What are Outliers

By definition outliers are extreme data points in a dataset. To make it easy to understand , suppose you want to find out the average salary of a company , you were given salary data of 6 people .

```
-- 5 associates - Drawing 25,000 monthly
-- 1 CEO - Drawing 50,000 monthly
```

Now if you try to find out the average by summing up all 6 salary figures here and divide it by 6 , you will get 29,166 , which not correct at all , because the ceo salary given here is working as an outlier , so we will try to find out how we can determine the outliers in a data and remove them.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

```
salary = pd.DataFrame(data = [['CEO',50000],
                              ['Associate',25000],
                              ['Associate',25500],
                              ['Associate',26000],
                              ['Associate',27000],
                              ['Associate',28900]],columns = 
['Designation','Salary'])
```

```
salary
```

```
  Designation  Salary
0         CEO   50000
1   Associate   25000
2   Associate   25500
3   Associate   26000
4   Associate   27000
5   Associate   28900
```

```
# Now if we try to find the average here , let's seet what happens
print(f"the average employee salary is, {salary['Salary'].mean()}")
```

```
the average employee salary is, 30400.0
```

It's Clear that the average salary we are seeing here is a mistake , let's try to resolve this with few methods

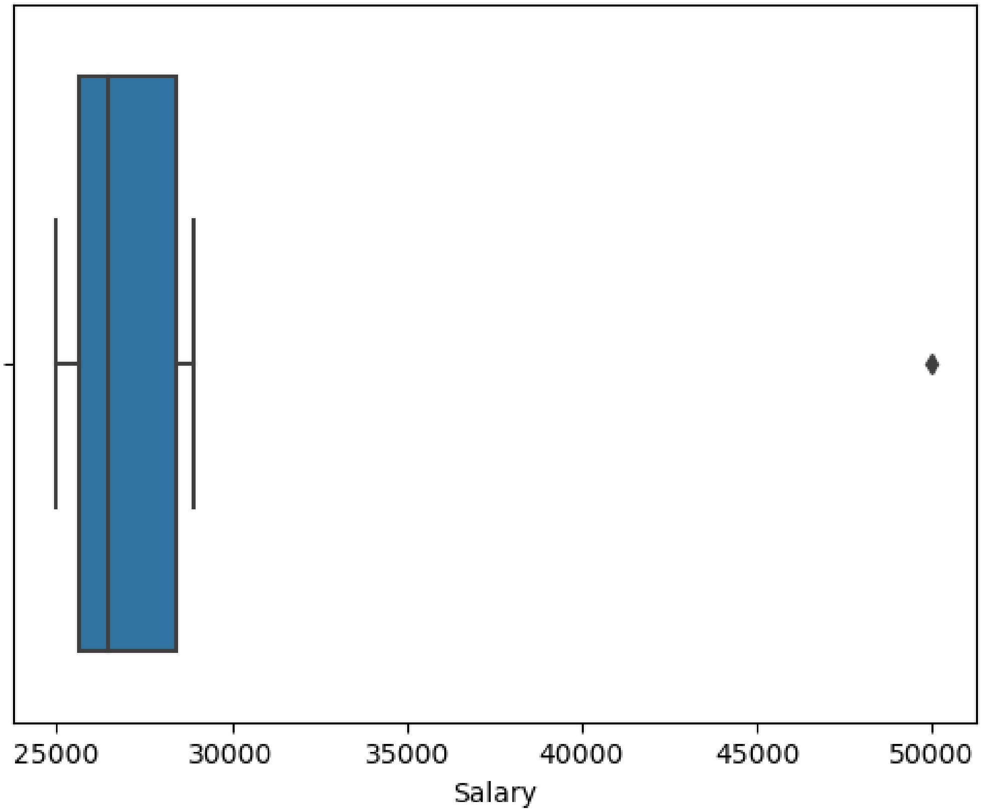
Method 1 - Visualization

With the help of visualization we can see the outliers in a data

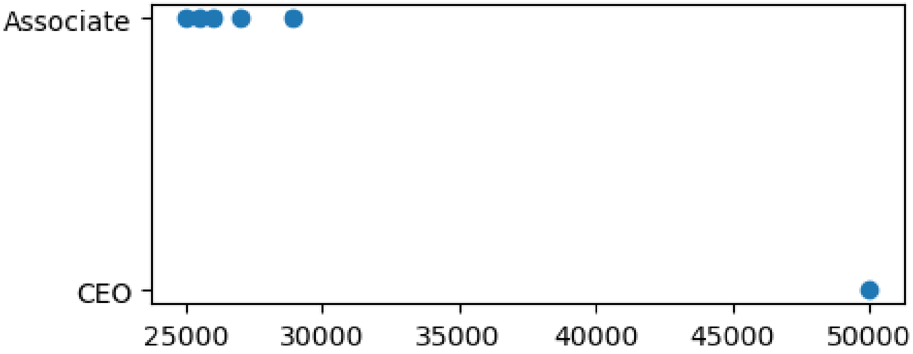
```
sns.boxplot(salary['Salary'])
plt.show
```

```
D:\Anaconda\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  warnings.warn(
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



