Educational data mining — Programming snapshot data

Link to GitHub: https://github.com/alepiere/CSC313_DataMine

First task: exploratory data analysis (9 points)

Question 1. How many *unique students* are represented in this data-set? Why do you suppose your answers would be different depending on which one you looked in?

There are 413 unique students represented in the data set. To find this answer, we found all the unique entries in the "SubjectID" column of the MainTable dataset. We did this using the nunique() function.
Our answers might be different depending on what dataset we looked at. Some students might not have received a final grade. The Main Table contains all students who originally enrolled. We assume some students may have dropped the course, which would cause a difference in the MainTable student amount vs the Subject Student amount.

Question 2. The data-set contains many Assignments, and each Assignment is made up of multiple Problem. Which Problem had the highest average number of attempts per student?

Problem 102 had the highest average number of attempts per student, with about 10.423 attempts per student. To find this answer, we concatenated the early and late datasets. Then we found the problem with the highest number of attempts by grouping the dataset by ProblemID and Attempts. Then we took the highest mean. We found the problem using idmax().

Question 3. On which Problem did students tend to face the most compiler errors? First, how do you interpret this question? Am I asking you to:

- compute, for each problem, the average number of compiler errors across all students, and then report the maximum, or
- compute the total number of compiler errors faced on each problem, and report the maximum

Which one would be more meaningful?

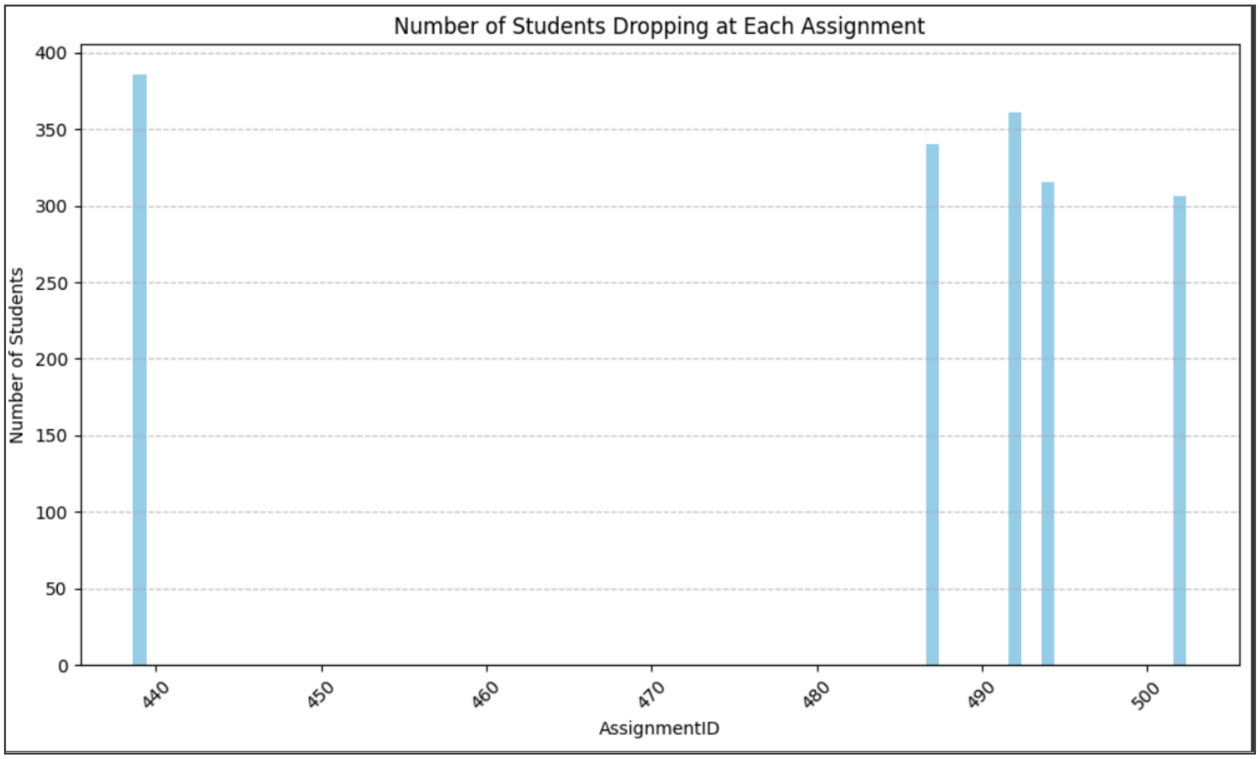
```
ProblemID     13.000000
ErrorCount     7.947883
```

