Outliers

- An outlier is a data point in a data set that is distant from all other observations, which is significantly different from the remaining data.
- A data point that lies outside the overall distribution of the dataset.

What are the impacts of having outliers in a dataset?

1. It causes various problems during our statistical analysis (It may cause a significant impact on the mean and the standard deviation) Statistics such as the mean and variance are very susceptible to outliers.
2. In addition, **some Machine Learning models are sensitive to outliers** which may decrease their performance. Thus, depending on which algorithm we wish to train, we often remove outliers from our variables.

Reasons for Outliers

1. Data Entry Errors (Ex: Entering salary as 1,00,000 instead of 10,000)
2. Measurement Errors (Ex: Measuring in meters instead of KM)
3. Instrumental Errors

Types of Outliers

1. Univariate Outliers --> Indentifing outlier for single variable
2. Bivariate Outliers --> Indentified as outlier by analyzing 2 variables at a time

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df= pd.read_csv("claimants.csv")
df.head()
```

Out[2]:

	CASENUM	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS	ATTORNEY
0	5	0.0	1.0	0.0	50.0	34.940	0
1	3	1.0	0.0	0.0	18.0	0.891	1
2	66	0.0	1.0	0.0	5.0	0.330	1
3	70	0.0	1.0	1.0	31.0	0.037	0
4	96	0.0	1.0	0.0	30.0	0.038	1

In [3]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1340 entries, 0 to 1339
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   CASENUM   1340 non-null   int64  
 1   CLMSEX    1328 non-null   float64
 2   CLMINSUR  1299 non-null   float64
 3   SEATBELT  1292 non-null   float64
 4   CLMAGE    1151 non-null   float64
 5   LOSS      1340 non-null   float64
 6   ATTORNEY  1340 non-null   int64  
dtypes: float64(5), int64(2)
memory usage: 73.4 KB
```

In [4]:
```
1 df.isnull().sum()
```

Out[4]:
```
CASENUM       0
CLMSEX       12
CLMINSUR     41
SEATBELT     48
CLMAGE      189
LOSS          0
ATTORNEY      0
dtype: int64
```

In [5]:
```
1 df['LOSS'].fillna(df["LOSS"].median(),inplace=True)
2 df
```

Out[5]:

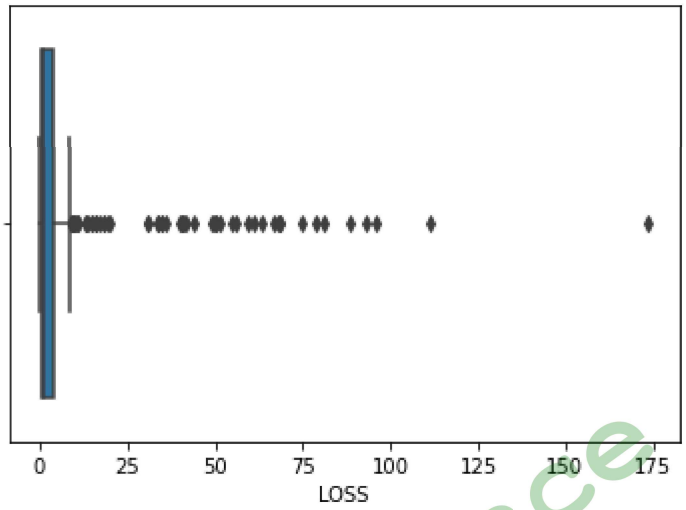
	CASENUM	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS	ATTORNEY
0	5	0.0	1.0	0.0	50.0	34.940	0
1	3	1.0	0.0	0.0	18.0	0.891	1
2	66	0.0	1.0	0.0	5.0	0.330	1
3	70	0.0	1.0	1.0	31.0	0.037	0
4	96	0.0	1.0	0.0	30.0	0.038	1
...	...	...	...	...	...	...	...
1335	34100	0.0	1.0	0.0	NaN	0.576	1
1336	34110	1.0	1.0	0.0	46.0	3.705	0
1337	34113	1.0	1.0	0.0	39.0	0.099	1
1338	34145	1.0	0.0	0.0	8.0	3.177	0
1339	34153	1.0	1.0	0.0	30.0	0.688	1

1340 rows × 7 columns

Various ways of finding the outlier.

