Name: - Shresthi Singh

Reg. NO.: - 21BCE2682

Assessment 3

---

1. Download the dataset: Dataset

2. Load the dataset into the tool

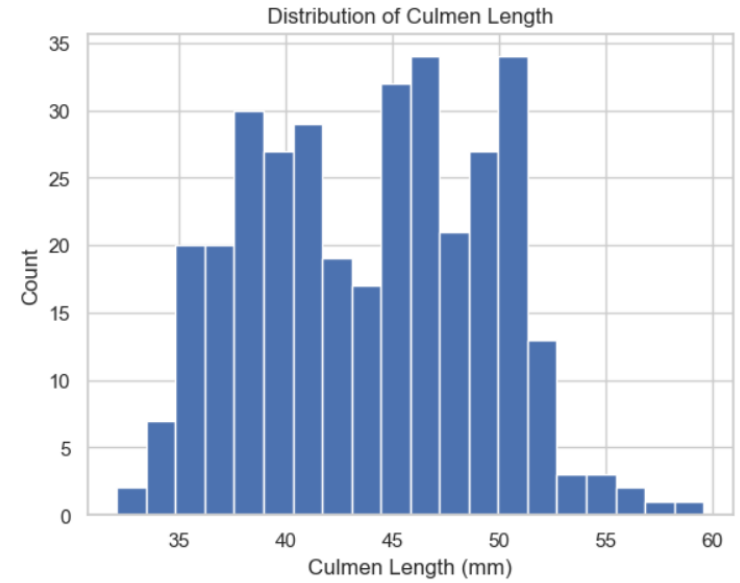
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix


d = pd.read_csv('C:\\Users\\wwwad\\Downloads\\penguins_size.csv')
df=pd.DataFrame(d)
print(d)
```

```
    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
..      ...        ...               ...              ...                ...
339  Gentoo     Biscoe               NaN              NaN                NaN
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
3            NaN     NaN
4         3450.0  FEMALE
..           ...     ...
339          NaN     NaN
340       4850.0  FEMALE
341       5750.0    MALE
342       5200.0  FEMALE
343       5400.0    MALE

[344 rows x 7 columns]
```

3. Perform Below Visualizations.

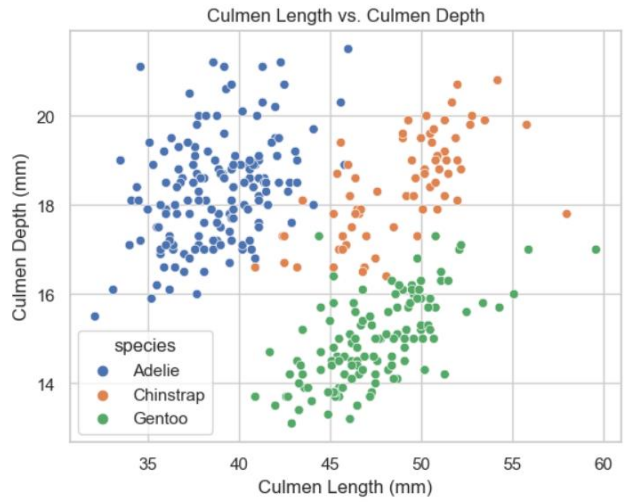
• *Univariate Analysis*

