Assignment 2

CSCD01 Winter 2021

Spaghetti Code

Julian Barker, Adam Ah-Chong, Andrew Gao, Jesse Francispillai, Laphonso (Poncie) Reyes, Saad Ali

Table of Contents

Introduction

First, we prepared a list of bugs/features to choose from the scikit-learn remote repository, with each team member adding 1 or 2 to the list. After, 5 of those existing bugs/features were chosen to implement over the course of the project as a team. We designed a development cycle with respect to agile methodologies by issuing a sprint plan. The first sprint plan consisted of selecting 2 out of the 5 bugs/features to divide into subtasks: documentation, implementation, and testing. These sub-tasks were assigned to team members and incorporated into our team JIRA and sprint plan.

Our first sprint was one week in length starting from March 1st and ending on March 7th. This consisted of three working days and included 6 estimation points assigned to our team, such that each estimation point represents 1 hour of work. As a team, we made the decision to select issues based on our TA's advice. Such issues reflected a shorter sprint duration compared to planned future sprints (next assignments). This was due to the nature of the "Easy" labeled issues we were tasked with selecting in Step 1. The project velocity went as expected and no issues regarding the development process during this sprint needed to be brought over to our next sprint.

Step 1

The following is the list of bugs and features we have selected:

1. **Bug**: Sequential forward selection - unsupervised fit_transform bug
   https://github.com/scikit-learn/scikit-learn/issues/19538
2. **Bug**: RANSAC - perfect horizontal line generate an exception
   https://github.com/scikit-learn/scikit-learn/issues/19497
3. **Bug**: PolynomialFeatures doesn't work correctly when degree=0
   https://github.com/scikit-learn/scikit-learn/issues/19551
4. **Bug**: ColumnTransformer treats empty arrays differently depending on their representation
   https://github.com/scikit-learn/scikit-learn/issues/19550
5. **Feature**: Poisson criterion in RandomForestRegressor
   https://github.com/scikit-learn/scikit-learn/issues/19304

Step 2

The Development Process

This assignment is the first time we truly worked together as a design team, and so it was focused heavily on creating, modifying, and optimizing a design process that we felt worked well for our team.

Initially, we held about two meetings a week. During the first week of the assignment, we used this time to decide on which issues we would be working on, how we would be delegating the tasks, and the specific routines that we wanted to have running for this week. After much deliberation, we decided on the following design processes to follow for our group:

Scrum and Team Organization

It was decided a few weeks beforehand that our team would be following Agile methodologies, as it is what we felt was most efficient and what we were most comfortable with. In doing so, we held weekly scrum meetings on Zoom to assess and discuss task assignments and their progression.

During the weeks of the assignment itself we tended to have one extra team meeting each week to help organize our workflow and understand the expectations of the assignment and team members (Scrum leader). During these meetings, we built up our sprint tasks, and would modify them as needed for the assignment work ahead. The extra meetings were also held when finalizing work, to ensure everything that needed to be delivered was completed before the due date.

See the section Sprint 1 Report for more information.

Jira and Work Delegation

For Assignment 2, we decided that our development process needed to focus on three different tasks: Implementation, Testing, and Documentation. As such, we decided to structure our work flow and JIRA boards accordingly.

Each of the tasks we created on JIRA were headed as the title of a specific issue. For this week in particular, we had tasks labeled *19538* and *19304* to remain consistent with existing issues of the Scikit-learn remote repository. Each of these tasks were given their own Implementation, Testing, and Documentation subtasks, which were each assigned to an individual team member.

Git and Code Organization

During this assignment week, we had our planned git processes put to the test. After much back and forth, and several changes over what we thought would work best, we ended up with a well-separated workflow that involved strict separation of changes and strong team involvement for every modification and change.

Our process involves creating a separate branch for each JIRA subtask we create on an issue (for example, for issue 19304 we have branches 19304/Implementation, 19304/Testing, and 19304/Documentation) for each assigned team member to work on. Upon finishing the required work, a pull request is created which must be verified by at least three other team members before it can be merged into the master branch. This makes it so the majority of the team agrees and understands the changes being made.

Bug #19538

Files Changed

- _sequential.py: https://github.com/UTSCCSCD01/course-project-spaghetti-code/blob/master/scikit-learn/sklearn/feature_selection/_sequential.py
    - Documentation component: Line 125
    - Implementation component: Line 118, Lines 133-145, and Line 219
- Test_feature_select.py: https://github.com/UTSCCSCD01/course-project-spaghetti-code/blob/master/scikit-learn/sklearn/feature_selection/tests/test_feature_select.py
    - Testing component: Lines 671-691

Documentation

The docstring for the method "*fit*" in the class SequentialFeatureSelector had to be changed to reflect that it takes the parameter "*y*" as an optional parameter, with a default value of *None*. Other related methods (such as "*fit_transform*" from the class TransformerMixin, which is what calls the method "*fit*" from SequentialFeatureSelector in the Customer Acceptance Test) already had this in their docstrings and implementations and did not need to be changed.

