MST UID:18BCS6074

```
# Supress Warnings
import warnings
warnings.filterwarnings('ignore')
```

```
# Uploading files to drive
from google.colab import files
uploaded = files.upload()
```

Choose Files A2.csv

- **A2.csv**(application/vnd.ms-excel) - 2643 bytes, last modified: 10/29/2020 - 100% done

```
Saving A2.csv to A2.csv
```

```
# Importing necessary libraries
import numpy as np
import pandas as pd
```

```
# Importing csv file
df = pd.read_csv('A2.csv', encoding='unicode_escape')
df.head()
```

	Head Size(cm^3)	Brain Weight(grams)
0	4512	1530
1	3738	1297
2	4261	1335
3	3777	1282
4	4177	1590

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 237 entries, 0 to 236
Data columns (total 2 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Head Size(cm^3)      237 non-null    int64
 1   Brain Weight(grams)  237 non-null    int64
dtypes: int64(2)
memory usage: 3.8 KB
```

```
df.describe()
```

	Head Size(cm^3)	Brain Weight(grams)
count	237.000000	237.000000
mean	3633.991561	1282.873418
std	365.261422	120.340446
min	2720.000000	955.000000
25%	3389.000000	1207.000000
50%	3614.000000	1280.000000
75%	3876.000000	1350.000000

```
df.shape
```

```
(237, 2)
```

1. Check for missing values

```
# Looking for null values
round(df.isnull().sum()/len(df.index)*100,2)
```

```
Head Size(cm^3)        0.0
Brain Weight(grams)    0.0
dtype: float64
```

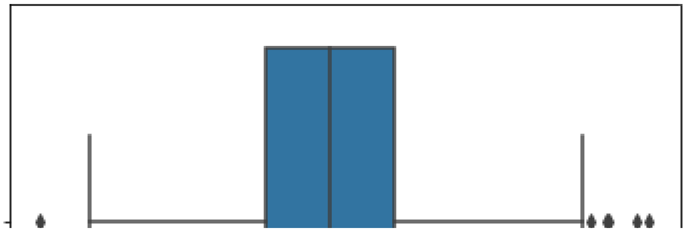
2. Checking for outliers

```
X = df.iloc[:,:-1].values
y = df.iloc[:,1].values
```

```
# Importing libraries for plotting
import matplotlib.pyplot as plt
import seaborn as sns
```

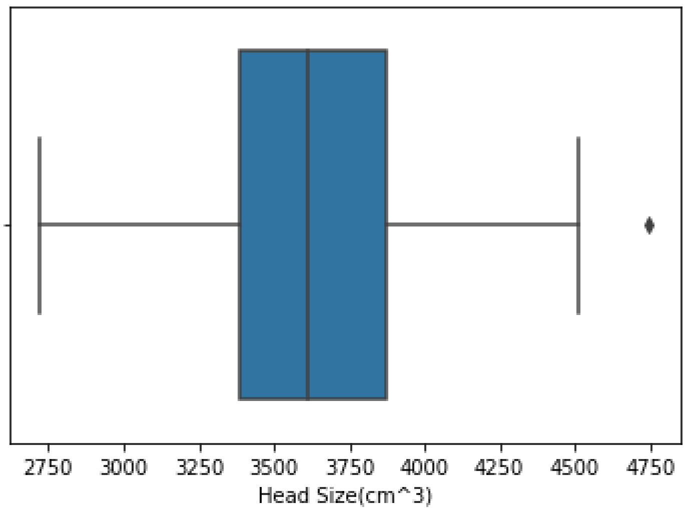
```
# Boxplot
sns.boxplot(df['Brain Weight(grams)'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f11846dc9e8>
```



```
sns.boxplot(df['Head Size(cm^3)'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1182975a20>
```




```
col = df.select_dtypes(include=['int64'])
```

```
for x in col:
    ten = df[x].quantile(0.10)
    nin = df[x].quantile(0.90)
    df[x] = np.where(df[x] < ten, ten,df[x])
    df[x] = np.where(df[x] > nin, nin,df[x])
```

```
q1 = df.quantile(0.25)
q3 = df.quantile(0.75)
IQR = q3 - q1
print(IQR)
```

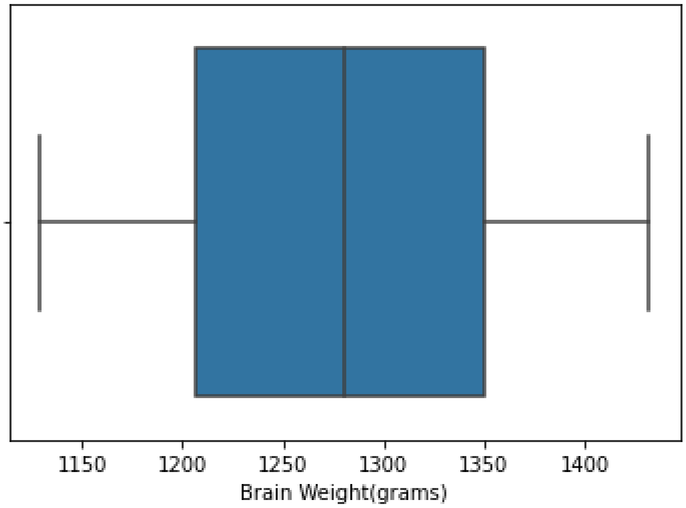
```
Head Size(cm^3)        487.0
Brain Weight(grams)    143.0
dtype: float64
```

```
df_out = df[~((df < (q1 - 1.5 * IQR)) |(df > (q3 + 1.5 * IQR))).any(axis=1)]
print(df_out.shape)
```

