University of Exeter Medical School

Assignment Brief - Coursework 2

Module Title:	Coding for ML & DS	**Module Convenor:**	Thomas Monks
Module Code:	HPDM139	**Summative/Formative**	Summative
Module Credits:	15	**Assignment Type:**	Group
% of Module Mark:	50%	**Date Set:**	07/11/25

Date Due:	**07/01/26**		

1 Required Task

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. **As part of the assignment evidence must be submitted of at least 3 group meetings**.

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 Assignment Components

Assignments must be handed in before 2pm on 8th January 2025. Please submit a **single** .zip file to ELE containing the following:

1. A Python package:

- An appropriately structured local or installable Python package that can be used as part of a health data science project of workflow;
- A **README** file containing an overview of your submission and all files included.
- **Instructions** to install and run your code. This should be a short file submitted in markdown (.md) or in a Juypter notebook (.ipynb).
- **Documentation** and a tutorial/user guide for your package.
- Details of any **package and code testing**.
- **Dependency management:** 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

2. Evidence of group working:

   - As this is a group assignment, we want to see evidence that the group operated in a manner akin to what would be expected in a professional environment. Therefore we are requiring that you submit a portfolio of documents to showcase this.
     - Meeting agendas (x3)
     - Meeting minutes; including attendees and actions assigned to individuals (x3)
     - Action plan with milestones

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. 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 Criteria

Marking is split into two components.

1.3.1 Evidence of group working (10%)

Please note that the evidence of group working will not be marked for content but compliance only. Therefore, if you submit all the documents and they are fit for purpose you will receive the full 10% for this component.

1.3.2 Health data science package (90%)

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

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

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 repository, and hence work, private to your group.

- 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.
- 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. Don't underestimate this work. Plan in time to generate it.

3 Marking Scheme

Design and organisation of package

- **0 - 49**: Absent or limited evidence of appropriate design of a python analysis package.
- **50 - 59**: Limited but mostly appropriate design of a python analysis package. The code and the structure may not meet all of the requirements for a functional python package.
- **60 - 69**: Generally appropriate design that meets the majority of the basic requirements for a python package.
- **70 - 85**: Appropriate design that meets all of the basic requirements for a python package for the topic chosen. Advanced package features may have been implemented to limited success.
- **86 - 100**: A professional design of a python analysis package with intuitive organisation.

Appropriateness and comprehensiveness of package

- **0 - 49**: Poor rational for topic and the package has a very limited range features and tools to support users in the chosen topic area.
- **50 - 59**: Limited rationale provided for topic and the package has a limited but passable range of features and tools to support users in the chosen topic area.
- **60 - 69**: 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.
- **70 - 85**: Features and tools are comprehensive for the chosen topic.
- **86 - 100**: An internationally or nationally important topic and a package offering an unrivaled set of features and tools.

Package usability and user guide

- **0 - 49**: 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.

- **50 - 59**: 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.
- **60 - 69**: 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.
- **70 - 85**: 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 a remote install from GitHub or TestPyPI to deploy python packages to enhance user experience.
- **86 - 100**: The package ships with an 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.

Coding standards

- **0 - 49**: Poorly organised code that is hard to read, understand, maintain and debug. The code does not follow PEP8 standards and has absent or poor quality documentation.
- **50 - 59**: Code is passable, but does not make use of standard organisation structures such as reusable functions. The code may not follow PEP8 guidelines or PEP257 guidelines for docstrings. Minimal documentation is provided overall.
- **60 - 69**: The code is reasonably well organised and makes good use of Jupyter notebook (or equivalent) functionality to support readability and documentation. There is evidence that an attempt to follow PEP8 and PEP257 guidelines has been made. Documentation is good, but with fails to meet the highest standards.
- **70 - 84**: The code is elegantly organised and is relatively simple to reuse and maintain. Documentation is provided to a high standard making excellent use of Jupyter notebook (or equivalent) functionality. Code is presented to PEP8 standards although there may be some inconsistency. PEP257 docstrings are provided, but may not meet the highest standards seen in industry.
- **85+**: Industry standard code that is effortless to read and easy to reuse and maintain. The organisation of code makes outstanding and creative use use of Jupyter notebooks (or equivalent) in order to increase accessibility. Code is provided to a consistent PEP8 and standard across the notebook. Docstrings are PEP257 compliant and meaningful adding to the reusability of the work.

Code execution

- **0 - 49**: 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.

- **50 - 59**: Code reproduces some results, but contains some minor runtime errors and minor bugs. Absent or very limited evidence of testing. No manual or automated procedures are available for other data scientists to recreate the software environment.
- **60 - 69**: 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.
- **70 - 85**: 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.
- **86 - 100**: The code reproduces the results exactly with no runtime errors, no bugs and may employ advanced testing strategies and tools. The package ships with a cross operating system conda environment for development: only installing the necessary packages.