Implementation

Implementation involved rewriting the parameter arguments of the SequentialFeatureSelector class to include setting the default *"y"* parameter as *None*, allowing for the class to be constructed without an input *"y"* specified.

To allow the code to run in this situation, allowing a SequentialFeatureSelector to be created without a *"y"* parameter specified, we first modified the class' tags, setting the tag *"requires_y"* to be set to *"False"*. This will allow the program to actually attempt to create a

SequentialFeatureSelector to be created without a *"y"* parameter, as scikit-learn classes are set up to raise an exception if a *"y"* parameter is inputted without the required tag specifying it is allowed.

Finally, the actual reason that a *"y"* parameter would not initially work in the initialization of a SequentialFeatureSelector is that the input parameters *"X"* and *"y"* were being validated as proper inputs using the function *"_validate_data(X, y,...)"* function. Without a *"y"* value, this validation step would raise an exception, even though the class itself can work properly without needing a specified *"y"* value. To correct this, we simply added a statement that checks for the existence of a *"y"* value, and calls *"_validate_data(X,...)"* without a *"y"* value, allowing for the class to validate the data and run through the rest of the initialization function without issue.

Testing

Testing for issue 19538 consists of two tests.

One is the test_unsupervised_model_fit, this test takes in the "*X*" value and no "*y*" value and sees if the SequentialFeatureSelector functions without a value for "*y*" passed in. In this test we create a new SequentialFeatureSelector where we call fit on it with no "*y*" value and check to see if the shape is properly defined after fit is called.

The second test is test_no_y_validation_model_fit which takes in "*X*" and "*y*" values and checks if the fit function properly throws a TypeError or ValueError in the case of an invalid y value.

Customer Acceptance Test

Previously running the code snippet below ended with:

```
>>> data = sfs.fit_transform(data)
TypeError: fit() missing 1 required positional argument: 'y'
```

when fit_transform is run as the bug was that a y value was required to be passed into fit. Now with the bug fixed the following snippet properly outputs the expected result in data.shape:

```
>>> import numpy as np
>>> from sklearn.feature_selection import SequentialFeatureSelector as SFS
>>> from sklearn.cluster import KMeans
>>> data = np.array([[1,1,1,1,1], [2,2,2,2,2], [3,3,3,3,3], [4,4,4,4,4],
[5,5,5,5,5], [6,6,6,6,6]])
>>> model = KMeans(3)
>>> sfs = SFS(model, n_features_to_select=3)
>>> data = sfs.fit_transform(data)
>>> data.shape
(6, 3)
```

Feature #19304

Files Changed

- _forest.py: https://github.com/UTSCCSCD01/course-project-spaghetti-code/blob/master/scikit-learn/sklearn/ensemble/_forest.py
    - Documentation component: Line 1318 and Lines 1321-1323
    - Implementation component: Lines 310-314
- Test_forest.py: https://github.com/UTSCCSCD01/course-project-spaghetti-code/blob/master/scikit-learn/sklearn/ensemble/tests/test_forest.py
    - Testing component: Lines 78-81 and Lines 1501-1537

Documentation

This feature involves the integration of a new regression computation criterion for the RandomForestRegressor class. As a result, it was necessary to add Poisson as an official valid criterion to it's docstring. The updated documentation now lists Poisson as a new feature to the criterion parameter, as well as what it's purpose is.

Implementation

The implementation process required the need for validating the inputted data. The RandomForestRegressor allows for two parameters in order to call it's respective fit estimator. These parameters are:

- X: The input samples
- y: The target values

The random forest regressor fits the input samples with a reduction in Poisson deviance of the target values to find splits. The key functionality of the Poisson criterion, in contrast to other criterions, is that the target values must be nonnegative. Hence, the implementation of this feature consists of validating these nonnegative values. In the case that there are such values, a ValueError exception is thrown.

Testing

The testing for feature #19304 consists of two tests in the file test_forest.py: test_poisson_regression and test_random_forest_regressor_negative_value.

