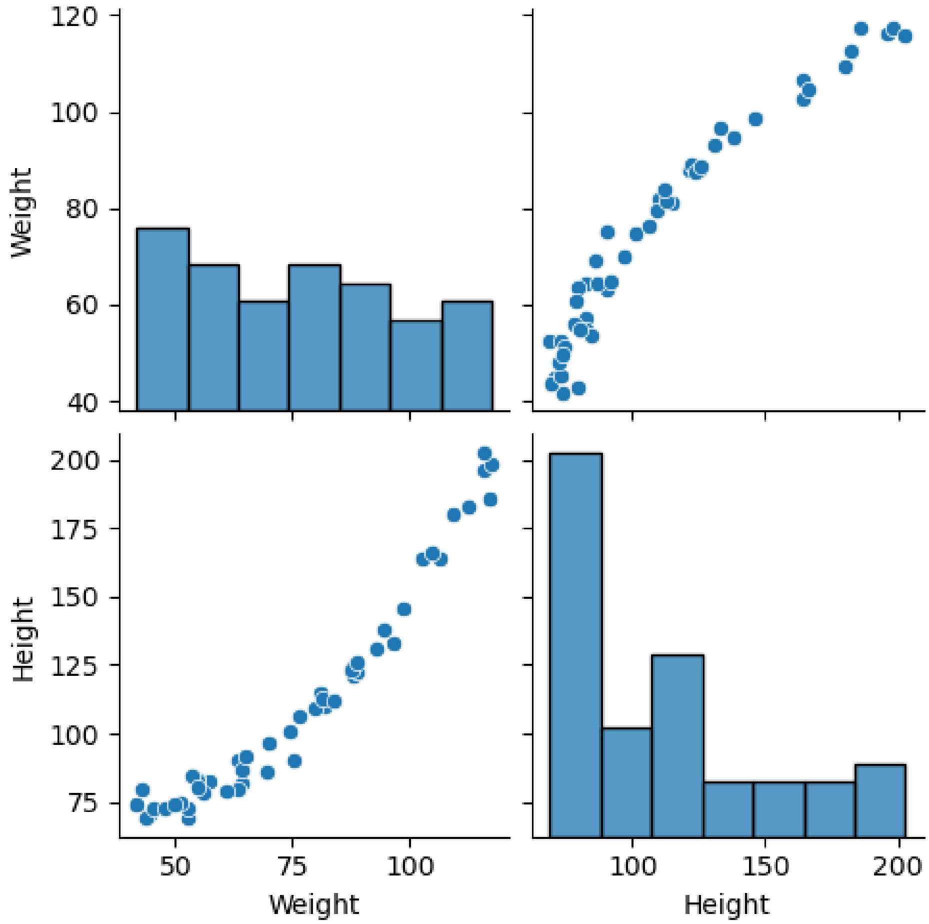
[171]:
```
df.shape
```

[171]: (50, 2)

Visualizing the Data

[172]:
```
sns.pairplot(df)
```

[172]: <seaborn.axisgrid.PairGrid at 0x252f1d9db50>



Scaling the data

[173]:
```
sca = StandardScaler()
X = df['Weight'].values.reshape(-1,1)
```

Creating X and y

[174]:
```
X = df.Weight.values
y = df.Height.values
```

Train Test Split

[175]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

Reshaping the data

[176]:
```
X_train = X_train.reshape(-1, 1)
X_test = X_test.reshape(-1, 1)
```

```
#y_train=y_train.reshape(-1,1)
#y_test=y_test.reshape(-1,1)
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
X_train shape: (40, 1)
X_test shape: (10, 1)
y_train shape: (40,)
y_test shape: (10,)
```

Model Creation

Simple Linear Regression

[177]:
```
linear = LinearRegression()
linear.fit(X_train.reshape(-1,1), y_train)
```