```
# Scatter plot
fig, ax = plt.subplots(figsize = (5,2))
ax.scatter(salary['Salary'],salary['Designation'])
plt.show()
```



As we can see above that the data point in '50000' is far fetched from all other data we have in the set and in the plot , the upper section is nearing the '29000' mark , so we can find out all data above , 29,000 and remove them

```
# The datapoint identification with numpy
print(f"The outlier data points are")
print(np.where(salary['Salary']>29000))
```

```
The outlier data points are
(array([0], dtype=int64),)
```

We can see above that the outlier index is '0' , now let's find out what data sit's at 0 index position

```
print(salary.iloc[0])
```

```
Designation      CEO
Salary         50000
Name: 0, dtype: object
```

Method 2 - Z Score Method

with the help of z score we can identify and set a thresold of the data points , beyond which if any data is present will be called outliers , this score defines that how far the data point is from the mean of data

```
-- Zscore = (data_point -mean) / std. deviation
```

```
from scipy import stats
```

```
z = np.abs(stats.zscore(salary['Salary']))
print(z)
```

```
0    2.213359
1    0.609803
2    0.553340
3    0.496877
4    0.383950
5    0.169390
Name: Salary, dtype: float64
```

We are again able to identify that which index is beyond of all others , as per the score for this case we can identify a thresold limit as 1 , generally we take it as 3(As 99.7% of the data points lie between +/- 3 standard deviation (using Gaussian Distribution approach). for real data , but as our data is small and we can see that 99% of the data is falling withing 1 zscore , we can keep the thresold as 1

```
threshold = 1

# Position of the outlier
print(np.where(z > 1))
```

```
(array([0], dtype=int64),)
```

```
print(salary.iloc[0])
```

```
Designation      CEO
Salary         50000
Name: 0, dtype: object
```

Method 3 - IQR Method

IQR or interquartile method takes into consideration the percentile position of data , and then set upper and lower bound in it to identify the outliers .

IQR is used to measure variability by dividing a data set into quartiles. The data is sorted in ascending order and split into 4 equal parts. Q1, Q2, Q3 called first, second and third quartiles are the values which separate the 4 equal parts.

```
--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 a dataset has 2n / 2n+1 data points, then

```
--Q1 = median of the dataset.
--Q2 = median of n smallest data points.
--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. The data points which fall below Q1 – 1.5 IQR or above Q3 + 1.5 IQR are outliers.

```
Q1 = np.percentile(salary['Salary'], 25, interpolation = 'midpoint')
Q2 = np.percentile(salary['Salary'], 50, interpolation = 'midpoint')
Q3 = np.percentile(salary['Salary'], 75, interpolation = 'midpoint')

print('Q1 25 percentile of the given data is, ', Q1)
print('Q1 50 percentile of the given data is, ', Q2)
print('Q1 75 percentile of the given data is, ', Q3)

IQR = Q3 - Q1
print('Interquartile range is', IQR)
```

```
Q1 25 percentile of the given data is,  25750.0
Q1 50 percentile of the given data is,  26500.0
Q1 75 percentile of the given data is,  27950.0
Interquartile range is 2200.0
```

```
# Finding lower and upper limit in data
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 22450.0
up_limit is 31250.0
```

```
# Storing the outlier in a list
outlier =[]
for x in (salary['Salary']):
    if ((x> up_lim) or (x<low_lim)):
         outlier.append(x)
print(' outlier in the dataset is', outlier)
```

```
 outlier in the dataset is [50000]
```

```
print(f"The outlier data points are")
print(np.where(salary['Salary']==50000))
```

```
The outlier data points are
(array([0], dtype=int64),)
```

```
print(salary.iloc[0])
```

```
Designation      CEO
Salary         50000
Name: 0, dtype: object
```

Conclusion

So we can conclude from the analysis above that the 'CEO' record in the data is an outlier , so let's drop the same

```
salary.drop(salary.index[[0]], inplace = True)
```

```
salary
```

```
  Designation  Salary
1   Associate   25000
2   Associate   25500
3   Associate   26000
4   Associate   27000
5   Associate   28900
```