In [3]:
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [4]:
```python
data=pd.read_csv('possum.csv')
```

In [5]:
```python
data.head()
```

Out[5]:

	case	site	Pop	sex	age	hdlngth	skullw	totlngth	taill	footlgth	earconch	eye	chest	be
0	1	1	Vic	m	8.0	94.1	60.4	89.0	36.0	74.5	54.5	15.2	28.0	36
1	2	1	Vic	f	6.0	92.5	57.6	91.5	36.5	72.5	51.2	16.0	28.5	33
2	3	1	Vic	f	6.0	94.0	60.0	95.5	39.0	75.4	51.9	15.5	30.0	34
3	4	1	Vic	f	6.0	93.2	57.1	92.0	38.0	76.1	52.2	15.2	28.0	34
4	5	1	Vic	f	2.0	91.5	56.3	85.5	36.0	71.0	53.2	15.1	28.5	33

In [6]:
```python
data.shape
```

Out[6]:
```
(104, 14)
```

In [7]:
```python
data.columns
```

Out[7]:
```
Index(['case', 'site', 'Pop', 'sex', 'age', 'hdlngth', 'skullw', 'totlngth',
       'taill', 'footlgth', 'earconch', 'eye', 'chest', 'belly'],
      dtype='object')
```

In [8]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 104 entries, 0 to 103
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   case      104 non-null    int64  
 1   site      104 non-null    int64  
 2   Pop       104 non-null    object 
 3   sex       104 non-null    object 
 4   age       102 non-null    float64
 5   hdlngth   104 non-null    float64
 6   skullw    104 non-null    float64
 7   totlngth  104 non-null    float64
 8   taill     104 non-null    float64
 9   footlgth  103 non-null    float64
 10  earconch  104 non-null    float64
 11  eye       104 non-null    float64
 12  chest     104 non-null    float64
 13  belly     104 non-null    float64
dtypes: float64(10), int64(2), object(2)
memory usage: 11.5+ KB
```

In [9]:
```
data.describe()
```

Out[9]:

	case	site	age	hdlngth	skullw	totlngth	taill	fo
count	104.000000	104.000000	102.000000	104.000000	104.000000	104.000000	104.000000	103.0
mean	52.500000	3.625000	3.833333	92.602885	56.883654	87.088462	37.009615	68.4
std	30.166206	2.349086	1.909244	3.573349	3.113426	4.310549	1.959518	4.3
min	1.000000	1.000000	1.000000	82.500000	50.000000	75.000000	32.000000	60.3
25%	26.750000	1.000000	2.250000	90.675000	54.975000	84.000000	35.875000	64.6
50%	52.500000	3.000000	3.000000	92.800000	56.350000	88.000000	37.000000	68.0
75%	78.250000	6.000000	5.000000	94.725000	58.100000	90.000000	38.000000	72.5
max	104.000000	7.000000	9.000000	103.100000	68.600000	96.500000	43.000000	77.9

In [10]:
```
pd.isnull(data).sum()
```

Out[10]:
```
case        0
site        0
Pop         0
sex         0
age         2
hdlngth     0
skullw      0
totlngth    0
taill       0
footlgth    1
earconch    0
eye         0
chest       0
belly       0
dtype: int64
```

In [11]:
```
data.value_counts()
```

Out[11]:
```
case  site  Pop    sex  age  hdlngth  skullw  totlngth  taill  footlgth  earc
onch  eye   chest  belly
1     1     Vic    m    8.0  94.1     60.4    89.0      36.0   74.5      54.5
15.2  28.0   36.0     1
68    5     other  m    5.0  94.1     56.0    88.5      38.0   65.9      43.1
17.4  27.0   30.0     1
78    6     other  m    1.0  86.5     51.0    81.0      36.5   63.0      44.3
13.2  23.0   28.0     1
77    6     other  m    6.0  93.6     56.2    84.0      36.0   62.8      42.9
16.2  25.0   35.0     1
76    6     other  m    5.0  92.4     56.8    89.0      41.0   64.5      46.4
17.8  26.0   33.0     1

