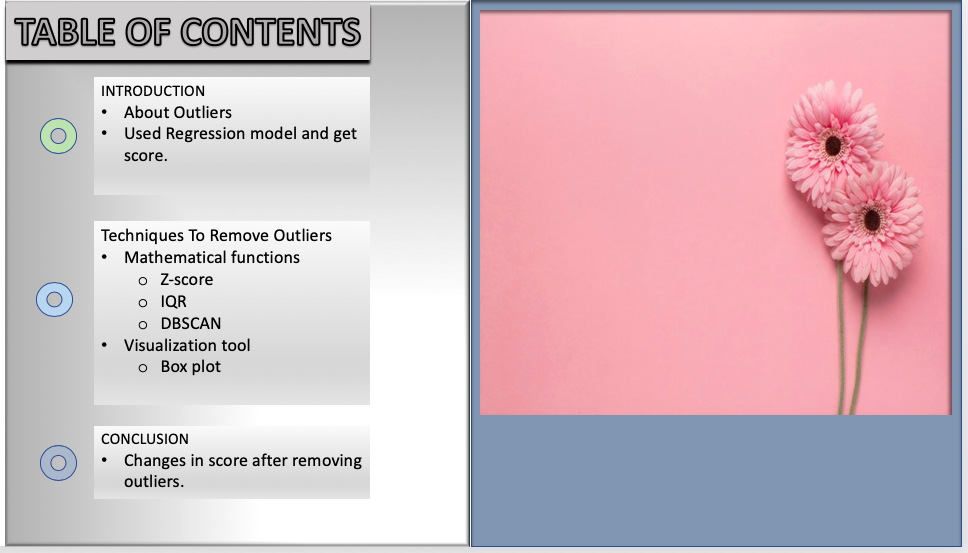
Does outlier removal affects Machine Learning Results..

****

This article shows the use of statistical methods to identify and remove outliers and how it impacts the score of the model.

1. Introduction

While working with datasets to train a Machine learning model, one of the most important part of data analysis is to remove noisy data. It is very important to get rid of irregularities and clean the data before we train the model. One of the most important downfalls in model performance is the presence of outliers in data. It ideally affects the model performance. For example

|  |  |  |
| --- | --- | --- |
|  | Data without outlier | Data with outlier |
| Data | ﻿x=[1,2,3,4,3,2,2,5] | ﻿x=[1,2,3,4,3,2,2,5,450] |
| Mean | ﻿2.75 | 52.45 |
| s.d | 1.19 | 140.56 |
| Median | 2.5 | 3 |
| Mode | 2 | 2 |

A dataset with outlier always has different values. This changes the result completely. According to [Wikipedia](https://en.wikipedia.org/wiki/Outlier), an **outlier** is an observation point that is distant from other observations. There are various techniques have proposed to remove outliers.

Out of many outlier techniques this article shows the result of z-score, Box plot, IQR and DBSCAN techniques and its effect on the prediction model’s result.

Let’s get start. The dataset used here is Boston House Pricing dataset from sklearn dataset API. The first step is to apply python libraries and input dataset.

from sklearn.datasets import load\_boston

import numpy as np

import pandas as pd

import matplotlib.pylab as plt

from scipy import stats

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.cluster import DBSCAN

from collections import Counter

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

boston = load\_boston()

columns = boston.feature\_names

Check for any NaN values and normalize data.

﻿﻿boston\_df\_x= pd.DataFrame(boston.data, columns = columns)

﻿boston\_df\_x.isna().any()

#Flase…there is no null values.

boston\_df\_x\_scaler = StandardScaler().fit\_transform(boston\_df\_x)

dataset\_x = pd.DataFrame(boston\_df\_x\_scaler, columns = columns)

boston\_df\_y= pd.DataFrame(boston.target, columns = ['target'])

boston\_df\_y\_scaler = StandardScaler().fit\_transform(boston\_df\_y)

dataset\_y = pd.DataFrame(boston\_df\_y\_scaler,columns = ['target'])

df\_combine = dataset\_x.join(dataset\_y)

df\_combine.head()

