ASSIGNMENT 3

The Penguin Classification Analysis problem involves predicting the species of a penguin based on various physical characteristics. The dataset includes information about the body mass, culmen length, culmen depth, flipper length, and sex of different penguin specie

Clustering the data and performing classification algorithms

1. Download the dataset: Dataset
2. Load the dataset into the tool.
3. Perform Below Visualizations. ● Univariate Analysis ● Bi- Variate Analysis ● Multi-Variate Analysis
4. Perform descriptive statistics on the dataset.
5. Check for Missing values and deal with them.
6. Find the outliers and replace them outliers 7.Check the correlation of independent variables with the target
7. Check for Categorical columns and perform encoding.
8. Split the data into dependent and independent variables.
9. Scaling the data
10. Split the data into training and testing 12.check the training and testing data shape

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df=pd.read_csv(r"D:\MachineLearning\DataScienceCourse\
penguins_size.csv")
df
```

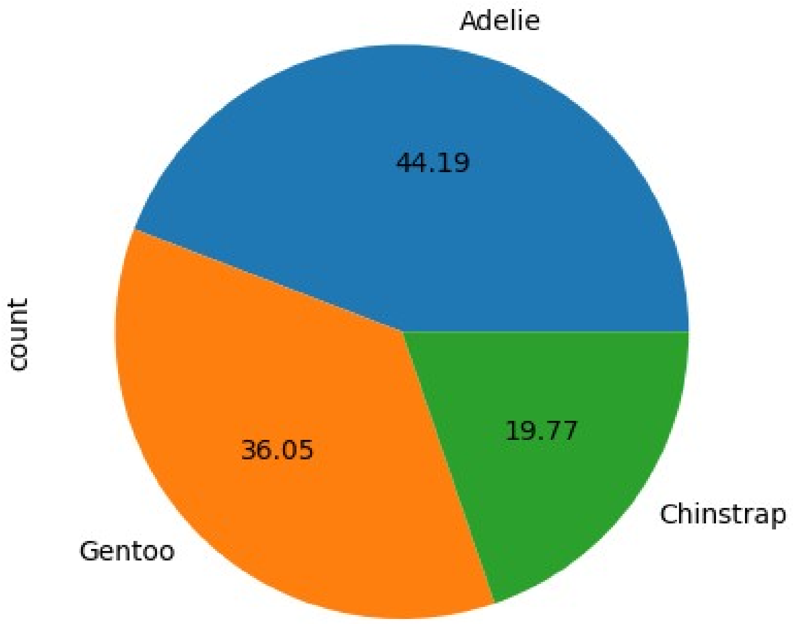
```
    species     island  culmen_length_mm  culmen_depth_mm  
flipper_length_mm  \
0    Adelie  Torgersen              39.1             18.7  
181.0   
1    Adelie  Torgersen              39.5             17.4  
186.0   
2    Adelie  Torgersen              40.3             18.0  
195.0   
3    Adelie  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]
```

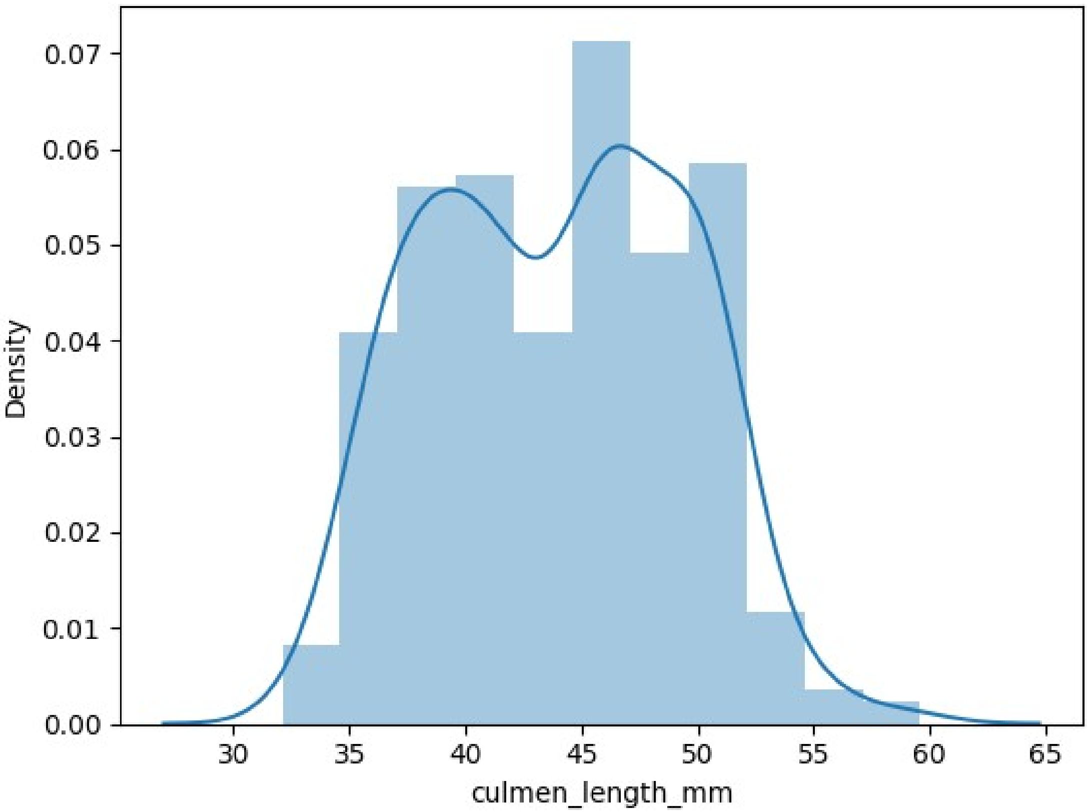
Univariate Analysis

