Observe what you see with the agent's behavior as it takes random actions. Does the smart cab eventually make it to the destination? Are there any other interesting observations to note?

At this point the smart cab might make it to the destination, but since all actions are taken randomly, it would be because of luck. Regarding to other interesting observations, the smart cab is completely naive at it would try to do illegal turns or go straight through a red traffic light. The smart cab knowledge of what's going out in it's environment is null, and that's why it doesn't care/prefer doing the best actions to arrive to destination.

What states have you identified that are appropriate for modeling the smart cab and environment? Why do you believe each of these states to be appropriate for this problem?

The appropriate states for modeling the smart cab and environment consist of a tuple built up by the following information:

- $Light \in \{green, red\}$: Traffic light color at current intersection. It is appropriate for this problem given that it allows the car to decide if what it should do in a given intersection. Not knowing about it would lead to try invalid movements (penalizing the vehicle and losing time) or crashes.
- $Next \in \{left, forward, right, None\}$: Direction suggested by planner that will lead us to the destination point. It is appropriate for this problem given that it gives the smart cab an idea of where the destination point is located. Without it, the smart cab has no idea where it should go to.
- $Oncoming \in \{left, forward, right, None\}$: Relative direction of other vehicle at current intersection.
- $Right \in \{left, forward, right, None\}$: Direction that vehicle at the right (if any) is going to take after current intersection.
- $Left \in \{left, forward, right, None\}$: Direction that vehicle at the left (if any) is going to take after current intersection.

The last three elements of the tuple are appropriate for this problem, given that if the smart-cab knows what decisions other cars are taking, it can decide to find a better path to arrive to destination. Not knowing where other cars are moving could lead to car crashes ar going through busy routes.

The $Deadline$ sub-state is not used for modeling the environment or smartcab given that it would increase the size of the space in about 30 times (or more, depending on the dest. point).

On the other hand, the deadline sub-state provides us some information about how fast we should arrive to the destination point. Nevertheless, we want to arrive always in the least amount of time, so this information is not very relevant and the benefit of this information is much smaller than the disadvantages of increasing our space.

How many states in total exist for the smartcab in this environment? Does this number seem reasonable given that the goal of Q-Learning is to learn and make informed decisions about each state? Why or why not?

For the model description given above, In total, there are 512 possible different states, given by:

$$2\,(light) \, * \, 4\,(oncoming) \, * \, 4\,(right) \, * \, 4\,(left) \, * \, 4\,(next) \, = \, 512\,.$$

It seems like a big number of states given that the agent has to explore every single one at least one time (preferably more than one), in order to arrive to the global optimum. Nevertheless, given that there are 100 iterations of different trips from origin to destination point, and that for each trip the agent might take 30 or more steps, there is enough time for the agent to visit every space state at least 6 times.

What changes do you notice in the agent's behavior when compared to the basic driving agent when random actions were always taken? Why is this behavior occurring?

As the agent tries to get to the destination point, the agent reaches the destination with shorter trips. This means that as time goes on, the agent starts making more accurate predictions on what are the best actions to make, and it'' decisions are less random.

This happens because for every single transition that the agent could make from a state *s,* it knows which action *a* will give the highest reward to its behaviour. And every time the agent explores a new state or arrives to a previously visited state, its utility for that given state gets more and more accurate. It is important to notice that even though the smart cab has leaned for a long time, there is a chance that it makes a random choice, so it can explore further the total space of the environment.

Report the different values for the parameters tuned in your basic implementation of Q-Learning. For which set of parameters does the agent perform best? How well does the final driving agent perform?

Each parameter used to tune the basic implementation of Q-Learning could take any of the following values:

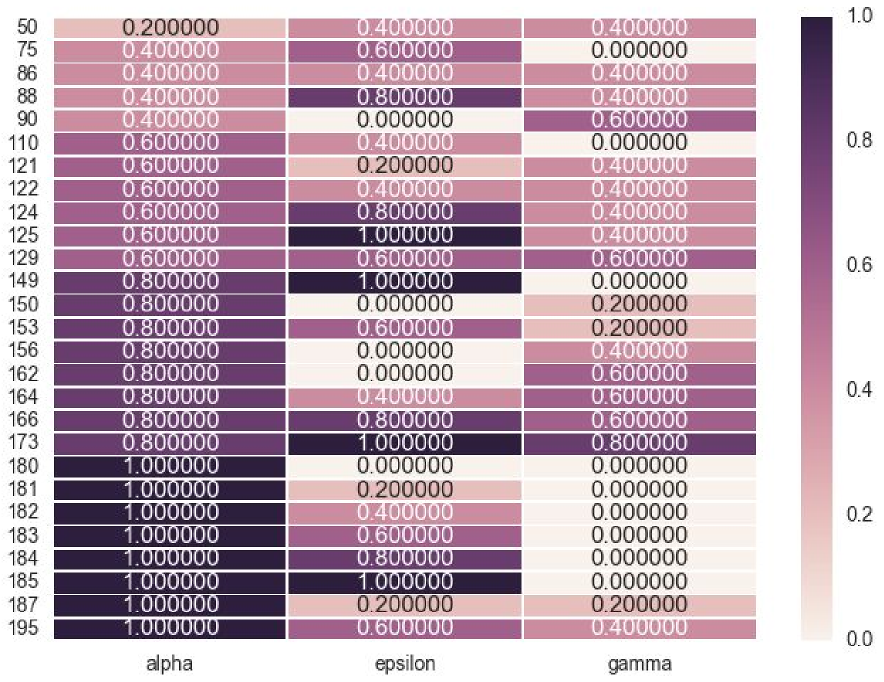
0.0
0.2
0.4
0.6
0.8
1.0

Thus, giving a total of 216 different combinations. Training the car was done 20 times with a number of 100 trials every time, for each distinct set of parameters alpha, beta and gamma. It was done in this way in order to have a normalized dataset that wouldn't depend on the randomness of the origin-destination configuration that the simulation creates. Results for the data sampled with each combination of parameters was averaged.

In order to find which set of parameters make the agent perform best, it is necessary to figure out what does 'best' mean. For this case, we will say that best means that the smart cab learns fast, arrives to destination in every single trip and that it doesn't take too long traveling to the destination. According to this, 6 different parameters were measured for each simulation, with distinct parameters:

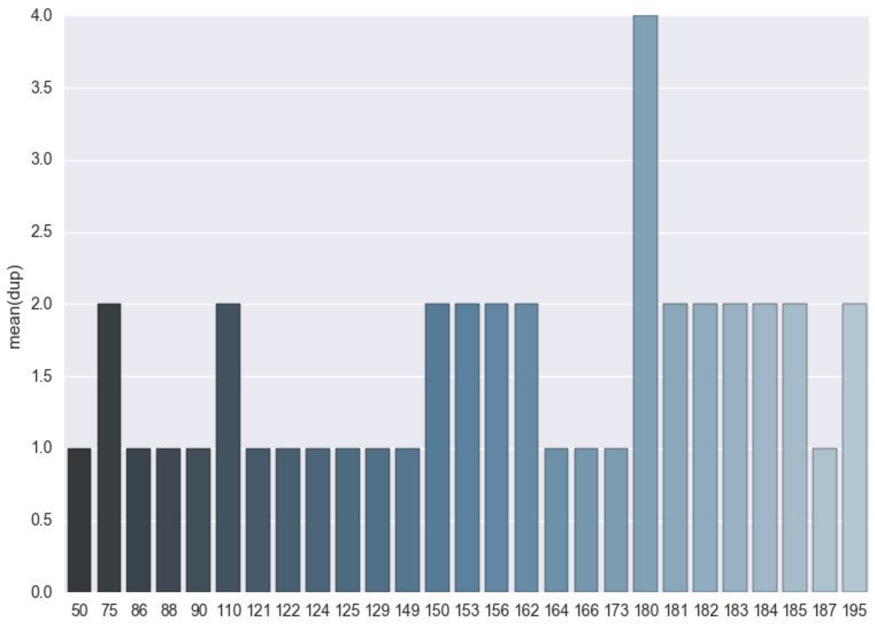
- ***Average percent to deadline:*** What is the average proportion of time vs deadline required to arrive to destination (for all 100 iterations). The lower, the better.
- ***Average percent to deadline last 10:*** Same as above, but only for last 10 iterations.
- ***Successes:*** How many times did smart cab arrive to destination (from all 100 iterations). The higher, the better.
- ***Successes last 10:*** Same as above, but only for last 10 iterations.
- ***Scores:*** What were the total scores during all 100 iterations. The higher, the better.
- ***Scores last 10:*** Same as above, but for only last 10 iterations.

