Published in Analytics Vidhya

Anuganti Suresh Follow
Nov 30, 2020 · 8 min read · Listen

How to Remove Outliers for Machine Learning?

> *What are outliers and how to deal with them?*

In this post we will try to understand all about outliers by answering the following questions, and at the end of the paper, will use Python to create some examples.

1. What Outlier is?

4. Why is it important to identify the outliers?

5. What are the types of Outliers?

6. What are the methods to prevent Outliers?

1. What Outlier is?

Outliers are those data points which differs significantly from other observations present in given dataset. It can occur because of variability in measurement and due to misinterpretation in filling data points.

2. How the Outlier are introduced in the datasets?

Most common causes of outliers on a data set:

- ***Data Entry Errors:*** Human errors such as errors caused during data collection, recording, or entry can cause outliers in data.
- ***Measurement Error* (instrument errors)**: It is the most common source of outliers. This is caused when the measurement instrument used turns out to be faulty.
- ***Experimental errors*** (data extraction or experiment planning/executing errors)
- ***Intentional*** (dummy outliers made to test detection methods)
- ***Data processing errors*** (data manipulation or data set unintended mutations)
- ***Sampling errors*** (extracting or mixing data from wrong or various sources)
- ***Natural Outlier*** (not an error, novelties in data): When an outlier is not artificial (due to error), it is a natural outlier. Most of real world data belong to this category.

3. How to detect Outliers?

Different outlier detection technique

a) Hypothesis Testing

b) Z-score method

c) Robust Z-score

d) I.Q.R method

e) Winsorization method (Percentile Capping)

f) DBSCAN Clustering

g) Isolation Forest

h) Linear Regression Models (PCA, LMS)

