Projects

The projects will be available from week 3 and are designed to be extended pieces of work produced largely independently.

Research degrees (PhD or Masters)

For those working on research degrees (PhD or Masters) you are welcome to choose any topic you like, but we would suggest picking something that may be helpful for your research. Please use the instructions for the taught masters project as a guide for what is required. The restriction around machine learning for the taught masters project students does not apply. Please also use the template for submission to assist us in giving you useful feedback.

Taught Masters

For those on a taught master's courses we provide 3 "starting points". Each project contains some basic starter code, a bit of background information and some templates for the submission. Choose a project which piques your interest. Whilst each project provides access to an interesting source of data it is up to you to decide how to develop the project. We suggest coming up with a "scenario" (see individual projects for a few suggestions / examples). With that scenario in mind, we'd like you to think about how you could create a tool that would enable someone to easily interact with the data and extract something meaningful to them. Perhaps a way to visualise and interact with the data graphically or a set of functions and classes with a demo usage, that simplify extracting meaning from the data. **For students doing the taught masters course please do not use machine learning in your project.** Try to design your code so that it could be easily reused or adapted for broader related purposes beyond the one outlined in your scenario.

Project Code Guidelines

All the starter materials for the projects are supplied as an assignment, which you can synchronise in the same way you do with the exercises each week. If you use multiple computers please be careful to synchronise your files at the start and end of each session. **Please delete the files associated with the projects you decide not to use.**

We would prefer submissions as a Jupyter Notebook (*.ipynb). A template Jupyter notebook is provided that you can use. This should not only provide code but be structured with Markdown cells which explain:

- The aim of the project / scenario and the problem it is solving
- A high level overview in words of how the code works
- A discussion of design decisions taken in structuring the code.
- How you would develop / improve the code if given more time?

However, we are aware that some code eg guis, does not work well in Jupyter notebooks. If this is the case:

- you should provide a `main.py` file which runs your code.

- include any tests in a separate .py file.
- Provide a pdf which covers the points above. As a guide we'd expect ~ 300-500 words.

In both formats, additional code / modules can also be included in other python files in the same repository if necessary and then imported. **Make sure you use relative imports in your code.**

****Please make every effort to structure your code in a logical fashion to assist us in understanding it****

In addition to the Jupyter Notebook / code files you should also submit:

- a Conda environment file which enables your code to be run. Please ensure you test this on a Windows machine.
- any output that cannot easily be embedded in the notebook (e.g an example data file or some visualisation, pdfs)

Your program should (as a rough guide)…

- do something meaningful within the context of your scenario
- define at least two user functions (but typically more)
- make use of appropriate specialist modules
- comprise >~ 50 lines of actual code (excluding comments, imports and other 'boilerplate')
- contain no more than 1000 lines in total (if you have written more, please isolate an individual element)

<u>Referencing sources</u>

No one builds their computer programmes from scratch. At the very least pre-written compilers/interpreters turn high-level code into machine-level instructions. Typically we use many built-in functions, modules and packages. However, it should always be clear which modules you have imported. The assumption will be that the contents of such modules do not constitute part of your submission, unless their code is included in your submission.

Furthermore, it is common practice to refer to code provided in documentation, examples, Q&A sites, and open source repositories, in the course of developing one's own code. If you wish to directly include code from another source, you must clearly indicate its origin. Preferably, such code should be put it in a separate file, and imported into your own code in the same manner as any other standard-library or third-party module.

It is best practice to draw from multiple resources to instruct the creation of your own original code. However, where you do copy or adapt significant portions of code from a specific source and incorporate it in your own work, you should make it explicit that you have done so, and indicate where the original code came from, by commenting the code in an obvious manner.

Appropriate use of third-party code, eg specialist modules is encouraged and recognised positively in marking. However, we expect such material to be combined with significant amounts of your own code, to produce an original solution. We will also be looking for evidence that you understand your entire code at reasonable level.

Note that we actively attempt to detect where code has been taken verbatim, or with trivial changes, from online sources or other student's work and used without declaration.

Academic integrity statement:

Each file of code you submit must contain the following statement as a comment at the top of the file:

> *I, [your name], have read and understood the School's Academic Integrity Policy, as well as guidance relating to this module, and confirm that this submission complies with the policy. The content of this file is my own original work, with any significant material copied or adapted from other sources clearly indicated and attributed.*

Assessment

The assessment for MSc students will consist of two parts:

1. a short viva ~ 5-10 mins (20%)
2. the project submission (45%)

The short viva: (5-10 mins) with members of staff will be held during the 8th week of the course (week commencing 27th Nov). It will concern your project as it currently is (we understand you will not have finished it at this point, but please ensure an up-to-date version of the project is available at the end of the previous week). The only preparation necessary for this viva is to ensure you are familiar with your project. The viva will take the form of a code review. You'll be asked questions about your code, design decisions and alternative approaches you might have taken.

