A writeup of your process for tuning the parameters. Include one chart per parameter with parameter value on the x-axis and policy quality on y-axis (average score on 20 runs). For each step be sure to indicate the other parameter value you used (tuned or untuned).

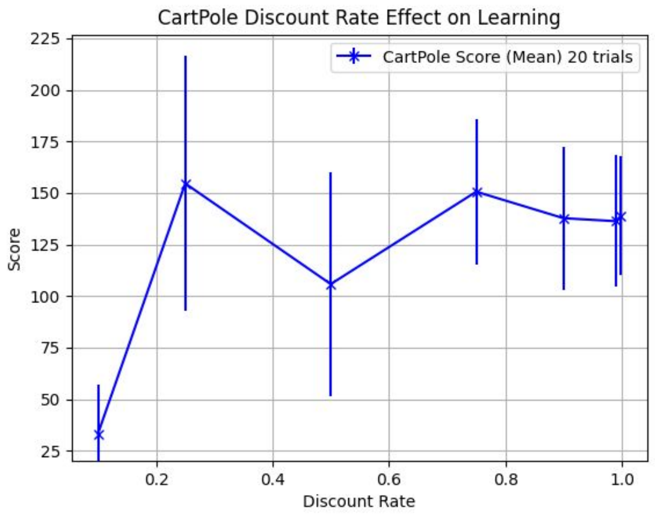
Include a final parameter setting and the average score it produced across 100 runs.

I first played around with some parameters to get a sense how jittery the results are. Turns out, there is quite a large standard deviation given a learned Q table. I took a look at my learned Q table and states, it seems there was trouble in expanding to search for all the states. Perhaps limiting the bins for position vectors and expanding bins for velocity vectors would help? But that is outside of scope of the assignment. I tried to increase the random action rate, but that only made things worse. I guess if the initial position is always the same and you only need to get to 200 rewards (time steps), you can just really explore the state space around that vicinity and not bother with learning how to recover if we end up sliding too far one way, because my reinforcement learning model definitely did not learn that.

For fine-tuning, I did one dimension at a time. I probably should have expanded to use as many values of each run as possible that are within 1 standard deviation of each other in terms of final mean scores, but for time-constraint, I picked the best and used that on the tuning for the next dimension. The tuning and graphs for each dimensions are below and the final model parameters are:

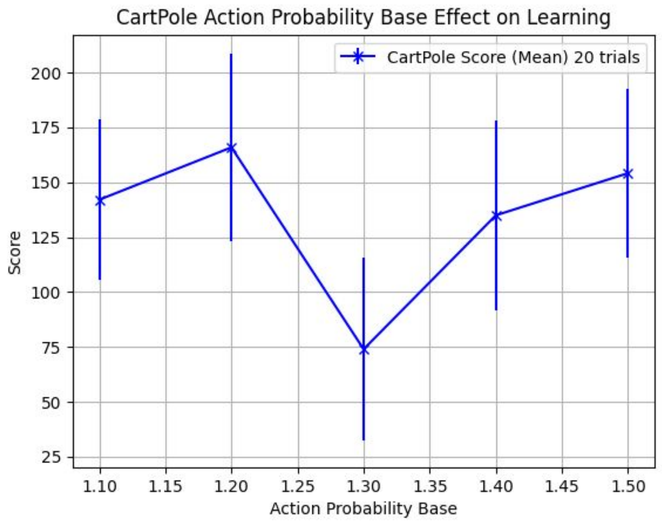
***discountRate*: 0.75**
***actionProbabilityBase:* 1.2**
***randomActionRate*: 0.01**
***learningRateScale:* 0.05**
***binsPerDimension:* 8**

Where the final accuracy over 100 runs where 168.19 +/- 35.87. It seems this is well below what is considered “solved”, which is a score above >195 with far lower standard deviation.