..
32    1     Vic    f    4.0  94.3     56.7    94.0      39.0   74.8      52.0
14.9  28.0   34.0     1
31    1     Vic    m    3.0  92.8     54.1    93.0      37.0   68.0      52.5
14.5  27.0   31.0     1
30    1     Vic    f    2.0  92.1     54.4    84.0      33.5   70.6      50.8
14.5  24.5   33.0     1
29    1     Vic    f    3.0  92.8     56.0    88.0      35.0   74.9      51.8
14.0  24.0   32.0     1
104   7     other  f    3.0  93.6     59.9    89.0      40.0   67.6      46.0
14.8  28.5   33.5     1
Name: count, Length: 101, dtype: int64
```

In [12]:
```
data.dropna(inplace=True)
```

In [13]:
```
pd.isnull(data).sum()
```

Out[13]:
```
case        0
site        0
Pop         0
sex         0
age         0
hdlngth     0
skullw      0
totlngth    0
taill       0
footlgth    0
earconch    0
eye         0
chest       0
belly       0
dtype: int64
```

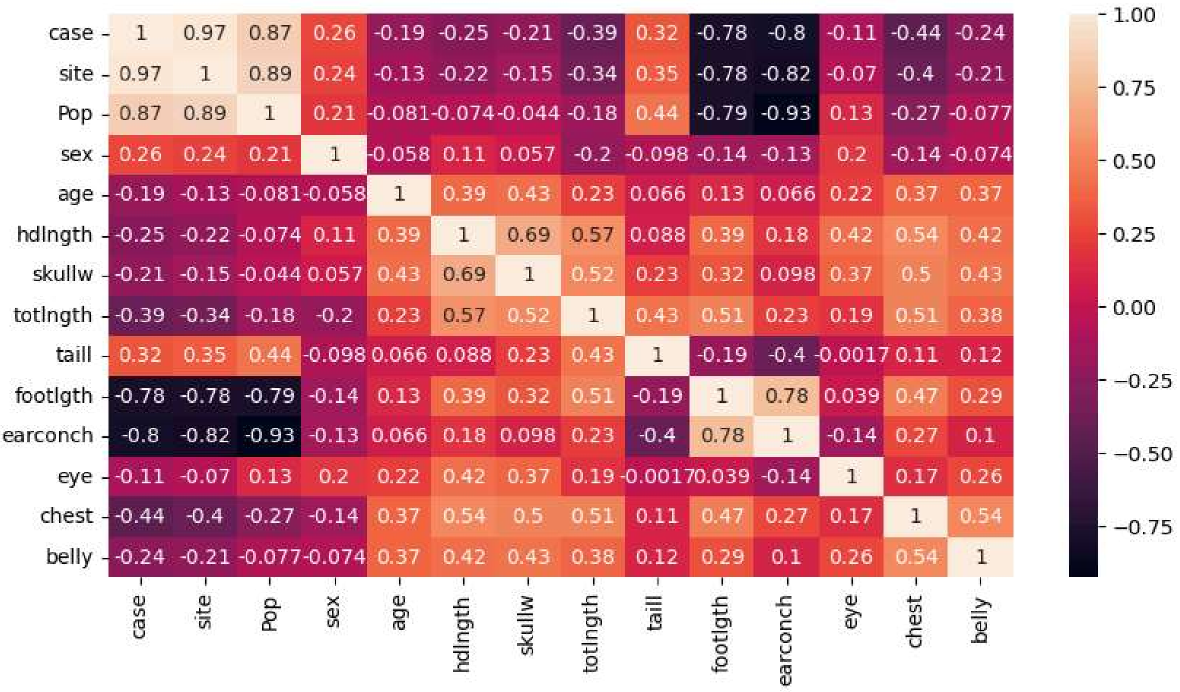
In [14]:
```
data
```

Out[14]:
```
     case  site    Pop  sex  age  hdlngth  skullw  totlngth  taill  footlgth  earconch   eye  chest
  0     1     1    Vic    m  8.0     94.1    60.4      89.0   36.0      74.5      54.5  15.2   28.0
  1     2     1    Vic    f  6.0     92.5    57.6      91.5   36.5      72.5      51.2  16.0   28.5
  2     3     1    Vic    f  6.0     94.0    60.0      95.5   39.0      75.4      51.9  15.5   30.0
  3     4     1    Vic    f  6.0     93.2    57.1      92.0   38.0      76.1      52.2  15.2   28.0
  4     5     1    Vic    f  2.0     91.5    56.3      85.5   36.0      71.0      53.2  15.1   28.5
...   ...   ...    ...  ...  ...      ...     ...       ...    ...       ...       ...   ...    ...
 99   100     7  other    m  1.0     89.5    56.0      81.5   36.5      66.0      46.8  14.8   23.0
100   101     7  other    m  1.0     88.6    54.7      82.5   39.0      64.4      48.0  14.0   25.0
101   102     7  other    f  6.0     92.4    55.0      89.0   38.0      63.5      45.4  13.0   25.0
102   103     7  other    m  4.0     91.5    55.2      82.5   36.5      62.9      45.9  15.4   25.0
103   104     7  other    f  3.0     93.6    59.9      89.0   40.0      67.6      46.0  14.8   28.5