```
df["species"].value_counts().plot(kind='pie',autopct='%.2f')
```

```
<Axes: ylabel='count'>
```



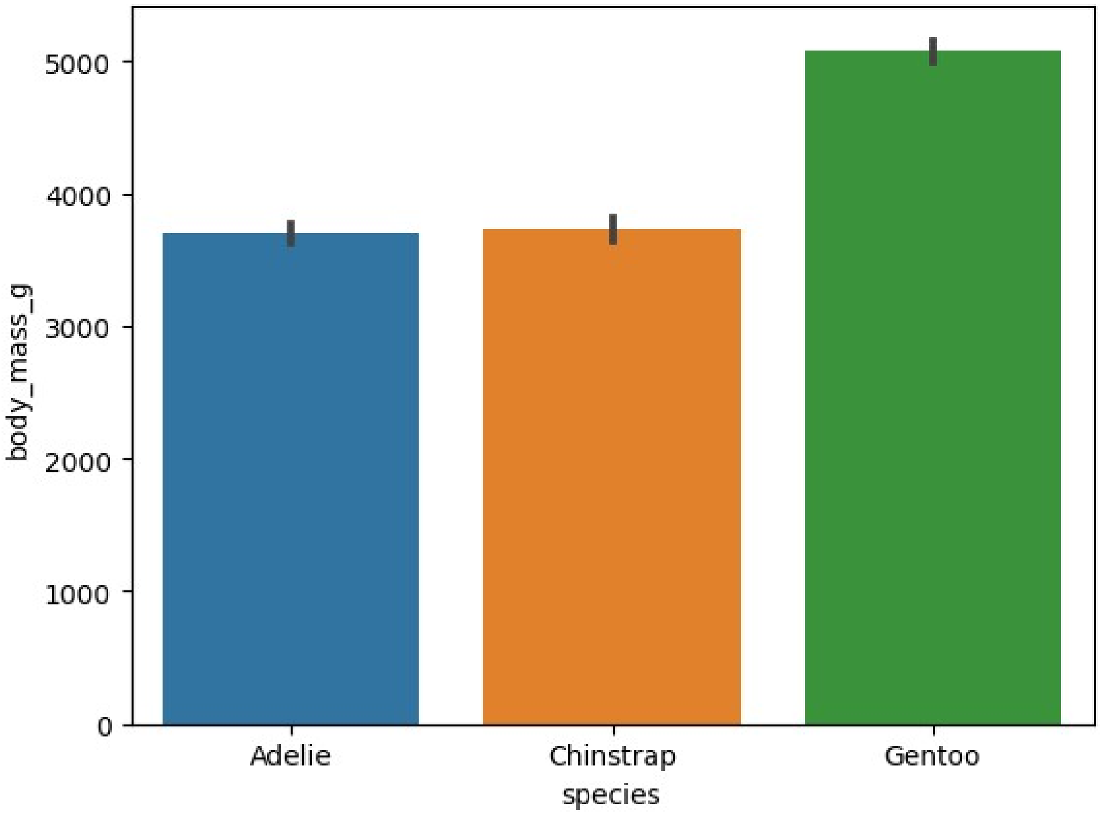
```
sns.distplot(df["culmen_length_mm"])

