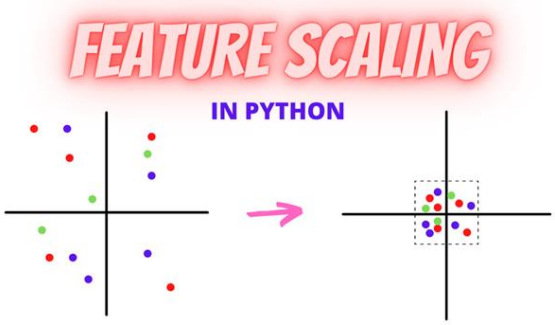
FEATURE SCALING

CCDATSCL | Data Science

Feature Scaling

- In data pre-processing, we try to change the data in such a way that the model can process it without any problems.
- Feature Scaling is one such process in which we transform the data into a better version.
- Feature Scaling is done to normalize the features in the dataset into a finite range.



House ID	Price (USD)	Lot Size (sq ft)	Distance to City Center (km)	Number of Rooms	Age of House (years)
H001	850,000	6,500	12.4	3	28
H002	1,250,000	12,300	5.6	5	12
H003	420,000	3,200	18.9	2	42
H004	2,300,000	20,500	3.1	7	5
H005	690,000	8,700	15.2	4	31

For example, in a house price dataset. It will have many features such as number of bedrooms, lot size, distance to the city center, number of rooms and house age.

House ID	Price (USD)	Lot Size (sq ft)	Distance to City Center (km)	Number of Rooms	Age of House (years)
H001	850,000	6,500	12.4	3	28
H002	1,250,000	12,300	5.6	5	12
H003	420,000	3,200	18.9	2	42
H004	2,300,000	20,500	3.1	7	5
H005	690,000	8,700	15.2	4	31

For the price column, we can see that the price has a wide range which starts at 420 thousand and up to 2.3 million.

House ID	Price (USD)	Lot Size (sq ft)	Distance to City Center (km)	Number of Rooms	Age of House (years)
H001	850,000	6,500	12.4	3	28
H002	1,250,000	12,300	5.6	5	12
H003	420,000	3,200	18.9	2	42
H004	2,300,000	20,500	3.1	7	5
H005	690,000	8,700	15.2	4	31

For the Lot Size column, its values also have a wide range starting from 3,200 up to 20,500

House ID	Price (USD)	Lot Size (sq ft)	Distance to City Center (km)	Number of Rooms	Age of House (years)
H001	850,000	6,500	12.4	3	28
H002	1,250,000	12,300	5.6	5	12
H003	420,000	3,200	18.9	2	42
H004	2,300,000	20,500	3.1	7	5
H005	690,000	8,700	15.2	4	31

While Distance, Rooms, and Age are in very small ranges

This is a huge difference in the range of all features.

House ID	Price (USD)	Lot Size (sq ft)	Distance to City Center (km)	Number of Rooms	Age of House (years)
H001	850,000	6,500	12.4	3	28
H002	1,250,000	12,300	5.6	5	12
H003	420,000	3,200	18.9	2	42
H004	2,300,000	20,500	3.1	7	5
H005	690,000	8,700	15.2	4	31

Algorithms that use distances (kNN, K-Means), gradient-based optimization (linear/logistic regression), or regularization will treat Price and Lot Size as more "important"…

Simply because their numeric scales dominate the others. That is unless we scale the features.

Why Feature Scaling?

Real Life Datasets have many features with a wide range of values

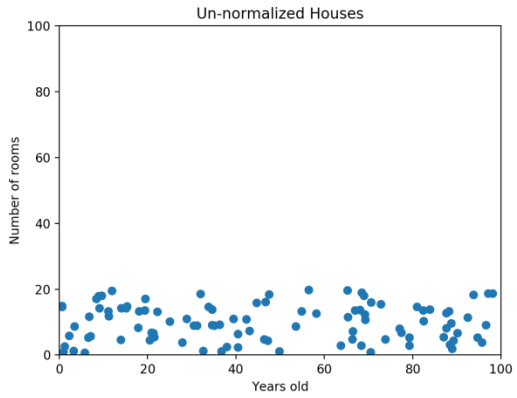
Many machine learning algorithms attempt to find trends in the data by comparing features of data points. However, there is an issue when the features are on drastically different scales.

