Name: Zhaohu Xing; Student ID: 50008973
1.

Mean Squared Error: 60.43793205018627
R^2 Score: 0.7300272723725933

code is :

```
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

# Step 1: Load the data into a pandas DataFrame
data = """1   7.13  55.7  4.1   9.0   39.6  279  1  4  207  241  60.0
2   8.82  58.2  1.6   3.8   51.7   80  1  2   51   52  40.0
3   8.34  56.9  2.7   8.1   74.0  107  1  3   82   54  20.0
4   8.95  53.7  5.6  18.9  122.8  147  1  4   53  148  40.0
5  11.20  56.5  5.7  34.5   88.9  180  1  1  134  151  40.0
6   9.76  50.9  5.1  21.9   97.0  150  1  2  147  106  40.0
7   9.68  57.8  4.6  16.7   79.0  186  1  3  151  129  40.0
8  11.18  45.7  5.4  60.5   85.8  640  0  2  399  360  60.0
9   8.67  48.2  4.3  24.4   90.8  182  1  3  130  118  40.0
10   8.84  56.3  6.3  29.6   82.6   85  1  1   59   66  40.0
11  11.07  53.2  4.9  28.5  122.0  768  0  1  591  656  80.0
12   8.30  57.2  4.3   6.8   83.8  167  1  3  105   59  40.0
13  12.78  56.8  7.7  46.0  116.9  322  0  1  252  349  57.1
14   7.58  56.7  3.7  20.8   88.0   97  1  2   59   79  37.1
15   9.00  56.3  4.2  14.6   76.4   72  1  3   61   38  17.1
16  11.08  50.2  5.5  18.6   63.6  387  1  3  326  405  57.1
17   8.28  48.1  4.5  26.0  101.8  108  1  4   84   73  37.1
18  11.62  53.9  6.4  25.5   99.2  133  1  1  113  101  37.1
19   9.06  52.8  4.2   6.9   75.9  134  1  2  103  125  37.1
20   9.35  53.8  4.1  15.9   80.9  833  1  3  547  519  77.1
21   7.53  42.0  4.2  23.1   98.9   95  1  4   47   49  17.1
22  10.24  49.0  4.8  36.3  112.6  195  1  2  163  170  37.1
23   9.78  52.3  5.0  17.6   95.9  270  0  1  240  198  57.1
24   9.84  62.2  4.8  12.0   82.3  600  1  3  468  497  57.1
25   9.20  52.2  4.0  17.5   71.1  298  0  4  244  236  57.1
26   8.28  49.5  3.9  12.0  113.1  546  0  2  413  436  57.1
27   9.31  47.2  4.5  30.2  101.3  170  1  1  124  173  37.1
28   8.19  52.1  3.2  10.8   59.2  176  1  1  156   88  37.1
29  11.65  54.5  4.4  18.6   96.1  248  1  1  217  189  37.1
30   9.89  50.5  4.9  17.7  103.6  167  1  2  113  106  37.1
```