The idea of using a "last 10" measure for each of the global ones is that it leads us to know how well our trained model performs. Additionally, for each one of these parameters, the top 7 parameter configurations was obtained and is shown below (each parameter configuration has a unique id shown on the left of the plot):



	alpha	epsilon	gamma
50	0.200000	0.400000	0.400000
75	0.400000	0.600000	0.000000
86	0.400000	0.400000	0.400000
88	0.400000	0.800000	0.400000
90	0.400000	0.000000	0.600000
110	0.600000	0.400000	0.000000
121	0.600000	0.200000	0.400000
122	0.600000	0.400000	0.400000
124	0.600000	0.800000	0.400000
125	0.600000	1.000000	0.400000
129	0.600000	0.600000	0.600000
149	0.800000	1.000000	0.000000
150	0.800000	0.000000	0.200000
153	0.800000	0.600000	0.200000
156	0.800000	0.000000	0.400000
162	0.800000	0.000000	0.600000
164	0.800000	0.400000	0.600000
166	0.800000	0.800000	0.600000
173	0.800000	1.000000	0.800000
180	1.000000	0.000000	0.000000
181	1.000000	0.200000	0.000000
182	1.000000	0.400000	0.000000
183	1.000000	0.600000	0.000000
184	1.000000	0.800000	0.000000
185	1.000000	1.000000	0.000000
187	1.000000	0.200000	0.200000
195	1.000000	0.600000	0.400000

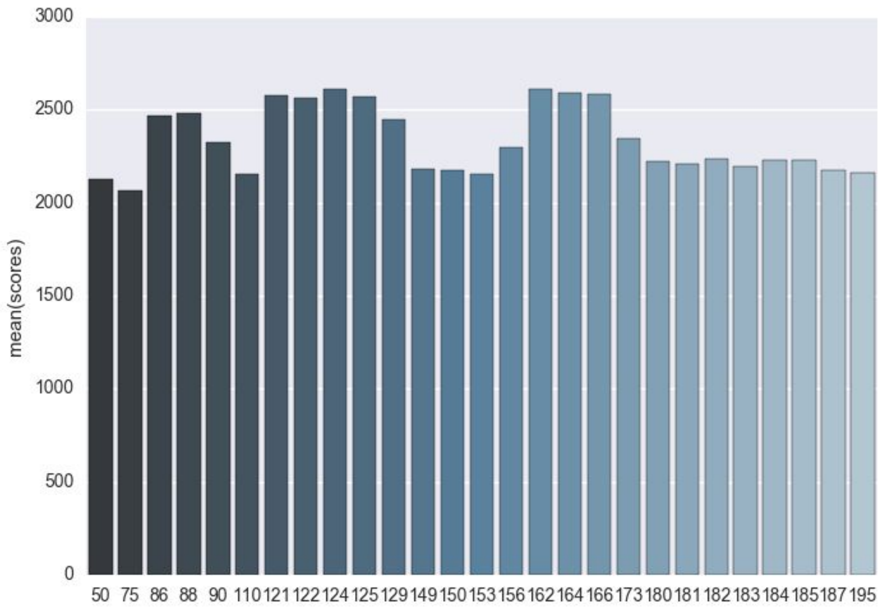
Some configurations appeared in the top 7 of more than one of the 6 measures taken:

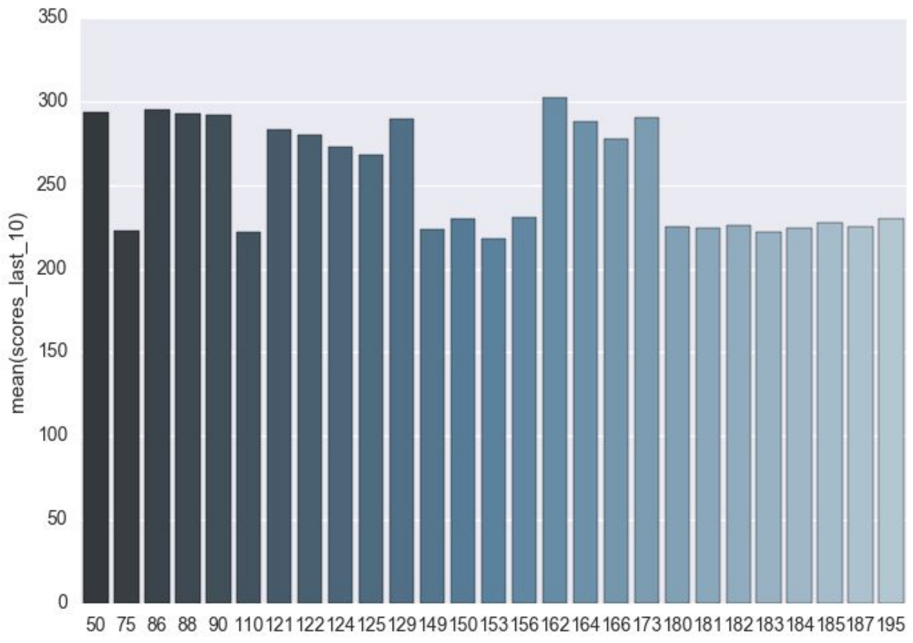


As seen, configuration 180 seems to be a well balanced configuration, where most of the measures taken are in the top 10 of all configurations. The measures where it excels are:

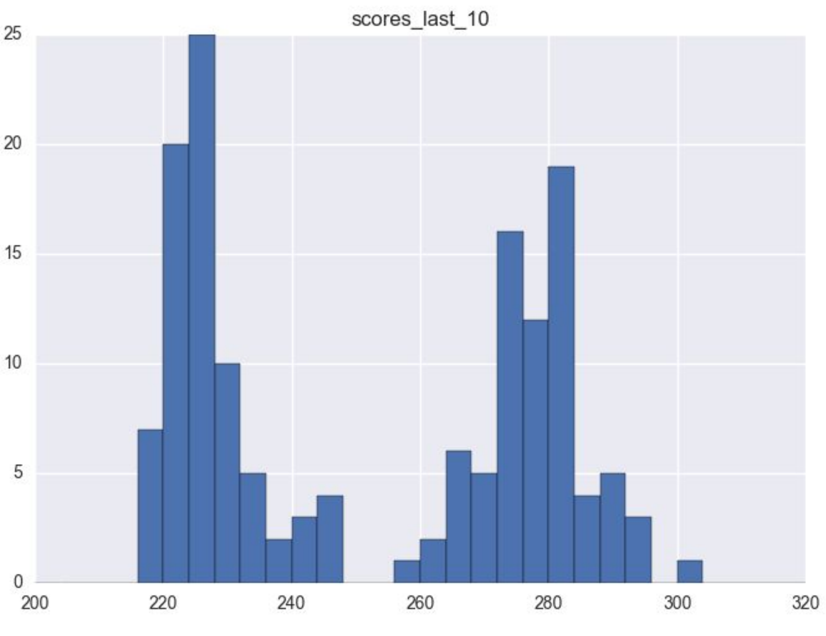
- ***Average percent to deadline***
- ***Average percent to deadline last 10***
- ***Successes***
- ***Successes last 10***

Other configurations that appear 2 times in the top 7 seem also valuable, but not as much as config. 180. Let' see how it performed on the other two measures taken (scores and scores last 10):





