Detect and Remove the Outliers using Python

- Read
- Discuss
- Courses
- Practice

An Outlier is a data-item/object that deviates significantly from the rest of the (so-called normal)objects. They can be caused by measurement or execution errors. The analysis for outlier detection is referred to as outlier mining. There are many ways to detect the outliers, and the removal process is the data frame same as removing a data item from the panda's data frame.

Here pandas data frame is used for a more realistic approach as in real-world projects need to detect the outliers arouse during the data analysis step, the same approach can be used on lists and series-type objects.

Dataset Used For Outlier Detection

The dataset used in this article is the Diabetes dataset and it is preloaded in the sklearn library.

- Python3

```
# Importing

import sklearn

from sklearn.datasets import load_diabetes

import pandas as pd

import matplotlib.pyplot as plt


# Load the dataset

diabetics = load_diabetes()


# Create the dataframe

column_name = diabetics.feature_names
```

```
df_diabetics = pd.DataFrame(diabetics.data)
df_diabetics.columns = column_name
df_diabetics.head()
```

Output:

first five rows of the dataset

Outliers can be detected using visualization, implementing mathematical formulas on the dataset, or using the statistical approach. All of these are discussed below.

Outliers Visualization

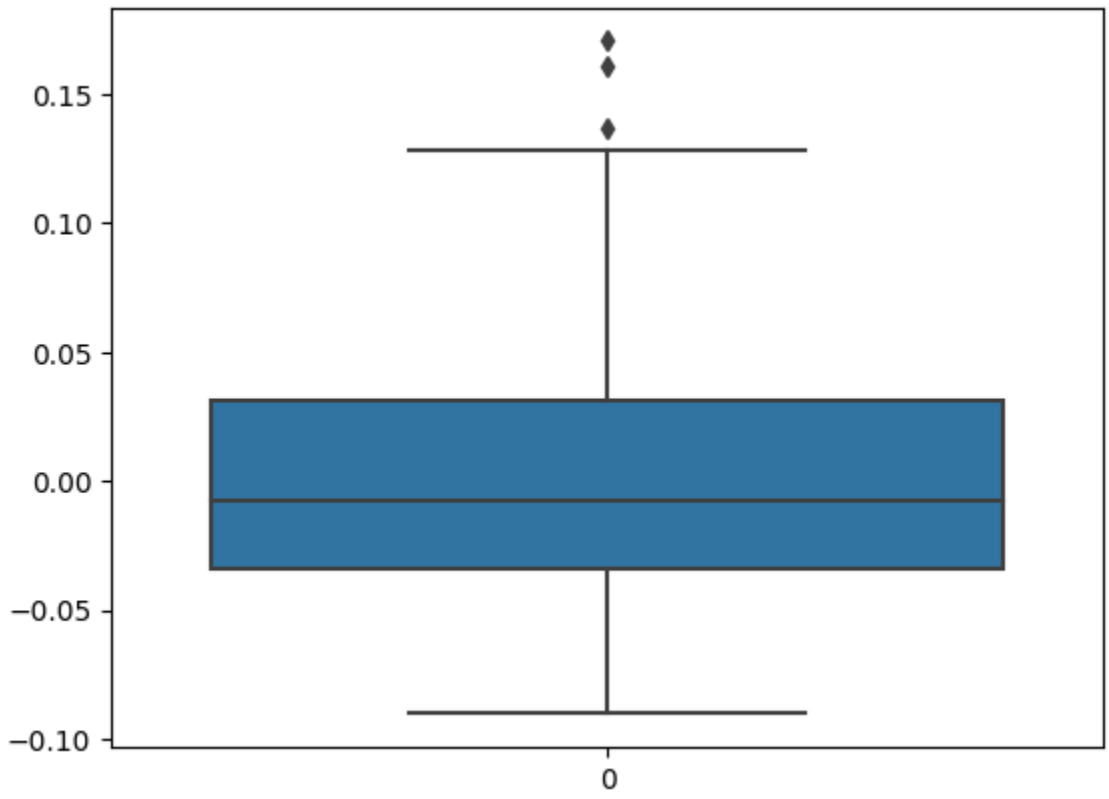
Visualizing Outliers Using Box Plot

It captures the summary of the data effectively and efficiently with only a simple box and whiskers. Boxplot summarizes sample data using 25th, 50th, and 75th percentiles. One can just get insights(quartiles, median, and outliers) into the dataset by just looking at its boxplot.

- Python3

```
# Box Plot

import seaborn as sns

sns.boxplot(df_diabetics['bmi'])
```

Output:



Outliers present in the bmi columns

In the above graph, can clearly see that values above 10 are acting as outliers.

- Python3

```
# Position of the Outlier

import numpy as np

print(np.where(df_diabetics['bmi']>0.12))
```

Output:

(array([32, 145, 256, 262, 366, 367, 405]),)

Visualizing Outliers Using ScatterPlot.