31 11.03 49.9 5.0 19.7 102.1 318 1 1 270 335 57.1
32 9.84 53.0 5.2 17.7 72.6 210 1 2 200 239 54.3
33 11.77 54.1 5.3 17.3 56.0 196 1 1 164 165 34.3
34 13.59 54.0 6.1 24.2 111.7 312 1 1 258 169 54.3
35 9.74 54.4 6.3 11.4 76.1 221 1 2 170 172 54.3
36 10.33 55.8 5.0 21.2 104.3 266 1 1 181 149 54.3
37 9.97 58.2 2.8 16.5 76.5 90 1 2 69 42 34.3
38 7.84 49.1 4.6 7.1 87.9 60 1 3 50 45 34.3
39 10.47 53.2 4.1 5.7 69.1 196 1 2 168 153 54.3
40 8.16 60.9 1.3 1.9 58.0 73 1 3 49 21 14.3
41 8.48 51.1 3.7 12.1 92.8 166 1 3 145 118 34.3
42 10.72 53.8 4.7 23.2 94.1 113 1 3 90 107 34.3
43 11.20 45.0 3.0 7.0 78.9 130 1 3 95 56 34.3
44 10.12 51.7 5.6 14.9 79.1 362 0 3 313 264 54.3
45 8.37 50.7 5.5 15.1 84.8 115 1 2 96 88 34.3
46 10.16 54.2 4.6 8.4 51.5 831 0 4 581 629 74.3
47 19.56 59.9 6.5 17.2 113.7 306 1 1 273 172 51.4
48 10.90 57.2 5.5 10.6 71.9 593 1 2 446 211 51.4
49 7.67 51.7 1.8 2.5 40.4 106 1 3 93 35 11.4
50 8.88 51.5 4.2 10.1 86.9 305 1 3 238 197 51.4
51 11.48 57.6 5.6 20.3 82.0 252 1 1 207 251 51.4
52 9.23 51.6 4.3 11.6 42.6 620 1 2 413 420 71.4
53 11.41 61.1 7.6 16.6 97.9 535 1 3 330 273 51.4
54 12.07 43.7 7.8 52.4 105.3 157 1 2 115 76 31.4
55 8.63 54.0 3.1 8.4 56.2 76 1 1 39 44 31.4
56 11.15 56.5 3.9 7.7 73.9 281 1 1 217 199 51.4
57 7.14 59.0 3.7 2.6 75.8 70 1 4 37 35 31.4
58 7.65 47.1 4.3 16.4 65.7 318 1 4 265 314 51.4
59 10.73 50.6 3.9 19.3 101.0 445 0 2 374 345 51.4
60 11.46 56.9 4.5 15.6 97.7 191 1 3 153 132 31.4
61 10.42 58.0 3.4 8.0 59.0 119 1 1 67 64 31.4
62 11.18 51.0 5.7 18.8 55.9 595 0 2 546 392 68.6
63 7.93 64.1 5.4 7.5 98.1 68 1 4 42 49 28.6
64 9.66 52.1 4.4 9.9 98.3 83 1 2 66 95 28.6
65 7.78 45.5 5.0 20.9 71.6 489 1 3 391 329 48.6
66 9.42 50.6 4.3 24.8 62.8 508 1 1 421 528 48.6
67 10.02 49.5 4.4 8.3 93.0 265 1 2 191 202 48.6
68 8.58 55.0 3.7 7.4 95.9 304 1 3 248 218 48.6
69 9.61 52.4 4.5 6.9 87.2 487 1 3 404 220 48.6
70 8.03 54.2 3.5 24.3 87.3 97 1 1 65 55 28.6
71 7.39 51.0 4.2 14.6 88.4 72 1 2 38 67 28.6
72 7.08 52.0 2.0 12.3 56.4 87 1 3 52 57 28.6
73 9.53 51.5 5.2 15.0 65.7 298 1 3 241 193 48.6
74 10.05 52.0 4.5 36.7 87.5 184 0 1 144 151 68.6

75  8.45  38.8  3.4  12.9  85.0  235  1  2  143  124  48.6
76  6.70  48.6  4.5  13.0  80.8  76  1  4  51  79  28.6
77  8.90  49.7  2.9  12.7  86.9  52  1  1  37  35  28.6
78  10.23  53.2  4.9  9.9  77.9  752  0  2  595  446  68.6
79  8.88  55.8  4.4  14.1  76.8  237  1  2  165  182  48.6
80  10.30  59.6  5.1  27.8  88.9  175  1  2  113  73  45.7
81  10.79  44.2  2.9  2.6  56.6  461  0  2  320  196  65.7
82  7.94  49.5  3.5  6.2  92.3  195  1  2  139  116  45.7
83  7.63  52.1  5.5  11.6  61.1  197  1  4  109  110  45.7
84  8.77  54.5  4.7  5.2  47.0  143  1  4  85  87  25.7
85  8.09  56.9  1.7  7.6  56.9  92  1  3  61  61  45.7
86  9.05  51.2  4.1  20.5  79.8  195  1  3  127  112  45.7
87  7.91  52.8  2.9  11.9  79.5  477  1  3  349  188  65.7
88  10.39  54.6  4.3  14.0  88.3  353  1  2  223  200  65.7
89  9.36  54.1  4.8  18.3  90.6  165  1  1  127  158  45.7
90  11.41  50.4  5.8  23.8  73.0  424  0  3  359  335  45.7
91  8.86  51.3  2.9  9.5  87.5  100  1  3  65  53  25.7
92  8.93  56.0  2.0  6.2  72.5  95  1  3  59  56  25.7
93  8.92  53.9  1.3  2.2  79.5  56  1  2  40  14  5.7
94  8.15  54.9  5.3  12.3  79.8  99  1  4  55  71  25.7
95  9.77  50.2  5.3  15.7  89.7  154  1  2  123  148  25.7
96  8.54  56.1  2.5  27.0  82.5  98  1  1  57  75  45.7
97  8.66  52.8  3.8  6.8  69.5  246  1  3  178  177  45.7
98  12.01  52.8  4.8  10.8  96.9  298  1  1  237  115  45.7
99  7.95  51.8  2.3  4.6  54.9  163  1  3  128  93  42.9
100  10.15  51.9  6.2  16.4  59.2  568  0  3  452  371  62.9
101  9.76  53.2  2.6  6.9  80.1  64  1  4  47  55  22.9
102  9.89  45.2  4.3  11.8  108.7  190  1  1  141  112  42.9
103  7.14  57.6  2.7  13.1  92.6  92  1  4  40  50  22.9
104  13.95  65.9  6.6  15.6  133.5  356  1  1  308  182  62.9
105  9.44  52.5  4.5  10.9  58.5  297  1  3  230  263  42.9
106  10.80  63.9  2.9  1.6  57.4  130  1  3  69  62  22.9
107  7.14  51.7  1.4  4.1  45.7  115  1  3  90  19  22.9
108  8.02  55.0  2.1  3.8  46.5  91  1  2  44  32  22.9
109  11.80  53.8  5.7  9.1  116.9  571  0  2  441  469  62.9
110  9.50  49.3  5.8  42.0  70.9  98  1  3  68  46  22.9
111  7.70  56.9  4.4  12.2  67.9  129  1  4  85  136  62.9
112  17.94  56.2  5.9  26.4  91.8  835  0  1  791  407  62.9
113  9.41  59.5  3.1  20.6  91.7  29  1  3  20  22  22.9"""

*# Convert the string data to a pandas DataFrame*
from io import StringIO

data = StringIO(data)

```
df = pd.read_csv(data, sep="\s+", header=None)

# Step 2: Split the data into features (X) and target variable (y)
X = df.iloc[:, 1:-1]  # all columns except the first and last
y = df.iloc[:, -1]    # the last column is the target variable

# Step 3: Split the dataset 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)

# Step 4: Train a Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Step 5: Make predictions using the test set
y_pred = model.predict(X_test)

# Step 6: 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"R^2 Score: {r2}")
```

2.

**Hasse Diagram:**

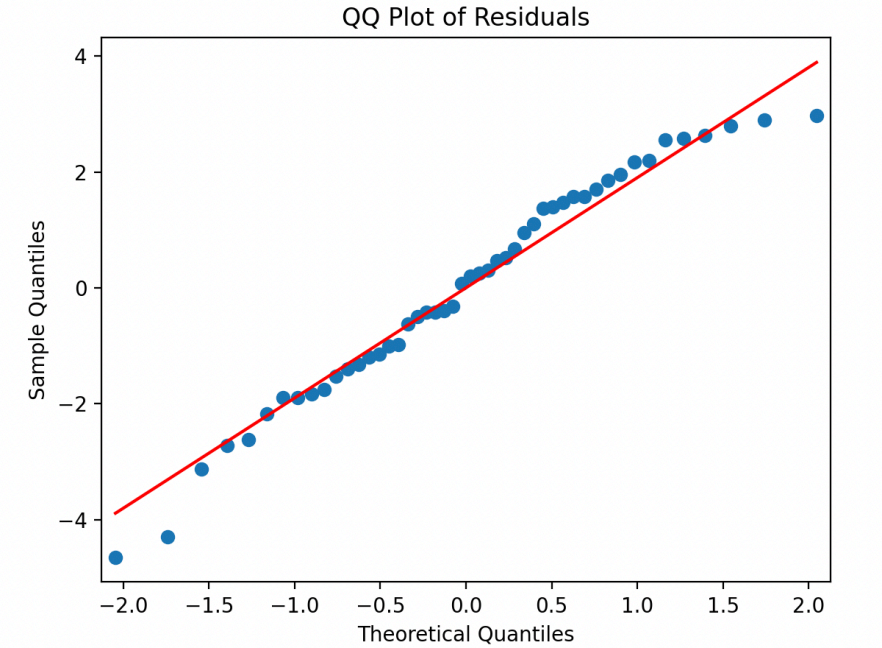
Total

├— Factor A (Coats)

├— Factor B (Batch)

├— Interaction (A x B)

└— Error




```
Shapiro-Wilk Test: ShapiroResult(statistic=0.9680246114730835, pvalue=0.21193252503871918)
Levene's Test: LeveneResult(statistic=0.15220050867647705, pvalue=0.8592562918714701)
```

code is :

```
import pandas as pd

# Data
data = {
   'Value': [72.0, 74.6, 67.4, 72.8, 72.1, 76.9, 74.8, 73.3, 75.2, 73.8, 75.7, 77.8, 70.4, 68.1, 72.4,
72.4,
          76.9, 78.1, 72.9, 74.2, 80.3, 79.3, 76.6, 77.2, 80.2, 76.6, 77.3, 79.9, 74.3, 77.6, 74.4, 72.9,
          76.3, 74.1, 77.1, 75.0, 80.9, 73.7, 78.6, 80.2, 79.2, 78.0, 77.6, 81.2, 71.6, 77.7, 75.2,
74.4],
   'Coats': [1]*16 + [2]*16 + [3]*16,
   'Batch': [1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4] * 3,
   'Bead': list(range(1, 5)) * 12
}

df = pd.DataFrame(data)

import statsmodels.api as sm
from statsmodels.formula.api import ols
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# Perform ANOVA
```

```
model = ols('Value ~ C(Coats) + C(Batch) + C(Coats):C(Batch)', data=df).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
print(anova_table)

# QQ plot
sm.qqplot(model.resid, line='s')
plt.title('QQ Plot of Residuals')
plt.show()

# Shapiro-Wilk test
shapiro_test = stats.shapiro(model.resid)
print('Shapiro-Wilk Test:', shapiro_test)

# Levene's test
levene_test = stats.levene(*[group["Value"].values for name, group in df.groupby('Coats')])
print('Levene\'s Test:', levene_test)
```

3.

The student's statement contains a common misconception about regression models and R-squared. It is true that adding additional predictor variables to a regression model will not reduce the R-squared value; in fact, R-squared will either increase or stay the same. This is because R-squared represents the proportion of variance in the dependent variable that is explained by the independent variables in the model, and adding more variables can only increase or leave unchanged this proportion.

However, this does not mean that you should include all available predictor variables in the model. There are several reasons why including all predictors may not be a good idea:

1. **Overfitting**: Including all available predictors can lead to overfitting, where the model captures the noise in the data rather than the underlying pattern. This can result in a model that performs well on the training data but poorly on new, unseen data.
2. **Multicollinearity**: Including too many predictors can introduce multicollinearity, where two or more predictors are highly correlated with each other. This can make it difficult to determine the individual effect of each predictor and can lead to inflated standard errors.
3. **Interpretability**: Models with too many predictors can become difficult to interpret, especially if the relationships between predictors and the outcome are complex or if there are interactions between predictors.
4. **Parsimony**: In general, simpler models are preferred over more complex ones if they achieve similar performance. A model with fewer predictors is often more interpretable and generalizes better to new data.
5. **Computational Efficiency**: Including a large number of predictor variables can increase the computational cost and complexity of fitting the model, especially with very large datasets or complex algorithms.

Instead of including all predictor variables, it's often better to use techniques like stepwise selection, regularization methods (such as LASSO or Ridge regression), or domain knowledge to select a subset of meaningful predictors that balance model complexity and predictive power.