Group 24
Adam Sjöberg, Andreas Hammarstrand, Casper Kristiansson, Martin Lindefors, Victor Österman

Assignment 4 report

Onboarding

Repo changed

We've changed the project that we are working on due to the complexity of the open issues and software in the *Pytensor* project, which was used in assignment 3. Therefore, the group has chosen to work on another project for assignment 4, named *Pandas*.

Onboarding documentation

The documentation *Creating a Development Environment* describes how to set up the developer environment required to build the project. Whilst it is informative, it seems to contain some flaws. There are multiple options for creating the development environment, but only a handful of them seem to work correctly. We have tested the four installation options with success on only two of them. Worth noting is that the way that worked was not a straight path for all group members. They seem to be emigrating towards a more stable build process, which might be why we encountered numerous issues trying to set up the environment. Some details in the documentation, which are only presented as small notes, seemed very important for the build to be completed successfully. Despite these issues, we managed to successfully build the project without issues when we identified the correct installation option. Around 1K of the tests still failed due to what we interpreted as a lack of permissions to connect to certain services. Considering the fact that there are more than 200K tests and that there is no apparent way to get the correct permissions, this was not regarded as an issue.

Experience in running a new project

Building the previous project was significantly simpler than building the *pandas* developer environment. Multiple of the group members stumble upon issues when trying to install and set up the development environment. However, due to the improvement in terms of clearness in GitHub issues and the understandability of the code, the new project is more suitable for this assignment. In addition, it was also a lot more interesting to work with a bigger project in terms both of code but also in usage. Pandas is typically a package that most people that has used Python before know of. When working with pandas we also noticed that they had a lot more documentation regarding both the code but also the processes of contributing to the repository which helped a lot to understand the code we were working with.

Part 2: Issue resolution

Issue Description

The issue is a bug when doing subtraction between two pd.Series with different names and types. These pd.Series are, in short, tables functioning similarly to matrices containing values. The issue presented is a non-trivial one since it is a quite hard thing to get into due to the fact that the problem could be anywhere in the code and also that it is one of the core

features within the whole *pandas* module. The issue wasn't fully described since the problem also was for each different arithmetic operator (+-*/).

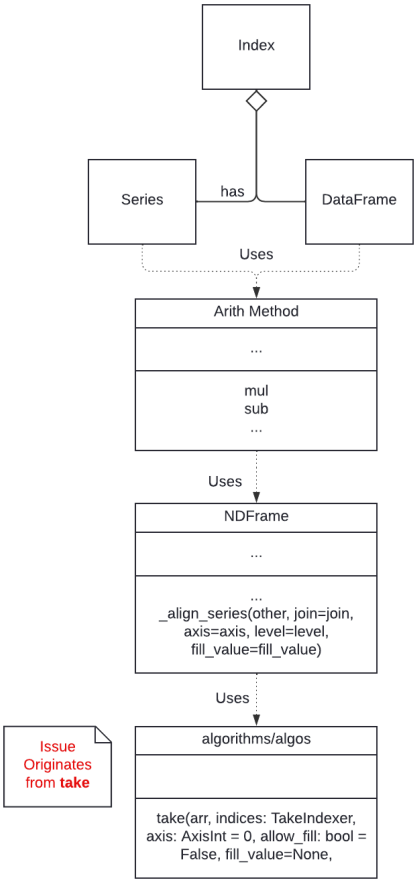

Figure 1: *UML Diagram of Issue*

Existing Regression Tests

Pandas test suite contains around 212,000 tests. Initially, we ran the tests whole suite on several group members computers to minimise the risk of failing multiple tests due to faulty onboarding. We all got a bit different results with a diff of tests failing within a bound of 5-10 tests. In total, around 700 failed tests. After these tests, we found the source of our problem to determine whether the failed tests interfered with our issue, we concluded after some digging that none of the failing tests were related to our issue.

Requirements

Prior to starting the process of solving the issue, we discussed all relevant requirements in terms of the issue at hand. The author of the issue described the problem in some detail but did not mention where the issue occurred other than on a high level, where the error output was the only concise information available. We therefore concluded that we needed to examine the code and existing tests related to the issue, in an attempt to get the whole picture. After this, we decided that the following requirements were necessary to solve the issue:

ID	Title	Description	Plan of action
RQ1	Identify the root of the issue	Since there is no clear description of where the issue occurs, we need to examine the error output and determine where the fault is located	Each member of the team examines the error output and attempts to figure out at what point the code shows faulty behaviour. The whole team will then proceed to discuss their findings and collectively determine the root of the issue
RQ2	Identify actual expected behaviour	The author does not describe why this is considered a bug, only that it is one. We need to determine why this is a bug, and what the fix looks like	When the root of the issue is known, we will study the related code to understand the reasoning behind classifying the issue mentioned in the issue as a bug.
RQ3	Implement test coverage for the related code prior to solving the problem	Test-driven development is part of this assignment, and an effective way to understand what the problem is, and how the system is expected to function	When we understand the problem and its causes, we will create tests that should represent the expected behaviour. The test must resemble the code to reproduce the bug, mentioned in the issue, as well as coverage of any additional branches implemented.
RQ4	Solve problem to meet the expected behaviour	To complete the task, we hope to solve the problem and submit the pull request to the main repository	When the tests have been implemented, we will produce a solution which results in minimal code and which does not affect backwards compatibility problems or affects other aspects of the system in any negative way

