AI&ML ASSIGNMENT-3

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

2. Loading the dataset

In [10]:
```
df=pd.read_csv("penguins_size.csv")
df.head()
```

Out[10]:

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mas
0	Adelie	Torgersen	39.1	18.7	181.0	37
1	Adelie	Torgersen	39.5	17.4	186.0	38
2	Adelie	Torgersen	40.3	18.0	195.0	32
3	Adelie	Torgersen	NaN	NaN	NaN	
4	Adelie	Torgersen	36.7	19.3	193.0	34

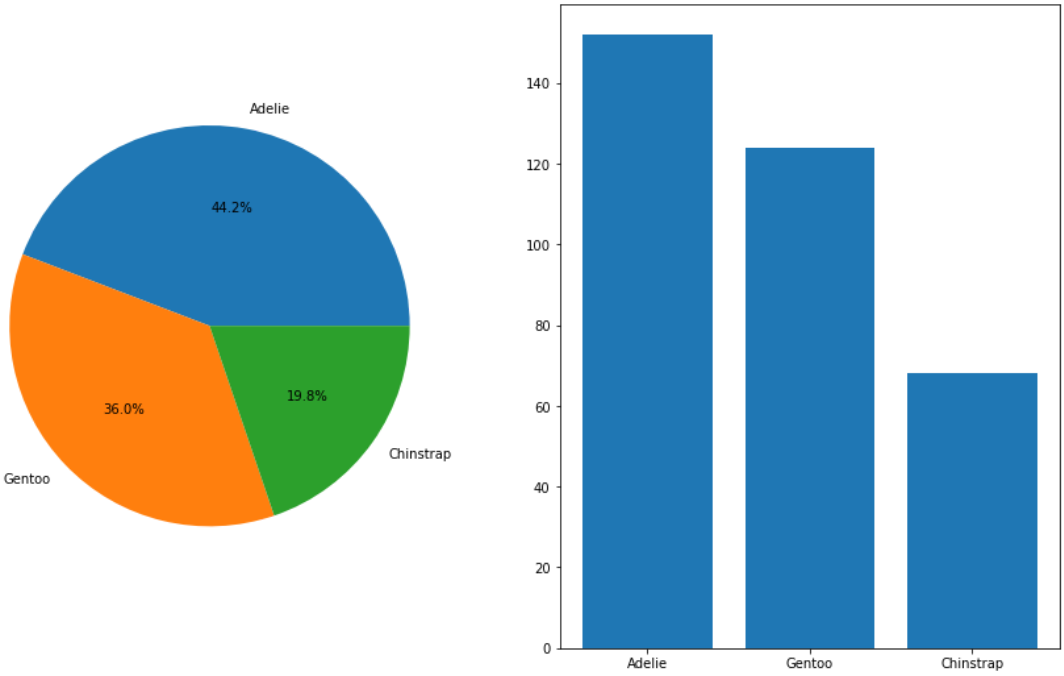
Univariate Analysis

In [12]:
```
df['species'].value_counts()
```

Out[12]:
```
Adelie       152
Gentoo       124
Chinstrap     68
Name: species, dtype: int64
```

In [254]:
```
a=df['species'].value_counts().index
fig,ax = plt.subplots(1,2, figsize=(15,9))
ax[0].pie(df['species'].value_counts(),labels=a,autopct = "%1.1f%%"
ax[1].bar(a,df['species'].value_counts())
```