The average per student is more meaningful. We need to account for the number of students who have attempted the problem because otherwise a problem with hardly any attempts but many errors per attempt will not be considered by calculating only totals.
To get this answer we extracted only the columns 'SubjectID', 'ProblemID', and 'Compile.Result' from the df_main DataFrame. Then we filtered the DataFrame to keep only the rows where the 'Compile.Result' is 'Error', indicating compiler errors.
Then we grouped the filtered data by 'SubjectID' and 'ProblemID' and count the number of errors for each group. Finally, we calculated the average number of errors per student for each problem. And we lcoated the row with the maximum average error count.

Second task: open-ended analysis (11 points)

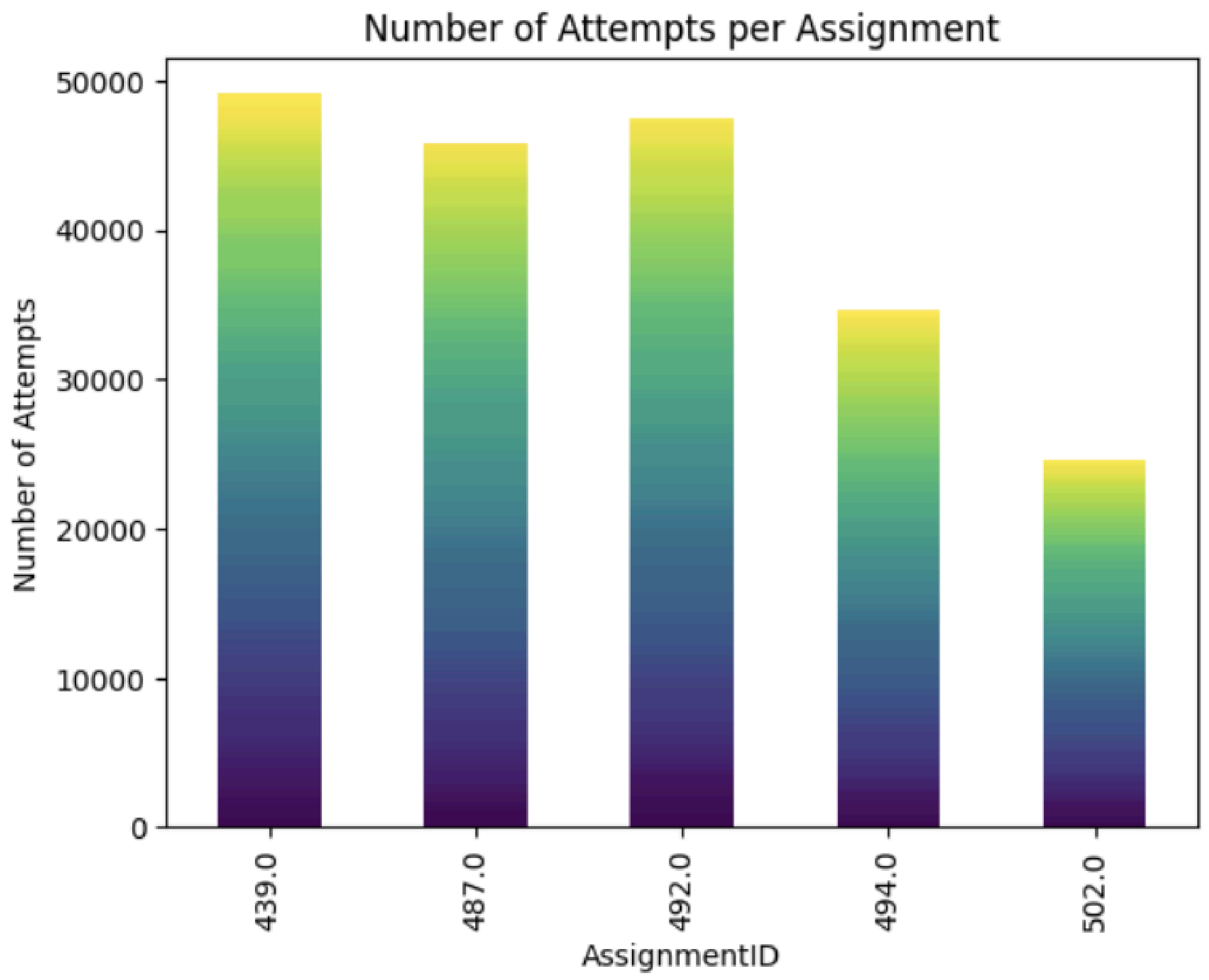
1. Does early programming performance predict final exam scores?
2. Can we predict student struggles on later programming problems using early performance data?
3. Can we identify students at risk of dropping the course based on their early programming performance?

