Detecting and Treating Outliers | Treating the odd one out!

Home

Harika Bonthu — Published On May 21, 2021 and Last Modified On June 26th, 2023

Beginner Data Science Data Visualization Python Structured Data

This article was published as a part of the Data Science Blogathon

Wow, these are lovely! Wait, where does this yellow Tulip come from?

Introduction:

One of the most important steps as part of data preprocessing is detecting and treating the outliers as they can negatively affect the statistical analysis and the training process of a machine learning algorithm resulting in lower accuracy.

Table of contents

Quiz Time

Sharpen your knowledge of Detecting and Treating Outliers!

Start Quiz

We all have heard of the idiom 'odd one out which means something unusual in comparison to the others in a group.

Similarly, an Outlier is an observation in a given dataset that lies far from the rest of the observations. That means an outlier is vastly larger or smaller than the remaining values in the set.

Why do they occur?

An outlier may occur due to the variability in the data, or due to experimental error/human error.

They may indicate an experimental error or heavy skewness in the data(heavy-tailed distribution).

What do they affect?

In statistics, we have three measures of central tendency namely Mean, Median, and Mode. They help us describe the data.

Mean is the accurate measure to describe the data when we do not have any outliers present.

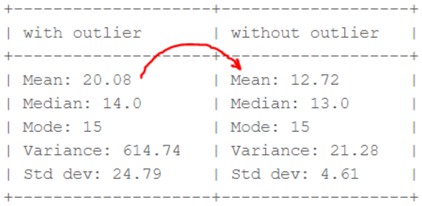
Median is used if there is an outlier in the dataset.

Mode is used if there is an outlier AND about ½ or more of the data is the same.

'Mean' is the only measure of central tendency that is affected by the outliers which in turn impacts Standard deviation.

Example:

Consider a small dataset, sample= [15, 101, 18, 7, 13, 16, 11, 21, 5, 15, 10, 9]. By looking at it, one can quickly say '101' is an outlier that is much larger than the other values.



```
+--------------------+-------------------+
| with outlier       | without outlier   |
+--------------------+-------------------+
| Mean: 20.08        | Mean: 12.72       |
| Median: 14.0       | Median: 13.0      |
| Mode: 15           | Mode: 15          |
| Variance: 614.74   | Variance: 21.28   |
| Std dev: 24.79     | Std dev: 4.61     |
+--------------------+-------------------+
```

computation with and without outlier (Image by author)

From the above calculations, we can clearly say the Mean is more affected than the Median.

Detecting Outliers

If our dataset is small, we can detect the outlier by just looking at the dataset. But what if we have a huge dataset, how do we identify the outliers then? We need to use visualization and mathematical techniques.

Below are some of the techniques of detecting outliers

Detecting and Treating Outliers | Treating the odd one out!

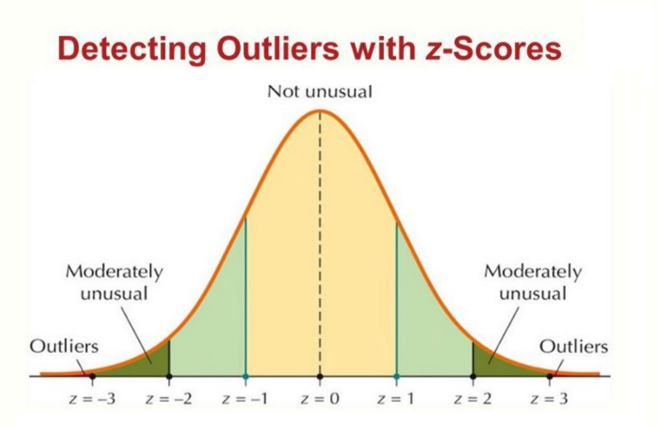
- Inter Quantile Range(IQR)

4.1 Detecting outliers using Boxplot:

Python code for boxplot is:

4.2 Detecting outliers using the Z-scores

Criteria: any data point whose Z-score falls out of 3rd standard deviation is an outlier.



Detecting Outliers with Z-scores. Image source: https://laptrinhx.com/

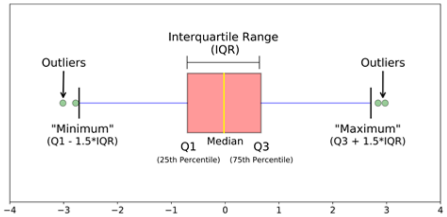
Steps:

- loop through all the data points and compute the Z-score using the formula (Xi-mean)/std.
- define a threshold value of 3 and mark the datapoints whose absolute value of Z-score is greater than the threshold as outliers.

```
import numpy as np
outliers = []
def detect_outliers_zscore(data):
    thres = 3
    mean = np.mean(data)
    std = np.std(data)
    # print(mean, std)
    for i in data:
        z_score = (i-mean)/std
        if (np.abs(z_score) > thres):
            outliers.append(i)
    return outliers# Driver code
sample_outliers = detect_outliers_zscore(sample)
print("Outliers from Z-scores method: ", sample_outliers)
```

The above code outputs: **Outliers from Z-scores method: [101]**

Detecting and Treating Outliers | Treating the odd one out!



