Karl M. Heuer / David E. Roberson

Autumn 2024

Final Project

02807 Computational Tools for Data Science

Deadline for handing in

23:59 on December 6, 2024
To be submitted on DTU Learn

Project Details

In this project, you should demonstrate you are able to use relevant computational tools for data science for a problem of your choice. This project is rather open, so it is important you make a realistic plan for the project such that you will manage to finish in time but at the same time do something that is technically challenging.

Your project must use at least one, and preferably two of the topics we have discussed during the course. Beyond this, it should include at least one new thing that was not directly taught in the lectures, but that you find and understand on your own. For some projects, it may not make sense to apply two different topics from the course. In such a case, you should make sure the course topic you use is crucial for your solution and is implemented in a nontrivial way and preferably extended beyond what was taught in the course.

To be clear, each of the following bullets count as one topic from the course:

- MapReduce (but this should only be used if it makes sense, e.g., you get access to hardware that can make use of the parallelism needed for MapReduce to be useful)
- Similar items/locality-sensitive hashing
- frequent items (mainly: A-priory algorithm and PCY-algorithm, or potential enhancements of this)
- general clustering algorithms (k-means, DBSCAN, hierarchical clustering, CURE algorithm) and evaluation measures (e.g. Davies-Bouldin index)
- graph clustering algorithms (betweenness centrality, spectral clustering, Louvain algorithm) and evaluation measures (e.g. modularity)

Note that things such as TF.IDF or similar topics that were not the main focus of a full lecture do not count as a topic from the course, but it could be used as part of an outside topic.

In this project, you should:

- Find/make a problem that is relevant within data science and that is technically challenging.
- Motivation/explanation of your problem should be part of the report.
- Analyze the problem and choose relevant tools to solve the problem.
- Implement/use the relevant tools to actually solve the problem.
- Argue clearly for the choices made when designing and developing the solution.
- Write a well-written and concise report about your problem and solution.

Note that it is okay to use existing Python packages for your project, but in order for the project to be considered technically challenging enough for you to receive a good grade, you will likely need to implement some things yourself.

Example high-level project ideas

- Recommendation systems
- Detecting fraud or plagiarism
- Sentiment analysis

Groups

The project work must be completed as a team task. The group/team size is 4 or 5 students. You must sign up for groups on DTU Learn. Groups of size 5 will be expected to do approximately 25% more in order to receive the same grade.

Submission

All of the relevant materials for your project (e.g. your report, jupyter notebook, and any data required for your code to be run) should be submitted on Learn. If your data is too large to be submitted on Learn, then you can put it on github and provide a link in your submission.

Project report

The report should be at most 11,000 characters, not including white spaces (12,500 for groups of size 5). This should be approximately 4 pages (4.5 for groups of size 5). The names of all the groups members should be clearly written on the title page of the report. In the report you should

- Briefly explain/motivate your problem.
- Describe the data you used and any preprocessing you had to do. Make sure it is clear how you obtained your data and where you got it from.
- Present the work you have done, the decisions you have made, report your results, etc.
- Argue for whether or not your method of solving your chosen problem was successful.
- Make sure to clearly identify the technique/algorithm that you used that was not from the course. There should be a brief explanation of this method and how it works/what it does.

- Avoid repetition, convoluted explanations, etc.
- It is extremely helpful if you include somewhere near the beginning of the report a concise summary/outline of the entire project/report.

As an appendix to the report you should submit a Jupyter Notebook that includes your code. The notebook should be well documented. Also, sufficient instruction should be provided for how to use the code. Furthermore, you must include an outline detailing each group member's contribution to the project. This outline can also be part of the appendix. The report together with the appendix must be uploaded on DTU Learn at latest on the 6th of December.

Gathering data

The project should make use of some source of data. For example you could use Pandas to extract some data from a website. You can also take a look at https://www.kaggle.com, which contains many different types of data sets.

Grading

The project will be graded primarily on the final report but the submitted Jupyter Notebook will also be evaluated (see below for specifics). Some key criteria in the evaluation are:

- The project should be based on a relevant and interesting question/problem, ideally coming from or inspired by the real world. However, you can also focus on enhancing/improving a known algorithm, as long as it has real world relevance. In either case, it should be a technically challenging task (i.e. solving a very easy problem well will not give you a high grade.). Your chosen problem should be clearly explained in the report.
- The topics/algorithms from the course that you choose to use for your project should be essential to your solution, not simply incidental.
- You should make clear arguments for the choices you made in designing your solution. E.g., why are the algorithms you chose applicable to your problem and/or data (for instance, requirements on the format/shape of data should be met). Please keep in mind that this does not mean that you should waste space in your report by re-explaining how an algorithm works if this was explained in class.
- The report should include an evaluation of how well your approach/solution performed in solving your problem. This should include some reflection on why you think it performed the way it did, i.e., if it did poorly, why do you think it did poorly.
- It is not a requirement that you scrape your own data, however, doing so will contribute to your grade, e.g., if your data requires significant cleaning/processing, then this will be taken into account in our evaluation. Additionally, effectively working with a very large data set will positively effect your grade.
- It is not required that you implement everything on your own, however, your code should demonstrate that you have accomplished something technically challenging. E.g., if you only use routines from already existing packages, then these should be combined in a creative and nontrivial manner.
- The report must be concise and well-written. Make sure to use the space allowed effectively.
- Your code should be well-documented, and as self-contained as possible, i.e., we should be able to run the code you submit with as great an ease as possible.