We thank the reviewers for their insightful responses. Due to space limitations, we were unable to respond to all of the comments we found valuable, e.g., properly defining the term 'Seldonian', strengthening the introduction with material from the related work, properly describing the recursive process of computing bounds on terms in the expression E, potential avenues for giving high probability guarantees for a non-iid setting, etc. We will incorporate this feedback.

Reviewers suggested moving material from the supplemental section into the main body, such as an explanation of theoretical assumptions, more detailed algorithm descriptions, introduction of quantities used in lemmas and theorems, and additional experimental figures. We will do our best to incorporate this material in the main body. If accepted, NeurIPS allows an additional page in the main body, which will help us to do this, along with massaging existing text.

R1: Extending to other statistical definitions, like equalized odds and other variants. RobinHood applies to all definitions that can be represented as certain operations (listed in Section 4) on variables for which high-confidence upper and lower bounds can be computed. This includes variables with unbiased estimators, e.g., false positive rates (FPR) and true positive rates (TPR), and variables without unbiased estimators, e.g., standard dev. We will make a point to elaborate on how this can be extended to other statistical definitions in the main text. As an example of how to create a behavioral constraint that enforces (approximate) equalized odds in the loan approval problem, we assume that the user has unbiased estimators of TPR and FPR. Equalized odds requires that FPR and TPR are equal between protected and unprotected groups. To satisfy $g(\theta) \leq 0$ if θ is fair, we can set $g = |\mathbf{E}[\texttt{FPR}|\texttt{f}] - \mathbf{E}[\texttt{TPR}|\texttt{m}]| + |\mathbf{E}[\texttt{FPR}|\texttt{m}] - \mathbf{E}[\texttt{TPR}|\texttt{f}]| - \epsilon$.

R1: How is the candidate selection done? What is the algorithm to optimize over different candidate sets? In our experiments, RobinHood used CMA-ES [45] to find candidate solutions. RobinHood randomly partitions the data into 60% candidate and 40% safety data sets. One avenue of future work we will discuss is to optimize this partitioning to maximize the probability of success. **R1: How does Algorithm 1 update the iteration, or how to construct the set Θ? Is it agnostic to the ML used?** Algorithm 1 relies on the feasible set and optimization algorithm (OA) the user chooses to find candidate solutions. It is agnostic w.r.t. that OA. However, if the user chooses a poor OA that cannot find solutions, our approach will return NSF. We found that CMA-ES [45] works well. We will make it clear that the ability of our algorithm to find a fair solution depends on the user's choice of OA.

R1: The word 'fair' is problematic. Whether a solution is fair depends on more than the statistics of the model. Thank you for pointing out our imprecise wording. We will clearly differentiate between ensuring that models are fair, and ensuring that fairness constraints defined by the user are satisfied. We do the latter, and will not claim the former. It is the user's responsibility to provide a $g(\theta)$ that captures their notion of fairness for the application at hand; if the user's definition does not sufficiently capture fairness, then the solutions RobinHood produces will not either. RobinHood is designed to give the user flexibility in providing fairness definitions that capture domain knowledge.

R1: In line 261, fairness is mentioned without a proper explanation of why that is fair. We will make our writing more clear and rigorous in what we mean by fair in this experiment. Our definition is just one choice; RobinHood can be applied with other definitions of fairness the user finds more relevant.

R1: Limitations regarding the ability to "satisfy multiple criteria." Thank you for pointing this out. We will clarify that our results do not contradict the references you provide, and discuss the theoretical limitations. RobinHood returns NSF when impossibilities such as conflicting fairness constraints exist.

R2: Col 3 in figs suggests that baseline algs approach the same failure rate as RobinHood given enough data. Any insights? The fairness-unaware baselines only try to maximize expected reward. When reward maximization and fairness are nonconflicting, there can exist fair high-performing solutions. When *only* the high-performing solutions are fair, the failure rate of reward maximization algorithms should decrease as more data is provided. Note that when reward maximization and fairness *are* conflicting, e.g., in the skewed proportions experiment, the failure rates of the unfair baselines do not diminish. Importantly, while the baselines might be fair in some cases, unlike RobinHood, these approaches do not provide fairness guarantees.

R3: Regarding "no mention of the support assumption". This is captured by Assumption 4 for Thm 2, but is something we should, and will, discuss around (2) in the supplemental. **R3: Wouldn't it be ideal to return a uniform random policy as the solution rather than an NSF?** If no fair policy exists, RobinHood returns NSF. The user has control over what to do in this case. For some domains, deploying a known fair policy, (e.g., uniform random) may be appropriate; for others, it might be more appropriate to issue a warning and deploy no policy.

R3: Why is inflateBounds needed to compute the candidate utility but not when certifying fairness? The candidate selection method (CSM) searches for a solution that will pass the safety test (ST), which requires testing multiple solutions. Essentially, the CSM is performing multiple comparisons with one data set, resulting in over-estimation of its confidence that the solution it picks will pass the ST. This results in RobinHood frequently returning NSF. Inflating the width of the confidence intervals in the CSM is an effective remedy. Note that this multiple comparisons problem does not impact the ST, which only tests one solution, and so does not invalidate our theoretical guarantees.