101 rows × 14 columns
```

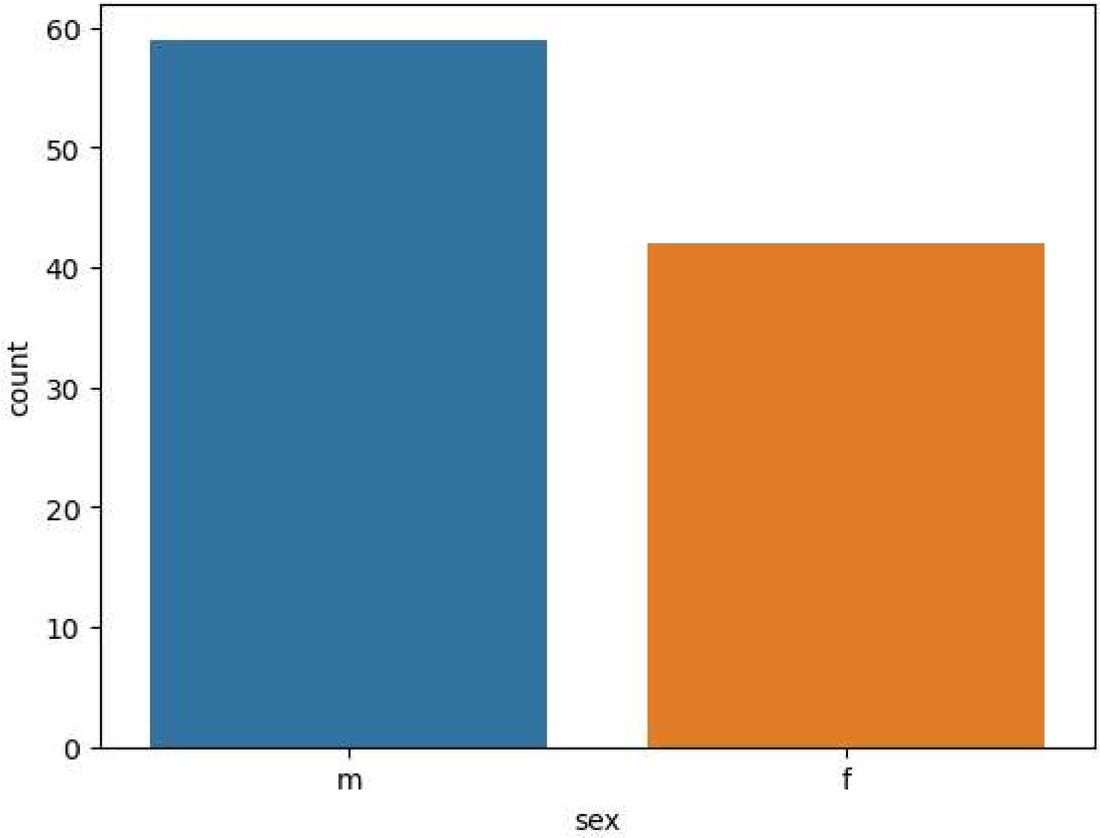
In [65]:
```
plt.figure(figsize=(10,5))
sns.heatmap(data.corr(),annot=True)
```

Out[65]:
```
<Axes: >
```



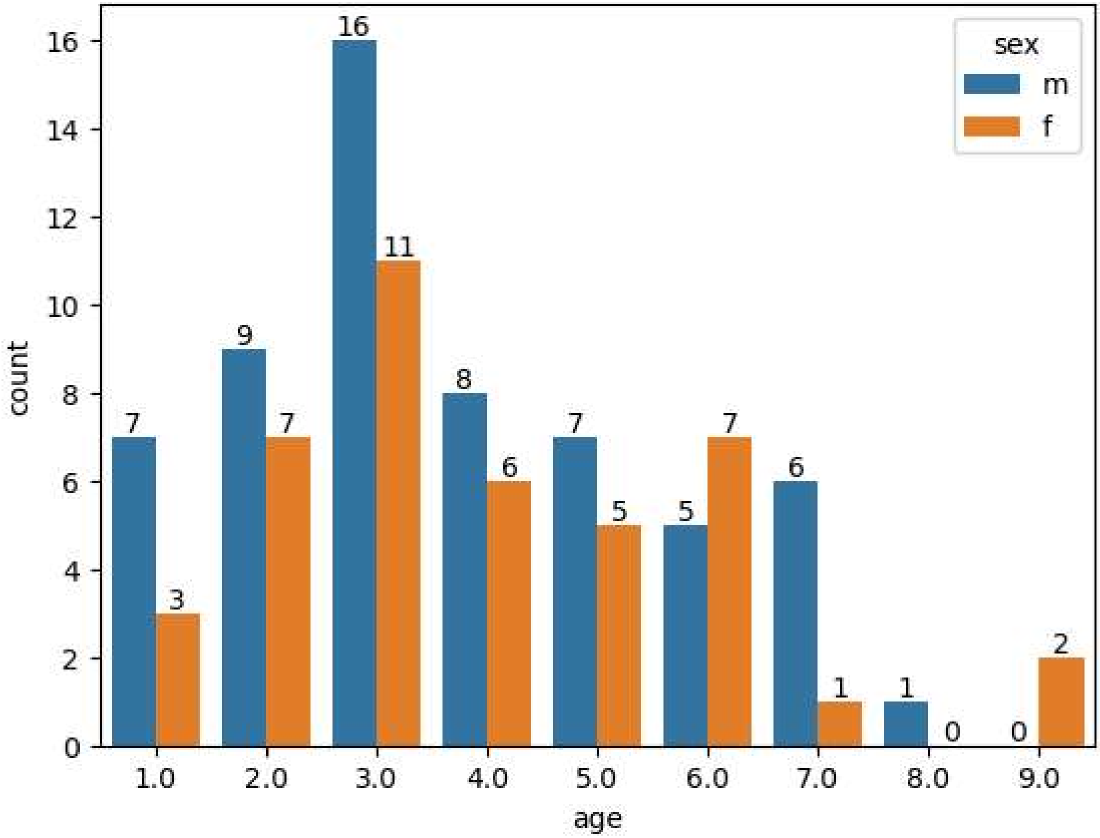
In [15]:

```
ax=sns.countplot(data=data, x='sex')
plt.show()
```



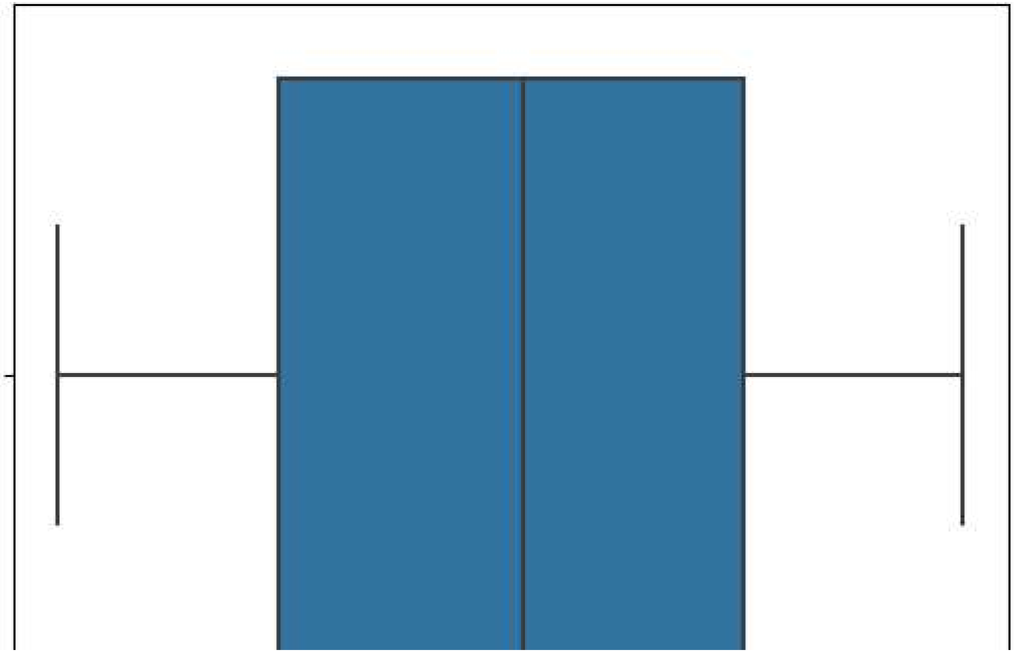
In [16]:

```
ax=sns.countplot(data=data, x='age', hue='sex')
for bars in ax.containers:
 ax.bar_label(bars)
plt.show()
```



In [17]:

```
for i in data.select_dtypes(['int','float']):
 sns.boxplot(data=data,x=i)
 plt.show()
```

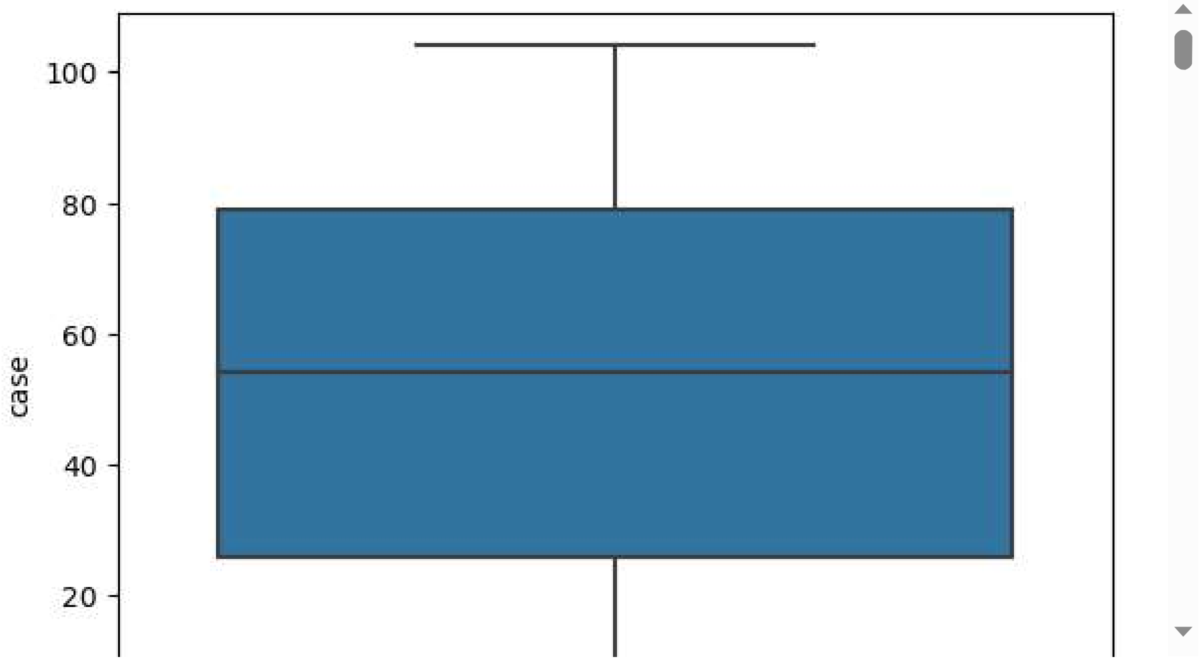


Outliers_treatment

In [18]:
```
def outliers_treatment(col):
 Q1=data[col].quantile(0.25)
 Q3=data[col].quantile(0.75)
 IQR= Q3 - Q1
 UB=Q3+1.5*IQR
 LB=Q1-1.5*IQR
 Upper_Outlier=data[col]>UB
 Lower_Outlier=data[col]<LB
 data.loc[Upper_Outlier,col]=data[col].median()
 data.loc[Lower_Outlier,col]=data[col].median()
```

In [19]:
```
for i in data.select_dtypes(['int','float']):
 outliers_treatment(i)
```

In [20]:
```
for i in data.select_dtypes(['int','float']):
 sns.boxplot(data=data,y=i)
 plt.show()
```



In [21]:
```
data.head()
```

Out[21]:

	case	site	Pop	sex	age	hdlngth	skullw	totlngth	taill	footlgth	earconch	eye	chest	be
0	1	1	Vic	m	8.0	94.1	60.4	89.0	36.0	74.5	54.5	15.2	28.0	36
1	2	1	Vic	f	6.0	92.5	57.6	91.5	36.5	72.5	51.2	16.0	28.5	33
2	3	1	Vic	f	6.0	94.0	60.0	95.5	39.0	75.4	51.9	15.5	30.0	34
3	4	1	Vic	f	6.0	93.2	57.1	92.0	38.0	76.1	52.2	15.2	28.0	34
4	5	1	Vic	f	2.0	91.5	56.3	85.5	36.0	71.0	53.2	15.1	28.5	33

Encoding

In [26]:
```
from sklearn.preprocessing import LabelEncoder
```

In [27]:
```
LE=LabelEncoder()
```

In [28]:
```
LE
```

Out[28]:
```
LabelEncoder()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [29]:
```
def HCD(col):
 data[col]=LE.fit_transform(data[col])
```

