In this project, I have considered 4 different Markov Decision Processes. Though, the requirements are only for 2 processes, I have considered so due to the following reasons:

1) First process that I have considered is a typical grid world scenario. I have leveraged the Frozen Lake 8 x 8 from Open AI gym for this purpose. As taught in the lectures a grid world is an ideal example of an MDP. The states, actions and rewards at each state can be both visualized and followed upon easily. At the same time, the Frozen Lake environment available in Open AI gym captures the stochasticity of the process where a particular action could result in undesired consequences. To me this scenario is simple enough to get the feet wet in MDP.

2) Next I chose the Taxi-v3 available in Open AI gym. Though, this is another grid world scenario the main drawback of Frozen Lake is its simple enough with only 64 states. The Taxi scenario on the other hand involves 500 states which makes the problem complex. Also, another reason for including two grid world scenarios is to compare the performance of different algorithms between a simple and a complex grid world scenario. In other words, it would be hard to judge the effectiveness of an algorithm based on the simple grid world scenario, so comparing the same with a more complex process would provide better insights.

3) Third I considered a Forest Management process mainly because I wanted to see how the different algorithms perform on a non-grid world scenario. Moreover, I felt like the number of states in this scenario can be increased to 1,000 or so to see how this affects the performance.

4) Lastly, I wanted to apply everything I have learned by creating a custom MDP process so that I can vary the parameters and compare the results. The main purpose of this exercise is to enhance my understanding of the MDP, and the associated algorithms used for analysis. Conceptually, this is similar to the NChain process available in Open AI gym. This process has 400 states and three actions are possible at each state: stay there, go forward or backward. There is no reward to stay in the same state, reward of less than 1 for moving backward and a reward of 8 for moving forward. The process also involves a huge reward at the end and small negative reward at the beginning. Also, I modeled this process to include stochasticity where an action could result in undesired consequences.

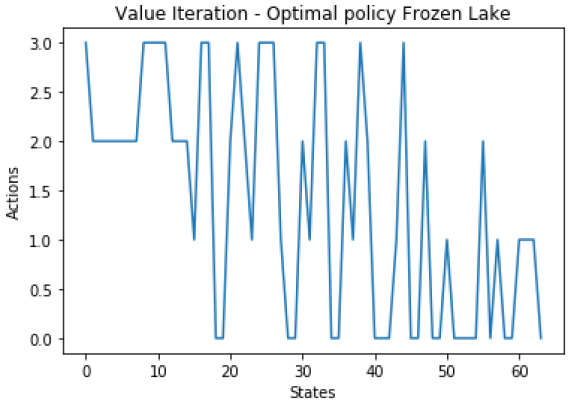
Value Iteration:

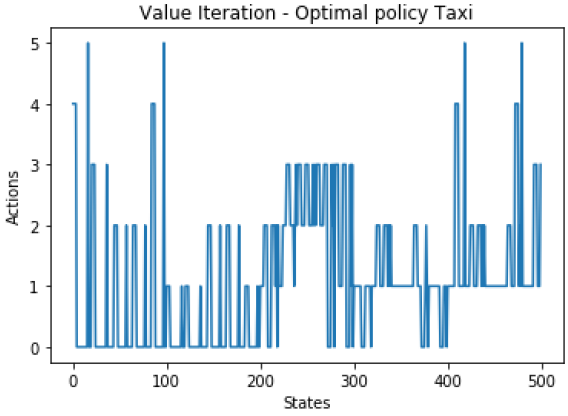
Value iteration is an iterative method where the MDP is run for each state and every action that's possible at that state to determine the maximum value that is yielded by an action at that state. If I look at the results of value iterations as shown below, I can see that between Frozen Lake and Taxi processes, Taxi takes more iterations to converge, hence takes more time as well. The fact that the average reward is negative in the Taxi case implies that both illegal drops and the number of steps to drop off a customer are big factors in determining the reward. It can also be inferred that though the values are converged, value iteration doesn't yield the best policy in case of the Taxi example. The Forest management on the other hand, converges very quickly, runs fast and returns a good average reward. I can attribute this performance to the fact that when an action is performed there are no unintended consequences unlike in Frozen Lake or Taxi. For example, in Frozen Lake an action to move up might not always move up but move right or left sometimes which is not the case in Forest management. If I look at how the average

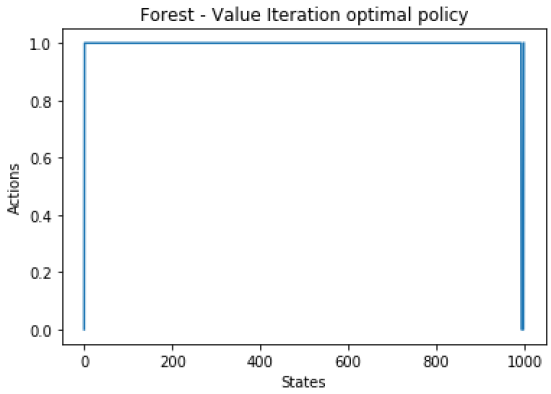
values of all states ramp up over the number of iterations, I see a similar pattern across the different processes. This means that all models achieve the best value fairly quickly.

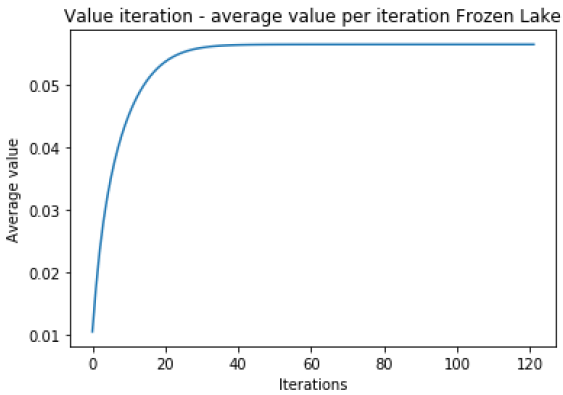
Value Iteration

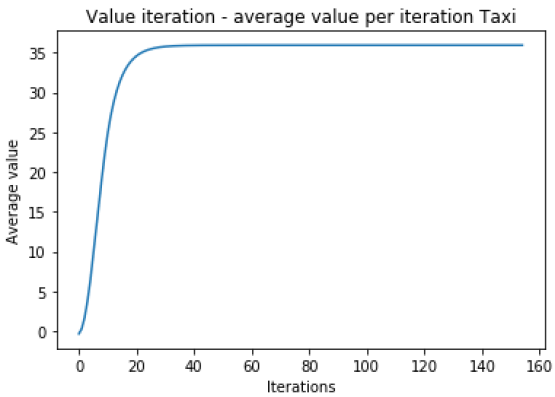
MDP	States	Iterations	Time (seconds)	Average Reward
Frozen Lake	64	122	0.149	0.01
Taxi	500	155	1.258	(1.18)
Forest Management	1,000	33	0.016	3.86
Custom	400	322	2.133	(1.00)

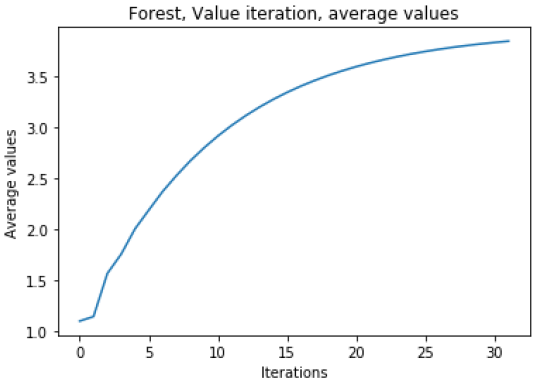










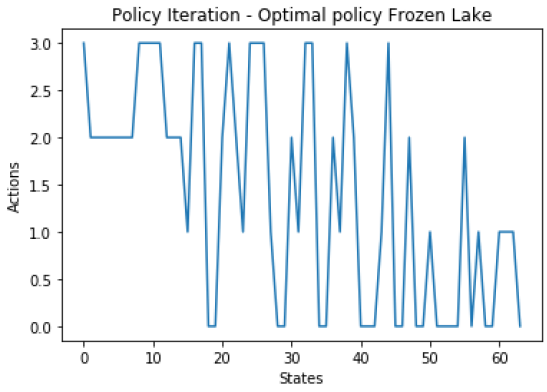


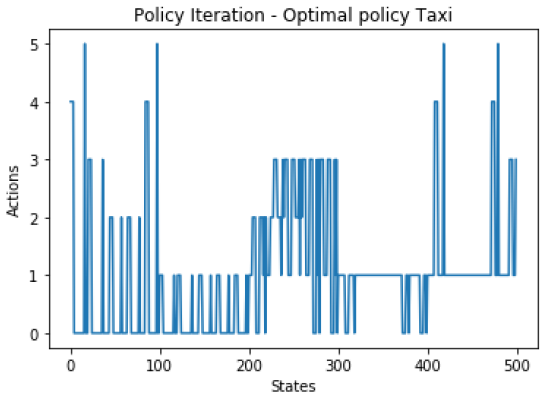
Policy Iteration:

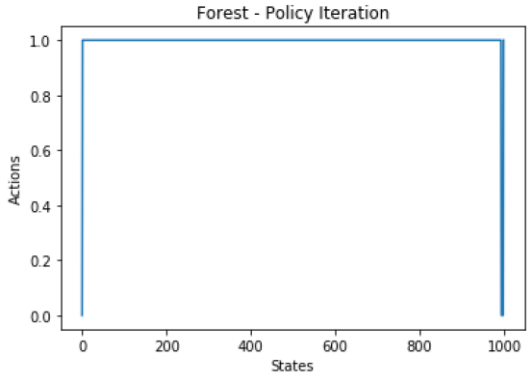
Policy iteration assumes a random policy, determines the values at each state for that policy until the values converge and then improves upon the actions at each state to determine the optimal policy. If I look at the values below, I can see that time taken for policy iteration is more than value iteration for the corresponding MDP. This is because many iterations are performed for each policy to enable the values converge. Also, the epsilon values I have chosen are very small which requires more iterations for the values to converge. However, the rewards obtained in the policy iteration are better than in value iteration. Though Policy iteration takes more time, it is guaranteed to have found the optimal policy since the search was done in the policy space whereas value iteration searches in the value space and then comes up with the policy for those optimal values. If I look at the average rewards for each iteration, I can see how the rewards keep increasing with the number of iterations. In fact, for the Taxi model, the rewards in earlier iterations were negative and show improvement with the number of iterations.

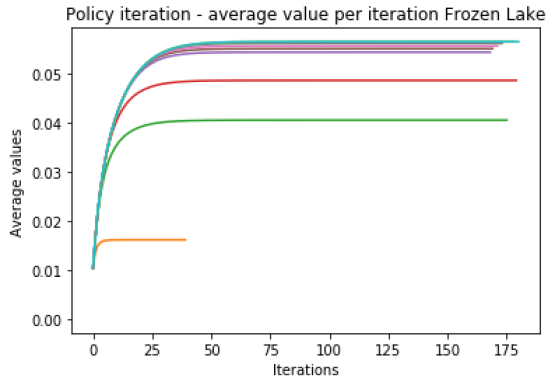
Policy Iteration

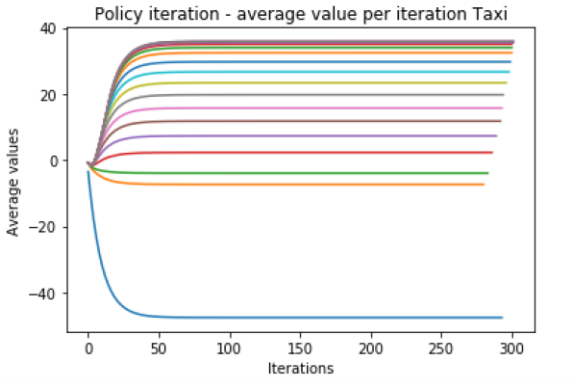
MDP	States	No. of Policies	Total Iterations	Time (seconds)	Average Reward
Frozen Lake	64	10	1,442	0.43	0.01
Taxi	500	18	5,325	7.84	(1.39)
Forest Management	1,000	6	594	0.09	3.96
Custom	400	2	171	0.70	79.83

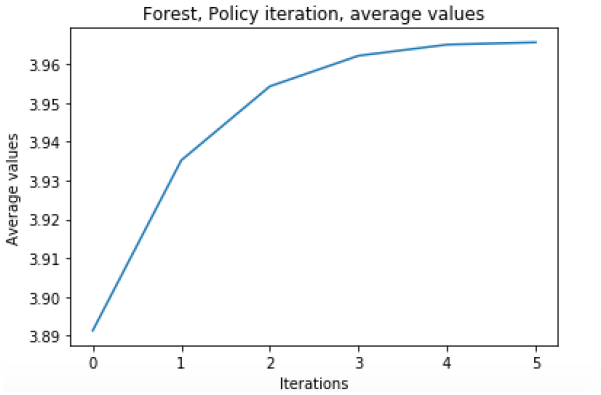












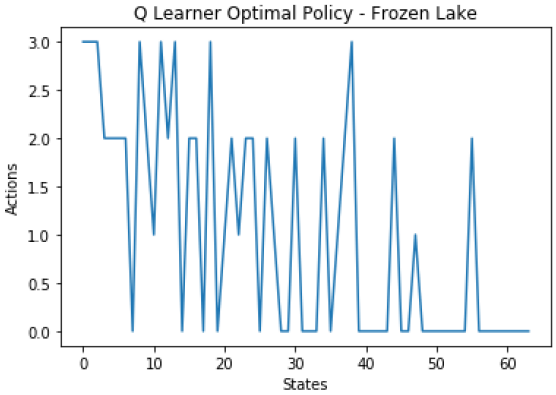
Q – Learner:

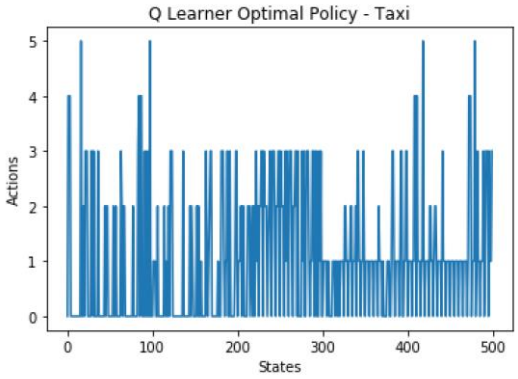
This is a model free reinforcement learning algorithm. The transition models that had a probability associated with moving to a new state from the current state for a specific action is not prevalent here. Instead, the model learns by taking an action, going to a new state and learning based on the rewards obtained in that state. As shown below the time taken to train the Q learner is bigger than in both Value and Policy iterations. However, once trained the model quickly learns the process and achieves its goal in fewer steps/iterations when compared to value and policy iterations. This is shown in the total iterations' column, where the iterations are just an average of all the steps taken by the learner to achieve its goal. Also, the Q learner performs well for all the cases except for the Taxi model. A possible reason for that is the number of steps over which the model was trained was not enough mainly because the model involves

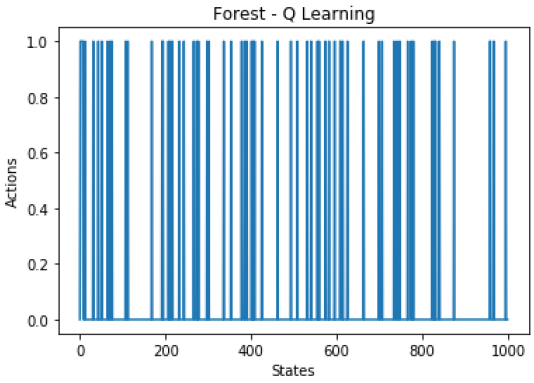
500 states, perhaps increasing those steps to over 10,000 or so would enable the learner to learn the process well.

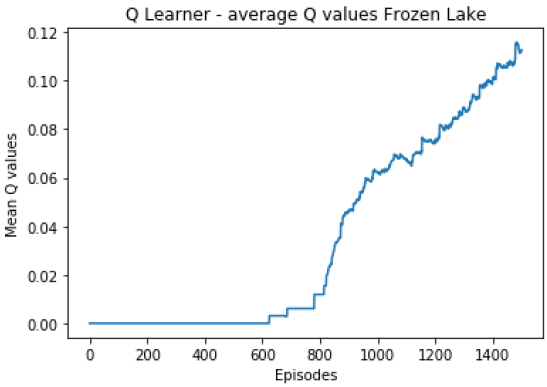
Q Learner

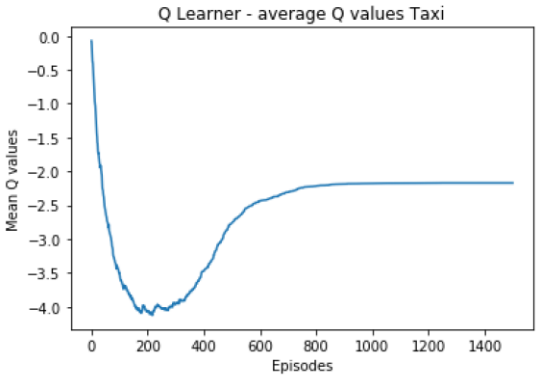
MDP	States	Total Iterations	Time (seconds)	Average Reward
Frozen Lake	64	26	1.95	0.0013
Taxi	500	12	9.55	(768.8340)
Forest Management	1,000	99	0.36	1,299.5000
Custom	400	399	2.14	6,118.6200

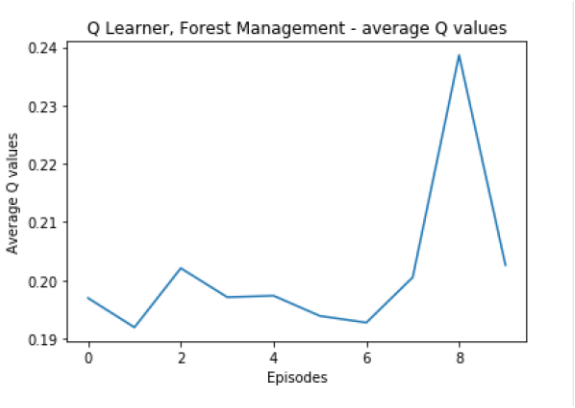












Comparison of Optimal Policies:

If I look at the optimal policy graphs between Value iteration, Policy iteration and Q-Learning, I can see the following behavior:

- **Frozen Lake:** The optimal policy does seem to agree between Value and Policy iterations. In other words, the best action to take at any state seem to be the same irrespective of the algorithm. However, in Q Learning the best action to take seems to vary towards the later states. The possible reason is that the model has learnt well from experience and thus able to choose the best action based on experience i.e. from the q values obtained for earlier states. However, in case of policy and value iterations the model is subjected to stochastic variation throughout hence the actions taken are different than what is being taken by Q-learner.
- **Taxi:** The optimal policy does seem to differ between all the algorithms. What I see is that the both the iteration algorithms and the Q-learner don't arrive at a policy that involves picking-up and dropping off many customers. Instead, I see that many steps are involved in travelling to pick up a guest. This is more prevalent in the Q-learner hence the rewards obtained are very poor in that case. I'm arriving at this conclusion by looking how many states have actions as 4 and 5, where a 4 means pick-up a passenger and 5 means drop off the passenger. As stated before, the Q-learner has not learnt well on this process.
- **Forest Management:** To me Q-learner performs the best in this case. The optimal policy obtained is similar between Value and Policy iterations but significantly different in Q-learning. I attribute this difference to how the model learns through exploration and exploitation. In other words, the epsilon chosen is high at the beginning thus the model takes random actions and as it learns enough the epsilon is reduced through a decay rate enabling the learner to take an action based on the q-table. This is evident from the policy graph of the Q-learner where the actions vary for each state, whereas in the policy and value iteration cases the action is fairly the same for many states.

Learning Rate:

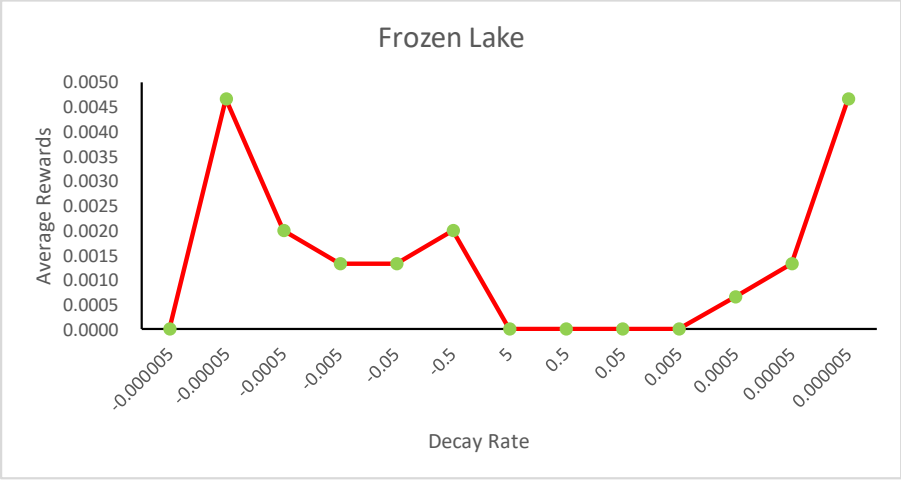
Next, I varied the learning rate across all the models to see how it affects the average rewards to find out it does not make any huge impact on any of the models chosen.

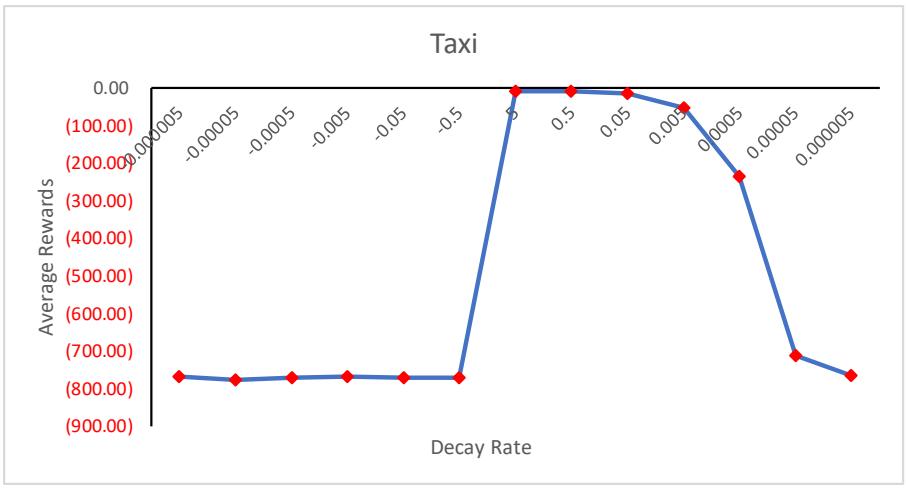
Learning Rate	Frozen Lake	Taxi	Forest Management	Custom
0.1	0.0033	(771.10)	1,299.50	6,367.18
0.2	0.0007	(768.67)	1,299.50	6,368.21
0.3	0.0007	(767.96)	1,299.50	6,336.85
0.4	0.0027	(768.66)	1,299.50	6,163.67
0.5	0.0013	(765.42)	1,299.50	6,165.48
0.6	0.0020	(768.46)	1,299.50	6,065.80
0.7	0.0007	(774.00)	1,299.50	6,112.10
0.8	0.0007	(773.77)	1,299.50	6,110.62
0.9	0.0020	(770.96)	1,299.50	6,129.60
1	0.0013	(768.15)	1,299.50	6,115.62

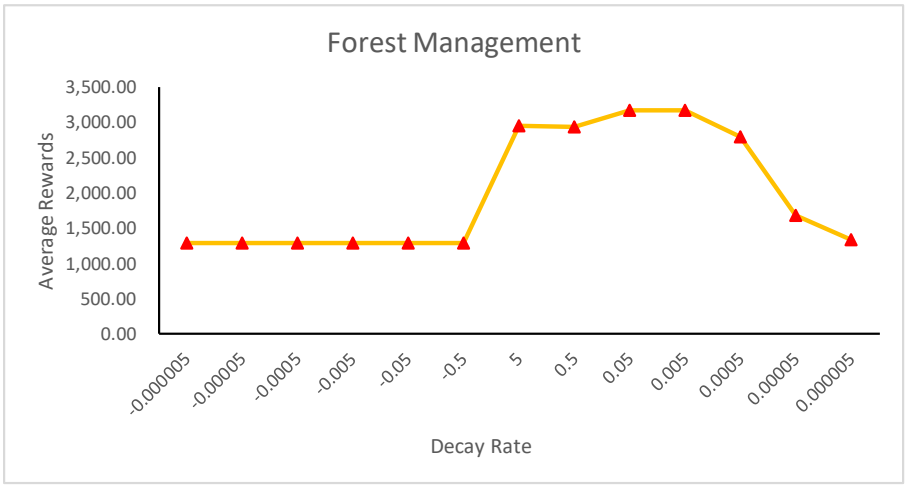
Exploitation vs. Exploration:

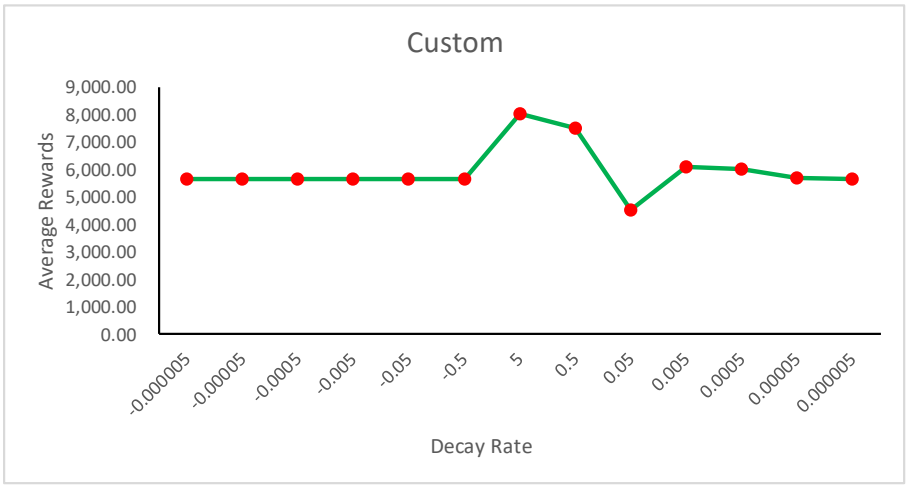
In this phase I varied the epsilon parameter to assess the trade-offs between choosing a random action versus an action that maximizes the Q-value from the Q-table. For example, if the epsilon value is very low then an action based on the values in the qtable would be chosen. On the other hand, for high values of epsilon a random action from the action space would be chosen. During the learning process the epsilon value is varied using a decay rate. The following graphs are plotted for varying decay rates against the average rewards obtained.

- Frozen Lake: Extreme decay values work well which means the model prefers exploitation first then exploration.
- Taxi: Unlike the above case, the model learns better by taking random actions, so exploration at the beginning works well.
- Forest Management: Similar to Taxi, the model does learn better by taking random actions at the beginning.
- Custom: Random actions work better for this model as well.



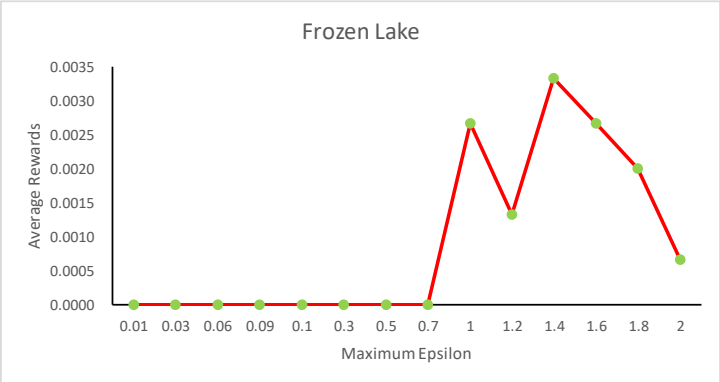

Taxi
0.00
(100.00)
(200.00)
(300.00)
(400.00)
(500.00)
(600.00)
(700.00)
(800.00)
(900.00)
Average Rewards
-0.000005
-0.00005
-0.0005
-0.005
-0.05
-0.5
5
0.5
0.05
0.005
0.0005
0.00005
0.000005
Decay Rate