Existing tests related to the issue

The existing testing for both the general usage of arithmetic operations on pandas series and the individual tests for the “take” function were thoroughly developed. The general arithmetic operations on pandas series had already over 17 tests made for them. Most of these tests have wide ranges of different values and indices. As for the “take” function, there existed about 14 different tests. These tests converse a wide range of usage cases on the function which results in a good coverage of the function. Even though the function “take” had good coverage it didn't handle the case where a specific input could be None type. This meant after fixing the issue all of the existing tests still passed.

Our new tests

Our new tests cover the two different areas which in this case are for the specific function “take” and the arithmetic operations of the pandas series. Because the function “take” is widely used in pandas operations (240+ references) it was important to make sure that no other tests in the project would fail with our newly added bug fix. Here is the test result prefix

to the implementation postfix of it. The first area of tests which was for the function “take” consisted of simply checking for the case where the input parameter indices might be None type. The function would then handle the issue and still return a viable result. The second area of tests we added was the arithmetic operations of the pandas series. Here we added five new test cases to handle the situations where the index of a pandas series could be of different data types or different names. These tests are related to the specific issue that we solved where pandas gave an exception trying to convert (infer) the indexes to the same type.

The solution

Solving the issue was a bit difficult. Because the issue originates back to the overall usage of arithmetic operations of pandas we did have to read up on a lot of code/documentation to understand the code. After doing some digging and using a debugger we could narrow down the issue to why the code would give an exception when the indexes of a pandas series were different from each other. After tracking down why the issue was happening we had to spend a lot of time trying to understand how we could solve this in the best way possible. The reason why just understanding where we would implement the fix took a lot of time was because at the beginning we thought that the pandas series wasn't even calling the correct methods or that it shouldn't have been handled in the specific way it was being handled. After a lot of discussions, we decided to implement our solution in one of the core algorithms in Pandas “take”. This approach will simply handle the case where the indices might be of the type None. After implementing the solution we ran all of the tests in the codebase as well as our new test cases to make sure that our new feature doesn't break any existing code. The last part of implementing the solution was following the contribution guide of pandas. This consisted of running pre-commit (linit and styling guides) and writing to release notes.

Patch Notes

Pandas uses a tool called Sphinx to generate documentation for the project. As part of their documentation each PR needed to consist of comments in a release notes file where we needed to describe which function we modified and the issue referenced to it.

Part 3: Documentation

Time spent on the issue

Initially, we estimated the time required for this issue resolution to take around 20-25 hours for each group member. The reason for the time requirement was argued by the group that the code itself is poorly documented, which will result in some hours to understand the parts of the code that we need to resolve the issue. Furthermore, as we have mentioned earlier in the documentation, the issue itself was poorly described, which will result in some hours spent discussing how to interpret the issue and what the expected behavior should be. Lastly, we estimated that the time to write this documentation should take about 1 hour per group member and put a max time limit of two hours for each member for the documentation.

- Talk about how the time was spent during the issue resolution, maybe include the time logs.

The time spent during this issue resolution is shown in the table below

Group activity/individual work	Date: Time spent	Description
Group activity	2024-02-26: **2 h**	Meeting to determine what project to work on and to find an issue. We also preliminary divided the work between each group member.
	2024-02-27: **3 h**	Meeting to get the onboarding to work for each group member. We had some issues creating a dev-env.
	2024-02-29: **6 h**	Meeting where we resolved the last issues regarding the dev-env. We also discussed the contribution guidelines for the project. Made test for correct behavior (TDD), using pair programming Identified the bug and started to solve the issue, by using pair programming. Created a PR to the projects repo to
	2024-03-01: **4 h**	Wrote the remaining parts of the report. Checked for updates regarding our PR on the project repo. Went through the assignment to assert that we have covered everything in the grading criteria. Generated test logs after the issue was resolved and analyzed that everything was correct.
	2024-03-03: **2h**	Finished the report

Adam	2024-02-26: **2h**	Read about the project and tried to understand it
	2024-02-27: **3h**	Setup the dev environment and ran some tests then try to see where in the code the problem could be
	2024-02-28: **1h**	Tried to make some fast solutions.
Andreas	2024-02-27: **4 h**	Set up dev env (took most of the time), Generated test logs and coverage reports
	2024-02-28: **2 h**	Created understanding by analyzing crash logs, source code, and the issue.
	2024-02-28: **1 h**	Worked on a report on miscellaneous parts.
Casper	2024-02-27: **2h**	Look more at different issues
	2024-02-28: **4h**	Fix C++ bug with onboarding
Victor	2024-02-27: **3h**	Tried to make a dev-env in pandas to be able to run all tests.
	2024-02-27: **2h**	Tried to get a grip on what parts of the code that were relevant to the issue.
	2024-02-29: **1h**	Logged initial tests into a file and shared it with the group.
Martin	2024-02-26: **3h**	Tried, and eventually succeeded, in setting up the dev environment
	2024-02-27: **2h**	Getting accustomed to the project by inspecting the source code and tests. Also ran the tests.
	2024-02-29: **1h**	Debugged the stack trace to identify the reason behind the crash of the code

