How to Easily Understand and Implement Label Encoding vs. One-Hot Encoding

Authored by

stats writer

November 28, 2025

Label Encoding is a fundamental technique for transforming non-numerical, or categorical variables, into a numerical format suitable for algorithms. It achieves this by assigning a unique integer value to each distinct category within a feature. In contrast, One Hot Encoding (OHE) represents data in a sparse, binary format. OHE creates a new column for every unique category and uses binary indicators (1 or 0) to denote the presence or absence of that category for each row.

While both methods serve the crucial purpose of preparing data for machine learning models, they differ significantly in their output and implications. One Hot Encoding provides superior information fidelity because it treats each category as independent, thereby avoiding the introduction of artificial ordering or magnitude. Label Encoding, by assigning sequential integers, inadvertently suggests a hierarchical relationship between categories, a limitation that must be carefully managed depending on the nature of the variable being encoded.

The Necessity of Encoding in Machine Learning

In the realm of machine learning, the vast majority of mathematical models require input data to be strictly numerical. Algorithms such as linear regression, decision trees, or neural networks rely on calculations and distance metrics, which cannot be performed directly on text or nominal labels. Therefore, when preparing a dataset, converting non-numeric features--specifically categorical variables--into a usable numerical format is a mandatory preprocessing step.

Categorical data can be generally split into two types: nominal (categories with no inherent order, like 'Color' or 'Country') and ordinal (categories with a meaningful rank, like 'Education Level' or 'T-Shirt Size'). The choice between the two primary encoding techniques, Label Encoding and One Hot Encoding, is highly dependent on whether the variable possesses an inherent rank. Failing to choose the appropriate method can lead to misleading model interpretations and reduced predictive accuracy.

We rely on these encoding techniques to bridge the gap between human-readable text labels and the numerical language understood by predictive models. The two most widely used approaches are summarized below:

Label Encoding: This method maps each unique categorical value to a distinct integer. By default, this assignment often follows an alphabetical or dictionary order of the categories.

One Hot Encoding: This approach transforms the categorical column into multiple binary columns, where each new column represents one unique category from the original feature.

To illustrate these methods, consider a simple dataset containing player records with two variables: **Player ID** and **Team**. Our objective is to successfully convert the categorical **Team** variable into a

numeric representation that can be used effectively in a model.

Original Data

Team	Points
A	25
A	12
B	15
B	14
B	19
B	23
C	25
C	29

The following examples provide detailed demonstrations of how both techniques handle the transformation of the **Team** variable.

Understanding Label Encoding

Label Encoding is the simplest form of conversion. It involves iterating through the unique categories in a feature column and assigning them sequential integer values starting from zero. If the categories are "Team A," "Team B," and "Team C," they will typically be assigned 0, 1, and 2, respectively, based on their alphabetical arrangement, though the specific mapping depends on the implementation library.

While straightforward to implement and computationally efficient--as it does not increase the dimensionality of the dataset--it introduces a significant conceptual challenge. By assigning an integer like 2 to "Team C" and 0 to "Team A," the algorithm is tricked into believing that "Team C" is quantitatively 'greater' or 'more important' than "Team A." This inherent ranking is problematic unless the categories truly represent ordinal data.

Example: Applying Label Encoding

When applying **Label Encoding** to the **Team** column in our example dataset, the unique values are mapped to integers based on their alphabetical order, resulting in the following transformation:

Original Data

Team	Points
A	25
A	12
B	15
B	14
B	19
B	23
C	25
C	29

Label Encoded Data

Team	Points
0	25
0	12
1	15
1	14
1	19
1	23
2	25
2	29

Upon transformation, the original categories are converted as follows:

Each original "A" value has been successfully converted to the integer **0**.

Each original "B" value has been converted to the integer **1**.

Each original "C" value has been converted to the integer **2**.

We have successfully converted the **Team** column from a textual, categorical variable into a single numeric feature. However, as noted, the resulting numerical values (0, 1, 2) now impose an unintended mathematical distance that models may misinterpret during calculation-based training.

Understanding One Hot Encoding (OHE)

One Hot Encoding (OHE) is the standard method for handling nominal categorical variables, where no inherent order exists. Instead of assigning a single integer, OHE expands the feature set by creating new binary columns, often referred to as dummy variables. For a feature with N unique categories, OHE creates N new columns in the dataset.

