Introduction to AI - 236501
HW3

Yair Nahum 034462796
and
Hala Awwad 209419134

June 28, 2022

MDP

A

A.1.a

In order to work with our R(s) instead of R(s,a,s') one can define the following:

$$\tilde{R}(s_t = s) = \mathbb{E}_{s' \in S, a \in A(s)}[R(s_{t+1} = s', a_t = a | s_t = s] =$$

$$= \sum_{s' \in S} \sum_{a \in A(s)} P(s_{t+1} = s', a_t = a | s_t = s) R(s_t = s, a_t = a, s_{t+1} = s') =$$

$$= \sum_{s' \in S} \sum_{a \in A(s)} P(s_{t+1} = s' | a_t = a, s_t = s) \pi(a_t = a | s_t = s) R(s_t = s, a_t = a, s_{t+1} = s') =$$

$$= \sum_{a \in A(s)} \pi(a_t = a | s_t = s) \sum_{s' \in S} P(s_{t+1} = s' | a_t = a, s_t = s) R(s_t = s, a_t = a, s_{t+1} = s') =$$

Thus, the value function on states $s \in S$ with some given policy $\pi(a|s)$ (probability to make an action a when we're in s) is defined recursively by:

$$V^{\pi}(s) = \sum_{a \in A(s)} \pi(a|s) \sum_{s' \in S} P(s'|s,a)[R(s,a,s') + \gamma V(s')]$$

In case we have a deterministic given policy, we can write it as:

$$V^{\pi}(s) = \sum_{s' \in S} P(s'|s, \pi(s))[R(s, \pi(s), s') + \gamma V(s')]$$

(We assumed a stationary policy and dynamics so we didn't subscript/relate to time on probabilities.)

A.1.b

The Bellman operator:

$$V^*(s) = \max_{a \in A(s)} Q^*(s, a) = \max_{a \in A(s)} \sum_{s' \in S} P(s'|s, a)[R(s, a, s') + \gamma V^*(s')]$$

A.1.c

Value Iteration (VI):

1. Start with any initial value function (zero or random) $V_0(s), \quad \forall s \in S$.
2. Compute recursively, for $n = 0, 1, 2, \ldots$ umtil stopping rule is met (see next the stopping rule details:

$$V_{n+1}(s) = \max_{a \in A(s)} \sum_{s' \in S} P(s'|s, a)[R(s, a, s') + \gamma V_n(s')], \quad \forall s \in S$$

3. Extract optimal policy from optimal value by:

$$\pi(s) \in arg \max_{a \in A(s)} \sum_{s' \in S} P(s'|s, a)[R(s, a, s') + \gamma V^*(s')], \quad \forall s \in S$$

Notes:
1. Stopping rule to obtain a required accuracy.
If

$$||V_{n+1} - V_n||_\infty < \epsilon \cdot \frac{1 - \gamma}{\gamma}$$

then

$$||V_{n+1} - V^*||_\infty \leq \epsilon$$

(this is a theorem that can be proved as done in tutorials)
2. For a fixed policy value iteration, the recursive computation is changed to have $\pi(s)$ instead of a

$$V^{\pi}_{n+1}(s) = \sum_{s' \in S} P(s'|s, \pi(s))[R(s, \pi(s), s') + \gamma V^{\pi}_{n}(s')], \quad \forall s \in S$$

This can be solved also by linear system of equations solver as these are linear equations (usually when the state space is not too big)

When $\gamma = 1$ VI is not guaranteed to converge unless there are absorbing/goal states ($s \in S_G$ no action to transition out from it) that we can get to with probability 1 eventually (This is known as episodic MDPs or Stochastic Shortest Path problems).

A.1.d

Policy Iteration (PI):
PI consists of 2 main parts: Policy evaluation and Policy improvement.

