**Don’t Throw Away Your Outliers!**

**First, look for the causes of errors in your dataset.**

**After** [**detecting outliers**](https://hennie-de-harder.medium.com/are-you-using-feature-distributions-to-detect-outliers-48e2ae3309) **or anomalies, you need to decide how to handle them. This post explains techniques in taking care of outliers.**

**The first step, investigation**

Investigate your outliers. Why did they occur? Are they truly errors? Will they never happen in real life? They were in the data, so it’s important to discover where the errors came from. Outliers may occur due to experimental or human error. Something could go wrong when loading or processing the data. After discovering the cause, you can decide what to do about it.

Is the problem solved? If you decide the outliers shouldn’t be in the data and you remove them, make sure you can define why. Or even better: document it. If you discover it’s possible new data can have the same values as the outliers, you should take care of them using other techniques, like the ones described below.

**Taking care of outliers**

If the outliers are just irregular compared to the other data points, but can happen, take care of them! You can try several techniques to improve the results of a machine learning model without removing the outliers.

These techniques are based on transforming the data:

**Cap the data**

A special transformation you can try is capping or [**winsorizing**](https://www.statology.org/winsorize/)the data. This is a method in which you set the maximum values of a feature to a certain value. You can decide to set the bottom 2% to the value of the second percentile, and the top 2% to the value of the 98th percentile. Here is [an example with python code](https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mstats.winsorize.html).

**Other data transformations**

Besides capping, there are other ways to transform the data. You can use scalers, log transformation, or binning.

Scalers are a technique you should try, because scaling can have a huge impact on the outcome of your model. Here you can find [a comparison between different scalers](https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html) by sci-kit learn. You might want to [read more about the **RobustScaler**](https://machinelearningmastery.com/robust-scaler-transforms-for-machine-learning/), it’s designed to handle outliers.

**Log transformation** is preferred when a variable grows or decays on an exponential scale, because the variable is more on a linear scale after the transformation. The impact of the tail is reduced.

**Binning**, or bucketing, replaces a range of values to a single representative value. You can use binning in different ways: you can set the [distance](https://pandas.pydata.org/docs/reference/api/pandas.cut.html) of the bins, you can use the [frequency](https://pandas.pydata.org/docs/reference/api/pandas.qcut.html) (every bin gets the same number of observations), or [sampling](https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.binned_statistic.html).



Photo by [Marina Khrapova](https://unsplash.com/es/@mimiori?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com?utm_source=medium&utm_medium=referral)

Techniques based on models:

**Use regularization**

Regularization reduces variance and that makes it useful in handling outliers. The two most common types of regularization are Lasso and Ridge regularization, also known as L1 and L2 regularization respectively.

Regularization works by penalizing high-valued regression coefficients. It simplifies the model and makes it more robust and less prone to overfitting.

Regularizers are implemented in all famous ML packages. For instance, take a look at the scikit-learn [Lasso](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html#sklearn.linear_model.Lasso) and [Ridge](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html) or [add regularization to Keras layers](https://keras.io/api/layers/regularizers/).

**Model selection**

Some models are better in handling outliers than others. For instance, if you are working on a regression problem, you could try random sample consensus (RANSAC) regression or [Theil-Sen regression](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.TheilSenRegressor.html#sklearn.linear_model.TheilSenRegressor). [RANSAC](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RANSACRegressor.html) uses samples from the data, builds a model and uses the residuals to separate inliers from outliers. The final model uses only the inliers.

If you want to use all data points, you can use tree based models. Tree based models are generally better in handling outliers than other models. An intuitive explanation is: trees care about on which side of the split a data point is located, not about the distance between the split and the data point.

**Change the error metric**

The final option is to change the error metric. An easy example: if you are using mean squared error, you are punishing outliers harder, because you square the errors. You might want to switch to mean absolute error, because you take the absolute value of the error instead of squaring it.

**Summary**

Before throwing away your outliers, investigate them. Look for the causes the errors came into the dataset. If you can’t solve them, handle the data using transformations, add regularization or choose an appropriate model or error metric.