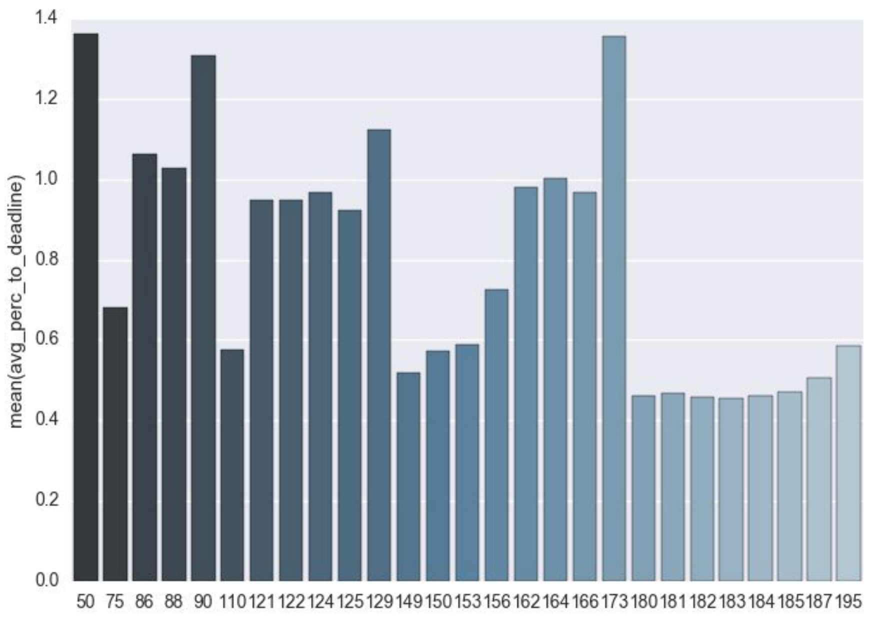
It can be inferred from the visualizations that the scores for the configuration #180 are lower than most of the other configurations, both for all the total scores and for the last 10 iterations score. And as shown in the following image, the score for the configuration #180 is in the lowest range of values for the positive scores of the whole set of trials.

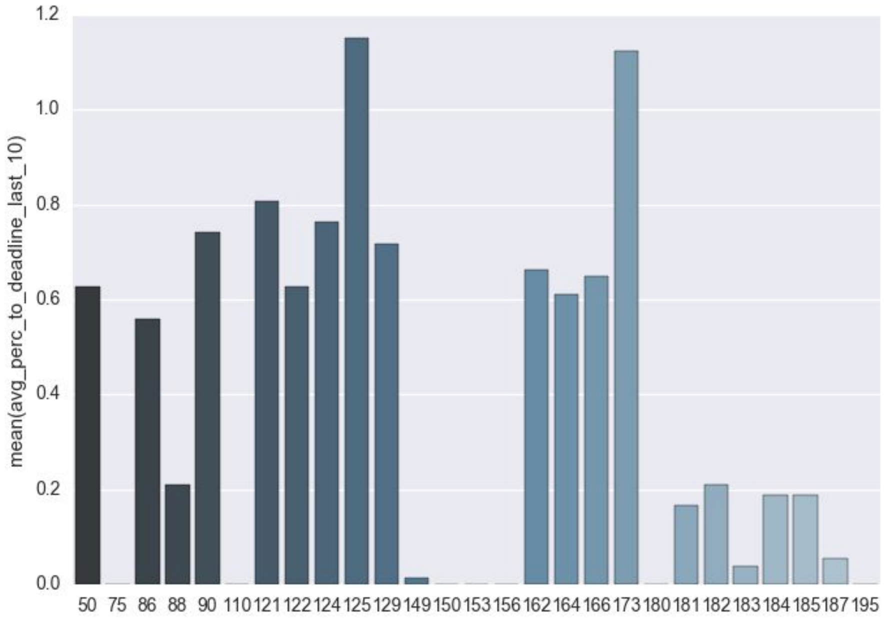


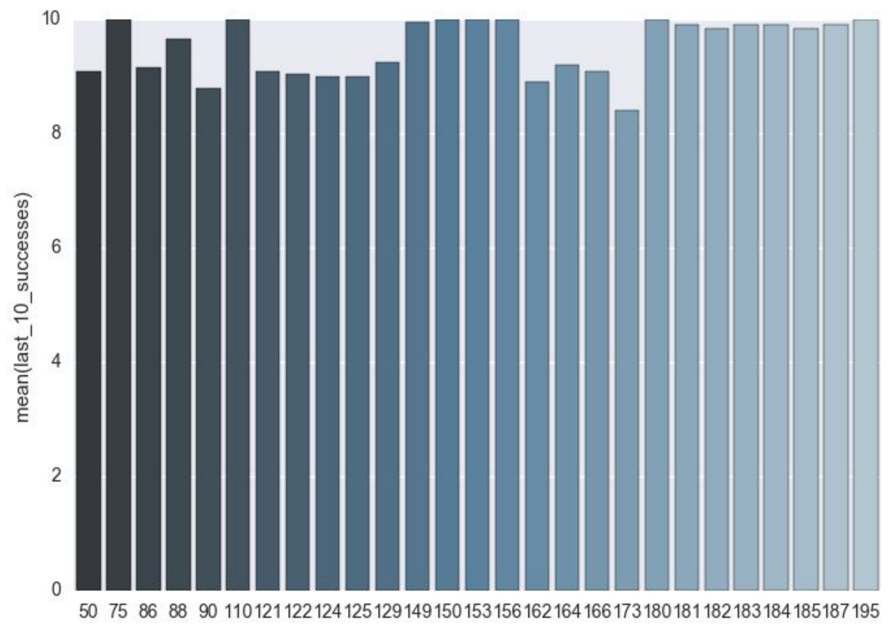
Nevertheless, what's going on here is that this configuration is taking the least average percent to deadline, making it earn less rewards than other configurations which might make mistakes. The cost (negative reward) for a mistake is -1 or -0.5, while the reward for moving correctly is 2.