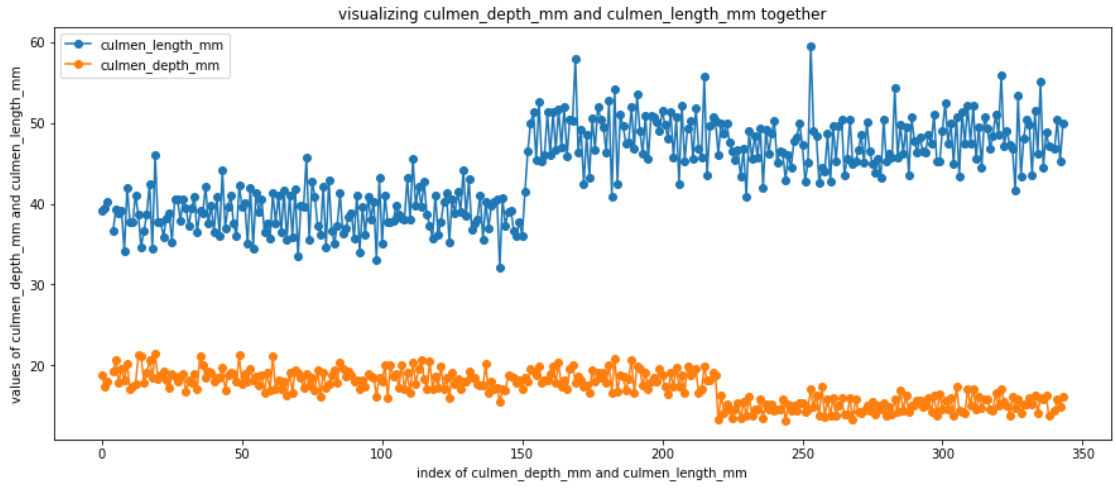
Out[254]: <BarContainer object of 3 artists>



Bi- Variate Analysis

In [262]:
```
plt.figure(figsize=(15,6))
plt.plot(df['culmen_length_mm'],'o-')
plt.plot(df['culmen_depth_mm'],'o-')
plt.title("visualizing culmen_depth_mm and culmen_length_mm togethe
plt.xlabel("index of culmen_depth_mm and culmen_length_mm")
plt.ylabel("values of culmen_depth_mm and culmen_length_mm")
plt.legend(['culmen_length_mm','culmen_depth_mm'])
```

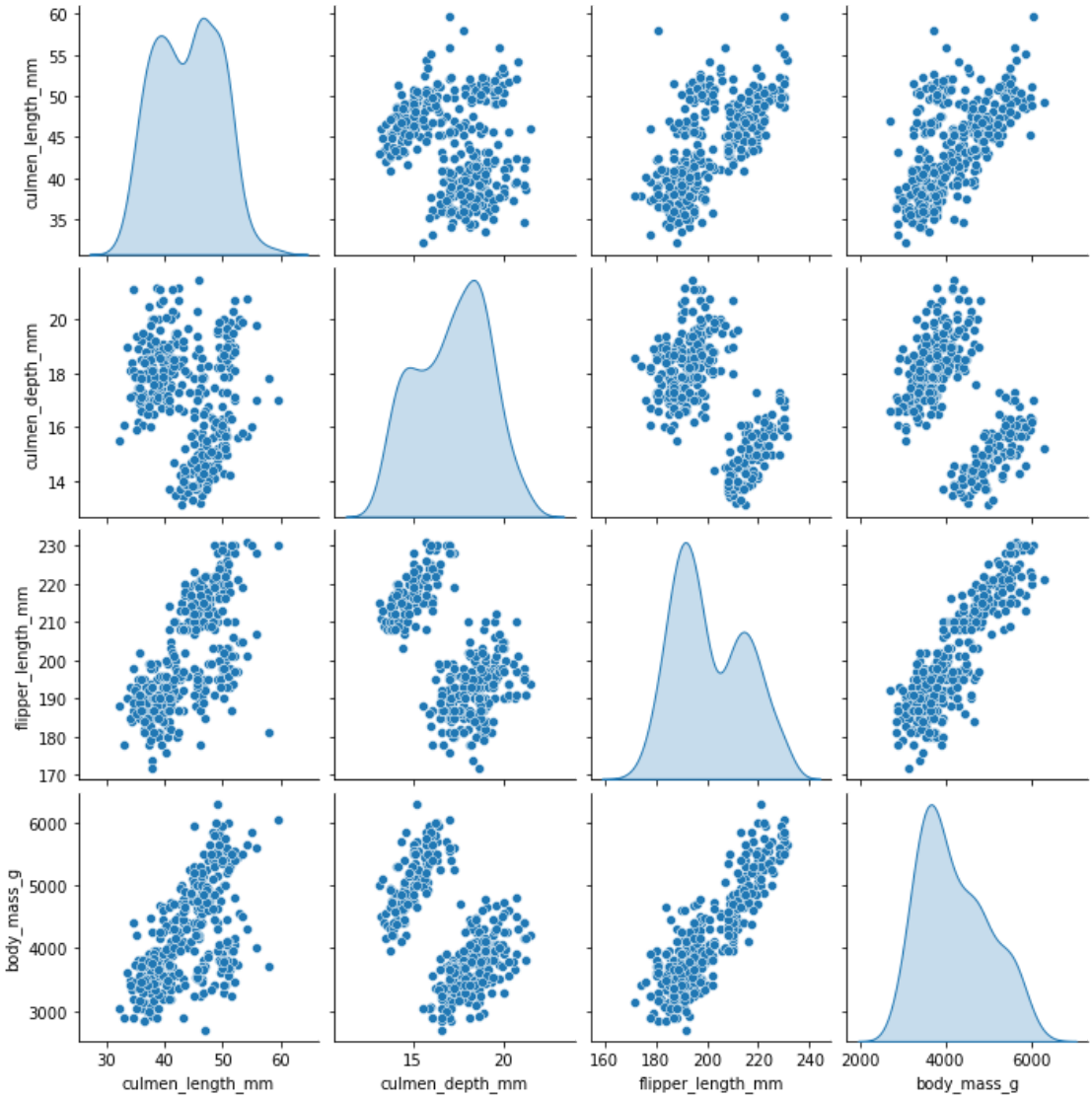
Out[262]: <matplotlib.legend.Legend at 0x1c1d4aad850>



