1. Input Liberies

```
import pandas as pd
import numpy as np

from sklearn.linear_model import LinearRegression , Ridge, Lasso
from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

from sklearn.preprocessing import OneHotEncoder,LabelEncoder

from statsmodels.stats.outliers_influence import variance_inflation_factor

import matplotlib.pyplot as plt
import seaborn as sns

from scipy.stats import shapiro , kstest, normaltest, skew
```

2. Problem statement

To predict FISH WEIGHT by using below features features = ['Species', 'Length1', 'Length2', 'Length3', 'Height', 'Width'] Target = 'Weight'

3. Data Gathering

In [144]:

```
raw_df = pd.read_csv("Fish.csv")
raw_df.head()
```

Out[144]:

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340

4,5,6 EDA (Exploratory Data Analysis), Feature Enginnering & Feature selection

In [52]:

```
raw_df.isna().sum()  # checked if there are Null or missing values
```

Out[52]:

```
Species    0
Weight     0
Length1    0
Length2    0
Length3    0
Height     0
Width      0
dtype: int64
```

In [53]:

```
raw_df.info()  # checked if any null value is represented by any symbol and what is data ty
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159 entries, 0 to 158
Data columns (total 7 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Species  159 non-null    object 
 1   Weight   159 non-null    float64
 2   Length1  159 non-null    float64
 3   Length2  159 non-null    float64
 4   Length3  159 non-null    float64
 5   Height   159 non-null    float64
 6   Width    159 non-null    float64
dtypes: float64(6), object(1)
memory usage: 8.8+ KB
```

1. Linearity

In [54]:

```
raw_df.corr()
```

Out[54]:

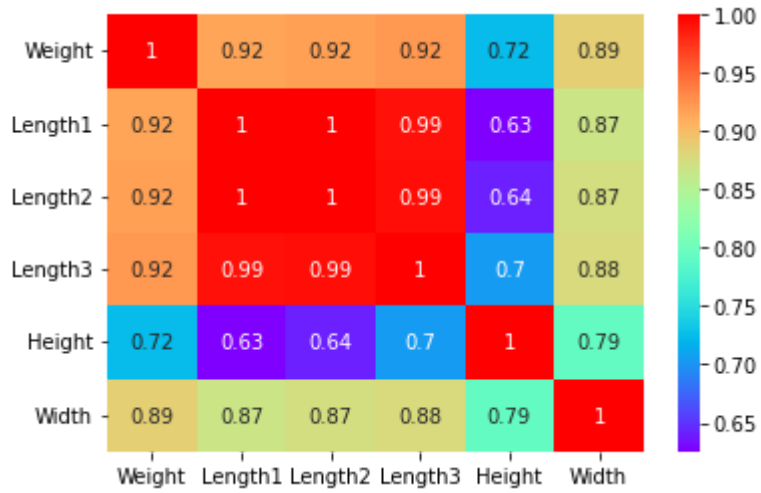
	Weight	Length1	Length2	Length3	Height	Width
Weight	1.000000	0.915712	0.918618	0.923044	0.724345	0.886507
Length1	0.915712	1.000000	0.999517	0.992031	0.625378	0.867050
Length2	0.918618	0.999517	1.000000	0.994103	0.640441	0.873547
Length3	0.923044	0.992031	0.994103	1.000000	0.703409	0.878520
Height	0.724345	0.625378	0.640441	0.703409	1.000000	0.792881
Width	0.886507	0.867050	0.873547	0.878520	0.792881	1.000000

In [55]:

```
sns.heatmap(raw_df.corr(),annot = True, cmap = 'rainbow')
```

Out[55]:

<AxesSubplot:>



2. MultiColinearity

In [56]:

```
vif_list = []
for i in range(raw_df.shape[1]):
    vif = variance_inflation_factor(raw_df.to_numpy(), i)
    vif_list.append(vif)

s1 = pd.Series(vif_list, index = raw_df.columns)     # converted into series
s1.sort_values().plot(kind - 'barh')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [56], in <cell line: 2>()
      1 vif_list = []
      2 for i in range(raw_df.shape[1]):
----> 3     vif = variance_inflation_factor(raw_df.to_numpy(), i)
      4     vif_list.append(vif)
      6 s1 = pd.Series(vif_list, index = raw_df.columns)     # converted int
o series

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels
\stats\outliers_influence.py:194, in variance_inflation_factor(exog, exog_id
x)
    192 mask = np.arange(k_vars) != exog_idx
    193 x_noti = exog[:, mask]
--> 194 r_squared_i = OLS(x_i, x_noti).fit().rsquared
    195 vif = 1. / (1. - r_squared_i)
    196 return vif

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels
\regression\linear_model.py:890, in OLS.__init__(self, endog, exog, missing,
hasconst, **kwargs)
    887     msg = ("Weights are not supported in OLS and will be ignored"
    888            "An exception will be raised in the next version.")
    889     warnings.warn(msg, ValueWarning)
--> 890 super(OLS, self).__init__(endog, exog, missing=missing,
    891                            hasconst=hasconst, **kwargs)
    892 if "weights" in self._init_keys:
    893     self._init_keys.remove("weights")

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels
\regression\linear_model.py:717, in WLS.__init__(self, endog, exog, weights,
missing, hasconst, **kwargs)
    715 else:
    716     weights = weights.squeeze()
--> 717 super(WLS, self).__init__(endog, exog, missing=missing,
    718                            weights=weights, hasconst=hasconst, **kwar
gs)
    719 nobs = self.exog.shape[0]
    720 weights = self.weights

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels
\regression\linear_model.py:191, in RegressionModel.__init__(self, endog, ex
og, **kwargs)
    190 def __init__(self, endog, exog, **kwargs):
--> 191     super(RegressionModel, self).__init__(endog, exog, **kwargs)
    192     self._data_attr.extend(['pinv_wexog', 'wendog', 'wexog', 'weight
s'])

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels
\base\model.py:267, in LikelihoodModel.__init__(self, endog, exog, **kwargs)
```

```
    266 def __init__(self, endog, exog=None, **kwargs):
--> 267     super().__init__(endog, exog, **kwargs)
    268     self.initialize()

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels
\base\model.py:92, in Model.__init__(self, endog, exog, **kwargs)
     90 missing = kwargs.pop('missing', 'none')
     91 hasconst = kwargs.pop('hasconst', None)
---> 92 self.data = self._handle_data(endog, exog, missing, hasconst,
     93                                **kwargs)
     94 self.k_constant = self.data.k_constant
     95 self.exog = self.data.exog

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels
\base\model.py:132, in Model._handle_data(self, endog, exog, missing, hascon
st, **kwargs)
    131 def _handle_data(self, endog, exog, missing, hasconst, **kwargs):
--> 132     data = handle_data(endog, exog, missing, hasconst, **kwargs)
    133     # kwargs arrays could have changed, easier to just attach here
    134     for key in kwargs:

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels
\base\data.py:673, in handle_data(endog, exog, missing, hasconst, **kwargs)
    670     exog = np.asarray(exog)
    672 klass = handle_data_class_factory(endog, exog)
--> 673 return klass(endog, exog=exog, missing=missing, hasconst=hasconst,
    674              **kwargs)

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels
\base\data.py:86, in ModelData.__init__(self, endog, exog, missing, hascons
t, **kwargs)
     84 self.const_idx = None
     85 self.k_constant = 0
---> 86 self._handle_constant(hasconst)
     87 self._check_integrity()
     88 self._cache = {}

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels
\base\data.py:131, in ModelData._handle_constant(self, hasconst)
    129 check_implicit = False
    130 exog_max = np.max(self.exog, axis=0)
--> 131 if not np.isfinite(exog_max).all():
    132     raise MissingDataError('exog contains inf or nans')
    133 exog_min = np.min(self.exog, axis=0)

TypeError: ufunc 'isfinite' not supported for the input types, and the input
s could not be safely coerced to any supported types according to the castin
g rule ''safe''
```

In [57]:

```
vif = [variance_inflation_factor(raw_df.tonumpy(), i) for i in range(raw_df.shape[1])]
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [57], in <cell line: 1>()
----> 1 vif = [variance_inflation_factor(raw_df.tonumpy(), i) for i in range
(raw_df.shape[1])]

Input In [57], in <listcomp>(.0)
----> 1 vif = [variance_inflation_factor(raw_df.tonumpy(), i) for i in range
(raw_df.shape[1])]

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core
\generic.py:5575, in NDFrame.__getattr__(self, name)
   5568 if (
   5569     name not in self._internal_names_set
   5570     and name not in self._metadata
   5571     and name not in self._accessors
   5572     and self._info_axis._can_hold_identifiers_and_holds_name(name)
   5573 ):
   5574     return self[name]
-> 5575 return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'tonumpy'
```

In [58]:

```
raw_df.to_numpy()
```

Out[58]:

```
array([['Bream', 242.0, 23.2, ..., 30.0, 11.52, 4.02],
       ['Bream', 290.0, 24.0, ..., 31.2, 12.48, 4.3056],
       ['Bream', 340.0, 23.9, ..., 31.1, 12.3778, 4.6961],
       ...,
       ['Smelt', 12.2, 12.1, ..., 13.8, 2.277, 1.2558],
       ['Smelt', 19.7, 13.2, ..., 15.2, 2.8728, 2.0672],
       ['Smelt', 19.9, 13.8, ..., 16.2, 2.9322, 1.8792]], dtype=object)
```

Detecting Outliers in each column

In [59]:

```
raw_df['Weight']
```

Out[59]:

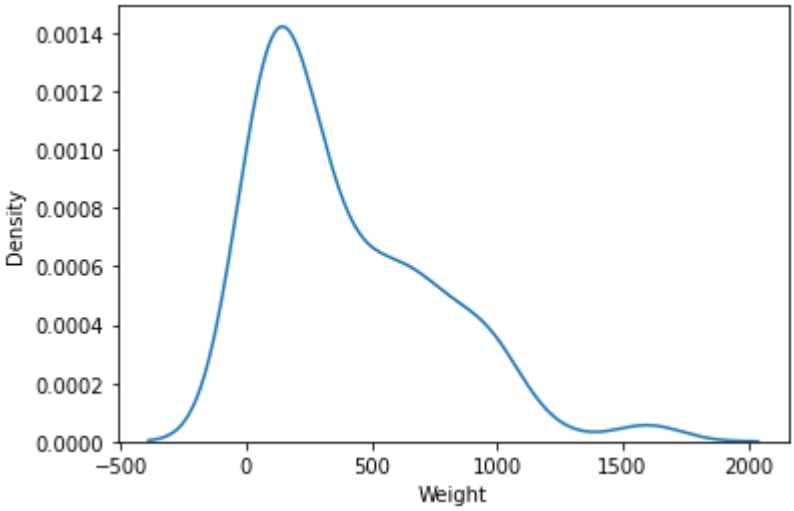
```
0      242.0
1      290.0
2      340.0
3      363.0
4      430.0
       ...
154     12.2
155     13.4
156     12.2
157     19.7
158     19.9
Name: Weight, Length: 159, dtype: float64
```

