FAQs - Feature Engineering and Cross Validation

1. What are imbalanced datasets?

Imbalanced datasets are those where there is a severe skew in the class distribution, such as 1:100 or 1:1000 examples in the minority class to the majority class.

This bias in the training dataset can influence many machines learning algorithms, leading some to ignore the minority class entirely. This is a problem as it is typically the minority class on which predictions are most important.

2. What are the techniques to handle imbalanced datasets?

One approach to addressing the problem of class imbalance is to randomly resample the training dataset. The two main approaches to randomly resampling an imbalanced dataset are to delete examples from the majority class, called **under sampling**, and to duplicate examples from the minority class, called **oversampling**.

3. What is Cross-Validation?

Cross-Validation is a very useful technique for assessing the performance of machine learning models. It helps in knowing how the machine learning model would generalize to an independent data set. You want to use this technique to estimate how accurate the predictions your model will give in practice.

4. How to deal with outliers?

How to treat outliers is a very tricky topic, it mainly depends on the problem statement. Sometimes we can't even remove outliers or treat them because that's just the way data is and should be kept like that for analysis.

Let us consider an example of a bank

In a bank, account balance might have high outliers as few people will have a high amount in their accounts, but should we be treating this value or removing it? So, I think here we have to consider that in practical life situations we cannot remove these accounts and if we replace them with any other value that will make the data biased.

So the best option would be thinking w.r.t. the problem statement. If the problem statement is: Who is more likely to do a credit card default then I think we can remove people with a high balance in their account as they are less likely to default. In another case, if the problem statement is: Who is going to take a home loan then also I think we can remove outliers but if our problem statement is about business loans then we have to consider these outliers having high balance as they might take a loan for their business.

All the above judgments should be made by doing the necessary EDA on the data set to figure out the relationship of balance with the target (credit card default, home loan, business loan, etc.). None of the

conclusions should come without the data supporting them.

Replacing the outliers is done when outliers are few in number and we think that this would be because of the wrong imputation. So, we remove these outliers and treat them as missing data.

5. How K-Fold Cross-validation works?

Here is the thing:

- If k-fold cross-validation is used to optimize the model parameters, the **training set** is split into k parts.
- Training happens k times, each time leaving out a different part of the training set.
- Typically, the error of these k-models is averaged.
- This is done for each of the model parameters to be tested, and the model with the lowest cross-validated and validation error is chosen.
- The test set has not been used so far.
- Only at the very end, the test set is used to test the performance of the (optimized) model.

example: k-fold cross validation for hyperparameter optimization (k=3)

original data split into training and test set:

|---------- train ----------| |--- test ---|

cross-validation: test set is not used, error is calculated from

validation set (k-times) and averaged:

|---- train ----------|- validation -| |--- test ---|

|---- train ---|- validation -|---- train ---| |--- test ---|

|- validation -|---------- train ----------| |--- test ---|

final measure of model performance: model is trained on all training data and the error is calculated from test set:

|---------- train ----------|--- test ---|

- In some cases, k-fold cross-validation is used on the entire data set if no parameter optimization is needed (this is rare, but it happens).
- In this case, there would not be a validation set and the k parts are used as a test set one by one.
- The error of each of these k tests is typically averaged.

example: k-fold cross validation

|----- test -----|---------- train ----------|

|----- train ----|----- test -----|----- train ----|

|---------- train ----------|----- test -----|

6. Should we apply the transformations AFTER splitting the data into train/test/validation to prevent data leaks?

Yes. Data preparation must be fit on the training dataset only. That is, any coefficients or models prepared for the data preparation process must only use rows of data in the training dataset.

Once fit, the data preparation algorithms or models can then be applied to the training dataset, and the test dataset.

1. Split Data.
2. Fit Data Preparation on Training Dataset.
3. Apply Data Preparation to Train and Test Datasets.
4. Evaluate Models.

7. How to install imblearn library?

To install imblearn you can use

Please run the below code in your jupyter notebook and install imblearn as shown below not in Anaconda prompt

```
pip install -U imbalanced-learn
```

Also, please check the below link

https://pypi.org/project/imbalanced-learn/

8. How logarithmic transformation convert skewed distribution into normal distribution?

Log transformation does not convert any data (any random variable) to normal distribution. Log transformation, in general, reduces the amount of fluctuation in a set of data. Taking logs "pulls in" more extreme values on the right (high values) relative to the median, while values at the far left (low values) tend to get stretched back, further away from the median.

The logarithmic distribution is a discrete distribution. The normal distribution is a continuous distribution.

9. How to choose K value in cross validation?

The key configuration parameter for k-fold cross-validation is k that defines the number folds in which to split a given dataset.

One approach to answering this question is to perform a sensitivity analysis for different k values. That is, evaluate the performance of the same model on the same dataset with different values of k and see how they compare.

The expectation is that low values of k will result in a noisy estimate of model performance and large values of k will result in a less noisy estimate of model performance.

10. I am getting this error while trying to import SMOTE

ValueError: Input contains NaN, infinity or a value too large for dtype('float32')

How to fix it?

In this case the problem is that many scikit functions return numpy arrays, which are without or lacking of pandas index. So, there is an index mismatch when you used those numpy arrays to build new DataFrames. Please try to mix them with the original data.

Or there could be many other reasons as well. Did you treat null values or NaN's? You might be having any such column with nulls or empty rows. To rectify this please check your data for NaN vales and replace them with suitable value.

If your data has NaN values, please refer the below link to remove or impute them

https://numpy.org/doc/stable/reference/generated/numpy.nan_to_num.html

Or try this,

```
X = X.values.astype(np.float)
y = y.values.astype(np.float)
```

11. Why do we need to do one-hot encoding after splitting the data even though one-hot encoding doesn't lead to data leakage?

We might do one hot encoding after splitting data if we have missing values in categorical columns as we can't do one-hot encoding for variables with missing values

And missing value imputation leads to data leakage, so should be done after splitting the data.

12. How can we tackle the mismatch of the size of data due to different categories after creating dummies for train and test sets.

The feature mismatch issue happens when the features/columns in the test set are different from the train set after creating the dummies. We can align the features using the following code -

train, test = train.align(test, join="outer", axis=1, fill_value=0)

This will create a dummy column for the category that is not present in any of the sets.