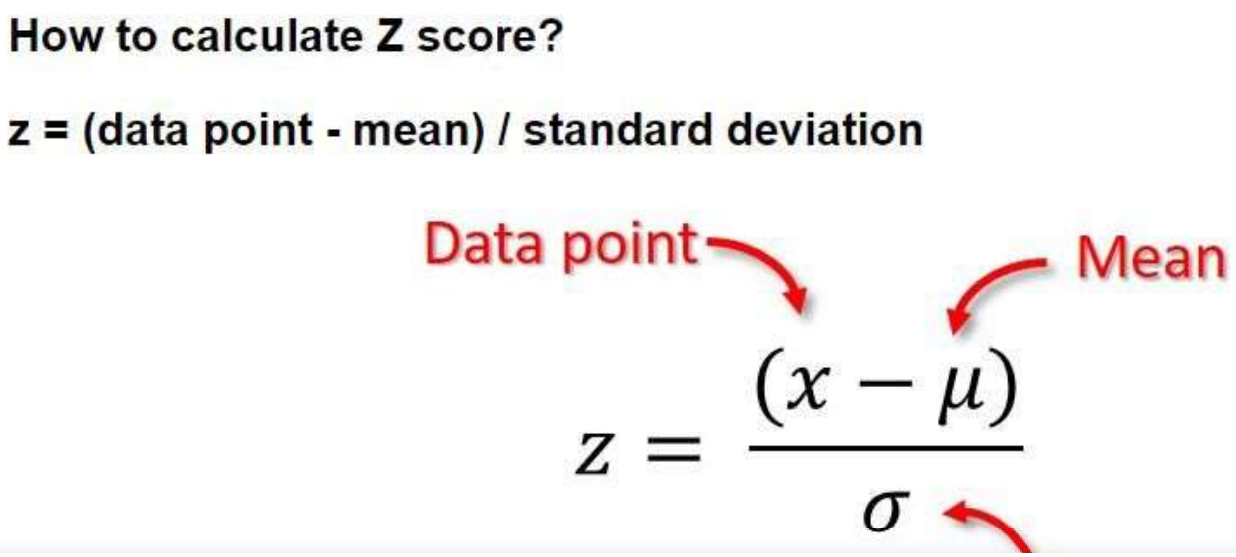
i) Standard Deviation

j) Percentile

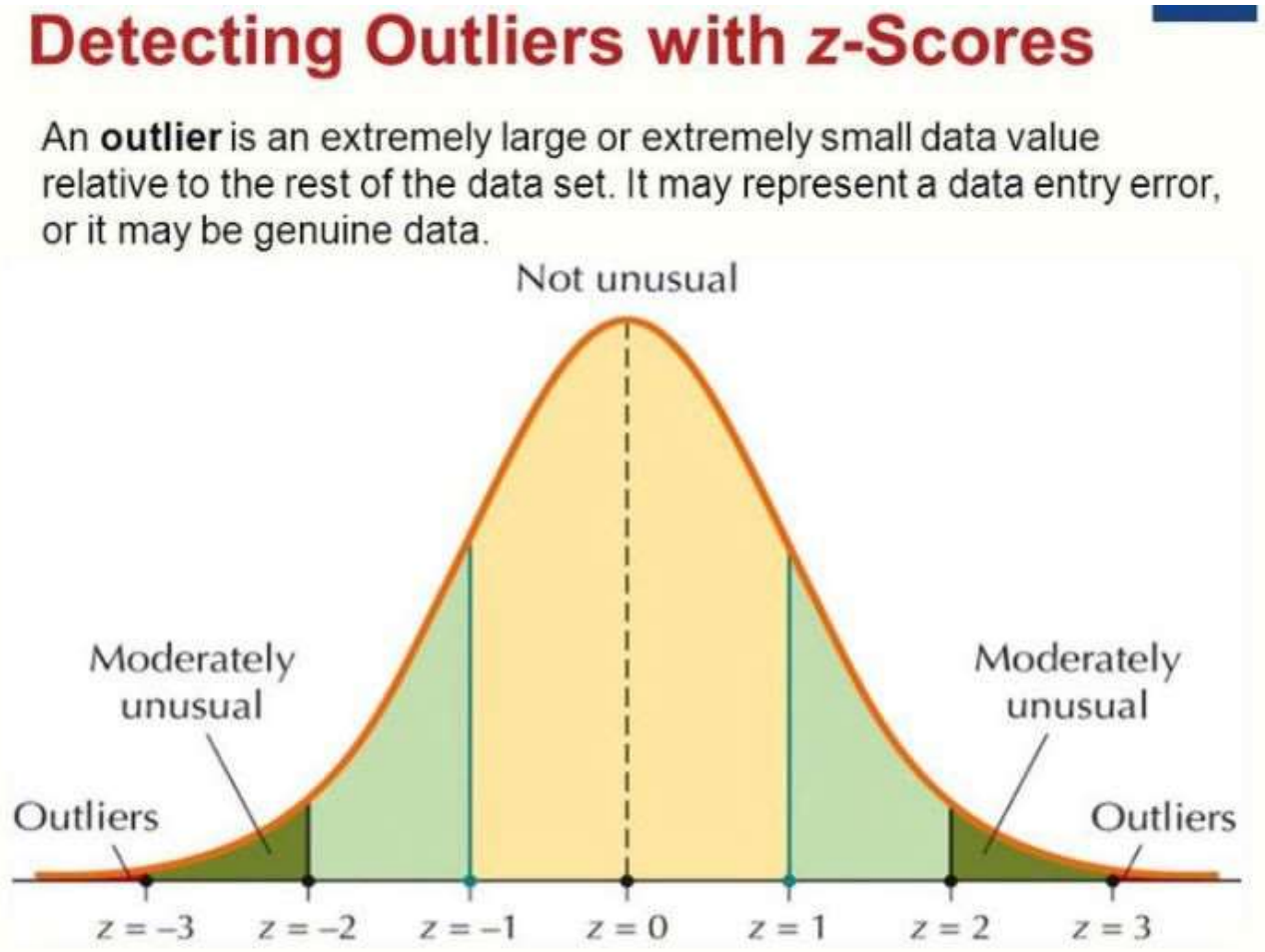
k) Visualizing the data

b) z score

This method assumes that the variable has a Gaussian distribution. It represents the number of standard deviations an observation is away from the mean:



Here, we normally define outliers as points whose modulus of z-score is greater than a threshold value. This threshold value is usually greater than 2 (3 is a common value).



d) IQR Method

In this method by using Inter Quartile Range(IQR), we detect outliers. IQR tells us the variation in the data set. Any value, which is beyond the range of -1.5 x IQR to 1.5 x IQR treated as outliers.

- Q1 represents the 1st quartile/25th percentile of the data.
- Q2 represents the 2nd quartile/median/50th percentile of the data.
- Q3 represents the 3rd quartile/75th percentile of the data.
- (Q1–1.5*IQR) represent the smallest value in the data set and (Q3+1.5*IQR) represent the largest value in the data set

Data visualization is useful for data cleaning, exploring data, detecting outliers and unusual groups, identifying trends and clusters etc. Here the list of data visualization plots to spot the outliers.

a) Box and whisker plot (box plot)

b) Scatter plot

c) Histogram

d) Distribution Plot

e) QQ plot

i) Univariate method

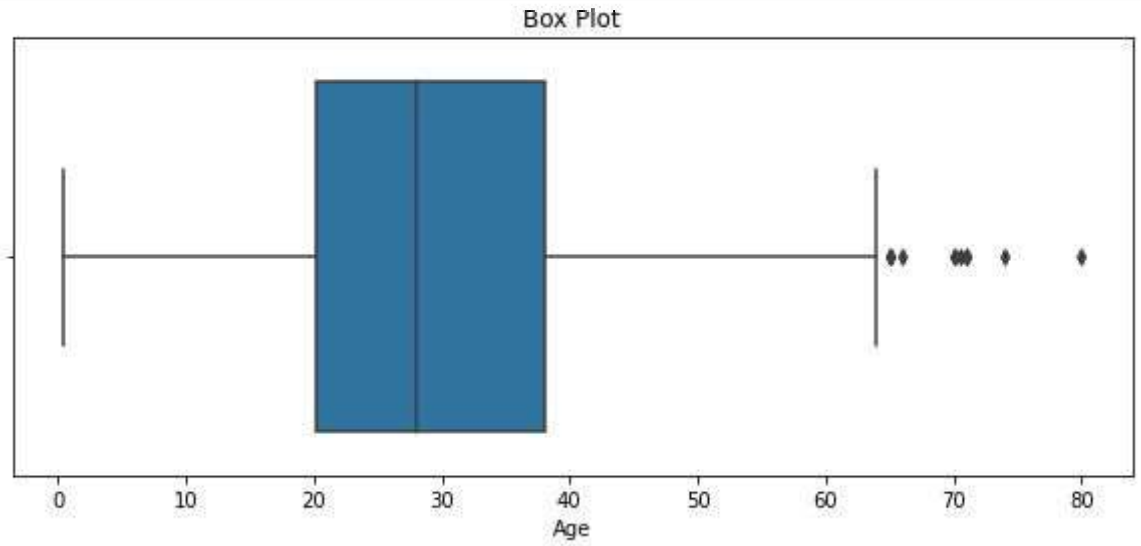
This method looks for data points with extreme values on one variable.

