Assignment 2 part 2:
Machine Learning, MDP & RL (15%)

Due October 27th (Sunday), 11:59 pm

Exercise 1: ***[Supervised ML] Implementation of Decision Trees*** **(** $2 \times 26 = 52$ ***marks)***

(a) Implement decision tree learning as described in Section 18.3: 'Learning Decision Trees' of the textbook. Your implementation should work for datasets with binary features and binary labels (while you may implement it for other kinds of features and labels too, we only require that it works for binary features and labels). Your code should use the maximum *information gain* heuristic as its splitting rule. Implement the recursive training procedure given in Figure 18.5 of the book, with the following extension:

- Your code should take in a parameter *depth*, which specifies the maximum height of the tree. If depth is 0, then the tree produced should be a single node. If depth is 1, then the tree should be at most a node whose children are leaf nodes (this is sometimes called a decision stump). More generally, if depth is n then there should be at most n non-leaf nodes *along any path from the root to a leaf node*.

In addition to the (extended) training procedure, also implement a procedure for predicting the label of an unlabeled test example using the learned tree.

To help you out, we have provided some skeleton PYTHON code on Moodle, in the file **a2p2.zip**. Here are some implementation hints:

- Test frequently, using the "small_test.txt" instance provided in the zip. This is a simplified version of the Game Playing problem seen in the lectures (see "small_text_strings.csv" for the variable names).
- When calculating the entropy of a Boolean random variable X, make sure to correctly handle the cases where $\Pr(X) = 0$ and $\Pr(X) = 1$.
- Similarly, watch out for division by zero when computing probabilities.
- Stick to binary-valued features. This will simplify your code.
- If you are getting strange results, try printing out your information gain values. These should all be between 0 and 1.

(b) Test your implementation on the dataset provided in the **a2p2.zip** file. In this zip you will find a file "train.txt" and "test.txt", containing training and test data respectively. Each file is in 'comma separated value' format. The first 38 values on each line are binary features, and the last value on each line is the binary label. The task in this dataset is to predict the winner in a chess end-game involving King and Rook versus King and Pawn.[1] The features are derived from the chess positions and the label is 1

[1] Based on https://archive.ics.uci.edu/ml/datasets/Chess+%28King-Rook+vs.+King-Pawn%29. Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

if white can win, or 0 if white cannot win. There are 2109 examples in the training set and 1087 examples in the test set (about a 2/3 train/test split).

Write a brief (no more than 1 page) discussion of your results. This write-up should include one or more plots showing the training and test set accuracy on the provided data set, for all values of *depth* between 0 and 30. A simple way to plot all this would be as a line graph with depth on the x-axis and accuracy on the y-axis. You can then plot a training accuracy and test accuracy line. Also discuss in your write-up:

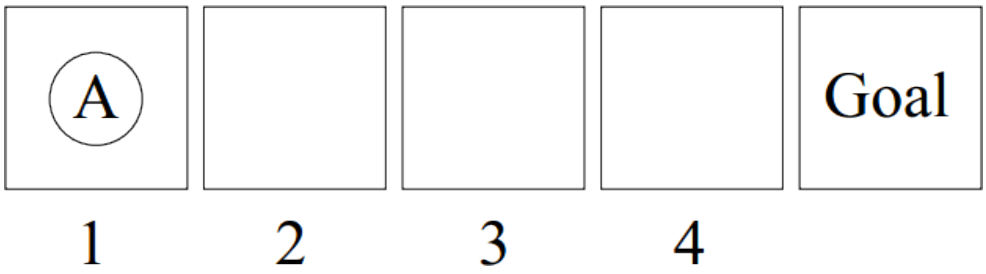
- Which value of *depth* gave the best test set accuracy?
- Which value of *depth* gave the best training set accuracy?
- Does increasing the *depth* lead to overfitting on this dataset?

