# **Reference guide: How to handle outliers**

## How to solve a problem like outliers

Once you’ve detected outliers in your dataset—whether global, contextual, or collective—how do you handle them? When it comes to exploratory data analysis, or EDA, there are essentially three main ways to handle them: delete, reassign, or leave them in.

Whether you keep outliers as they are, delete them, or re-assign values is a decision that you make on a dataset-by-dataset basis. To help you make the decision, you can start with these general guidelines:

* **Delete them**: If you are sure the outliers are mistakes, typos, or errors and the dataset will be used for modeling or machine learning, then you are more likely to decide to delete outliers. Of the three choices, you’ll use this one the least.
* **Reassign them**: If the dataset is small and/or the data will be used for modeling or machine learning, you are more likely to choose a path of deriving new values to replace the outlier values.
* **Leave them**: For a dataset that you plan to do EDA/analysis on and nothing else, or for a dataset you are preparing for a model that is resistant to outliers, it is most likely that you are going to leave them in.

In this reading, you will learn about some Python techniques for deleting and reassigning outliers.

### **Delete them**

One way to handle outlier values is by deleting them. You can start by coding a box plot to help you visualize outliers, as shown here:

sns.boxplot(x=df[‘number\_of\_strikes’])

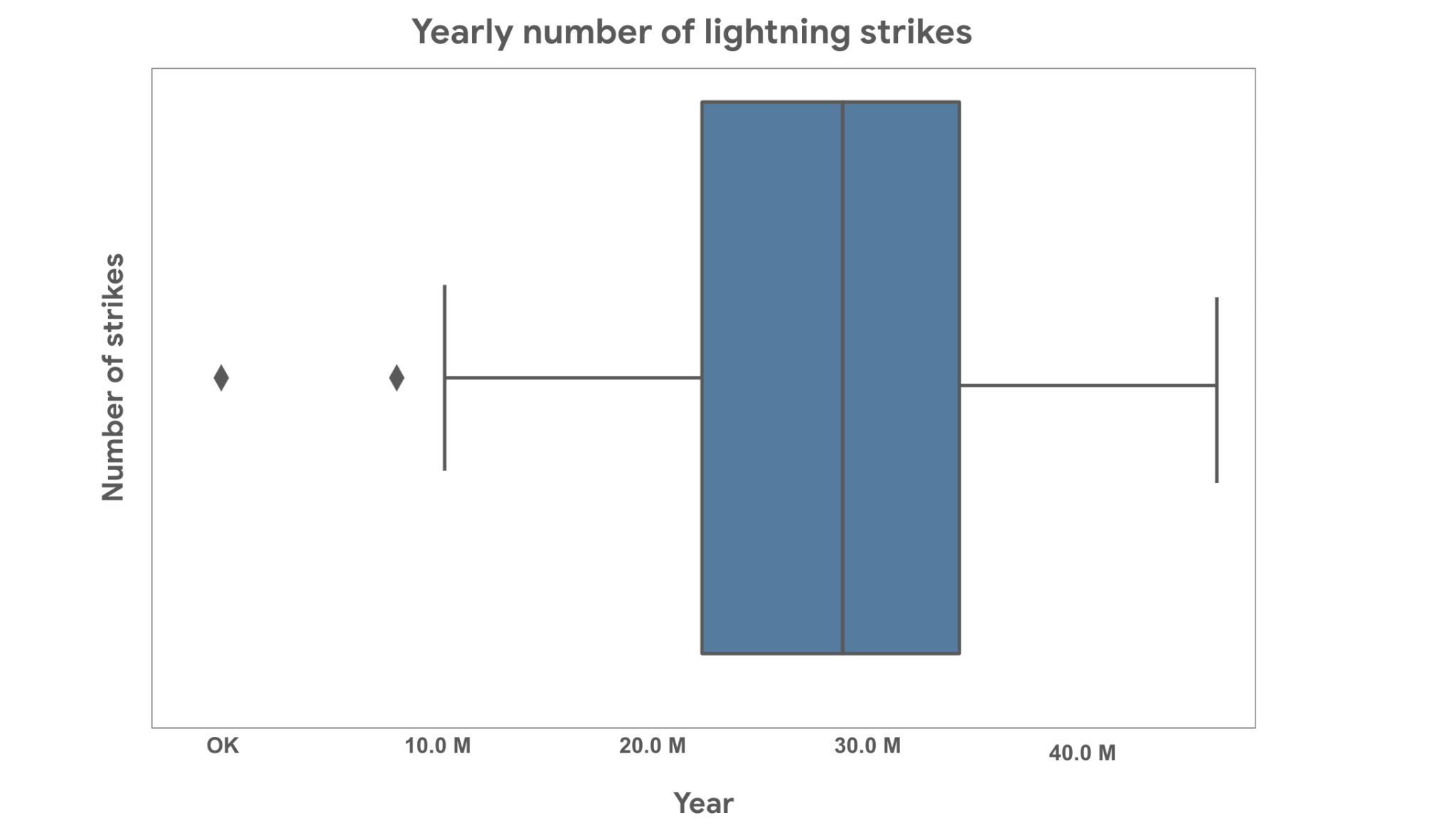
plt.xticks(np.array([readable\_numbers(x) for x in g.get\_xticks()]))