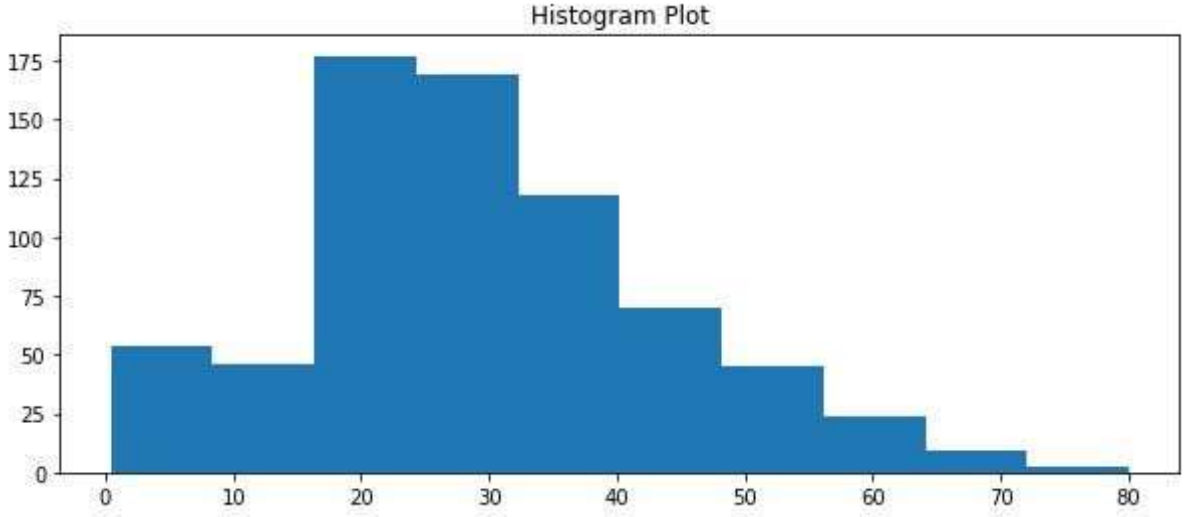
One of the simplest methods for detecting outliers is the use of ***box plots***. A box plot is a graphical display for describing the distributions of the data. Box plots use the median and the lower and upper quartiles.

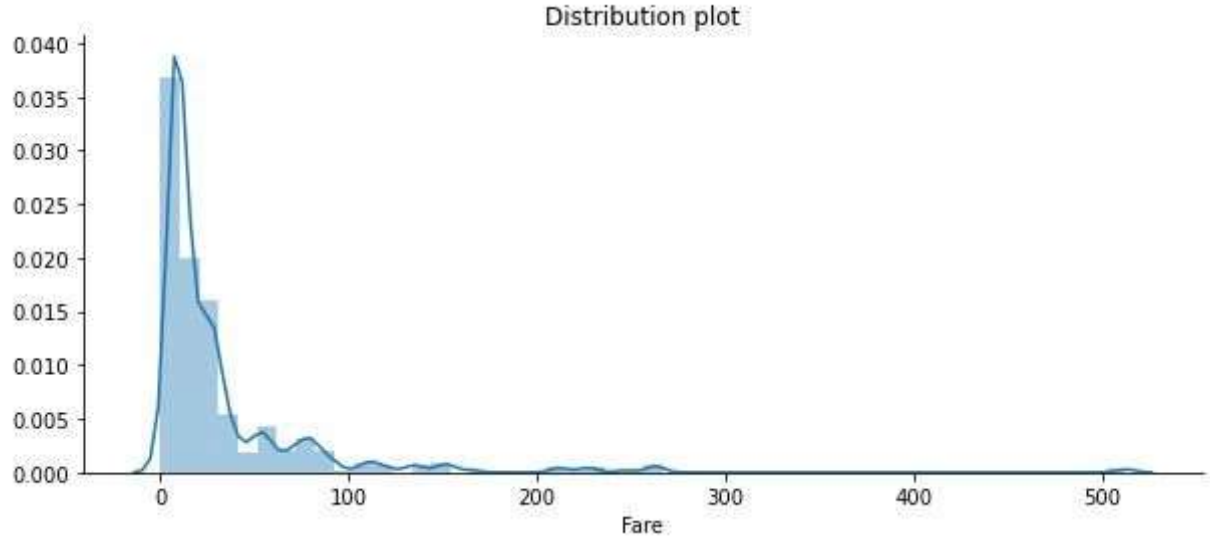


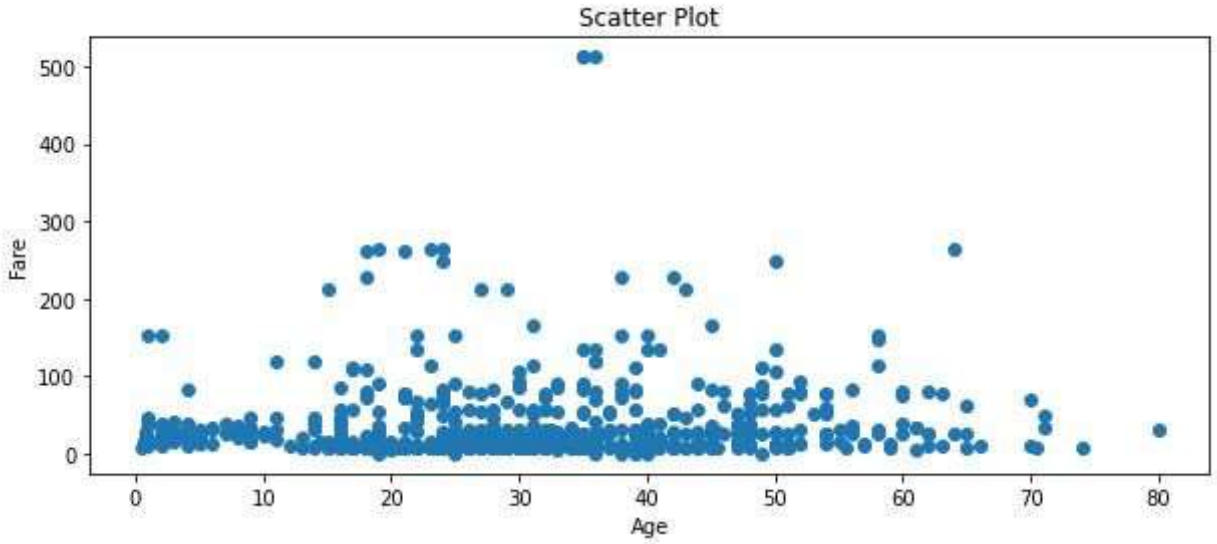

In [8]:

```
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from statsmodels.graphics.gofplots import qqplot
train = pd.read_csv('../input/titanic/train.csv')
def Box_plots(df):
    plt.figure(figsize=(10, 4))
    plt.title("Box Plot")
    sns.boxplot(df)
    plt.show()
Box_plots(train['Age'])

def hist_plots(df):
    plt.figure(figsize=(10, 4))
    plt.hist(df)
    plt.title("Histogram Plot")
    plt.show()
hist_plots(train['Age'])

def scatter_plots(df1,df2):
    fig, ax = plt.subplots(figsize=(10,4))
    ax.scatter(df1,df2)
    ax.set_xlabel('Age')
    ax.set_ylabel('Fare')
    plt.title("Scatter Plot")
    plt.show()
scatter_plots(train['Age'],train['Fare'])

def dist_plots(df):
    plt.figure(figsize=(10, 4))
    sns.distplot(df)
    plt.title("Distribution plot")
    sns.despine()
    plt.show()
dist_plots(train['Fare'])

def qq_plots(df):
    plt.figure(figsize=(10, 4))
    qqplot(df,line='s')
    plt.title("Normal QQPlot")
    plt.show()
qq_plots(train['Fare'])
```