House ID	Number of Rooms	Age of House (years)
H001	2	1
H002	5	12
H002	2	42
…	…	…
H990	20	100

For example, consider a dataset of houses. Two potential features might be the number of rooms in the house, and the total age of the house in years.

A machine learning algorithm could try to predict which house would be best for you.

However, when the algorithm compares data points, the feature with the larger scale will completely dominate the other.

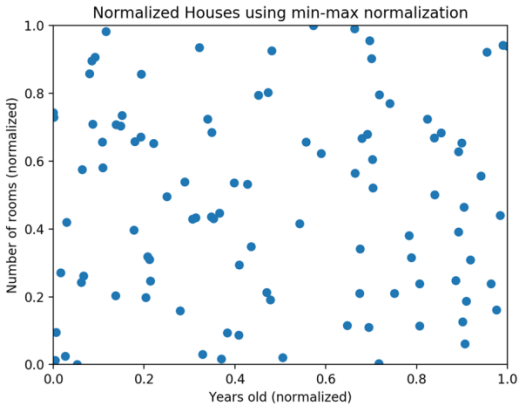


For example, if we use a scatter plot to visualize our data by the age of the house in years and number of rooms, notice that it has a certain pattern.

When the data looks squished like that, **we know we have a problem**

If we have a dataset like this, the machine learning algorithm should realize that there is a huge difference between a house with 2 rooms and a house with 20 rooms.

But right now, because two houses can be 100 years apart, the difference in the number of rooms contributes less to the overall difference.



The goal of normalization is to make every datapoint have the same scale so each feature is equally important. The image above shows the same house data normalized using min-max normalization.

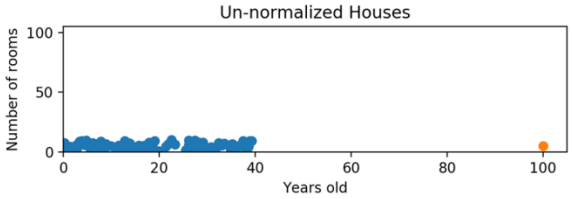
Min-Max Normalization

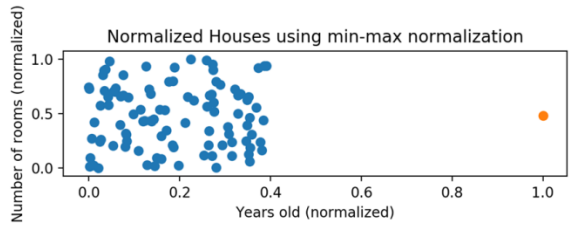
- Min-max normalization is one of the most common ways to normalize data.
- For every feature, the minimum value of that feature gets transformed into a 0, the maximum value gets transformed into a 1, and every other value gets transformed into a decimal between 0 and 1.

$$x_{norm} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

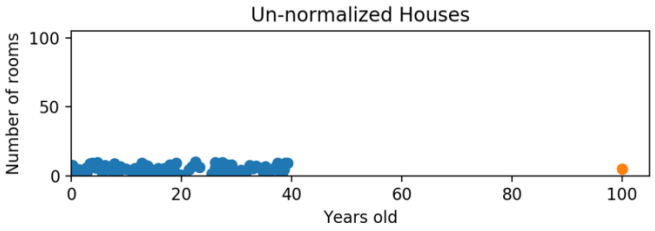
Where:

- x_{norm} is the normalized form of x
- x is a feature vector from the original data set
- $\min(x)$ and $\max(x)$ is the minimum and maximum value of x respectively.

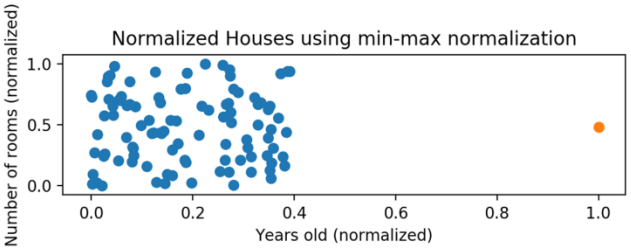




Min-max normalization has one fairly significant downside. it does not handle outliers very well



For example, if you have 99 values between 0 and 40, and one value is 100…



Then the 99 values will all be transformed to a value between 0 and 0.4.