The project submission: projects must be submitted in their final form by 5pm on Wednesday 13th December. In assessing the projects we will be looking at the following features:

- Does it show a good grasp of python?
- Does the code tackle the stated scenario well?
- Is there evidence of self-driven learning (e.g incorporating relevant new modules)?
- Does it demonstrate a novel approach to the stated problem?
- Is the code "clean" and appropriately documented and therefore easy to follow?
- Does it evidence good design decisions in the way it is structured?
- Does it have a set of appropriate tests?
- Does the description in the accompanying markdown / report indicate an ability to explain what is being done and why?

Whilst we ask you to come up with a scenario for your project, we are not assessing this directly, and don't want you to be too concerned about what you choose. Any sensible

scenario that provides a direction / goal for your project and a non-trivial problem to solve is acceptable. We ultimately would like you to pick something that you think would be fun and interesting to work on, and fires your imagination. Hopefully reading through the examples next to each project will demonstrate what we mean, but if you are still concerned please discuss this with us early on.

Viva – 20% of module grade

80+% They can articulate both the way in which their code works and the reasons for choosing such an approach with superb clarity. When asked questions about their code they are adept in explaining the particulars and logic of small sections and the overview high-level design decisions made. They can extremely clearly reason about alternative approaches they may have taken, other projects that could be solved and how, and outline the pros and cons.

60+% They able to explain how their code works, and the reasons for choosing such an approach with in a comprehensible way (though follow up questioning might be required to draw this out). When asked questions about their code they can explain logic of small sections and the why the code is organised in a particular way. With some prompting, they can reason about alternative approaches they may have taken, outline the pros and cons or suggest how their code might be applied to a related problem.

40+% It is hard to understand from the explanations given how the code works at either small or larger scale. It is not always obvious that the student understands it well. They struggle to suggest how things might have been done differently or the code applied to a different but related problem.

Code submission - 45% of module grade

80+%: Excellent code, which addresses the (sufficiently meaningful) problem identified in the scenario in a readable, efficient and elegant manner, making excellent use of appropriate Python language elements and modules. If the implementation is incomplete, this is clear.

60+%: Good code, which mostly addresses the (sufficiently meaningful) problem identified in the report in a fairly readable and efficient manner, making use of appropriate Python language elements and modules. It is reasonably clear where the implementation is incomplete.

40+%: Poor code, which barely addresses the (possibly rather simplistic) problem identified in the report, in a manner that is difficult to comprehend and possibly very inefficient. Poor use of appropriate Python language elements and modules.

Each project has an associated Jupyter notebook that contains the following discussion as well as some starter code to get you up and running.

Project 1: Beating the weather

"Visual Crossing" provides free access to historical weather data globally and also forecasts for up to 15 days in the future (https://www.visualcrossing.com/weather-api). To use its API, you'll need to sign up for a free account (https://www.visualcrossing.com/sign-up). You can explore the data available through their website (https://www.visualcrossing.com/weather/weather-data-services). To use python you will need an api key. On the webpage above click "View Data" and then "API". Add the location Nottingham. You should see a url in the main window below that looks like this:

https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timeline/Nottingham?unitGroup=metric&key=YOURAPIKEY&contentType=json

Copy the string of letters and numbers that appear in place of YOURAPIKEY into the example json file. Once updated you need to move this file somewhere safe outside of your project and create a path to it inside your project.

Example scenarios:

1) You are approached by the owner of a hot air balloon company. They are looking to expand by adding a new location for hot air balloon flights. Hot air balloon flights can only take place when the wind is below 7mph and the weather is fine (ie sunny or cloudy but very little chance of rain). It needs to be in the UK but it also needs to be somewhere beautiful, perhaps a national park or area of outstanding natural beauty. If they can give you a list of places, can you work out where, based on historical data is the best place (ie most number of days they will be able to fly per year). Perhaps you could even help them visualise number of estimated days flying on a contour map of the UK. By the way his brother owns a windsurfing business in Spain, where strong winds are good and weak winds are bad.
2) A sports car enthusiast has an open top sports car (ie with no roof). Often on a Saturday he likes to go for a big drive round the country maybe 200miles. The average speed might be 50mph. The weather in Britain can be quite local so it might be raining to the West and then dry to the East. The weather also moves. I'm wondering if given a known route, a start time and a known speed it would be possible to work out if they are likely to get wet? Even better would be to adjust the start time, or choose between two different routes to maximise the chances of staying dry.
3) Looking at the underlying causes of changes in the weather has been constantly in the news due to the emphasis on climate change. The Visual Crossing weather api only has 50 years of historical data so unlikely (?) we could study this but there are other interesting patterns. Sunspots are small dark regions of the Sun. There is a known cycle ~12 years in the number of visible sunspots. You can download csv datafiles of the number of sunspots (https://www.sidc.be/SILSO/datafiles). Does this have any effect on our Weather?