```
(237, 2)
```

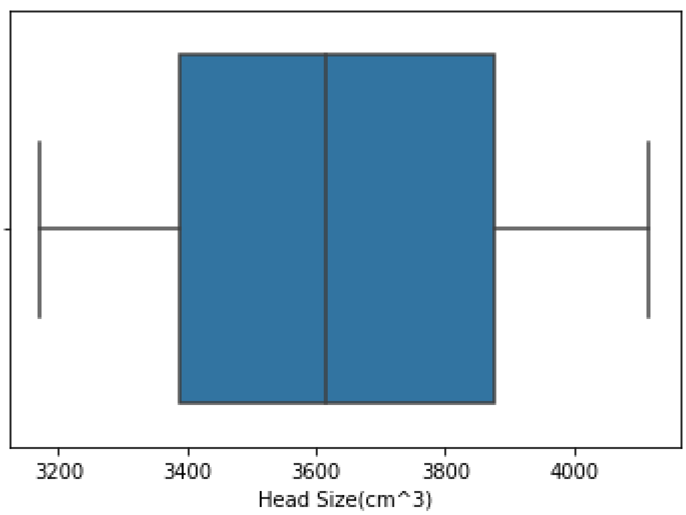
```
sns.boxplot(df['Brain Weight(grams)'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f11829673c8>
```




```
sns.boxplot(df['Head Size(cm^3)'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f11829212b0>
```




```
X = df_out.iloc[:,:-1].values
y = df_out.iloc[:,1].values
```

3. Model Building.

```
from sklearn.linear_model import LinearRegression
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.31,random_state=101)
```

```
lm = LinearRegression()
lm.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
import statsmodels.api as sm
lm = sm.OLS(y_train,X_train).fit()
```

```
print(lm.summary())
```

```
                                 OLS Regression Results
=======================================================================================
Dep. Variable:                      y   R-squared (uncentered):                   0.997
Model:                            OLS   Adj. R-squared (uncentered):              0.997
Method:                 Least Squares   F-statistic:                          6.030e+04
Date:                Thu, 29 Oct 2020   Prob (F-statistic):                  2.95e-210
Time:                        06:35:25   Log-Likelihood:                        -915.18
No. Observations:                 163   AIC:                                     1832.
Df Residuals:                     162   BIC:                                     1835.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             0.3531      0.001    245.554      0.000       0.350       0.356
==============================================================================
Omnibus:                        1.166   Durbin-Watson:                   2.127
Prob(Omnibus):                  0.558   Jarque-Bera (JB):                0.781
Skew:                           0.068   Prob(JB):                        0.677
Kurtosis:                       3.311   Cond. No.                         1.00
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
```

```
import statsmodels.api as sm
lm = sm.OLS(y_test,X_test).fit()
```

```
print(lm.summary())
```

```
                                 OLS Regression Results
=======================================================================================
Dep. Variable:                      y   R-squared (uncentered):                   0.997
Model:                            OLS   Adj. R-squared (uncentered):              0.997
Method:                 Least Squares   F-statistic:                          2.627e+04
Date:                Thu, 29 Oct 2020   Prob (F-statistic):                   4.22e-95
Time:                        06:54:51   Log-Likelihood:                        -416.85
No. Observations:                  74   AIC:                                     835.7
Df Residuals:                      73   BIC:                                     838.0
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             0.3499      0.002    162.088      0.000       0.346       0.354
```

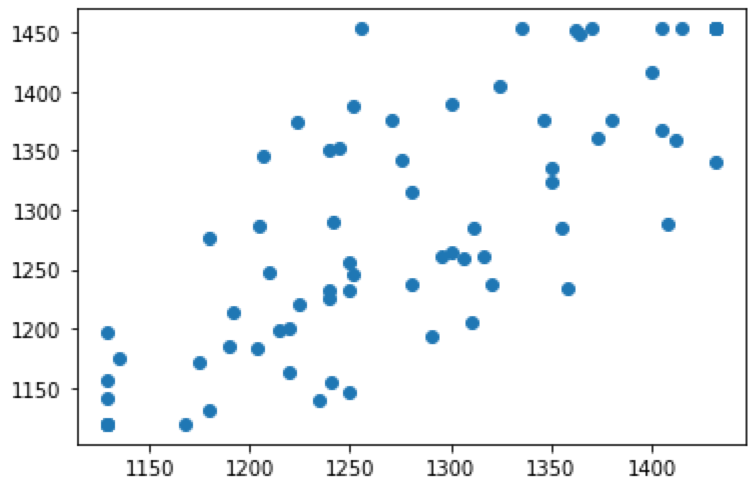
```
==============================================================================
Omnibus:                        1.442   Durbin-Watson:                   1.839
Prob(Omnibus):                  0.486   Jarque-Bera (JB):                1.359
Skew:                          -0.322   Prob(JB):                        0.507
Kurtosis:                       2.835   Cond. No.                         1.00
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
```

4. Model Evaluation

```
predictions = lm.predict(X_test)
```

```
plt.scatter(y_test,predictions)
```

```
<matplotlib.collections.PathCollection at 0x7f117539ec50>
```




```
# Regression Evaluation Metrics
from sklearn import metrics
```

```
print('MAE:' , metrics.mean_absolute_error(y_test,predictions))
print('MSE:' , metrics.mean_squared_error(y_test,predictions))
print('RMSE:' , np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

```
MAE: 53.53143624374527
MSE: 4709.43998707502
RMSE: 68.6253596498774
```