Feature Engineering – Feature Scaling

By Binoy Patra

**Feature Scaling** is a preprocessing technique used to adjust the range of numerical features in a dataset, so they fall within a specific scale, often to make them comparable and improve model performance.

**Why is Feature Scaling Important?**

**1. Ensures Fairness**

Suppose we have two features:

* **Age**: Ranges from 18 to 100.
* **Income**: Ranges from 10,000 to 1,00,000.

If we don’t scale the features, the **Income** feature (larger range) will dominate models like Linear Regression or Logistic Regression because its numerical values are much higher than those of the **Age** feature. This will make the model prioritize **Income** over **Age**, which might not be ideal.

**Example (Without Scaling):**  
The model could interpret a change of +1 in **Age** (e.g., 25 → 26) as much less important than a +1 in **Income** (e.g., 50,000 → 50,001).

**2. Model Convergence**

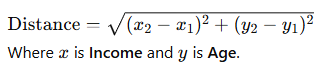
Gradient Descent-based models (e.g., Logistic Regression, Neural Networks) rely on minimizing a loss function.  
If one feature has values between 0 and 1 and another between 1,000 and 10,000, the gradient updates will vary significantly, slowing convergence.

**Example:**  
Imagine climbing a hill where one side is smooth, but the other side is bumpy and steep. we’ll struggle to move smoothly toward the peak. Similarly, the optimization algorithm struggles when feature scales are inconsistent.

**3. Improves Accuracy for Distance-Based Algorithms**

Distance-based algorithms like KNN and SVM rely on calculating the distance between points.

**Example (Without Scaling):**  
If we calculate the Euclidean distance between two data points:



If **Income** is on a much larger scale, the distance will primarily be influenced by differences in **Income**, ignoring **Age**.

**4. Consistency**

Features on vastly different scales can confuse the model during training, leading to inconsistent results.

**Example (With Scaling):**  
After scaling, both **Age** and **Income** will have a mean of 0 and a standard deviation of 1 (if using StandardScaler). This ensures the model treats all features equally, leading to more reliable predictions.

**When is Feature Scaling Needed?**

* **Required** for algorithms that rely on distances or gradients (e.g., KNN, SVM, Logistic Regression, Neural Networks).
* **Not Required** for tree-based algorithms like Decision Trees, Random Forests, and XGBoost (they are scale-invariant).

**Types of Feature Scaling Techniques**

**1. Standardization (Z-Score Normalization)**

Standardization is the process of rescaling the features so that they have a mean of 0 and a standard deviation of 1. Scales the data to have a mean of 0 and a standard deviation of 1.This technique is commonly used when the data follows a Gaussian (normal) distribution.

**Formula:**



Where:

* X is the feature value.
* μ is the mean of the feature.
* σ is the standard deviation of the feature.

**When to Use:**

* When the data is normally distributed (or close to it).
* For algorithms like Logistic Regression, KNN, SVM, and Neural Networks that are sensitive to the scale of the data.

**Pros:**

* Preserves the distribution of the data.
* Useful for many machine learning algorithms.
* Works well when features have different units.

**Cons:**

* Sensitive to outliers; can distort the data if outliers are present.

**2. Min-Max Scaling (Normalization)**  
Min-Max Scaling scales the data to a fixed range, typically [0, 1], based on the minimum and maximum values of each feature.

**Formula:**

A black and white math equation

Description automatically generated

Where:

* X is the feature value.
* min(X) is the minimum value of the feature.
* max(X) is the maximum value of the feature.

**When to Use:**

* When we want to scale the features to a specific range.
* For algorithms like KNN, Neural Networks, and Gradient Descent where the scale affects the distance metric.

**Pros:**

* Simple and easy to interpret.
* Scales the data into a predefined range.

**Cons:**

* Sensitive to outliers (outliers can distort the range).

**3. Robust Scaling**  
Robust Scaling uses the median and the interquartile range (IQR) for scaling, which makes it less sensitive to outliers compared to Standardization and Min-Max Scaling.

**Formula:**

A black text on a white background

Description automatically generated

Where:

* Median(X) is the median of the feature.
* IQR(X) is the interquartile range of the feature.

**When to Use:**

* When the dataset contains outliers that we do not want to distort the scaling.
* Works well for algorithms that are robust to outliers (e.g., Tree-based algorithms).

**Pros:**

* Not sensitive to outliers.
* Works well with skewed data.

**Cons:**

* Can be less effective if the data is normally distributed.

**4. Log Transformation**  
Log Transformation applies the logarithmic function to each feature, which compresses large values and helps to normalize data that is positively skewed.

**Formula:**

A black text on a white background

Description automatically generated

Where:

* X is the feature value.
* The "+1" ensures there are no issues with zero or negative values.

**When to Use:**

* When the data is highly skewed (e.g., income, population).
* When we want to reduce the effect of large values in the dataset.

**Pros:**

* Reduces skewness and helps with non-linear relationships.
* Can handle data with exponential growth patterns.

**Cons:**

* Only works for positive values (log transformation is undefined for zero or negative values).
* Can distort the interpretation of original values.

**5. Power Transformation (Box-Cox)**  
Power Transformation is used to stabilize variance and make data more Gaussian. Box-Cox is a family of transformations that includes log, square root, and others, based on a parameter λ\lambda.

**Formula (Box-Cox):**

A black text with a line between them

Description automatically generated with medium confidence

If λ= 0, the transformation is equivalent to the log transformation.

**When to Use:**

* When data is skewed or has non-constant variance.
* Used when we want to transform the data to approximate a normal distribution.

**Pros:**

* Stabilizes variance.
* Useful for linear models that assume normality.

**Cons:**

* Requires the data to be strictly positive (cannot be used with zero or negative values).
* The optimal value of λ\lambda may need to be found using cross-validation.

**How to Choose the Best Scaling Technique**

Choosing the right scaling technique depends on the nature of our data and the type of algorithm we’re using. Here are key considerations:

* **Standardization** is ideal for models that assume data is normally distributed or when we need to standardize the data for comparison (e.g., linear regression, logistic regression).
* **Min-Max Scaling** is preferable when we need to scale the features to a fixed range (e.g., for neural networks).
* **Robust Scaling** is effective when our dataset contains many outliers, as it minimizes their impact on the scaling process.
* **Log Transformation** is best used when our data is heavily skewed and we want to reduce this skewness to improve the model's learning.
* **Power Transformation (Box-Cox)** works well when we need to transform data to approximate a normal distribution.

Selecting the right scaling technique is crucial for the performance of machine learning models. It depends on:

* The distribution of the data.
* The algorithm being used.
* The presence of outliers.

We should test multiple scaling techniques, evaluate model performance, and use visualizations (Histogram, Boxplot, Kde Plot) to identify the best approach for the dataset.

**Pros and Cons of Feature Scaling**

**Pros**:

* Ensures faster convergence in optimization.
* Prevents bias towards features with larger values.
* Improves accuracy for distance-based models.

**Cons**:

* Adds preprocessing complexity.
* Can distort data if applied incorrectly (e.g., before train-test splitting).