Multi-Variate Analysis

In [266]:
```
sns.pairplot(df,diag_kind='kde')
```

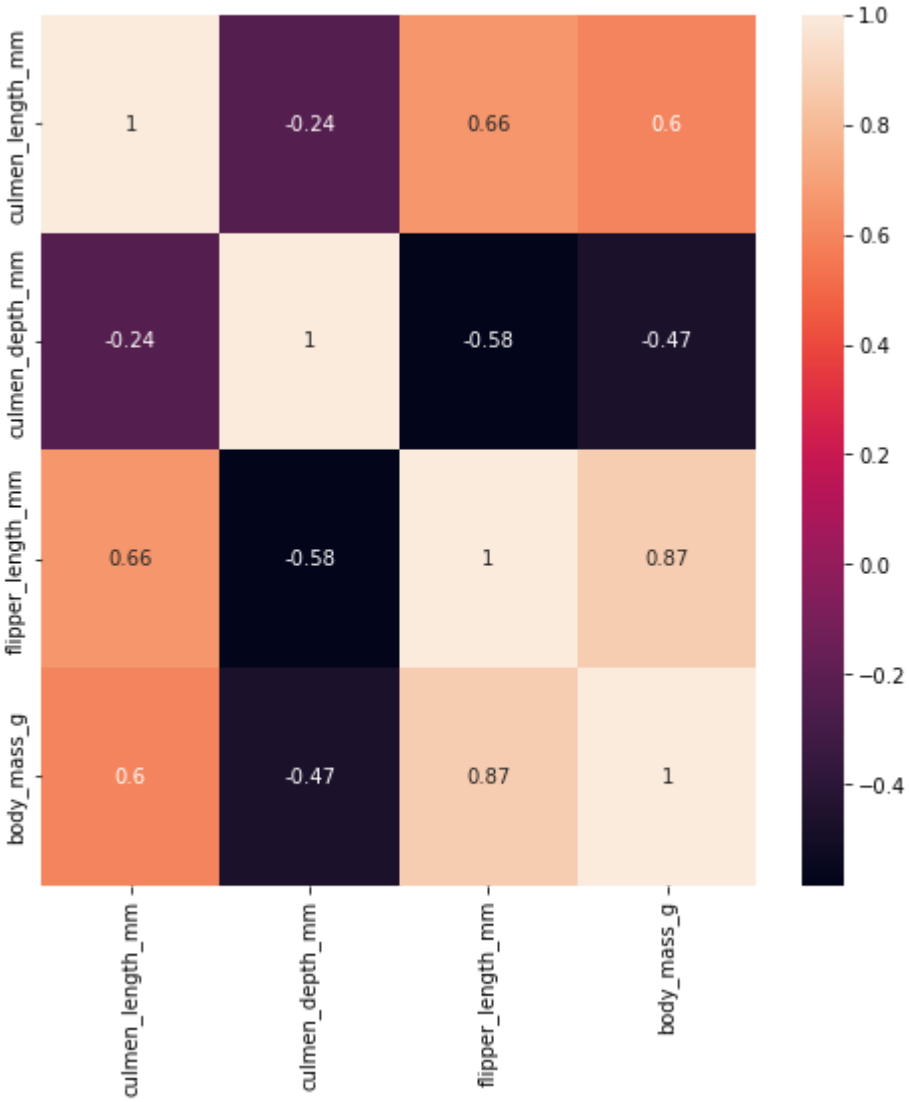
Out[266]:
```
<seaborn.axisgrid.PairGrid at 0x1c1d5231a30>
```