The first test test_poisson_regression checks if the RandomForestRegressor fit() method works with a valid dataset with criterion "poisson". It is based off of an existing test in test_forest.py: test_regression. The original test_regression was parameterized to include testing for all regressors and regressor criteria in _forest.py. However the test datasets used in test_regression had negative y-values in the variable y_reg, which are not allowed for the poisson criterion. So the new test test_poisson_regression uses a new variable y_reg_non_neg

which is created by adding the minimum value in y_reg + 1 to every element in y_reg to create a regression dataset with non-negative y values. Possibly due to the adjusted dataset, the RandomForestRegression with criterion poisson, has a lower score value than 0.94, which is the expected score for regression tests in test_regressor. So test_poisson_regression checks for a lower score value 0.9.

The second test test_random_forest_regressor_negative_value checks if RandomForestRegression with criterion “poisson” throws a ValueError when there is a negative y value.

- New y regression dataset: https://github.com/UTSCCSCD01/course-project-spaghetti-code/blob/master/scikit-learn/sklearn/ensemble/tests/test_forest.py#L79
- test_poisson_regression: https://github.com/UTSCCSCD01/course-project-spaghetti-code/blob/master/scikit-learn/sklearn/ensemble/tests/test_forest.py#L1524
- test_random_forest_regressor_negative_value: https://github.com/UTSCCSCD01/course-project-spaghetti-code/blob/master/scikit-learn/sklearn/ensemble/tests/test_forest.py#L1530

Customer Acceptance Test

The newly implemented feature enables users to expand on the specification of the regression criterion they wish to utilize. This issue specifically indicates the desire to have officially enabled the usage of “Poisson”. Although this has been implemented in DecisionTreeRegressor the abstracted parent of RandomForestRegressor, no validation checks were performed to ensure the sanitization of the inputted data. Thus the following snippet of it’s usage ensures indicates how a customer would be using the above feature:

```
>>> import numpy as np
>>> from sklearn.ensemble import RandomForestRegressor
>>> y = [0, 1, 2]
>>> X = np.arange(6).reshape(3, 2)
>>> rf = RandomForestRegressor(criterion="poisson")
>>> rf.fit(X, y)
```

Furthermore, the validity check on the target values (parameter y) prevents any continuation of the RandomForestRegressor by preventing the fit() method from executing due to invalid inputs. Hence, the following snippet illustrates the ValueError occurring, which validates the Customer Acceptance criteria for this feature.

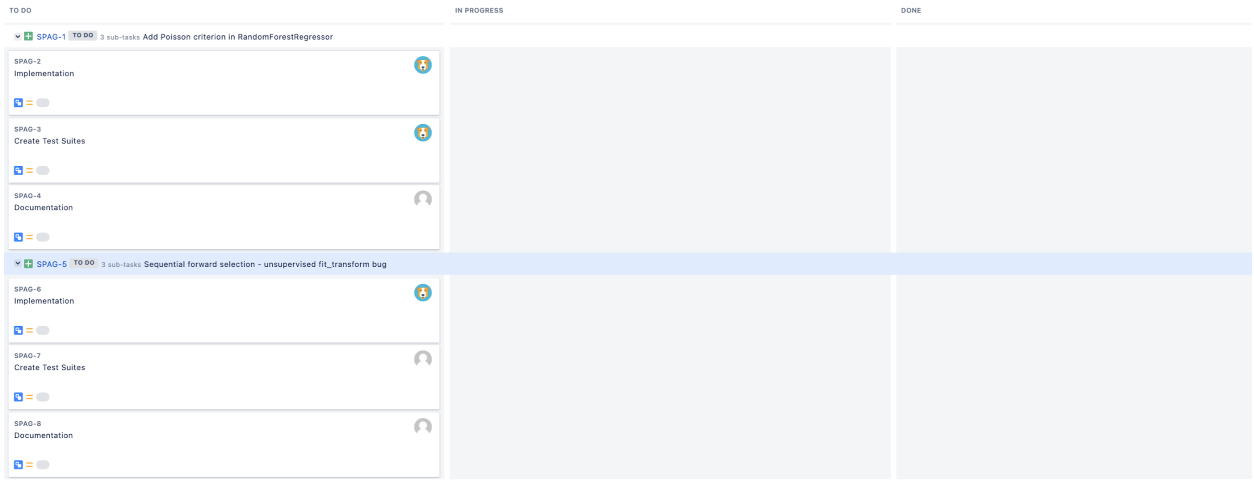
```
>>> import numpy as np
>>> from sklearn.ensemble import RandomForestRegressor
>>> X = np.array([[1, 2, 3]]).T
```

```
>>> y = [-1, 0, 1]
>>> rf = RandomForestRegressor(criterion="poisson", random_state=4)
>>> rf.fit(X, y)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
                                                                    File
"/mnt/c/Users/Julia/Documents/uoft/CSCD01/course-project-spaghetti-code/scikit-learn/sklearn/ensemble/_forest.py", line 311, in fit
    raise ValueError("The array of target training values (inputted y) "
ValueError: The array of target training values (inputted y) contain a negative entry, which is invalid for the Poisson criterion.
```