```
plt.hist(df['culmen_length_mm'], bins=20)
plt.xlabel('Culmen Length (mm)')
plt.ylabel('Count')
plt.title('Distribution of Culmen Length')
plt.show()
```



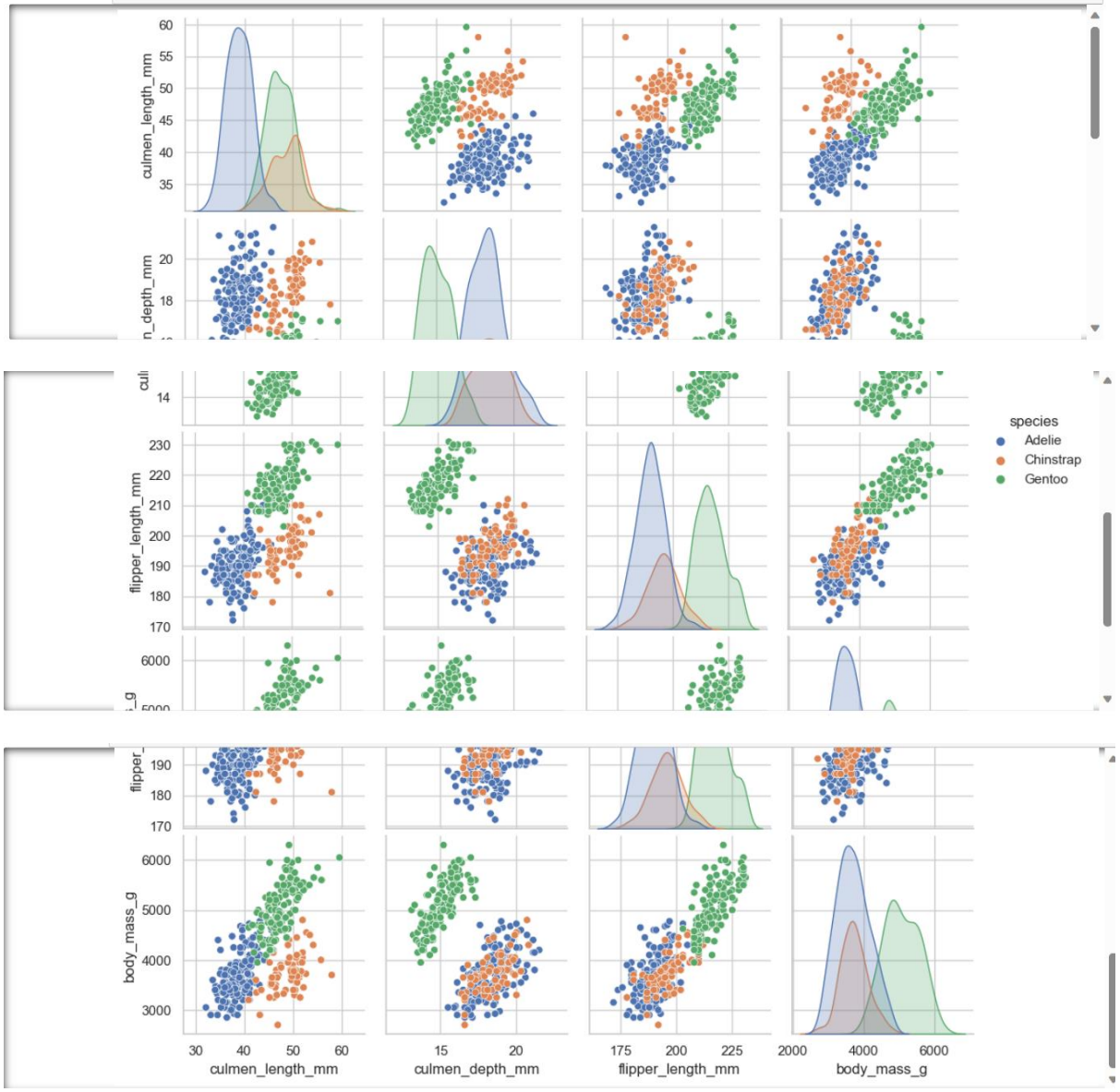
• *Bi- Variate Analysis*

```
sns.scatterplot(x='culmen_length_mm', y='culmen_depth_mm', data=df, hue='species')
plt.xlabel('Culmen Length (mm)')
plt.ylabel('Culmen Depth (mm)')
plt.title('Culmen Length vs. Culmen Depth')
plt.show()
```



● *Multi-Variate Analysis*

```
In [18]: sns.pairplot(df, hue='species')
         plt.show()
```



4. Perform descriptive statistics on the dataset.

```
In [19]: print(df.describe())
```

	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

5. Check for Missing values and deal with them.

In [20]:
```
print(df.isnull().sum())
df.dropna(inplace=True)
```