﻿x\_train,x\_test,y\_train,y\_test = train\_test\_split(dataset\_x, dataset\_y, test\_size = 0.3, random\_state=42)

linR = LinearRegression()

linR.fit(x\_train,y\_train)

*Find the score of the model.*

prediction = linR.predict(x\_test)

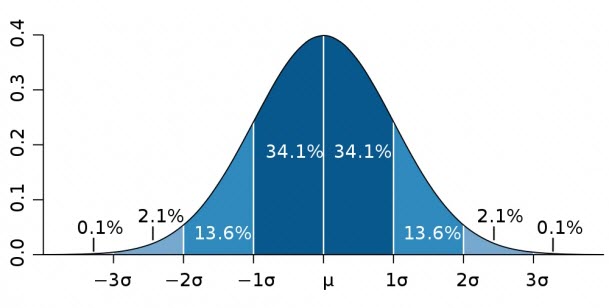
score = linR.score(x\_test,y\_test)

*score* ***0.712***

1. Techniques to remove Outliers
2. **Outliers with mathematical function**
3. *Z-SCORE Method*

[statisticshowto](https://www.statisticshowto.com/probability-and-statistics/z-score/) definition

A **z-score**(also called a standard score) gives an idea of how far from the mean of a data point is. A z-score can be placed on a normal distribution curve. Z-score ranges from -3 standard deviation to +3 standard deviation. In order to find z-score, one should know the mean μ and standard deviation σ. The idea behind z-score is to find the relationship of any data point from its group with the help of its mean and standard deviation.



\* 68% of the data points lie between + or - 1 standard deviation.

\* 95% of the data points lie between + or - 2 standard deviation

\* 99.7% of the data points lie between + or - 3 standard deviation

z-score formula

If z score value is more than 3 or less than -3, it assumes that the data set has outliers or some different values.

abs\_z\_scores = np.abs(z\_scores)

filtered\_entries = ((abs\_z\_scores > -3) & (abs\_z\_scores < 3)).all(axis=1)

new\_df = df\_combine[filtered\_entries]

*Applying Regression model and calculating score of the model*

prediction = linR.predict(x\_test)

score = linR.score(x\_test,y\_test)

# *﻿score* ***0.674***

1. **Outliers with visualization tools**
2. **Box plot**

Wikipedia definition

In descriptive statistics, a box plot or boxplot is a method for graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending from the boxes indicating variability outside the upper and lower quartiles. Outliers comes as individual points.

Plotting with dataset columns:

sns.boxplot(boston\_df\_x['CRIM'])

boston\_df\_x['CRIM'].describe()

sns.boxplot(boston\_df\_x['ZN'])

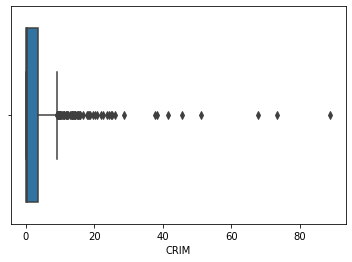
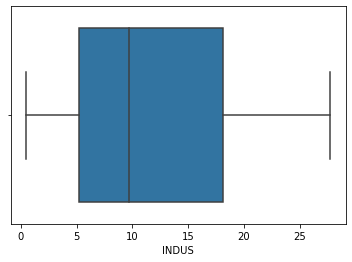
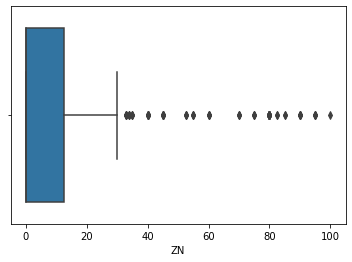
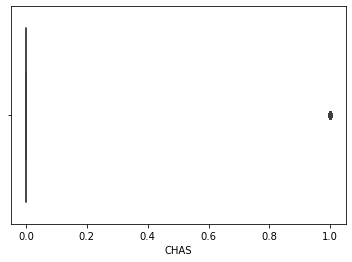
boston\_df\_x['ZN'].describe()