1. Select some stationary policy π_0.
2. For $k = 0, 1, 2, \ldots$ until stopping rule is met (see next the stopping rule details:
   (a) Policy Evaluation: compute V^{π_k} by fixed policy VI or by solving linear system of equations:

   $$V^{\pi_k}_{n+1}(s) = \sum_{s' \in S} P(s'|s, \pi_k(s))[R(s, \pi_k(s), s') + \gamma V^{\pi_k}_{n}(s')], \quad \forall s \in S$$

   Or:

   $$V^{\pi_k} = (I - \gamma P^{\pi_k})^{-1} P^{\pi_k} R^{\pi_k}$$

   (at the last closed solution there is the R^{π_k} which corresponds to the vector of rewards $R(s, \pi_k(s), s')$ multiplied by the transition probabilities matrix for expectancy)

(b) Policy Improvement: compute π_{k+1}, a greedy policy with respect to V^{π_k}:

$$\pi_{k+1}(s) \in arg \max_{a \in A(s)} \sum_{s' \in S} P(s'|s,a)[R(s,a,s') + \gamma V^{\pi_k}(s')], \quad \forall s \in S$$

(c) Stop if $\pi_{k+1} = \pi_k$

When $\gamma = 1$ PI is not guaranteed to converge unless there are absorbing/goal states ($s \in S_G$ no action to transition out from it) that we can get to with probability 1 eventually (This is known as episodic MDPs or Stochastic Shortest Path problems).

A.1.e

	$U_0(s_i)$	$U_1(s_i)$	$U_2(s_i)$	$U_3(s_i)$	$U_4(s_i)$	$U_5(s_i)$	$U_6(s_i)$	$U_7(s_i)$	$U_8(s_i)$
s_1	0	-1	-1	-1	-1	-1			
s_2	0	-1	-2	-2	-2	-2			
s_3	0	-1	-2	-3	-3	-3			
s_4	0	-1	-2	-3	-4	-4			
s_5	0	-1	-2	-3	-4	-5			
s_6	0	-1	-2	-3	-4	-4			
s_7	0	-1	-2	-3	-3	-3			

A.1.f

	$\pi_0(s_i)$	$\pi_1(s_i)$	$\pi_2(s_i)$	$\pi_3(s_i)$	$\pi_4(s_i)$	$\pi_5(s_i)$	$\pi_6(s_i)$	$\pi_7(s_i)$	$\pi_8(s_i)$
s_1	↑	↑	↑						
s_2	↑	↑	↑						
s_3	←	←	←						
s_4	↑	↑	↑						
s_5	→	→	→						
s_6	→	→	↑						
s_7	↓	→	→						

B

Wet part code intro.

C

Wet part in code.

Bonus

We've implemented the code and used the previous section parts in order to calculate optimal policies. in specific the policy iteration:

from value_and_policy_iteration import *

All optimal policies when the given immediate rewards are all 0 and $\gamma = 1$:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@ The board and rewards @@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
| 0      | 0      | 0      | +1     |
| 0      | WALL   | 0      | -1     |
| 0      | 0      | 0      | 0      |

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@ Value iteration @@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Initial utility:
| 0.0    | 0.0    | 0.0    | 0.0    |
| 0.0    | WALL   | 0.0    | 0.0    |
| 0.0    | 0.0    | 0.0    | 0.0    |


Final utility:
| 1.0    | 1.0    | 1.0    | 1.0    |
| 1.0    | WALL   | 1.0    | -1.0   |
| 1.0    | 1.0    | 1.0    | 1.0    |


Final policy:
| UP     | UP     | UP     | +1     |
| UP     | WALL   | LEFT   | -1     |
| UP     | UP     | LEFT   | DOWN   |


Print all possible optimal policy:
| ↑↓→←   | ↑↓→←   | ↑→←    | +1     |
| ↑↓→←   | WALL   | ←      | -1     |
| ↑↓→←   | ↑↓→←   | ←      | ↓      |


Process finished with exit code 0
```

Reward ranges with specific optimal policy when $\gamma = 1$:

```
Print all reward ranges:
The MDP's optimal policy for r <= -1.650390625
| RIGHT | RIGHT | RIGHT | +1    |
| UP    | WALL  | RIGHT | -1    |
| RIGHT | RIGHT | RIGHT | UP    |

The MDP's optimal policy for -1.650390625 < r <=-1.572265625
| RIGHT | RIGHT | RIGHT | +1    |
| UP    | WALL  | UP    | -1    |
| RIGHT | RIGHT | RIGHT | UP    |

The MDP's optimal policy for -1.572265625 < r <=-0.732421875
| RIGHT | RIGHT | RIGHT | +1    |
| UP    | WALL  | UP    | -1    |
| RIGHT | RIGHT | UP    | UP    |