First, we were curious to know how many students dropped out, when they did, and why. To start, we made a bar graph of how many students dropped after each assignment. We did this by grouping the MainTable by AssignmentID and SubjectID. Then we made DropCount the index and used Matplotlib to create the visual bar graph (shown below).

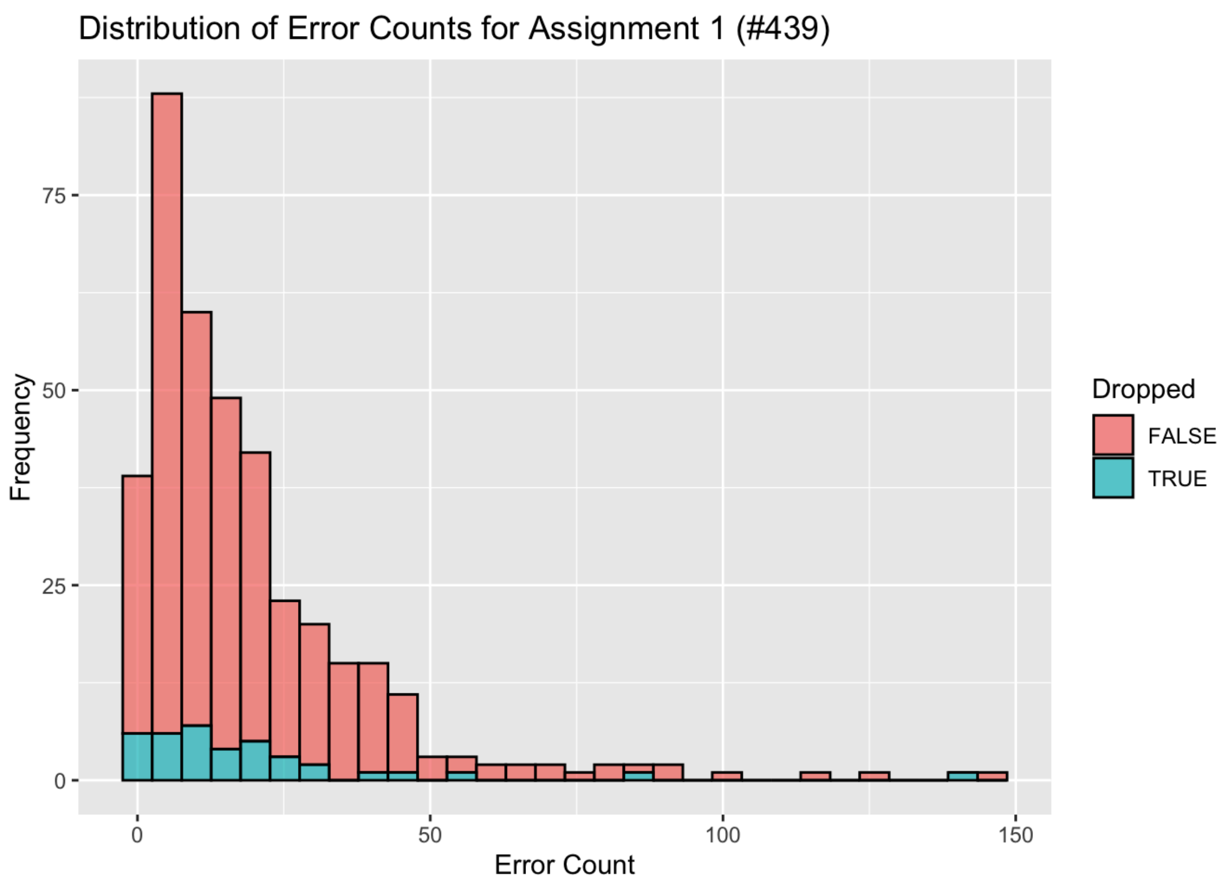


After seeing the graph, it was clear that most students dropped out after the first assignment. There could be many reasons for this, such as class size limits or financial reasons, but we wanted to focus on the psychological reasons most capable students drop out after this assignment. We needed more information.

Then we calculated the number of attempts per assignment. We found that it is normal for most students to struggle at the beginning of the class. The first assignment had the most number of attempts. We grouped the MainTable by AssingmentID and SubjectID then plotted it on a bar graph (Shown below).

