Feature Engineering in Machine Learning

Ayush Singh[1]

Antern Department of Artificial Intelligence
ayush@antern.co

Abstract. This document contains contents on data preparation in machine learning and we also cover several components of data preparation like feature engineering, feature selection, dimensionality reduction, etc. We provide readers with several traditional and modern techniques to handle complicated data tasks.

Key words: data preparation, feature engineering, feature selection, data cleansing, data transformation, dimensionality reduction

1 Introduction to Categorical Variables

In contrast to continuous numerical values, categorical variables are a sort of data that reflect discrete values or categories. They are frequently employed to represent qualitative data from a dataset, like brand, gender, or colour. Since numerical inputs are frequently required by machine learning algorithms, it is vital to encode categorical variables into numerical values in a coherent and understandable manner.

The two primary categories of categorical variables are:

- Ordinal categorical variables are ones that naturally rank or have an order. The relative ranking of the categories is significant, and the sequence of the categories conveys significant information. A Ph.D. is regarded as having a higher degree of education than a Bachelor's Degree, hence the variable "Education Level," which has the categories "High School," "Bachelor's Degree," "Master's Degree," and "Ph.D.," has an intrinsic order.
- Nominal Categorical Variables: Nominal variables lack any inherent hierarchy or order. The order in which the categories appear is arbitrary; they are essentially different labels. For instance, because there is no intrinsic order between the hues red, blue, and green, the variable "Color" is nominal.

Proper categorical variable encoding is crucial.

Correct categorical variable encoding is essential for a machine learning model to succeed for a number of reasons:

- Compatibility: Several machine learning methods demand numerical inputs, including neural networks, support vector machines, and linear regression. These algorithms can process the dataset's data by encoding categorical variables.

– Interpretability: Correct encoding makes sure that the connections between categories are maintained, making the predictions of the model easier to understand and more significant.
– Performance: Effective encoding methods can aid in capturing the underlying structure in the data, improving the performance of the model.

1.1 Label Encoding

A quick method for transforming category data into numerical values is label encoding. It entails giving each category in the variable a different number. The allocated integers are normally ordered sequentially, beginning with 0 or 1. For ordinal variables, label encoding is especially useful since it may maintain the categories' natural order.

When to Employ Label Encoding and Why:

– Label encoding works best for ordinal variables because the encoded integers can accurately reflect the categories' natural order. Hence, the ordinal relationship between the categories may be captured by machine learning techniques.
– Label encoding can be used for ordinal and nominal variables with some tree-based algorithms, such as decision trees and random forests, because they can handle the encoded values without assuming any hierarchy between the categories.

Consider a dataset with the variable 'Size' representing T-shirt sizes:

Size
Small
Medium
Large
Small
Large

Using label encoding, we can assign a unique integer to each category:

– Small: 0
– Medium: 1
– Large: 2

The encoded dataset will look like this:

Encoded Size
0
1
2
0
2

However it's important to exercise caution when employing label encoding for nominal variables since the encoded integers can provide an erroneous hierarchy that might not accurately reflect the relationships between the categories.
Example:

Let's consider a dataset with information about cars, including a nominal categorical variable 'Color':

Car	Color
A	Red
B	Blue
C	Green
D	Red
E	Green

If we use label encoding for the 'Color' variable, we might assign integers like this:

- Red: 0
- Blue: 1
- Green: 2

The encoded dataset will look like this:

Car	Encoded Color
A	0
B	1
C	2
D	0
E	2

However, this encoding creates an artificial order among the colors: Red < Blue < Green. This order might not reflect any true relationship between the categories and could lead to incorrect assumptions by the machine learning algorithm. In such cases, it's better to use encoding techniques like one-hot encoding or dummy encoding, which do not impose an order on nominal variables.
To perform label encoding in Python, you can use the LabelEncoder class from the scikit-learn library:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Create a sample dataset
data = {'Size': ['Small', 'Medium', 'Large', 'Small', 'Large']}
df = pd.DataFrame(data)
```

```
# Initialize the LabelEncoder
encoder = LabelEncoder()

# Apply label encoding to the 'Size' column
df['Encoded Size'] = encoder.fit_transform(df['Size'])

# Display the encoded dataset
print(df)
```

Output of this program will yield to:

Size	Encoded Size
Small	0
Medium	1
Large	2
Small	0
Large	2

Table 1. Output