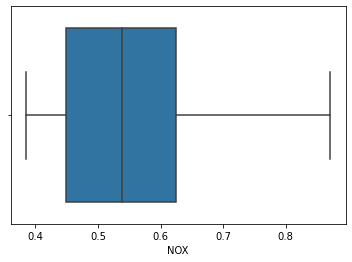
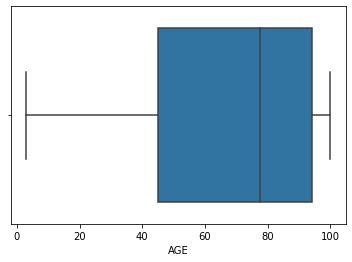
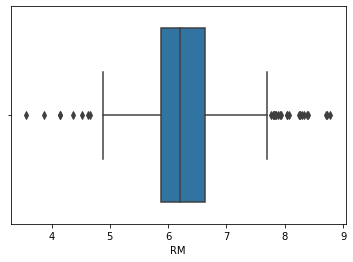
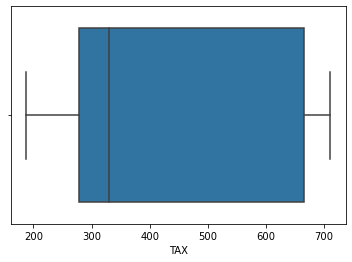
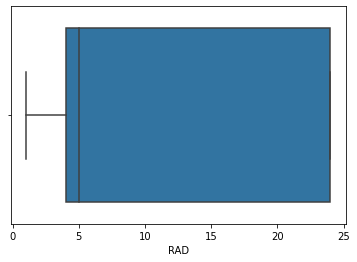
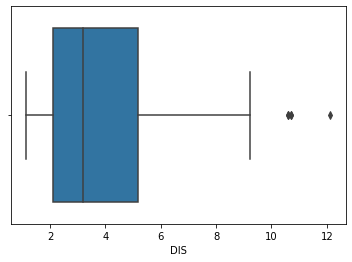
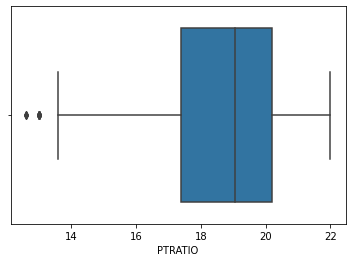
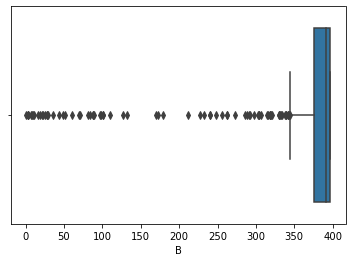
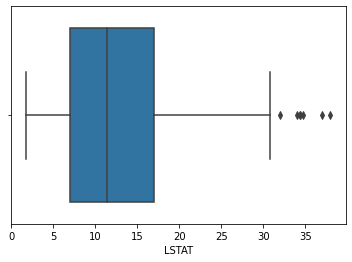
sns.boxplot(boston\_df\_x['INDUS'])

boston\_df\_x['INDUS'].describe()

………

………

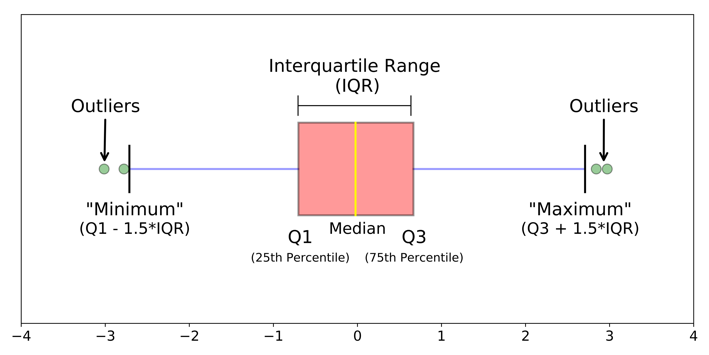
 

Use Mathematical function to remove outlier points and check result with **Box** plot again.