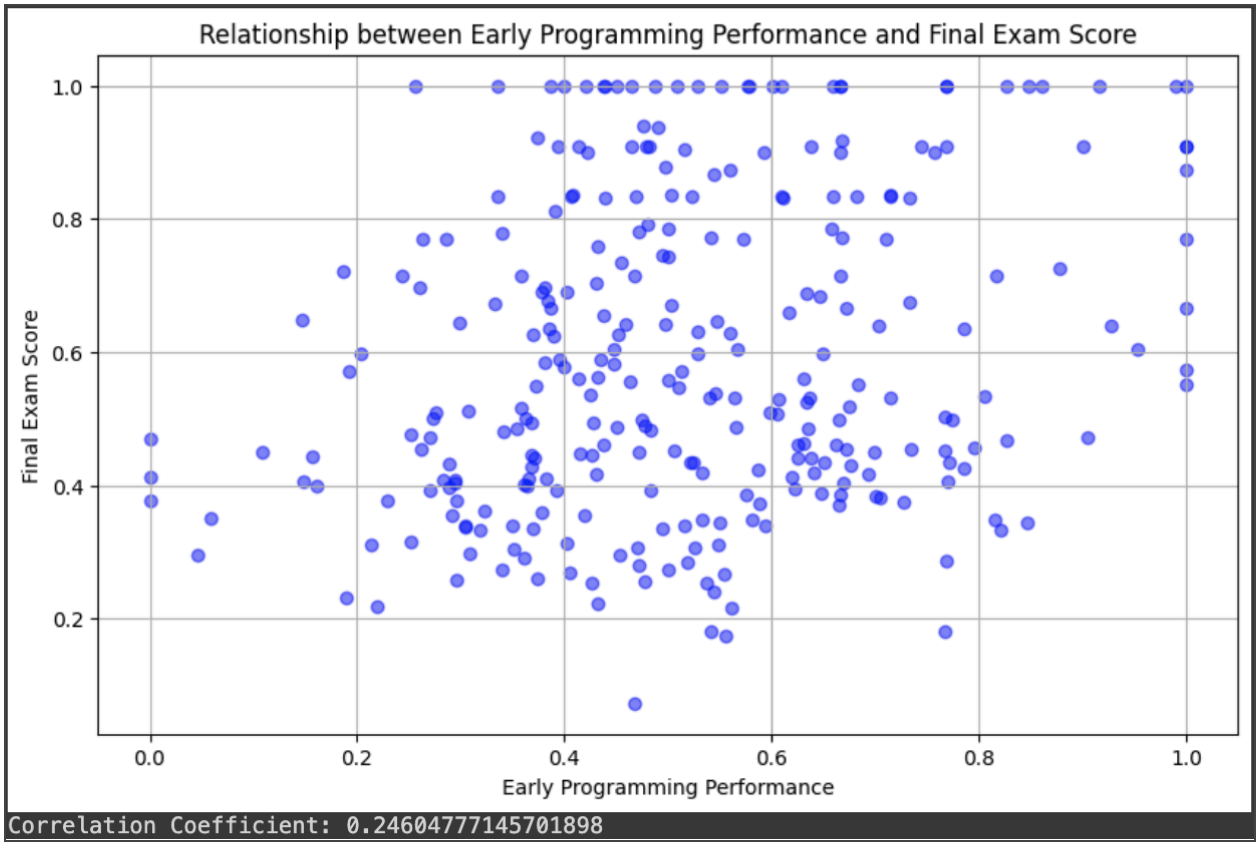


In addition, we calculated the number of errors students had for assignment #1 specifically, and contrasted which students dropped and which ones stayed in the course (Shown below). We predicted that a higher error count would lead to higher dropout rates, but between the two groups they are pretty even. This told us that the difficulty of the assignment doesn't necessarily mean more people dropped out.



At this point, we had a few hypotheses as to why most students drop after the first assignment:
1. Difficulty Level: The first assignment might have been more challenging than expected, leading to frustration and loss of motivation among students.
2. Lack of Confidence: Students may have lacked confidence in their programming abilities after the first assignment. If they struggled significantly or received poor grades, they might have doubted their ability to succeed in the course.
3. High Expectations: Students might have had high expectations for their performance in the course, and when they didn't meet these expectations on the first assignment, they became discouraged and decided to drop the course.
4. Inadequate Support: Students may have felt that they did not receive adequate support or guidance from the instructor or teaching assistants, making it difficult for them to succeed in the course.

Finally, we wanted to know if how well you do early in the class has any impact on your success in the class. To do this we compared performance on early programming assignments to final exam scores. We merged the two data sets of early assignment scores and final exam scores, then plotted the relationship using a scatter plot (Shown below).



```
Correlation Coefficient: 0.24604777145701898
```

This graph shows no real linear relationship between the two. We then calculated the correlation coefficient to understand their relationship. The correlation coefficient was 0.246. This tells us that there is a weak positive connection between early programming performance and final exam scores. So while on average, as early programming performance increases, final exam scores increase, one does not necessarily give us any information on the other. This means that a student struggling at the beginning of the course can still be very successful and vice versa.

Our analysis revealed several important insights into student performance and course retention.

Firstly, we found that a significant number of students drop out after the first assignment, which may indicate that they struggle to adapt to the course's difficulty or feel overwhelmed by the workload.

However, after more research, we found that it was common for students to struggle on the first assignment. The first assignment also had the highest number of attempts, which suggests that students may find it particularly challenging. This may be because programming is often new to beginners, and they require time and support to adapt and develop the necessary skills.

Additionally, our exploration of the relationship between early programming performance and final exam scores showed a weak correlation, indicating that early performance may not be a strong predictor of final exam success. So while most students struggle on the first assignment, it is not a good predictor of this success in the class. Rather, a better predictor to success is whether a student stays in the class after the first assignment.

We found that it is crucial to provide adequate support and resources to students early in the course, especially after the first assignment, to help them overcome challenges and build confidence.