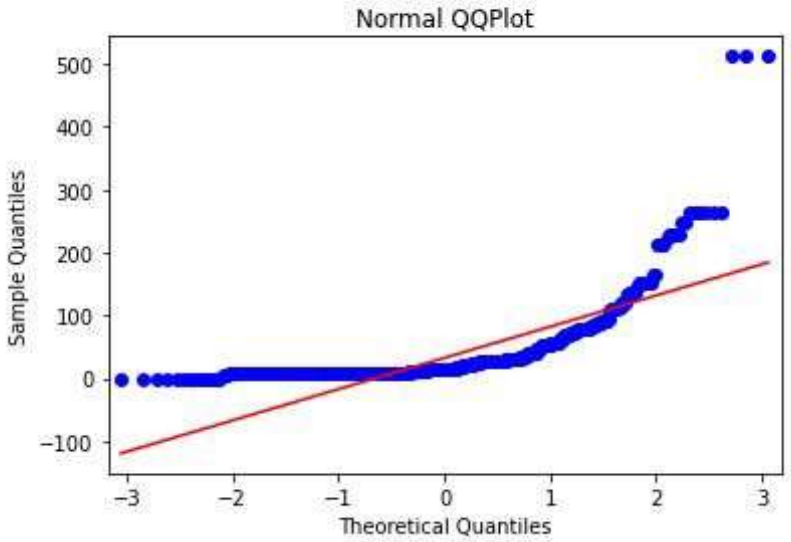








<Figure size 720x288 with 0 Axes>



4. Why is it important to identify the outliers?

Often outliers are discarded because of their effect on the total distribution and statistical analysis of the dataset. This is certainly a good approach if the outliers are due to an error of some kind (measurement error, data corruption, etc.), however often the source of the outliers is unclear. There are many situations where occasional 'extreme' events cause an outlier that is outside the usual distribution of the dataset but is a valid measurement and not due to an error. In these situations, the choice of how to deal with the outliers is not necessarily clear and the choice has a significant

5. What are the types of Outliers?

There are mainly 3 types of Outliers.

1. **Point or global Outliers:** Observations anomalous with respect to the majority of observations in a feature. In-short A data point is considered a global outlier if its value is far outside the entirety of the data set in which it is found.

Example: In a class all student age will be approx. similar, but if see a record of a student with age as 500. It's an outlier. It could be generated due to various reason.

2. Contextual (Conditional) Outliers: Observations considered anomalous given a specific context. A data point is considered a contextual outlier if its value significantly deviates from the rest of the data points in the same context. Note that this means that same value may not be considered an outlier if it occurred in a different context. If we limit our discussion to time series data, the "context" is almost always temporal, because time series data are records of a specific quantity over time. It's no surprise then that contextual outliers are common in time series data.In Contextual Anomaly values are not outside the normal global range but are abnormal compared to the seasonal pattern.

Example: World economy falls drastically due to COVID-19. Stock Market crashes due to the scam in 1992; in 2020 due to COVID-19. Usual data points will be near to each other whereas data point during the specific period will either up or down very far. This is not due to erroneous, but it's an actual observation data point.

3. Collective Outliers: A collection of observations anomalous but appear close to one another because they all have a similar anomalous value.

A subset of data points within a data set is considered anomalous if those values as a collection deviate significantly from the entire data set, but the values of the individual data points are not themselves anomalous in either a contextual or global sense. In time series data, one way this can manifest is as normal peaks and valleys occurring outside of a time frame when that seasonal sequence is normal or as a combination of time series that is in an outlier state as a group.

After detecting the outlier we should remove\treat the outlier because **it is a silent killer!!**.yes..

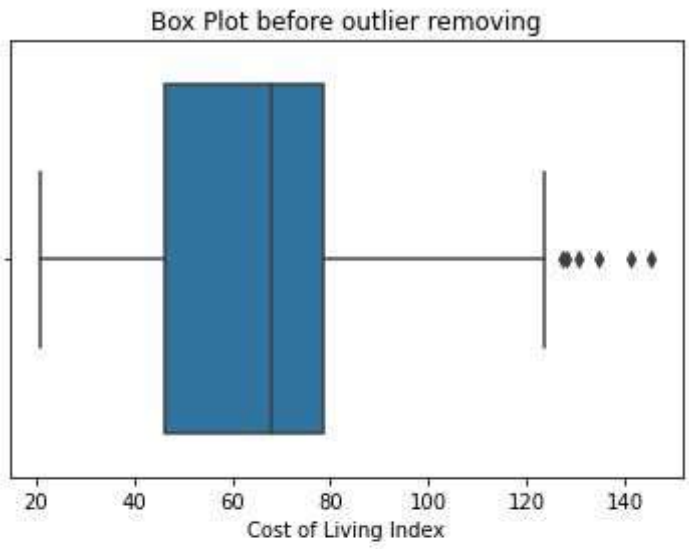
- Outliers badly affect mean and standard deviation of the dataset. These may statistically give erroneous results.
- It increases the error variance and reduces the power of statistical tests.
- If the outliers are non-randomly distributed, they can decrease normality.
- Most machine learning algorithms do not work well in the presence of outlier. So it is desirable to detect and remove outliers.
- They can also impact the basic assumption of Regression, ANOVA and other statistical model assumptions.