In [30]:
```
for i in data.select_dtypes(include=['object']):
 HCD(i)
```

In [31]:
```
data.head()
```

Out[31]:

	case	site	Pop	sex	age	hdlngth	skullw	totlngth	taill	footlgth	earconch	eye	chest	be
0	1	1	0	1	8.0	94.1	60.4	89.0	36.0	74.5	54.5	15.2	28.0	36
1	2	1	0	0	6.0	92.5	57.6	91.5	36.5	72.5	51.2	16.0	28.5	33
2	3	1	0	0	6.0	94.0	60.0	95.5	39.0	75.4	51.9	15.5	30.0	34
3	4	1	0	0	6.0	93.2	57.1	92.0	38.0	76.1	52.2	15.2	28.0	34
4	5	1	0	0	2.0	91.5	56.3	85.5	36.0	71.0	53.2	15.1	28.5	33

In [32]:
```
data.describe()
```

Out[32]:
```
             case        site         Pop         sex         age     hdlngth      skullw   tc
count  101.000000  101.000000  101.000000  101.000000  101.000000  101.000000  101.000000  101.(
mean    52.762376    3.673267    0.574257    0.584158    3.821782   92.637624   56.450495   87.3
std     30.573894    2.366892    0.496921    0.495325    1.915182    3.049126    2.094737    4.(
min      1.000000    1.000000    0.000000    0.000000    1.000000   84.700000   51.000000   77.(
25%     26.000000    1.000000    0.000000    0.000000    2.000000   91.000000   55.200000   84.5
50%     54.000000    4.000000    1.000000    1.000000    3.000000   92.900000   56.400000   88.(
75%     79.000000    6.000000    1.000000    1.000000    5.000000   94.500000   57.700000   90.(
max    104.000000    7.000000    1.000000    1.000000    9.000000   99.900000   61.500000   96.5
```

