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 Mean Encoding:

Mean encoding, also referred to as target encoding, is a method for transforming categorical variables into numerical values by substituting the mean of the target variable for each group in place of each one. As it reduces the dimensionality of the data without significantly reducing the amount of information, this technique can be especially useful when working with categorical variables with high cardinality.

Advantages

- Reduces the dimensionality of the problem, which can enhance model efficiency and lower computational demands.
- Encapsulates in a unique numerical value the relationship between the categorical variable and the target variable.

Disadvantages

- Potential leakage of target information if not implemented correctly, leading to overfitting.
- Not suitable for cases where the relationship between the categorical variable and the target variable is not monotonic.

Worked Example Consider a small dataset containing information about customers, including their age group and their spending score (target variable):

Age Group	Spending Score
Youth	75
Adult	55
Youth	80
Senior	40
Adult	60

To perform target encoding, we replace each 'Age Group' category with the mean spending score for that category:
Youth: (75 + 80) / 2 = 77.5
Adult: (55 + 60) / 2 = 57.5
Senior: 40

The encoded dataset will look like this:

Encoded Age Group	Spending Score
77.5	75
57.5	55
77.5	80
40.0	40
57.5	60

Target encoding must be done independently for each fold during cross-validation in order to prevent target leakage. This stops data from the validation set from entering the encoding process by only using the training data for each fold to compute the mean of the target variable.

Now, because the encoding procedure used data from the entire dataset, including the validation or test set, if we train a model on this encoded dataset and use the same data for validation or testing, we run the risk of overfitting. On this dataset, the algorithm would probably perform well, but it would not generalise well to new, unforeseen data.

We can prevent target leakage and guarantee that the model is trained on data that is independent of the validation or test set by employing cross-validation and carrying out target encoding independently for each fold. The model's ability to generalise to new data is improved through this procedure, which also offers a more precise assessment of the model's performance.

Here's how we do it using CV:

To demonstrate how to avoid target leakage with target encoding, let's use a small dataset and perform k-fold cross-validation. In this example, we'll use a 3-fold cross-validation.

Dataset:

Age Group	Spending Score
Youth	75
Adult	55
Youth	80
Senior	40
Adult	60
Senior	45

We'll first split the dataset into 3 folds:

Age Group	Spending Score
Youth	75
Adult	55

Age Group	Spending Score
Youth	80
Senior	40

Age Group	Spending Score
Adult	60
Senior	45

Now, for each fold, we'll perform target encoding using only the training data for that fold:

Fold 1 (train on Fold 2 and Fold 3, validate on Fold 1):
Training data:

Age Group	Spending Score
Youth	80
Senior	40
Adult	60
Senior	45

Target encoding:

Age Group	Target Encoding
Youth	80
Adult	60
Senior	(40 + 45) / 2 = 42.5

Encoded validation data:

Encoded Age Group	Spending Score
80	75
60	55

Fold 2 (train on Fold 1 and Fold 3, validate on Fold 2):
Training data:

Age Group	Spending Score
Youth	75
Adult	55
Adult	60
Senior	45

Target encoding:

Age Group	Target Encoding
Youth	75
Adult	(55 + 60) / 2 = 57.5
Senior	45

Encoded validation data:

Encoded Age Group	Spending Score
75	80
45	40

Fold 3 (train on Fold 1 and Fold 2, validate on Fold 3):
Training data:

Age Group	Spending Score
Youth	75
Adult	55
Youth	80
Senior	40

Target encoding:

Age Group	Target Encoding
Youth	(75 + 80) / 2 = 77.5
Adult	55
Senior	40

Encoded validation data:

Encoded Age Group	Spending Score
55	60
40	45

By performing target encoding separately for each fold during cross-validation, we ensure that the target variable's mean is calculated only using the training data for each fold, preventing information from the validation set from leaking into the encoding process.

Further Research

As a student, it's important to explore and research various encoding techniques to deepen your understanding of their use cases and implementation. We encourage you to research the following encoding techniques on your own:

- Binary Encoding
- Base-N Encoding
- Hashing

By studying these techniques, you will develop a stronger foundation in categorical variable encoding and be better prepared to select the most appropriate encoding method for your specific machine learning problems.

1 Choosing the right encoding technique

Selecting the appropriate encoding technique for categorical variables is essential for the success of your machine learning models. The choice depends on the problem, dataset, and algorithm. Here are some guidelines and recommendations to help you decide which encoding technique to use:

1. **Ordinal vs. Nominal Variables**: Determine if the categorical variable is ordinal (having a natural order) or nominal (having no order). For ordinal variables, label encoding is a suitable choice, as it preserves the order. For nominal variables, consider one-hot encoding, dummy encoding, or other advanced techniques like binary encoding, base-N encoding, or hashing.
2. **Cardinality**: High cardinality categorical variables (i.e., those with many unique categories) can lead to a large number of columns when using one-hot encoding or dummy encoding. In such cases, consider using binary encoding, base-N encoding, or hashing to reduce dimensionality.
3. **Algorithm Sensitivity**: Some machine learning algorithms, like decision trees and random forests, can handle categorical variables directly, while others, like linear regression and support vector machines, require numerical input. Consider the algorithm's sensitivity to categorical variables when choosing an encoding technique.
4. **Memory and Computational Resources**: If memory and computational resources are limited, consider using encoding techniques like binary encoding, base-N encoding, or hashing that reduce the number of columns compared to one-hot encoding.