Project 2018

Emerging Technologies

Due: last commit on or before November 30th

This document contains the instructions for Project 2018 for Emerging Technologies. The project involves writing code, comments and documentation predominantly in the programming language Python [2] and using the Jupyter notebook [5] software. You must use git [1] to track your work and you will submit your project by providing a URL to your git repository. It is suggested you use GitHub [3] for this purpose and that you consider making your repository publicly available so that prospective employers may view it. However, should you wish to, you may restrict general public access to your repository so long as you give permission to the lecturer to view it. Furthermore, any git repository URL to which you provide access to the lecturer will suffice – you don't have to use GitHub.

Please be advised that all students are bound by the Quality Assurance Framework [4] at GMIT which includes the Code of Student Conduct and the Policy on Plagiarism. The onus is on the student to ensure they do not, even inadvertently, break the rules. A clean and comprehensive git history is the best way to demonstrate to the examiner that your submission is your own work. Note, however, that it is expected that you draw on work that is not your own to build your submission and you should systematically reference this work to enhance your submission.

Submission content

Your submission will have five main components:

1. **numpy random notebook:** a jupyter notebook explaining the use of the numpy random package, including plots of the various distributions.
2. **Iris dataset notebook:** A jupyter notebook explaining the famour iris data set.

3. **MNIST dataset notebook:** a jupyter notebook explaining how to read the MNIST dataset efficiently in Python.
4. **Digit recognition script:** a Python script that takes an image file containing a handwritten digit and identifies the digit.
5. **Digit recognition notebook:** a jupyter notebook explaining how the above Python script works, and its performance.

Minimum Viable Project

The minimum standard for this project is a git repository containing the foure notebooks and one Python script listed above, along with a README, LICENSE and gitignore. Your README should clearly document how to run the script and notebooks in your repository. A better project will be well organised and contain detailed explanations. The architecture of the code and notebooks will be well conceived.

Submissions

git must be used to manage the development of the software and you must make your repository available to the lecturer by URL. Your git repository will form the main submission of the project. You must submit the URL of your git repository using the link on the course Moodle page before the deadline. You can do this at any time, as the last commit before the deadline will be used as your submission for this project.

Any submission that does not have a full and incremental git history with informative commit messages over the course of the project timeline will be accorded a proportionate mark. It is expected that your repository will have at least tens of commits, with each commit relating to a reasonably small unit of work. In the last week of term, or at any other time, you may be asked by the lecturer to explain the contents of your git repository. While it is encouraged that students will engage in peer learning, any unreferenced documentation and software that is contained in your submission must have been written by you. You can show this by having a long incremental commit history and by being able to explain your code.

Marking scheme

This assignment is worth 100% of your marks for this module. However, it is broken into three distinct parts: 40% for the first three Jupyter notebooks (on numpy, iris, and MNIST), 40% for the Python script and associated Jupyter notebook on recognising digits, and finally 20% for presentation of these ideas. The presentation will consist of both the way your work is presented in your repository and also a final one-on-one end of semester meeting with the lecturer. The following marking scheme will be used. Students should note, however, that in certain circumstances the examiner's overall impression of the project may influence marks in each individual component.

Notebooks	**Research**	10%	Investigation of problem and possible solutions.
	Development	10%	Clear architecture and well-written code.
	Consistency	10%	Good planning and pragmatic attitude to work.
	Documentation	10%	Detailed descriptions and explanations.
Digits	**Research**	10%	Investigation of problem and possible solutions.
	Development	10%	Clear architecture and well-written code.
	Consistency	10%	Good planning and pragmatic attitude to work.
	Documentation	10%	Detailed descriptions and explanations.
Presentation	**Clarity**	20%	Investigation of problem and possible solutions.

Advice for students

- Your git log history should be extensive. A reasonable unit of work for a single commit is a small function, or a handful of comments, or a small change that fixes a bug. If you are well organised you will find it easier to determine the size of a reasonable commit, and it will show in your git history.
- Using information, code and data from outside sources is sometimes acceptable – so long as it is licensed to permit this, you clearly reference

the source, and the overall project is substantially your own work. Using a source that does not meet these three conditions could jeopardise your mark.

- You must be able to explain your project during it, and after it. Bear this in mind when you are writing your README. If you had trouble understanding something in the first place, you will likely have trouble explaining it a couple of weeks later. Write a short explanation of it in your README, so that you can jog your memory later.

- Everyone is susceptible to procrastination and disorganisation. You are expected to be aware of this and take reasonable measures to avoid them. The best way to do this is to draw up an initial straight-forward project plan and keep it updated. You can show the examiner that you have done this in several ways. The easiest is to summarise the project plan in your README. Another way is to use a to-do list like GitHub Issues.

- Students have problems with projects from time to time. Some of these are unavoidable, such as external factors relating to family issues or illness. In such cases allowances can sometimes be made. Other problems are preventable, such as missing the submission deadline because you are having internet connectivity issues five minutes before it. Students should be able to show that up until an issue arose they had completed a reasonable and proportionate amount of work, and took reasonable steps to avoid preventable issues.

- Go easy on yourself – this is one project in one module. It will not define you or your life. A higher overall course mark should not be determined by a single project, but rather your performance in all your work in all your modules. Here, you are just trying to demonstrate to yourself, to the examiners, and to prospective future employers, that you can take a reasonably straight-forward problem and solve it within a few weeks.

References

[1] Software Freedom Conservancy. Git.
`https://git-scm.com/`.

[2] Python Software Foundation. Welcome to python.org.
`https://www.python.org/`.

[3] Inc. GitHub. Github.
https://github.com/.

[4] GMIT. Quality assurance framework.
https://www.gmit.ie/general/quality-assurance-framework.

[5] Project Jupyter. Project jupyter home.
http://jupyter.org/.