CSCD01

Assignment 3

March 15th, 2021

**Table of Contents:**

Bugs examined

Bug #1

Description

Modifications and Design

Bug #2

Bug selected

Decision to select bug

Implementation

Documentation and Testing

**Bugs Examined:**

**Bug #1:** “Missing features removal with SimpleImputer #16426”

<https://github.com/scikit-learn/scikit-learn/issues/16426>

Description:

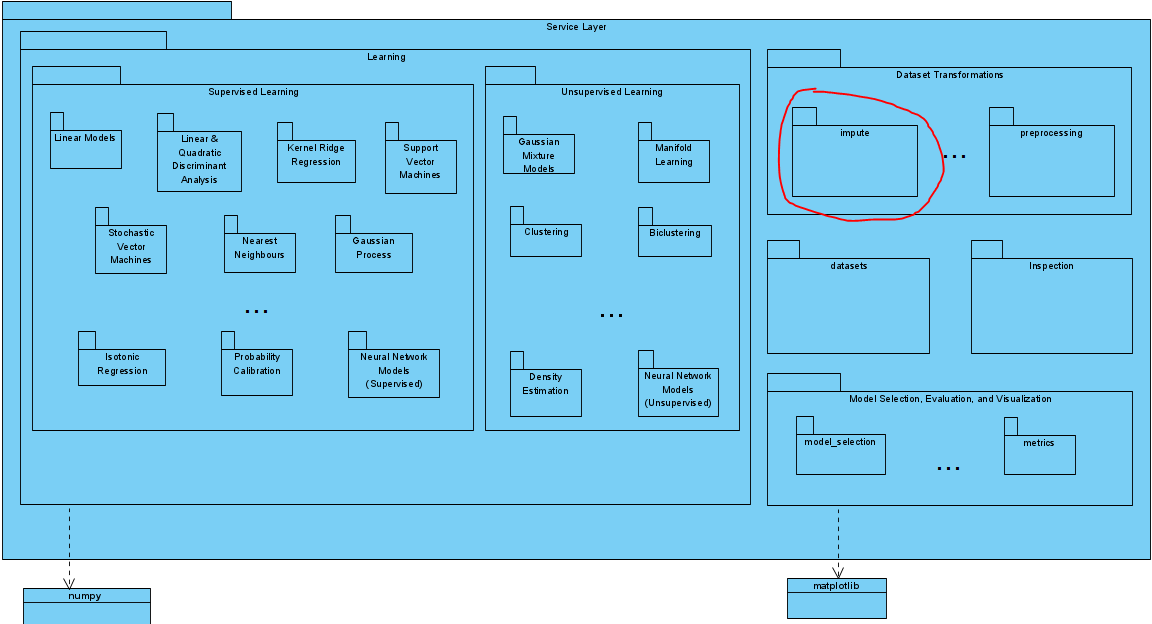
The first issue selected is a mix between a bug and also a potential feature/enhancement of the imputer module. Some background information to consider is the role of imputation. Imputation may be performed when a set of data has missing values (e.g. when say, a respondent fails to fully complete a survey). The intention is to fill in these missing values instead of deleting chunks of data, avoiding problems such as introducing biases.

The issue of the imputer module occurs when the imputer is fitted with data (in the form of an array, representing a matrix) containing a column full of missing values (all NaN). Upon using the transform function on subsequent data inputs, the output will now have certain columns (corresponding to the column of missing values) removed. As an example: an imputer fitted with an array whose elements in the first column are all NaN, upon imputing another set of data, will now have an output with the first column discarded. The user has no control over this behaviour, and is not given any way to know what data is dropped, nor is there any option to retrieve dropped data.

From a statistics standpoint, dropping data can arguably be an acceptable outcome. However, from a design standpoint, the imputation module is only expected to fill in missing values, and not expected to change the shape of the input in any way; this would be considered the “bug” portion. The enhancement portion to the module, is for the user to be able to decide whether dropping the column corresponding to NaNs is appropriate. It is also requested that there be a way to access information on which columns are dropped.

