**The right way of using SMOTE with Cross-validation**

This article discusses the right way to use SMOTE to avoid inaccurate evaluation metrics while using cross-validation.

[KSV Muralidhar](https://towardsdatascience.com/author/ksvmuralidhar/" \t "_self)

Mar 29, 2021

7 min read

Share

Image by [Mitchell Luo](https://unsplash.com/@mitchel3uo?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com/?utm_source=medium&utm_medium=referral)

This article assumes the reader to have a working knowledge of SMOTE, an oversampling technique to handle imbalanced class problem. We’ll discuss the right way to use SMOTE to avoid inaccurate evaluation metrics while using cross-validation techniques. First, we’ll look at the method which may result in an inaccurate cross-validation metric. We’ll use the breast cancer dataset from Scikit-Learn whose classes are slightly imbalanced.

Method 1Method 1

In the above code snippet, we’ve split the breast cancer data into training and test sets. Then we’ve oversampled the training examples using SMOTE and used the oversampled data to train the logistic regression model. We computed the cross-validation score and the test score on the test set. The method followed above is not a right way to use SMOTE or any undersampling/oversampling technique from the ‘[Imblearn](https://towardsdatascience.com/tag/imblearn/" \o "Imblearn)’ package. This method may result in an inaccurate cross-validation score, which may be quite different from the test score or score on unseen data. Let’s look at the right way to use SMOTE while using cross-validation.

Method 2Method 2

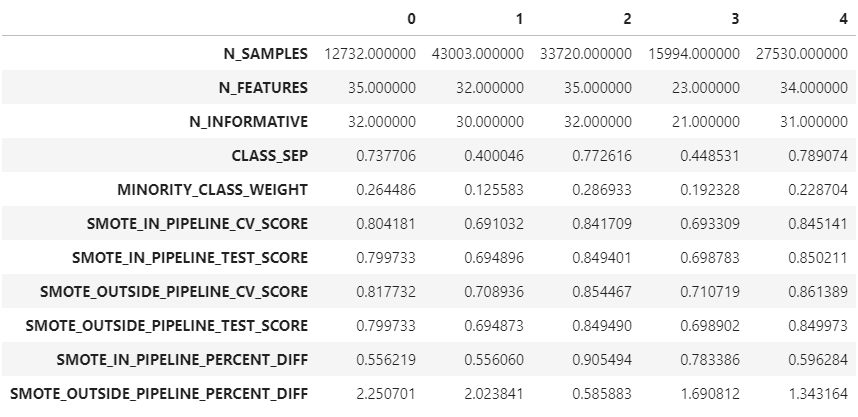
In the above code snippet, we’ve used [Smote](https://towardsdatascience.com/tag/smote/" \o "Smote) as a part of a pipeline. This pipeline is not a ‘Scikit-Learn’ pipeline, but ‘imblearn’ pipeline. Since, SMOTE doesn’t have a ‘fit\_transform’ method, we cannot use it with ‘Scikit-Learn’ pipeline.

From the results of the above two methods, we aren’t able to see a major difference between the cross-validation scores of the two methods. However, the second method resulted in a cross-validation score that is very slightly closer to the test score compared to the first method. This might have happened just by chance and also may be because the dataset is not highly imbalanced.

Now we’ll run an experiment by generating 500 synthetic datasets with class imbalance using the ‘make\_classification’ method of ‘Scikit-Learn’ package. This experiment is conducted to evaluate if the above observations are just by chance. Since, we are using Logistic Regression, we’ll ensure that there is no multi-collinearity in the datasets generated.

In the above code snippet, we’ve defined a function named ‘model’ that takes the input features (X), classes (y) and a boolean value ‘smote’ (indicates whether SMOTE should be a part of pipeline) as arguments. We first split the data into training and test sets, then based on the ‘smote’ argument we include SMOTE in the pipeline (SMOTE will be a part of the pipeline if ‘smote’ is True) and calculate the cross-validation and test scores.

In the above code snippet, we generated synthetic dataset one at a time, passed it to the ‘model’ function, computed the cross-validation and test scores with and without including SMOTE in the pipeline and stored the results in a data frame. Below, we can see a snapshot of the transposed output data frame.