In [269]:
```
sns.heatmap(df.corr(),annot=True)
```

```
C:\Users\HP\AppData\Local\Temp/ipykernel_15880/4277794465.py:1: Fu
tureWarning: The default value of numeric_only in DataFrame.corr i
s deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silen
ce this warning.
  sns.heatmap(df.corr(),annot=True)
```

Out[269]: <AxesSubplot:>



4. DESCRIPTIVE STATISTICS ON OUR DATASET

In [273]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            344 non-null    object 
 1   island             344 non-null    object 
 2   culmen_length_mm   342 non-null    float64
 3   culmen_depth_mm    342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                334 non-null    object 
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

In [274]:
```
df.dtypes
```

Out[274]:
```
species               object
island                object
culmen_length_mm     float64
culmen_depth_mm      float64
flipper_length_mm    float64
body_mass_g          float64
sex                   object
dtype: object
```

In [275]:
```
df.describe()
```

Out[275]:

	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. CHECKING FOR MISSING VALUES AND IMPUTING THEM WITH STATISTICAL METHODS

(MEAN FOR NUMERIC DATA AND MODE FOR CATEGORICAL DATA)

In [277]:
```
df.isnull().sum() # columns except species and island are having nu
#except sex columns remaining columns are of type float
```

Out[277]:
```
species              0
island               0
culmen_length_mm     2
culmen_depth_mm      2
flipper_length_mm    2
body_mass_g          2
sex                 10
dtype: int64
```

In [4]:
```
x=['culmen_length_mm', 'culmen_depth_mm','flipper_length_mm', 'body
for column in x:
    df[column].fillna(df[column].mean(),inplace =True)
df['sex'].fillna(df['sex'].mode()[0],inplace =True)
```