In [60]:

```
sns.kdeplot(raw_df['Weight'])
```

Out[60]:

```
<AxesSubplot:xlabel='Weight', ylabel='Density'>
```



In [61]:

```
q1 = raw_df['Weight'].quantile(0.25)
q2 = raw_df['Weight'].quantile(0.50)
q3 = raw_df['Weight'].quantile(0.75)

IQR = q3 - q1

upper_tail = q3 + 1.5*IQR
lower_tail = q1 - 1.5*IQR

median = raw_df['Weight'].median()

print('q1', q1)
print('q2', q2)
print('q3', q3)
print("Median", median)
print('upper_tail', upper_tail)
print('lower_tail', lower_tail)
```

```
q1 120.0
q2 273.0
q3 650.0
Median 273.0
upper_tail 1445.0
lower_tail -675.0
```

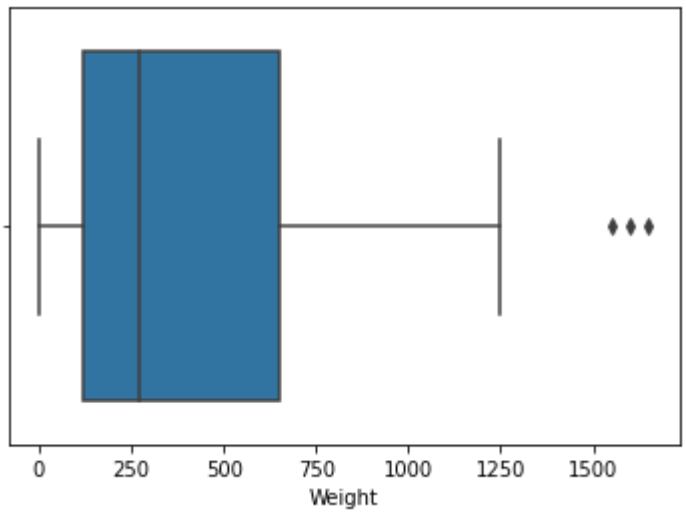
In [62]:

```
sns.boxplot(raw_df['Weight'])
```

```
C:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\sea
born\_decorators.py:36: FutureWarning: Pass the following variable as a keyw
ord arg: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  warnings.warn(
```

Out[62]:

```
<AxesSubplot:xlabel='Weight'>
```



In [63]:

```
raw_df[['Weight']].loc[raw_df['Weight'] > upper_tail ]
```

Out[63]:

	Weight
142	1600.0
143	1550.0
144	1650.0

In [64]:

```
raw_df.loc[raw_df['Weight'] > upper_tail, 'Weight'] = upper_tail
print(raw_df[['Weight']].loc[raw_df['Weight'] > upper_tail ])
```

```
Empty DataFrame
Columns: [Weight]
Index: []
```

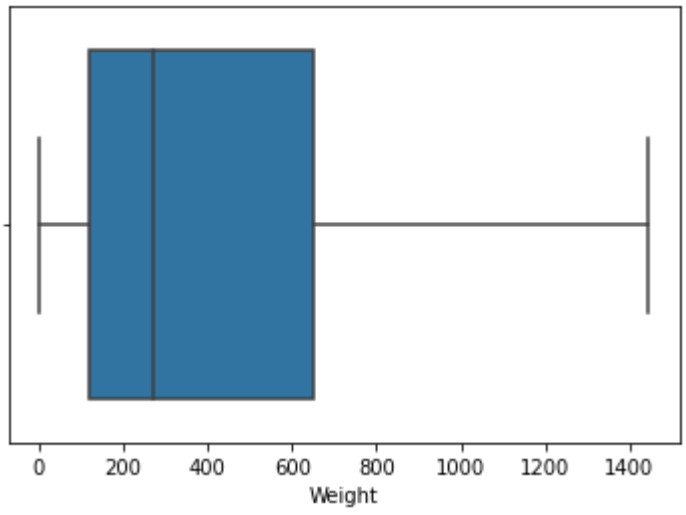
In [65]:

```
sns.boxplot(raw_df['Weight'])
```

```
C:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\sea
born\_decorators.py:36: FutureWarning: Pass the following variable as a keyw
ord arg: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  warnings.warn(
```

Out[65]:

```
<AxesSubplot:xlabel='Weight'>
```



In [72]:

```
raw_df['Length1']
```

Out[72]:

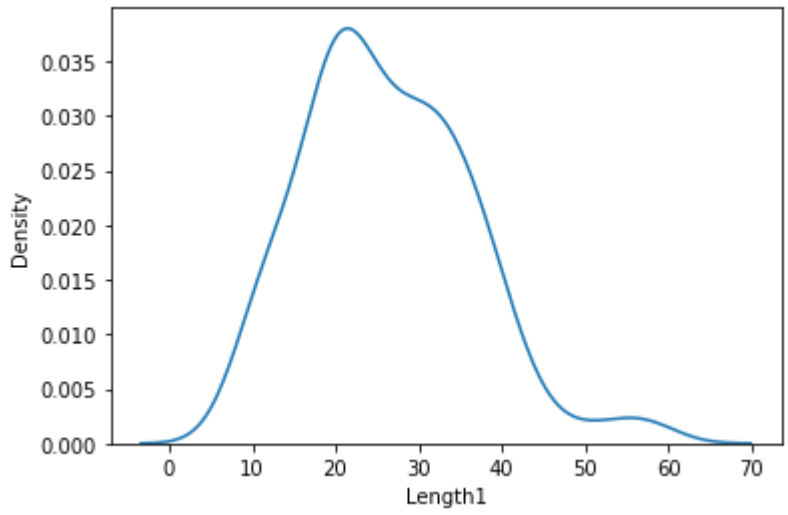
```
0      23.2
1      24.0
2      23.9
3      26.3
4      26.5
       ...
154    11.5
155    11.7
156    12.1
157    13.2
158    13.8
Name: Length1, Length: 159, dtype: float64
```

In [73]:

```
sns.kdeplot(raw_df['Length1'])
```

Out[73]:

```
<AxesSubplot:xlabel='Length1', ylabel='Density'>
```



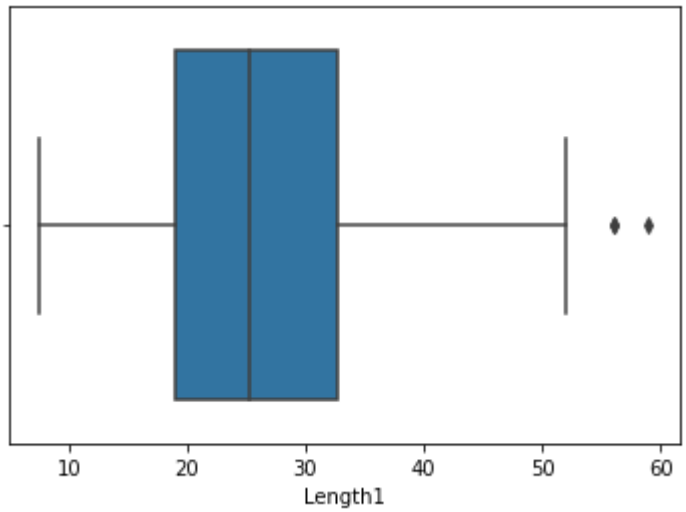
In [74]:

```
sns.boxplot(raw_df['Length1'])
```

```
C:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\sea
born\_decorators.py:36: FutureWarning: Pass the following variable as a keyw
ord arg: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  warnings.warn(
```

Out[74]:

```
<AxesSubplot:xlabel='Length1'>
```



In [75]:

```
q1 = raw_df['Length1'].quantile(0.25)
q2 = raw_df['Length1'].quantile(0.50)
q3 = raw_df['Length1'].quantile(0.75)

IQR = q3 - q1

upper_tail = q3 + 1.5*IQR
lower_tail = q1 - 1.5*IQR

median = raw_df['Length1'].median()

print('q1', q1)
print('q2', q2)
print('q3', q3)
print("Median", median)
print('upper_tail', upper_tail)
print('lower_tail', lower_tail)
```

```
q1 19.05
q2 25.2
q3 32.7
Median 25.2
upper_tail 53.175000000000004
lower_tail -1.4250000000000007
```

In [78]:

```
raw_df[['Length1']].loc[raw_df['Length1'] > upper_tail]
```

Out[78]:

	Length1
142	56.0
143	56.0
144	59.0

In [79]:

```
raw_df.loc[raw_df['Length1'] > upper_tail, 'Length1'] = upper_tail
print(raw_df[['Length1']].loc[raw_df['Length1'] > upper_tail ])
```

```
Empty DataFrame
Columns: [Length1]
Index: []
```

In [80]:

```
raw_df['Length2']
```

Out[80]:

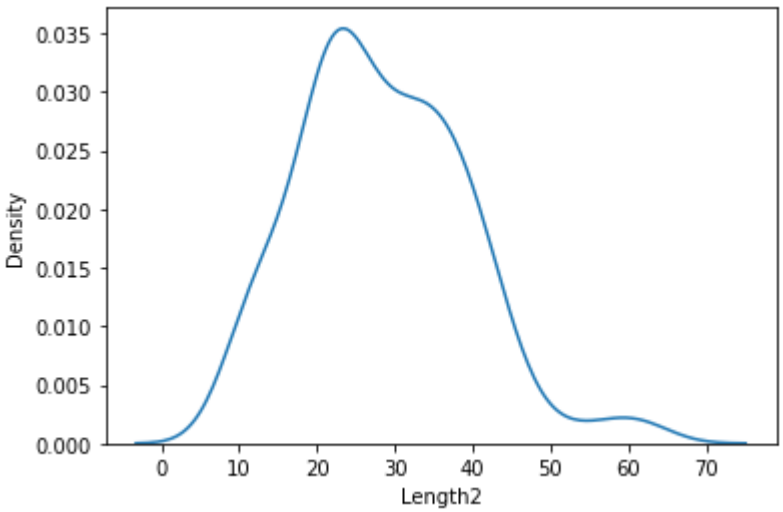
```
0      25.4
1      26.3
2      26.5
3      29.0
4      29.0
       ...
154    12.2
155    12.4
156    13.0
157    14.3
158    15.0
Name: Length2, Length: 159, dtype: float64
```