C:\Users\Vidul\AppData\Local\Temp\ipykernel_7360\2669382467.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["culmen_length_mm"])

<Axes: xlabel='culmen_length_mm', ylabel='Density'>
```



Bivariate Analysis

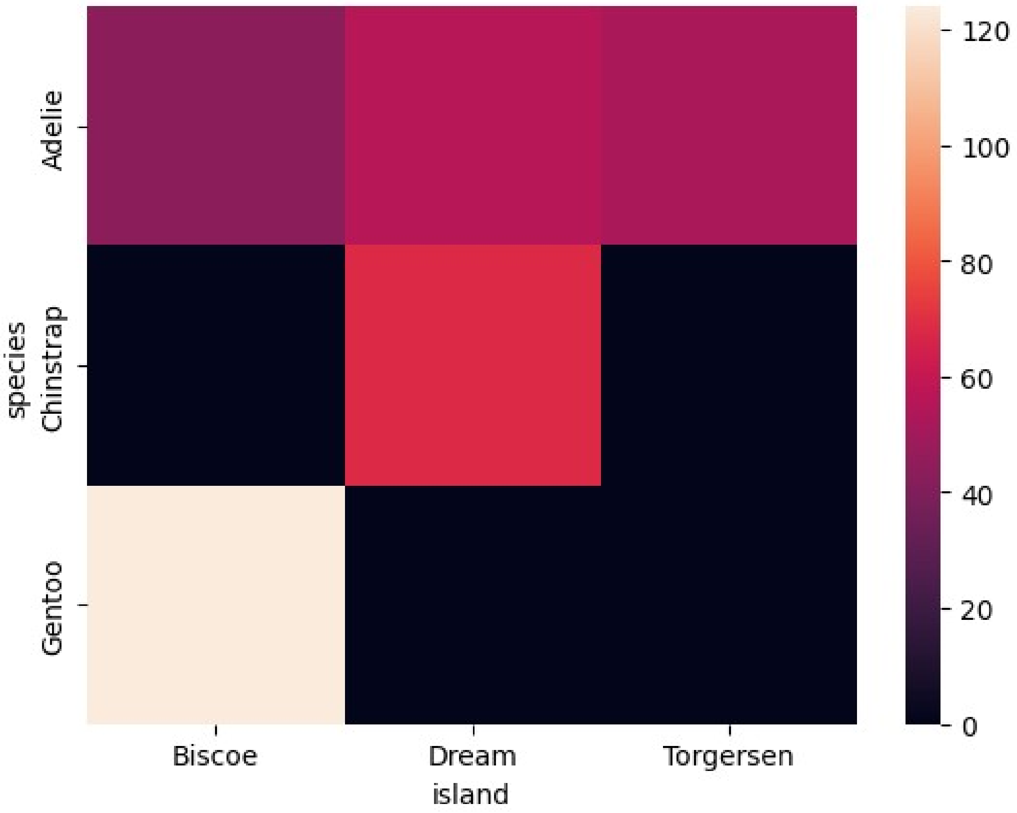
```
sns.barplot(x=df["species"],y=df["body_mass_g"])
```