1. ***IQR*** *(****interquartile range****)*

Is a measure of statistical dispersion, being difference between third quartile(75%) and the first quartile(25%). Just like z-score, IQR score can be used to filter out the outliers by keeping only valid values. The IQR of a set of values is calculated as the difference between the upper and lower quartiles, Q3 and Q1. A **box plot** shows the five number summary of a set of data: minimum, lower quartile, median, upper quartile and maximum.



Following function shows the application of IQR and the use of box plot to remove list of outliers as we did in z-score calculation.

﻿ ﻿def drop\_outliers(df, field\_name):

# iqr = 1.5\*(75%A - 25%A)

iqr = 1.5 \* (np.percentile(df[field\_name], 75) - np.percentile(df[field\_name], 25))

# drop those fields whose column values < (25% of field value-iqr) and > (75% of field value + iqr)

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)

﻿

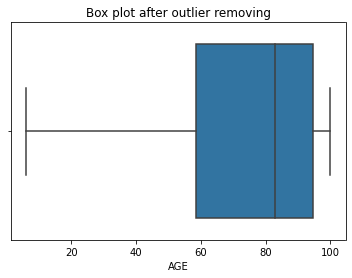
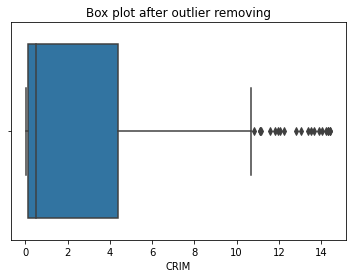
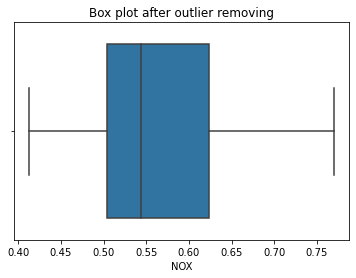
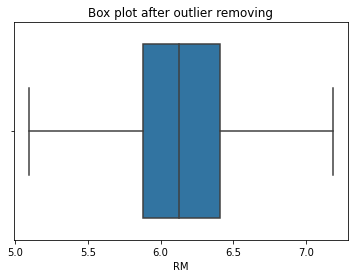
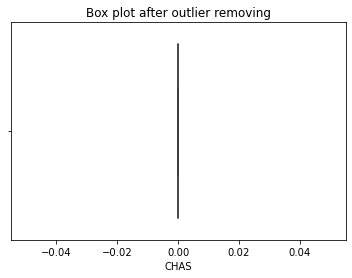
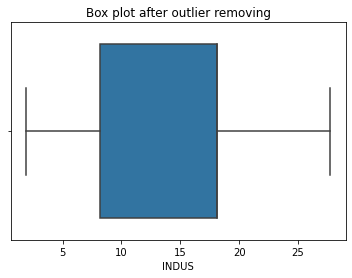
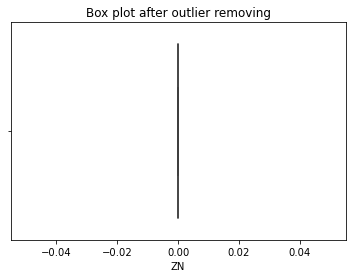
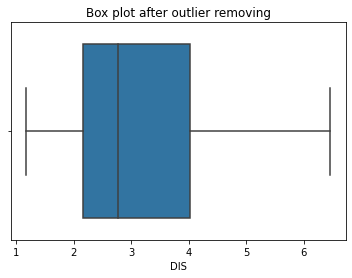
﻿drop\_outliers(df\_combine, 'CRIM')

sns.boxplot(df\_combine['CRIM'])

plt.title("Box plot after outlier removing")

plt.show()

Box plot for some of columns after removing outliers from dataset.

