HPDM139: Coding Assignment 2

Health Data Science Analysis Package

September 20, 2022

1 Overview of assignment

In this assignment you will work in a group (size 2 or 3) to design and implement a python package of your choice. Students are expected to self-organise and agree the sharing of the workload between themselves.

Students can assume the package they develop will be used by a fellow health data scientist. You will employ best practice in the package's design, implementation and deployment. The specific application area of the package can be chosen by the group, but it must be applicable with the general area of health and medicine. The package might for example:

- Provide high-level visualisation tools for a certain type or class of health data;
- Provide a comprehensive analysis framework for a specific health data set;
- Provide a set of modelling tools or algorithms that a health service might use to answer a specific set of questions.

1.1 Group Assignment Submission.

Assignments must be handed in before 2pm on 11th January 2023. This deadline is **strict**.

The procedure for submitting group work to eBART is as follows:

- **Only one student per group should submit the code, user guide etc**. Nominate someone in your group to do this and make sure you arrange a time, before the deadline, to confirm that this has been done. If possible do this together (or via a shared screen in Teams) so you can all check that the submission is correct, it has been uploaded to the correct link, and that the file hasn't been corrupted.
- The remaining members of your group **must** submit a PDF with the names of their group members on it.
- Please submit a **single** .zip file to Bart containing your python source code and associated files.

- Your code **should be runnable**, PEP8 compliant, carefully organised, fully tested and free from bugs/errors.
- Please provide instructions to run your code. This should be submitted in markdown (.md) or in a Juypter notebook (.ipynb).
- A marker must be able to recreate the python software environment you have used. For example:
  - A .yml file specifying conda and pip installs (preferred).
  - A list of installed packages and version numbers

1.2 Advice on the choice of topic

The topic / application that your python package will support is chosen by you and your group. Groups are expected to do their own research to choose a topic and features to include in the package. It is strongly recommended that you think about the choice of topic early in the module. You are free to discuss the choice of topic with the module lead. **The chosen topic does not need to be complex.** Example packages include:

- A toolkit to download, wrangle, visualise, and/or combine health data. For example, one or more data sets from NHS England Statistics (`https://www.england.nhs.uk/statistics/`) or Covid-19 data from Zenodo (`https://zenodo.org/communities/covid-19`)
- A toolkit that implements some mathematical or statistical equations/algorithms that a user can parameterise and run. For example, basic queuing equations that the NHS can use to analyse waiting time for a service given different levels of demand or capacity. (`https://en.wikipedia.org/wiki/Queueing_theory`)
- A toolkit to support machine learning model selection (for example, models found in frameworks such as sklearn and Tensorflow) when applied to a specific health dataset. For example, offering different types of preprocessing and model comparison tools.
- A toolkit to explore potential bias and unfairness in the results of classification problems. For example, analysis of results by features such as gender, ethnicity and age.

One option students may wish to explore is to think about their learning in other MSc modules. Look for opportunities to write a high level python package that makes the analysis you are doing cleaner, easier for yourself and others, and more efficient (less code, that is more readable when it is reused or faster execution).

For projects looking to use real datasets please use public and open datasets only.

Students may choose a topic that is applicable more widely than health. However, it should be demonstrated by application to a health or medical problem.

1.3 Assessment

A detailed assessment scheme is provided at the end of this document. In summary there are five criteria.

- **Design** and organisation of the package.
- **Appropriateness and comprehensiveness** of package and its features for topic.
- **Usability** of the package including how it is deployed and documented.
- **Coding standards** including appropriate use of Python features, organisation and PEP8 compliance.
- **Code execution** including ability to recreate the software environment, testing and ability to run without errors.

1.4 Documentation / User guide

You must provide documentation with your package that both describes the need for it, highlights its functionality and provides a user guide. One way to do this is a Jupyter Notebook.