```
<Axes: xlabel='species', ylabel='body_mass_g'>
```



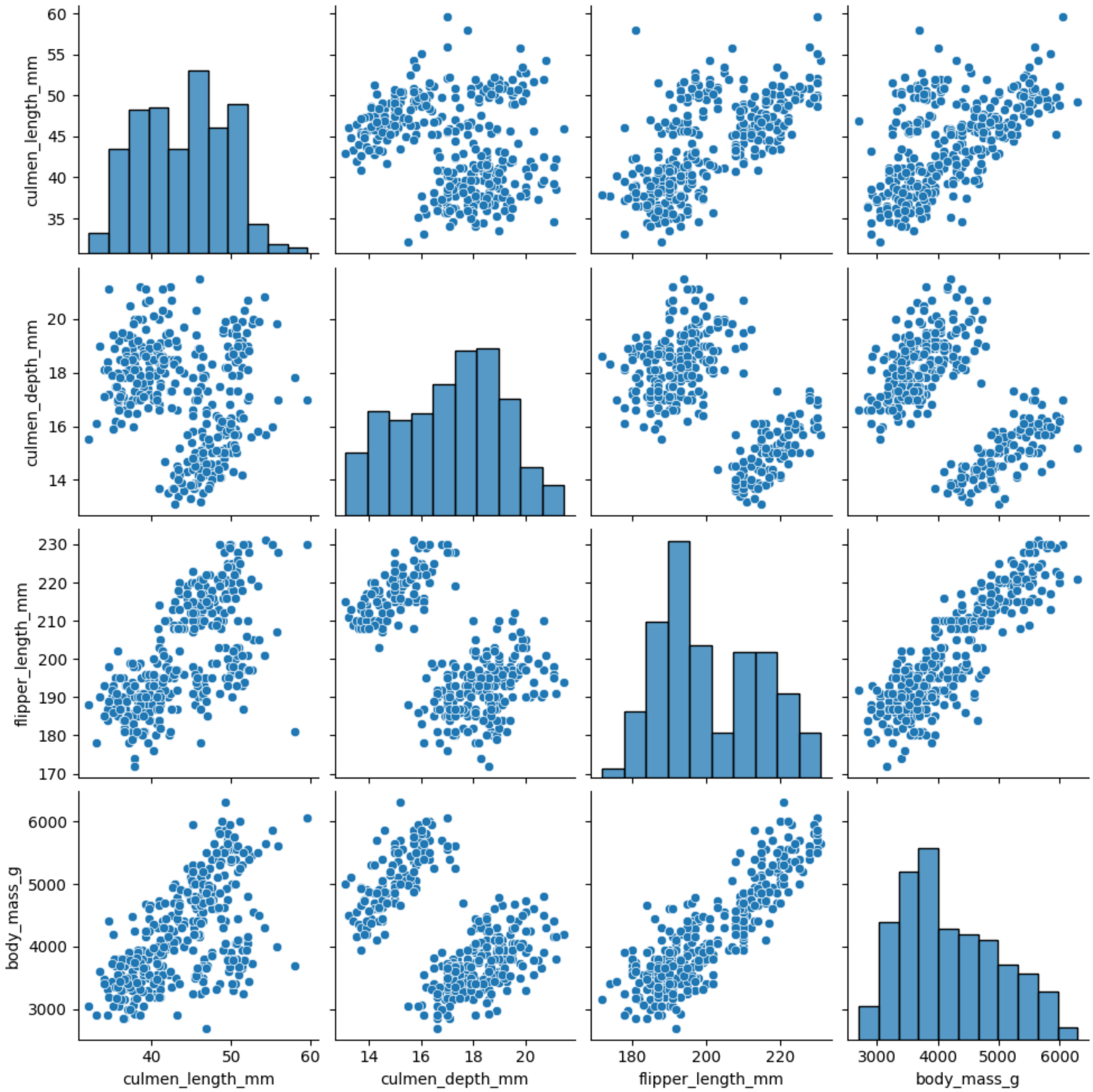
```
sns.heatmap(pd.crosstab(df["species"],df["island"]))
```

```
<Axes: xlabel='island', ylabel='species'>
```



Multivariate Analysis

```
sns.pairplot(df)

C:\Users\Vidul\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)

<seaborn.axisgrid.PairGrid at 0x14901835410>
```



Descriptive Statistics

```
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
```

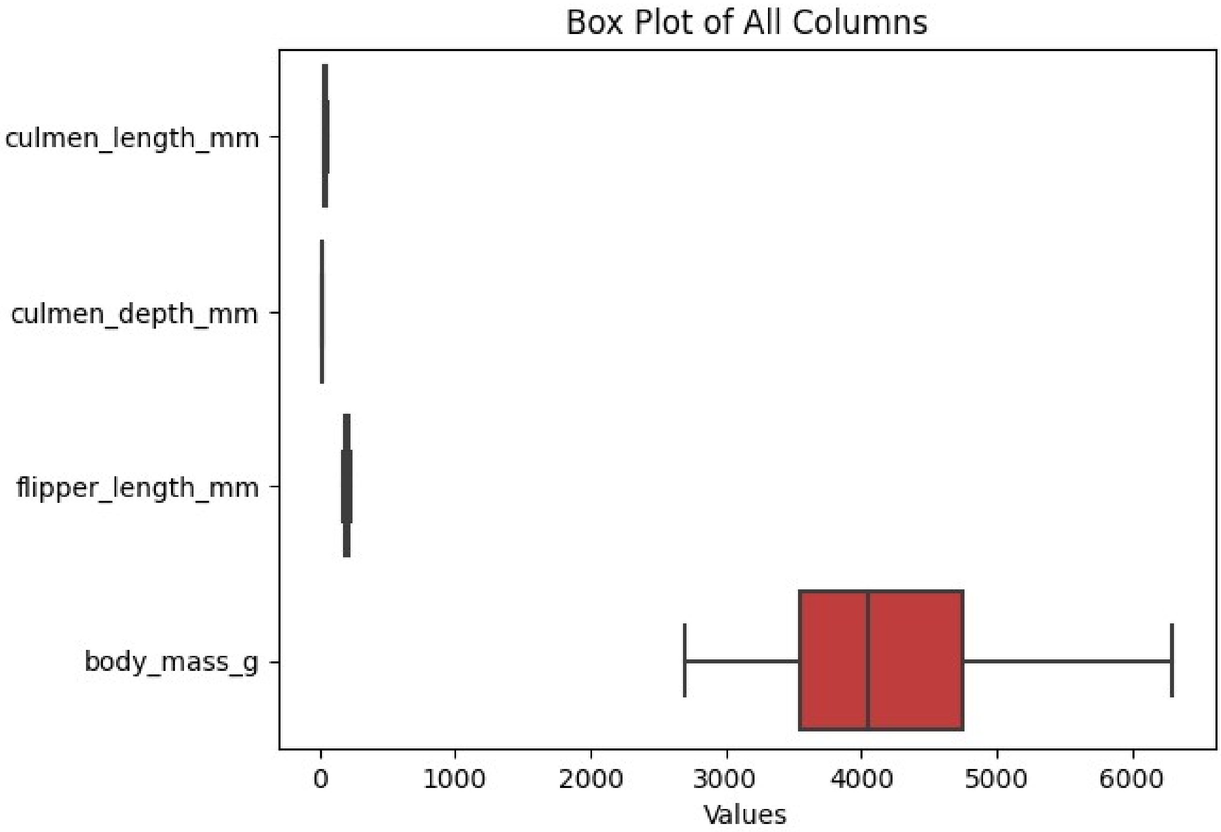
Handling Missing values

```
df.isnull().sum()

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

# Handling missing values for numerical data (using median())
l=['culmen_length_mm','culmen_depth_mm','flipper_length_mm','body_mass
_g']
for i in l:
    df[i]=df[i].fillna(df[i].median())

# Handling missing values for categorical data (using mode)
df["sex"]=df["sex"].fillna(df["sex"].mode().iloc[0])

df.isnull().sum()
df

    species     island  culmen_length_mm  culmen_depth_mm
flipper_length_mm  \
0    Adelie  Torgersen             39.10             18.7
181.0
1    Adelie  Torgersen             39.50             17.4
186.0
2    Adelie  Torgersen             40.30             18.0
195.0
3    Adelie  Torgersen             44.45             17.3
197.0
4    Adelie  Torgersen             36.70             19.3
193.0
..      ...        ...               ...              ...
...
```

