‘Clustering Python Programs’
Mihhail Mihhailov, Mikk Märtin

Task 1:

Link to GitHub repository:
https://github.com/mihhailmihhailov/IDS2022_ClusteringPythonPrograms

Task 2:

Identifying business goals:

University of Tartu offers a number of courses related to programming. Many of them involve students submitting their homeworks in the form of code. Much like any other type of writing, code can be plagiarized. Though checking whether a piece of code was plagiarized presents a challenge because comparing functions is more time-consuming than comparing paragraphs in essays. If there are a lot of pieces of programs to go through, the time required might become pretty substantial.

Therefore, a different solution is needed - one that relies on a statistical model to establish if certain homework submissions have too many similarities to be considered a coincidence. Such a solution would reduce the time that the teaching assistants would need to spend checking the homeworks because they would no longer have to compare every single program to one another. A good estimate of a successful model would be one where at least 85% of the homeworks flagged as plagiarized were deemed to be so by the teaching assistants after manual review.

Assessing the situation:

At present, there is a team of two students ready to work on creating this model. Both people have knowledge of the Python programming language, Jupyter Notebooks as well as Python libraries designed for data science purposes, such as scikit-learn, and have had at least half a year of experience using them.

A dataset to be used for creating the model is also provided, though there is still some work needed to extract features from it. The dataset consists of homework submissions for 13 homeworks in the form of ‘.py’ files, which are programs written in Python.

In terms of requirements, the project is to be finished by Thursday, December 15th 2022. In addition to the statistical model itself, a poster describing it and the work behind it has to be made by Monday, December 12th 2022.

There are not that many factors, which could hamper being able to deliver the finished product on time. The only notable one would be loss of data due to hard-drive failure. Such a problem could be easily remedied by frequently pushing code changes to the project’s GitHub repository.

Finally, a cost-benefit analysis is not appropriate for the given project as no money is exchanging hands.

Defining data-mining goals:

As stated before, the main deliverable of this project is a statistical model which is able to divide the '.py' files given to it into groups based on how similar they are to one another. In addition to that, a program that converts '.py' files into rows in a dataframe is also needed, with the extracted features being the columns.

As far as success criteria go, we believe that maximizing precision is the first priority. While it is true that both precision and recall are very much important in this kind of model, in our understanding, a model that, for example, correctly finds half of the cases of plagiarism without false positives is better than one that finds all the cases but then also just as many works that aren't plagiarized. Therefore, as once again previously stated, we are saying that a successful model should have at least 85% precision.

Although it should be noted that ignoring recall is also quite problematic because that would mean too many people would get away with plagiarizing. As such, we are also placing a criterion on the recall: it should be no lower than 60%.

Task 3:

- **Data requirements** - To achieve the goals that we've set out to achieve with this project, we need solutions to homeworks, that correspond to people submitting the aforementioned homeworks. The data needs to be in plain text, to make it more suitable for analysis with whichever ML algorithms we end up employing.

- **Data availability** - In this regard, everything is clear. The dataset that we intend to analyze has been provided to us, and it does indeed contain the data that we require. The data does need to be transformed to a format more suitable for data science; the amount of folders will otherwise prove cumbersome. Furthermore, as mentioned before, Python code needs to first be converted to plain text.

- **Selection criteria** - We will use the homework dataset provided by Reimo Palm. Within the dataset, we will use only the last submission by any given student for any given problem. In other words, we will analyze the final version a student submitted and disregard any previous attempts. In addition, we will disregard the moodle metadata found in .ceg folders

- **Describing data -** The data is structured by a couple interconnected principles. Firstly, the homeworks for each week are stored in zip files, corresponding to the week. Within each zip file, there is a large number of folders, which correspond to students that have submitted homework for that week. In each of these folders, there is an undetermined amount of folder pairs – one that contains the programs, and another that stores certain moodle metadata. The amount of folder pairs is determined by how many times a particular student submitted their solutions. Within the program folder, there are one or more programs that correspond to an exercise in the corresponding week's homework. Generally, each solution is in its own file, labeled koduX, where X stands for 1, 2, 3…. The programs themselves contain an undetermined amount of code, which we need to attempt to cluster using ML.

- **Exploring data -** Due to time constraints, we have been unable to get the data into a dataframe by the homework deadline. Therefore, an explanation of the difficulties and some speculative predictions of future problems will have to suffice. Firstly, getting the data into a familiar format (such as a pandas dataframe) will require a

couple hours of work. The first challenge is writing a function that will iterate through the folder tree, select the folder containing the latest submission for every student, and enter all the programs contained within a dataframe.
On a preliminary glance, there is missing data: sometimes, students have not submitted anything, other times, they have not submitted solutions to all problems. This needs to be accounted for.
Finally, though this could also be considered data preparation, the solutions themselves need to be split up into smaller parts somehow (though trying to brute-force it with any easily available NLP algos does seem alluring). Within one particular submission, the structure of commands/logic should be prioritized, to catch the common cheating methods outlined in the dataset readme.

> *"Typical ways of cheating are changing variable names, changing input-output texts, and adding or removing comments. This can be taken into account when estimating the similarity of programs. The general logic and flow of the program usually remain the same."*

As such, there is also a need for some method that could disregard variable names, or at least deprioritize them for the purposes of clustering.

- **Verifying data quality** - In this case, missing data will be inevitable and impossible to overcome. Generating data does not make sense in this case – if a student has not submitted a solution, it would not be accurate to "generate" one for them (not to mention the question of how one would go about doing that in the first place). As such, we need to work with the data that we have. However, this should not interfere too much with our goals. If a solution to a problem is not submitted, it cannot be an outlier, nor can it be copied off another solution, therefore making it irrelevant in finding instances of the former.

Task 4:

List of steps:

1) Figuring out what features can be extracted from the dataset and how.
   Time estimate: ca. 15h Mihhail, ca. 15h Mikk
2) Writing a program that converts each zip file of the dataset into a dataframe, containing the relevant data and metadata.
   Time estimate: ca. 10h Mikk
   Input: dataset as zip files
   Output: dataset as Pandas dataframes
   Dependencies: there is a plan on how to extract features from the dataset
3) Researching models and figuring out which might be suitable for the current project.
   Time estimate: ca. 5h Mihhail
4) Training multiple models of different types with different hyper-parameters.
   Time estimate: ca. 5h Mihhail
   Input: dataframe of the dataset
   Output: multiple models
   Dependencies: there is a plan on what kind of models to train; the dataset has been converted to a dataframe

5) Evaluating the models and picking the one with the best test results.
   Time estimate: ca. 5h Mihhail, ca. 5h Mikk
   Input: multiple models
   Output: one model
   Dependencies: there are models that have been trained; an evaluation set was made during the process of converting to a dataframe
   Actions: in case none of the models reach the expected results, the steps ‘Researching models’ and ‘Training models’ are performed again
6) Making a poster for the project.
   Time estimate: ca. 5h Mihhail, ca. 5h Mikk
   Output: a poster
   Dependencies: the project itself has been completed or mostly completed