Group-disjunct and stratified data partitioning for machine learning

Uwe Reichel, audEERING GmbH, Gilching, Berlin
ureichel@audeering.com

1 Introduction

A common task in partitioning a machine learning dataset is to obtain a split that at the same time

1. is **disjunct** with respect to a selected grouping variable like *speaker ID*, and
2. yields an optimal **stratification** for a target variable (e.g. *emotion class*) as well as for one or more further grouping variables (e.g. *gender*, *mother tongue*)

This article introduces the Python package `splituils` developed at *audEERING GmbH* that offers one general solution how to achieve these two goals for categorical variables. Furthermore, this package provides several possibilities to bin numeric variables so that they can be used for disjunct group-splitting and stratification. The package is available in the PyPI software repository:

```
$ pip install splitutils
```

To download the source code, please visit the GitHub page [2].

In the following two sections we will describe `splitutils`' approaches for split optimization and for binning of one or many numeric variables.

1.1 Split optimization

Finding the optimal split is simply done by brute force optimization. We create k group-disjunct splits and select the split among all candidates that minimizes some score. For creating group-disjunct splits we use the `GroupShuffleSplit()` class from `scikit-learn` [1].

1.1.1 Two partitons: training and test

Given a set of stratification variables S, we calculate for each variable $V \in S$ its reference probability distribution over the entire data. Probabilities are calculated simply in terms of maximum likelihood estimates.

For each split i into training and test partition suggested by `GroupShuffleSplit()` we then measure for each V the distance of its probability distribution in the test set from its reference distribution in terms of the Jensen-Shannon divergence (JSD). The higher the JSD, the lower the stratification quality. Finally, we take the weighted mean of the JSD values over all stratification variables. Weights are user-defined.

In addition to the weighted mean JSD, we calculate for each split i the absolute difference of the proportion of groups that go into the test set (say 20% of the speakers) and the proportion

of test set samples (20% of all observations). Ideally, these two proportions would be equal. However, they do not necessarily match when we have different numbers of observations per stratification variable level.

Taken together we search for the split $\hat{\imath}$ into training and test set, that minimizes the following score:

$$\hat{\imath} \quad = \quad \underset{i \in 0 \ldots k}{\arg\min} \left[\frac{\sum_{V \in S} \left[w_V \cdot \mathrm{JSD}(P_{V,i} || R_V) \right] + w_d \cdot d_i}{\sum_{V \in S} [w_V] + w_d} \right]$$

where:

- S: set of categorical stratification variables V
- w_V: user-assigned weight for variable V
- $\mathrm{JSD}(P_{V,i}||R_V)$: Jensen-Shannon divergence between $P_{V,i}$, the test set probability distribution for split i, and R_V, the reference distribution for V
- d_i: absolute difference between the proportion of test set samples and the proportion of groups that go into the test set for split i, i.e. $d_i = |\frac{n_{i,\mathrm{test}}}{n} - \frac{g_{i,\mathrm{test}}}{g}|$, where n is the size of the entire data and $n_{i,\mathrm{test}}$ the size of the test set for split i; g is the total number of different grouping variable values and $g_{i,\mathrm{test}}$ is the number of different grouping variable values in the test set for split i.
- w_d: user-assigned weight of d_i

1.1.2 Three partitons: training, development, and test

As for the split into 2 partitions, we calculate for each stratification variable $V \in S$ its reference probability distribution over the entire data. Probabilities are again calculated in terms of maximum likelihood estimates.

For each split i into training, development, and test set suggested by the application of `GroupShuffleSplit()` in a nested loop, we measure for each V the maximum distance of its reference distribution to the test and development set in terms of the Jensen-Shannon divergence (JSD). The higher the maximum JSD, the lower the stratification quality. Finally, we take the weighted mean of the maximum JSD values over all stratification variables. Weights are user-defined.

In addition to the weighted mean JSD, we calculate for each split the absolute difference of the proportion of groups that go into the test set and the proportion of test set samples, as well as the proportion of development groups and development set samples. Ideally, for each partition the group and sample proportions would be equal, thus the higher this distance the worse the partition quality. From these two absolute differences, one per development and test partition, we again keep the maximum. Taken together we search for the split $\hat{\imath}$ into training, development, and test set, that minimizes the following score:

$$\hat{\imath} \quad = \quad \underset{i \in 0 \ldots k}{\arg\min} \left[\frac{\sum_{V \in S} \left[w_V \cdot \max\left(\mathrm{JSD}(P_{V,i} || R_V), \mathrm{JSD}(Q_{V,i} || R_V)\right) + w_d \cdot \max\left(d_i, e_i\right) \right]}{\sum_{V \in S} [w_V] + w_d} \right]$$

where:

- S: set of categorical stratification variables V
- w_V: user-assigned weight for variable V
- $\max\big(\text{JSD}(P_{V,i}||R_V), \text{JSD}(Q_{V,i}||R_V)\big)$: Maximum of the Jensen-Shannon divergence between the reference distribution R_V of V and its test set $P_{V,i}$ and development set $Q_{V,i}$ distribution for split i
- $\max\big(d_i, e_i\big)$: maximum of d_i, the absolute difference between the proportion of test set samples and the proportion of groups that go into the test set for split i, and e_i, the absolute proportion difference for the development set
- w_d: user-assigned weight for the maximum absolute proportion difference

1.2 Numeric variable treatment

Numeric target or other stratification variables need to be binned in order to use them for stratification. With `splitutils` variables can be binned in isolation or in combination as outlined in the following two paragraphs.

Binning of single variables For single variables binning can be done either intrinsically into n classes based on an equidistant percentile split, or extrinsically by means of a list of user defined lower boundary values.

Joint binning of multiple variables Multiple variables can be intrinsically binned into a single categorical variable with n levels by KMeans clustering for which we use the `KMeans` class of `scikit-learn` [1]. Before clustering all variables are centered and scaled with [1] 's `StandardScaler()` class. We run 'KMeans()' with `k-means++` cluster initialization, the Lloyd algorithm, a maximum iteration number of 300 and a tolerance of .0001.

Please find several minimal examples for partitioning and binning on the GiHub project page [2].

References

[1] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay: *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12:2825–2830, 2011.

[2] Reichel, U.: *splitutils*. GitHub project page, 2023. `https://github.com/reichelu/splitutils`.