CSCD01

Assignment 4

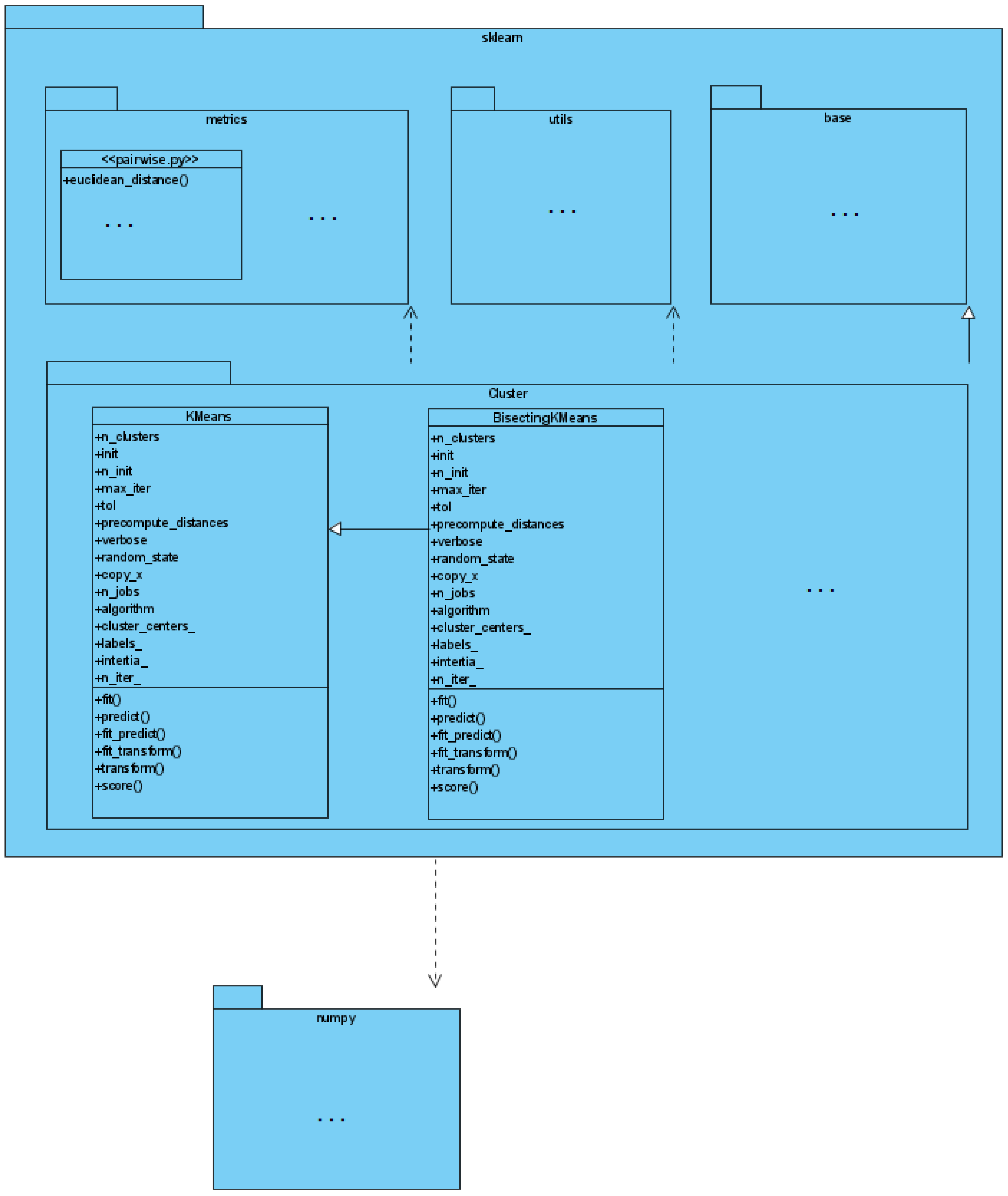
April 4th, 2021

Table of Contents:

<u>Issue #1: “Bisecting K-means #14214</u>
https://github.com/scikit-learn/scikit-learn/issues/14214