plt.xlabel(‘Year’)

plt.ylabel(‘Number of strikes’)

plt.title(‘Yearly number of lightning strikes’)

plt.show()



[**Alt-Text**: A box plot is shown with two outliers to the left, outside of the left whisker.]

The following code can be used to find the lower limit.

percentile25 = df[‘number\_of\_strikes’].quantile(0.25)

percentile75 = df[‘number\_of\_strikes’].quantile(0.75)

iqr = percentile75 - percentile25

upper\_limit = percentile75 + 1.5 \* iqr

lower\_limit = percentile25 - 1.5 \* iqr

print(‘Lower limit is: ‘ + readable\_numbers(lower\_limit))

Lower limit is: 8.6M

In the next code block, you’ll find the code to identify and print the actual outlier data points.

df[df[‘number\_of\_strikes’] < lower\_limit]

Year number\_of\_strikes number\_of\_strikes\_readable

1 2019 209166 209K

33 1987 7378836 7.4M

Once you know the points, and you have decided to delete the outliers, you can use a line of code like this to delete the points:

df = df[df['number\_of\_strikes']>=lower\_limit].copy()

Next, you’ll consider reassigning outliers by deriving new values that are a better fit for the dataset.

### **Reassign them**

Another approach: instead of deleting outliers, you can reassign them, that is, change the values to ones that fit within the general distribution of the dataset. There are two common ways to do this:

1. **Create a floor and ceiling at the quantile:** Walls are placed at the 90th and 10th percentile of the distribution of data values. Any value above the 90% mark or below the 10% mark are changed to fit within the walls you set. Here is an example of what that code might look like:

tenth\_percentile = np.percentile(df[‘number\_of\_strikes’], 10)

ninetieth\_percentile = np.percentile(df[‘number\_of\_strikes’], 90)

df[‘number\_of\_strikes’] = df[‘number\_of\_strikes’].apply(lambda x: tenth\_percentile if x < tenth\_percentile else ninetieth\_percentile if x > ninetieth\_percentile else x)

1. **Median imputation:** In some cases, it might be best to reassign all outlier values to match the median or mean value. This will ensure that your median and distribution are based solely on the non-outlier values, leaving the original outliers excluded. The actual imputation or reassigning of values can be pretty simple if you’ve already found the outliers. The following code block acknowledges the outliers and finds the median.

median = np.median([df[‘number\_of\_strikes’] < lower\_limit])

df['number\_of\_strikes'] = np.where(df['number\_of\_strikes'] < lower\_limit, median, df['number\_of\_strikes'])

## Key Takeaways

After detecting the outliers in a dataset, it is essential that you determine a strategy for how to handle them. Because every dataset and data-based problem is different, your strategy will vary. For the most part, you will be choosing between deleting, reassigning, or leaving outliers.