Detection of outliers (based on Boxplot)

In [6]:
```
sns.boxplot(x=df["LOSS"])
plt.show()
```



Detection of outliers (based on IQR)

- Calculate first(q1) and third quartile(q3)
- Find interquartile range (q3-q1)
- Find lower bound q1*1.5* & *Find upper bound q31.5*

In [7]:
```
Q1=df["LOSS"].quantile(0.25)
Q1
```

Out[7]: 0.4

In [8]:
```
Q3=df["LOSS"].quantile(0.75)
Q3
```

Out[8]: 3.7815

In [9]:
```
IQR = Q3 - Q1
IQR
```

Out[9]: 3.3815

In [10]:
```
lower_limit = Q1 - (IQR * 1.5)
upper_limit = Q3 + (IQR * 1.5)

lower_limit,upper_limit
```

Out[10]: (-4.67225, 8.85375)

In [11]:
```
1 df[df["LOSS"]>8.85]
```

Out[11]:

	CASENUM	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS	ATTORNEY
0	5	0.0	1.0	0.0	50.0	34.940	0
11	148	0.0	1.0	0.0	41.0	19.610	0
22	550	0.0	0.0	0.0	38.0	16.161	0
24	580	0.0	1.0	0.0	54.0	10.040	1
43	941	1.0	1.0	0.0	55.0	13.100	1
...	...	...	...	...	...	...	...
1179	30430	1.0	1.0	0.0	37.0	13.789	0
1188	30588	1.0	1.0	0.0	44.0	13.000	0
1256	31159	0.0	1.0	1.0	30.0	30.640	0
1286	33037	1.0	1.0	1.0	44.0	55.709	0
1312	33661	1.0	1.0	0.0	45.0	14.884	0

66 rows × 7 columns

Dealing the Outliers (3'R' Technique)

1. Remove (Trimming: remove the outliers from our dataset)

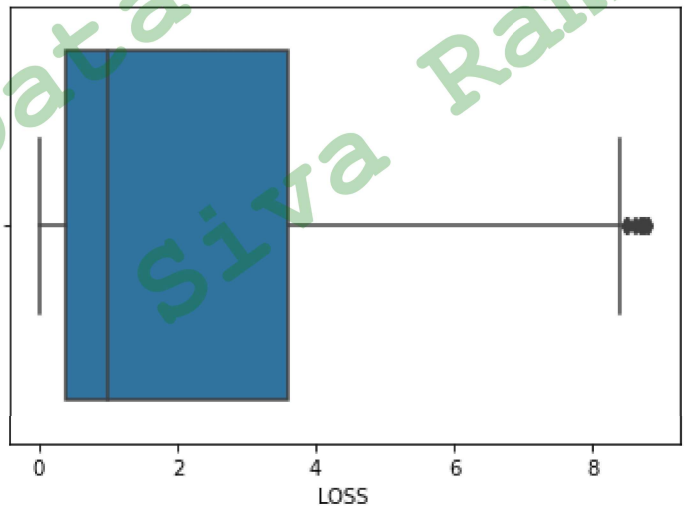
In [12]:
```
df_trimmed = df[(df["LOSS"] > lower_limit) & (df["LOSS"] < upper_
df_trimmed
```

Out[12]:

	CASENUM	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS	ATTORNEY
1	3	1.0	0.0	0.0	18.0	0.891	1
2	66	0.0	1.0	0.0	5.0	0.330	1
3	70	0.0	1.0	1.0	31.0	0.037	0
4	96	0.0	1.0	0.0	30.0	0.038	1
5	97	1.0	1.0	0.0	35.0	0.309	0
...	...	...	...	...	...	...	...
1335	34100	0.0	1.0	0.0	NaN	0.576	1
1336	34110	1.0	1.0	0.0	46.0	3.705	0
1337	34113	1.0	1.0	0.0	39.0	0.099	1
1338	34145	1.0	0.0	0.0	8.0	3.177	0
1339	34153	1.0	1.0	0.0	30.0	0.688	1

1274 rows × 7 columns

In [13]:
```
sns.boxplot(x=df_trimmed["LOSS"])
plt.show()
```



- We can still see some outliers...
- When we remove data points from our dataset,all the parameters of the distribution are re-calculated,
- those are the mean, quantiles and inter-quantile range,
- therefore, in the new -trimmed- variable, values that before were not considered outliers
- This is an unwanted characteristic of this way of coping with outliers.

