Projects
Machine Learning Course
Fall 2017

EPFL
School of Computer and Communication Sciences
Martin Jaggi & Rüdiger Urbanke
mlo.epfl.ch/page-146520.html
epfmlcourse@gmail.com

In previous years, we have found that in particular the report quality (and also code quality) were sometimes not prioritized by some groups. Here is some advice based on common mistakes found in last years projects to help you prepare the report and submission.

Report Guidelines

The goal of the written report is to show what you did and that you understand what you are doing. This does not mean that you need to explain everything that was shown in class. You can assume that the reader knows what cross-validation or gradient descent are. However, the reader should be able to reproduce your work, your results and understand why you made particular choices. Your choice of method need to be **well motivated** and you need to **show evidence** that your work has an effect.

The simplest way to do so is to start with a simple model as a baseline, evaluate it, find a way to improve it, evaluate again and repeat. Explain the process that leads to your various improvements, evaluate the results carefully and present evidence using plots and tables.

Cross validate
Do not rely solely on the Kaggle submission system to get an error estimate. You need cross-validation to get an idea of the variance of your model.

Tune Hyperparameters
When comparing two models, make sure you tuned the hyper-parameters for both models beforehand. Comparing un-tuned / ill-defined models is not meaningful.

Explain your decisions
Do not throw a list of 10 methods you tried without explanation. We are more interested in your decision process than in what you tried.

Writing

Check, re-check and spellcheck
Proofread your report. Run a spellchecker. For each paragraph, have someone that did not write it check that the writing makes sense.

Write scientifically
You are not writing an essay, you are writing a scientific report. Be clear and concise. Avoid starting your report by "Since the beginning of the 20th century...". Try to structure your report like a scientific paper; Introduction, Method, Result, Conclusion.

Citation
Do not cite other people's work verbatim, but give correct references. Do not copy & paste Wikipedia, or any other source for that matter. Show that you understand your reference material by explaining it in your own words. Always give correct reference when using other people's work.

Plots
You are encouraged to use plots and tables to show your results and compare different models. It is often easier to convey an idea through a graphic than by writing. However, make sure that a plot is actually helpful. A bar graph with two bars should not be a plot. Make sure that your plots are understandable, have labeled axes, a title, a sensible scale and axes limits, a caption that provides enough information to

understand what is going on and what can be learned from it. Also, your plot should be readable in black & white, not only in color.

Code and README

README

The README file should not mirror the report. What you did and why goes in the report, and the report needs to be self-contained. How you did small technical details and how users can re-use your code goes into the README file. It should contain the full instructions on how to run your code, how to reproduce your obtained results, and give an overview of the architecture of your code (what are the different files and what they contain). For Project 2, you will need to explain what packages to install, how to install them and be precise about the version numbers.

Try your own dog food

Run your submission on another computer by following your instructions. You can borrow a MacBook at the Rolex Learning Center, run a VM, run a VM on vdi.epfl.ch, use one from the computer labs. This will ensure that your zip contains all the necessary files and that your instructions are clear.

Modularisation

Avoid copy pasting of code as much as possible. Use helper functions and separate your code into different files with appropriate names.

Style

Clear variable and function names are better than comments. Indent your code properly. Use an IDE to have a consistent format. Use Python Docstring convention to explain what a function does. Make multiple short functions with explicit names rather than a 200-lines `run` function. The more readable your code is, the more likely you are to be understood and given points.