Z-Score Normalization

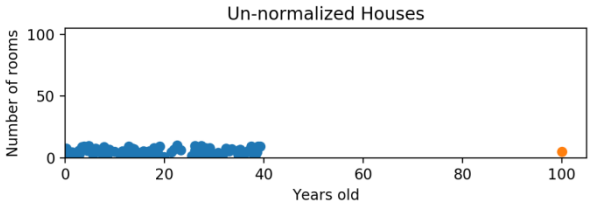
- Z-score normalization is a strategy of normalizing data that avoids the outlier issue.
- The formula for Z-score normalization is given by:

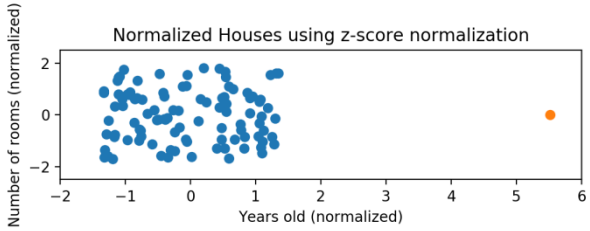
$$Z = \frac{x - mean}{sd}$$

$$Z = \frac{x - mean}{sd}$$

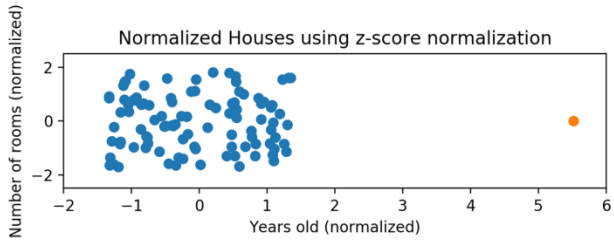
Where:

- Z is the normalized form of x
- x is the feature vector from the original data set
- mean is the mean of the feature vector x
- sd is the standard deviation of x





While the data still looks squished, notice that the points are now on roughly the same scale for both features

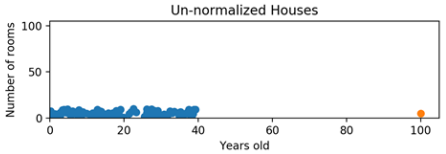


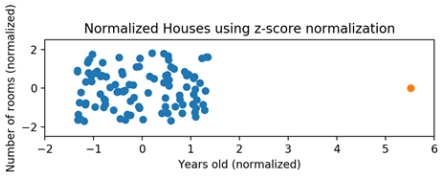
Almost all points are between -2 and 2 on both the x-axis and y-axis.

The only potential downside is that the features aren't on the exact same scale.

With min-max normalization, we were guaranteed to reshape both of our features to be between 0 and 1.

Using z-score normalization, the x-axis now has a range from about -1.5 to 1.5 while the y-axis has a range from about -2 to 2.





This is certainly better than before. the x-axis, which previously had a range of 0 to 40, is no longer dominating the y-axis.

Machine Learning Models that require Scaling

Certain machine learning algorithms such as distance-based algorithms, curve-based algorithms or matrix factorization, decomposition or dimensionality reduction or gradient descent-based algorithms are sensitive towards feature scaling.

And there are certain tree-based algorithms which are insensitive towards feature scaling as they are rule based algorithms such as Classification and Regression trees, Random Forests or Gradient Boosted decision Trees.

Algorithm	Feature Scaling
Linear Regression	YES
Logistic Regression	YES
KNN	YES
SVM	YES
Neural Networks	YES
Kmeans Clustering	YES
Random Forest	NO
Gradient Boosted Decision Trees	NO
Naïve Bayes	NO
PCA	YES
SVD	YES

Disadvantages of Feature Scaling

You will lose the original value will transforming to other values. So, there is loss of interpretation of the values.

Summary

- Normalizing your data is an essential part of machine learning.
- You might have an amazing dataset with many great features, but if you forget to normalize, one of those features might completely dominate the others. It's like you're throwing away almost all of your information! Normalizing solves this problem.
- Having features on same scale that can contribute equally to the result. It can enhance the performance of machine learning algorithms.
- If you don't scale features then large scale variables will dominate the small scale features.
- **Min-max normalization**: Guarantees all features will have the exact same scale but does not handle outliers well.
- **Z-score normalization**: Handles outliers, but does not produce normalized data with the exact same scale.