In [66]:
```
X=data.drop('taill',axis=1)
```

In [67]:
```
X
```

Out[67]:
```
     case  site  Pop  sex  age  hdlngth  skullw  totlngth  footlgth  earconch   eye  chest  belly
0       1     1    0    1  8.0     94.1    60.4      89.0      74.5      54.5  15.2   28.0   36.0
1       2     1    0    0  6.0     92.5    57.6      91.5      72.5      51.2  16.0   28.5   33.0
2       3     1    0    0  6.0     94.0    60.0      95.5      75.4      51.9  15.5   30.0   34.0
3       4     1    0    0  6.0     93.2    57.1      92.0      76.1      52.2  15.2   28.0   34.0
4       5     1    0    0  2.0     91.5    56.3      85.5      71.0      53.2  15.1   28.5   33.0
...   ...   ...  ...  ...  ...      ...     ...       ...       ...       ...   ...    ...    ...
99    100     7    1    1  1.0     89.5    56.0      81.5      66.0      46.8  14.8   23.0   27.0
100   101     7    1    1  1.0     88.6    54.7      82.5      64.4      48.0  14.0   25.0   33.0
101   102     7    1    0  6.0     92.4    55.0      89.0      63.5      45.4  13.0   25.0   30.0
102   103     7    1    1  4.0     91.5    55.2      82.5      62.9      45.9  15.4   25.0   29.0
103   104     7    1    0  3.0     93.6    59.9      89.0      67.6      46.0  14.8   28.5   33.5

101 rows × 13 columns
```

In [68]:
```
data
```

Out[68]:

	case	site	Pop	sex	age	hdlngth	skullw	totlngth	taill	footlgth	earconch	eye	chest
0	1	1	0	1	8.0	94.1	60.4	89.0	36.0	74.5	54.5	15.2	28.0
1	2	1	0	0	6.0	92.5	57.6	91.5	36.5	72.5	51.2	16.0	28.5
2	3	1	0	0	6.0	94.0	60.0	95.5	39.0	75.4	51.9	15.5	30.0
3	4	1	0	0	6.0	93.2	57.1	92.0	38.0	76.1	52.2	15.2	28.0
4	5	1	0	0	2.0	91.5	56.3	85.5	36.0	71.0	53.2	15.1	28.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...
99	100	7	1	1	1.0	89.5	56.0	81.5	36.5	66.0	46.8	14.8	23.0
100	101	7	1	1	1.0	88.6	54.7	82.5	39.0	64.4	48.0	14.0	25.0
101	102	7	1	0	6.0	92.4	55.0	89.0	38.0	63.5	45.4	13.0	25.0
102	103	7	1	1	4.0	91.5	55.2	82.5	36.5	62.9	45.9	15.4	25.0
103	104	7	1	0	3.0	93.6	59.9	89.0	40.0	67.6	46.0	14.8	28.5

101 rows × 14 columns

In [69]:
```
y=data.taill
```

In [70]:
```
y
```

Out[70]:
```
0      36.0
1      36.5
2      39.0
3      38.0
4      36.0
       ... 
99     36.5
100    39.0
101    38.0
102    36.5
103    40.0
Name: taill, Length: 101, dtype: float64
```

Train_Test_Split

In [71]:
```
from sklearn.model_selection import train_test_split
```

In [87]:
```
x_train,x_test,y_train,y_test=train_test_split(X,y,train_size=0.70,random_state
```

In [88]:

```
y
```

Out[88]:

```
0      36.0
1      36.5
2      39.0
3      38.0
4      36.0
       ...
99     36.5
100    39.0
101    38.0
102    36.5
103    40.0
Name: taill, Length: 101, dtype: float64
```

LinearRegression

In [89]:

```
from sklearn.linear_model import LinearRegression,Lasso,Ridge
```

In [90]:

```
LR=LinearRegression()
```

In [91]:

```
LR
```

Out[91]:

```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [92]:

```
LR.fit(x_train,y_train)
```

Out[92]:

```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [93]:

```
LR_pred=LR.predict(x_test)
```