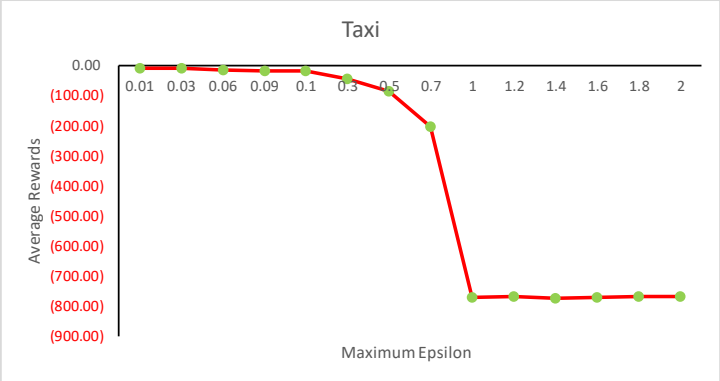

Forest Management
3,500.00
3,000.00
2,500.00
2,000.00
1,500.00
1,000.00
500.00
0.00
Average Rewards
-0.000005
-0.00005
-0.0005
-0.005
-0.05
-0.5
5
0.5
0.05
0.005
0.0005
0.00005
0.000005
Decay Rate

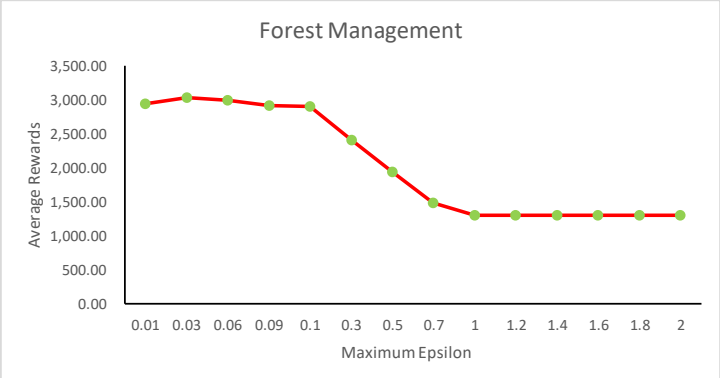

Custom
9,000.00
8,000.00
7,000.00
6,000.00
5,000.00
4,000.00
3,000.00
2,000.00
1,000.00
0.00
Average Rewards
-0.000005
-0.00005
-0.0005
-0.005
-0.05
-0.5
5
0.5
0.05
0.005
0.0005
0.00005
0.000005
Decay Rate

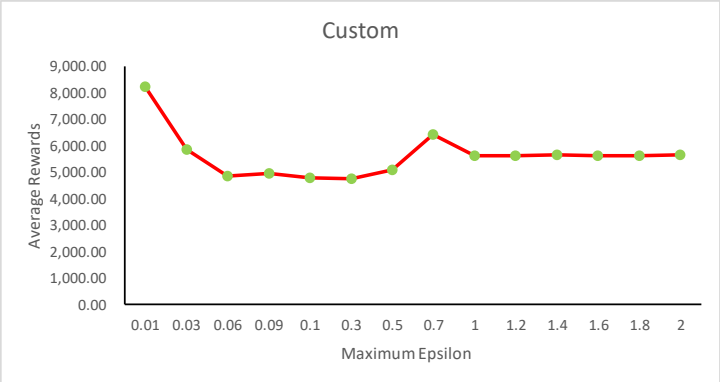
I addition, I varied the Maximum Epsilon parameter to observe the performance of the various algorithms. It has to be noted that by varying this maximum epsilon the epsilon that the model uses to decide if a random action or an action based on the Q-table needs to be taken.

- Frozen Lake: As shown before, exploitation works better than exploration.
- Taxi: Exploration at the beginning works better.
- Forest Management: same as Taxi, random actions at the beginning work well.
- Custom MDP: Both exploration and exploitation are beneficial.









Conclusion:

Overall, I conclude that both value and policy iterations behave in a similar ways for both Frozen Lake and Forest Management processes. I attribute to the simplicity of the MDP for this behaviour. However, Q learner does behave differently for the Forest Management process and I attribute that to how the epsilon is varied i.e. how the model learns. In the case of Taxi model, Q-learner does the worst and I think that's because the model couldn't learn enough in the steps allowed. The custom MDP that I have created differs from the above in the sense that it performs the best in Q-learning.

All relevant files can be found in: https://github.com/mnarasim/ML7641/tree/master/HW4

I have used the approved libraries in this project. In addition, the source code for Policy and Value iterations are written with inspiration from many sources on the public domain, mainly using the below:

https://medium.com/@m.alzantot/deep-reinforcement-learning-demysitifed-episode-2-policy-iteration-value-iteration-and-q-978f9e89ddaa

https://towardsdatascience.com/reinforcement-learning-demystified-solving-mdps-with-dynamic-programming-b52c8093c919