INDUSTRIAL TRAINING DAILY DIARY
DAY 17

15 July, 2025

Topic : Introduction to Label Encoding

What is Label Encoding in Python?

Categorical variables in Python can be transformed into numerical labels using the label encoding technique. It gives each category in a variable a distinct numerical value, enabling machine learning algorithms to interpret and analyze the data effectively. Let's look at a few instances to learn how to label encoding functions.

Examples of Label Encoding in Python
Example 1: Customer Segmentation

Say you have a customer segmentation dataset with data on the demographic traits of your clients. Dataset elements include "Gender," "Age Range," and "Marital Status." You can give several labels to each category inside these variables to execute label encoding. For instance:

Gender	Age Range	Marital Status
Male	25-34	Married
Female	18-24	Single
Male	35-44	Married
Female	45-54	Single
Male	25-34	Single

By applying label encoding to the categorical variables, you can represent the data in a numerical format suitable for customer segmentation analysis.

Example 2: Product Categories

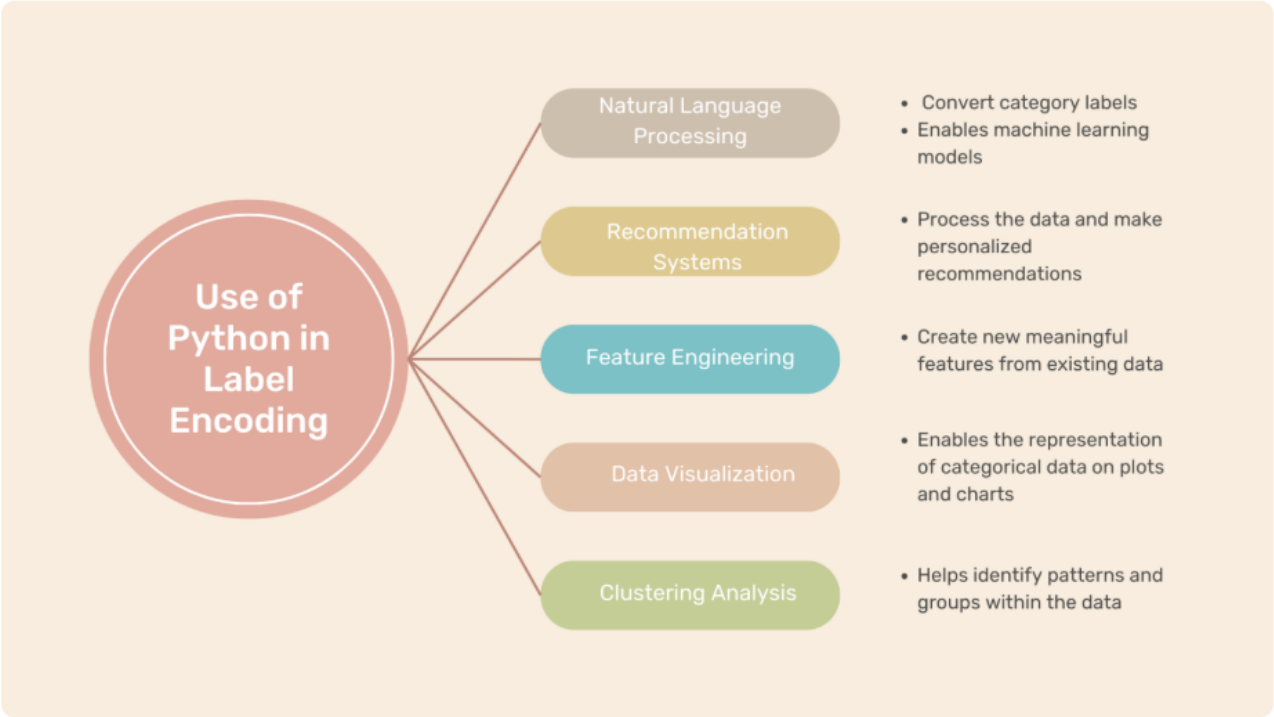
Consider a product categorization dataset that classifies various products into different categories. The dataset contains variables such as “Product Name” and “Category.” To perform label encoding, assign numerical labels to each unique category:

Product	Category
iPhone	1
Samsung TV	2
Adidas Shoes	3
MacBook Pro	4
Nike T-shirt	5

Label encoding allows you to represent the product categories numerically, enabling further analysis or modeling tasks.

