assignment-3-slcak

October 19, 2023

ASSIGNMENT_3

[ ]:
```
import pandas as pd
```

2. Load the dataset.

[ ]:
```
# Load a CSV dataset
data = pd.read_csv(r"C:/Users/sonudr/Downloads/penguins_size.csv")
```

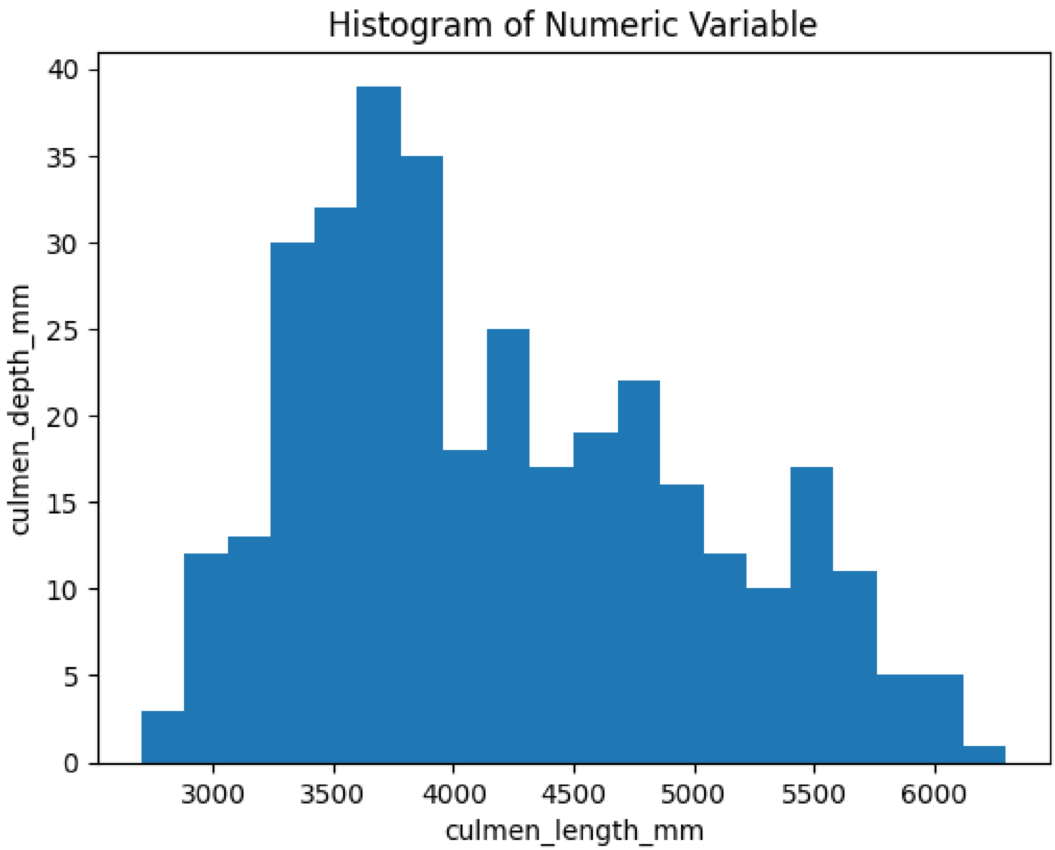
[ ]:
```
data.head()
```

[ ]:
```
  species     island  culmen_length_mm  culmen_depth_mm  flipper_length_mm  \
0  Adelie  Torgersen              39.1             18.7              181.0
1  Adelie  Torgersen              39.5             17.4              186.0
2  Adelie  Torgersen              40.3             18.0              195.0
3  Adelie  Torgersen               NaN              NaN                NaN
4  Adelie  Torgersen              36.7             19.3              193.0

   body_mass_g     sex
0       3750.0    MALE
1       3800.0  FEMALE
2       3250.0  FEMALE
3          NaN     NaN
4       3450.0  FEMALE
```