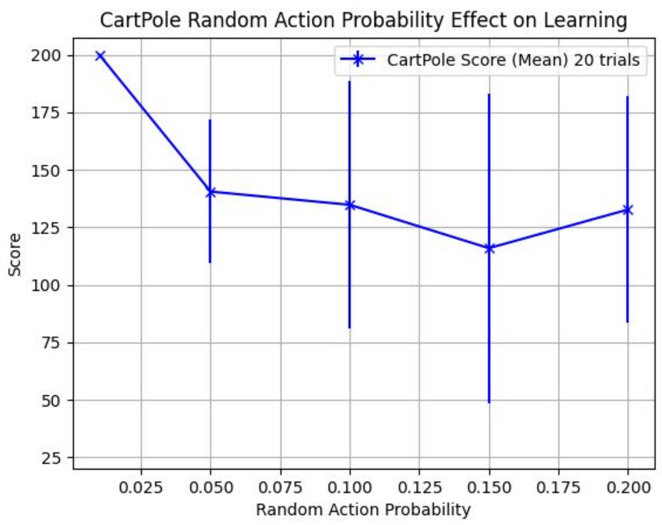
actionProbabilityBase: 1.25
randomActionRate: 0.1
learningRateScale: 0.01
binsPerDimension: 8

These values are obtained by ball-parking with initial random exploration to get the lay of the land.

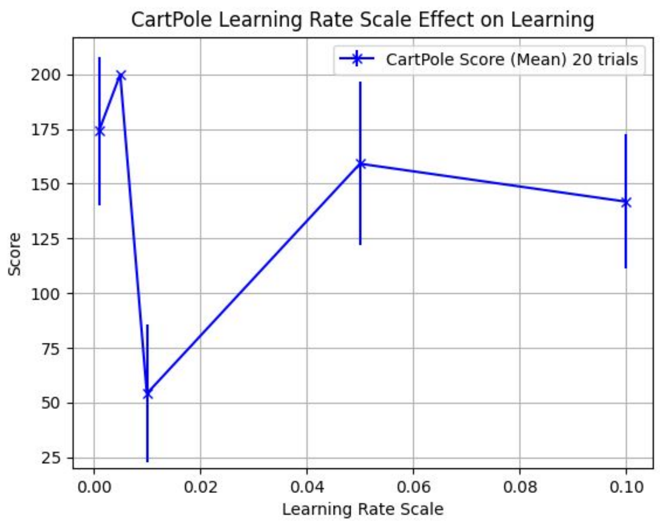


***discountRate*: 0.75**
randomActionRate: 0.1
learningRateScale: 0.01
binsPerDimension: 8

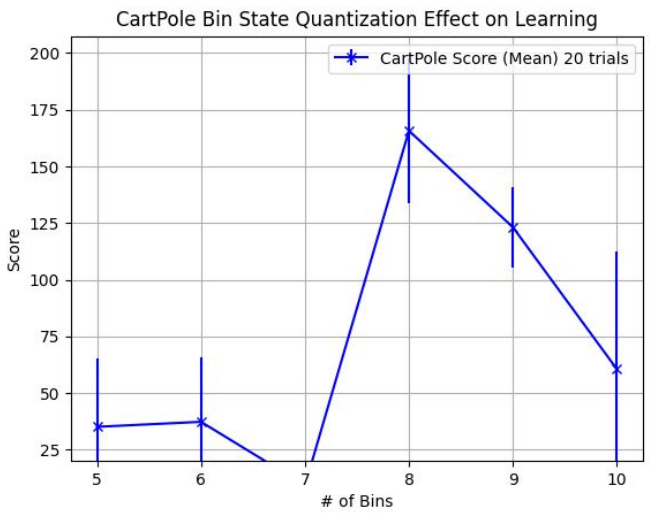
Here we locked in the discount rate from previous. While we probably could try out more discount rates given how close results were, for time-expediency, we just picked the best



***discountRate*: 0.75**
***actionProbabilityBase:* 1.2**
learningRateScale: 0.01
binsPerDimension: 8



***discountRate*: 0.75**
***actionProbabilityBase:* 1.2**
***randomActionRate*: 0.01**
binsPerDimension: 8



***discountRate*: 0.75**
***actionProbabilityBase:* 1.2**
***randomActionRate*: 0.01**
***learningRateScale:* 0.05**