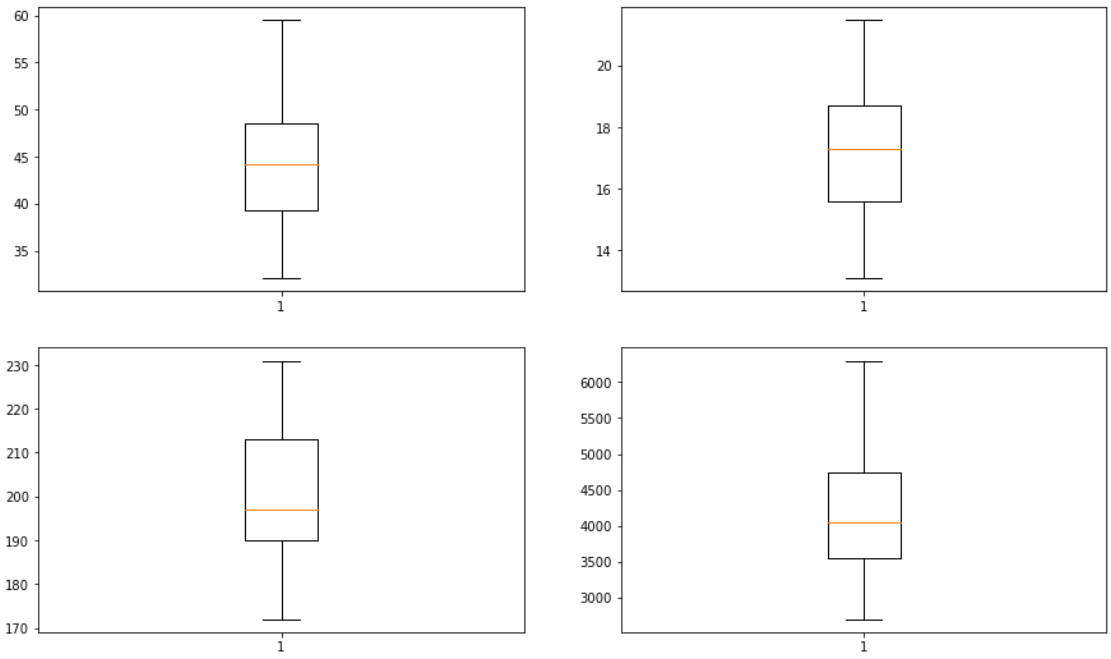
In [279]:
```
#again checking for null values
df.isnull().sum()
```

Out[279]:
```
species             0
island              0
culmen_length_mm    0
culmen_depth_mm     0
flipper_length_mm   0
body_mass_g         0
sex                 0
dtype: int64
```

6. FINDING THE OUTLIERS AND REPLACING THEM

In [280]:
```
fig,ax = plt.subplots(2,2, figsize=(15,9))
ax[0][0].boxplot(df['culmen_length_mm'])
ax[0][1].boxplot(df['culmen_depth_mm'])
ax[1][0].boxplot(df['flipper_length_mm'])
ax[1][1].boxplot(df['body_mass_g']) #no outliers in following colum
```

Out[280]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1c1d75d5eb0>,
  <matplotlib.lines.Line2D at 0x1c1d75e2280>],
 'caps': [<matplotlib.lines.Line2D at 0x1c1d75e2610>,
  <matplotlib.lines.Line2D at 0x1c1d75e29a0>],
 'boxes': [<matplotlib.lines.Line2D at 0x1c1d75d5b20>],
 'medians': [<matplotlib.lines.Line2D at 0x1c1d75e2d30>],
 'fliers': [<matplotlib.lines.Line2D at 0x1c1d75ef100>],
 'means': []}
```



In [282]:
```
def funct(col):
    print(col +"\n")
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    print(f" First quartile of {col} is q1= {q1} \n Second quartile
    iqr=q3-q1
    print(f" IQR OF {col} is {iqr}")
    upper_limit = q3+1.5*iqr
    lower_limit =q1-1.5*iqr
    print(f" Upper limit of {col} is: {upper_limit} \n Lower limit
    print()
```

In [283]:
```
z=['culmen_length_mm', 'culmen_depth_mm','flipper_length_mm', 'body
for col in z:
    funct(col)
```