[177]:
```
LinearRegression()
```

Predcicting using Simple Linear Regression

[178]:
```
pred_linear = linear.predict(X_test.reshape(-1,1))
```

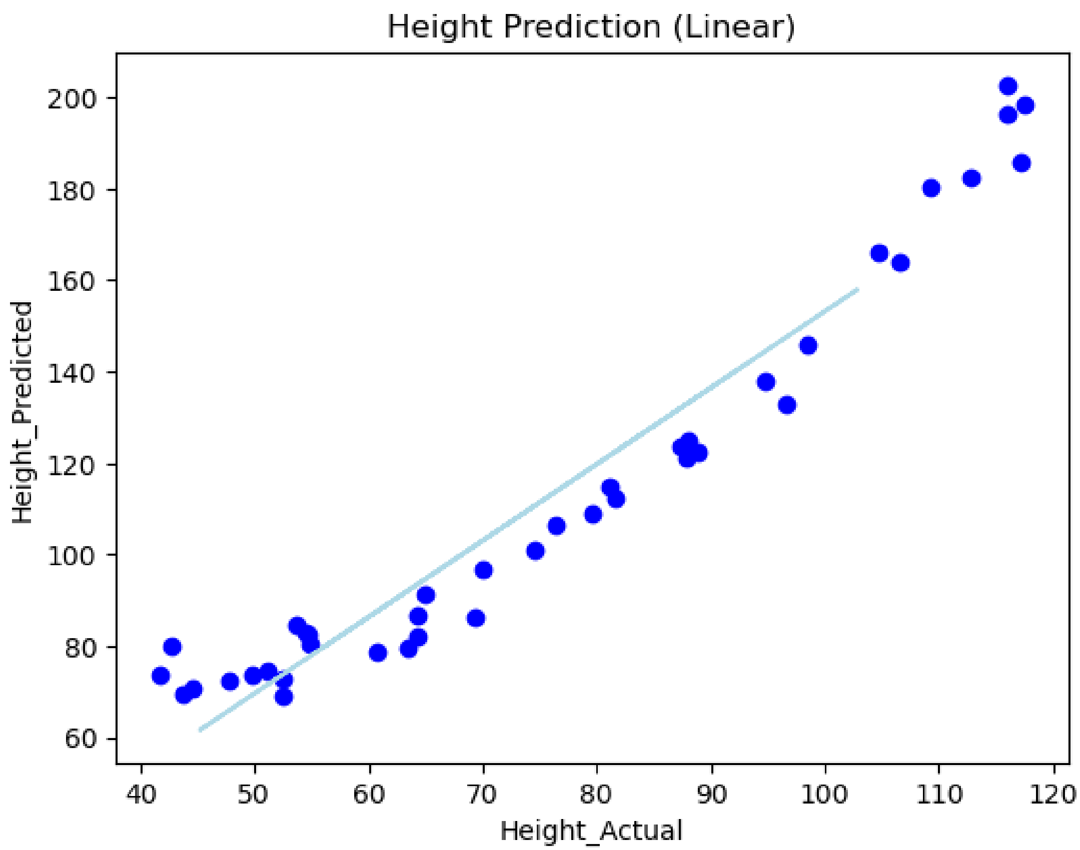
Evaluating Simple Linear Regression

[179]:
```
mse = mean_squared_error(y_test, pred_linear)
rmse = math.sqrt(mse)
r2 = r2_score(y_test, pred_linear)
mae = mean_absolute_error(y_test, pred_linear)

print('MSE:', mse)
print('RMSE:', rmse)
print('R2:', r2)
print('MAE:', mae)
```

```
MSE: 117.22192004841368
RMSE: 10.826907224522323
R2: 0.8403417139170934
MAE: 8.882234589447728
```

Visualizing Using Simple Linear Regression

[180]:
```
plt.scatter(X_train, y_train, color = 'blue')
plt.plot(X_test, pred_linear, color = 'lightblue')
plt.title("Height Prediction (Linear)")
plt.xlabel("Height_Actual")
plt.ylabel('Height_Predicted')
plt.show()
```



2 Using Polynomial Regression

[181]:
```
Poly = PolynomialFeatures(degree=2)
```

Transforming the data

[182]:
```
X_train_poly = Poly.fit_transform(X_train)
X_test_poly = Poly.fit_transform(X_test)
#y_train=y_train.reshape(-1,1)
#y_test=y_test.reshape(-1,1)
```