Where Can Label Encoding in Python Be Used?

Label encoding can be used in various scenarios when working with categorical data. Here are a few examples:



- **Natural Language Processing (NLP):** Label encoding can convert category labels such as positive, negative, and neutral into numerical representations in NLP applications such as text categorization or sentiment analysis. This enables machine learning models to comprehend and analyze text data successfully.

  •**Recommendation Systems:** Recommendation systems often use categorical variables to represent user preferences or item categories. By label encoding these variables, recommendation algorithms can process the data and make personalized recommendations based on user preferences.

  •**Feature Engineering:** Label encoding can be a crucial step in feature engineering, where we create new meaningful features from existing data. By encoding categorical variables into numerical labels, we can create new features that capture the relationships between different categories, enhancing the predictive power of our models.

- **Data Visualization:** Label encoding can also be used for data visualization purposes. It enables the representation of categorical data on plots and charts that require numerical input. By encoding categorical variables, we can create meaningful visualizations that provide insights into the data.
- **Clustering Analysis:** Label encoding can be utilized in clustering analysis, where categorical variables must be transformed into numerical labels for clustering algorithms to identify patterns and groups within the data.

Preparing the Data for Label Encoding in Python

Before performing label encoding, it is essential to prepare the data appropriately. Here are some common steps in preparing data for label encoding:

Handling Missing Values

Missing values are a common occurrence in datasets. It is essential to address these missing values before performing label encoding. One approach is to remove the rows or columns with missing values if they are insignificant in quantity. Alternatively, you can attribute the missing values using techniques like mean, median, mode, or advanced imputation methods such as regression or multiple imputations.

Dealing with Categorical Features

Identify and isolate the categorical features in your dataset. These are the variables that will be subjected to label encoding. Categorical features are typically represented as text or discrete values and can be nominal or ordinal. Nominal variables have no inherent order or hierarchy, while ordinal variables have a specific order or ranking.

Handling Ordinal Variables

If your data contains ordinal variables, it's crucial to encode it in a method that preserves that order. Using the predetermined order as a guide, one method manually gives numerical labels. Suppose the ordinal variable denotes a person's degree of education (for example, "High School," "Bachelor's Degree," or "Master's Degree"); you may, for example, supply labels like 0, 1, and 2. Alternatively, you can use mapping dictionaries to specify the order and assign numerical labels accordingly.

Dealing with Nominal Variables

For nominal variables, where no inherent order exists, you can use one-hot encoding as an alternative to label encoding. One-hot encoding creates binary columns for each category, representing the presence or absence of a category. This approach is especially useful when dealing with categorical variables with multiple levels or when the absence of a particular category carries some significance.

Data Preprocessing

Apart from handling missing values and encoding categorical features, performing additional data preprocessing steps is often beneficial before label encoding. These steps may include feature scaling, normalization, or outlier removal. Data preprocessing can improve the performance of machine learning models by ensuring that the data is in a suitable format for analysis.

Performing Label Encoding in Python

Using the label encoder in Python class from the sci-kit-learn library, we can conduct label encoding in Python. An instruction manual for doing label encoding is provided below:

Import the necessary libraries:

```
from sklearn.preprocessing import LabelEncoder
```

Create an instance of the LabelEncoder:

label_encoder = LabelEncoder()

Fit the label encoder in Python to the categorical variable:

```
label_encoder.fit(categories)
```

Here, “categories” represents the categorical variable we want to label encode.

Transform the categorical variable into numerical labels:

```
encoded_labels = label_encoder.transform(categories)
```

The “encoded_labels” variable now contains the transformed numerical labels.

Reverse the label encoding in python (optional):

If you need to reverse the label encoding and convert the numerical labels back into their original categorical form, you can use the following:

```
original_categories = label_encoder.inverse_transform(encoded_labels)
```