```
culmen_length_mm

 First quartile of culmen_length_mm is q1= 39.275
 Second quartile of culmen_length_mm is q3= 48.5
 IQR OF culmen_length_mm is 9.225000000000001
 Upper limit of culmen_length_mm is: 62.337500000000006
 Lower limit of culmen_length_mm is: 25.437499999999996

culmen_depth_mm

 First quartile of culmen_depth_mm is q1= 15.6
 Second quartile of culmen_depth_mm is q3= 18.7
 IQR OF culmen_depth_mm is 3.0999999999999996
 Upper limit of culmen_depth_mm is: 23.349999999999998
 Lower limit of culmen_depth_mm is: 10.95

flipper_length_mm

 First quartile of flipper_length_mm is q1= 190.0
 Second quartile of flipper_length_mm is q3= 213.0
 IQR OF flipper_length_mm is 23.0
 Upper limit of flipper_length_mm is: 247.5
 Lower limit of flipper_length_mm is: 155.5

body_mass_g

 First quartile of body_mass_g is q1= 3550.0
 Second quartile of body_mass_g is q3= 4750.0
 IQR OF body_mass_g is 1200.0
 Upper limit of body_mass_g is: 6550.0
 Lower limit of body_mass_g is: 1750.0
```

7. CHECKING THE CORRELATION OF INDEPENDENT VARIABLES WITH TARGET

In [18]:
```
df.corr().species.sort_values(ascending=False)
```

Out[18]:
```
species              1.000000
flipper_length_mm    0.854307
body_mass_g          0.750491
culmen_length_mm     0.731369
sex                  0.002262
island              -0.635659
culmen_depth_mm     -0.744076
Name: species, dtype: float64
```

8. CHECKING FOR CATEGORICAL COLUMNS AND PERFORMING

In [14]:
```
df.species.value_counts()
```

Out[14]:
```
Adelie       152
Gentoo       124
Chinstrap     68
Name: species, dtype: int64
```

In [15]:
```
df.island.value_counts()
```

Out[15]:
```
Biscoe       168
Dream        124
Torgersen     52
Name: island, dtype: int64
```

In [16]:
```
df['sex']=df['sex'].replace(".","MALE")
df.sex.value_counts()
```

Out[16]:
```
MALE      169
FEMALE    165
Name: sex, dtype: int64
```

In [17]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

df.species= le.fit_transform(df.species)
df.island= le.fit_transform(df.island)
df.sex = le.fit_transform(df.sex)
```

In [290]:
```
df.head()
```

Out[290]:

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_
0	0	2	39.10000	18.70000	181.000000	3750.00000
1	0	2	39.50000	17.40000	186.000000	3800.00000
2	0	2	40.30000	18.00000	195.000000	3250.00000
3	0	2	43.92193	17.15117	200.915205	4201.75438
4	0	2	36.70000	19.30000	193.000000	3450.00000

9. SPLIT THE DATA INTO DEPENDENT AND INDEPENDENT VARIABLES.

In [291]:
```
X=df.drop(columns =['sex'],axis =1) #dependent variables
y=df.sex #independent variables
```

10. SCALING THE DATA

In [292]:
```
from sklearn.preprocessing import MinMaxScaler
scale =MinMaxScaler()
```

In [293]:
```
X_scaled= pd.DataFrame(scale.fit_transform(X),columns =X.columns)
X_scaled.head()
```

Out[293]:

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_
0	0.0	1.0	0.254545	0.666667	0.152542	0.29166
1	0.0	1.0	0.269091	0.511905	0.237288	0.30555
2	0.0	1.0	0.298182	0.583333	0.389831	0.15277
3	0.0	1.0	0.429888	0.482282	0.490088	0.41715
4	0.0	1.0	0.167273	0.738095	0.355932	0.20833

11. SPLIT THE DATA INTO TRAINING AND TESTING

In [294]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X_scaled,y,test_si
```

12.CHECK THE TRAINING AND TESTING DATA SHAPE.

In [295]:
```
X_train.shape
```

Out[295]: (275, 6)

In [296]:
```
X_test.shape
```

Out[296]: (69, 6)

In [297]:
```
y_train.shape
```

Out[297]: (275,)

In [298]:
```
y_test.shape
```

Out[298]: (69,)