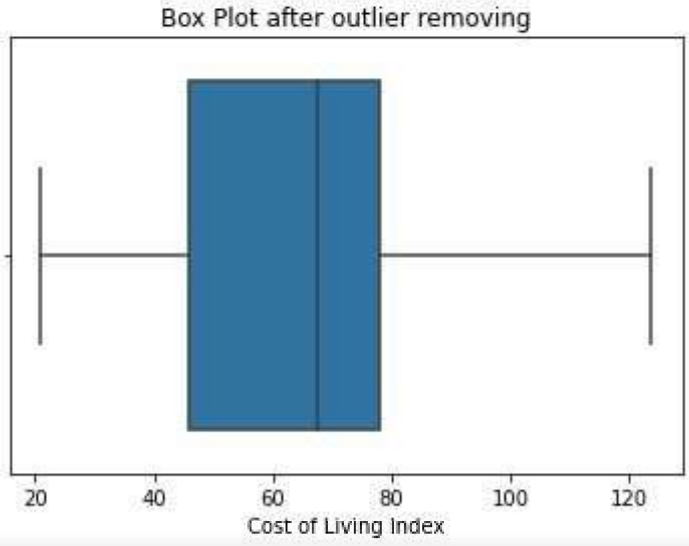
With all these reasons we must be careful about outlier and treat them before build a statistical/machine learning model. There are some techniques used to deal with outliers.

1. **Deleting observations**
2. **Transforming values**
3. **Imputation**
4. **Separately treating**
5. **Deleting observations**
   Sometimes it's best to completely remove those records from your dataset to stop them from skewing your analysis. We delete outlier values if it is due to *data entry error, data processing error or outlier observations are very small in numbers*. We can also use trimming at both ends to remove outliers. But **deleting** the observation is not a good idea when we have **small dataset**.




```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt

train = pd.read_csv('../input/cost-of-living/cost-of-living-2018.csv')
sns.boxplot(train['Cost of Living Index'])
plt.title("Box Plot before outlier removing")
plt.show()

def drop_outliers(df, field_name):
    iqr = 1.5 * (np.percentile(df[field_name], 75) - np.percentile(df[field_name], 25))
    df.drop(df[df[field_name] > (iqr + np.percentile(df[field_name], 75))].index, inplace=True)
    df.drop(df[df[field_name] < (np.percentile(df[field_name], 25) - iqr)].index, inplace=True)

drop_outliers(train, 'Cost of Living Index')
sns.boxplot(train['Cost of Living Index'])
plt.title("Box Plot after outlier removing")
plt.show()
```





Transforming variables can also eliminate outliers. These transformed values reduces the variation caused by extreme values.

1. Scaling
2. Log transformation
3. Cube Root Normalization
4. Box-transformation

- These techniques convert values in the dataset to smaller values.
- If the data has to many extreme values or skewed, this method helps to make your data normal.
- But These technique not always give you the best results.
- There is no lose of data from these methods
- In all these method box cox transformation gives the best result.




In [10]:
```
#Scalling
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn import preprocessing
train = pd.read_csv('../input/cost-of-living/cost-of-living-2018.csv')
plt.hist(train['Cost of Living Index'])
plt.title("Histogram before Scalling")
plt.show()
scaler = preprocessing.StandardScaler()
train['Cost of Living Index'] = scaler.fit_transform(train['Cost of Living Index'].values.reshape(-1,1))
plt.hist(train['Cost of Living Index'])
plt.title("Histogram after Scalling")
plt.show()
```

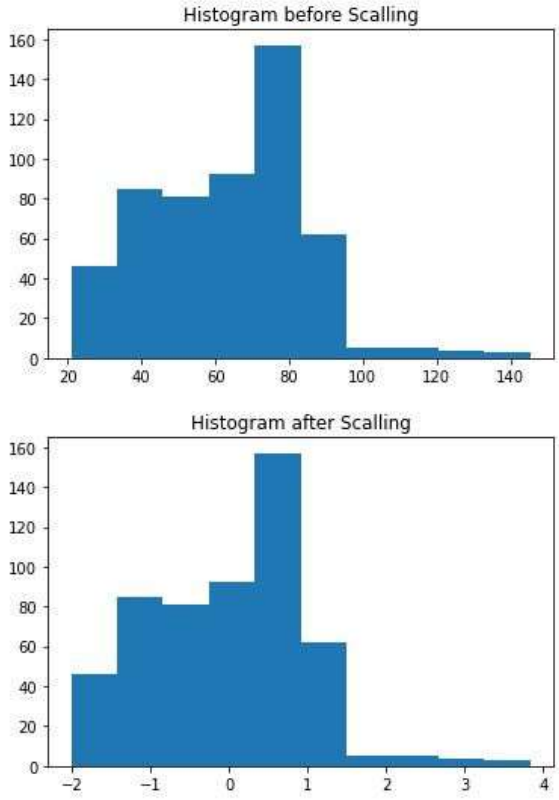
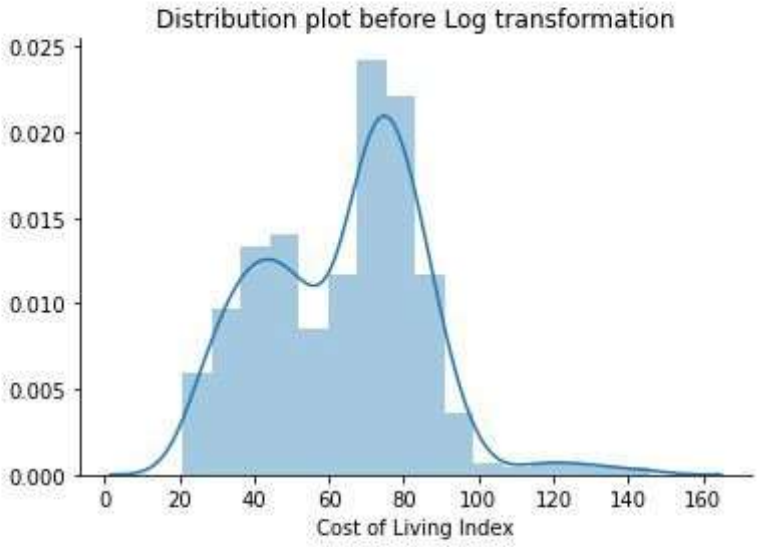
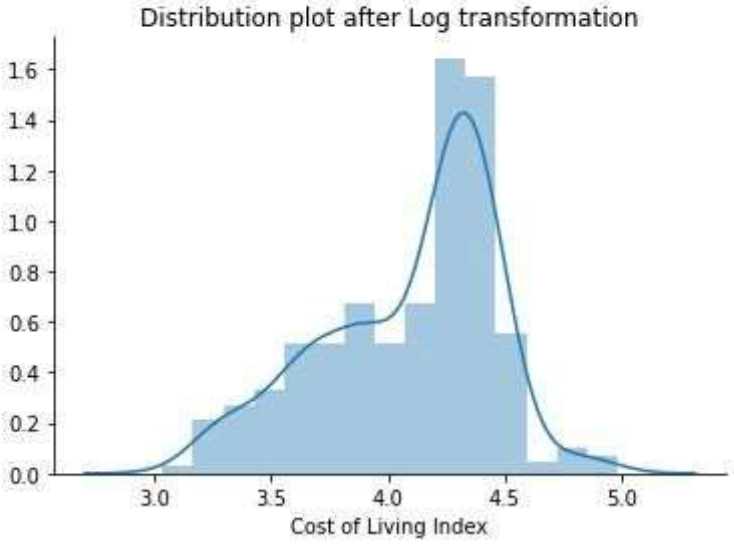