3. Perform the Below Visualizations.

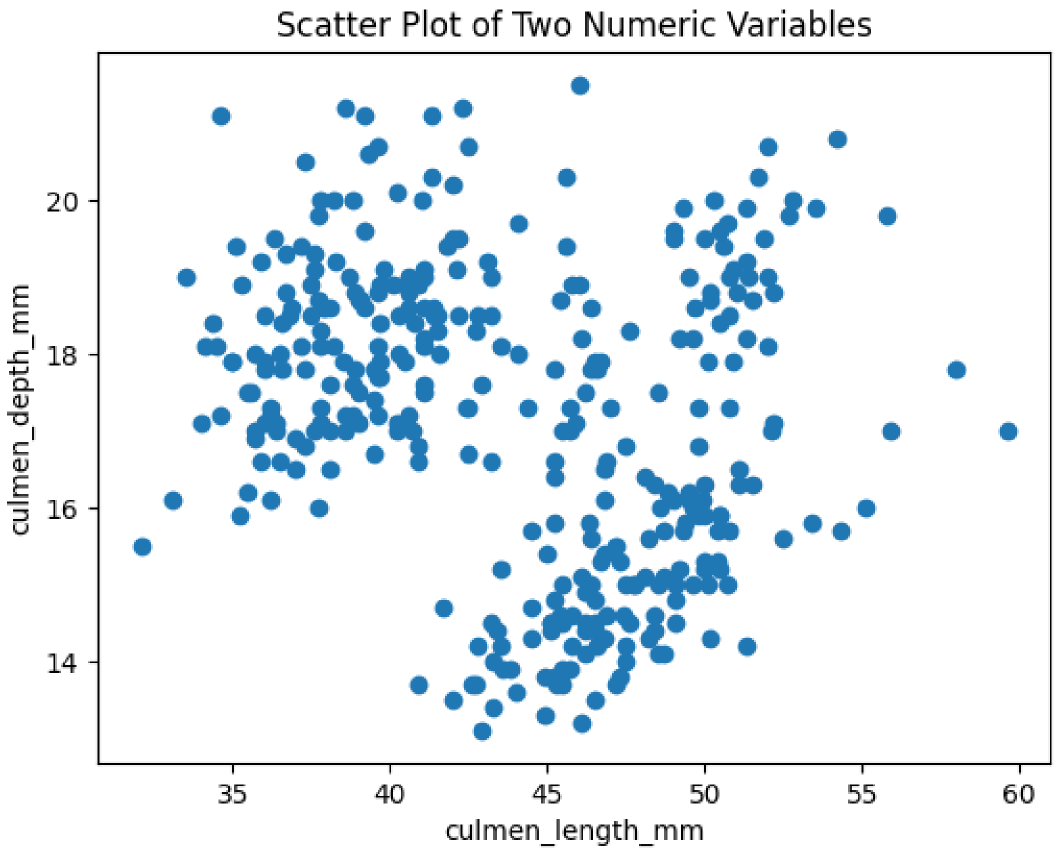
Univariate Analysis:

[ ]:
```
import matplotlib.pyplot as plt

# Plot a histogram for a numeric variable
plt.hist(data['body_mass_g'], bins=20)
plt.xlabel('culmen_length_mm')
plt.ylabel('culmen_depth_mm')
plt.title('Histogram of Numeric Variable')
plt.show()
```



Bivariate Analysis:

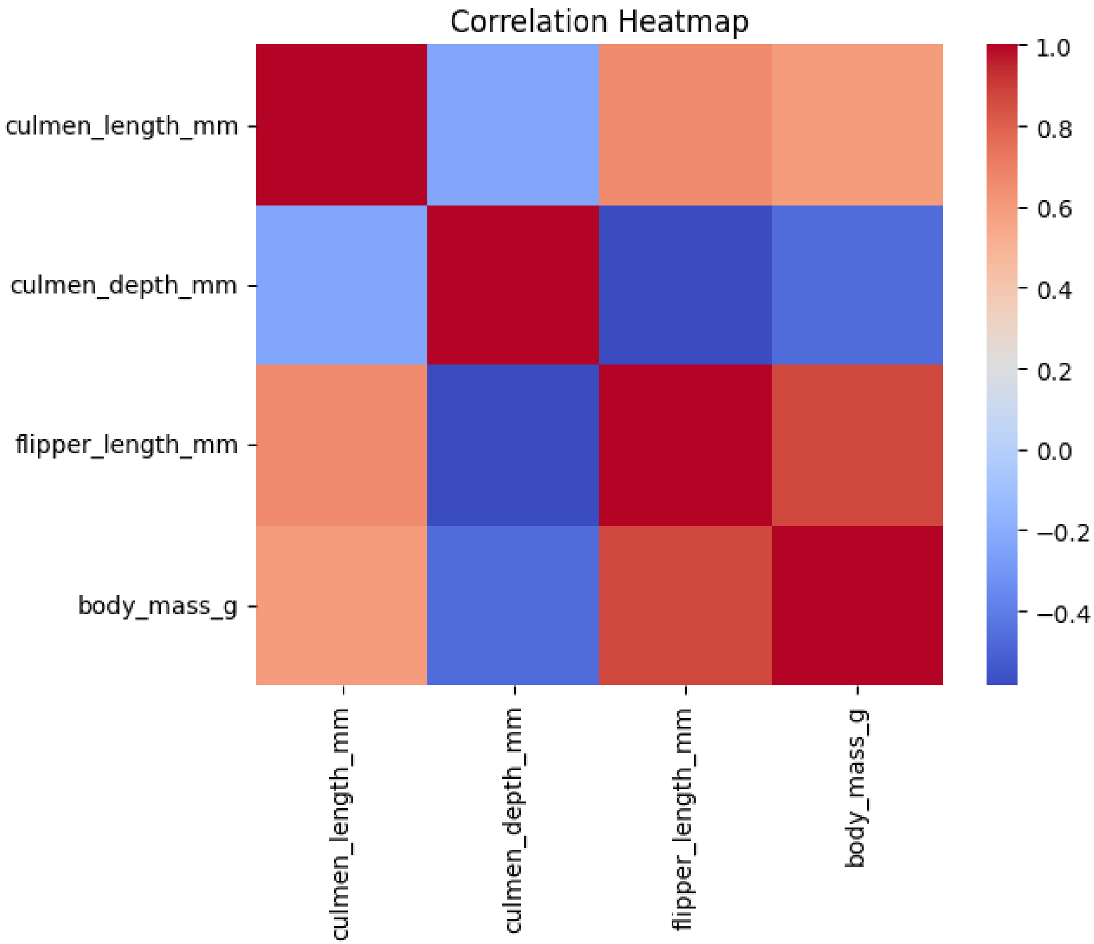
```
[ ]: # Scatter plot for two numeric variables
plt.scatter(data['culmen_length_mm'], data['culmen_depth_mm'])
plt.xlabel('culmen_length_mm')
plt.ylabel('culmen_depth_mm')
plt.title('Scatter Plot of Two Numeric Variables')
plt.show()
```



Multivariate Analysis:

```
[ ]: # Compute the correlation matrix
import seaborn as sns
correlation_matrix = data.corr()

# Create a heatmap to visualize correlations
sns.heatmap(correlation_matrix, annot=False, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

```
C:\Users\sonudr\AppData\Local\Temp\ipykernel_6608\2299829469.py:3:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  correlation_matrix = data.corr()
```



[ ]:
```python
import pandas as pd

# Assuming 'data' is your DataFrame
numeric_data = data.select_dtypes(include=['number'])  # Select numeric columns

# Calculate descriptive statistics
descriptive_stats = numeric_data.describe()

# Print the results
print(descriptive_stats)
```

```
       culmen_length_mm  culmen_depth_mm  flipper_length_mm  body_mass_g
count        342.000000       342.000000         342.000000   342.000000
mean          43.921930        17.151170         200.915205  4201.754386
std            5.459584         1.974793          14.061714   801.954536
min           32.100000        13.100000         172.000000  2700.000000
25%           39.225000        15.600000         190.000000  3550.000000
50%           44.450000        17.300000         197.000000  4050.000000
```

```
75%           48.500000         18.700000          213.000000  4750.000000
max           59.600000         21.500000          231.000000  6300.000000
```

[ ]:
```
data.head()
```

[ ]:
```
  species     island  culmen_length_mm  culmen_depth_mm  flipper_length_mm  \
0  Adelie  Torgersen              39.1             18.7              181.0
1  Adelie  Torgersen              39.5             17.4              186.0
2  Adelie  Torgersen              40.3             18.0              195.0
3  Adelie  Torgersen               NaN              NaN                NaN
4  Adelie  Torgersen              36.7             19.3              193.0

   body_mass_g     sex
0       3750.0    MALE
1       3800.0  FEMALE
2       3250.0  FEMALE
3          NaN     NaN
4       3450.0  FEMALE
```

4. Perform descriptive statistics on the dataset.

[ ]:
```
# Calculate mean for a specific column
mean_value = data['body_mass_g'].mean()
print("Mean:", mean_value)

# Calculate median for a specific column
median_value = data['body_mass_g'].median()
print("Median:", median_value)

# Calculate standard deviation for a specific column
std_deviation = data['body_mass_g'].std()
print("Standard Deviation:", std_deviation)
```