In [94]:

```
LR_pred[:5]
```

Out[94]:

```
array([36.35840531, 37.08933477, 36.8646217 , 35.67548074, 37.5810423 ])
```

```
In [95]: y_test[:5]
```

```
Out[95]: 83    36.0
         78    36.5
         66    36.5
         15    34.5
         38    34.0
         Name: taill, dtype: float64
```

```
In [96]: LR.score(x_train,y_train)
```

```
Out[96]: 0.5679271411932616
```

```
In [97]: LR.score(x_test,y_test)
```

```
Out[97]: 0.4678861278447136
```

```
In [98]: from sklearn.metrics import r2_score
```

```
In [99]: r2_score(y_test,LR_pred)
```

```
Out[99]: 0.4678861278447136
```

```
In [100]: r2=r2_score(y_test,LR_pred)
```

```
In [101]: r2
```

```
Out[101]: 0.4678861278447136
```

Lasso Regression

```
In [102]: LA=Lasso(alpha=5)
```

```
In [103]: LA
```

```
Out[103]: Lasso(alpha=5)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [104]:
```
LA.fit(x_train,y_train)
```

Out[104]:
```
Lasso(alpha=5)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [105]:
```
LA_pred=L1.predict(x_test)
```

In [106]:
```
LA_pred
```

Out[106]:
```
array([37.37598045, 37.32729988, 37.21046651, 36.7139247 , 36.93785532,
       36.67498024, 36.96706366, 37.03521646, 36.94759143, 37.2007304 ,
       37.41492491, 36.60682744, 37.05468869, 36.83075807, 37.33703599,
       36.73339693, 37.44413325, 36.88917475, 36.65550801, 37.50254993,
       37.29809154, 37.23967485, 37.57070273, 37.25914708, 36.82102195,
       37.2688832 , 37.35650822, 36.58735522, 37.34677211, 37.38571656,
       36.76260527])
```

In [107]:
```
LA.score(x_train,y_train)
```

Out[107]:
```
0.06819570129753028
```

In [108]:
```
LA.score(x_test,y_test)
```

Out[108]:
```
0.10425380864549905
```

In [109]:
```
LA.coef_
```

Out[109]:
```
array([ 0.00973611,  0.        ,  0.        , -0.        ,  0.        ,
        0.        ,  0.        ,  0.        , -0.        , -0.        ,
        0.        ,  0.        ,  0.        ])
```

In [110]:
```
LA.intercept_
```

Out[110]:
```
36.558146873188235
```

In [111]:
```
from sklearn.metrics import r2_score
```

In [113]:
```
r2=r2_score(y_test,LA_pred)
```

In [114]:
```
r2
```

Out[114]:
```
0.10425380864549905
```

```
In [115]: adjusted_r2=1-((1-r2)*(len(data)-1)/(len(data)-80-1))
```

```
In [116]: adjusted_r2
```

```
Out[116]: -3.4787309567725044
```

Ridge Regression

```
In [117]: l2=Ridge(alpha=10)
```

```
In [118]: l2.fit(x_train,y_train)
```

```
Out[118]: Ridge(alpha=10)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [119]: l2_pred=l2.predict(x_test)
```

```
In [120]: l2_pred
```

```
Out[120]: array([36.45173509, 37.14100834, 36.88677323, 35.7488068 , 37.63107464,
                 37.1051278 , 34.59908081, 36.31900786, 35.91031699, 37.81842762,
                 38.78427032, 35.45968928, 36.62334348, 35.58918232, 37.49867681,
                 36.05067364, 37.36507418, 35.70360251, 36.85545216, 37.58608006,
                 38.16312899, 38.22688742, 38.82933325, 36.89231847, 35.82086369,
                 35.53736045, 37.72580104, 38.37435421, 37.8102911 , 39.9997276 ,
                 38.36433345])
```

```
In [121]: l2.score(x_train,y_train)
```

```
Out[121]: 0.5603370276443359
```

```
In [122]: l2.score(x_test,y_test)
```

```
Out[122]: 0.472383070733693
```

```
In [123]: l2.coef_
```

```
Out[123]: array([ 0.01477082, -0.00624274,  0.11440015, -0.21342006,  0.01732292,
                 -0.17692705,  0.06651803,  0.29915226, -0.02115113, -0.11115813,
                 -0.08231434,  0.03916063,  0.00233735])
```

```
In [125]: l2.intercept_
```

```
Out[125]: 29.688851934399292
```

In [ ]: