Encoding

Feature Encoding

Feature encoding is the process of converting **categorical variables** (non-numeric data) into a **numerical format** so that machine learning algorithms can understand and process them.

Label Encoding

Label Encoding is a technique used to convert **categorical text data** (like strings) into **numeric labels**, where each unique category is assigned an integer value.

When to Use:

- When the **categorical variable has no ordinal relationship** (but you're using tree-based models like Decision Trees or Random Forests, which can handle these encodings).
- Or when you're using **ordinal categorical data** (like `Low`, `Medium`, `High`) and the order **matters**.

Pros	Cons
Simple and fast	Can introduce unintended ordinal relationships
Suitable for tree-based models	Not ideal for linear models or distance-based models

Integer Encoding

Integer Encoding (also known as **Label Encoding**) is a method of converting **categorical variables** into **integers** so that they can be used in machine learning algorithms that require numerical input.

Use Cases:

Use It When...	Be Cautious If...
Categories are ordinal (have an order)	Categories are nominal (no natural order)

Use It When...	Be Cautious If...
You use tree-based models	You use models sensitive to numerical values (like Linear Regression, KNN, etc.)

One-Hot Encoding

One-Hot Encoding is a technique used to convert **categorical variables** into a **binary matrix representation**—where each category is represented by a vector with a `1` in the column corresponding to that category and `0`s elsewhere.

Why Use It?

- It avoids giving the model a **false sense of order** (which can happen with integer/label encoding).
- It's ideal for **nominal** (unordered) categorical variables.

Drawbacks:

- It **increases the number of features**, especially when the category has many unique values (called the **curse of dimensionality**).
- For high-cardinality columns, it may not be the best choice.

Count Encoding

Count Encoding (also called **Frequency Encoding**) is a technique used to convert **categorical variables** into numerical values by replacing each category with the **number of times it appears in the dataset**.

Advantages:

- Keeps the **dimensionality low** (unlike one-hot encoding).
- Simple to implement and often works well with **tree-based models** like Random Forest, XGBoost, etc.

Disadvantages:

- The model might **misinterpret the encoded values as ordinal** (i.e., assuming higher value = higher importance).
- Can be misleading for **linear models** (like Logistic Regression) where numerical magnitude has strong implications.

Target encoder

Target Encoding (also called **Mean Encoding**) is a **categorical feature encoding technique** where each category is replaced with the **mean of the target variable** for that category.

Pros:

- Keeps information from the target variable.
- Useful for high-cardinality features (many unique categories).

Cons:

- Can **cause overfitting**, especially on small datasets.
- Must be applied carefully — usually with **cross-validation or smoothing** to avoid data leakage.

Common Fix: Smoothing

Instead of using raw means, apply **smoothing**:

$$\text{Encoded Value} = \frac{n \cdot \text{category mean} + k \cdot \text{global mean}}{n + k}$$

Where:

- n = number of samples in that category
- k = smoothing factor (larger = more weight to global mean)

Binary encoder

Binary Encoding is a **categorical feature encoding** technique that combines the benefits of **One-Hot Encoding** and **Hash Encoding**. It is especially useful for handling **high-cardinality categorical variables** (i.e., features with many unique categories).

Pros:

- More compact than one-hot encoding (uses fewer columns).
- Good for **high-cardinality** features.

- Keeps some ordinal relationships from label encoding.

Cons:

- May still introduce noise if the category labels are not meaningful.
- Not as interpretable as one-hot.