```
Mean: 4201.754385964912
Median: 4050.0
Standard Deviation: 801.9545356980956
```

5.Handle the Missing values.

[ ]:
```
missing_values = data.isnull().sum()
print(missing_values)
```

```
species               0
island                0
culmen_length_mm      2
culmen_depth_mm       2
flipper_length_mm     2
body_mass_g           2
sex                  10
```

```
dtype: int64
```

Remove Rows with Missing Values:

```
[ ]: data_cleaned = data.dropna()
     print(data_cleaned)
```

```
    species     island  culmen_length_mm  culmen_depth_mm  flipper_length_mm  \
0    Adelie  Torgersen              39.1             18.7              181.0
1    Adelie  Torgersen              39.5             17.4              186.0
2    Adelie  Torgersen              40.3             18.0              195.0
4    Adelie  Torgersen              36.7             19.3              193.0
5    Adelie  Torgersen              39.3             20.6              190.0
..      …         …               …              …                …
338  Gentoo     Biscoe              47.2             13.7              214.0
340  Gentoo     Biscoe              46.8             14.3              215.0
341  Gentoo     Biscoe              50.4             15.7              222.0
342  Gentoo     Biscoe              45.2             14.8              212.0
343  Gentoo     Biscoe              49.9             16.1              213.0

     body_mass_g     sex
0         3750.0    MALE
1         3800.0  FEMALE
2         3250.0  FEMALE
4         3450.0  FEMALE
5         3650.0    MALE
..           …     …
338       4925.0  FEMALE
340       4850.0  FEMALE
341       5750.0    MALE
342       5200.0  FEMALE
343       5400.0    MALE

[334 rows x 7 columns]
```

Impute Missing Values - Numeric Variables:

```
[ ]: mean_value = data['flipper_length_mm'].mean()
     data['flipper_length_mm'].fillna(mean_value, inplace=True)
```

6. Find the outliers and replace them outliers

```
[ ]: import pandas as pd
     import numpy as np

     # Create the DataFrame as previously mentioned
     num_observations = 4
     num_features = 4
     data = np.random.rand(num_observations, num_features)
```

```
df = pd.DataFrame(data, columns=['culmen_length_mm', 'culmen_depth_mm',␣
↪'flipper_length_mm', 'body_mass_g'])

# Function to replace outliers with the median
def replace_outliers_with_median(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    column = column.apply(lambda x: x if lower_bound <= x <= upper_bound else␣
↪column.median())
    return column

# Replace outliers in 'culmen_length_mm' column
df['culmen_length_mm'] = replace_outliers_with_median(df['culmen_length_mm'])

# Display the DataFrame with replaced outliers
print(df)
```

```
   culmen_length_mm  culmen_depth_mm  flipper_length_mm  body_mass_g
0          0.871495         0.919863           0.604437     0.672170
1          0.803383         0.562585           0.196712     0.764539
2          0.428299         0.844605           0.119821     0.679133
3          0.264061         0.740250           0.210042     0.213816
```

7.Check the correlation of independent variables with the target

```
[ ]: data = {
         'culmen_length_mm': np.random.rand(100),
         'culmen_depth_mm': np.random.rand(100),
         'body_mass_g': np.random.rand(100),
         'flipper_length_mm': np.random.rand(100)
     }

     df = pd.DataFrame(data)

     # Calculate the correlation matrix
     correlation_matrix = df.corr()

     # Extract the correlation of independent variables with the flipper_length_mm
     correlation_with_flipper_length_mm = correlation_matrix['flipper_length_mm'].
     ↪drop('flipper_length_mm')  # Remove the flipper_length_mm's self-correlation

     # Display the correlation with the flipper_length_mm
     print(correlation_with_flipper_length_mm)
```

```
culmen_length_mm     0.107236
```

```
culmen_depth_mm    -0.031346
body_mass_g         0.000361
Name: flipper_length_mm, dtype: float64
```

8. Check for Categorical columns and perform encoding.