<u>Issue Description:</u>
Issue #14214 is a feature request for a clustering algorithm called Bisecting K-Means. Clustering algorithms are unsupervised learning algorithms that specialize in finding subgroups (called clusters) within datasets. In the scikit-learn library, there is already a similar algorithm implemented (called K-Means). However, Bisecting K-Means offers some things K-Means does not; K-Means typically identifies spherical shaped clusters only, while Bisecting K-Means does not have this limitation. Bisecting K-Means also offers better entropy measurement.
As a brief summary, K-Means divides a dataset into K clusters, each cluster having its own centroid (the mean of each cluster). K-Means starts by choosing K centroids, and improves upon estimation of the set of centroids.
On the other hand, Bisecting K-Means starts out by choosing two centroids of the whole dataset. It then chooses the cluster with the highest sum of squared errors (SSE), and replaces the centroid of such cluster with two more centroids by performing 2-Means (K-Means with 2 clusters). As such, conceptually, Bisecting K-Means uses K-Means within its own algorithm.

<u>Issue investigation and design:</u>
The changes to the library will be local to the clustering module, located at the path `sklearn/cluster/_kmeans.py`.
Each clustering algorithm in the library is its own class. Therefore, to stay consistent with the API, Bisecting K-Means will be its own class, much like Mini-Batch K-Means (another clustering algorithm related to K-Means) is its own class. Although the fit and predict methods are different, Mini-Batch K-Means inherits from K-Means. This makes sense because their instantiations are identical. This is the same case with Bisecting K-Means and K-Means, and as such, Bisecting K-Means will also inherit from K-Means. This also stays consistent with the architecture of the module as-is.
Since Bisecting K-Means uses K-Means within its own algorithm, it will also depend on K-Means in addition to inheritance. We would show this dependence by creating the K-Means object during the call to the `fit` method.
The change made to the module to incorporate this design is described by the following UML class diagram:



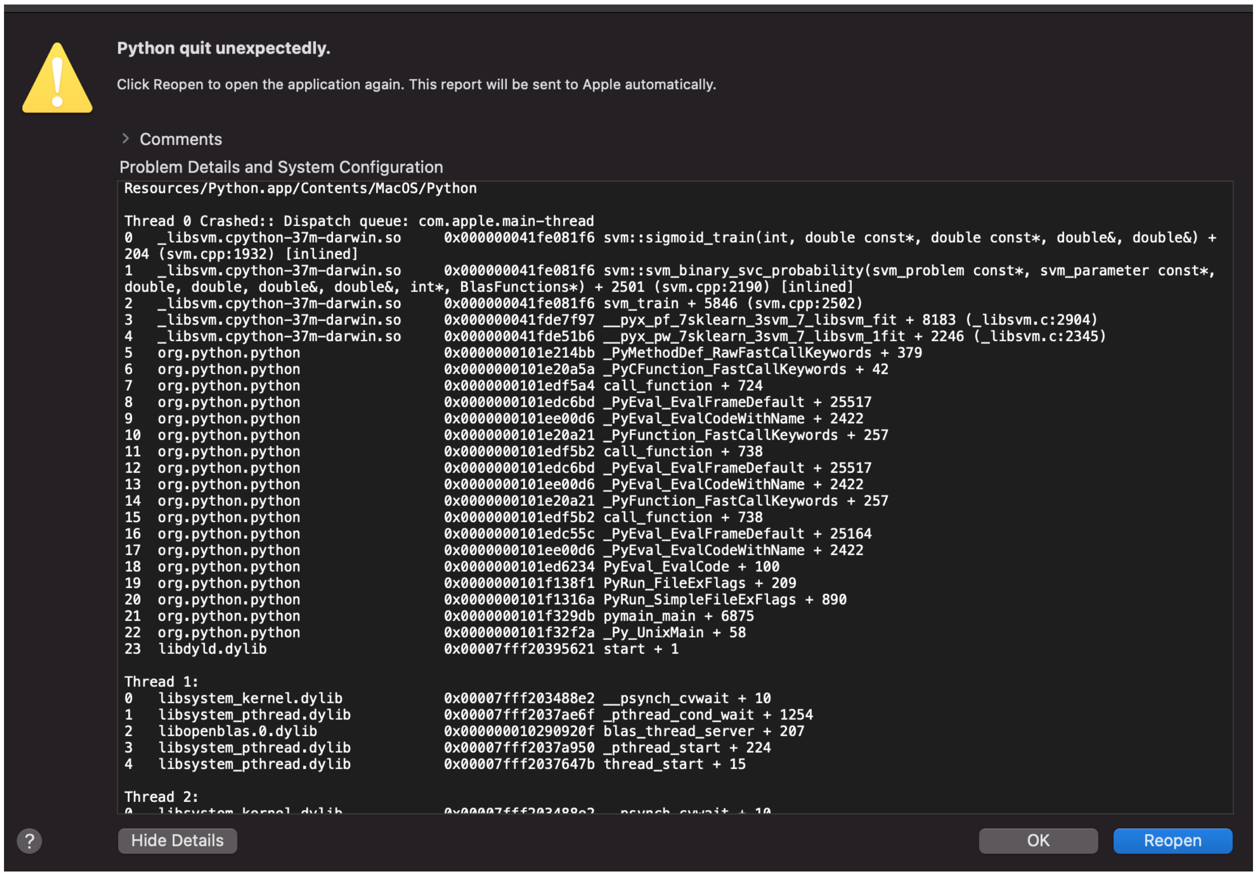
All of the attributes will be identical, as most will be passed on to KMeans when the algorithm needs to perform 2-Means clustering. An exception would be `n_clusters`, as this will dictate the final number of clusters to end on; Bisecting KMeans will pass on the value of “2” to KMeans in every step that requires 2-Means clustering.