2 Assignment Tips

- Keep things simple. Your group should carefully design features and functionality together before dividing up work and coding.
- Break the development of your package, userguide and documentation up into milestones (or versions). For example, if you plan to implement 10 features then prioritise 3-5 of them for an initial milestone and work towards that as a group.
- Explore other packages on the SciPy stack and PyPi and note how they organise their code and documentation. Try to learn 1-3 things from their best practice and incorporate it into your own work to improve it.
- If you think of a new feature discuss it with the rest of the group to debate and/or refine the idea before implementing it.
- It is recommended that you use git (and possibly GitHub or GitLab) to collaborate and manage this project. If you use GitHub/Lab you must

keep your repo, and hence work, private to your group until after the submission deadline. **Note that GitHub only supports private repositories for 3 collaborators. If you have a group of four you will need to use GitLab.**

- Test your code. In particular:
  - Write some tests to check that the framework is not broken after a new feature, function, module, clean-up, documentation improvement or change is added.
  - Test one final time that everything works in advance of submission.
- Please make sure everyone in your group is using the same version of python, pandas, numpy etc. See the module notes on conda environments for help.
- Coding standards count. Write good quality code that is easy to read and maintain. Follow PEP8 guidance on line length, variable naming and docstrings.

3 Marking Scheme

Fail: 0-49

- Absent or limited evidence of appropriate design of a python analysis package.
- Poor rational for topic and the package has a very limited range features and tools to support users in the chosen topic area.
- Unintuitive package interface design; absent or limited user guide, containing a large number of grammatical and typographical errors; code is not in a state where it is deployable.
- Poorly documented and formatted code that does not attempt to follow PEP8 guidelines; limited use of good practice python.
- Code may contain run time errors, may not run at all, contain no evidence of testing, and does not provide a reproducible software environment for data scientists.

Pass: 50-59

- Limited but mostly appropriate design of a python analysis package. The code may not meet all of the requirements for a python package.
- Limited rationale provided for topic and the package has a limited but passable range features and tools to support users in the chosen topic area.

- Borderline passable python interface to package and user guide. The user guide contains multiple grammatical and typographical errors; The code could be deployed, but there is no use of basic or advanced python deployment options to improve user experience.

- Demonstration of limited understanding of how to organise code and follow best practice; however, the code does not follow PEP8 guidelines. The readability and maintainability of the code could be improved by adopting best practices.

- Code mostly reproduces results, but contains some minor runtime errors and minor bugs. No or very limited evidence of testing. No manual or automated procedures are available for other data scientists to recreate the software environment.

Merit 60-69

- Generally appropriate design that meets the majority of the basic requirements for a python package.

- Generally well argued rationale for topic and the package and a good selection of features and tools to support users in the chosen topic area.

- Generally well thought out python interface to package and user guide. User guide has shortcomings, but is generally useful and may contain grammatical and typographical errors; evidence of ability to use basic methods to deploy python packages, such as a pip install of a local python package, to enhance user experience.

- Generally well presented code that attempts to follow PEP8 guidelines. Code has limited documentation and demonstrates some evidence of following best practice for organisation and reuse. The readability and maintainability of the code could be further improved.

- Code reproduces reported results and contains no run time errors, may contain minor bugs and includes evidence of testing. It is possible for other data scientists to manually recreate the software development environment by following a set of installation instructions.

Distinction 70-85

- Appropriate design that meets all of the basic requirements for a python package. Advanced package features may have been implemented to limited success.

- Features and tools are comprehensive for the chosen topic.

- The package ships with a well presented and functional user programming interface. User guide is well presented and helpful to readers, but may not highlight all functionality. The guide has minimal grammatical and typographical errors. Evidence of ability to use advanced methods such as PyPi to deploy python packages to enhance user experience.

- Code makes use of best practices for organisation and reuse. Evidence that design takes account of code maintainability and extension.
- Code reproduces results, contains no runtime errors, minimal bugs and strong evidence of testing. It is possible for other data scientists to recreate the software development environment using a conda environment file.

Distinction++ 86-100

- A professional design of a python analysis package with intuitive organisation.
- An internationally or nationally important topic and a package offering an unrivalled set of features and tools.
- The package ships with a intuitive, but simple python interface and framework. The user guide is succinct highlighting all features of the package. Excellent English is used throughout containing very limited, if any, grammatical and typographical errors. Deployment offers the highest levels of user experience with guarantees of working across multiple operating systems.
- Flawless publication quality code that closely follows PEP8 guidelines for style and documentation. The code is easy to read and maintain. The code makes use of advanced concepts from Python where appropriate.
- The code reproduces the results exactly with no runtime errors, no bugs and may contain advanced testing strategies and tools. The package ships with a conda environment installing only what is necessary.