Handling missing values of categorical variables: Intelligent Randomization

Handling missing values of categorical variables: Intelligent Randomization | by Ketan Suhaas Saichandran | Oct, 2023 | Artificial Intelligence in Plain English (medium.com)

Handling missing values of categorical variables could be tricky and might need a lot of attention while working with. Let me introduce some of the common approaches to solve this problem:

1. Removing rows with missing values: If we have a large number of features in the dataset then using this approach could potentially cause a huge loss of information that could be gained from the other features.
2. Impute with Mode: If the number of missing values is comparable to the number of datapoints in the minority category then using this approach introduces new unwanted bias as we populate the missing values with the majority category.
3. Impute with a new category: Again, why would you want to introduce a new behavior when you know these missing values would've certainly chosen one of the available behaviors?
4. Train a model to predict the category based on other features: Do you really want to work that hard? :)

For example, I was working with patients' data for my research, and encountered a lot of missing categorical variables as below.

```
0.0    19336
1.0     5580
2.0     2058
NaN     1382
```

We clearly know that NaN could've taken one of these values {0,1,2}. For example, if you consider this as a survey, you already know the probability distribution of the values chosen by the people. **It's basically representing the statistical trend of the population. But as a counterargument, is it possible that the people who did not survey could all have chosen 2? Which is the most extreme case. In that case, what is the probability that it happens? We still have to use the known probability distribution to answer this question.**

P(All NaN values are 2) = ((2058)/(2058+5580+19336))^(1382) {It's quite rare}

This is how I came up with an idea to populate the missing values.

For every missing value, choosing a categorical value from {0,1,2} have the probabilities according to the distribution [19336,5580,2058]/ (2058+5580+19336), which basically follows the probabilistic characteristics of the population. **Although this may work well in most cases when the sample size is small, we have to be careful when the sample size is comparable**

to the size of the minority population class. In this case, 1382 is comparable to 2058. But I still prefer this approach over using uniform distribution to populate the missing values in this case.

I implemented it using the following one-liner:

```
df['feature'].apply(lambda x: np.random.choice(df['feature'].value_counts().index, \

p=np.array(df['feature'].value_counts())/np.sum(df['feature'].value_counts())) \

if np.isnan(x) else x)
```

These were the results that I obtained:

```
0.0    20339
1.0     5861
2.0     2156
```

If you're still not satisfied with the intuition of the idea, you can try adding some noise to the probability distribution and observe the results.

In the case of the unknown sample size being comparable to the size of minority class of the population, we could consider a hybrid approach (Inspired by the epsilon-greedy approach of reinforcement learning) defined by:

```
epsilon = probability_intelligence    #lies between 0 and 1

for every missing value:

 choice = uniform_random(0,1)
```

```
if choice<epsilon:

  missing_value = intelligent_randomization()  #randomize with the above approach

else:

  missing_value = uniform_randomization()  #randomize uniformly with categorical variables
```

Now, intelligently choosing the value of epsilon could go a long way, which could be a sequel to this article.

Thanks for reading.