The general idea of the deliverable is to enable automatic documentation generation (as best as possible) given code as input. The ML model should be able to observe any part of the file (function name, parameters, return value, exceptions raised, etc.) in order to document its containing members.

1. Dataset: data is mostly going to be scraped from open-source projects which have quality documentation. Projects available on Git repositories will be evaluated. Statically typed languages will likely be preferable to start the project due to their explicitly typed nature, however type hints or inferred type using compiler services on other languages should also be possible to implement.
2.
    a. All the documented code could potentially be useful because the model might observe word patterns related to code patterns, therefore keeping most of the information will be important. However, it would be very cumbersome to feed the files directly to the model, thus preprocessing could be done in order to separate code sections, such as function names, identifiers, type names, etc. In order to test the preprocessing, work should be done to get the system working for simple things, like isolated function without much context.
    b. The model should try to generate text as close as possible to manually hand-written documentation by developers. RNN models seem popular for text generation, but more investigation will need to be done in order to validate that this type of model will suit the needs of the project.
    c. As proposed in the first part of the deliverable instructions, the evaluation metric will be one for text generation and matching, like the BLEU score with brevity penalty.
    d. As this is a proof of concept, I suspect only one (or maybe two) languages will be supported, in very limited contexts and depending on the initial results. We can imagine a simple interface where one uploads a file containing code, this file is preprocessed like for the training data, is fed into the model, then documentation is either directly returned to the user or is integrated within the code of the uploaded file.

No specific metric goals have been identified at the moment, but certainly the criteria for success will be, in decreasing order of importance:

- Correctness (no misleading words/phrases)
- Readability
- Brevity
- Completeness (p.s. this would be very nice to have, but it is almost unthinkable to suspect that the model will be able to recognize intricate patterns in the actual source code which would normally be documented. E.g. a guard clause checks for a parameter to be within a certain range or be of a certain kind; completeness would imply detecting patterns of the sort. To be evaluated)