Essence

The team is on the status of “In Place”. Previously, the open source status retracted us down to “In Use” due to being new and confusing. However, in this case, we knew how to work with all of the workflows, despite being different. During the Open Source work, we had to employ a different way of working since issues could not be created or applied in the same way as we had previously done since now the project was on a public level and owned separately. We did, however, do as much as we felt comfortable doing on that front and made issues related to the different requirements on our end such that we could keep track of everything. Furthermore, the team was quickly able to adapt the remaining aspects of the appropriate workflows to the open-source project, such as the correct form in testing and documentation or atomic commits. To get higher, the team needs practice and experience such that the methodology becomes second nature and does not require time or consensus to adapt to new projects.

SEMAT Kernel

We began in the state of *“Initiated”,* on the 26th of February during a meeting of the team, as part of the lab assignment, where we were introduced to the topic and work requirements. In this scenario, the work requirements were to find a project and associated issues to solve within our team. We ended up choosing the Python library *pandas* and a bug within the library related to basic arithmetics between complex objects. The project and issue were chosen due to our immaturity within the field, such that we can realistically complete the task. The task at hand is then not big nor very impactful, but the nature of Open Source is frequent small updates to slowly improve upon the project. This can be related to the old Software idiom of “if it's painful do it more often”.

We then proceeded with assigning the team members to the issue in the open-source repository. A forked repository was created as per the *Contribution Guidelines* on the official documentation for *pandas*. All members cloned the repository and created the developer environment required to build the project. At this point, we were ready to start the task and thus had entered the *“Prepared”* state by the 27th of February. Despite the advantages of individually building the project, which made sure all members were onboarded in terms of the source code, there were some apparent drawbacks in our way of handling this state. As mentioned in the *Onboarding* section, the time spent on building the project was quite extensive. It is rather likely that if we would’ve collaborated in this process we would’ve avoided testing the same inaccurate ways of installing the project, and hence lessened the time to install the project.

Each member acquainted themselves with the source code and attempted to understand the issue and find its context within the overall source code. This is the point where the task had been started and when we had transitioned into the “*Started”* state by the 27th of February. The reason behind individually inspecting the source code was to make sure all members were equally informed about the source code and the overall architecture of the project. However, working in smaller teams might’ve resulted in more discussion about the different aspects of the source code and the purpose of the system. These types of discussions will more often than not increase the understanding, as you are forced to put your thoughts into words and compare them to the opinions and insights of others. This was a missed opportunity in our case.

We then moved on to *“Under Control”,* from 27th to 29th of February, wherein we worked on the project and issue at hand to solve the bug. Much of the time was spent together for a unified understanding of both the issue but also the underlying library, *pandas*, and the defined requirements are being handled as their tasks. During this time there were not many unexpected events or work tasks that popped up, most of the ones that did happen were because of misunderstandings and mistakes that were quickly reversed. We spent many man-hours on the project but did not complete a very significant issue. This can largely be attributed to the methodologies we were working with, *Essence* and the *Contribution Guidelines*, that required extra steps of documentation and double-checking what we had done to ensure that it did not violate any properties or checklists. This meant that the work we put in was inefficient in terms of productivity, however, we would argue that the outcome is more satisfactory as we can guarantee fewer mistakes, and when we do solve the actual problems we are aware of what needs to be done. That is to say, methodologies, such as Essence, can reduce efficiency but increase the quality of productivity, which is a trade-off that may be necessary or wanted in appropriately selected scenarios.

On the 29th of March, we entered the penultimate state *“Concluded”* when we published the pull request for our patch according to the *Contributions Guideline*, which marked the end of the primary work in this task. By following the guidelines for publishing a pull request for the repository, we made sure that we had created the optimal circumstances for actually receiving a positive review, and perhaps acceptance, of our solution. Despite this, there are probably additional actions we could have taken to make sure that the icing on the cake, i.e. merge of pull request, was achieved. We did not research all aspects of contributing to the repository and have most likely missed some aspects that could have increased our chances of having our pull request accepted.

Currently, we find ourselves in the *“Closed”* state as all remaining tasks related to documenting the experience in this report have been finalised. We are still eagerly waiting for an update on our pull request, which we hope will be accepted. This state is difficult to achieve and to evaluate, as there are always more things that can be done with finishing up work or closing down. There are always details that can be added or improved. For example, the PR we have done could have more documentation on what has been done, more integration and communication with the main team responsible, and we could have had a closer feel on the pulse of the PR. However, it is also important to put on realistic and healthy expectations for contributions, especially for such a huge organisation and repository, as otherwise a fix to a simple bug like this can become endless work just to end. By that standard, we can determine ourselves to be in the *“Closed”* state, rather than in a void of *“Concluded”* or else until the PR is closed in which we can check off this box as well.