INTELLIFY - WINTER INTERNSHIP

Final Report

NEN201

Prakhar Ganesh - 2015CS10245

Task 1

Data Analysis and Preparing reports for every school

Code ->

analyse.py

script.py

Steps ->

1. Read raw data from CSV file
2. Process data to
   a. Identify correct school for students who wrongly filled their OMR
   b. Remove School IDs which don't actually exist, but got created in wrong OMR filling
   c. Sum performance for every skill, and storing skill wise performance instead of question wise performance
   d. Calculating average performance of School for every level (both skill wise and overall)
3. Calculate Average performance in the Olympiad for every level (both skill wise and overall)
4. Rank schools based on their performance (24 different rankings -> 4 levels * (5 skills + 1 overall))
5. Once data for every school is collected, format it properly, create charts as required and finally create a HTML file.

6. Once this html file is created, use **'xhtml2pdf'** to convert all HTML files to PDF reports

Final Output Directory Structure ->

```
Data
        -- school_id1
        -- school_id2
        ..
        ..
        -- school_idk
                -- chart0.png
                -- chart1.png
                ..
                ..
                -- data.html
                -- data.pdf  (The final report file)
```

Task 2

Recommendation System

Code ->

recommend.py

Steps ->

1. The whole data was read and the percentage of students who correctly answered every question was found.
2. This way, every question was given a difficult, medium or easy tag.
3. The first 10 questions asked and their response is taken as input by the code
4. An array of 45 questions is initiated with score = 0 for all
5. The questions that have already been asked are given score = -inf (Since they should never be asked again)
6. For every skill, a value is calculated as QAS - 0.2*QT, where QAS is the number of questions asked from that skill level and QT is the total number of questions asked till now. Now this value is subtracted from the score of all questions of that skill.

Reasoning -> Negative value means that the skill doesn't have enough questions asked and thus should be given more importance while recommending the next questions, which is why its score will increase (negative values is subtracted), and vice versa.

7. For every skill, the average performance of the student is calculated, giving 3 marks weightage to difficult questions, 2 marks for medium and 1 for easy questions. Three threshold sections are created to rank student performance as low, medium or high. Dependingly, easy, medium and difficult questions are given some value. This value is subtracted to get score of every question of that skill. Higher values is given to difficult questions in case performance is low and vice versa.
   Reasoning -> If the student performance is high, clearly the student is performing well and thus should be asked more difficult questions. In case of high performance, the difficult questions are given less value and since this value is subtracted from their score, they in turn tend to have more score and thus better chance of getting picked.
8. The question with the highest score is selected. If there is more than one question tied, choose one at random.
9. Repeat the process again for the next recommendation.

Final Output ->
On the command line

Examples ->
Definition -> For every question, I calculated how many students were able to answer it correctly (the percentage)
If percentage < 20% -> Difficult Question
If 20% < percentage < 40% -> Medium Question
If percentage > 40% -> Easy Question

Example 1

Questions

[13, 21, 30, 7, 5, 33, 25, 42, 9, 2]

Response

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Skill Index

[1, 2, 3, 0, 0, 3, 2, 4, 0, 0]

Difficulty Index

[0, 1, 1, 1, 1, 1, 1, 1, 1, 1]

New Questions

[13, 21, 30, 7, 5, 33, 25, 42, 9, 2, 44, 17, 38, 20, 18, 40, 26, 19, 31, 43]

Skill Index

[1, 2, 3, 0, 0, 3, 2, 4, 0, 0, 4, 1, 3, 2, 1, 4, 2, 1, 3, 4]

Difficulty Index

[0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 2, 1, 1, 1]

Notice ->

1. Since the first 10 questions had a large number of questions from skill 0, the recommendation has arranged itself such that the skill 0 questions are not asked in the next 10 questions, and questions from other skills are asked. Finally after 20 questions, the count of number of questions for all the skills is 4, 4, 4, 4, 4 (perfectly equal)
2. Since the initial questions asked were all, except one, of Medium Difficulty, and the student responded to all of them correctly, the recommendation has arranged itself such that difficult questions are asked now, since the student have done medium difficult correctly.

Example 2

Questions

[13, 21, 30, 7, 5, 33, 25, 42, 9, 2]

Response

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Skill Index

[1, 2, 3, 0, 0, 3, 2, 4, 0, 0]

Difficulty Index

[0, 1, 1, 1, 1, 1, 1, 1, 1, 1]

New Questions

[13, 21, 30, 7, 5, 33, 25, 42, 9, 2, 11, 43, 32, 10, 14, 22, 44, 15, 35, 27]

Skill Index

[1, 2, 3, 0, 0, 3, 2, 4, 0, 0, 1, 4, 3, 1, 1, 2, 4, 1, 3, 2]

Difficulty Index

[0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1]

Notice ->

1. Since the first 10 questions had a large number of questions from skill 0, the recommendation has arranged itself such that the skill 0 questions are not asked in the next 10 questions, and questions from other skills are asked. Finally after 20 questions, the count of number of questions for all the skills is 4, 5, 4, 4, 3 (almost equal)
2. Since the initial questions asked were all, except one, of Medium Difficulty, and the student responded to all of them incorrectly, the recommendation has arranged itself such that easy questions are asked now, since the student have not done medium difficult correctly.

Example 3

Questions

[16, 19, 30, 7, 5, 33, 25, 42, 9, 2]

Response

[0, 0, 1, 1, 1, 1, 1, 1, 1, 1]

Skill Index

[1, 1, 3, 0, 0, 3, 2, 4, 0, 0]

Difficulty Index

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

New Questions

[16, 19, 30, 7, 5, 33, 25, 42, 9, 2, 20, 13, 41, 26, 38, 14, 44, 28, 15, 35]

Skill Index

[1, 1, 3, 0, 0, 3, 2, 4, 0, 0, 2, 1, 4, 2, 3, 1, 4, 2, 1, 3]

Difficulty Index

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 0, 1, 2, 2, 0, 1, 1, 0, 1]

Notice ->

Mixture of Example 1 and 2. For skill 1, the student was not able to answer any question (they were Medium level) and for other skills, the student got all the questions right (they were also Medium level). So for other skills, the level increased and more difficult questions are asked. But for skill 1, level decreased and easy questions are asked.

Note ->

You will find that at places where more, say skill 1 question, were to be asked, instead other questions are asked. Similarly, you may find that places where the difficulty level should have increased/decreased, it actually didn't change or maybe went the other way.

All of this is because of the limited availability of number of questions. The recommendation wanted to find a more difficult question from that skill, but all the difficult and Medium questions were already asked, so it instead asked the easy questions. Similarly, it wanted to ask more skill 4 questions, but since we only have 5 skill 4 questions, we will have soon asked them all and thus the count will not remain balanced after a few questions.

These kind of anomalies can occur, but will go away if we give more questions to this program to work with (more in every skill and more in variety of easy medium difficult also).