**Feature Scaling**

What is feature scaling? Feature scaling is a way of transforming your data into a common range of values. There are two common scalings:

1. Standardizing
2. Normalizing

**Standardizing**

**Standardizing** is completed by taking each value of your column, subtracting the mean of the column, and then dividing by the standard deviation of the column. In Python, let's say you have a column in df called height. You could create a standardized height as:

df["height\_standard"] = (df["height"] - df["height"].mean()) / df["height"].std()

This will create a new "standardized" column where each value is a comparison to the mean of the column, and a new, standardized value can be interpreted as the number of standard deviations the original height was from the mean. This type of feature scaling is by far the most common of all techniques (for the reasons discussed here, but also likely because of precedent).

**Normalizing**

A second type of feature scaling that is very popular is known as **normalizing**. With normalizing, data are scaled between 0 and 1. Using the same example as above, we could perform normalizing in Python in the following way:

df["height\_normal"] = (df["height"] - df["height"].min()) / \

(df["height"].max() - df['height'].min())

**When Should I Use Feature Scaling?**

In many machine learning algorithms, the result will change depending on the units of your data. This is especially true in two specific cases:

1. When your algorithm uses a distance-based metric to predict.
2. When you incorporate regularization.

**Distance Based Metrics**

In future lessons, you will see one common supervised learning technique that is based on the distance points are from one another called [Support Vector Machines (or SVMs)](https://en.wikipedia.org/wiki/Support_vector_machine). Another technique that involves distance based methods to determine a prediction is [k-nearest neighbors (or k-nn)](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm). With either of these techniques, choosing not to scale your data may lead to drastically different (and likely misleading) ending predictions.

For this reason, choosing some sort of feature scaling is necessary with these distance based techniques.

**Regularization**

When you start introducing regularization, you will again want to scale the features of your model. The penalty on particular coefficients in regularized linear regression techniques depends largely on the scale associated with the features. When one feature is on a small range, say from 0 to 10, and another is on a large range, say from 0 to 1 000 000, applying regularization is going to unfairly punish the feature with the small range. Features with small ranges need to have larger coefficients compared to features with large ranges in order to have the same effect on the outcome of the data. (Think about how ab=ba*ab*=*ba* for two numbers a*a* and b*b*.) Therefore, if regularization could remove one of those two features with the same net increase in error, it would rather remove the small-ranged feature with the large coefficient, since that would reduce the regularization term the most.

Again, this means you will want to scale features any time you are applying regularization.

* [A useful Quora post on the importance of feature scaling when using regularization.](https://www.quora.com/Why-do-we-normalize-the-data)

A point raised in the article above is that feature scaling can speed up convergence of your machine learning algorithms, which is an important consideration when you scale machine learning applications.