In terms of the algorithm; the implementation will heavily rely on numpy arrays and the operations offered by numpy arrays, as well as the `eculidean_distance()` function offered by the metrics module of sklearn.

<u>**Issue #2: “Segmentation Fault with SVC with Probability = True and degree 3”**</u>
https://github.com/scikit-learn/scikit-learn/issues/15008

<u>Issue Description:</u>
A segmentation fault occurs when attempting to instantiate a SVC object with probability = True and degree 3. When running the code below, users will hit a segmentation fault after about ten minutes.

```
import numpy as np
features = np.random.rand(2100000,768)
target = np.random.randint(low =100000,high=500000, size=2100000)
from sklearn.svm import SVC
svc =
SVC(kernel='poly',gamma=100,C=10,degree=3,decision_function_shape='ovo',probability=Tr
ue,random_state=42,verbose=True)
svc.fit(features,target)
```



Python quit unexpectedly.

Click Reopen to open the application again. This report will be sent to Apple automatically.

Problem Details and System Configuration

```
Resources/Python.app/Contents/MacOS/Python

Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   _libsvm.cpython-37m-darwin.so     0x000000041fe081f6 svm::sigmoid_train(int, double const*, double const*, double&, double&) +
204 (svm.cpp:1932) [inlined]
1   _libsvm.cpython-37m-darwin.so     0x000000041fe081f6 svm::svm_binary_svc_probability(svm_problem const*, svm_parameter const*,
double, double, double&, double&, int*, BlasFunctions*) + 2501 (svm.cpp:2190) [inlined]
2   _libsvm.cpython-37m-darwin.so     0x000000041fe081f6 svm_train + 5846 (svm.cpp:2502)
3   _libsvm.cpython-37m-darwin.so     0x000000041fde7f97 __pyx_pf_7sklearn_3svm_7_libsvm_fit + 8183 (_libsvm.c:2904)
4   _libsvm.cpython-37m-darwin.so     0x000000041fde51b6 __pyx_pw_7sklearn_3svm_7_libsvm_1fit + 2246 (_libsvm.c:2345)
5   org.python.python                 0x0000000101e214bb _PyMethodDef_RawFastCallKeywords + 379
6   org.python.python                 0x0000000101e20a5a _PyCFunction_FastCallKeywords + 42
7   org.python.python                 0x0000000101edf5a4 call_function + 724
8   org.python.python                 0x0000000101edc6bd _PyEval_EvalFrameDefault + 25517
9   org.python.python                 0x0000000101ee00d6 _PyEval_EvalCodeWithName + 2422
10  org.python.python                 0x0000000101e20a21 _PyFunction_FastCallKeywords + 257
11  org.python.python                 0x0000000101edf5b2 call_function + 738
12  org.python.python                 0x0000000101edc6bd _PyEval_EvalFrameDefault + 25517
13  org.python.python                 0x0000000101ee00d6 _PyEval_EvalCodeWithName + 2422
14  org.python.python                 0x0000000101e20a21 _PyFunction_FastCallKeywords + 257
15  org.python.python                 0x0000000101edf5b2 call_function + 738
16  org.python.python                 0x0000000101edc55c _PyEval_EvalFrameDefault + 25164
17  org.python.python                 0x0000000101ee00d6 _PyEval_EvalCodeWithName + 2422
18  org.python.python                 0x0000000101ed6234 PyEval_EvalCode + 100
19  org.python.python                 0x0000000101f138f1 PyRun_FileExFlags + 209
20  org.python.python                 0x0000000101f1316a PyRun_SimpleFileExFlags + 890
21  org.python.python                 0x0000000101f329db pymain_main + 6875
22  org.python.python                 0x0000000101f32f2a _Py_UnixMain + 58
23  libdyld.dylib                     0x00007fff20395621 start + 1

Thread 1:
0   libsystem_kernel.dylib            0x00007fff203488e2 __psynch_cvwait + 10
1   libsystem_pthread.dylib           0x00007fff2037ae6f _pthread_cond_wait + 1254
2   libopenblas.0.dylib               0x000000010290920f blas_thread_server + 207
3   libsystem_pthread.dylib           0x00007fff2037a950 _pthread_start + 224
4   libsystem_pthread.dylib           0x00007fff2037647b thread_start + 15

Thread 2:
0   libsystem_kernel.dylib            0x00007fff203488e2 __psynch_cvwait + 10
```