```
species              0
island               0
culmen_length_mm     2
culmen_depth_mm      2
flipper_length_mm    2
body_mass_g          2
sex                 10
dtype: int64
```

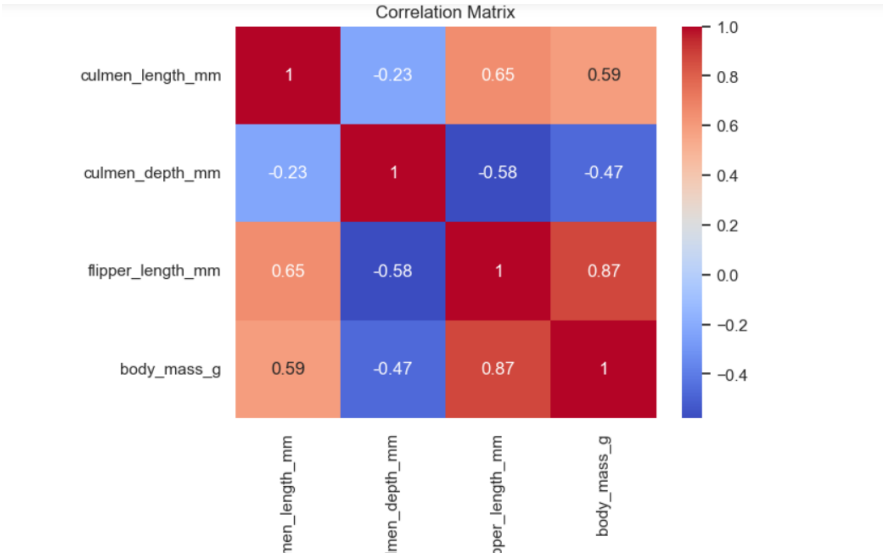
6. Find the outliers and replace them outliers

In [23]:
```
from scipy import stats
z_scores = np.abs(stats.zscore(df.select_dtypes(include='number')))
threshold = 3
df_no_outliers = df[(z_scores < threshold).all(axis=1)]

print(f"Number of rows before removing outliers: {df.shape[0]}")
print(f"Number of rows after removing outliers: {df_no_outliers.shape[0]}")
```

```
Number of rows before removing outliers: 334
Number of rows after removing outliers: 334
```

7.Check the correlation of independent variables with the target

```
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



8. Check for Categorical columns and perform encoding.

In [26]:
```
categorical_columns = df.select_dtypes(include=['object']).columns.tolist()
label_encoders = {}
for col in categorical_columns:
    le = LabelEncoder()
    df_no_outliers[col] = le.fit_transform(df_no_outliers[col])
    label_encoders[col] = le
for col, le in label_encoders.items():
    print(f"Label encoding for {col}:")
    for label, code in zip(le.classes_, le.transform(le.classes_)):
        print(f"{label}: {code}")
```

```
Label encoding for island:
Biscoe: 0
Dream: 1
Torgersen: 2
Label encoding for sex:
.: 0
FEMALE: 1
MALE: 2
```

9. Split the data into dependent and independent variables.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
target_column = 'your_target_column_name_here'  # Replace with the actual target column name

columns_to_scale = [col for col in df_no_outliers.columns if col != target_column and df_no_outliers[col].dtype in ['int64', 'flo
if columns_to_scale:
    df_no_outliers[columns_to_scale] = scaler.fit_transform(df_no_outliers[columns_to_scale])

    print("Mean of scaled features:")
    print(df_no_outliers[columns_to_scale].mean())
    print("\nStandard deviation of scaled features:")
    print(df_no_outliers[columns_to_scale].std())
else:
    print("No numeric columns to scale.")
```

```
Mean of scaled features:
culmen_length_mm    -4.254747e-17
culmen_depth_mm     -1.276424e-16
flipper_length_mm    0.000000e+00
body_mass_g          4.254747e-17
dtype: float64

Standard deviation of scaled features:
culmen_length_mm     1.0015
culmen_depth_mm      1.0015
flipper_length_mm    1.0015
body_mass_g          1.0015
dtype: float64
```

10. Scaling the data

```
scaler = StandardScaler()
target_column = 'species'

columns_to_scale = [col for col in X.columns if X[col].dtype in ['int64', 'float64']]
if columns_to_scale:
    X_scaled = scaler.fit_transform(X[columns_to_scale])
else:
    print("No numeric columns to scale.")

print("Scaled Data:")
print(X_scaled[:5])
```

```
Scaled Data:
[[-0.89765322  0.78348666 -1.42952144 -0.57122888]
 [-0.82429023  0.12189602 -1.07240838 -0.50901123]
 [-0.67756427  0.42724555 -0.42960487 -1.19340546]
 [-1.33783112  1.08883619 -0.5724501  -0.94453483]
 [-0.86097173  1.75042684 -0.78671793 -0.6956642 ]]
```

11. Split the data into training and testing

12.check the training and testing data shape.

[40]:
```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

[41]:
```
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: (267, 4)
X_test shape: (67, 4)
y_train shape: (267,)
y_test shape: (67,)
```