Project checklist

This document gives you a checklist of different things we check when grading your projects and hopefully clarifies expectations.

Data analysis and preprocessing

- Have you analysed the dataset? For example, can you answer the following questions:
    - What do the input data and expected targets contained in the dataset represent?
    - How is this data encoded?
    - Does the dataset contain missing / extreme values?
    - What does a typical input / output look like?
    - Is the dataset normalized?
    - Is the dataset balanced?
- Have you preprocessed the dataset according to your analysis of the dataset and domain knowledge?

Models and training

- Have you implemented / used different model architectures?
- Have you implemented / used a baseline model?
- Are they relevant for the data and task?
- Have you taken any measures to check the presence of underfitting/overfitting?
- Have you taken any measures to alleviate underfitting/overfitting?
- Have you made any quick sanity checks? For example, have you checked that:
    - the outputs are not constant
    - the losses are not constant
    - the validation loss isn't much better than the training loss.
    - the training loss is not increasing with respect to the number of epochs

Model selection and evaluation

- Is the dataset correctly split?
- Is the performance measure relevant for this task and dataset?
- Is the performance measure used correctly? (e.g accuracy has to be maximized but MSE has to be minimized.)
- Is the model selection done based on the validation performance of each model?
- Is the best model selected automatically in the code and not manually? (so there must be a argmin/argmax/min/max somewhere in the code...)
- Is the performance of the (automatically!) selected model assessed using the test dataset?
- Is each dataset used correctly? For example:
    - The preprocessing is based only on the training data
    - For a basic validation pipeline, the validation dataset is not used during the training, only to select the best model. For more complex pipelines such as cross-validation, each training/validation subset is created and used correctly.)
    - No test dataset during training in any case. Test data used only once, and only on the selected model.

Report

Comments on your approach and design choices include (but are not limited to):

- What was your approach to analyse the data you were given? Why? Are there any challenges that you should take into account with this dataset?
- Do all your preprocessing steps / sanity checks / performance measure / model architectures / different sets of hyperparameters tried, etc. match the data and the task you were given? Justify.
- Have you tried other approaches that were not successful and therefore not used in the final pipeline? Do you know what went wrong? Otherwise, what would be your best explanation? Have you attached the code of these failed attempts in your code?
- What was your approach to analyse your results? Why?

Comments on your implementation include (but are not limited to):

- Which libraries did you use (pandas? numpy? sklearn? bare lists/dictionaries? etc.)? Are they relevant for the data and the tasks you were given? Why?
- What type of programming did you use and why? (Object-oriented programming? Recursion? Split the problem into smaller functions?)
- Is your implementation efficient and "pythonic"?
- Did you feel limited by your programming skills? If so, why?

Reporting results

- Does the selected model along with its performance appear clearly and distinctly in your report? They should not be hidden somewhere in the middle of a paragraph.
- For each performance reported, have you clearly specified the metric used and on which dataset? (And have you specified how you split your dataset in the first place?)

Comments on your results include (but are not limited to):

- Does the selected model seem to underfit/overfit?
- Are your results surprising? In other words, relative to your general knowledge in machine learning, your understanding of the data and the problem, and relative to the confidence you put in your design choices and implementation, did you expect such performance?
- Are your results satisfying? In other words, is your model better than random outputs? Does your model outperform any baseline model? Do you think you could use your model in real life? If so, in which conditions?
- In which cases does your selected model seem to struggle? In which cases does your model seem to yield good results? How would you explain that?
- If you were given unlimited time, what would you try to improve your performance?
- Does the code run slower/faster than what you would expect? If so, what would be your best guess to explain why?

Figures

- Are there any relevant figures in your report? (E.g. about the data, your results, etc.)
- Are they correctly annotated? (E.g., title, axis labels, legend, etc. Are fonts large enough to be read easily? Do we know precisely on which dataset your figures were generated?)
- Do you explain what can be concluded from the figures?

Clarity of the report

- Is your report well structured or is it one big paragraph?
- Can the reader fully understand your report without having opened your code files beforehand?
- Can the reader find quickly the piece of information they wanted or is it necessary to read it through?
- Are you sure all snippets of code you have included in your report (if you have put any) are relevant in a report?

Code quality

- Is it easy to read your code? (Relevant variable/function/class names? Useful comments? Project appropriately divided into several chunks of code?)

    1. "Not really. To understand your code the reader had to be familiar with the project description, to know how to solve the task and to have good programming and python skills"
    2. "To some extent. To follow what you are doing, the reader needs to have had a quick glance at the project description first."
    3. "Yes, anyone with some basic python skills would understand your code, even if they were not given the project description."

- Is your code efficient?

    - Have you tried to use vectorized computations and/or list comprehensions depending on the situation to avoid for loops?
    - Do you compute useless things? Or the same thing several times instead of storing it somewhere?
    - Do you create variables that you never use?

- Is your code clear and consistent? (E.g., variable naming conventions are consistent throughout your project and your code doesn't look like a copy paste from different sources.)

- Is your code unnecessary long? (E.g., are there many similar blocks of code that could have been encapsulated into a function?)

Reproducibility

- Can the code corresponding to all results, figures, etc. in the report be found in the submitted files?
- If we re-run the code ourselves, would we get exactly the same results? (E.g., are all random processes seeded with a fixed seed?)
- Does running the cells of your jupyter notebook from top to bottom give the expected results? (E.g., a cell should not call a function that is defined only later on in the notebook)