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

0.1 Dummy Encoding

To demonstrate the multicollinearity problem with one-hot encoding in linear regression models, let's look at a straightforward example. We have a dataset with details about houses, such as their square footage, the area they are located, and their prices.

Size	Neighborhood	Price
1000	A	200,000
1500	B	250,000
2000	C	300,000
1200	A	220,000
1800	B	280,000

We want to build a linear regression model to predict the house prices based on their size and neighborhood.

First, let's apply one-hot encoding to the 'Neighborhood' column:

Size	A	B	C	Price
1000	1	0	0	200,000
1500	0	1	0	250,000
2000	0	0	1	300,000
1200	1	0	0	220,000
1800	0	1	0	280,000

Now, we build a linear regression model using 'Size', 'A', 'B', and 'C' as independent variables:

$$Price = Intercept + \beta_1 * Size + \beta_2 * A + \beta_3 * B + \beta_4 * C \tag{1}$$

The numbers in columns "A," "B," and "C" added together for every row will always equal 1. This is so because every home must be a part of a specific neighbourhood. The constant term (the intercept) in the linear regression model also denotes a constant value of 1 for each observation. As the constant term and the binary columns are perfectly multicollinear, it is challenging for the model to predict the specific effects of each neighbourhood on the price of a home.

If we use dummy encoding instead, we can remove one neighborhood (e.g., 'A') as the reference category:

Size	B	C	Price
1000	0	0	200,000
1500	1	0	250,000
2000	0	1	300,000
1200	0	0	220,000
1800	1	0	280,000

$$\hat{Price} = \hat{\beta}_0 + \hat{\beta}_1 * Size + \hat{\beta}_2 * B + \hat{\beta}_3 * C \tag{2}$$

In this model, there is no perfect multicollinearity between the binary columns ('B' and 'C') and the constant term. The intercept now represents the baseline price for neighborhood 'A', and the coefficients 2 and 3 represent the price difference between neighborhoods 'B' and 'C' relative to neighborhood 'A'. This prevents multicollinearity issues and allows for more stable and interpretable estimates.

When and Why to Use Dummy Encoding: Dummy encoding is particularly useful when working with linear regression models or other linear models that assume no multicollinearity among independent variables. By omitting one category and using it as a reference, dummy encoding eliminates the linear dependence between the created binary features, ensuring that multicollinearity does not adversely impact the model's estimates.

To perform dummy encoding in Python, you can use the get_dummies function from the pandas library with the drop_first parameter set to True:

```
import pandas as pd

# Create a sample dataset
data = {'Animal': ['Dog', 'Cat', 'Elephant', 'Dog', 'Elephant']}
df = pd.DataFrame(data)

# Apply dummy encoding to the 'Animal' column
encoded_df = pd.get_dummies(df, columns=['Animal'], drop_first=True)
```

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

How to choose reference category or category to be dropped?

$$\begin{tikzcd}
\text{Frequency} \arrow[d] \\
\text{Interpretability} \arrow[d] \\
\text{Domain Knowledge}
\end{tikzcd}$$