In [81]:

```
sns.kdeplot(raw_df['Length2'])
```

Out[81]:

```
<AxesSubplot:xlabel='Length2', ylabel='Density'>
```



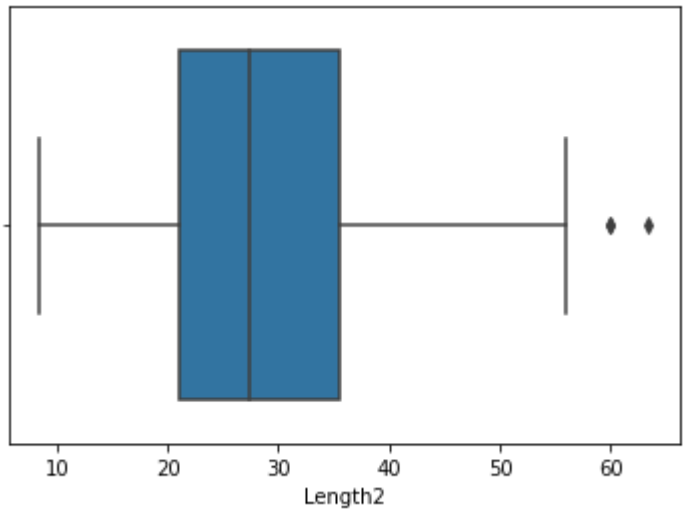
In [82]:

```
sns.boxplot(raw_df['Length2'])
```

```
C:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\sea
born\_decorators.py:36: FutureWarning: Pass the following variable as a keyw
ord arg: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  warnings.warn(
```

Out[82]:

```
<AxesSubplot:xlabel='Length2'>
```



In [83]:

```
q1 = raw_df['Length2'].quantile(0.25)
q2 = raw_df['Length2'].quantile(0.50)
q3 = raw_df['Length2'].quantile(0.75)

IQR = q3 - q1

upper_tail = q3 + 1.5*IQR
lower_tail = q1 - 1.5*IQR

median = raw_df['Length2'].median()

print('q1', q1)
print('q2', q2)
print('q3', q3)
print("Median", median)
print('upper_tail', upper_tail)
print('lower_tail', lower_tail)
```

```
q1 21.0
q2 27.3
q3 35.5
Median 27.3
upper_tail 57.25
lower_tail -0.75
```

In [84]:

```
raw_df[['Length2']].loc[raw_df['Length2'] > upper_tail]
```

Out[84]:

	Length2
142	60.0
143	60.0
144	63.4

In [85]:

```
raw_df.loc[raw_df['Length2'] > upper_tail, 'Length2'] = upper_tail
print(raw_df[['Length2']].loc[raw_df['Length2'] > upper_tail ])
```

```
Empty DataFrame
Columns: [Length2]
Index: []
```

In [86]:

```
raw_df['Length3']
```

Out[86]:

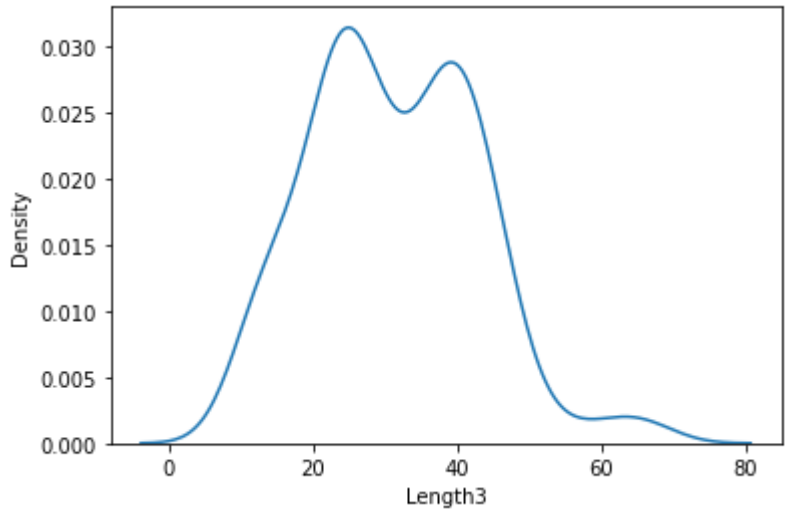
```
0      30.0
1      31.2
2      31.1
3      33.5
4      34.0
       ...
154    13.4
155    13.5
156    13.8
157    15.2
158    16.2
Name: Length3, Length: 159, dtype: float64
```