IQR to detect Outliners

Criteria: data points that lie 1.5 times of IQR above Q3 and below Q1 are outliers. This shows in detail about outlier treatment in Python.

steps:

- Sort the dataset in ascending order
- calculate the 1st and 3rd quartiles(Q1, Q3)
- compute IQR=Q3-Q1
- compute lower bound = (Q1–1.5*IQR), upper bound = (Q3+1.5*IQR)
- loop through the values of the dataset and check for those who fall below the lower bound and above the upper bound and mark them as outliers

Python Code:

```
outliers = []
def detect_outliers_iqr(data):
    data = sorted(data)
    q1 = np.percentile(data, 25)
    q3 = np.percentile(data, 75)
    # print(q1, q3)
    IQR = q3-q1
    lwr_bound = q1-(1.5*IQR)
    upr_bound = q3+(1.5*IQR)
    # print(lwr_bound, upr_bound)
    for i in data:
        if (i<lwr_bound or i>upr_bound):
            outliers.append(i)
    return outliers# Driver code
sample_outliers = detect_outliers_iqr(sample)
print("Outliers from IQR method: ", sample_outliers)
```

The above code outputs: **Outliers from IQR method: [101]**

How to Handle Outliers?

Till now we learned about detecting the outliers. The main question is how to deal with outliers?

Below are some of the methods of treating the outliers

1. **Trimming/Remove the outliers**

   In this technique, we remove the outliers from the dataset. Although it is not a good practice to follow.

   Python code to delete the outlier and copy the rest of the elements to another array

Detecting and Treating Outliers | Treating the odd one out!

```
a = np.delete(sample, np.where(sample==i))
print(a)
# print(len(sample), len(a))
```

The outlier '101' is deleted and the rest of the data points are copied to another array 'a'.

2. **Quantile based flooring and capping**

   In this technique, the outlier is capped at a certain value above the 90th percentile value or floored at a factor below the 10th percentile value.

   Python code to delete the outlier and copy the rest of the elements to another array.

```
# Computing 10th, 90th percentiles and replacing the outliers
tenth_percentile = np.percentile(sample, 10)
ninetieth_percentile = np.percentile(sample, 90)
# print(tenth_percentile, ninetieth_percentile)b =
np.where(sample<tenth_percentile, tenth_percentile, sample)
b = np.where(b>ninetieth_percentile, ninetieth_percentile, b)
# print("Sample:", sample)
print("New array:",b)
```

   The above code outputs: **New array:** [15, 20.7, 18, 7.2, 13, 16, 11, 20.7, 7.2, 15, 10, 9]

   The data points that are lesser than the 10th percentile are replaced with the 10th percentile value and the data points that are greater than the 90th percentile are replaced with 90th percentile value.

3. **Mean/Median imputation**

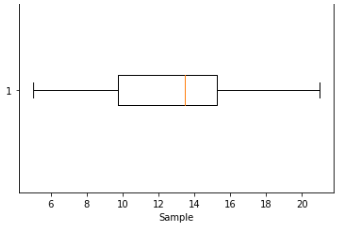
   As the mean value is highly influenced by the outliers, it is advised to replace the outliers with the median value.

   Python Code:

```
median = np.median(sample)# Replace with median
for i in sample_outliers:
c = np.where(sample==i, 14, sample)
print("Sample: ", sample)
print("New array: ",c)
# print(x.dtype)
```

Visualizing the data after treating the outlier

```
plt.boxplot(c, vert=False)
plt.title("Boxplot of the sample after treating the outliers")
plt.xlabel("Sample")
```



Data after treating Outliner

Frequently Asked Questions

Q1. Which methods are used to treat outliers?

A. There are several methods commonly used to treat outliers, depending on the nature of the data and the specific analysis being conducted. Here are a few common approaches:

1. **Deletion**: This method involves simply removing the outliers from the dataset. It can be appropriate when the outliers are believed to be due to data entry errors or measurement errors. However, caution should be exercised as deleting outliers without a good reason can potentially bias the analysis and distort the results.

2. **Transformation**: Transforming the data using mathematical functions can sometimes reduce the impact of outliers. Common transformations include taking the logarithm, square root, or reciprocal of the data. These transformations can help make the data more normally distributed and stabilize the variance.

3. **Winsorization**: Winsorization replaces extreme data values with less extreme values. The process involves capping or truncating the extreme values at a certain percentile (e.g., replacing values above the 95th percentile with the value at the 95th percentile). This approach reduces the influence of outliers while still retaining some information from the extreme values.