Transposed output data frame

N\_SAMPLES:

Number of examples in the dataset.

N\_FEATURES:

Number of input features.

N\_INFORMATIVE:

Number of features carrying information.

CLASS\_SEP:

Magnitude of class separation. The higher the magnitude, the simpler would be the classification problem.

MINORITY\_CLASS\_WEIGHT:

% of minority class samples. 0.26 means the minority class (1) forms 26% of the dataset.

SMOTE\_IN\_PIPELINE\_CV\_SCORE:

CV score when SMOTE is included in the pipeline. This is the right way of using SMOTE.

SMOTE\_IN\_PIPELINE\_TEST\_SCORE:

Test score or score on unseen data when SMOTE is included in the pipeline. This is the right way of using SMOTE.

SMOTE\_OUTSIDE\_PIPELINE\_CV\_SCORE:

CV score when SMOTE is not included in the pipeline. This is the wrong way of using SMOTE.

SMOTE\_OUTSIDE\_PIPELINE\_TEST\_SCORE:

Test score or score on unseen data when SMOTE is not included in the pipeline. This is the wrong way of using SMOTE.

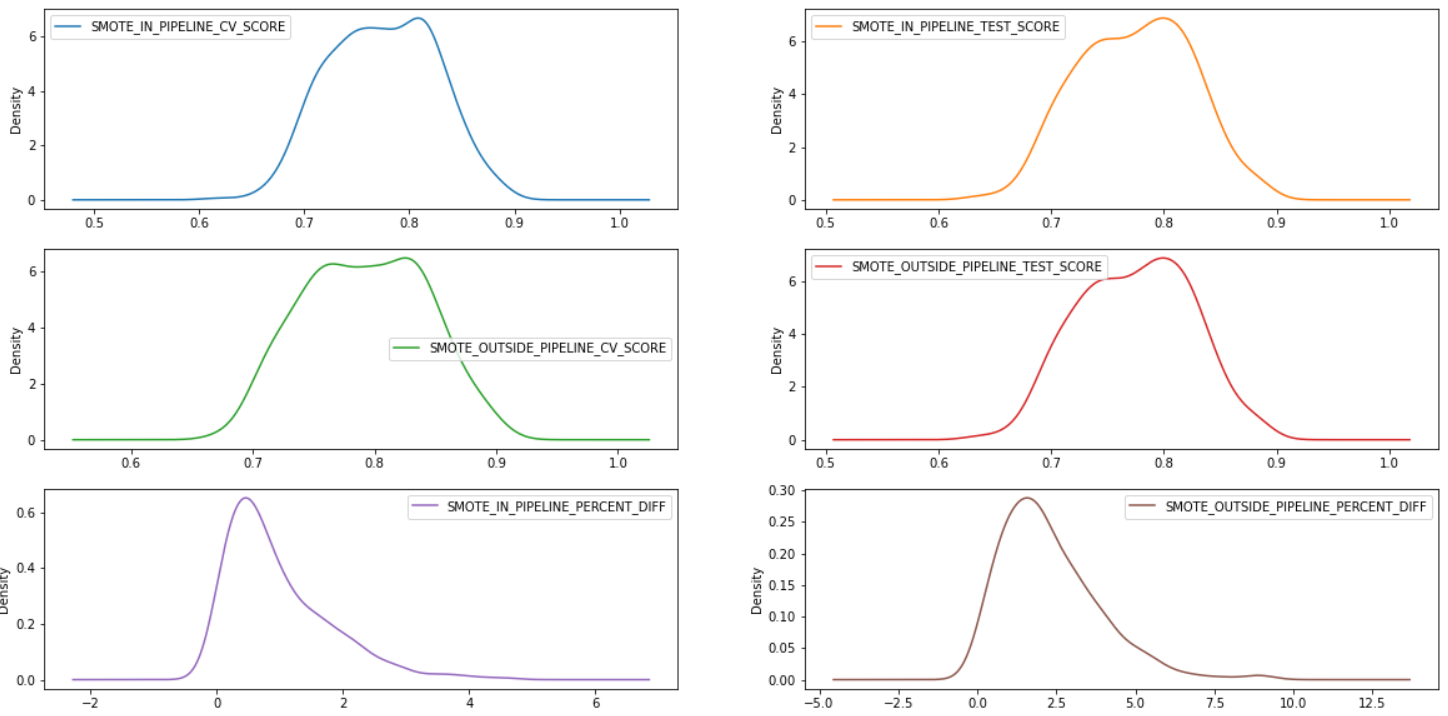
SMOTE\_IN\_PIPELINE\_PERCENT\_DIFF:

The difference between the cross-validation and test score when SMOTE is included in the pipeline. This is the right way of using SMOTE.

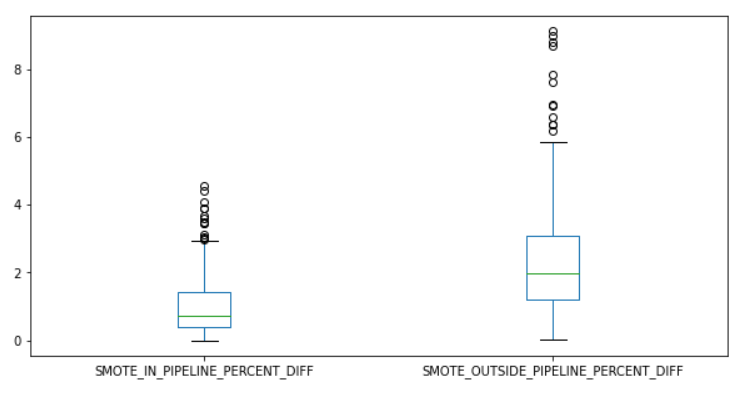
SMOTE\_OUTSIDE\_PIPELINE\_PERCENT\_DIFF:

The difference between the cross-validation and test score when SMOTE is not included in the pipeline. This is the wrong way of using SMOTE.

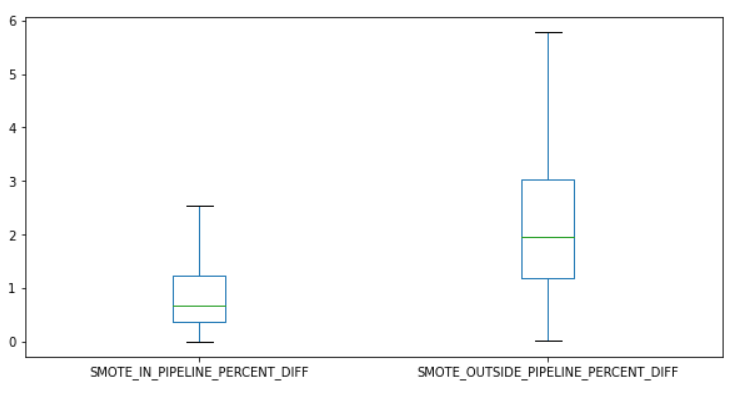
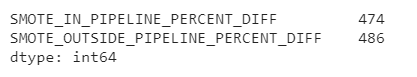
From the above few records of the data frame, it can be seen that in majority of the cases ‘SMOTE\_IN\_PIPELINE\_PERCENT\_DIFF’ is very low compared to ‘SMOTE\_OUTSIDE\_PIPELINE\_PERCENT\_DIFF’. We’ll take the help of statistics to ensure that these observations are not by chance.

