Categorical Encoding

This means that categorical data must be encoded into numbers before we can use it to fit and evaluate a model. There are many ways to encode categorical variables for modeling as follows.

Types of Categorical Data

21 Nominal Data

- Categories have no inherent order
- Examples:
  - Color: Red, Blue, Green
  - City: Delhi, Mumbai, Chennai 2.2 Ordinal Da

ta

- Categories have a meaningful order

- Examples:
  - Size: Small < Medium < Large
  - Rating: Poor < Average < Good < Excellent

Common Categorical Encoding Techniques

1.Label Encoding

Label Encoding assigns each category a unique integer. I is simple and memory-efficient but may unintentionally imply an order among categories when none exists.

- Used in tree-based models like Decision Trees or XGBoost.

2.One-Hot Encoding

One-Hot Encoding converts categories into binary columns with each column representing one category. It prevents false ordering but can lead to high dimensionality if there are many unique values.

- Used in linear models, logistic regression and neural networks.

Color
Red
Green
Blue
Green

Red	Green	Blue
1	0	0
0	1	0
0	0	1
0	1	0

3. Ordinal Encoding

Ordinal Encoding maps categories to integers while preserving their natural order. This works well for ordered data like ratings but is not suitable for nominal variables.

- Used for ordered features like ratings or education levels.

Original Encoding	Ordinal Encoding
Poor	1
Good	2
Very Good	3
Excellent	4

4.Binary Encoding

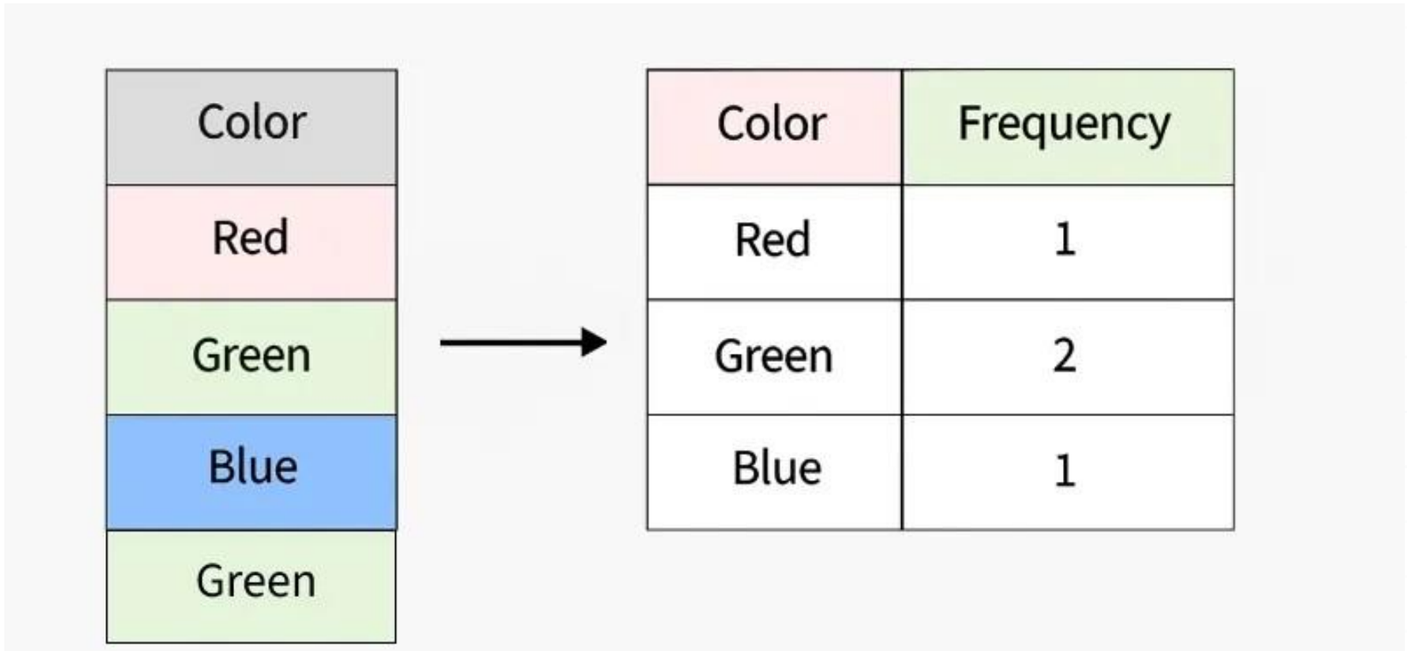
Binary encoding represents categories as binary codes and splits them across multiple columns. It is efficient for high-cardinality data but slightly more complex to implement.

