CSCD01

Assignment 3

March 15th, 2021

Table of Contents:

Bugs examined

Bug #1 …… 3

Description …… 3

Modifications and Design …… 4

Bug #2 …….. 6

Bug selected …….. 8

Decision to select bug …… 8

Documentation ……..…….. 9

Testing …….. 10

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.

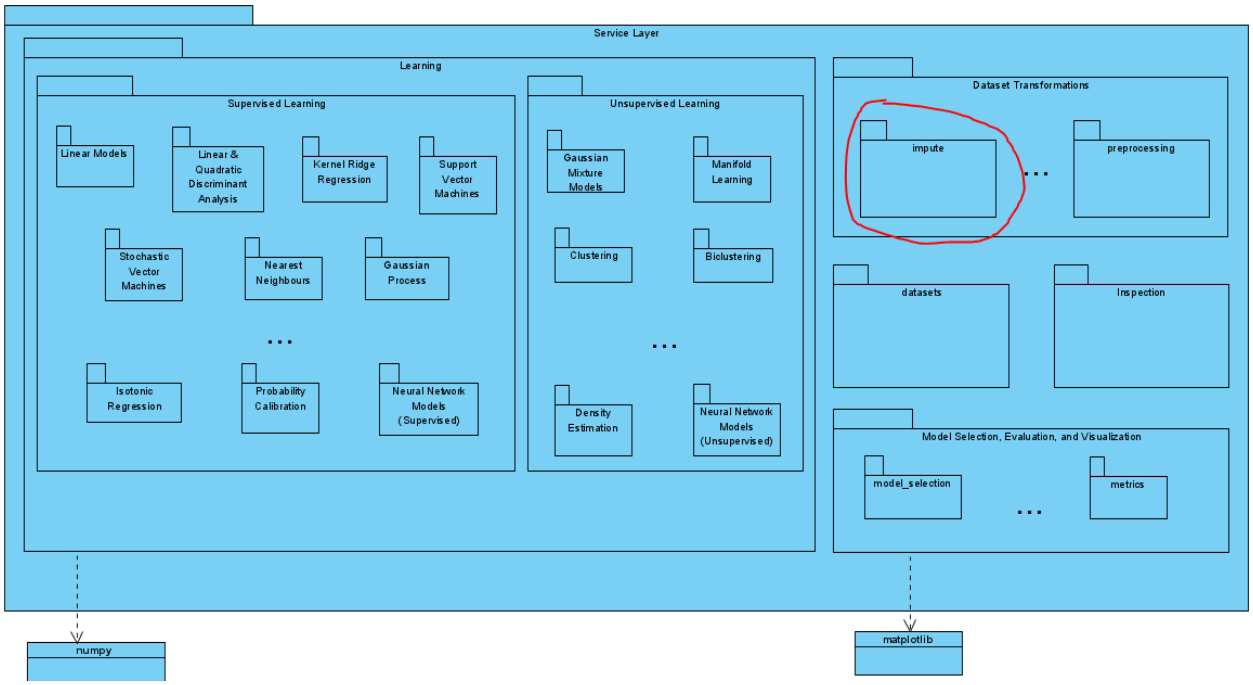
The issue of the imputer module occurs when the imputer is fitted with data (in the form of an array, representing a matrix) containing a column full of missing values (all NaN). Upon using the transform function on subsequent data inputs, the output will now have certain columns (corresponding to the 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.



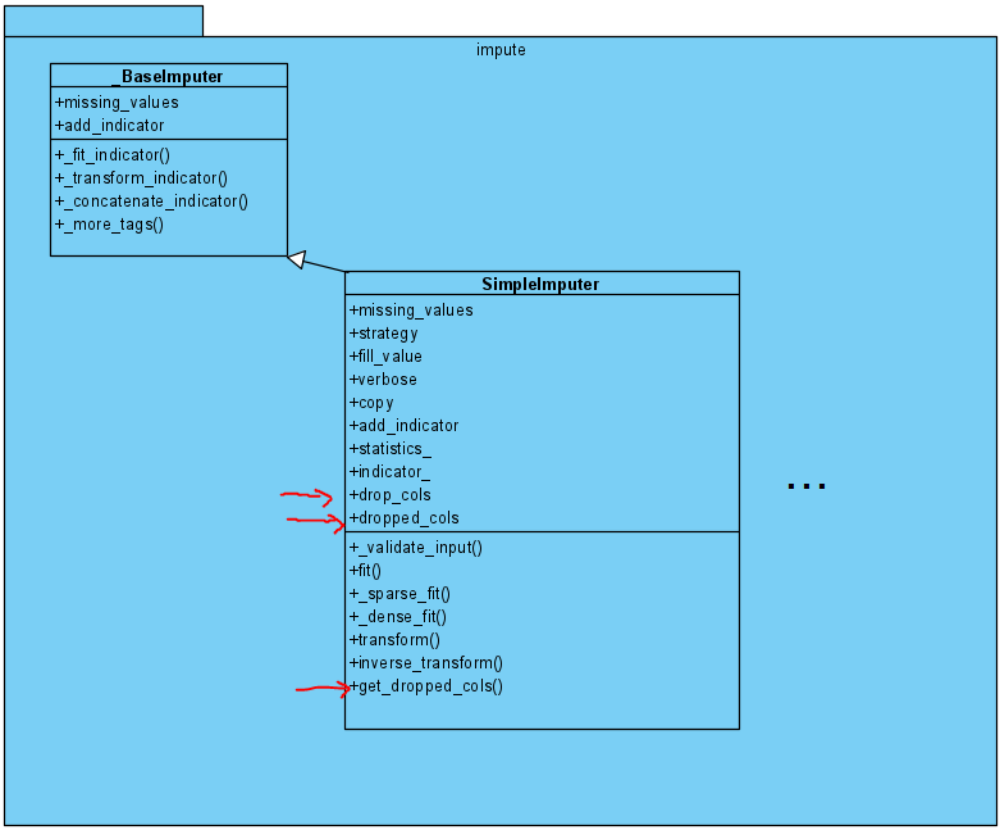
The issue affects the SimpleImputer class, which is under the path “sklearn/impute/_base.py”.

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.

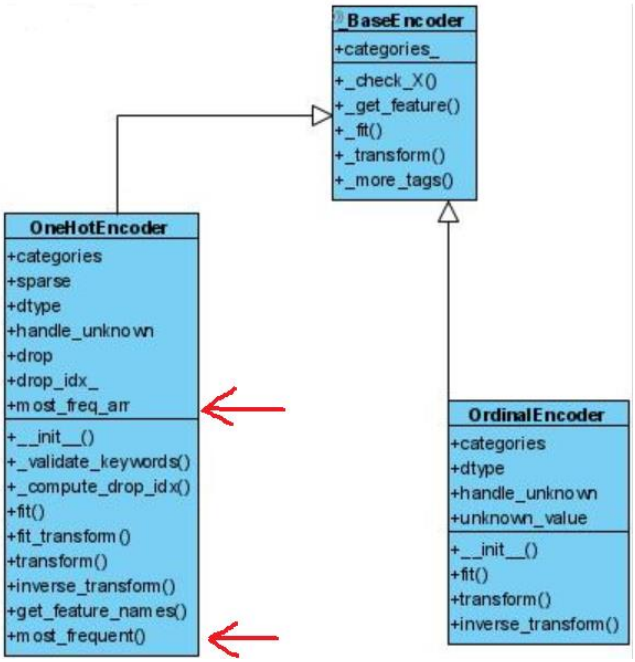
Example:

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:



Bug Selected:

For our bug fix, we selected bug #1 to implement. The choice was mainly made because bug #1 is easier to grasp with our current level of understanding in the field of statistics. With a more solid understanding of the matter at hand, it is more likely that our solution to the bug will be “correct”. Furthermore, the issue seems to pop up frequently, so it seemed to be a good choice for us to select this bug.

Documentation:

The documentation for the bug fix / feature is shown in the module at path “sklearn/impute/_base.py”, under the class SimpleImputer. In summary, there is added description on an additional parameter “drop_columns”:

```
drop_columns : boolean, default=False
    If True, then columns which only contained missing values at :meth: `fit` are
    discarded upon :meth: `transform`, if strategy is not "constant"
```

Additionally, there are docstrings for the two new methods “get_invalid_columns()” and “get_invalid_columns_indicies()”:

```
564     def get_invalid_columns(self):
565         """Returns columns that were not imputed during the last imputation from
566         calling :meth: `transform`.
567 
568         Parameters
569         ----------
570         None
571 
572         Returns
573         -------
574         self._invalid_columns : ndarray of shape (n_samples, n_features)
575             The original values of X that were not imputed during the last
576             transformation. If no transformations have been made yet, then
577             returns an empty array.
578         """
```

```
584     def get_invalid_columns_indicies(self):
585         """Returns the indicies of columns that were not imputed during the last
586         imputation from calling :meth: `transform`.
587 
588         Parameters
589         ----------
590         None
591 
592         Returns
593         -------
594         self._invalid_statistics_indexes : ndarray of shape (, n_features)
595             The indicies of the columns of X that were not imputed during the
596             last transformation. If no transformations have been made yet, then
597             returns an empty array.
598         """
```

Testing:

The test cases were all done under the path "sklearn/impute/tests/test_impute.py". In total, 10 test cases exist, many of which can be considered both "customer acceptance" tests and unit tests. For example, "test_base_simple_imputer_keep_invalid_columns()" can be considered a "customer acceptance" test because the test case uses it as a user of the module would use it; first fit a SimpleImputer with a model, before transforming another set of data. However, it is also a unit test, because it also confirms that our "drop_columns" parameter is working as intended. Aside from such test cases, there are some cases that can be categorized as unit tests only; for example, "test_base_simple_imputer_store_deleted_columns()" tests that the module correctly stores invalid columns during transformations, which is something that end users will not be accessing.

In essence, we can generalize all customer acceptance test cases as:

Drop:

1) Create a SimpleImputer with the default "drop_columns" parameter value
2) Fit the SimpleImputer with an ndarray, with at least one column full of np.nan. Suppose it was the first column.
3) Transform a different ndarray of the same shape
4) The result should be an imputed ndarray, missing the first column

Not Drop:

1) Create a SimpleImputer with the parameter "drop_columns" as True

2) Fit the SimpleImputer with an ndarray, with at least one column full of np.nan. Suppose it was the first columns.
3) Transform a different ndarray of the same shape
4) The result should be an ndarray, with all columns (except the first column) imputed. The first column should remain as it was when it was given as an input to the transformation.

The customer acceptance tests for the new function will add a 5th step to the above general tests:

5) Call “get_invalid_columns()” on the SimpleImputer. It should now return the first column of the ndarray.

In summary, our test suite starting from line 1504 under the given path “sklearn/impute/tests/test_impute.py”, in conjunction with the above customer acceptance cases, covers both acceptance and unit tests.