Your program for this question **must** be executable using the following command line
python question2.py <train_file> <depth> <test_file> <output_file>
where <train_file> is the name of the input training file, <depth> is the maximum depth allowed for the decision tree, <test_file> is the name of the test file, and <output_file> is the name of the output file into which the accuracy of the learned model on the test set is written into a line. If the output file already contains some lines, the accuracy must be *appended* as the last line of the file.

For example, consider the following command lines:
python question2.py train.txt 5 test.txt output.txt
This should learn a decision tree of maximum depth 5, based on the training data in train.txt, and then write the accuracy of the learned model on the test.txt into the output file output.txt.

Exercise 2: ***[MDP & RL] Soccer (*** $6 \times 8 = 48$ ***marks)***

A soccer robot A is on a fast break toward the goal, starting in position 1. From positions 1 through 3, it can either take the action shoot (a_S), or dribble the ball forward (a_D). From 4 it can only shoot. If it shoots, it either scores a goal (state s_G) or misses (state s_M). If it dribbles, it either advances a square (e.g., from 1 to 2), or it loses the ball, ending up in s_M. When shooting, the robot is more likely to score a goal from states closer to the goal; when dribbling, the likelihood of missing is independent of the current state.



In this MDP, the states are $S = \{1, 2, 3, 4, s_G, s_M\}$, where s_G and s_M are terminal states. The transition model depends on the parameter y, which is the probability of dribbling

success. Assume a discount factor of $\gamma = 1$.

$$
\begin{aligned}
T(k, a_S, s_G) &= \frac{k}{6} \\
T(k, a_S, s_M) &= 1 - \frac{k}{6} \\
T(k, a_D, k+1) &= y \text{ for } k \in \{1, 2, 3\} \\
T(k, a_D, s_M) &= 1 - y \text{ for } k \in \{1, 2, 3\} \\
R(k, a_S, s_G) &= 1
\end{aligned}
$$

Rewards are 0 for all other transitions.

(a) What is $V^\pi(1)$ for the policy $\pi(\cdot) = a_S$, the policy that always shoots?

(b) What is $Q^*(3, a_D)$ in terms of y?

(c) Using $y = 3/4$, complete the first two iterations of value iteration.

(d) After how many iterations will value iteration compute the optimal values for all states, for this particular problem?

(e) For what range of values of y is $Q^*(3, a_S) \geq Q^*(3, a_D)$?

(f) Now consider Q-learning, in which we do not have a model (T, y, and k above), and instead learn from a series of experienced transitions. Using a learning rate of $\alpha = 1/2$ execute Q-learning on these episodes:

- $\langle 1, a_D \rangle, \langle 2, a_D \rangle, \langle 3, a_D \rangle, \langle 4, a_S \rangle, \langle s_G \rangle$
- $\langle 1, a_D \rangle, \langle 2, a_D \rangle, \langle 3, a_D \rangle, \langle 4, a_S \rangle, \langle s_M \rangle$
- $\langle 1, a_D \rangle, \langle 2, a_D \rangle, \langle 3, a_S \rangle, \langle s_G \rangle$
- $\langle 1, a_D \rangle, \langle 2, a_S \rangle, \langle s_M \rangle$
- $\langle 1, a_D \rangle, \langle 2, a_D \rangle, \langle 3, a_D \rangle, \langle s_M \rangle$

Submission instructions:

- Remember that it is an individual assignment; you are **not** allowed to work in pairs.
- Submit on Moodle a zip file that contains (1) A PDF file containing your solutions to the questions, and (2) .py file(s) containing your implementation for question 1. The zip file should be named A2P2_<StudentID>.zip, where <StudentID> is your Student ID number.
- **You can only use Python as the programming language.** Please use the skeleton code that we have provided for you on Moodle for this assignment.
- Your Python file must be executable on **lab computers** by the command-line mentioned in question 1. If your program does not run smoothly on a lab computer with this command-line, then you may get zero marks for that implementation question. We emphasize that you need to use **exactly** the name "question2.py" for your program.
- **Late submission policy:** 5% of the maximum mark will be deducted for every work day a submission is late.