In this method, each observation receives a value of 1 in the column corresponding to its category and 0 in all other new columns. This structure ensures that the distances between categories are equal, effectively eliminating the ordinal ranking issue associated with Label Encoding. For example, in a Euclidean space, the distance between Team A (1, 0, 0) and Team B (0, 1, 0) is mathematically identical to the distance between Team B and Team C (0, 0, 1).

OHE is crucial for models like linear regression or K-Nearest Neighbors that rely on the magnitude of the input features, ensuring that the model attributes equal importance to all categories unless

otherwise indicated by the training outcome.

Example: Applying One Hot Encoding

When applying **One Hot Encoding** to the **Team** column, the original feature is transformed into three distinct binary columns, one for each unique category: **Team_A**, **Team_B**, and **Team_C**.

Original Data

Team	Points
A	25
A	12
B	15
B	14
B	19
B	23
C	25
C	29

One-Hot Encoded Data

Team_A	Team_B	Team_C	Points
1	0	0	25
1	0	0	12
0	1	0	15
0	1	0	14
0	1	0	19
0	1	0	23
0	0	1	25
0	0	1	29

This approach provides clear representation based on presence or absence:

The value in the new **Team_A** column is 1 if the original value in the **Team** column was 'A'. Otherwise, the value is 0.

The value in the new **Team_B** column is 1 if the original value in the **Team** column was 'B'. Otherwise, the value is 0.

The value in the new **Team_C** column is 1 if the original value in the **Team** column was 'C'. Otherwise, the value is 0.

The conversion has successfully transformed the single categorical feature into three numeric dummy variables. A crucial statistical note: when utilizing these binary features in certain models (especially regression), it is necessary to avoid the dummy variable trap by dropping one of the generated columns. The information for the dropped category is perfectly inferred when all remaining category columns are equal to zero.

The Critical Difference: Ranking and Ordinality

The decision between these two encoding methods hinges primarily on the type of categorical data

being handled. For variables that are strictly nominal--meaning they lack any intrinsic order or hierarchy--One Hot Encoding is overwhelmingly the preferred and safest choice in most scenarios, particularly with models sensitive to feature scale like linear models or neural networks.

The critical failure point of Label Encoding becomes apparent when we re-examine the results:

Original Data

Team	Points
A	25
A	12
B	15
B	14
B	19
B	23
C	25
C	29

Label Encoded Data

Team	Points
0	25
0	12
1	15
1	14
1	19
1	23
2	25
2	29

The encoded data implies that Team C (value 2) is somehow mathematically superior to Team B (value 1) and Team A (value 0). If the machine learning model uses these values for distance calculation (as K-Nearest Neighbors or Support Vector Machines might), the results will be severely skewed, reflecting an artificial, linear relationship rather than the true difference between independent categories.

This artificial ranking is only appropriate if the original variable is genuine ordinal data, where a natural ordering already exists (e.g., "Small" < "Medium" < "Large"). In such specific cases, **Label Encoding** can be highly efficient because the numerical assignment directly reinforces the true hierarchy. However, since most machine learning features are nominal, OHE is generally recommended to prevent the introduction of erroneous quantitative bias.

Drawbacks and Trade-offs for Encoding Methods

While One Hot Encoding solves the ordinality problem, it introduces a potential drawback known as the "curse of dimensionality." If a categorical variable contains a large number of unique values (high cardinality), OHE will generate an equally large number of new columns. For example, encoding a "Country" variable with 150 unique entries would create 150 new features.

This drastic increase in the feature space can make models slower to train, more memory-

intensive, and potentially lead to issues like data sparsity. In scenarios involving extremely high cardinality, alternative encoding methods, such as target encoding or feature hashing, are often preferred over standard OHE, although OHE remains the gold standard for low-to-moderate cardinality features.

Conversely, Label Encoding maintains the original dimensionality (it replaces one column with one column), making it highly advantageous in datasets with many high-cardinality nominal features where the performance impact of OHE would be too severe. The critical requirement, however, is that the data scientist must use a model that is inherently robust to the induced ordinality, such as tree-based algorithms (e.g., Random Forests or Gradient Boosting Machines), which handle numerical splits effectively without relying on continuous distance metrics.

Further Resources and Implementation

Depending on the size of your dataset and the specific nature of your variables, you may find one approach more suitable than the other. Understanding the practical application of these methods in popular data science libraries is essential for implementation.

The following tutorials explain how to perform **label encoding** in practice:

The following tutorials explain how to perform **one hot encoding** in practice: