Statistical Learning

Due Sunday, May 12 on Moodle

Homework-01 | Exercise 01

General Instructions

- You can use *any* programming language you want, as long as your work is runnable/correct/readable. Two examples:
  - **In `R`**: it would be nice to upload a well-edited and working `R Markdown` file (`.rmd`) + its `html` output.
  - **In `Python`**: it would be nice to upload a well-edited and working `notebook` (or similia).

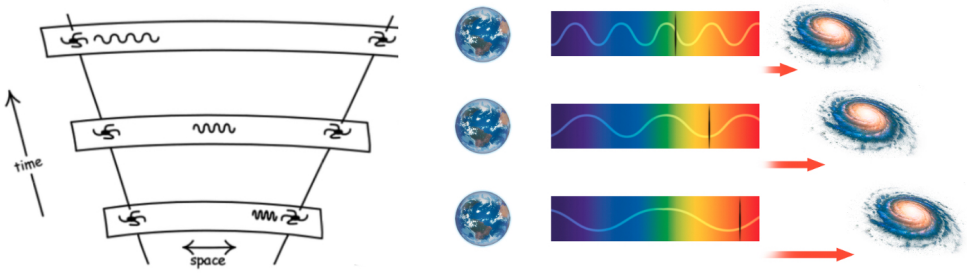
In case of `R`

If you go for `R`, to be sure that everything is working, start `RStudio` and create an empty project called `HW1`. Now open a new `R Markdown` file (`File > New File > R Markdown...`); set the output to `HTML mode`, press `OK` and then click on `Knit HTML`. This should produce a `html`. You can now start editing this file to produce your homework submission.

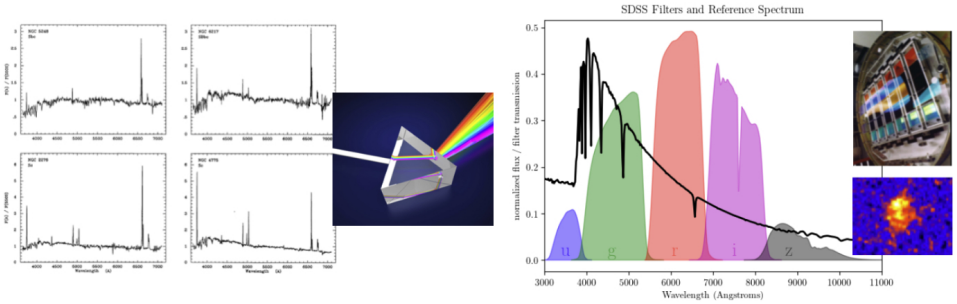
- For more info on `R Markdown`, check the support webpage: R Markdown from RStudio.
- For more info on how to write math formulas in LaTex: Wikibooks.

The Data | Photometric redshifts of galaxies

For this first part of the homework you'll be working on *redshift estimation*. Since 1929, when Edwin Hubble published his landmark work, we know that our universe is expanding and galaxies are moving away from Earth at speeds proportional to their distance (the farther they are, the faster they're moving). The velocity of the galaxies has been determined by their **redshift**, that is, an observable shift of the light they emit toward the red end of the visible spectrum. In other words, the expansion of the background universe stretches observed light spectra to longer (redder) wavelengths: this is called the Hubble's Law, and in a way it's at the center of a potentially big crisis in (modern) cosmology.



Nevertheless, being the redshift (z) of a galaxy a measurable proxy for its distance, it is crucial for studies in astrophysics and cosmology. All nice and dandy up to now, but the problem is, that *direct* redshift measurement via spectroscopy is however not feasible for a very large number of galaxies.



Left: Spectroscopy is hard! **Right**: Photometric measurement are everywhere in today's galaxy observations

Redshift estimates are hence often **predicted** from less resource-intensive *imaging data*, resulting in measurements called photometric redshifts or photo-z's: keep in mind, the Rubin Observatory's Legacy Survey of Space and Time (LSST), a multi-billion dollar cosmology experiment expected to be operation in January 2025, will crucially depend on photo-z's to achieve its science goals.

All this said, a few technical terms to clarify your task:

- typically, astronomical surveys collect broad-band filters images, which then get summarized as **magnitudes**: u, g, r, i and z (look at the **right** image of the last figure);
- (log-)difference of magnitudes are called **colors**: ug, gr, ri, iz and zy.
- Once properly imported, in your dataset you will have the previous **five colors** and **one magnitude**. The reason? Easy: empirically, it leads to better predictions of redshift, *feature engineering* 101!

Hence, based on this data...

...can we reliably estimate redshift from photometric colors?

⇝ Your job ⇜

1. Time to move to our little **Kaggle-competition-under-constraints** and its associated shared notebook (one per team, all your code and also comments must be available there).
   Before introducing the **constraints** under which you will work, let me list **the rules**:

   - **The Task(s)**
     Main Task: Prediction problem || details in the previous section
     Secondary Task: Methods and results explanation/presentation/visualization (I'll personally score this second task)
     Your *base* overall final score is composed as follow:

     $$\text{SCORE} = \big(0.6 \times \text{Main Task}\big) + \big(0.4 \times \text{Secondary Task}\big)$$

     The resulting score will be remapped in 0-30 and additional points will be assigned for originality/activity level.
   - **Cheating**
     No code-leakage allowed.
     If two codes are judged to be too similar, their final scores will be cut in half.
     If three codes are judged to be too similar, their final scores will be cut into thirds, and so on...
     No appeal shall be made at any time.
   - **Start-End**
     The competition will start today, Tuesday April 23 at 12:00 (pm) and will end on Sunday May 12 at 23:00 (...you know, for the sake of symmetry!)
   - **Availability**
     Right after the competition starts, the link to access the (private) competition will be sent to the team-leaders only by email (check your spam folder please!).
   - **Kaggle Notebooks**
     Right after the competition starts, I will create and share a Kaggle notebook (in the requested language) with the team-leaders of each group.
     It may take up to 10 mins to get it done, be patient.
     In case of troubles on my side, I may ask the team-leaders (again by email, check the spam!) to create a notebook on their own and share it with me as editor.
     These notebooks will be the only source of code/results/comments I will evaluate, nothing else will be considered.
   - **Reproducibility**
     You can work on your personal computers (of course) but all your pre-processing/models/submissions must be made available and be entirely reproducible as a separate code-version on the assigned Kaggle notebook: one code-version for each submission.
     As a suggestion, label each code-version with a meaningful name like: `Sub_01`, `Sub_02` etc.
     Set your random generators seeds properly and submit solutions **only** if they come directly from your notebook.
   - **Submissions Limits**
     The team-leader (and the team-leader only) can submit an entry.
     If you are organized as a **Kaggle-team**, this is not an issue.

Maximum Daily Submission: 20
Number of submissions eligible for the final private leaderboard: 2
Notice: The code-versions related to these entries must be clearly labeled and easy to find on the Notebook.

- **Prize(s)**
  ... for the more active teams? Surely there will be a prize!

And now back to the **constraints** that all submissions must verify to level the field.

1. You **must** use some flavor of **Local Polynomials**[1] within a **Generalized Additive Model** framework.
   For information: see my notes, Sections 7.6 and 7.7 of ISLR, and also Section 5 of `The purple book`.
2. You need to select the best combination of tuning parameters, typically the bandwidth h and the polynomial degree d, via a suitable predictive criterion. Regarding the latter, you may consider $d \in \{0, 1, 3\}$, that is constant, linear and cubic-local polynomials.
3. Finally, study and properly comment the *predictor importance* using at least **LOCO**[2] (again see my notes), and then anything else that you consider useful.

---

[1] You do **not** have to write it on your own, you can use dedicated libraries/packages/modules.
[2] In this case you have to provide your **own** implementation of the LOCO.