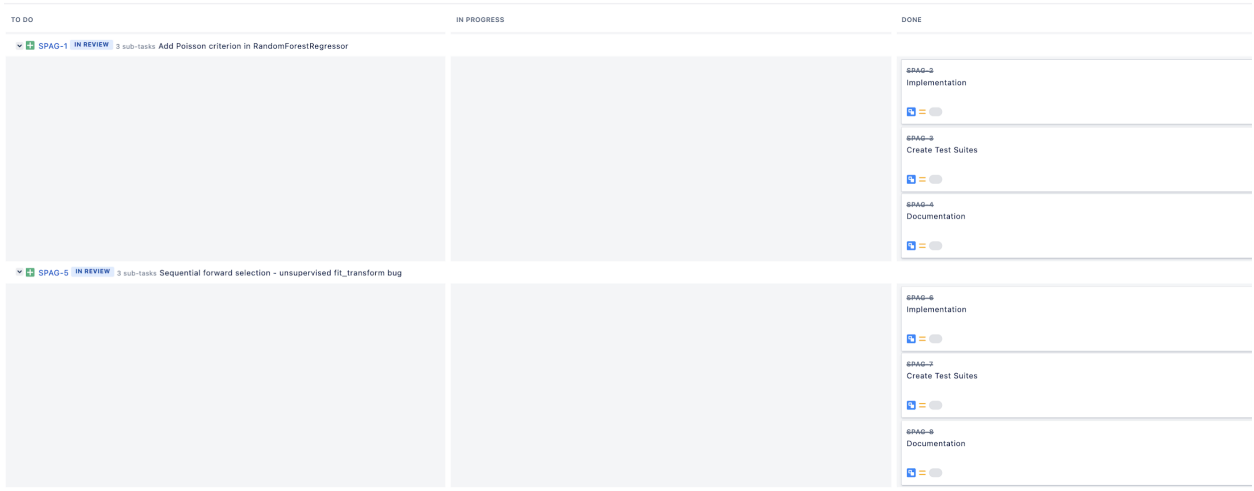
Sprint 1 Report

March 1, 2021 - March 7, 2021

Snapshot of sprint JIRA board at beginning of Sprint 1



Snapshot at end of Sprint 1



Sprint 1 Plan

SPAG-1: Add Poisson criterion in RandomForestRegressor
SPAG-5: Sequential forward selection - unsupervised fit_transform bug

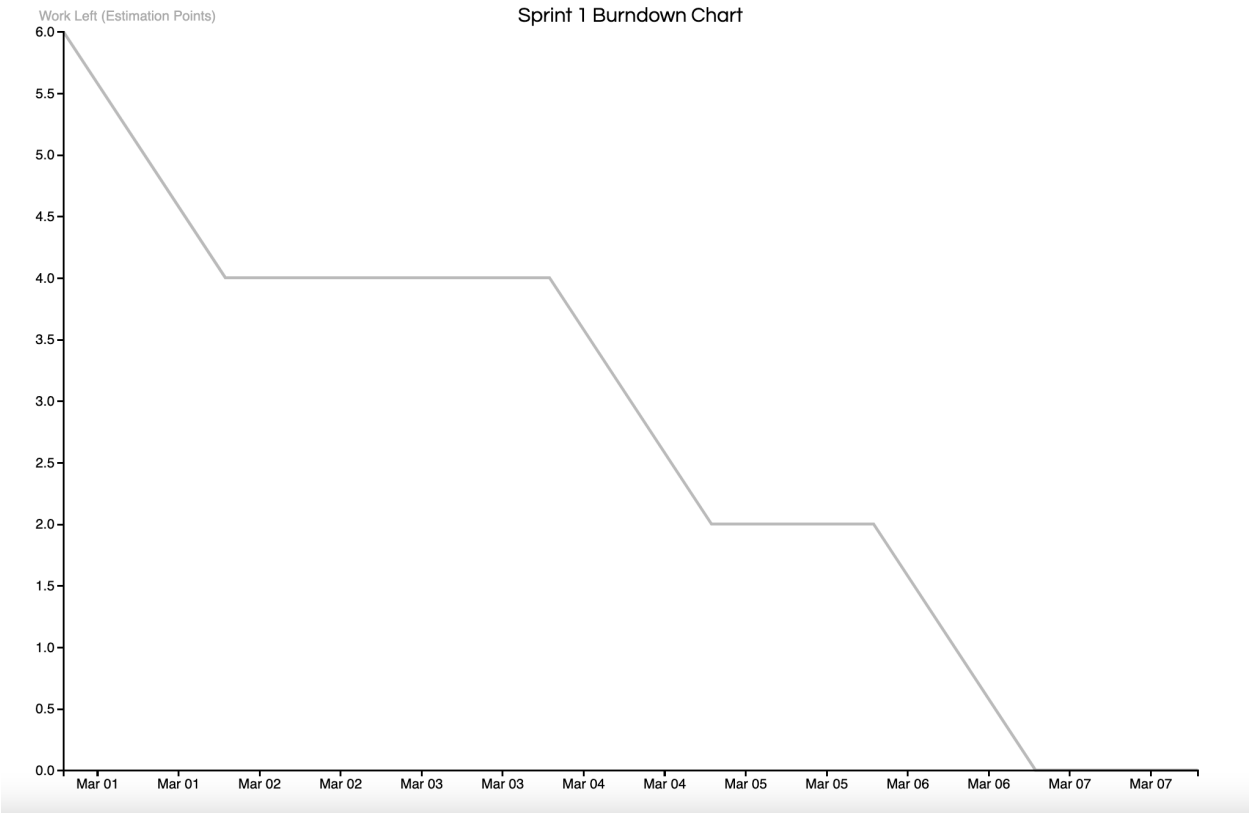
Feature/Bug	Task	Estimated Cost (hours)	Assigned

SPAG-1	SPAG-2: Implementation	1	Julian
SPAG-1	SPAG-3: Test Suites	1	Laphonso
SPAG-1	SPAG-4: Documentation	1	Jesse
SPAG-5	SPAG-6: Implementation	1	Saad
SPAG-5	SPAG-7: Test	1	Andrew
SPAG-5	SPAG-8:Documentation	1	Adam

Sprint 1 Workchart: March 1, 2021 - March 7, 2021

Feature	Task	Cost	Day 1	Day 2	Day 3	Assigned
SPAG-1	SPAG-2	1	1	0	0	Julian
SPAG-1	SPAG-3	1	0	1	0	Laphonso
SPAG-1	SPAG-4	1	0	0	1	Jesse
SPAG-5	SPAG-6	1	1	0	0	Saad
SPAG-5	SPAG-7	1	0	1	0	Andrew
SPAG-5	SPAG-8	1	0	0	1	Adam
Total		6	2	2	2	
Work Left		6	4	2	0	

Sprint 1 Burndown Chart



Report Summary

- Our sprint plan consisted of one feature and one bug. SPAG-1: Add Poisson criterion in RandomForestRegressor and SPAG-5: Sequential forward selection - unsupervised fit_transform bug.
- Each feature/bug had three corresponding subtasks which were completed and used as part of the acceptance criteria for the main feature/bug. Subtasks included tickets for Implementation, Test Suite creation, and Documentation.
- As shown in the sprint burndown chart and the JIRA issue board, this sprint was finished with no tasks remaining, which means no issues from this sprint will need to be brought into the next sprint.

What worked well? What could we have done differently?

- Project velocity went as expected and all 6 estimation points were completed by the end of this sprint with three working days. We organized the sprint so that each estimation point is equivalent to one hour of work, and we felt that this also gave an accurate representation of the cost of each task.

- One thing we could have improved or done differently in this sprint could be the level of communication. Although we had regularly scheduled meetings, it would be convenient to include or introduce new ways to quickly communicate such as for example, ad-hoc style meetings when someone in the team needs some quick clarification on a topic.

Evidence of regular team meetings

Meeting logs:

Meeting 1 - February 19, 2021

- Go through and pick 5 issues from scikit-learn issue list
    - *19538*
    - *19497*
    - *19550*
    - *19551*
    - *19304*
- Decide upon presentation order for Tutorial
    - Each of us picks an issue to present
    - Julian will screen-share terminal
- Quickly prep A1 presentation
    - Each of us will present the section we wrote
    - Julian will screen-share report

Meeting 2 - February 26, 2021

- Create a Sprint board
- Write down our sprint goals
- Create tasks for 19538 & 19304
    - Decide on creating Implementation, Documentation, and Testing subtasks for each task

Meeting 3 - March 1st, 2021

- Sprint planning and estimation points
- Assigning tasks
- Start Assignment 2 report and assign sections to team members
- Beginning of Sprint 1 (March 1st - March 7th)

Meeting 4 - March 7th 2021

- Review and merge all the pull requests
- Sprint Burndown chart
- Finalize the report and submit