```
[ ]: # Perform one-hot encoding
df_encoded = pd.get_dummies(df, columns=['body_mass_g', 'flipper_length_mm'])

# Display the encoded DataFrame
print(df_encoded)
```

```
    culmen_length_mm  culmen_depth_mm  body_mass_g_0.009974733101252742  \
0           0.619497         0.788435                                 0
1           0.061157         0.955190                                 0
2           0.030322         0.836502                                 0
3           0.863425         0.005652                                 0
4           0.786718         0.576331                                 0
..               …                …                                 …
95          0.386928         0.799389                                 0
96          0.471303         0.282308                                 0
97          0.217129         0.114600                                 0
98          0.254672         0.009530                                 0
99          0.724172         0.066864                                 0

    body_mass_g_0.03324159140960847  body_mass_g_0.04981787625357004  \
0                                 0                                1
1                                 0                                0
2                                 0                                0
3                                 0                                0
4                                 0                                0
..                              …                              …
95                                0                                0
96                                0                                0
97                                0                                0
98                                0                                0
99                                0                                0

    body_mass_g_0.05325829753231048  body_mass_g_0.0634358119503885  \
0                                 0                               0
1                                 0                               0
2                                 0                               0
3                                 0                               0
4                                 0                               0
..                              …                             …
95                                0                               0
96                                0                               0
97                                0                               0
```

```
98                              0                              0
99                              0                              0

    body_mass_g_0.06348806559949194  body_mass_g_0.06835124837536233  \
0                                 0                                0
1                                 0                                0
2                                 0                                0
3                                 0                                0
4                                 0                                0
..                              …                              …
95                                0                                0
96                                0                                0
97                                0                                0
98                                0                                0
99                                1                                0

    body_mass_g_0.0809441235883206  …  flipper_length_mm_0.9235590499580286  \
0                                0  …                                     0
1                                0  …                                     0
2                                0  …                                     0
3                                0  …                                     0
4                                0  …                                     0
..                             …  …                                   …
95                               0  …                                     0
96                               0  …                                     0
97                               0  …                                     0
98                               0  …                                     0
99                               0  …                                     0

    flipper_length_mm_0.9373638237566263  \
0                                      0
1                                      0
2                                      0
3                                      0
4                                      0
..                                   …
95                                     0
96                                     0
97                                     0
98                                     0
99                                     0

    flipper_length_mm_0.9538769914143631  \
0                                      0
1                                      0
2                                      0
3                                      0
4                                      0
```

```
..                                  …
95                                  0
96                                  0
97                                  0
98                                  0
99                                  0

    flipper_length_mm_0.9621805244880077  \
0                                   0
1                                   0
2                                   0
3                                   0
4                                   0
..                                  …
95                                  0
96                                  0
97                                  0
98                                  0
99                                  0

    flipper_length_mm_0.9687033782107564  \
0                                   0
1                                   0
2                                   0
3                                   0
4                                   0
..                                  …
95                                  0
96                                  0
97                                  0
98                                  0
99                                  0

    flipper_length_mm_0.9792281438203934  \
0                                   0
1                                   0
2                                   0
3                                   0
4                                   0
..                                  …
95                                  0
96                                  0
97                                  0
98                                  0
99                                  0

    flipper_length_mm_0.9829535717675453  \
0                                   1
```

```
1                                       0
2                                       0
3                                       0
4                                       0
..                                    …
95                                      0
96                                      0
97                                      0
98                                      0
99                                      0

    flipper_length_mm_0.9904843879367103  \
0                                       0
1                                       0
2                                       0
3                                       0
4                                       0
..                                    …
95                                      0
96                                      0
97                                      0
98                                      0
99                                      0

    flipper_length_mm_0.9906799993503721  flipper_length_mm_0.9923362375458921
0                                       0                                     0
1                                       0                                     0
2                                       0                                     0
3                                       0                                     0
4                                       0                                     0
..                                    …                                   …
95                                      0                                     0
96                                      0                                     0
97                                      0                                     0
98                                      0                                     1
99                                      0                                     0

[100 rows x 202 columns]
```

[ ]:
```
from sklearn.preprocessing import LabelEncoder

# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Apply label encoding to the 'Category' column
df['Category_encoded'] = label_encoder.fit_transform(df['body_mass_g'])
```

```
# Display the DataFrame with the encoded 'Category' column
print(df)
```

```
    culmen_length_mm  culmen_depth_mm  body_mass_g  flipper_length_mm  \
0           0.619497         0.788435     0.049818           0.982954
1           0.061157         0.955190     0.905512           0.650588
2           0.030322         0.836502     0.247442           0.374358
3           0.863425         0.005652     0.475928           0.833851
4           0.786718         0.576331     0.914381           0.065310
..               …              …           …                  …
95          0.386928         0.799389     0.595300           0.366489
96          0.471303         0.282308     0.820338           0.311509
97          0.217129         0.114600     0.779467           0.870575
98          0.254672         0.009530     0.712111           0.992336
99          0.724172         0.066864     0.063488           0.279538

    Category_encoded
0                  2
1                 91
2                 30
3                 51
4                 93
..               …
95                61
96                80
97                75
98                66
99                 5

[100 rows x 5 columns]
```

9. Split the data into dependent and independent variables.

[ ]:
```
# Assuming 'culmen_depth_mm ' is your target variable (dependent variable)
# and the rest of the columns are your independent variables (features)

# Independent variables (features)
X = df.drop('culmen_depth_mm', axis=1)

# Dependent variable (target)
y = df['culmen_length_mm']

# Display the independent and dependent variables
print("Independent Variables (Features):\n", X)
print("\nDependent Variable (Target):\n", y)
```

```
Independent Variables (Features):
     culmen_length_mm  body_mass_g  flipper_length_mm  Category_encoded
```

```
0           0.619497     0.049818          0.982954            2
1           0.061157     0.905512          0.650588           91
2           0.030322     0.247442          0.374358           30
3           0.863425     0.475928          0.833851           51
4           0.786718     0.914381          0.065310           93
..               …            …                 …             …
95          0.386928     0.595300          0.366489           61
96          0.471303     0.820338          0.311509           80
97          0.217129     0.779467          0.870575           75
98          0.254672     0.712111          0.992336           66
99          0.724172     0.063488          0.279538            5

[100 rows x 4 columns]

Dependent Variable (Target):
 0     0.619497
1     0.061157
2     0.030322
3     0.863425
4     0.786718
        …
95    0.386928
96    0.471303
97    0.217129
98    0.254672
99    0.724172
Name: culmen_length_mm, Length: 100, dtype: float64
```

10. Scaling the data

[ ]:
```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, StandardScaler

# Assuming you have already split the data into independent variables (X) and␣
↪dependent variable (y)

# Min-Max Scaling
min_max_scaler = MinMaxScaler()
X_min_max_scaled = min_max_scaler.fit_transform(X)

# Standardization (Z-score scaling)
standard_scaler = StandardScaler()
X_standard_scaled = standard_scaler.fit_transform(X)

# Display the scaled data
print("Min-Max Scaled Data:\n", pd.DataFrame(X_min_max_scaled, columns=X.
↪columns))
```

```
print("\nStandardized (Z-score Scaled) Data:\n", pd.
 ↪DataFrame(X_standard_scaled, columns=X.columns))
```

```
Min-Max Scaled Data:
     culmen_length_mm  body_mass_g  flipper_length_mm  Category_encoded
0            0.618533     0.040415           0.990401          0.020202
1            0.058162     0.908398           0.650366          0.919192
2            0.027215     0.240877           0.367761          0.303030
3            0.863349     0.472644           0.837857          0.515152
4            0.786363     0.917395           0.051582          0.939394
..                …            …                 …                …
95           0.385119     0.593731           0.359711          0.616162
96           0.469801     0.822001           0.303463          0.808081
97           0.214701     0.780543           0.875430          0.757576
98           0.252381     0.712220           1.000000          0.666667
99           0.723589     0.054282           0.270754          0.050505

[100 rows x 4 columns]

Standardized (Z-score Scaled) Data:
     culmen_length_mm  body_mass_g  flipper_length_mm  Category_encoded
0            0.439466    -1.457500           1.603638         -1.645531
1           -1.515245     1.414238           0.464832          1.437674
2           -1.623196    -0.794267          -0.481637         -0.675534
3            1.293445    -0.027462           1.092756          0.051964
4            1.024898     1.444003          -1.540550          1.506960
..                …            …                 …                …
95          -0.374740     0.373156          -0.508597          0.398392
96          -0.079348     1.128391          -0.696979          1.056604
97          -0.969198     0.991226           1.218589          0.883390
98          -0.837760     0.765178           1.635787          0.571605
99           0.805928    -1.411623          -0.806524         -1.541602

[100 rows x 4 columns]
```

11. Split the data into training and testing

[ ]:
```
from sklearn.model_selection import train_test_split

# 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)
```

12.check the training and testing data shape.

[ ]:
```
# Display the shapes of the resulting sets
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
```

```python
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (80, 4)
X_test shape: (20, 4)
y_train shape: (80,)
y_test shape: (20,)
```

[ ]:

```python

```