Implementation and Fix:
The problem occurs in the file *sklearn/svm/src/libsvm/svm.cpp*, specifically in the function *sigmoid_train*. On first glance it looks like the issue is with an index overflowing as suggested by a commenter on the issue. However, after debugging, the error is with a malloc in the function **PREFIX(train)*. When the variable nr_classes gets above ~390000 the error occurs since it mallocs something of size 79154844021 (nr-classes*(nr_classes-1)/2). This is above the maximum value for an integer in C++, (see figure 2) and thus a segmentation fault occurs.

By casting this to an int64_t the segmentation vault is avoided, as by figure 1, the maximum value for an int64_t is 9,223,372,036,854,775,807, which is greater than 79154844021.

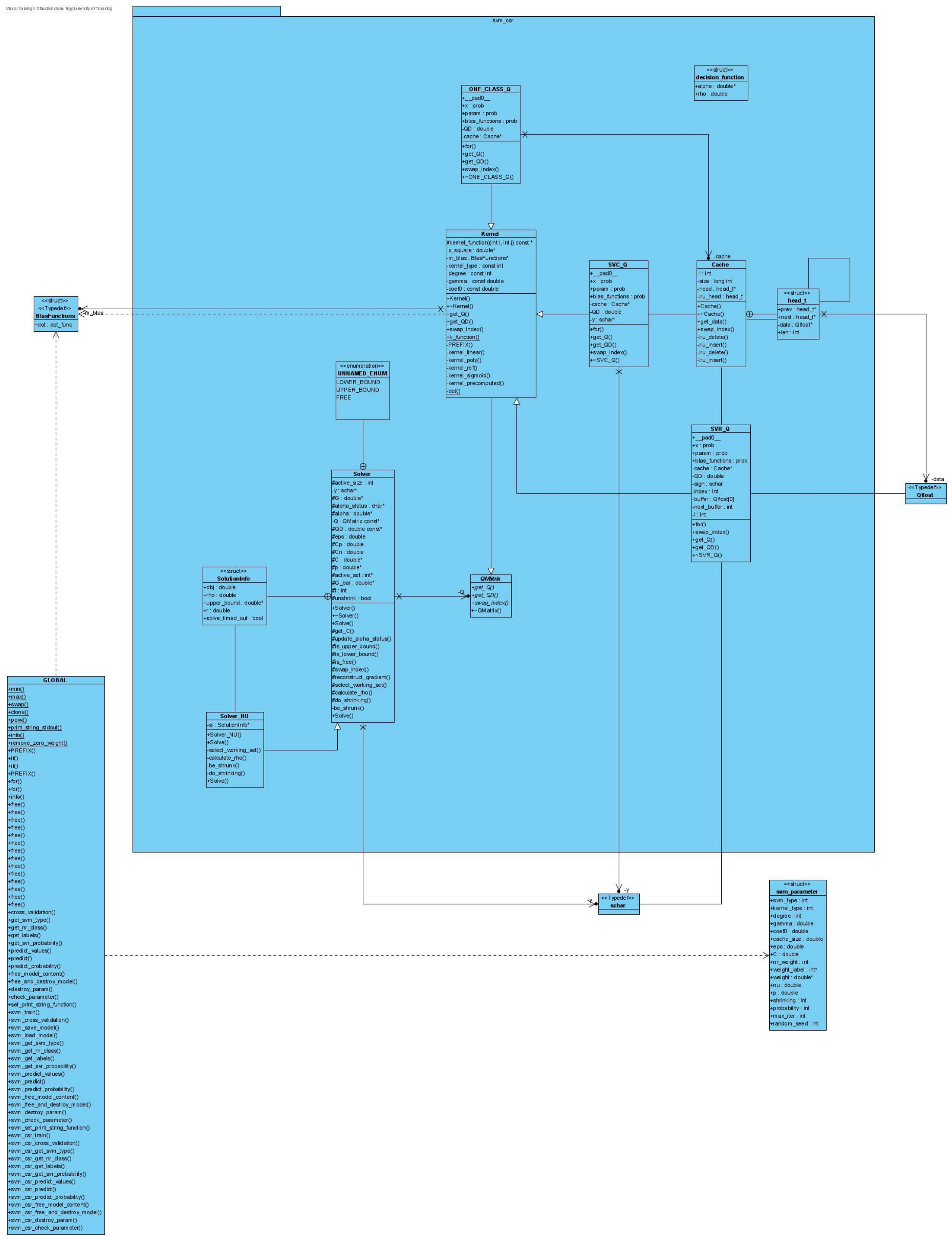
As a preventative measure we also looked at other areas of the code where this integer overflow could happen. To ensure the error doesn't also occur there, we changed the appropriate variables to an int64_t.

	int	int64_t
Memory allocation size	4 bytes	8 bytes
Max Value	2,3147,483,647	9,223,372,036,854,775,807

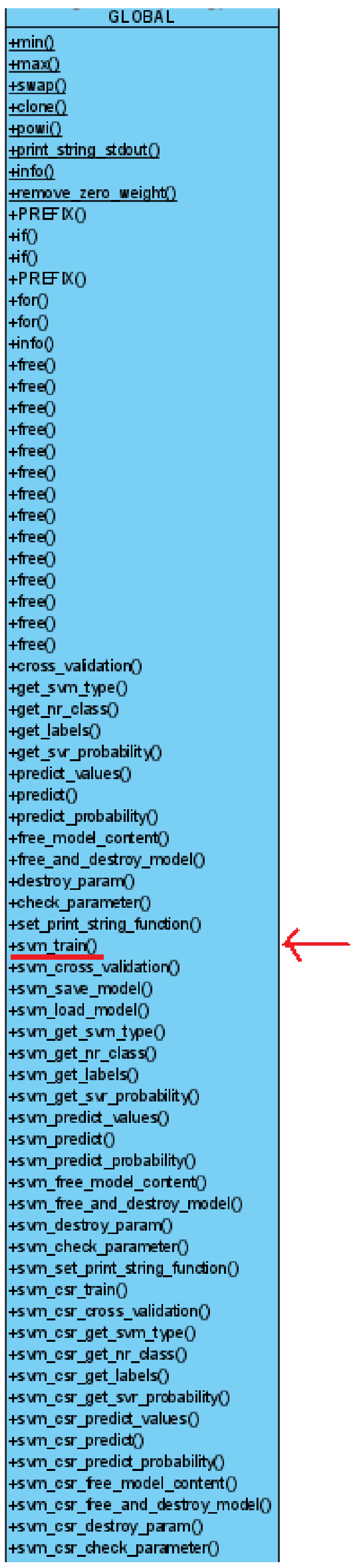
Figure 1

UML Changes

The bug in question concerns/affects the svm section of the UML as seen here:



In terms of the UML impact for our bug, no new classes or functions were added to the uml. For the bug in question, the main part of the uml we changed and are concerned with is the svm_train() function as displayed here:



Bug Selected:

For our bug fix, we selected bug #2 to implement. The choice was made because we felt it was an opportunity to make a significant contribution to a frequently used feature. We are also confident about the fix's chance to merge into the main sci-kit learn branch.