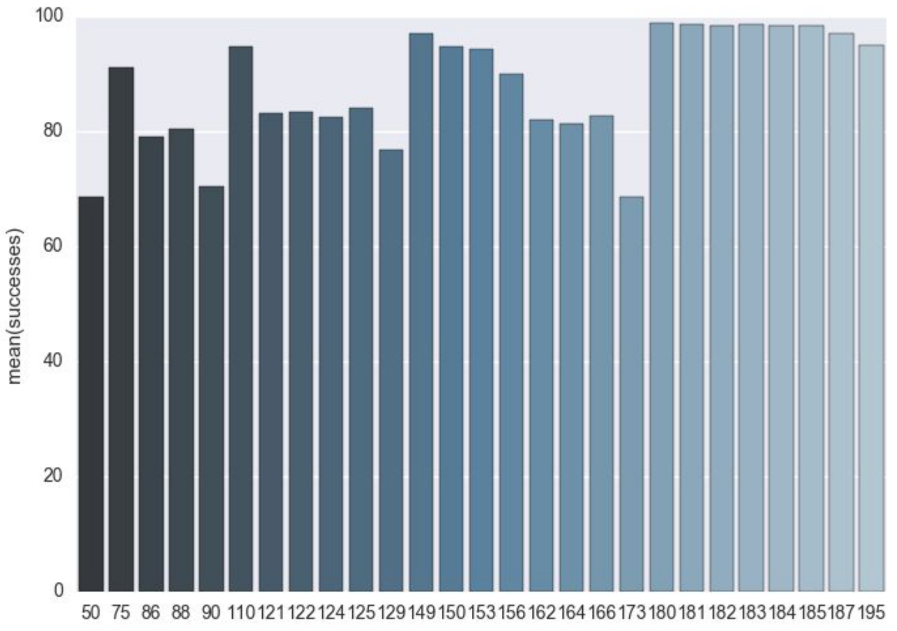
So as a smart car I might follow loops so that my positive rewards outbalance the negative ones, making me gain an overall higher score.

Comparing configuration #162 (which had the highest score for both measures) with our favourite #180, and with the following graph, we realize that having a higher score doesn't really mean performing better. The proportion of time taken to reach destination is much higher for the configuration #162, and it does not succeed to arrive to destination for all the last 10 iterations, while configuration #180 scores the highest at both measurements. This just tells us that the score measure only explains if a smart car with given configuration is not performing too bad, for instance, by getting a score of 0.









This leads us to affirm that according to the definition of 'best' mentioned above, the best configuration is the one that arrives the fastest, learns fast and succeeds in most of the iterations. For our scenario, configuration #180 with params: $\alpha = 1,\ \varepsilon = 0,\ \gamma = 0$ is the best configuration.

QUESTION: Does your agent get close to finding an optimal policy, i.e. reach the destination in the minimum possible time, and not incur any penalties? How would you describe an optimal policy for this problem?

An optimal policy for this problem would be: the smart cab reaches destination always, it reaches in the least possible time, and it doesn't have any accidents or tries invalid moves (it does not incur any penalties).

Yes, the agent gets close to finding an optimal policy. For this case the optimal policy is reaching the destination in the least possible time, and is shown in the avg. percent to deadline graphs. It also reaches the point where all its iterations are successful (arrives to destination). Nevertheless, not incurring into any penalties is not a parameter that is being taken into account, so this piece of information is what the current model lacks, not that it does fail, but we don't know.