The MDP's optimal policy for -0.732421875 < r <=-0.458984375
| RIGHT | RIGHT | RIGHT | +1    |
| UP    | WALL  | UP    | -1    |
| UP    | RIGHT | UP    | UP    |

The MDP's optimal policy for -0.458984375 < r <=-0.087890625
| RIGHT | RIGHT | RIGHT | +1    |
| UP    | WALL  | UP    | -1    |
| UP    | RIGHT | UP    | LEFT  |

The MDP's optimal policy for -0.087890625 < r <=-0.048828125
| RIGHT | RIGHT | RIGHT | +1    |
| UP    | WALL  | UP    | -1    |
| UP    | LEFT  | UP    | LEFT  |

The MDP's optimal policy for -0.048828125 < r <=-0.029296875
| RIGHT | RIGHT | RIGHT | +1    |
| UP    | WALL  | UP    | -1    |
| UP    | LEFT  | LEFT  | LEFT  |

The MDP's optimal policy for -0.009765625 < r
| RIGHT | RIGHT | RIGHT | +1    |
| UP    | WALL  | LEFT  | -1    |
| UP    | LEFT  | LEFT  | DOWN  |


Process finished with exit code 0
```

Note: In printing all reward ranges, we used VI instead of PI.
We didn't use PI since it uses policy evaluation by inverting a matrix and $(I - \gamma P)^{-1}$ is not invert-able (0 eigen value by definition as P has and eigen value of 1) if $\gamma == 1$.
For the case when $\gamma = 1$, we've limited the range to find by an upper bound of 0 immediate reward as otherwise VI will not converge.

Learning Introduction

A

Wet part code intro.

B

B.1

B.1.a. The entropy $H(Passed)$:

$$H(Passed) = -[\frac{2}{6}\log_2(\frac{2}{6})+\frac{4}{6}\log_2\frac{4}{6}] = -[\frac{1}{3}(\log_2 1-\log_2 3)+\frac{2}{3}(\log_2 2-\log_2 3)] =$$

$$-[\frac{2}{3} - \log_2 3] = \log_2 3 - \frac{2}{3} \Rightarrow$$