Why This Bug is "Hard":

Our chosen issue did not have a labelled difficulty. We determined that the fix should be considered "hard" for the following reasons:

1. The stack trace for the error is not actually where the bug is located
2. The code takes upwards of 10 minutes to run and produce the error
3. The fix was in C++, a language our group is less comfortable working in

We verified these reasons through email correspondences with our TA's, and were given approval to fix the bug.

Testing:

Unit Testing:

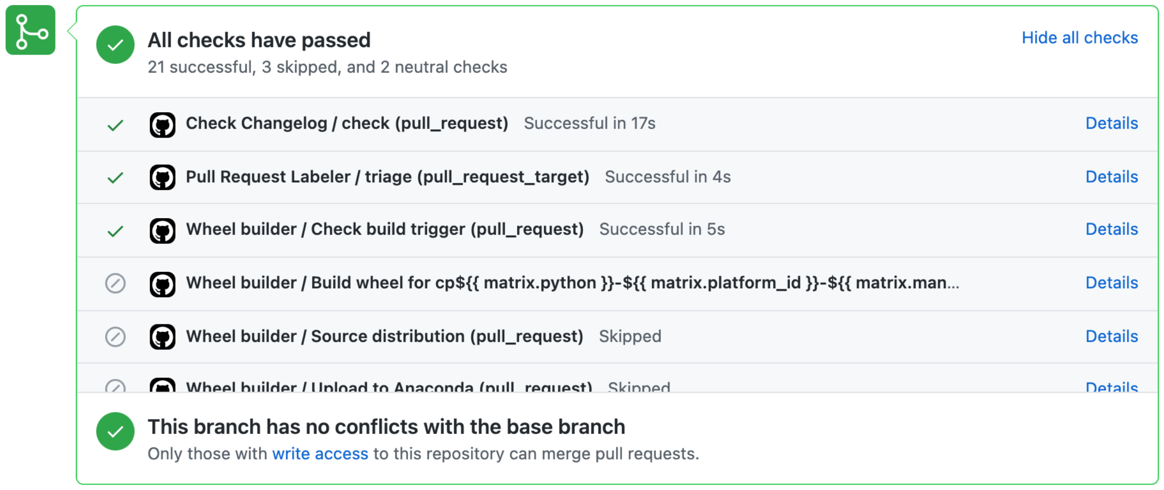
We do not believe this fix requires unit testing since it takes many hours to fully run, and uses a lot of computing resources. However, for the sake of the assignment, to unit test, we would create a test that runs the code above. If the process terminates without a segmentation fault, it would mean that our fix is successful.

Acceptance Testing:

A customer acceptance test to validate the correctness of our fix would go as follows:

1. Create a matrix represented as a nested array of size (2100000 x 768) and populate it with random samples of a uniform distribution over 0,1
2. Create an array of size 2100000, with random values in it between 100,000 and 500,000.
3. Instantiate an SVC object with specific parameters specified above
4. Try to fit the matrix and array to the SVC object, using the SVC's fit method

Our fix also passed scikit-learn's suite of acceptance tests, as shown here in our pending pull request: https://github.com/scikit-learn/scikit-learn/pull/19782



All checks have passed
21 successful, 3 skipped, and 2 neutral checks

Hide all checks

✓ Check Changelog / check (pull_request) Successful in 17s Details
✓ Pull Request Labeler / triage (pull_request_target) Successful in 4s Details
✓ Wheel builder / Check build trigger (pull_request) Successful in 5s Details
⊘ Wheel builder / Build wheel for cp${{ matrix.python }}-${{ matrix.platform_id }}-${{ matrix.man... Details
⊘ Wheel builder / Source distribution (pull_request) Skipped Details
⊘ Wheel builder / Upload to Anaconda (pull_request) Skipped Details

This branch has no conflicts with the base branch
Only those with write access to this repository can merge pull requests.