##### Questions

1. Describe your **goal** (not your company’s goal) in this scenario.

We want 6 applicants that are the best out of the bunch and meet all of the requirements that are down below.

1. Describe one **task** that you could try using that can help achieve this goal.

Setup requirements for the applicants, such as quantitative data on resume like gpa, years of experience, projects average ratings, etc.

1. Describe one other task that could help achieve this same goal.

After people have been sorted through the requirements, they must be interviewed to find the best out of them.

Model 2 Question 1.

Om- Persons 0, 1, 4 (5 min)

* 0 is chosen due to their proficiency in moderate programming as well as moderate knowledge of algorithms and databases.
* 1 is chosen due to a consistently above average proficiency in advanced programming tasks
* 4 is chosen due to a basic understanding of programming while being proficient in database systems.

Austin- 0,1,6

0- is chosen since they are highly qualified in their work in terms of algorithms, networking, and advanced computer science.

1- is chosen since they are also highly qualified in their work in terms of algorithms, and high GPA.

6- is chosen since they have a high algorithms grade, despite their lower GPA.

These took around 3 minutes to complete.

Keval - 0, 1, and 2

* There are four classes that I feel would be relevant to this problem: Introduction to computer programming (the basics), advanced computer programming (more of the basics), database systems (to manage the machine learning databases), and algorithms (to determine the best algorithm to predict what customers are going to order). 0 has the highest average grade in these four categories, while 1 has the second highest average grade and 2 has the third highest average grade. It took me approximately 6 minutes to determine the best candidates for the position.

2. It took us three minutes to choose person #2 over #4 and #6. We already agreed on #0 and #1.

3.

Om- 5.1 days selecting 100 candidates individually

Austin- 3.9 days selecting the 100 candidates from 18000 peeps

Keval - 7.8 days

Team Time- 15 hours

Model 3

##### Questions

* + Don’t consider any candidate who has an average of less than 84 out of the following four classes: Introduction to Computer Programming, Advanced Computer Programming, Database Systems, and Algorithms

1. See #1
2. Once your team has a set of rules that works for the candidates you’ve seen so far, obtain a second set of candidates from your instructor. Apply your rules to that set of candidates. Do your rules select the top two or three from this new set of candidates? If your rules do not give a definitive answer for a candidate, either add a rule or modify the ones you have so that it does. (And then you need to apply your rules again to the candidates you’ve already evaluated.)
   * Our rules select #5 and #6 from this set of candidates.
3. Review all the candidates that your rules rejected. Would you team still want to consider any of these candidates? If so, describe the features of the candidate that your team’s rules did not account for.
   * A candidate that I may want to consider is #7. Although they had a low Introduction to Computer Programming score, they had an amazing Algorithms grade, which is very important to machine learning.
4. Exchange your rules with one other team.
   * Apply the other team’s rules to the original set of candidates, and verify that you get the same results as that team did.
     1. Their rules selected candidates 0, 1, and 2, the same as the team did.
   * Apply the other team’s rules to the second set of candidates, *without modifying their rules in any way*. Note any candidate for which the rules give an ambiguous result.
     1. Their rules selected 3 and 7. There is no ambiguous candidate.
5. Describe the differences in the candidates your rules selected and those candidates the other team’s rules selected.
   * Our rules and the other team’s rules selected the same candidates for the first set of candidates, but we selected two entirely different sets of candidates for the second set of candidates ({5, 6} vs {3, 7}).
6. Describe the input and output for the algorithm your team developed.
   1. The input is each candidate’s grades in the four classes listed above. The output is the candidates that pass the algorithm’s test (in the first set of candidates, that would be 0, 1, and 2, while in the second set, it would be 5 and 6).
7. Explain whether your team’s algorithm is correct, reliable, and reproducible. Give specific examples.
   1. Our algorithm is all three of the above. It is correct because it outputs the candidates that we wanted it to. It is reliable because there is no situation where it may be ambiguous; all rules are binary. It is reproducible because applying the algorithm somewhere else (with the assumption that each candidate will still have the same format of input) will still result in the same people being selected.
8. Why do you think that it is important that an algorithm be
   1. Correct?
      1. It’s important for an algorithm to be correct so that the program can actually solve the problem that you’re trying to make it solve. If you’re trying to make it add 1 and 1 and you’re getting 3, you’re not solving the problem.
   2. Reliable?
      1. It’s important for an algorithm to be reliable so no matter which numbers you throw at the algorithm, it will solve it and return the correct answer.
   3. Reproducible?
      1. It’s important for an algorithm to be reproducible so that the algorithm can be used to solve the same problem somewhere else.
9. For each of the following, give one example of a problem that can occur.
10. The algorithm is not correct.
    1. If the algorithm isn’t correct, you’re not solving the problem at all. In fact, you’re doing something even worse: using an incorrect solution as an attempt to solve the problem. Like I said above, if you add 1 and 1, you should get 2, not 3. Getting 3 will lead you to believe that the answer is 3.
11. The algorithm is not reliable.
    1. If the algorithm is not reliable, it may not work for another input that you try, leading to the same problem as the algorithm not being correct.
12. The algorithm is not reproducible.
    1. If the algorithm is not reproducible, it may not work when applied somewhere else, leading to the same problem as the algorithm not being correct.