$$\boxed{H(Passed) = \log_2 3 - \tfrac{2}{3} \approx 0.9183}$$

B.1.b. The entropy $H(Passed|Average)$:

$$H(Passed|Average = Low) = -[\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}] = 1$$

$$H(Passed|Average = Medium) = -[\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}] = 1$$

$$H(Passed|Average = High) = -[1\log_2 1 + 0\log_2 0] = 0$$

The weighted entropy:

$$H(Passed|Average) = \sum_{i=1}^{3}\frac{|E_i|}{|E|}entropy(E_i) =$$

$$= \frac{2}{6}H(Passed|Average = Low) + \frac{2}{6}H(Passed|Average = Medium)+$$

$$+\frac{2}{6}H(Passed|Average = High) = \frac{2}{3} \Rightarrow$$

$$\boxed{H(Passed|Average) = \tfrac{2}{3}}$$

B.1.c. The entropy $H(Passed|Studied)$:

$$H(Passed|Studied = No) = -[\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}] = \log_2 3 - \frac{2}{3}$$

$$H(Passed|Studied = Yes) = -[1\log_2 1 + 0\log_2 0] = 0$$

The weighted entropy:

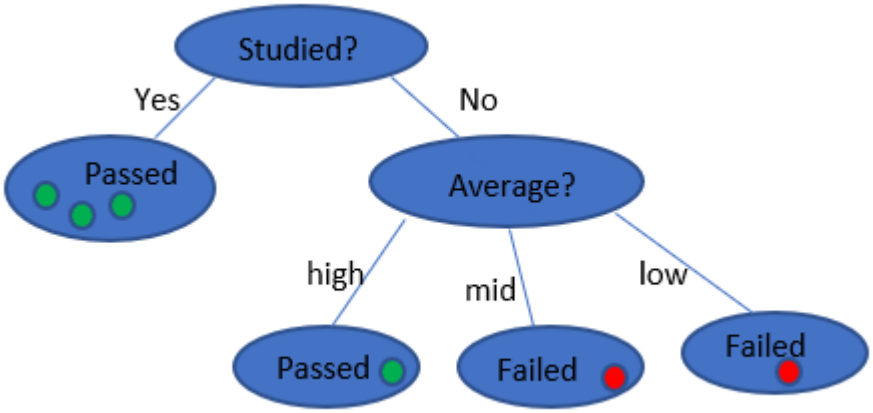
$$H(Passed|Studied) = \frac{3}{6}(\log_2 3 - \frac{2}{3}) + \frac{3}{6}0 = \frac{1}{2}\log_2 3 - \frac{1}{3}$$

$$\boxed{H(Passed|Studied) = \tfrac{1}{2}\log_2 3 - \tfrac{1}{3} \approx 0.4591}$$

.

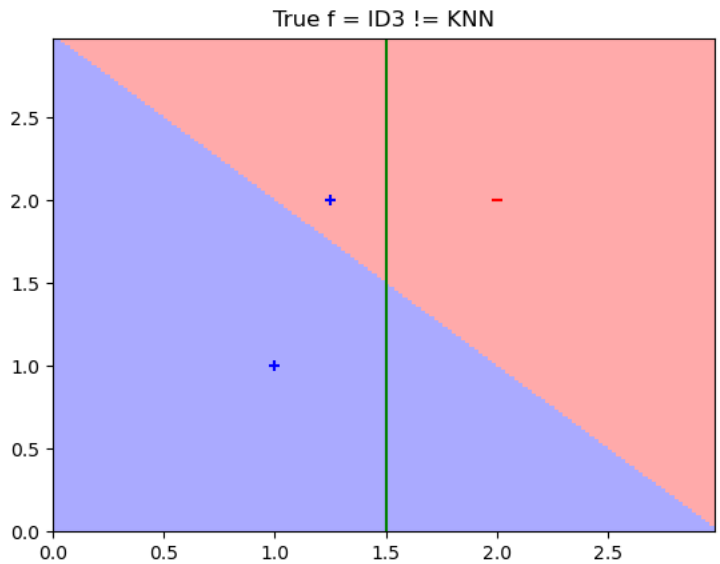
B.1.d.

The IG is bigger when selecting the "studied" feature. so we get the following tree (based on ID3 algo with the given samples):



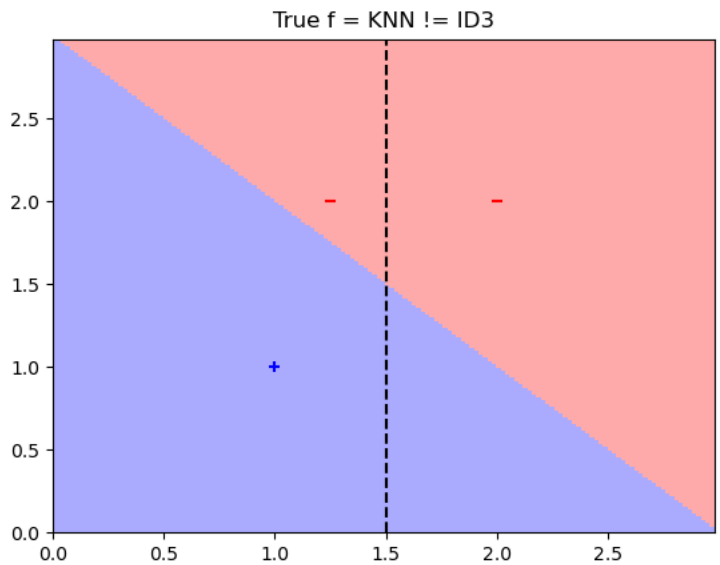
B.2

B.2.a.



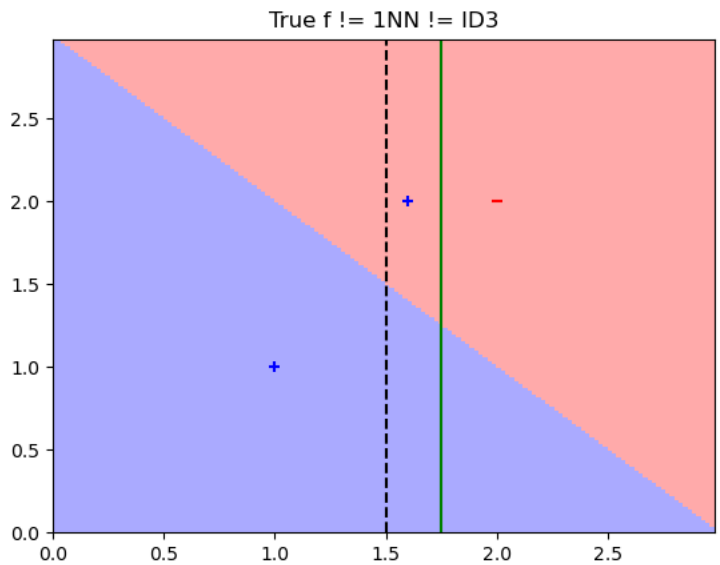
In green we see the True classifier f which is the same as ID3. The KNN built by the training examples (vornoy diagram) at (2,2) and (1,1) miss-classify the point at (1.25, 2).

B.2.b.



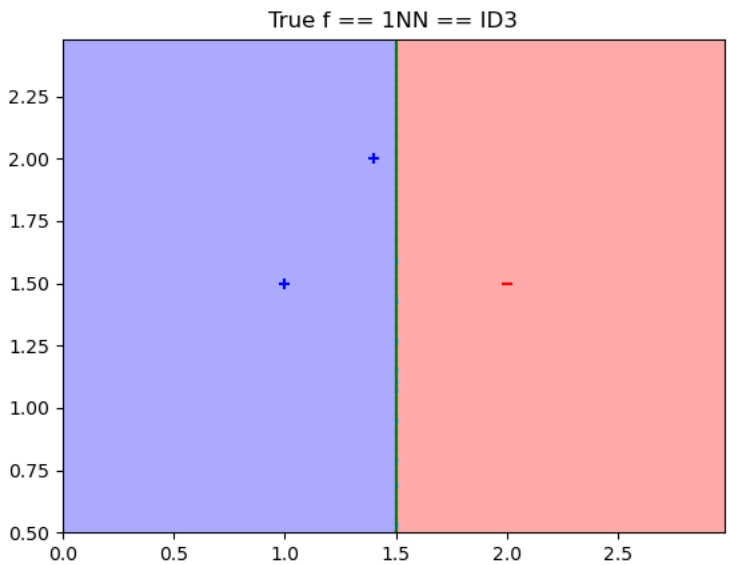
Above we see assume the True classifier f is the same as KNN (vornoy diagram). The ID3 built by the training examples at (2,2) and (1,1) miss-classify the point at (1.25, 2).

B.2.c.



Above in green is the True classifier f. Both KNN and ID3 built by the training examples at (2,2) and (1,1) miss-classify the point at (1.6, 2).

B.2.d.



Above in green is the True classifier f, 1NN and ID3 built by the training examples at (1.5,1) and (1.5,2).

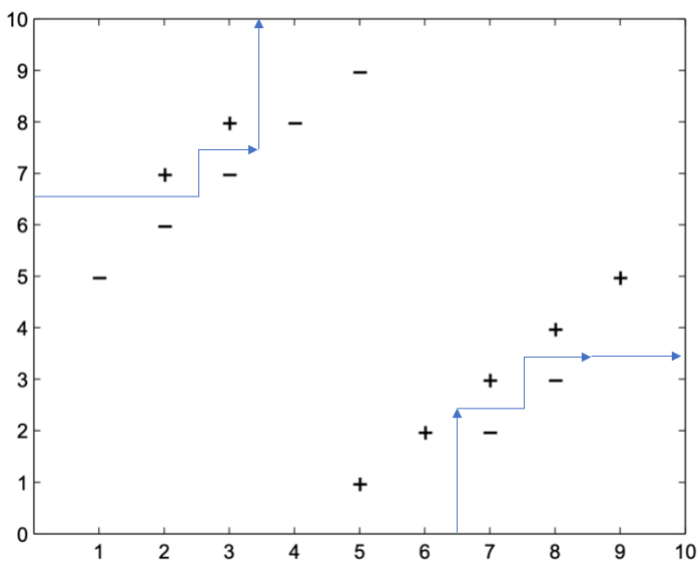
B.3

B.3.a.
For $K = 1$ we get 0 training error. since the nearest point to each example is itself.
B.3.b.
With too big K, for example $K >= 5$, we will get that for $(y > x + 7, x)$ half plane, we would predict wrongly as we don't have enough examples from the right prediction "+" and we will predict wrongly "-" (the same for $(y, x > y + 7)$ half plane, we will predict wrongly "+").
This basically increases bias.
With too small K, for example $K = 1, 3$, we would get another problem as there are again many areas in which we don't have enough samples from the needed kind and thus the appearance of the other kind will dominate the prediction.
This is the problem of over-fit (high variance) due too not enough examples.

B.3.c.
For $K = 1$, we predict wrongly on 10 out of 14 examples (the corners of the lines are predicted OK). Thus, for $K = 1$ the LOOCV error is $\frac{10}{14}$.