Project 2: Investing in stocks

In this project our starting point will be the data about different stocks available via yahoo finance (https://uk.finance.yahoo.com/). Using this huge database we can get live information and historical stock prices for a huge number of companies. For example, look at this page with Amazon's stock price (https://uk.finance.yahoo.com/quote/AMZN?p=AMZN&.tsrc=fin-srch).

The starter code shows you a very simple example, using a python library to extract this information for Amazon between two time points.

Think about some of the factors that affect stock markets:

- Stocks vary individually but there are also global trends. If the market is going up or down, most stocks will follow. You could try to identify whether the increases or decreases in a stock are due to the market or the stock itself.
- Stocks are often correlated with other factors. If the price of oil goes up, the price of airline stocks will go down. You could try to identify which stocks are correlated with which factors.
- Stocks fluctuate in value over time. Sometimes the fluctuation is significant and it means you should buy or sell fast. Sometimes it is insignificant and you should hold. You could try to identify whether the fluctuation is significant or not.
- Visualising the historical trends in stocks can be useful. You could try to make it easy to visually compare the historical trends of different stocks, perhaps annotating graphs in a particular way when some of the events above occur.

Some example scenarios:

1) A business woman has a portfolio of 10 different stocks. She is busy running her own business and doesn't have time to monitor the stock markets like some hedge fund managers do. What she wants is some code to monitor her stocks and alert her if there are "significant" changes in any of the share prices. You might want to think quite carefully about what significant means and how you'd work that out from the data. How are you going to alert her?
2) Modern historians are interested in understanding the effect of big global events on different types of industries. For example, COVID resulted in devastation for airlines but was a boom time for those selling conference software like Zoom. How could you make it easier for them to find patterns and trends?

Project 3: Analysing the stars

The Sloan Digital Sky Survey is a massive astronomy project that has been collecting images and optical spectra of objects in the night sky. It is used by researchers, but all the data is publicly available. One can deduce certain things from the spectra or associated information about the objects. For example, one can determine the redshift of a galaxy from its

spectrum. This is a measure of how fast the galaxy is moving away from us. The spectra can also be used to determine the chemical composition of stars and galaxies.

Project page : https://www.sdss3.org/dr14/

In order to come up with project scenarios we highly recommend reading through a few of the basic, advanced and research challenge sections on this page of tutorials:

https://skyserver.sdss.org/dr10/en/proj/advanced/advancedhome.aspx

You don't necessarily need to solve these challenges, but they will give you some basic background and a sense of what kind of project might make sense.

A challenge of working with this Astronomy dataset is the sheer size of it. There are hundreds of millions of objects in the database and so one cannot just blindly ask the server to send you images or spectra for all of them! Stars or Galaxies can also be given catchy names like NGC 5406 which are not things you are likely to know! Once you've come up with a scenario we suggest using the following search tool:

https://skyserver.sdss.org/dr14/en/tools/search/form/searchform.aspx

to narrow down a list of objects that you want to work with. You should be aware that only a small fraction of the objects for which pictures exist have spectra (it is still a lot) but you can filter with this tool. You could download a manageable number of object ids or coordinates to a small csv, but be sure to include it with your project files so that we can reproduce your results.

The coordinate system for identifying where objects in the sky frequently refers to *ra* and *dec.* This can be a bit confusing. See this page for a simple explanation:

https://solarsystem.nasa.gov/basics/chapter2-2/

Example Scenarios:

1) An astronomer is interested in studying different types of astronomical objects. The optical spectrum of a quasar looks different to that of a galaxy or star. Young stars are very rich in elemental Hydrogen. Could you write a program that could classify objects based on their spectra? Perhaps given a list of objects it could return images of the objects suspected to be quasars or perhaps given a picture of a small bit of the night sky the astronomer could click on different objects and a label would appear telling them what type of object it is.

2) Build a star thermometer. Although there are thousands of spectra the SDSS survey has a much larger collection of images taken through a number of different filters. These filters are designed to only let certain wavelengths of light through. The u filter lets through ultraviolet light, the g filter lets through green light, the r filter lets through red light etc. The temperature of a star determines how much light it emits at different wavelengths. A hot star will emit more light in the ultraviolet than a cooler star. A cooler star will emit more light in the infrared than a hotter star. If you look at the ratio of the brightness of a star in the u filter to the brightness in the z filter you can get a good estimate of the temperature of the star. A fair bit more background is explained in this tutorial

https://skyserver.sdss.org/dr10/en/proj/basic/color/colorandtemp.aspx

Star spectra are in many cases approximated quite well by a blackbody spectrum with a known shape that depends on the Temperature.

https://en.wikipedia.org/wiki/Black-body_radiation

Can you write a tool that when given a star's object id will estimate the temperature of the star?