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

When taking random actions the smartcab does sometimes reach the destination. With 100 trials the smartcab arrived at the destination 31 times. Arriving about one out of three times seems pretty high considering we are just taking random actions. One interesting observation is that the world is is bounded to a fairly small area with the Smartcab wrapping around to the other side if proceeding beyond the bounds. This explains why we have such a high success rate even with taking random actions since the bounds keep the Smartcab from wandering too far away from the destination.
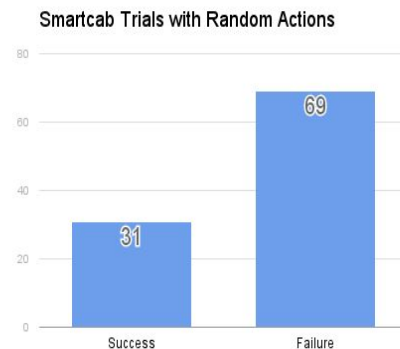
Another note is that failures are due to a hard cap on the deadline at -100. If allowed to roam indefinitely the Smartcab would eventually randomly land on the destination, especially since we are working with such a small bounded world.

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

I chose the following to use as parameters for modeling the smartcab state:

State Parameter	Possible Values	Justification
next_waypoint	Left, Right, Forward	Next waypoint is included because the agent needs to be guided to the destination.
light	Red, Green	Light was included because the agent needs to be able to stop at red lights, with the exception of making a right turn.
oncoming	None, Left, Right, Forward	Oncoming was included because the agent needs to be able to determine if it is safe to make a left turn.
left	None, Left, Right, Forward	Left was included because the agent needs be able to safely yield to forward traffic from the left.
right	None, Left, Right, Forward	Right is included for robustness. While not needed for right-side-of-road driving rules, it would prove useful in other countries. Allso it would be cheating a little to remove it because I'm inferring driving rules where the learner should figure those rules out.
deadline	Omitted to reduce state space	Even though deadline is removed from the state, it is still there implicitly because the smartcab does not get the large reward if not reaching the destination on time.

Some of these inputs could be condensed, for example *oncoming* could be condensed to True or False, however we could be limiting the agent's ability to learn some traffic rules. One such rule would be that it is okay to go right on red if there is nobody going forward from the left. Because of this I chose not to condense those inputs.

Another discarded idea was to condense light and inputs to contain right of way logic in the following way.

waypoint	Left, Right, Forward
ok_right	True, False
ok_left	True, False
ok_forward	True, False

=>

waypoint	Left, Right, Forward
ok_waypoint	True, False

This would drastically reduce the state space, however it would also be imposing those logic rules onto the learner and making the smartcab less able to adapt to different environments such as foreign countries where the rules are different.

OPTIONAL: 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?

The total number of states is the combination of all of the state parameters: next_weightpoint x light x oncoming x left x right = 3 x 2 x 4 x 4 x 4 = 384

There are 384 total states given the chosen state model.

With 384 states we have 384 x 4 = 1536 possible Q values, however when running a trial it seems unlikely that we need to know every value. A lot of those states will have negative reward, and since Q-Learning will guide the smartcab mostly to positive rewards, we will never see many of those states. Considering this, I believe 384 is a reasonable number to allow Q-Learning to make informed decisions.

QUESTION: 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?

At first the behavior of the behavior of the two agents is the same. Without knowledge of the world, the Q-Learning agent begins to explore randomly. During the second trial I start to see the agent actually stopping at about half of the red lights, and the agent successfully follows the waypoint several times, but still taking a lot of exploratory or random actions. On the third trial, I see that the agent has stopped at most red lights, and also followed the waypoint most of the time. I also see the agent still taking some exploratory actions for a state even though the agent had previously taken the correct action. From there the agent continues to improve, making less violations and following the waypoint more consistently as the trials go on.

Given the improved behavior over random actions we can infer that the Q-Learning Smartcab is able to estimate expected reward from transitions and find a policy that considers these rewards.

QUESTION: 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?