For $K = 3$, we predict wrongly on 6 out of 14 examples. Thus, for $K = 3$ the LOOCV error is $\frac{6}{14}$.
For $K = 5, 7$, we predict wrongly on 4 out of 14 examples (the "+" above the "-" 5-line and the "-" below the "+" 5-line). Thus, for $K = 5, 7$ the LOOCV error is $\frac{4}{14}$.
For $K = 9$ we get again $\frac{10}{14}$ as all the "-" 5-line and "+" 5-line are predicted wrongly. For other bigger K's, it is also very bad.

Bottom line, for smallest LOOCV $\frac{2}{7}$, one should take $K = 5$ or $K = 7$

B.3.d.



C

C.4

In code. passed L2/accuracy unit tests.

C.5 - ID3 algorithm

C.5.a.
In code.

C.5.b.
basic experiment passed needed accuracy with 94.69%

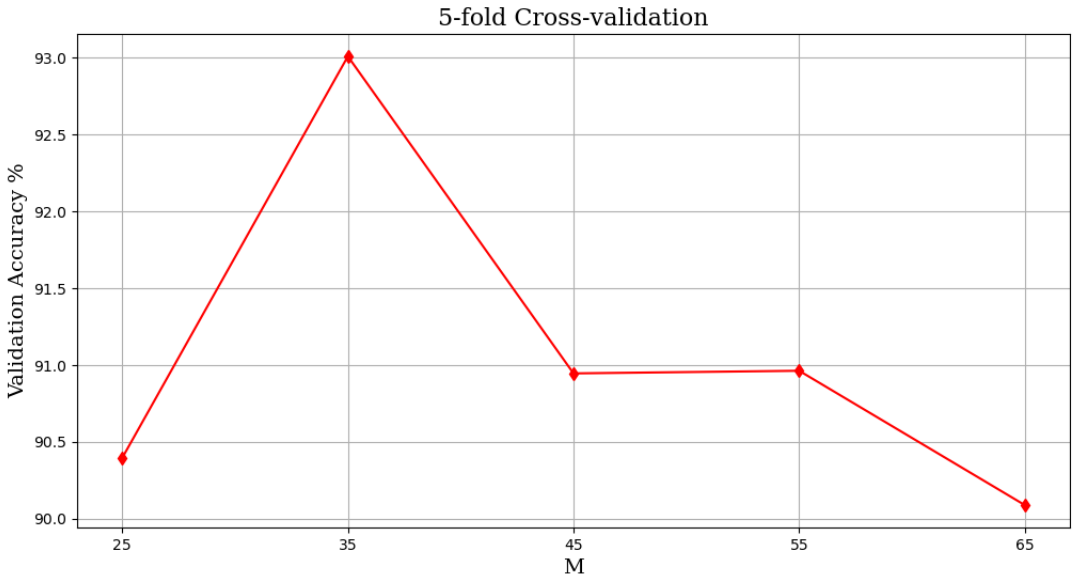
C.6 - early pruning

C.6.a.
Pruning improves the generalization and prevent over-fitting. By having leaves with more labels to decide from, we increase our hypothesis statistical significance (law of big numbers) and the variance of our model is reduced.

C.6.b.
In code.

C.6.c.

C.6.c.i


5-fold Cross-validation
Validation Accuracy %
93.0
92.5
92.0
91.5
91.0
90.5
90.0
25
35
45
55
65
M

The KFold CV run output:

M value	Validation Accuracy
25	90.40%
35	93.01%
45	90.95%
55	90.96%
65	90.09%

Best M	Validation Accuracy
35	93.01%

C.6.c.ii
According to the graph and output we can see that for minimum number of samples in a leaf $m = 35$, we get the best accuracy 93.01%.
We find this optimal value by K fold CV process.

C.6.d.
running with the $best\_m = 35$ parameter we get better performance over the test data set as the accuracy is now 97.35%