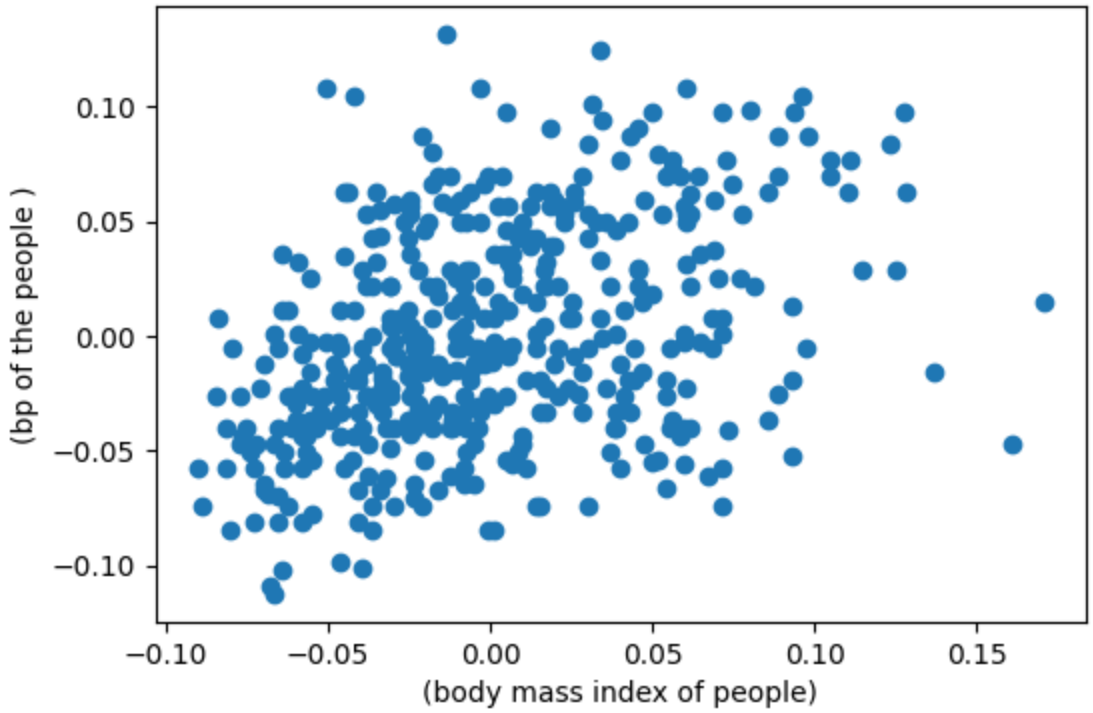
It is used when you have paired numerical data and when your dependent variable has multiple values for each reading independent variable, or when trying to determine the relationship between the two variables. In the process of utilizing the scatter plot, one can also use it for outlier detection.

To plot the scatter plot one requires two variables that are somehow related to each other. So here, 'Proportion of non-retail business acres per town' and 'Full-value property-tax rate per $10,000' are used whose column names are "INDUS" and "TAX" respectively.

- Python3

```
# Scatter plot

fig, ax = plt.subplots(figsize = (6,4))

ax.scatter(df_diabetics['bmi'],df_diabetics['bp'])


# x-axis label

ax.set_xlabel('(body mass index of people)')


# y-axis label

ax.set_ylabel('(bp of the people )')

plt.show()
```

Output:



Scatter plot of bp and bmi

Looking at the graph can summarize that most of the data points are in the bottom left corner of the graph but there are few points that are exactly;y opposite that is the top right corner of the graph. Those points in the top right corner can be regarded as Outliers.

Using approximation can say all those data points that are x>20 and y>600 are outliers. The following code can fetch the exact position of all those points that satisfy these conditions.

Outliers in BMI and BP Column Combined

- Python3

```
# Position of the Outlier

print(np.where((df_diabetics['bmi']>0.12) & (df_diabetics['bp']<0.8)))
```

Output:

(array([32, 145, 256, 262, 366, 367, 405]),)

Z-score

Z- Score is also called a standard score. This value/score helps to understand that how far is the data point from the mean. And after setting up a threshold value one can utilize z score values of data points to define the outliers.

Zscore = (data_point -mean) / std. deviation

- Python3

```
# Z score
from scipy import stats
import numpy as np


z = np.abs(stats.zscore(df_diabetics['age']))
print(z)
```

Output:

[0.80050009 0.03956713 1.79330681 1.87244107 0.11317236 1.94881082
0.9560041 1.33508832 0.87686984 1.49059233 2.02518057 0.57139085
0.34228161 0.11317236 0.95323959 1.1087436 0.11593688 1.48782782
0.80326461 0.57415536 1.03237385 1.79607132 1.79607132 0.95323959
1.33785284 1.41422259 2.25428981 0.49778562 1.10597908 1.41145807
1.26148309 0.49778562 0.72413034 0.6477606 0.34228161 1.02960933
0.26591186 0.19230663 0.03956713 0.03956713 0.11317236 2.10155031
1.26148309 0.41865135 0.95323959 0.57139085 1.18511334 1.64333183
1.41145807 0.87963435 0.72413034 1.25871858 1.1087436 0.19230663
1.03237385 0.87963435 0.87963435 0.57415536 0.87686984 1.33508832
1.49059233 0.87963435 0.57415536 0.72689486 1.41145807 0.9560041
0.19230663 0.87686984 0.80050009 0.34228161 0.03956713 0.03956713
1.33508832 0.26591186 0.26591186 0.19230663 0.65052511 2.02518057
0.11317236 2.17792006 1.48782782 0.26591186 0.34504612 0.80326461
0.03680262 0.95323959 1.49059233 0.95323959 1.1087436 0.9560041
0.26591186 0.95323959 0.42141587 1.03237385 1.64333183 1.49059233
1.18234883 0.57415536 0.03680262 0.03956713 0.34228161 0.34228161]

The above output is just a snapshot of part of the data; the actual length of the list(z) is 506 that is the number of rows. It prints the z-score values of each data item of the column

Now to define an outlier threshold value is chosen which is generally 3.0. As 99.7% of the data points lie between +/- 3 standard deviation (using Gaussian Distribution approach).

Rows where Z value is greater than 2

- Python3

```
threshold = 2


# Position of the outlier

print(np.where(z > 2))
```

Output:

(array([10, 26, 41, 77, 79, 106, 131, 204, 223, 226, 242, 311, 321,344, 374, 402]),)

IQR (Inter Quartile Range)

IQR (Inter Quartile Range) Inter Quartile Range approach to finding the outliers is the most commonly used and most trusted approach used in the research field.

IQR = Quartile3 – Quartile1

- Python3

```
# IQR

Q1 = np.percentile(df_diabetics['bmi'], 25, method='midpoint')

Q3 = np.percentile(df_diabetics['bmi'], 75, method='midpoint')

IQR = Q3 - Q1

print(IQR)
```

Output:

0.06520763046978838

Syntax: numpy.percentile(arr, n, axis=None, out=None)
Parameters :
arr :input array.
n : percentile value.

To define the outlier base value is defined above and below dataset's normal range namely Upper and Lower bounds, define the upper and the lower bound (1.5*IQR value is considered) :

*upper = Q3 +1.5*IQR*

*lower = Q1 – 1.5*IQR*

In the above formula as according to statistics, the 0.5 scale-up of IQR (new_IQR = IQR + 0.5*IQR) is taken, to consider all the data between 2.7 standard deviations in the Gaussian Distribution.

- Python3

```
# Above Upper bound

upper=Q3+1.5*IQR

upper_array=np.array(df_diabetics['bmi']>=upper)

print("Upper Bound:",upper)

print(upper_array.sum())


#Below Lower bound

lower=Q1-1.5*IQR

lower_array=np.array(df_diabetics['bmi']<=lower)

print("Lower Bound:",lower)

print(lower_array.sum())
```

Output:

Upper Bound: 0.12879000811776306

3

Lower Bound: -0.13204051376139045

0

Removing the outliers

For removing the outlier, one must follow the same process of removing an entry from the dataset using its exact position in the dataset because in all the above methods of detecting the outliers end result is the list of all those data items that satisfy the outlier definition according to the method used.

References: How to delete exactly one row in python?

dataframe.drop(row index,inplace=True)

The above code can be used to drop a row from the dataset given the row_indexes to be dropped. Inplace =True is used to tell Python to make the required change in the original dataset. row_index can be only one value or list of values or NumPy array but it must be one dimensional.

Example:

df_diabetics.drop(lists[0],inplace = True)

Full Code: Detecting the outliers using IQR and removing them.

- Python3

```
# Importing
import sklearn
from sklearn.datasets import load_diabetes
import pandas as pd

# Load the dataset
diabetes = load_diabetes()

# Create the dataframe
column_name = diabetes.feature_names
df_diabetes = pd.DataFrame(diabetes.data)
df_diabetes .columns = column_name
df_diabetes .head()
print("Old Shape: ", df_diabetes.shape)

''' Detection '''
# IQR
# Calculate the upper and lower limits
Q1 = df_diabetes['bmi'].quantile(0.25)
Q3 = df_diabetes['bmi'].quantile(0.75)
IQR = Q3 - Q1
lower = Q1 - 1.5*IQR
upper = Q3 + 1.5*IQR
```

```
# Create arrays of Boolean values indicating the outlier rows
upper_array = np.where(df_diabetes['bmi']>=upper)[0]
lower_array = np.where(df_diabetes['bmi']<=lower)[0]

# Removing the outliers
df_diabetes.drop(index=upper_array, inplace=True)
df_diabetes.drop(index=lower_array, inplace=True)

# Print the new shape of the DataFrame
print("New Shape: ", df_diabetes.shape)
```

Output:

Old Shape: (442, 10)

New Shape: (439, 10)