4. **Imputation**: Instead of deleting outliers, they can be replaced with estimated values. Imputation techniques include replacing outliers with the mean, median, or another suitable value based on the characteristics of the data. Imputation should be done carefully, as it may introduce bias if not appropriately handled.

5. **Robust methods**: Robust statistical methods are designed to be less sensitive to outliers. These methods estimate parameters using robust estimators that are not heavily influenced by extreme values. For example, the median is a robust measure of central tendency that is less affected by outliers compared to the mean.

6. **Model-based approaches**: In some cases, outliers can be detected and treated using specific models. For example, in regression analysis, influential outliers can be identified using diagnostic measures like Cook's distance or studentized residuals. Once identified, the outliers can be downweighted or excluded from the analysis.

The choice of method depends on the specific context, the nature of the data, and the goals of the analysis. It's important to consider the underlying reasons for the outliers and the potential impact of their treatment on the results.

Q2. When should I remove outliers?

situations where removing outliers may be considered:

1. **Data entry errors or measurement errors**: If you have strong evidence or suspicion that outliers are due to errors in data entry or measurement, it may be appropriate to remove them. For example, if you have a dataset of human heights and you notice an entry that is clearly a typo (e.g., a height of 8 feet), removing such an outlier would make sense.

2. **Violations of assumptions**: Some statistical analyses assume certain distributions or relationships between variables. If outliers are causing significant violations of these assumptions and are unlikely to be a part of the underlying population or process you are studying, their removal may be justified. For instance, if you're performing a linear regression analysis and outliers are causing a substantial departure from linearity, removing them can help ensure the validity of the regression model.

3. **Sensitive analysis**: In certain cases, outliers can have a disproportionate impact on the results, leading to biased estimates or inflated standard errors. In such situations, removing outliers may be considered to obtain more accurate and reliable results. However, it is crucial to document and justify the removal of outliers, as their removal can affect the interpretation of the analysis.

4. **Model performance improvement**: Outliers can sometimes adversely affect the performance of predictive models. If outliers are significantly influencing the model's predictions or leading to poor model performance, removing them might improve the model's accuracy or generalizability.

5. **Specific domain knowledge**: Domain knowledge or subject matter expertise can provide insights into whether outliers are meaningful or anomalous observations. If you have a good understanding of the data generating process and know that certain extreme values are implausible or unrelated to the phenomenon being studied, removing them could be reasonable.

However, it's important to exercise caution when removing outliers. Blindly removing outliers without a solid justification or a clear understanding of their origin can lead to biased or misleading results. It is advisable to carefully assess the impact of outliers on your analysis, explore alternative methods to handle outliers, and consider robust statistical techniques that are less sensitive to extreme values before deciding to remove them. Additionally, documenting the rationale and steps taken for outlier removal is crucial for transparency and reproducibility of your analysis.

Summary:

In this blog, we learned about an important phase of data preprocessing which is treating outliers. We now know different methods of detecting and treating outliers.

References:

Z-score for Outlier detection

IQR for outlier detection

Python numpy.where() Method

GitHub repo to check out the Jupyter notebook

I hope this blog helps understand the outliers concept. Please do upvote if you like it. Happy learning !! 😊

blogathon data preprocessing Outlier Detection

About the Author

Harika Bonthu

Our Top Authors

CHIRA Arnab Prateek view more

Download

Analytics Vidhya App for the Latest blog/Article

Previous Post

Comprehensive Guide on Python Data types with Examples

Next Post

Importance of Cross Validation: Are Evaluation Metrics enough?

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name*

Email*

Website

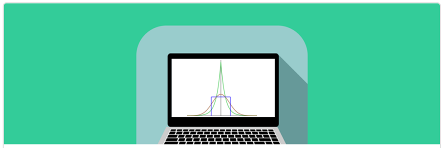
☑ Notify me of follow-up comments by email.

☑ Notify me of new posts by email.

Submit

Detecting and Treating Outliers | Treating the odd one out!

Top Resources



[25 Probability and Statistics Questions to Ace your Data Science..](#)

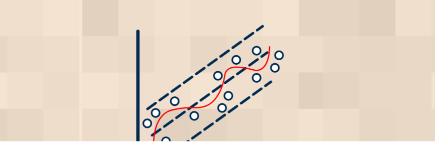
CHIRAG GOYAL - APR 23, 2021

[10 Best AI Image Generator Tools to Use in 2023](#)

avcontentteam - AUG 17, 2023

[Top 40 Machine Learning Questions & Answers for Beginners and..](#)

1201904 - APR 30, 2017



[Everything you need to Know about Linear Regression!](#)

KAVITA MALI - OCT 04, 2021

Download App

Analytics Vidhya
About Us
Our Team
Careers
Contact us

Companies
Post Jobs
Trainings
Hiring Hackathons
Advertising

Data Scientists
Blog
Hackathon
Join the Community
Apply Jobs

Visit us

Privacy Policy Terms of Use Refund Policy