2. Replace the ouliers

1.Rectify or Replace --> (data entry error) ---> Ask and confirm it from the Data Er team.

2. Replace with upper limit & lower limit (calculated based on IQR)

In [14]:
```
#pip install feature_engine
```

In [15]:
```
from feature_engine.outliers import Winsorizer
```

In [16]:
```
from feature_engine.outliers import Winsorizer

win = Winsorizer(capping_method='iqr', tail='both',
                 fold=1.5,variables=['LOSS'])

df_win = win.fit_transform(df[["LOSS"]])

df_win
```

Out[16]:

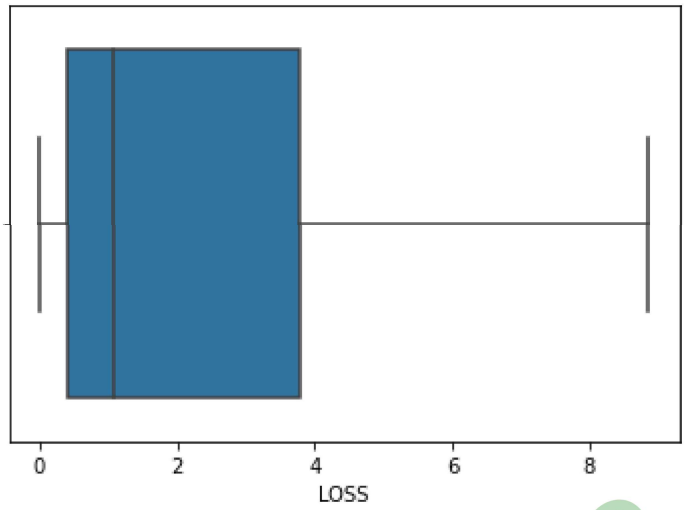
	LOSS
0	8.85375
1	0.89100
2	0.33000
3	0.03700
4	0.03800
...	...
1335	0.57600
1336	3.70500
1337	0.09900
1338	3.17700
1339	0.68800

1340 rows × 1 columns

In [17]:
```
print(win.left_tail_caps_, win.right_tail_caps_)
```

```
{'LOSS': -4.67225} {'LOSS': 8.85375}
```

In [18]:
```
1 sns.boxplot(x=df_win["LOSS"])
2 plt.show()
```



3. Replace Arbitrary Outlier Capper (the minimum and maximum values by a value determined by the user)

- We get the min and max values based on domain expertise

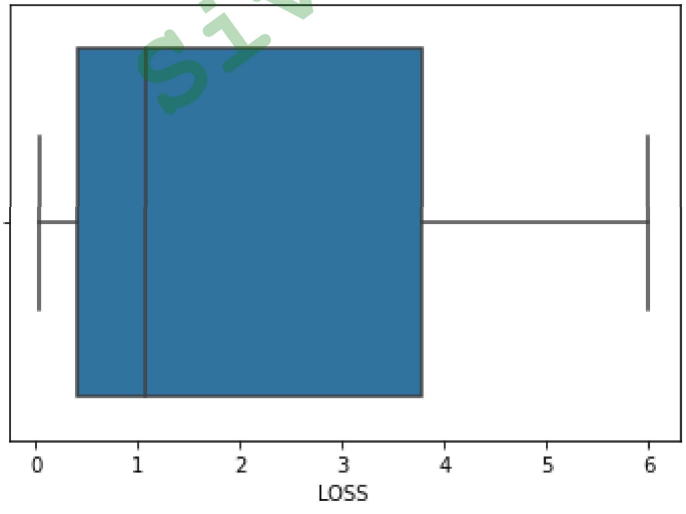
In [19]:
```
from feature_engine.outliers import ArbitraryOutlierCapper

capper = ArbitraryOutlierCapper(max_capping_dict = {'LOSS':6},
                                min_capping_dict = {'LOSS':0.03})

df_c = capper.fit_transform(df[["LOSS"]])

df_c
```

Out[19]:

	LOSS
0	6.000
1	0.891
2	0.330
3	0.037
4	0.038
...	...
1335	0.576
1336	3.705
1337	0.099
1338	3.177
1339	0.688

1340 rows × 1 columns

In [20]:
```
sns.boxplot(x=df_c["LOSS"])
plt.show()
```



3. Retain (consider for analysis) ---> Treat them separately