- Applied in high-cardinality text/NLP tasks to save memory.

Frequency Encoding

FrequencyEncoding assigns categories values based on how often they occur in the dataset. It is simple and compact but can introduce data leakage if applied improperly.

- Effective in retail, e-commerce or clickstream data for popularity trends.



Data Transformation

It refers to putting the values in the same range or same scale so that no variable is dominated by the other. Most of the time, the collected data set contains features highly varying in magnitudes, units, and ranges. If scaling is not done then the algorithm only takes magnitude into account and not units hence incorrect modeling. To solve this issue, we have to do scaling to bring all the variables to the same level of magnitude.

Standardization

Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

Normalization

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

Data normalization is a preprocessing method that resizes the range of feature values to a specific scale, usually between 0 and 1. It is a feature scaling technique used to transform data into a standard range. Normalization ensures that features with different

scales or units contribute equally to the model and improves the performance of many machine learning algorithms.

Key Features of Normalization:

- Maps the minimum and maximum of a feature to a defined range
- Preserves the relative relationships of the original data
- Useful for algorithms that rely on distance metrics such as k-Nearest Neighbors and clustering

Min-Max Scaling

Min-max scaling is very often simply called ‘ normalization.’ It transforms features to a specified range, typically between 0 and 1. The formula for min-max scaling is:

$$X_{normalized} = X - X_{min} / X_{max} - X_{min}$$

Where X is a random feature value that is to be normalized. X_{min} is the minimum feature value in the dataset, and X_{max} is the maximum feature value.

- When X is minimum value, the numerator is zero ($X_{min} - X_{min}$) and hence, the normalized value is 0
- When X is maximum value, the numerator is equal to the denominator ($X_{max} - X_{min}$) and hence, the normalized value is 1
- When X is neither minimum or maximum, the normalized value is between 0 and 1. This is referred to as min-max scaling technique

Min-max scaling is a good choice when:

- The approximate upper and lower bounds of the dataset is known, and the dataset has few or no outliers
- When the data distribution is unknown or non-Gaussian, and the data is approximately uniformly distributed across the range
- When maintaining the distribution’ s original shape is essential

Z-score normalization (standardization)

Z-score normalization (standardization) assumes a Gaussian (bell curve) distribution of the data and transforms features to have a mean (μ) of 0 and a standard deviation (σ) of 1. The formula for standardization is:

$$X_{standardized} = X - \mu / \sigma$$

This technique is particularly useful when dealing with algorithms that assume normally distributed data, such as many linear models. Unlike the min-max scaling technique, feature values are not restricted to a specific range in the standardization technique. This normalization technique basically represents features in terms of the number of standard deviations that lie away from the mean.

Before we delve into other data transformation techniques, let's perform a comparison of normalization (min-max scaling) and standardization.

Normalization	Standardization
Objective is to bring the values of a feature within a specific range, often between 0 and 1	Objective is to transform the values of a feature to have a mean of 0 and a standard deviation of 1
Sensitive to outliers and the range of the data	Less sensitive to outliers due to the use of the mean and standard deviation
Useful when maintaining the original range is essential	Effective when algorithms assume a standard normal distribution
No assumption about the distribution of data is made	Assumes a normal distribution or close approximation
Suitable for algorithms where the absolute values and their relations are important (e.g., k-nearest neighbors, neural networks)	Particularly useful for algorithms that assume normally distributed data, such as linear regression and support vector machines
Maintains the interpretability of the original values within the specified range	Alters the original values, making interpretation more challenging due to the shift in scale and units

Can lead to faster convergence, especially in algorithms that rely on gradient descent	Also contributes to faster convergence, particularly in algorithms sensitive to the scale of input features
Use cases: Image processing, neural networks, algorithms sensitive to feature scales	Use cases: Linear regression, support vector machines, algorithms assuming normal distribution