King Abdulaziz University

Faculty of Computing & Information Technology

Computer Science Department

CPCS-433

Artificial Intelligence Topics

Assignment 1

(Semester III, 2023)

Date: 11/5/2023

Khaled Alasmari – 2035189

Introduction

When the assignment was first assigned to us, I knew that I wanted to use all the files that were given, and so I did. The process was a bit tedious, but I managed to use around 68% of the 1400 assignments. I had to ignore some files that were not clearly labeled or input heavy. I'll expand on this later. Finding features was not that hard – after getting some guidance – but, on the other hand, I struggled to find a useful label. We were not given any information about the correctness of the assignments, and testing the assignments was not feasible. I used Python to automate the process of compiling and executing the files, finding features, and creating the dataset.

Automating the process

Naming the files and classes

In Java, if the class is declared as public. The class name must match the file name to compile and execute the program, unfortunately, the file names were automatically changed to something different than the class names.

I attempted to make the class name match the file's, but I ran into an obvious issue. The symbol "#" contained in the file name cannot be used in a class name. I had to change the file name and then change the class name to match it. Once I figured that out it did not take much to implement the fix.

Compiling and executing

Compiling the files was straightforward. I created a subprocess that executed the command 'javac' followed by the file's path. Running the compiled file was not hard either, but it was a bit tricky to measure the execution time. You will notice that the execution time is a lot higher than expected, and that is due to the overhead introduced by executing the files through Python's subprocess module.

I had to eliminate problem four from assignments 3 and 4 because they were input-heavy and were not submitted by everyone.

Extracting features

Now that I got the hardest part out of the way, I can focus on extracting features out of the files. This was not particularly hard, All I had to do was open the file and count/match the feature that I want.

I already had 2 features done, compilation status and execution time. The file name also contains very important features such as the student id and the submission time. Also, one of the suggested features was the file size so I extracted that too.

The most time-consuming thing was finding the regex pattern that matches the feature that I want. I referend from attempting to extract features such as number of nested loops or methods calls because it was near impossible to get right.

Labeling the data

I've struggled to come up with good labels, because there was not any way to check if the submitted answers were correct or not, and that severely limited my ability to come up with labels.

Using the hints that were given, I thought about categorizing the submissions based on the coding style.

Writing the data

This was the easiest thing I did. Instead of reading out of a bunch of files, I'll write to one file!

I used the csv module to properly format and handle the process of writing rows to the file.

Extracted Features

Here is a table of the features that I managed to extract out of the files:

Feature	Explanation
Student_ID	KAU student's identification number.
Assignment_number	CPCS-202 assignment number.
Problem_number	The problem number in the mentioned assignment.
Submission_time	Date and time of submission.
File_size	Self-explanatory.
Methods_count	The number of methods including main.
Code_lines_count	Self-explanatory.
Average_method_length	The number of lines per method (lines count divided by methods count).
if_statements_count	The number of if/else if statements.
else_statements_count	Else statements count (excluding else if).
switch_statements_count	The number of switch statements.
Loops_count	For and while loops count.
Comments_count	Comments count including the mandatory information one.
primitive_variables_count	Java's primitive variables count
Compiled_succesfully	The file compilation status.
Execution_time	Self-explanatory.

Extracted labels

Here is a table of the labels that I managed to extract out of the files:

Label	Explanation
Compiled_succesfully	The file compilation status.
Preferred_condition_statement	Uses if, switch or both
Preferred_loop_statement	Uses while, for or both

The quality and completeness

As mentioned before, I've omitted problem four from both assignments because they were not submitted by everyone, and they required a specific input to run correctly. Also, some submissions were not considered because the file name was not clear about the content.

As of the quality of the data, I cannot judge it properly given that I lack experience in the field of data science, but I've done my best to extract the most useful information that can aid in categorizing students based on patterns shown in their code.

Potential use cases

The dataset can be used to cluster students into groups to identify their preferred style of coding (if vs switch, for vs while). Also, with the addition of more information about the students' section you may be able to highlight the differences in teaching styles between professors. This can be used to identify professors who encourage students to write clean code.

Another good use case is to identify logic problems in conditions and loops. For example, a student may use 6 or 7 conditions when it requires 2 to solve the problem. This information can be used to improve lab exercises and exams.