```
339  Gentoo     Biscoe             44.45             17.3
197.0
340  Gentoo     Biscoe             46.80             14.3
215.0
341  Gentoo     Biscoe             50.40             15.7
222.0
342  Gentoo     Biscoe             45.20             14.8
212.0
343  Gentoo     Biscoe             49.90             16.1
213.0

     body_mass_g     sex
0         3750.0    MALE
1         3800.0  FEMALE
2         3250.0  FEMALE
3         4050.0    MALE
4         3450.0  FEMALE
..           ...     ...
339       4050.0    MALE
340       4850.0  FEMALE
341       5750.0    MALE
342       5200.0  FEMALE
343       5400.0    MALE

[344 rows x 7 columns]
```

```
sns.boxplot(data=df, orient='h')  # 'orient' is set to 'h' for
horizontal box plots

plt.xlabel('Values')
plt.title('Box Plot of All Columns')
```

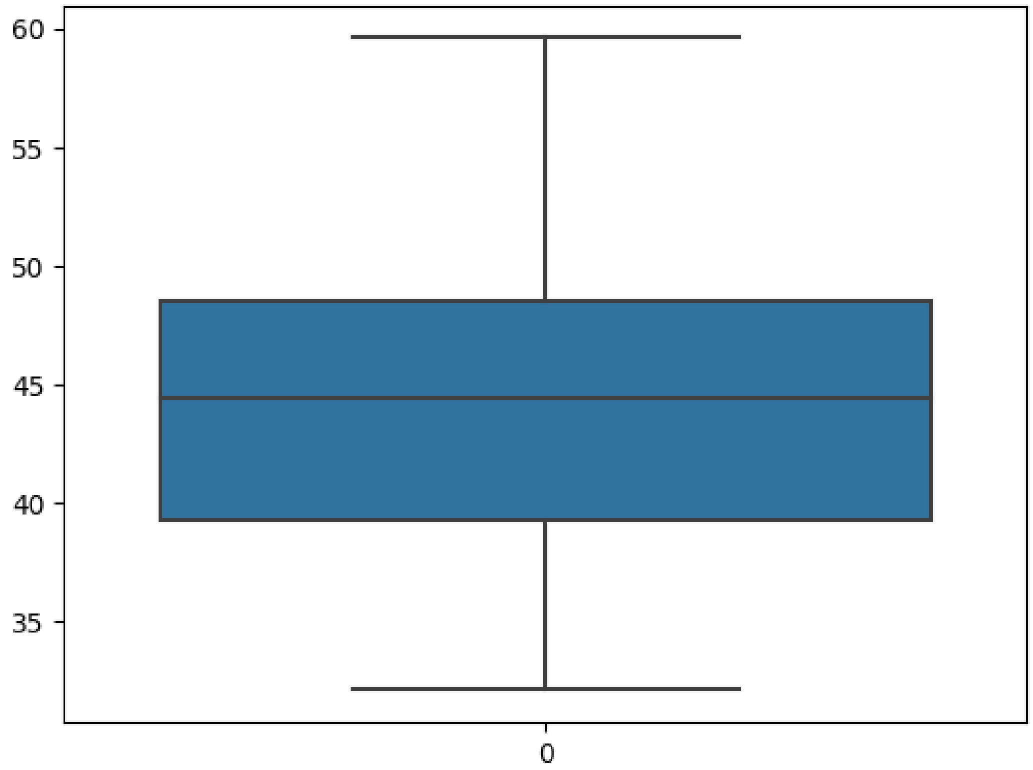
```
Text(0.5, 1.0, 'Box Plot of All Columns')
```



No outliers are there

```
sns.boxplot(df["culmen_length_mm"])
```

```
<Axes: >
```



Encoding

```
# One hot
df = pd.get_dummies(df, columns = ['sex'], dtype=int)

#Label
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
l=["species","island"]
for i in l:
  df[i]=le.fit_transform(df[i])
df.head(10)
```

```
   species  island  culmen_length_mm  culmen_depth_mm
flipper_length_mm  \
0        0       2             39.10             18.7
181.0
1        0       2             39.50             17.4
186.0
2        0       2             40.30             18.0
195.0
3        0       2             44.45             17.3
197.0
4        0       2             36.70             19.3
193.0
5        0       2             39.30             20.6
```

```
190.0
6         0       2             38.90            17.8
181.0
7         0       2             39.20            19.6
195.0
8         0       2             34.10            18.1
193.0
9         0       2             42.00            20.2
190.0

   body_mass_g  sex_.  sex_FEMALE  sex_MALE
0       3750.0      0           0         1
1       3800.0      0           1         0
2       3250.0      0           1         0
3       4050.0      0           0         1
4       3450.0      0           1         0
5       3650.0      0           0         1
6       3625.0      0           1         0
7       4675.0      0           0         1
8       3475.0      0           0         1
9       4250.0      0           0         1
```

```
df.drop("sex_.",axis=1,inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            344 non-null    int32  
 1   island             344 non-null    int32  
 2   culmen_length_mm   344 non-null    float64
 3   culmen_depth_mm    344 non-null    float64
 4   flipper_length_mm  344 non-null    float64
 5   body_mass_g        344 non-null    float64
 6   sex_FEMALE         344 non-null    int32  
 7   sex_MALE           344 non-null    int32  
dtypes: float64(4), int32(4)
memory usage: 16.3 KB
```

Checking correlations

```
df[["island","culmen_length_mm","culmen_depth_mm","flipper_length_mm",
"body_mass_g","sex_FEMALE","sex_MALE"]].corrwith(df["species"])
```

```
island              -0.635659
culmen_length_mm     0.728706
culmen_depth_mm     -0.741282
flipper_length_mm    0.850819
```

```
body_mass_g          0.747547
sex_FEMALE          -0.010240
sex_MALE             0.003185
dtype: float64
```

Splitting the data into dependent and independent variables

```
x=df.iloc[:,1:]
y=df.iloc[:,0]
x.info()
x.head(5)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   island             344 non-null    int32  
 1   culmen_length_mm   344 non-null    float64
 2   culmen_depth_mm    344 non-null    float64
 3   flipper_length_mm  344 non-null    float64
 4   body_mass_g        344 non-null    float64
 5   sex_FEMALE         344 non-null    int32  
 6   sex_MALE           344 non-null    int32  
dtypes: float64(4), int32(3)
memory usage: 14.9 KB
```

```
   island  culmen_length_mm  culmen_depth_mm  flipper_length_mm
body_mass_g  \
0       2             39.10             18.7              181.0
3750.0
1       2             39.50             17.4              186.0
3800.0
2       2             40.30             18.0              195.0
3250.0
3       2             44.45             17.3              197.0
4050.0
4       2             36.70             19.3              193.0
3450.0

   sex_FEMALE  sex_MALE
0           0         1
1           1         0
2           1         0
3           0         1
4           1         0
```

Scaling

```
# Feature scaling (MinMax Scaler or Standard Scaler)
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
x=scaler.fit_transform(x)
x=pd.DataFrame(x)
x
```

```
       0         1         2         3         4    5    6
0    1.0  0.254545  0.666667  0.152542  0.291667  0.0  1.0
1    1.0  0.269091  0.511905  0.237288  0.305556  1.0  0.0
2    1.0  0.298182  0.583333  0.389831  0.152778  1.0  0.0
3    1.0  0.449091  0.500000  0.423729  0.375000  0.0  1.0
4    1.0  0.167273  0.738095  0.355932  0.208333  1.0  0.0
..   ...       ...       ...       ...       ...  ...  ...
339  0.0  0.449091  0.500000  0.423729  0.375000  0.0  1.0
340  0.0  0.534545  0.142857  0.728814  0.597222  1.0  0.0
341  0.0  0.665455  0.309524  0.847458  0.847222  0.0  1.0
342  0.0  0.476364  0.202381  0.677966  0.694444  1.0  0.0
343  0.0  0.647273  0.357143  0.694915  0.750000  0.0  1.0

[344 rows x 7 columns]
```

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=0.2,random_state=32)

x_train.shape
```

```
(275, 7)
```

```
x_test.shape
```

```
(69, 7)
```

```
y_train.shape
```

```
(275,)
```

```
y_test.shape
```

```
(69,)
```