In [89]:

```
sns.kdeplot(raw_df['Length3'])
```

Out[89]:

```
<AxesSubplot:xlabel='Length3', ylabel='Density'>
```



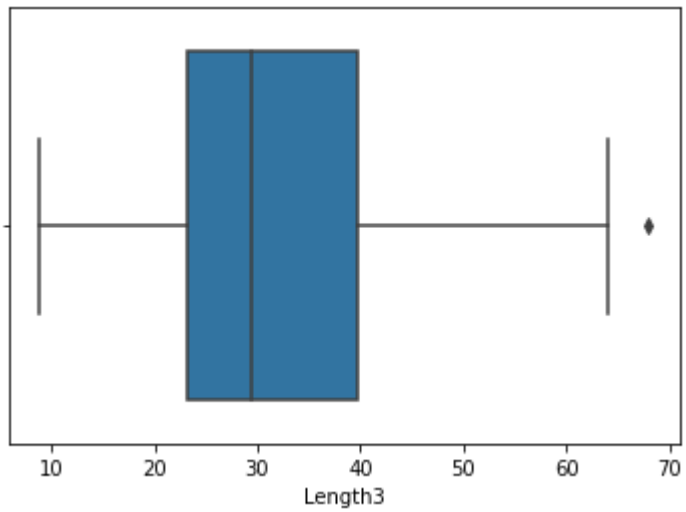
In [90]:

```
sns.boxplot(raw_df['Length3'])
```

```
C:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\sea
born\_decorators.py:36: FutureWarning: Pass the following variable as a keyw
ord arg: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  warnings.warn(
```

Out[90]:

```
<AxesSubplot:xlabel='Length3'>
```



In [91]:

```
q1 = raw_df['Length3'].quantile(0.25)
q2 = raw_df['Length3'].quantile(0.50)
q3 = raw_df['Length3'].quantile(0.75)

IQR = q3 - q1

upper_tail = q3 + 1.5*IQR
lower_tail = q1 - 1.5*IQR

median = raw_df['Length3'].median()

print('q1', q1)
print('q2', q2)
print('q3', q3)
print("Median", median)
print('upper_tail', upper_tail)
print('lower_tail', lower_tail)
```

```
q1 23.15
q2 29.4
q3 39.650000000000006
Median 29.4
upper_tail 64.40000000000002
lower_tail -1.600000000000012
```

In [93]:

```
raw_df[['Length3']].loc[raw_df['Length3'] > upper_tail]
```

Out[93]:

	Length3
144	68.0

In [94]:

```
raw_df.loc[raw_df['Length2'] > upper_tail, 'Length2'] = upper_tail
print(raw_df[['Length2']].loc[raw_df['Length2'] > upper_tail ])
```

```
Empty DataFrame
Columns: [Length2]
Index: []
```

In [95]:

```
raw_df['Width']
```

Out[95]:

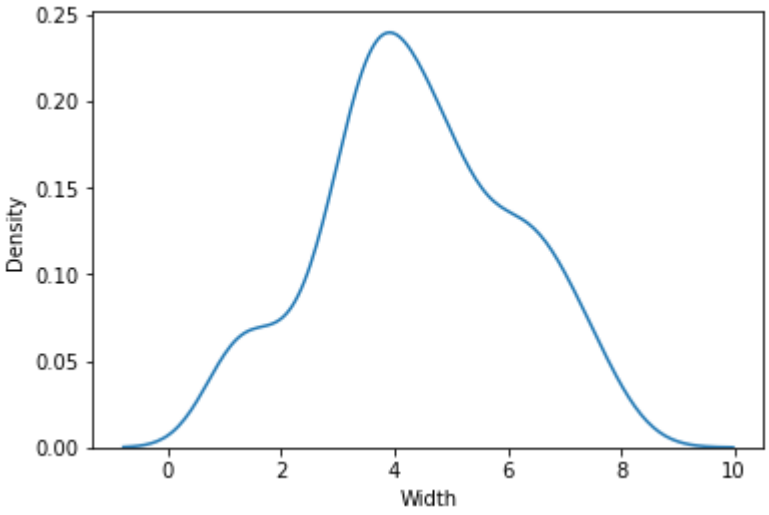
```
0      4.0200
1      4.3056
2      4.6961
3      4.4555
4      5.1340
        ...
154    1.3936
155    1.2690
156    1.2558
157    2.0672
158    1.8792
Name: Width, Length: 159, dtype: float64
```

In [96]:

```
sns.kdeplot(raw_df['Width'])
```

Out[96]:

```
<AxesSubplot:xlabel='Width', ylabel='Density'>
```



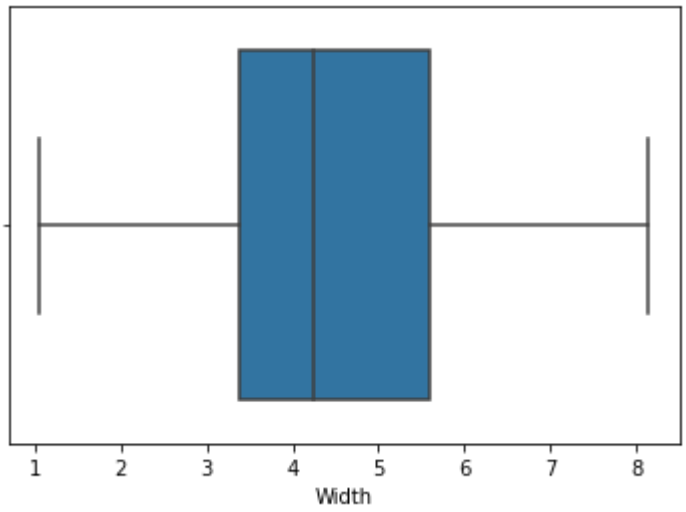
In [97]:

```
sns.boxplot(raw_df['Width'])
```

```
C:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\sea
born\_decorators.py:36: FutureWarning: Pass the following variable as a keyw
ord arg: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  warnings.warn(
```