Modifications:

Our last assignment gave the perspective of the library’s architecture as multiple layers, with the Service Layer being the point of interaction between the user and the library. The impute module that is affected lies here, as part of the dataset transformations package.

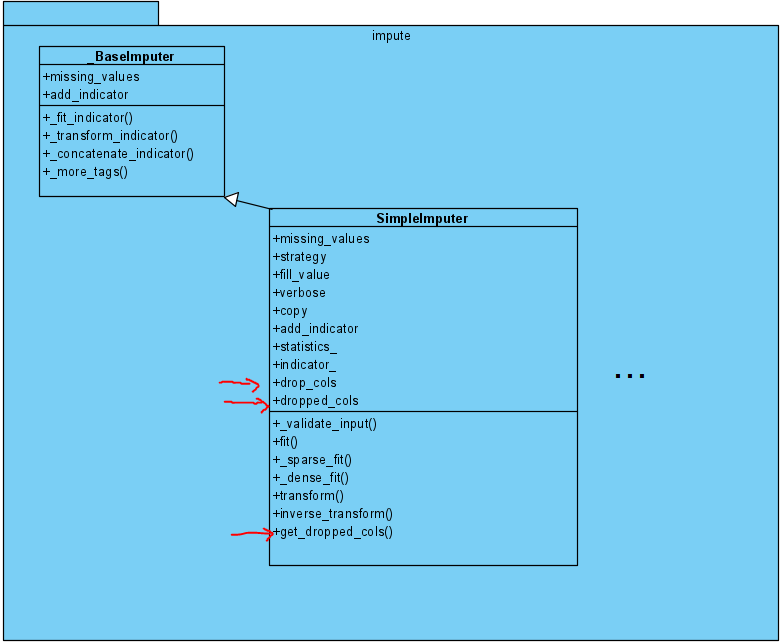


The issue affects the SimpleImputer class, which is under the path “sklearn/impute/\_base.py”.

The first modification that will be made is the number of available parameters SimpleImputer accepts; the parameter “drop\_columns” will be added, which allows the user to decide whether or not columns corresponding to the model’s NaN columns will be dropped. The default value will be set to “True”, as the default behaviour of SimpleImputer is already implemented to drop data and setting the default to drop columns will be more consistent with other imputation classes. The corresponding code that drops the columns will now depend on the new parameter.

The second modification would be the addition of an internal variable “dropped\_columns” to SimpleImputer that keeps track of dropped columns during the last transformation, as well as a function to retrieve such information “get\_dropped\_columns()”. This was a requested feature within the comment chain of the issue. Implementing this feature ensures that the module remains flexible, as it allows users to decide whether they want to drop columns, while still having access to data that is not imputed during the transformation call.

The changes are all local to the SimpleImputer class:



**Bug #2:** Add most\_frequent drop method to one hot encoder.

<https://github.com/scikit-learn/scikit-learn/issues/18553>

This feature request is to add a most\_frequent drop method to one hot encoder. This will drop the most frequent category in each feature. For example, we use a one hot encoder to encode the colour feature with categories green, red and yellow. If red is the most common category, then we drop it from the encoder. We can do this because if yellow and green are 0 for the colour feature then we know the feature is red, so no information is lost.

Potential solution:

1. Copy the current implementation for dropping a category but alter it so it checks which category is most frequent for that feature. Once the code checks which category is most frequent it will drop that one. Repeat this for each feature.

**Example:**

|  |  |  |  |
| --- | --- | --- | --- |
| ID # | GREEN | YELLOW | RED |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 |
| 4 | 0 | 0 | 1 |
| 5 | 1 | 0 | 0 |
| 6 | 0 | 1 | 0 |

Since red is the most common category, we drop it. So, after our data will look like this:

|  |  |  |
| --- | --- | --- |
| ID # | GREEN | YELLOW |
| 1 | 0 | 0 |
| 2 | 0 | 1 |
| 3 | 0 | 0 |
| 4 | 0 | 0 |
| 5 | 1 | 0 |
| 6 | 0 | 1 |

The changes are all local to the OneHotEncoder class:

