```
import numpy as np
data=[1,2,2,3,1,1,15,2,2,2,3,1,1,2]
mean=np.mean(data)
std=np.std(data)
print('mean of the dataset is',mean)
print('std.deviation is',std)
threshold=3
outlier=[]
for i in data:
  z=(i-mean)/std
  if z>threshold:
    outlier.append(i)
    print('outlier in dataset is: ',outlier)
```

```
 mean of the dataset is 2.7142857142857144
 std.deviation is 3.4729273660409197
 outlier in dataset is:  [15]
```

```
import pandas as pd
df=pd.read_csv("/content/2.2 dataset breast cancer.csv")
mean=np.mean(df)
std=np.std(df)
print('mean of the dataset is',mean)
print('std.deviation is',std)
threshold=3
outlier=[]
for i in df:
  z=(i-mean)/std
  if z>threshold:
       outlier.append(i)
print('outlier in dataset is: ',outlier)
```

```
mean of the dataset is id                         3.037183e+07
radius_mean                1.412729e+01
texture_mean               1.928965e+01
perimeter_mean             9.196903e+01
area_mean                  6.548891e+02
smoothness_mean            9.636028e-02
compactness_mean           1.043410e-01
concavity_mean             8.879932e-02
concave points_mean        4.891915e-02
symmetry_mean              1.811619e-01
fractal_dimension_mean     6.279761e-02
radius_se                  4.051721e-01
texture_se                 1.216853e+00
perimeter_se               2.866059e+00
area_se                    4.033708e+01
smoothness_se              7.040979e-03
compactness_se             2.547814e-02
concavity_se               3.189372e-02
concave points_se          1.179614e-02
symmetry_se                2.054230e-02
fractal_dimension_se       3.794904e-03
radius_worst               1.626919e+01
texture_worst              2.567722e+01
perimeter_worst            1.072612e+02
area_worst                 8.805831e+02
smoothness_worst           1.323686e-01
compactness_worst          2.542650e-01
concavity_worst            2.721885e-01
concave points_worst       1.146062e-01
symmetry_worst             2.900756e-01
fractal_dimension_worst    8.394582e-02
Unnamed: 32                         NaN
dtype: float64
std.deviation is id                       1.249107e+08
radius_mean                3.520951e+00
texture_mean               4.297255e+00
perimeter_mean             2.427762e+01
area_mean                  3.516048e+02
smoothness_mean            1.405176e-02
compactness_mean           5.276633e-02
concavity_mean             7.964973e-02
concave points_mean        3.876873e-02
symmetry_mean              2.739018e-02
fractal_dimension_mean     7.054156e-03
radius_se                  2.770689e-01
texture_se                 5.511634e-01
perimeter_se               2.020077e+00
area_se                    4.545101e+01
smoothness_se              2.999878e-03
compactness_se             1.789244e-02
concavity_se               3.015952e-02
concave points_se          6.164861e-03
symmetry_se                8.259104e-03
fractal_dimension_se       2.643745e-03
radius_worst               4.828993e+00
texture_worst              6.140854e+00
perimeter_worst            3.357300e+01
area_worst                 5.688565e+02
smoothness_worst           2.281236e-02
compactness_worst          1.571982e-01
concavity_worst            2.084409e-01
concave points_worst       6.567455e-02
symmetry_worst             6.181308e-02
fractal_dimension_worst    1.804539e-02
Unnamed: 32                         NaN
dtype: float64
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:3502: FutureWarning: In a future version, DataFrame.mean(axis=None
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:3502: FutureWarning: The default value of numeric_only in DataFram
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:3643: FutureWarning: The default value of numeric_only in DataFram
  return std(axis=axis, dtype=dtype, out=out, ddof=ddof, **kwargs)
---------------------------------------------------------------------------
UFuncTypeError                            Traceback (most recent call last)
<ipython-input-12-a0bae9e0a163> in <cell line: 9>()
      8 outlier=[]
      9 for i in df:
---> 10   z=(i-mean)/std
     11   if z>threshold:
     12         outlier.append(i)

                                    6 frames

/usr/local/lib/python3.10/dist-packages/pandas/core/roperator.py in rsub(left, right)
     13
     14 def rsub(left, right):
---> 15     return right - left
     16
     17


UFuncTypeError: ufunc 'subtract' did not contain a loop with signature matching types (dtype('<U2'), dtype('float64')) -> None
```

INTERQUARTILE RANGE TO DETECT OUTLIERS IN DATA

Q1 represents the 25th percentile of the data

Q2 represents the 50th percentile of the data

Q3 represents the 75th percentile of the data

if dataset has 2n/sn+1 data points then

Q1=median of the dataset

Q2=median of n smallest datapoints

Q3=median of n highest data points

IQR is the range between the first and the third quartiles namely Q1 and Q3: IQR=Q3-Q1.

```
import numpy as np
import seaborn as sns
data=[6,2,3,4,5,1,50]
sort_data=np.sort(data)
sort_data
```

```
array([ 1,  2,  3,  4,  5,  6, 50])
```

```
Q1=np.percentile(data,25,interpolation ='midpoint')
Q2=np.percentile(data,50,interpolation ='midpoint')
Q3=np.percentile(data,75,interpolation ='midpoint')
print('Q1 25 percentile of the given data is, ',Q1)
print('Q2 50 percentile of the given data is, ',Q2)
print('Q3 75 percentile of the given data is, ',Q3)
IQR=Q3-Q1
print('Interquartile range is',IQR)
```

```
Q1 25 percentile of the given data is,  2.5
Q2 50 percentile of the given data is,  4.0
Q3 75 percentile of the given data is,  5.5
Interquartile range is 3.0
```

```
low_lim=Q1-1.5*IQR
up_lim=Q3+1.5*IQR
print('low_limit is',low_lim)
print('up_limit is',up_lim)
```

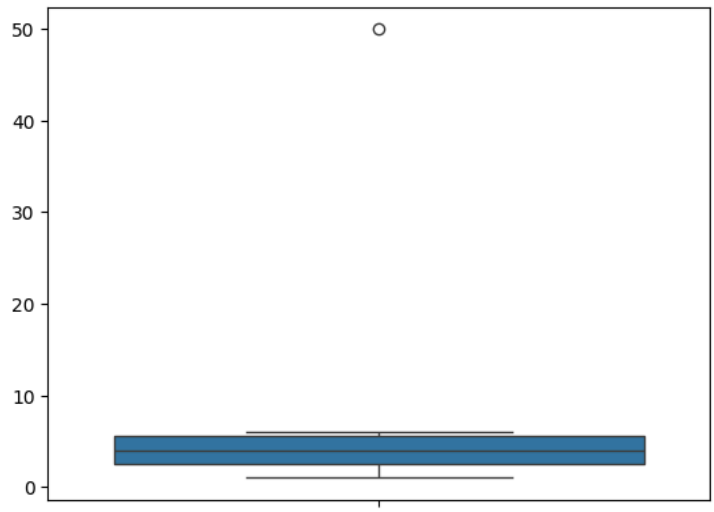
```
low_limit is -2.0
up_limit is 10.0
```

```
outlier=[]
for x in data:
  if((x>up_lim) or(x<low_lim)):
    outlier.append(x)
    print('outlier in the dataset is',outlier)
```

```
outlier in the dataset is [50]
```

```
sns.boxplot(data)
```

```
<Axes: >
```




```
df=pd.read_csv("/content/train.csv")
```

```
def load_data():
  return df.loc[:300,['Survived','Pclass','Sex','Cabin','Embarked']]
  df=load_data()
```

```
df.Cabin.duplicated()
```

```
0      False
1      False
2      False
3       True
4       True
       ...
707     True
708     True
709     True
710     True
711    False
Name: Cabin, Length: 712, dtype: bool
```

```
df.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
707    False
708    False
709    False
710    False
711    False
Length: 712, dtype: bool
```

```
df.duplicated(subset=['Survived','Pclass','Sex'])
```

```
0      False
1      False
2      False
3      False
4      False
       ...
707     True
708     True
709     True
710     True
711     True
Length: 712, dtype: bool
```

```
df.Cabin.duplicated().sum()
```

```
583
```

```
df.duplicated().sum()
```