Image by author

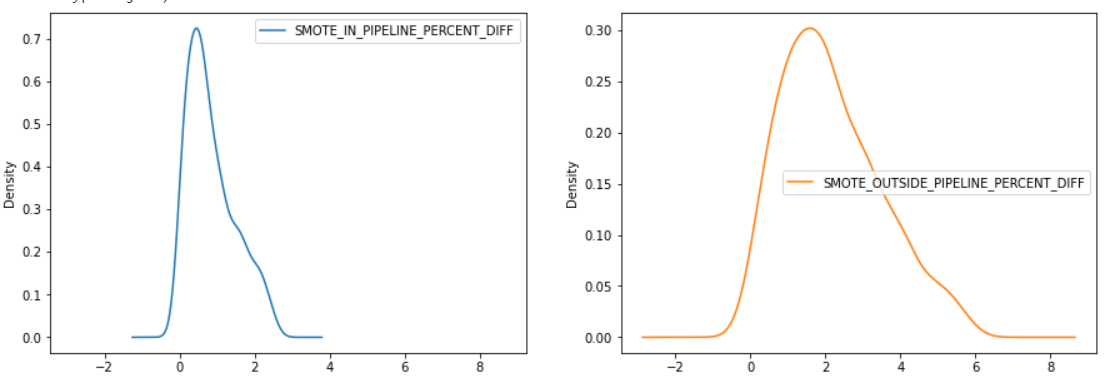
The above density plots show that the percentage differences are skewed. This may be due to datasets with highly imbalanced classes and the model (Logistic Regression) was unable to fit the data properly. Now let’s look for outliers in the percentage difference columns.

Outliers in the percentage difference columns

We can see that the percentage difference in the cross-validation and test scores when SMOTE is included in the pipeline (‘SMOTE\_IN\_PIPELINE\_PERCENT\_DIFF’) is low when compared to the ones where SMOTE is outside the pipeline (‘SMOTE\_OUTSIDE\_PIPELINE\_PERCENT\_DIFF’). Now we’ll remove the outliers.

Percentage difference after outlier removalRecords left in percentage difference after removing outliers

The outliers are fully removed after fitting and transforming the ‘OutlierRemover’ for four times. We can see that only a small percentage of data is lost as a part of the outlier removal process.

Density plots of percentage differences after removing outliers

From the above density plots, we can see that the percentage difference columns look fairly Gaussian. We’ll calculate the confidence interval (with a 95% confidence level) of the mean percentage differences and ensure that the observations are not by chance.

Image by authorImage by author

**Findings**

This experiment is conducted to ensure that the results from the breast cancer dataset are not by chance. We can see that the percentage difference between the cross-validation and test scores is high when SMOTE is outside the pipeline. This shows that including SMOTE in the pipeline results in a more accurate cross-validation score. This experiment shows that the observations from the breast cancer dataset are just by chance.

**Limitations**

There are a few limitations of the experiment conducted. Since, the above experiment is conducted in a controlled environment, the results may not be used to estimate the population parameters. The values in the confidence interval may be accurate only when

1. The chosen dataset has features within the ranges used to conduct the experiment (i.e. number of samples must be between 10,000 and 50,000 and so on). For example, you cannot estimate the parameters of the whole population just by surveying ‘females’ in the age range ’20–30′.
2. The model used is Logistic Regression. Effect of other algorithms on the outcomes isn’t evaluated in this experiment. Hence, we cannot generalize these observations to all the algorithms. For example, a random forest algorithm may not be affected much (as the Logistic Regression algorithm did) by the position of SMOTE in the pipeline. The experiment must be conducted with multiple algorithms to rule out the impact of an algorithm on the outcome.

**Suggestions for Future Research**

1. A similar experiment can be conducted by increasing the sample size from the current 500 and using more diverse datasets.
2. Evaluation of the effect of various algorithms on the outcomes can be done to ensure that the observations don’t vary from one algorithm to another.
3. The correlation coefficient between ‘MINORITY\_CLASS\_WEIGHT’ and ‘SMOTE\_OUTSIDE\_PIPELINE\_PERCENT\_DIFF’ is found to be -0.45, while that of ‘MINORITY\_CLASS\_WEIGHT’ and ‘SMOTE\_IN\_PIPELINE\_PERCENT\_DIFF’ is -0.2. This shows that as the size of minority class decreases, the percentage difference between the cross-validation and test score when SMOTE is outside the pipeline increases. A similar experiment can be performed by varying the ‘MINORITY\_CLASS\_WEIGHT’ and controlling all other parameters of the synthetic dataset. This enables us to directly find the effect of minority class weight on the percentage difference between cross-validation and test scores while using oversampling/undersampling techniques along with a cross-validation technique.

Know more about my work at <https://ksvmuralidhar.in/>