Homework 1
Decision Trees and kNN

UTK COSC 522: Machine Learning (Fall 2025)
OUT: Tuesday, Sep 2nd, 2025

DUE: Wednesday, Sep 17th, 2025, 11:59pm EST

Homework Instructions

- **Collaboration policy: Taking Responsibility for Your Career**

  The skills you learn in this course are tools for your future, and this policy is designed to help you sharpen them. You are responsible for your own learning and career, and this framework ensures you get the most from every assignment.

  We encourage collaboration, but it must be done *right*. *First*, make a genuine effort to solve problems on your own. This independent effort is where the most critical learning occurs. *Afterward*, you may discuss strategies with peers or consult resources to clarify concepts—the goal is to deepen your own understanding, not to simply get an answer.

  Finally, like any professional, you must stand behind your own work. Your submitted solution must be written entirely by you, from scratch, without collaborators present. This proves you have truly mastered the material. To maintain academic and professional integrity, you must also cite any person or resource that contributed to your understanding along the way.

- **Late Submission Policy:** See the late homework policy here: https://docs.google.com/document/d/1lYopNzjn_0AOipRiiwO4h2Ow64seQNeD/edit?usp=sharing&ouid=114000599194691965836&rtpof=true&sd=true

- **Submitting your work:**

  - **Canvas:** For this homework, you will submit a **single .zip file** to Canvas. Please name your file in the format **YourName_NetID_HW1.zip**. This zip file must contain exactly two files: your written solutions in a single PDF and your Python code.

    Your **single PDF file** must contain your answers to all questions, including written problems (proofs, short answers, plots) and the empirical questions from the programming section. You must use the provided homework template. Using the LaTeX version to typeset is strongly recommended, but you may also submit a scan of the template with legible handwriting; **illegible answers will not be graded**. Please complete all answers within the provided boxes.

    The second file in your zip archive must be your *single* completed programming file, named exactly as **decision_tree.py**. Please ensure your code is runnable, as we may execute it to verify your results.

    Regrade requests can be made after homework grades are released. Please be aware that a regrade request allows the TA to review your entire assignment, which may result in points being deducted if new errors are found.

For multiple-choice and select-all-that-apply questions, please shade the corresponding box or circle to indicate your answer(s). If you are using the LaTeX template, use the ■ command for shaded boxes and ● for shaded circles. **Note**: Do not alter the template in any other way. If a question includes a box for showing your work, you must **show your work** to receive full credit.

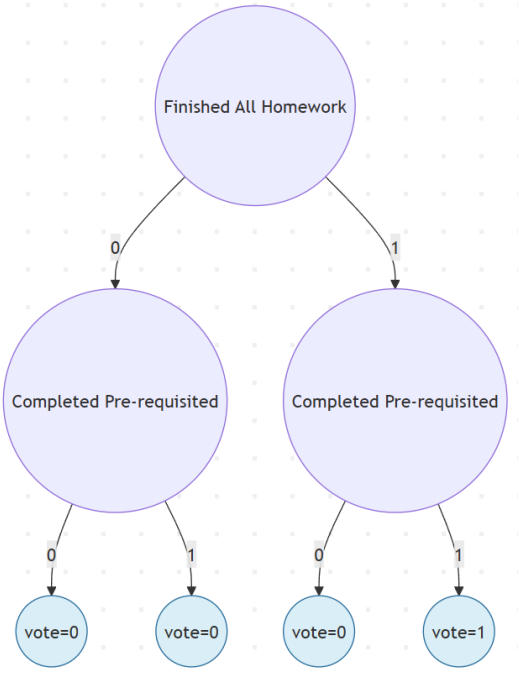
1 Decision Trees [20 pts]

1. Consider the dataset in Table 1 for this problem. Given the four attributes on the left, we want to predict if the student got an A in the course. The following questions involve some computation, so you may want to solve them programmatically. You can find this dataset in *data.txt*

Wakes Up Early	**Finished All Homework**	**Completed Pre-requisites**	**Likes Coffee**	**A**
1	1	0	0	0
1	1	1	0	1
0	0	1	0	0
0	1	1	0	1
0	1	1	0	1
0	0	1	1	0
1	0	0	0	0
0	1	0	1	1
0	0	1	0	1
1	0	0	0	0
1	1	1	0	1
0	1	1	1	0
0	0	0	0	0
1	0	0	1	0

Table 1: decision tree features and labels

Create a decision tree of depth 2 using the ID3 algorithm from class. The tree should have the same structure as the diagram in Figure ??. Note that 0 leads to the left branch and 1 leads to the right branch.



(i) **[4 pts]** What is the attribute at Node 1? What is the information gain of this attribute? Please round your answer to 4 decimal places.

Attribute to split on: Finished All Homework Mutual information: 0.2578

(ii) **[4 pts]** What is the attribute at Node 2? What is the information gain of this attribute? Please round your answer to 4 decimal places.

Attribute to split on: Completed Pre-requisited Mutual information: 0.1981

(iii) **[4 pts]** What is the attribute at Node 3? What is the information gain of this attribute? Please round your answer to 4 decimal places.

Note: If there is a tie on the attribute with the highest information gain, write any of the attribute that has the highest information gain.

Attribute to split on: Completed Pre-requisited M utual information: 0.0617

2. **[4 pts]** Consider learning a decision tree for some binary classification task. Assume that we do not restrict the size of the tree and that the tree can branch multiple times on the same attribute. Under what condition(s) would we be unable to construct a tree that perfectly classifies the dataset?

   If there are conflict duplicates available in dataset, we would be unable to construct a tree that perfectly classifies the dataset. For example, for same set of the features if there are two different labels available then we have two choose only one of them and one will be wrong.

3. **[4 pts]** Given enough time, is the ID3 algorithm guaranteed to output the smallest decision tree that is *consistent* with the training dataset i.e., achieves zero training error rate? Briefly explain why or why not.

   ID3 algorithm **chooses the best attribute at each node, which has the most significant information gain.** It keep doing this until it reaches stopping criteria. since it selects the attribute with highest information gain it each node, we can say it **"Greedy" algorithm**.
   Sometime the smallest decision tree(no error) requires taking **lower-gain split** early. Thus, ID3 algorithm **is not guaranteed** to output the smallest decision tree with 0 error.

2 Nearest Neighbors and Decision Trees [6 Points]

Consider a multiclass classification problem with 2 real-valued features. Suppose you are given n data points $P_1, P_2, \ldots, P_n$ and the corresponding label for each data point $C_1, C_2, \ldots, C_n$ (where $C_1, C_2, \ldots, C_n$ take values from the set of all possible class labels). Under the k nearest neighbors classification scheme, each new element Q is simply categorized by a majority vote among its k nearest neighbors. The 1-NN is a simple variant of this which divides up the input space for classification purposes into convex regions (see Figure 1), each corresponding to a point in the dataset.

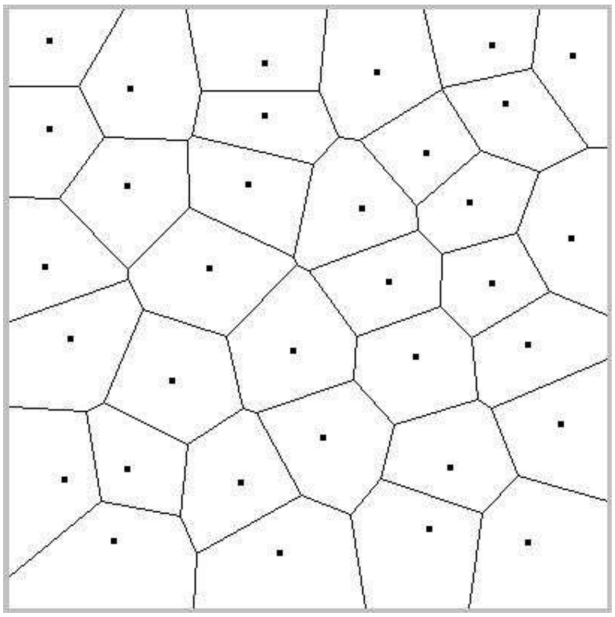


Figure 1: 1-NN decision boundaries using the Euclidean distance metric.

1. **[6 Points]** Is it possible to build a decision tree (with decisions at each node of the form "is $x_1 > a$", "is $x_1 < b$", "is $x_2 > c$", or "is $x_2 < d$" for any real constants a, b, c, d) which behaves identically to a 1-NN classifier using the Euclidean distance metric? If so, how and if not, why not.

No, since 1-NN looks for the nearest neighbor and creates diagonal boundaries. Whereas decision tree creates only vertical/horizontal boundaries. which is not identical to 1-NN classifier using the Euclidean distance metric.

3 k-Nearest Neighbors [19 Points]

1. **[6 pts]** In a kNN classification problem, assume that the distance measure is not explicitly specified to you. Instead, you are given a "black box" where you input a set of data points $P_1, P_2, \ldots P_n$ and an unlabelled data point Q, and the black box outputs the nearest neighbor of Q, say P_i and its corresponding label C_i. Is it possible to construct a kNN classification algorithm (w.r.t the unknown distance metrics) based on this black box alone? If so, how and if not, why not?

   Yes, it is possible.
   Here how can we do it:
   - We will **start from set nearest data points to a data point and remove neighbors**.
   - Using black box, we will **get 1 nearest neighbor (P1)** for that training point.
   - We will **note down that label and remove** it from training point.
   - We will keep doing this until we get **K-nearest neighbor**.

2. **[4 pts]** Now suppose the black box returns the j nearest neighbors (and their corresponding labels) instead of the single nearest neighbor (assume $j \neq k$). Is it still possible to construct a kNN classification algorithm based on the black box? If so, how and if not, why not? You should separately discuss two scenarios: (1) the returned k-nearest neighbors are *ordered/unoredered* based on the unknown distance metrics.

   Note: The black box operates with a *set* of data points, which assumes they are distinct.

   - Ordered j-NN list: **Yes**
     - if j >= k: We will note down first **K** item's labels.
     - if j < k: we will note down those **j** and remove from set. And we will explore next **j**. we will keep doing this until we find **K** neighbors.
   - Unordered j-NN list: **No**
     - **k-NN needs the k closest points in order** (1st, 2nd, 3rd, ... , kth).
     - Unordered **j-NN gives set of pointers but without index.**
     - Prediction can be change depending on which **K we pick.**

3. **[3 pts]** Assume we have a dataset with binary labels and we know that the leave-one-out cross validation accuracy (LOOCV) of a 1-NN classifier on this dataset is 1. Now suppose that we introduce some noise to the dataset: each label has a 10% chance of being flipped to the other class. What is the *expected* LOOCV accuracy of a 1-NN classifier in this setting? Note that the validation accuracy is always calculated with respect to the given ground-truth label, whether it being flipped or not.

- We have given validation accuracy (LOOCV) of a **1-NN classifier on this dataset is 1**. And each label has a $\mathbf{p = 0.1}$ chance of being flipped to the other class.
- If **both** flipped or not, then **accuracy is 100.**
- Either of them flip then accuracy = $(1\text{-p})^2 + p^2 = 1 - 2p(1-p)$.
- For $p = 0.1$: $0.9^2 + 0.1^2 = 0.82$.

4. **[3 pts]** Suppose that in the dataset from the previous question, exactly 10% of the labels got flipped. What is the *maximum* possible LOOCV accuracy (you can assume any data distribution in your preference)? Justify your answer using a formal proof or by constructing an example.

Maximum possible LOOCV accuracy will be **1**.
We can **make set of same-labels data point pairs far from others**. and then we will **flip 5 percentage of labels for each pair**, **total 10** percentage. Thus the accuracy will stay **1** for LOOCV.

5. **[3 pts]** How should the noise level (i.e., the probability of a label being flipped) affect your choice of k for the k-NN algorithm, if at all? You do not need to provide a formal proof but you should give a brief justification. *Hint:* think about variance-bias tradeoff in overfitting and underfitting scenarios.

If **label noise goes up**, then I pick a **larger K** so more neighbors vote and random labels flips get averaged out.
If **low noice or no noice**, then I choose **smaller K** to keep boundary flexible.
I would start with K = $\sqrt{n}$.$(Lecture3 - KNN.pdf, p.33)$.

4 Programming (55 points)

In this assignment, you will implement a decision tree learning algorithm for binary classification. In addition, we will ask you to run some end-to-end experiments on two tasks (predicting whether or not a patient has heart disease / predicting the final grade for high school students) and report your results. Please confine all code you write to a single, self-contained file titled `decision_tree.py`.

4.1 The Tasks and Datasets

Materials Download the zip file from the course website. The zip file will have a handout folder that contains all the data that you will need in order to complete this assignment.

Datasets The handout contains three datasets. Each one contains attributes and labels and is already split into training and validation data. The first line of each `.tsv` file contains the name of each attribute, and *the class label is always the last column.*

1. **heart:** The first task is to predict whether a patient has been (or will be) diagnosed with heart disease, based on available patient information. The attributes (aka. features) are:
   (a) `sex`: The sex of the patient—1 if the patient is male, and 0 if the patient is female.
   (b) `chest_pain`: 1 if the patient has chest pain, and 0 otherwise.
   (c) `high_blood_sugar`: 1 if the patient has high blood sugar (>120 mg/dl fasting), and 0 otherwise.
   (d) `abnormal_ecg`: 1 if the patient had an abnormal resting electrocardiographic (ECG) reading, and 0 otherwise.
   (e) `angina`: 1 if exercise induced angina in the patient, and 0 otherwise. Angina is a type of severe chest pain.
   (f) `flat_ST`: 1 if the patient's ST segment (a section of an ECG) was flat during exercise, and 0 if it had some slope.
   (g) `fluoroscopy`: 1 if a physician used fluoroscopy, and 0 otherwise. Fluoroscopy is an imaging technique used to see the flow of blood through the heart.
   (h) `thalassemia`: 1 if the patient is known to have thalassemia, and 0 otherwise. Thalassemia is a blood disorder that may impair the oxygen-carrying capacity of the patient's red blood cells.
   (i) `heart_disease`: 1 if the patient was diagnosed with heart disease, and 0 otherwise. This is the class label you should predict.

   The training data is in `heart_train.tsv`, and the validation data in `heart_valid.tsv`.
2. **education:** The second task is to predict the final grade for high school students. The attributes are student grades on 5 multiple choice assignments *M1* through *M5*, 4 programming assignments *P1* through *P4*, and the final exam *F*. Values of 1 indicate that a student received an A, and 0 indicates that the student did not receive an A. The training data is in `education_train.tsv`, and the validation data in `education_valid.tsv`.
3. **small:** We also include `small_train.tsv` and `small_valid.tsv`—a small, purely for demonstration version of the **heart** dataset, with *only* attributes `chest_pain` and `thalassemia`.

4.2 Program: Decision Tree

Your code should learn a decision tree with a specified maximum depth, print the decision tree in a specified format, predict the labels of the training and validation examples, and calculate training and validation errors.