*Applying Regression model and calculating score of the model.*

prediction = linR.predict(x\_test)

score = linR.score(x\_test,y\_test)

# ﻿score **0.6949**

***c*.** ***DBSCAN clustering***

Clustering analysis is an unsupervised learning method that segregates the data points into several bunches or groups depending on their similar quality or characteristics, such that data points in the same groups have similar properties and data points in different groups have different properties in some sense.

Clustering has many methods based on different distance measures. Such as K-Means(distance between points), Affinity propagation(graph distance), Gaussian(Mahalanobis distance to centers), Spectral clustering(graph distance), DBSCAN(distance between nearest points).If we will think all cluster methods use the same approach i.e find similarities and then use it to cluster or group data points. Here it focuses on the application Density-based spatial clustering to remove outliers from dataset.

*So what is the difference between K-Means and DBSCAN clusters?*

In K-Means, clustering depends on mean value of cluster elements and that’s why each data point plays an important role in forming clusters. So a slight change in data points might affect in clustering outcome. One more challenge is, one need to specify number of clusters "k" in order to use this. But sometimes we do not know what should be a reasonable number. While dealing with clusters with different density, size and shape, it would be a challenging condition to detect number of clusters. The task would be more difficult while dealing with a dataset that contains noise and outliers. The nice about DBSCAN is: *it can make clusters of different shapes and sizes from a large dataset but skips noisy points or outliers.* DBSCAN is a density-based clustering approach. DBSCAN makes binary predictions: a point is either an outlier or not. We can use DBSCAN as an outlier detection algorithm because points that do not belong to any cluster get their own class: -1.

DBSCAN algorithm uses two parameters:

minpts: minimum number of points considered within the circle of radius ε.

Core point

a

ε

For example: if minpts = 4 then at least 4 points should lie within this radius (< ε).

epsilon (ε) : a distance parameter that defines the radius to search for nearby neighbours.

There are three types of points after the DBSCAN clustering is complete:

Core — A point that has minimum number of points within the radius ε in a circle

and points that are close enough to those core points together form a cluster.

This is a point that has at least m points within distance ε in a circle.

Border — This is a point that has at least one Core point within the radius ε. If it does

not satisfy the minpts then it becomes a noise.

* If I don’t have any core point and minpts then pt ‘a’ is the noise point.

ε

a

Noise — This is a point that is neither a Core nor a Border.

#convert Pandas dataframe columns to Nmpy array

dbscan\_data = df\_combine.values.astype('float32',copy = False)

dbscan\_data = StandardScaler().fit\_transform(dbscan\_data)

model = DBSCAN(eps = 3, min\_samples = 6, metric = 'euclidean').fit(dbscan\_data)

#df\_combine.shape

# *(506, 14)*

outlier\_df = df\_combine[model.labels\_ == -1]

#*(19, 14)*

cluster\_df = df\_combine[model.labels\_ != -1]

#*[487 rows x 14 columns]*

#Get information about clusters

print(model.labels\_)

﻿[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0

0 0 0 0 0 0 0 0 -1 0 1 -1 0 0 0 0 1 0 -1 -1 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0…………

clusters = Counter(model.labels\_)

print('Number of clusters={}'.format(len(clusters)))

# ﻿*Number of clusters=3*

*Applying Regression model and calculating score of the model.*

﻿prediction = linR.predict(x\_test)

score = linR.score(x\_test,y\_test)

#score ﻿**0.81341**

1. Conclusion

Dropping of data is always a hard step and should be considered in extreme step when we are very sure about the variability in the measurement. So before deleting outliers, it is required to analyse both with or without an outlier. One should examine the changes before deleting the outlier. The result shows the the first Regression model score as 71%. While the scores of model after applying Z-score, IQR and DBSCAN methods are 67%,70% and 81%.

If I have missed any important steps for outlier treatment, I would love to know them in the comments. Thank you for reading.

**Thank you.**

**⊗**