Training the Model

[183]:
```
lin = LinearRegression()

lin.fit(X_train_poly, y_train)
```

[183]:
```
LinearRegression()
```

Getting Predictions

[184]:
```
y_pred = lin.predict(X_test_poly)
```

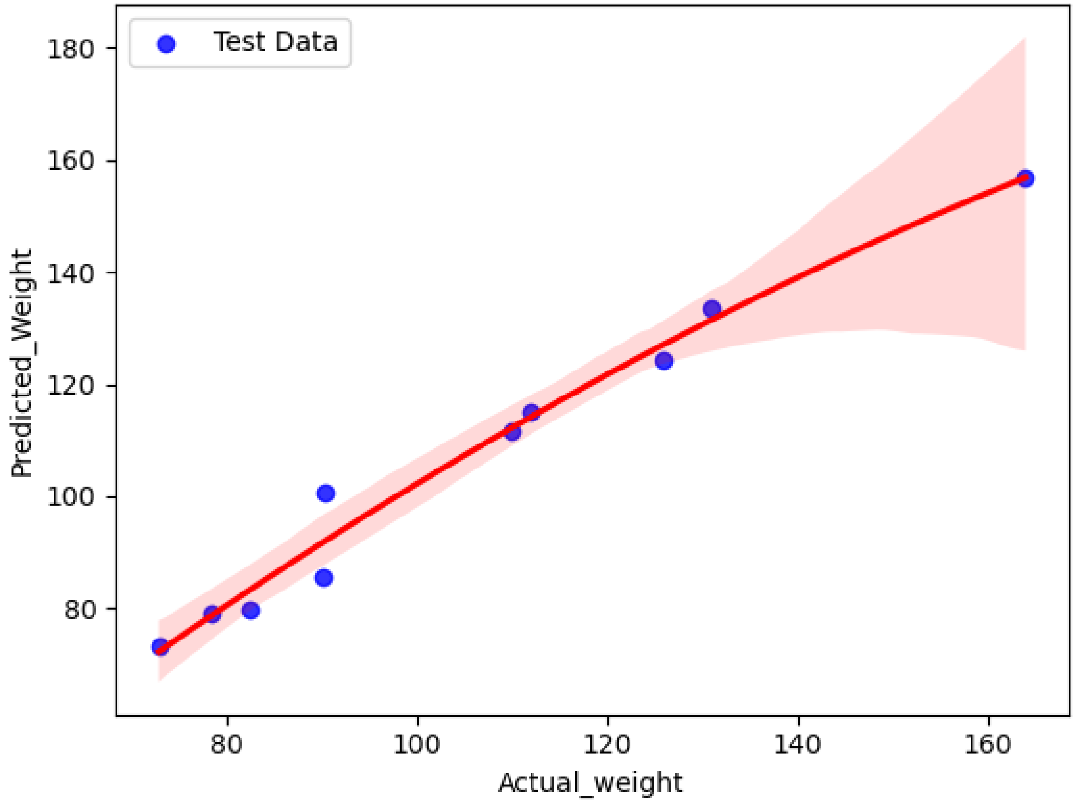
Evaluating the Model

[185]:
```
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test,y_pred)
rmse = math.sqrt(mse)

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

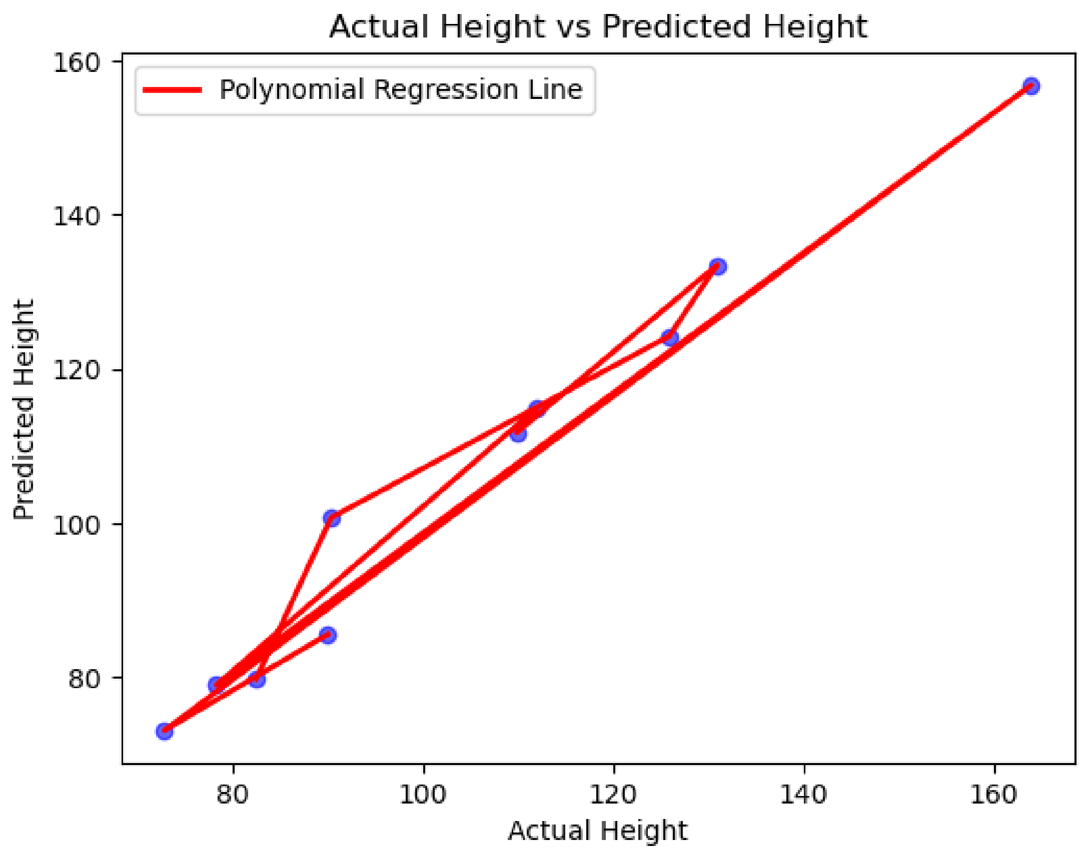
```
R2 Score: 0.9717532753949885
Mean Squared Error: 20.73888787431194
Mean Absolute Error: 3.461429854635168
Root Mean Squared Error: 4.553996911978745
```

Visualizing the Predicitons

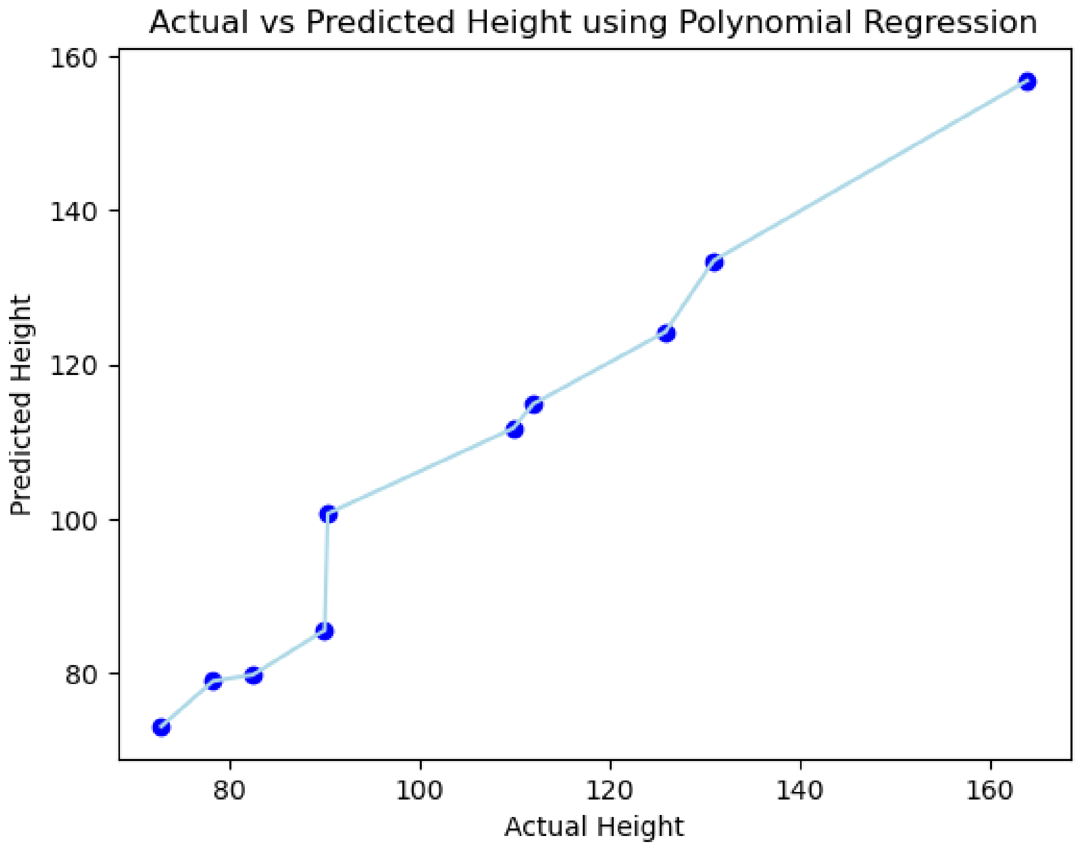
[186]:
```
sns.regplot(x=y_test, y=y_pred,  data=df, scatter_kws={"color": "blue"},
 ↪line_kws={"color": "red"}, order=2, label="Test Data")
plt.xlabel('Actual_weight')
plt.ylabel('Predicted_Weight')
plt.legend()
plt.show()
```




```
[187]: plt.scatter(y_test,y_pred, color='blue',alpha=0.6)
       plt.plot(y_test, y_pred, color='red', linewidth=2, label='Polynomial Regression␣
        ↪Line')
       plt.xlabel('Actual Height')
       plt.ylabel('Predicted Height')
       plt.title('Actual Height vs Predicted Height')
       plt.legend()
       plt.show()
```



[188]:
```
import numpy as np
sorted_idx = np.argsort(y_test)
Y_test_sorted = np.array(y_test)[sorted_idx]
y_pred_sorted = y_pred[sorted_idx]

plt.scatter(y_test, y_pred, color='blue')
plt.plot(Y_test_sorted, y_pred_sorted, color='lightblue')
plt.title("Actual vs Predicted Height using Polynomial Regression")
plt.xlabel("Actual Height")
plt.ylabel("Predicted Height")
plt.show()
```


Actual vs Predicted Height using Polynomial Regression
Predicted Height
160
140
120
100
80
Actual Height
80
100
120
140
160