Fig : Scaling

In [11]:
```
#Log Transformation
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
train = pd.read_csv('../input/cost-of-living/cost-of-living-2018.csv')
sns.distplot(train['Cost of Living Index'])
plt.title("Distribution plot before Log transformation")
sns.despine()
plt.show()
train['Cost of Living Index'] = np.log(train['Cost of Living Index'])
sns.distplot(train['Cost of Living Index'])
plt.title("Distribution plot after Log transformation")
sns.despine()
plt.show()
```

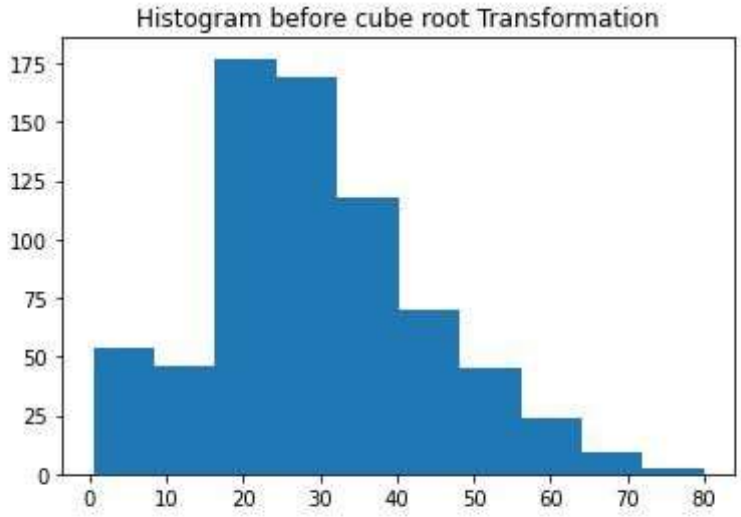


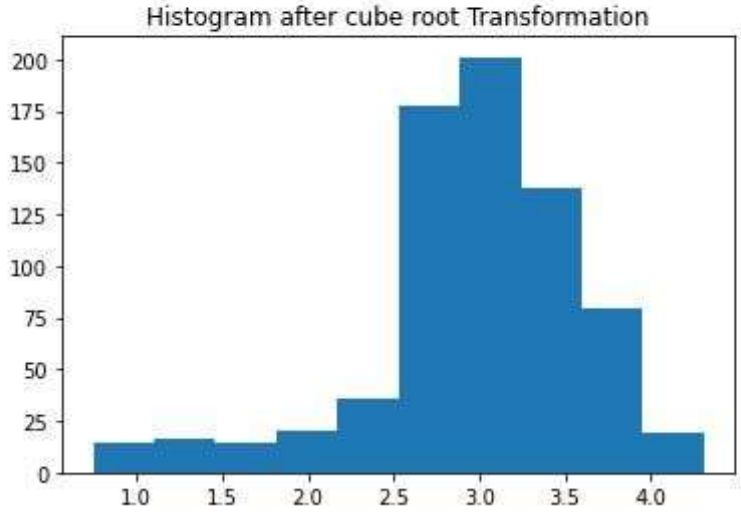


Log Transformation

In [12]:
```
#cube root Transformation
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
train = pd.read_csv('../input/titanic/train.csv')
plt.hist(train['Age'])
plt.title("Histogram before cube root Transformation")
plt.show()
train['Age'] = (train['Age']**(1/3))
plt.hist(train['Age'])
plt.title("Histogram after cube root Transformation")
plt.show()
```

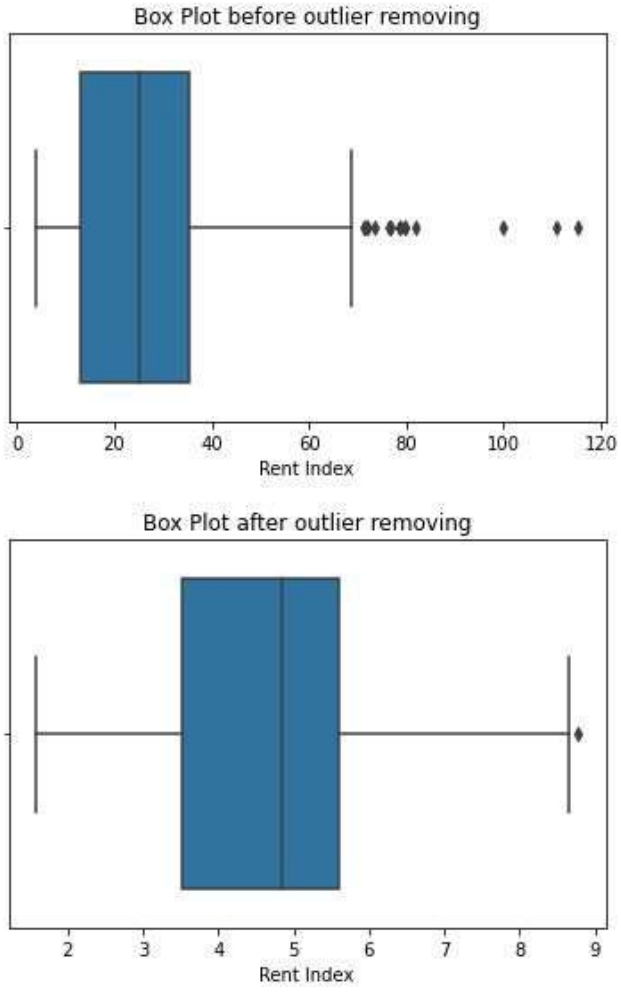
```
/opt/conda/lib/python3.7/site-packages/numpy/lib/histograms
  keep = (tmp_a >= first_edge)
/opt/conda/lib/python3.7/site-packages/numpy/lib/histograms
  keep &= (tmp_a <= last_edge)
```





