The first thing that come up in your mind to solve this problem might be "One-Hot Encoding", a method that turn categorical features into binary variables. But now, I'm gonna tell you that **"STOP USING ONE-HOT ENCODING".**

Why no more one-hot encoding?

We're not gonna talk about what is one-hot encoding here. If you are not familiar with this, please check out some descriptions and sample code at Scikit-learn website.

So why? why we should not use one-hot encoding? Here's the reason:

- *It increases the amount of features*

Increasing the amount of features is not appriciated for most machine learning methods. Imagine you are encoding categorical features like location or name, it usually end up with 1000+ dimensions.

The reason behine the thing is the "Curse of Dimensionality", which make our machine learning model harder to converge and generalize. Also, more features means more training time and memeory!

- *It generates too many zeros*

Too many zero values always confuse machine learning models, especially deep learning. This is because we are optimizing loss function at a relatively flat gradient with many zeros. That is to say that it might be harder to converge, which is what we don't want to see when training.

What else can we do?

So please~please tell me what else can we do to encode categorical features? Hopefully, we got bunch of method to do so!!! Let's take a look at them!

Frequency Encoding

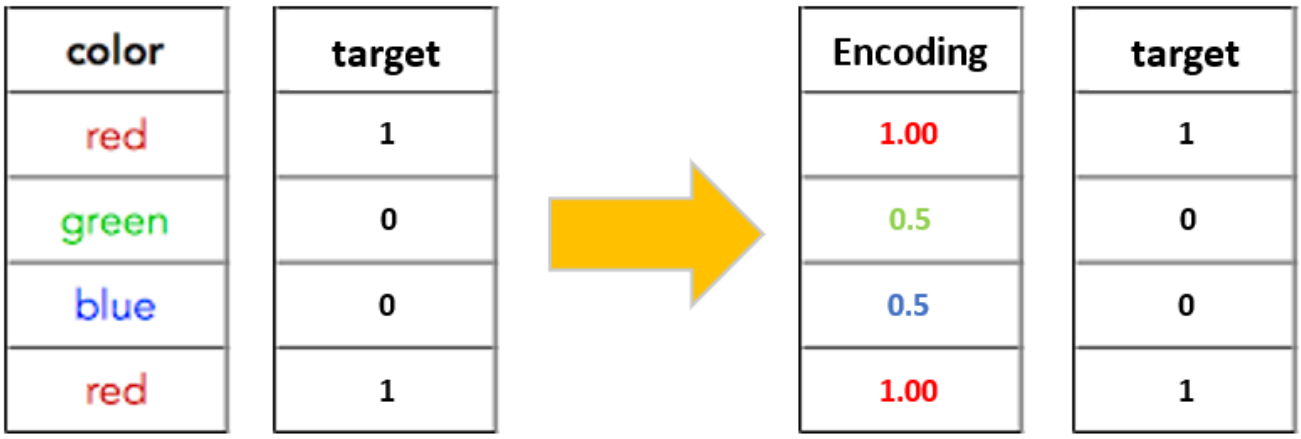
The simplest method is using frequency encoding. Just replace categorical features with their occurrence and we're done! The assumption behind this is that the frequency of a category is informative. So this method make sense for most cases

```
df['cat_col'].value_counts()
```

However, when we got the same frequency for multiple categories, the model might value different categories as the same.

Target Encoding

Also known as “mean encoding”, target encoding transform categorical values to numerical values by taking the average of all the target value for each category. We can see the following image to present this process:



However, when using this method, we need prevent model from overly rely on these encoded values. Also, we need to be careful of outlier because that might greatly influence the mean value.

But don’t worried! Let’s introduce “Categorical Encoders”! This package do all the things for you.

It’s similar to how you do in Scikit-learn package. Just use *.fit()*, *.transform()*,and *.fit_transform()* to transform your features.

```
# use target encoding
te = TargetEncoder(cols=['A', 'B'])
```

What's more?

That's not the end, let's also include some advance encoding methods!!!!

Leave-One-Out Encoding (LOO Encoding)

This is a modified version of target encoding. As we mention earlier, model might be overly rely on those value after target encoded. Fortunately, LOO encoding helps us mitigate this problem by leaving out its own target value when encoding. That is to say, we only include the other target values that is in the same category as the one that we are encoding.

To apply this method, we can also use the package - "Categorical Encoders".

```
import category_encoders as ce

loo_encoder = ce.LeaveOneOutEncoder(cols=[A, B], sigma=0.05)

loo_encoder.fit(X, y)
X_encoded = loo_encoder.transform(X)
```

The hyperparameter, **'sigma'**, in LOO encoder add an noise of normal distribution of **mean=0** and **standerd deviation='sigma'** to make our model more stable. The recommended value of '**sigma**' is between **0.05–0.6.**

Generalized Linearn Mixed Model (GLMM)

Simply, this method use "Generalized Linear Model" to analyize categorical feature. We're not including all the math and detail in this article, but you can just regard this method as applying "linear regression" on target encoding.

The advantage of using GLMM is that we don't need to adjust any hyperparameter, and we can still return a bunch of robust values.

```
glmm_encoder = ce.GLMMEncoder(cols=[A, B], binary_target=True)

# binomial_target = True (for Classification)
# binomial_target = False (for Regression)

glmm_encoder.fit(X, y)
X_encoded = glmm_encoder.transform(X)
```

However, this method is more time-consuming compare with other methods.

The end.

That's the end of this article. Please hit the clap button, leave a comment, and follow me if you're interested about my content!!!

Specially thanks to 林倢愷, his article inspire me to write this article. All credict belongs to him. Please check out his article for more AI information!! I'll list some of his article about this content below.

不要再做One Hot Encoding！！

Categorical feature的正確開啟方式

axk51013.medium.com

Kaggle Categorical Encoding 3大絕招

Leave-One-Out Encoding, Beta Target Encoding 跟 Generalized Linear Mixed Model Encoding簡介

axk51013.medium.com