To test performance against different parameter combinations, I implemented a basic grid search. For evaluating performance I chose look at the success rate among the final 10 trials in a set of 100 trials. This way we are not penalizing a good parameter tune if it does a lot of learning early on, but does well in the final trials. Similarly, a parameter tune could do well early on but never improve on itself.

The grid search was then repeated to make sure the algorithm performed well among many sets of trials. (100 x 100 trials). Results in the table are averaged from all trials. Best parameters are highlighted in green.

Epsilon	Alpha	Gamma	Initial Value	Mean Reward per Trial	Success Rate (%)
1/(t+1)	1/(.8*ln(t+2))	0.9^t	1	22.201	99.3
1/(t+1)	1/(.8*ln(t+2))	0.8^t	1	22.312	99.1
1/(t+1)	1/(ln(t+2))	0.8^t	1	22.3215	99.0
1/(t+1)	1/(.8*ln(t+2))	0.9^t	10	22.328	98.9
1/(t+1)	1/(ln(t+2))	0.9^t	1	22.1365	98.6
1/(t+1)	1/(ln(t+2))	0.9^t	10	22.318	98.4
1/(t+1)	1/(ln(t+2))	0.8^t	10	22.595	98.2
1/(t+1)	1/(.8*ln(t+2))	0.8^t	10	22.4195	98.0
1/(ln(t+1)+1)	1/(ln(t+2))	0.8^t	1	22.104	96.0
1/(ln(t+1)+1)	1/(.8*ln(t+2))	0.8^t	0	21.834	95.3
1/(ln(t+1)+1)	1/(.8*ln(t+2))	0.9^t	1	22.4385	95.3
1/(ln(t+1)+1)	1/(.8*ln(t+2))	0.8^t	1	22.1525	95.2
1/(ln(t+1)+1)	1/(ln(t+2))	0.9^t	1	21.9755	95.0
1/(ln(t+1)+1)	1/(ln(t+2))	0.9^t	10	22.0485	94.6

1/(ln(t+1)+1)	1/(.8*ln(t+2))	0.9^t	0	22.0245	94.4
1/(ln(t+1)+1)	1/(ln(t+2))	0.8^t	10	22.0695	94.4
1/(ln(t+1)+1)	1/(.8*ln(t+2))	0.9^t	10	22.2185	94.3
1/(ln(t+1)+1)	1/(.8*ln(t+2))	0.8^t	10	22.2835	94.1
1/(ln(t+1)+1)	1/(ln(t+2))	0.9^t	0	22.033	94.1
1/(ln(t+1)+1)	1/(ln(t+2))	0.8^t	0	22.067	92.9
1/(t+1)	1/(ln(t+2))	0.8^t	0	9.879	37.5
1/(t+1)	1/(.8*ln(t+2))	0.8^t	0	9.7895	36.8
1/(t+1)	1/(.8*ln(t+2))	0.9^t	0	9.1685	33.5
1/(t+1)	1/(ln(t+2))	0.9^t	0	8.4895	29.9

Below we can see a more detailed look at how the best parameters perform over a set of 100 trials. The success line is a threshold line. Failure is indicated when deadlines fall on or below this line.
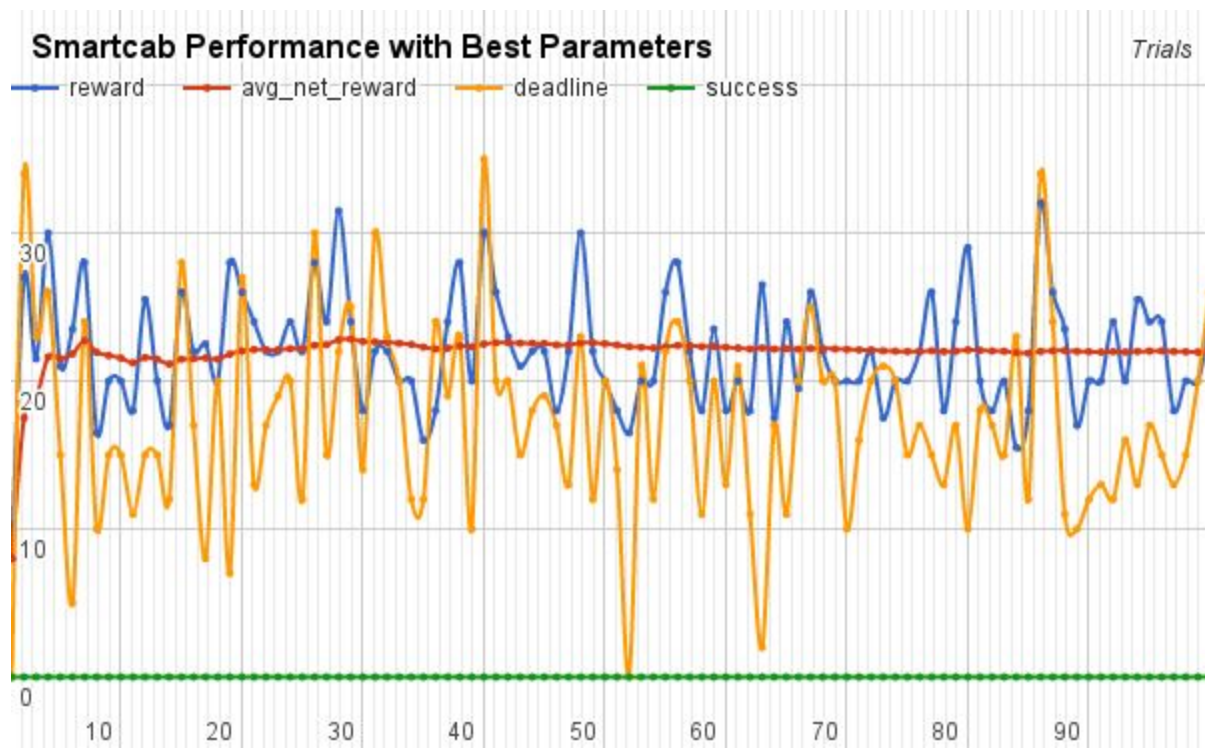
QUESTION: Does your agent get close to finding an optimal policy, i.e. reach the destination in the minimum possible time, and not incur any penalties? How would you describe an optimal policy for this problem?

For our applications here I think the optimal policy could be defined such that the smartcab does not run any red lights, does not get into any accidents, and makes it to the destination on time.

To determine If my agent is making traffic violations, I implemented a simple counter for each time the agent gets a negative reward. Again only the final 10 trials were considered for evaluating performance, and averaged over 100x100 trials.

Average Number of Penalties per Trial	Mean Reward	Average Success Rate
0.2217	22.32745	99.3%

On average we incurred 0.22 penalties per trial, and from the high success rate we can infer that the agent is reaching the destination in minimal time. From those metrics I believe my agent is close to finding an optimal policy.

Looking further into the mistakes I see that many are due to states that may not have been explored fully before. In the environment, there isn't a lot of traffic so the agent does not have many opportunities to see all the possible states.   In order to improve our policy, perhaps if we increased the number of dummy cars, the the agent would have more opportunities to explore those states early on. Another thing we could do is to allow the agent to continue exploring for more than 100 trials or simplify the state space. For example, when first deciding parameters to include in the state, I chose to keep the granularity of each input. However reducing those inputs to True/False would allow the agent to make good decision in general while losing the ability to discover more granular patterns such as the ability to take a right on red.

Sample State/Action Pairs for Traffic Violations

Action	Waypoint	Light	Oncoming	Left	Right
forward	left	green	forward	None	None
forward	left	green	left	None	None
forward	left	green	left	right	None
forward	left	green	None	forward	forward
forward	left	green	None	left	None
forward	left	green	None	None	forward
forward	left	green	None	None	left
forward	left	green	None	None	right
forward	left	green	right	None	None