Cube Root Transformation

In [13]:
```
#Box-transformation
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import scipy
train = pd.read_csv('../input/cost-of-living/cost-of-living-2018.csv')
sns.boxplot(train['Rent Index'])
plt.title("Box Plot before outlier removing")
plt.show()
train['Rent Index'],fitted_lambda= scipy.stats.boxcox(train['Rent Index'] ,lmbda=None)
sns.boxplot(train['Rent Index'])
plt.title("Box Plot after outlier removing")
plt.show()
```

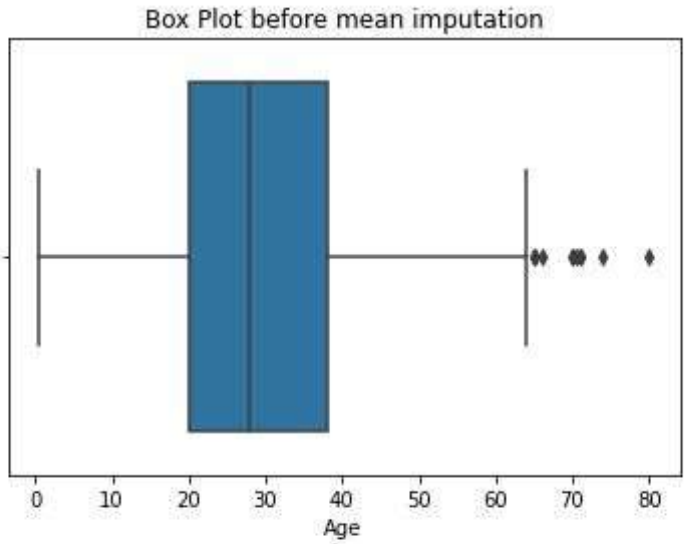


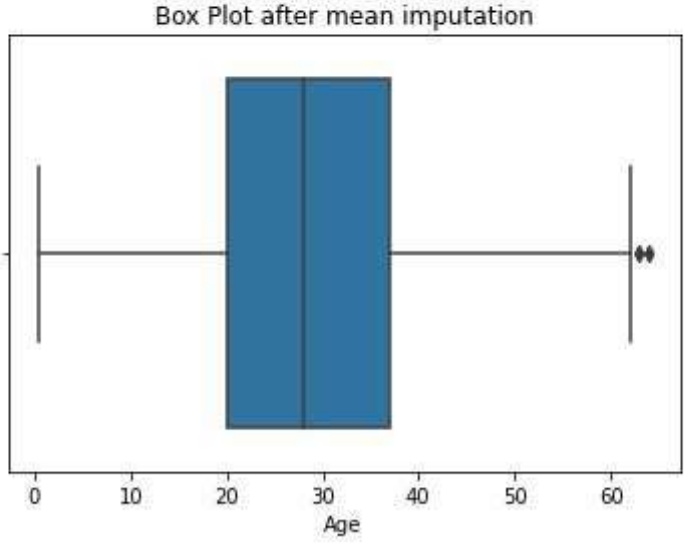
Box-transformation

3. Imputation

- Like imputation of missing values, we can also impute outliers. We can use mean, median, zero value in this methods. Since we imputing there is no loss of data. Here median is appropriate because it is not affected by outliers.

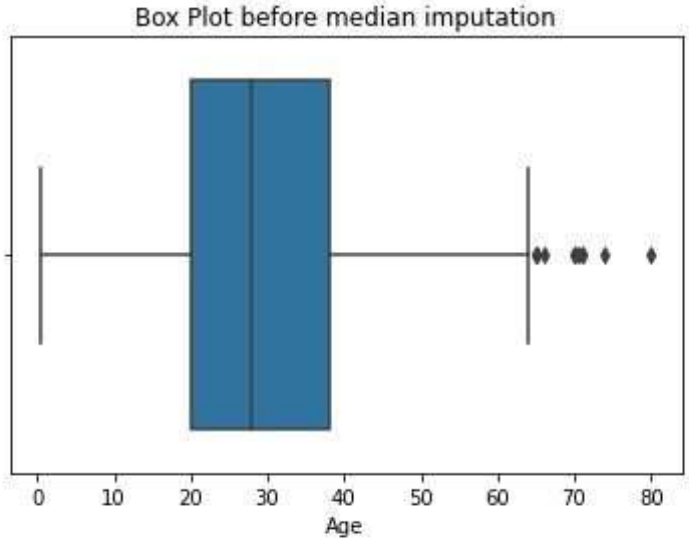
In [14]:
```
#mean imputation
import pandas as pd
import numpy as np
train = pd.read_csv('../input/titanic/train.csv')
sns.boxplot(train['Age'])
plt.title("Box Plot before mean imputation")
plt.show()
for i in train['Age']:
    q1 = train['Age'].quantile(0.25)
    q3 = train['Age'].quantile(0.75)
    iqr = q3-q1
    Lower_tail = q1 - 1.5 * iqr
    Upper_tail = q3 + 1.5 * iqr
    if i > Upper_tail or i < Lower_tail:
            train['Age'] = train['Age'].replace(i, np.mean(train['Age']))
sns.boxplot(train['Age'])
plt.title("Box Plot after mean imputation")
plt.show()
```

