Project Guidelines COMP 6321

Project evaluation

The 35% grade breaks down into the following deliverables, as in the syllabus:

- 4% project proposal explaining your project goals and plans (max 2 pages).
- 3% project presentation to be submitted with the report; the instructor will play these in final lecture. 3% originality to be assessed by the instructor, to reward groups that propose interesting projects.
- 15% project report describing your project data, methods, and conclusions (no more than 8 pages).
- 10% code and data to be submitted with the report; the instructor and TAs will inspect these files.
-

Project report guidance

Each project type has a `report.{tex,pdf}` template. This 2-column LaTeX\LaTeX{}LATEX template is based on the one for *CVPR*: the IEEE International Conference on Computer Vision and Pattern Recognition. CVPR is a top conference in computer vision and machine learning. You do not need to use the LaTeX\LaTeX{}LATEX template itself.

Guidance:

- **Word vs LaTeX.** You are encouraged to try writing their report in LaTeX\LaTeX{}LATEX, because then you'll see how scientific papers are written in the computer and engineering sciences. However, Word is OK too. Either way, there are options for collaborating on Word documents (Office 365; Google Docs + export) or on LaTeX\LaTeX{}LATEX documents (collaborative platforms like overleaf.com).
- **Length.** Aim for **3-4 pages** at a density that approximates the report template (font size, margins).
- **Clarity.** The most common failure mode of a project report is lack of clarity. Even if you are good at machine learning, describing the "what" and the "why" of what you did, and at the appropriate level of detail, is an essential skill.
- **Logical consistency.** The second-most common failure mode of a project is a disconnect between stated goal(s) and the experiments that were performed. For example, if your goal is classification, then saying "we used K-means" does not make logical sense on its own and would require further explanation.
- **Well-supported conclusions.** The third-most common failure mode of a project is when there is a disconnect between the conclusions versus what the experiments show. It is much better to depict an honest assessment of what you could / couldn't conclude, or

what you did / didn't succeed at, than to try to impress the instructor with broad sweeping claims that are not justified by your analysis. When your conclusions overreach or misrepresent, that indicates to the instructor (or, someday, to your boss) that your conclusions are not to be trusted.

- **Best practices.** Finally, if you demonstrate that at least some ML "best practices" were applied during your project, then this can strengthen your grade. For example, try to highlight things like:
    - Was the metric of success designed from a stakeholder point of view, *i.e.*, did it measure what users of the trained system would understand and care about?
    - Were imbalances in the data noted?
    - Was a feature preprocessing pipeline designed and applied consistently? Was its impact assessed?
    - Was the benefit over a simple baseline included, such as a dummy classifier / regressor?
    - Was a hyperparameter search conducted to give each method its "best chance"?
    - Was data splitting scheme used for a dataset (random, time split, grouped split, etc.) consistent with how a model would be expected to make predictions at test-time?
- **Supplementary pages.** You can include extra pages in your report, but your report should still 'conclude' within the page limit, because the instructor is not obliged to look at the extra pages before grading your work. If you do include supplementary pages, best to fill them with results that are easy to understand at a glance, like extra examples of data or predictions, or raw tables that were already summarized in the main text of the report.

Project presentation guidance

The purpose of the slide presentation is to give the instructor, and other students, a high-level preview of:

1. *What* goal you were trying to achieve.
2. The kind of *data* you were working with.
3. *How* you tried to achieve your goal.
4. To what degree you *did* or *did not* achieve your goal.

Guidance:

- **Length.** 2-3 minutes. should focus on conclusions and what their novelty component was.
- **Use slides.** You probably only need 4-6 slides total.
- **Multiple speakers is optional.** It would be nice if each group member would speak during the presentation, but it is not required.
- **Show the fun stuff.** You don't have to show all your results. Instead, just focus on the main points, and on getting the viewer excited to read your report.
- **Relax:** just make sure the material is clear, and be done with it.

Project code guidance

Some criteria / suggestions:

- **Acknowledge sources.** Using code from the internet as part of your project is not a problem. Using code from the internet *without acknowledging the source* is a major problem. The absolute worst thing you can do for you and your team is to paste chunks code from the internet into your project and then pretend that you wrote the code — that is a clear violation of academic integrity.
- **Avoid copy/paste.** If the TAs see that long blocks of code were copied and pasted over and over, across scripts or notebooks, without any attempt at modularizing that code for re-use, then you may lose marks.
- **Avoid hard-coded paths.** If your code has things like `open("/home/me/comp6321/data/foo.txt")` then there's no way it's going to run on another computer. Try to organize your data and configuration files so that your scripts can refer to the files using relative paths, rather than absolute paths.
- **Code organization.** The TA will assess whether the code was reasonably organized. This assessment is somewhat subjective, but you just have to live with that.
- **Code comments.** Try to comment important functions or important computations in the code.
- **Reproducibility is good.** Reproducible code is more likely to get a full grade. This includes simple universal best practices like setting the relevant random seeds (Numpy, PyTorch, TensorFlow) or, if code is intended to run on a local machine, exporting the conda/pip environment needed to run the code. However, if it is too late for your team do think about reproducibility, you can still get a good code grade.
- **Easy to run.** Even if your code is not fully scripted/automated, the fewer steps a TA would have to understand to run your code, and the clearer it is how to do those steps, the better.
- **Submit notebooks with results included.** In case a TA cannot run your code, it is better to submit your notebooks with the output cells (plots, images, output) included, so that the TA can at least see what the output was when *you* ran them.
- **Package small datasets with the project code.** Moodle accepts projects up to 250 MB in total size. If your compressed data set fits well within that limit, then it should be directly included in the submission.
- **Link to large data sets.** If your data is too large to include, you must provide a link (Dropbox, Google Drive, etc.) where the file's modification date is clearly visible. See "Submitting files."

Submission guidance

Only one group member should submit! If multiple group members submit, the instructor will pick only one version to grade, arbitrarily.

Citing source material in project reports

It's very important to cite sources of code, data, and even ideas. A citation can be a footnote with a URL, or it can be the author, title, and year of an article or textbook.

The LaTeX\LaTeX{}LATEX report template comes with a bibliography and examples of how to cite it. However, if you are unfamiliar with citations in LaTeX, you can ignore it and use your own citation convention. The template already includes the `hyperref` package, which provides the `\url{...}` command.

Tips for getting a good grade

- **Navigating uncertainty.** Uncertainty and anxiety is normal part of navigating a project with open-ended aspects. Your instructor and TAs will try to guide you with specific questions that you may have, but the questions have to come from you. Use office hours and TAs to resolve your uncertainties and give you confidence.
- **Apply practices from lecture and labs.** The general idea is to apply the techniques you learned in lecture. Try to be systematic in your analysis, for example ensuring the same hyperparameter search and training procedures are applied to all data sets. Obviously, something like "training on the testing data" is a huge no-no and would render your conclusions invalid. You are of course free to explore other techniques if they interest you or if you believe that they will provide insight or surprises to the analysis.
- **Aim for reasonably good predictive accuracy.** If you report that "model type *X* is the best because it it had an average test performance of 84% across the data sets" and yet most other groups found that model type *X* got an average test performance of 98%, then this may reflect some problem in the way that you either processed the data or trained models of type *X*. However, you will **not lose marks** just because a few other groups got better prediction accuracy than you on a particular dataset, or even just because some other group came to a different conclusion about what was "best." As long as your predictive accuracy is not indicative of a bug or weak attempt at modeling, do not worry about fine-tuning performance.
- **Reproducible results.** Always set the random seed or random state of whatever Python framework you're about to rely on, be it numpy, sklearn, or torch. Don't end up with a different train/test split the next time you run your training pipeline! At least not unintentionally!
- **Quality of scientific presentation** is important for this course. That means presenting figures, plots, and tables that are maximally informative for the scientific question being asked. For example, students should know when to use a scatter plot, a bar plot, a box plot, etc. Employers and professors often find that students and recent graduates lack these skills, so this is the best time to learn as much as possible! Use examples of plots and tables from labs or from your favorite research papers as a guide.
- **Quality of Python code** does have an impact on grade, though likely modest. Employers and professors find Python coding one of the most important skills that ML students and graduates lack.

- **Quality of writing** is somewhat important for this course. Apart from Python coding, scientific writing is the next thing that employers and professors wish that students and graduates were better at. This is a machine learning course, not a writing course, so poor writing will not ruin your grade, and minor grammatical errors will have no impact. But you are unlikely to get top grades if you have a report that is unclear or incoherent, even if your code itself is clean, concise, and systematic.
- **Methodological and code consistency across the project.** For the predictive accuracy component, avoid submitting a project where it's clear that each team member worked independently and did not share a common underlying methodology or codebase. Here are some examples where a team missed opportunities to consolidate their code/ideas:
    - One team member does classification, the other does regression, and they each use completely different methods and/or code for doing hyperparameter search.
    - Multiple versions of code to load data sets that are in the same format. If two data sets are in the same or similar format, they should be loaded by the same code.
    - The project code contains several scripts that must be run to reproduce results, but each script follows a different style for accepting arguments (input directories, output directors, etc.)
    - Different versions of code that generate what is essentially the same kind of plot.