4 Submission

Your assignment must be submitted to ELE by the stated deadline. For instructions on how to do this, please see the please see the Online Submission Student Handbook (`https://www.exeter.ac.uk/students/infopoints/yourinfopointservices/assessments/`).

If you submit your work late, but are within one hour of the deadline, a penalty of 5% will be deducted from your mark. For any work submitted more than an hour late your mark will be capped at 50% (up to 24hrs after the original deadline; after which a score of zero will be awarded). This is unless you have valid mitigation (please visit the Mitigation SharePoint page for more information.)

Please ensure that your name DOES NOT appear anywhere in the document, but that your personal ID number is included. You should submit your work as a **.zip** file. The naming of your file must follow the format below:

- `ID_number_ModuleCode_assessment_name_AC_year`
- E.g. `91000000_CSC3019_written_review_2025-26`

Please note that submitting an incorrect file, or submitting the correct file to the wrong location, will not be considered as grounds for mitigation. Late penalties will be applied if you submit the correct file to the right location after the submission deadline.

In submitting this you are declaring the work is your own independent piece of work, not produced in collusion with a fellow student or plagiarized from a fellow student, or a web site, or a textbook, or any other information source. You must demonstrate good referencing practise and ensure you have sufficiently paraphrased all sources of information. The direct copying of AI-generated content is also included under plagiarism, misrepresentation and contract cheating under definitions and offences in TQA Manual Chapter 12.3. During the submission process you will be asked to check a box, that confirms you have adhered to the university's academic conduct policy and the expectations on use of GenAI in your assessment brief. Failure to do so may result in being referred to an academic misconduct panel which has the power to grant penalties based on specific criteria. For more details, please revisit the Academic Honesty and Plagiarism information on ELE.

5 Use of AI in Assessment

This assessment falls under the category of AI-Assisted in the University's Guidance on use of GenAI in Assessment. This is because using AI tools in specific ways can assist with your learning and will not inhibit fair assessment of your achievement of the module's intended learning outcomes. This means: You may use GenAI tools ethically and responsibly to assist in the development of an assessment in accordance with the boxes checked below.

- ☑ To develop ideas
- ☑ To assist with research or information gathering
- ☑ To help you understand key concepts and theories
- ☑ To identify trends and themes as part of data analysis
- ☑ To provide feedback on a draft
- ☑ To improve the plan or structure of my assessment
- ☑ To generate images, figures or diagrams
- ☑ To proofread and correct spelling or grammar errors
- ☑ To format citations or references
- ☐ To translate material into English in line with guidance from your module lead. Unless this box is checked, you must never use AI translate more than a word or short phrase into English.
- ☑ OTHER: To support debugging of Python code you have written.
- ☑ OTHER: To improve the quality of Python docstrings
- ☑ OTHER: To improve code quality
- ☐ OTHER: To write the Python code from scratch

When writing your assessment, you must never use AI tools:

1. For uses other than those represented by checked boxes in the list above.
2. To translate more than a word or short phrase into English unless agreed above.
3. To upload sensitive or identifying material to an AI tool
4. To present material that has been generated by AI as your own work or the work of someone else.

When submitting your assessment, you must:

1. Check the box during the submission process, that confirms you have adhered to the university's academic conduct policy and the expectations on use of GenAI in your assessment brief.

2. Treat the AI tool like a citation from any other source.
3. Include a list of all AI prompts and hyperlinks to their output with your references, at the end of your work. You do not need to include the outputs themselves, just the links.
4. Retain the full outputs generated by the prompts you have used during your assignment. These outputs should be accessible at the hyperlinks which you have submitted with your assignment. You may be asked to produce this material in the event of an academic conduct inquiry.

6 Acquired skills

This assignment provides you with the skills for real world health data science projects that are built for teams of researchers and other users. You will apply the technical skills learnt in the module to build a high quality reusable Python package that provides functionality that could include analysis of real data, support various types of machine learning/modelling project, or provide reproducible analysis pipelines for researchers or industry professionals.

- Analyse and manipulate complex data sets in health and demonstrate competence in building statistical and computational models to work with them in python.
- Apply a wide range of supervised machine learning algorithms to model outcomes in complex datasets.
- Critically appraise data science problems and evaluate the tools that are needed to solve them.
- Use a wide range of python tools including modern data science tools to conduct quantitative analyses.
- Explain and demonstrate the steps to follow in a reproducible scientific work flow used modern data science tools.
- Explain the importance of coding for high quality data science and machine learning research.

7 Support for Students with English as a second language

International students and those who have English as a second language can make use of the English Language Skills Development programme. This programme provides face-to-face workshops and courses as well as one-to-one assignment writing tutorials and an extensive range of online resources via their Guided Independent Learning site. Find out more by browsing the timetables: `https://www.exeter.ac.uk/into/englishlanguage/howtoparticipate/timetablessupport/` or emailing insessional@exeter.ac.uk