Out[97]:

```
<AxesSubplot:xlabel='Width'>
```



In [98]:

```
raw_df['Height']
```

Out[98]:

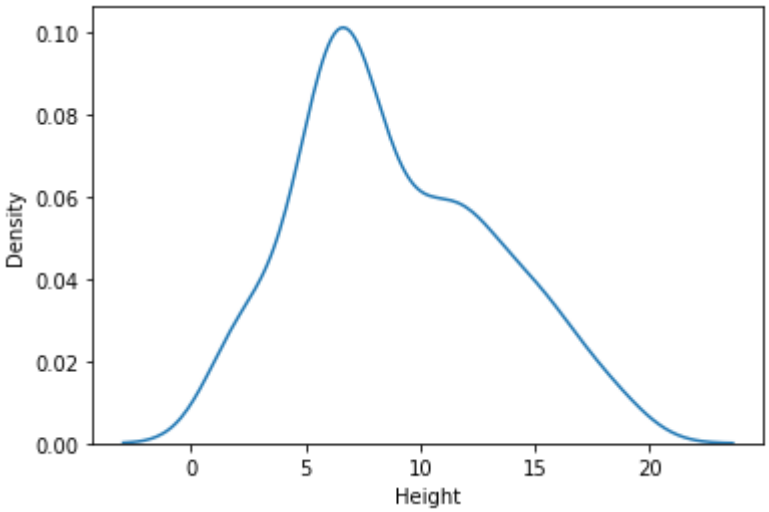
```
0      11.5200
1      12.4800
2      12.3778
3      12.7300
4      12.4440
        ...
154     2.0904
155     2.4300
156     2.2770
157     2.8728
158     2.9322
Name: Height, Length: 159, dtype: float64
```

In [99]:

```
sns.kdeplot(raw_df['Height'])
```

Out[99]:

```
<AxesSubplot:xlabel='Height', ylabel='Density'>
```



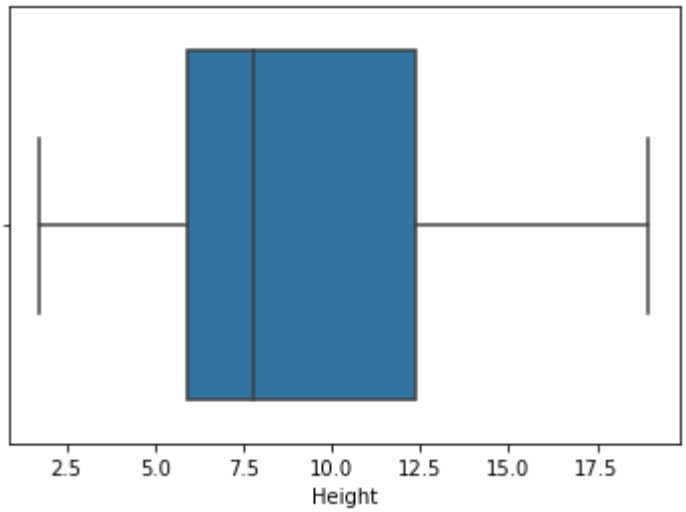
In [100]:

```
sns.boxplot(raw_df['Height'])
```

```
C:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\sea
born\_decorators.py:36: FutureWarning: Pass the following variable as a keyw
ord arg: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  warnings.warn(
```

Out[100]:

```
<AxesSubplot:xlabel='Height'>
```



In [101]:

```
raw_df['Species']
```

Out[101]:

```
0      Bream
1      Bream
2      Bream
3      Bream
4      Bream
       ...
154    Smelt
155    Smelt
156    Smelt
157    Smelt
158    Smelt
Name: Species, Length: 159, dtype: object
```

In [121]:

```
raw_df['Species'].unique().tolist()
```

Out[121]:

```
['Bream', 'Roach', 'Whitefish', 'Parkki', 'Perch', 'Pike', 'Smelt']
```

In [145]:

```
raw_df = pd.get_dummies(raw_df, columns=['Species'])
raw_df
```

Out[145]:

	Weight	Length1	Length2	Length3	Height	Width	Species_Bream	Species_Parkki	Spe
0	242.0	23.2	25.4	30.0	11.5200	4.0200	1	0	
1	290.0	24.0	26.3	31.2	12.4800	4.3056	1	0	
2	340.0	23.9	26.5	31.1	12.3778	4.6961	1	0	
3	363.0	26.3	29.0	33.5	12.7300	4.4555	1	0	
4	430.0	26.5	29.0	34.0	12.4440	5.1340	1	0	
...	...	...	...	...	...	...	...	...	
154	12.2	11.5	12.2	13.4	2.0904	1.3936	0	0	
155	13.4	11.7	12.4	13.5	2.4300	1.2690	0	0	
156	12.2	12.1	13.0	13.8	2.2770	1.2558	0	0	
157	19.7	13.2	14.3	15.2	2.8728	2.0672	0	0	
158	19.9	13.8	15.0	16.2	2.9322	1.8792	0	0	