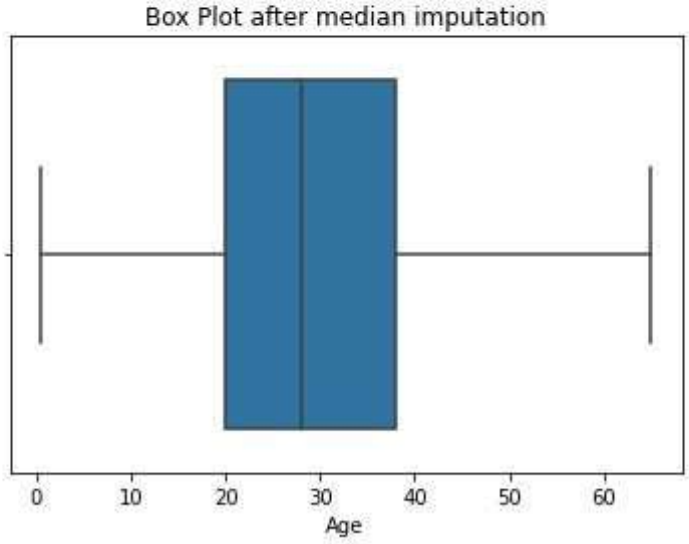




mean imputation

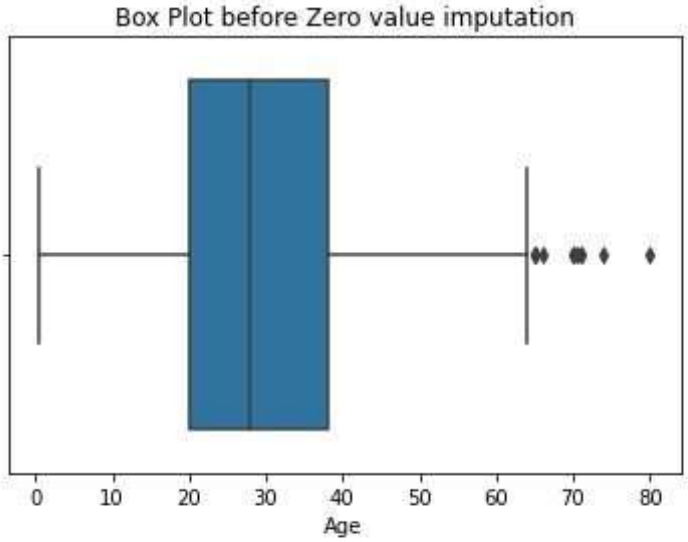
In [15]:
```
#median imputation
import pandas as pd
import numpy as np
train = pd.read_csv('../input/titanic/train.csv')
sns.boxplot(train['Age'])
plt.title("Box Plot before median imputation")
plt.show()
for i in train['Age']:
    q1 = train['Age'].quantile(0.25)
    q3 = train['Age'].quantile(0.75)
    iqr = q3-q1
    Lower_tail = q1 - 1.5 * iqr
    Upper_tail = q3 + 1.5 * iqr
    if i > Upper_tail or i < Lower_tail:
            train['Age'] = train['Age'].replace(i, np.median(train['Age']))
sns.boxplot(train['Age'])
plt.title("Box Plot after median imputation")
plt.show()
```

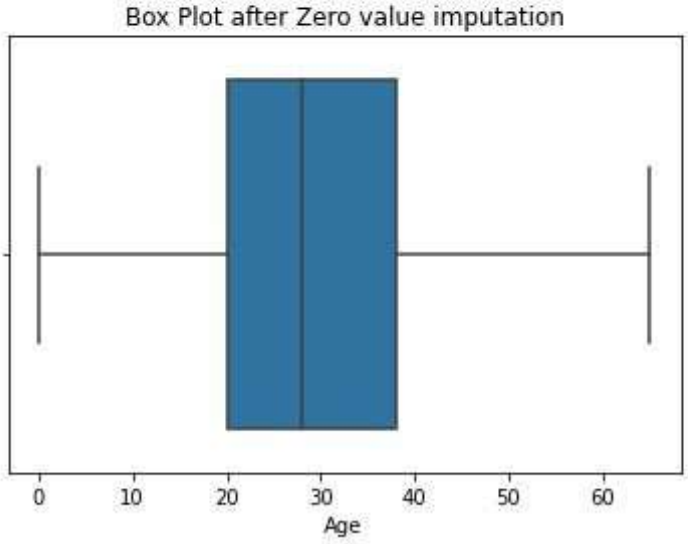




median imputation

In [16]:
```
#Zero value imputation
import pandas as pd
import numpy as np
train = pd.read_csv('../input/titanic/train.csv')
sns.boxplot(train['Age'])
plt.title("Box Plot before Zero value imputation")
plt.show()
for i in train['Age']:
    q1 = train['Age'].quantile(0.25)
    q3 = train['Age'].quantile(0.75)
    iqr = q3-q1
    Lower_tail = q1 - 1.5 * iqr
    Upper_tail = q3 + 1.5 * iqr
    if i > Upper_tail or i < Lower_tail:
            train['Age'] = train['Age'].replace(i, 0)
sns.boxplot(train['Age'])
plt.title("Box Plot after Zero value imputation")
plt.show()
```





Zero value imputation

If there are significant number of outliers and dataset is small , we should treat them separately in the statistical model. One of the approach is to treat both groups as two different groups and build individual model for both groups and then combine the output. But this technique is tedious when the dataset is large.

That's It!

Thanks for reading!

Thanks for the read. I am going to write more beginner-friendly posts in the future. Follow me up on Medium to be informed about them. I welcome feedback and can be reached out on LinkedIn anuganti-suresh. Happy learning!

Useful links

- http://www.kdnuggets.com/2017/01/3-methods-deal-outliers.html
- https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/
- http://machinelearningmastery.com/how-to-identify-outliers-in-your-data/

Clap if you liked the article!