We thank all reviewers for their valuable comments and suggestions. We respond to each comment below.

Contribution, novelty and technical strength. This section mainly responds to common concerns raised by reviewers. The MSSR model seems like a “minor" generalization of the standard ski rental; however, MSSR requires the skier to make a *two-fold* decision, i.e., *when* to buy and *where* to buy. This allows more heterogeneity in skier’s options and makes MSSR a more challenging problem that requires new design of online algorithms to achieve desired performance, especially with ML Advice. Finally, MSSR is a more general modeling framework for online algorithm design than the standard ski rental problem. Many new applications (see the supplementary material) can be modeled with MSSR.

While we follow the recent direction of online algorithms design with ML advice (e.g., Lykouris et. al., ICML 2018, Purohit et al. NeurIPS 2018, etc.), the MSSR requires us to design online algorithms by carefully crafting the specific structure of the problem which are different from existing literature, with a substantially different set of proof techniques. In particular, we would like to highlight that the probability distribution functions for the randomized algorithm is carefully designed to achieve a solid performance guarantee in terms of robustness and consistency metrics. Though the structure is used in previous work, we believe that our randomized algorithm analysis with new distributions provides the robustness and consistency results in a more systematic manner. Further, at the beginning of Section 3.3, we emphasize that a straightforward extension of the existing distribution used for standard ski-rental fails to guarantee solid robustness and consistency. This observation further clarifies the significance of the theoretical contribution.

Finally, as for the significance of the proposed algorithms in practice, our experimental results in Figure 6 demonstrate that the online algorithms with ML advice can effectively resolve major drawbacks of classic online algorithms. In other words, although online algorithms with ML advice and certain hyperparameter may not outperform pure online algorithms, it is always possible to find the right hyperparameter such that the performance of online algorithm with ML advice is better. This further demonstrates the importance and novelty of our proposed online algorithms with ML advice in the sense that the ML predictions needs to be incorporated in a judicious manner.

Assumption on the model. This section mainly responds to concerns raised by Reviewers $\#2$ and $\#3$. In this paper, we consider the basic setting of MSSR where the skier must choose one shop immediately after she starts the skiing and must rent or buy the skies in that particular shops since then. This basic setting occurs in many real-world applications, e.g., cloud computing systems as explained in the footnote on page 3.

Beyond this basic setting, there are several extensions of MSSR that can be considered, e.g., (i) MSSR with *switching cost* (MSSR-S), i.e., the skier is able to switch from one shop to another at some non-zero costs; and (ii) MSSR with *entry fee* (MSSR-E), i.e., there is an entry fee for each shop and no switching is allowed. In MSSR and MSSR-E, the skier needs to answer two questions *at the very beginning*: *where* to rent or buy, and *when* to buy the skis. In MSSR-S, the skier is able to decide where to rent or buy the skis *at any time*. It can be argued that MSSR-S and MSSR-E can be equivalently reduced to MSSR, e.g., switching happens only when buying (Ai et. al. 2014, see [1] in the paper). To that end, our proposed online algorithms with ML advice can be extended to these models with some minor changes in the constant terms. We will add discussions on this generalization in final version.

Lemma 1 and Algorithm 2. This section mainly responds to concerns raised by Reviewer $\#2$. In this paper, we focus on the basic setting of MSSR and design the corresponding learning-aided online algorithms with desired performance guarantee. The key motivation is two-fold: (1) to keep the core competency of online algorithms, i.e., performance guarantee against worst-case, which is characterized by *robustness*; and (2) to achieve a provably improved performance if the accuracy of ML-predictor is satisfactory, which is characterized by *consistency*. The hyperparameter $\lambda \in (0, 1)$ is a design parameter that determines the confidence level of the ML-predictors in the online algorithm. This provides a more powerful approach compared to traditional competitive algorithms that either does not rely on prediction at all, or *fully* rely on the predictions. With proper tuning of λ, one can achieve the “best of both worlds" paradigm with $\lambda \to 0$ represents full trust on ML and $\lambda \to 1$ indicates no trust at all.

Under this algorithmic framework, BDOA is the best deterministic online algorithm in the basic setting. Since our algorithm is evaluated using two criteria, i.e., robustness and consistency, one interesting problem is to investigate the optimality of an algorithm, which leads to the consideration of the Pareto optimality. In other words, if an algorithm A is Pareto-optimal, then there is no other algorithm that can achieve a better consistency (resp., robustness) than A without sacrificing the robustness (resp., consistency). We did not claim the Pareto optimality of our proposed online algorithms. Investigating their Pareto optimality or designing new online Pareto optimal algorithms with better robustness and consistency tradeoff is an interesting future direction. However, our proposed algorithms exhibit desired consistency and robustness performance, and are evaluated via extensive numerical evaluations including real-world datasets. To that end, Algorithm 2 does not fully equal to BDOA as $\lambda \to 1$ in our algorithm design. As $\lambda \to 1$, we reduce the impact of ML predictions on the algorithm to the minimum. Algorithm 2 can be easily generated to BDOA at $\lambda = 1$ with a minor change in the algorithm description. We apologize for this confusion and will make it clear in final version.

Proof sketch. We will add and clarify the proof sketch to highlight the technical contributions. Thanks Reviewer $\#3$.