159 rows × 13 columns

Train test split

In [146]:

```
x = raw_df.drop('Weight', axis = 1)
y = raw_df['Weight']
raw_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159 entries, 0 to 158
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Weight             159 non-null    float64
 1   Length1            159 non-null    float64
 2   Length2            159 non-null    float64
 3   Length3            159 non-null    float64
 4   Height             159 non-null    float64
 5   Width              159 non-null    float64
 6   Species_Bream      159 non-null    uint8  
 7   Species_Parkki     159 non-null    uint8  
 8   Species_Perch      159 non-null    uint8  
 9   Species_Pike       159 non-null    uint8  
 10  Species_Roach      159 non-null    uint8  
 11  Species_Smelt      159 non-null    uint8  
 12  Species_Whitefish  159 non-null    uint8  
dtypes: float64(6), uint8(7)
memory usage: 8.7 KB
```

In [147]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2, random_state = 20
```

In [148]:

```
# Instantiating Linear Regression

linear_reg_model = LinearRegression()
linear_reg_model.fit(x_train, y_train) # Gradient Descent >> best m and c values
```

Out[148]:

LinearRegression

LinearRegression()

Model Evaluation on Testing Data

In [151]:

```
## Model Evaluation on Testing Data

y_pred = linear_reg_model.predict(x_test)

mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error :",mse)

rmse = np.sqrt(mse)
print("Root Mean Squared Error :",rmse)

mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error :",mae)

r_squared_value = r2_score(y_test, y_pred)
print("R Squared Value :",r_squared_value)

adj_r2 = 1 - (((1-r_squared_value) * (x_test.shape[0] - 1))/(x_test.shape[0]- x_test.shape[
print("Adjusted R-Squared Value :",adj_r2)

r2 = linear_reg_model.score(x_test, y_test) # Without Predict Function
print("R2 :",r2)
```

```
Mean Squared Error : 10641.346482529983
Root Mean Squared Error : 103.1569022534604
Mean Absolute Error : 76.334073462639
R Squared Value : 0.9241345503088414
Adjusted R-Squared Value : 0.8762195294512676
R2 : 0.9241345503088414
```

Model Evaluation on Training Data

In [150]:

```
## Model Evaluation on Training Data

y_pred_train = linear_reg_model.predict(x_train) # 404 rows

mse = mean_squared_error(y_train, y_pred_train)
print("Mean Squared Error :",mse)

rmse = np.sqrt(mse)
print("Root Mean Squared Error :",rmse)

mae = mean_absolute_error(y_train, y_pred_train)
print("Mean Absolute Error :",mae)

r_squared_value = r2_score(y_train, y_pred_train)
print("R Squared Value :",r_squared_value)

adj_r2 = 1 - (((1-r_squared_value) * (x_train.shape[0] - 1))/(x_train.shape[0]- x_train.sha
print("Adjusted R-Squared Value :",adj_r2)

r2 = linear_reg_model.score(x_train, y_train)
print("R2 :",r2)
```

```
Mean Squared Error : 7657.374223249367
Root Mean Squared Error : 87.50642389704522
Mean Absolute Error : 63.392048561968075
R Squared Value : 0.9382732908398195
Adjusted R-Squared Value : 0.9317757425071689
R2 : 0.9382732908398195
```

Testing on Single Row

In [153]:

```
x.head(1).T
```

Out[153]:

	0
Length1	23.20
Length2	25.40
Length3	30.00
Height	11.52
Width	4.02
Species_Bream	1.00
Species_Parkki	0.00
Species_Perch	0.00
Species_Pike	0.00
Species_Roach	0.00
Species_Smelt	0.00
Species_Whitefish	0.00

In [167]:

```
Length1 = 23.20
Length2 = 25.40
Length3 = 30.00
Height = 11.52
Width = 4.02
Species = 'Parkki'


# Weight = ?
```

In [168]:

```
project_data = {'columns' : list(x.columns)}
```

In [169]:

```
column_names = x.columns
column_names
```

Out[169]:

```
Index(['Length1', 'Length2', 'Length3', 'Height', 'Width', 'Species_Bream',
       'Species_Parkki', 'Species_Perch', 'Species_Pike', 'Species_Roach',
       'Species_Smelt', 'Species_Whitefish'],
      dtype='object')
```

In [170]:

```
Species =  'Species_' + Species
Species_index = np.where(column_names == Species)
```

In [171]:

```
test_array = np.zeros(x.shape[1])
test_array[0] = Length1
test_array[1] = Length2
test_array[2] = Length3
test_array[3] = Height
test_array[4] = Width
test_array[Species_index] = 1

test_array
```

Out[171]:

```
array([23.2 , 25.4 , 30.  , 11.52,  4.02,  0.  ,  1.  ,  0.  ,  0.  ,
        0.  ,  0.  ,  0.  ])
```

In [173]:

```
linear_reg_model.predict([test_array])
```

```
C:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\skl
earn\base.py:450: UserWarning: X does not have valid feature names, but Line
arRegression was fitted with feature names
  warnings.warn(
```

Out[173]:

```
array([444.90795215])
```