Your implementation must satisfy the following requirements:

- Use mutual information to determine which attribute to split on.
- Be sure you're correctly weighting your calculation of mutual information. For a split on attribute X, $I(Y;X) = H(Y) - H(Y|X) = H(Y) - P(X=0)H(Y|X=0) - P(X=1)H(Y|X=1)$.
- As a stopping rule, only split on an attribute if the mutual information is $>$ certain threshold. By default, you should set the threshold to 0. Certain questions in 4.3 may require you to experiment with different threshold.
- Do not grow the tree beyond a certain max-depth, if specified in the question. For example, for a maximum depth of 3, split a node only if the mutual information is > 0 and the current level of the node is < 3.
- Use a majority vote of the labels at each leaf to make classification decisions. If the vote is tied, choose the label that is numerically larger (i.e., 1 should be chosen before 0)
- It is possible for different columns to have equal values for mutual information. In this case, you should split on the ***first column to break ties*** (e.g. if column 0 and column 4 have the same mutual information, use column 0).

4.2.1 Getting Started

Careful planning will help you to correctly and concisely implement your decision tree learning algorithm. Here are a few *hints* to get you started:

- Write helper functions to calculate entropy and mutual information.
- It is best to think of a decision tree as a collection of nodes, where nodes are either leaf nodes (where final decisions are made) or interior nodes (where we split on attributes). It is helpful to design a function to train a single node (i.e. a depth-0 tree), and then recursively call that function to create sub-trees.
- In the recursion, keep track of the depth of the current tree so you can stop growing the tree beyond the max-depth.
- Implement a function that takes a learned decision tree and data as inputs, and generates predicted labels. You can write a separate function to calculate the error of the predicted labels with respect to the given (ground-truth) labels.
- Be sure to correctly handle the case where the specified maximum depth is greater than the total number of attributes.
- Be sure to handle the case where max-depth is zero (i.e. a majority vote classifier).

4.2.2 Output: Printing the Tree

You should also write a function to pretty-print your learned decision tree. **Your function should print your tree only *after* you are done generating the fully-trained tree.** Each row should correspond to a node in the tree. They should be printed using a *pre-order depth-first-search* traversal (but you may print left-to-right or right-to-left, i.e. your answer does not need to have exactly the same order as the reference below). Print the `attribute` of the node's parent (e.g., `chest_pain` in example format below) and the `attribute value` corresponding to the node (e.g., value of 0 for `chest_pain`). Also include the sufficient statistics (i.e. count of negative (label as 0) / positive examples (label as 1)) for the data passed to that node. The row for the root should include *only* those sufficient statistics. A node at depth d, should be prefixed by d copies of the string '| '.

Below, we have provided the recommended format for printing the tree (the `2` in the python command below denotes the maximum tree depth). You can print it directly rather than to a file.

```
$ python decision_tree.py small_train.tsv small_valid.tsv 2

[14 0/14 1]
| chest_pain = 0: [4 0/12 1]
| | thalassemia = 0: [3 0/4 1]
| | thalassemia = 1: [1 0/8 1]
| chest_pain = 1: [10 0/2 1]
| | thalassemia = 0: [7 0/0 1]
| | thalassemia = 1: [3 0/2 1]
```

However, you should be careful that the tree might not be full. For example, with a different subset of the small dataset, there may be no nodes under `chest_pain = 0` if all labels are the same.

The following pretty-print shows the education dataset with max-depth 3. Use this example to check your code before submitting your pretty-print of the heart dataset.

```
$ python decision_tree.py education_train.tsv education_valid.tsv 3

[65 0/135 1]
| F = 0: [42 0/16 1]
| | M2 = 0: [27 0/3 1]
| | | M4 = 0: [22 0/0 1]
| | | M4 = 1: [5 0/3 1]
| | M2 = 1: [15 0/13 1]
| | | M4 = 0: [14 0/7 1]
| | | M4 = 1: [1 0/6 1]
| F = 1: [23 0/119 1]
| | M4 = 0: [21 0/63 1]
| | | M2 = 0: [18 0/26 1]
| | | M2 = 1: [3 0/37 1]
| | M4 = 1: [2 0/56 1]
| | | P1 = 0: [2 0/15 1]
| | | P1 = 1: [0 0/41 1]
```

The numbers in brackets give the number of positive and negative labels from the training data in that part of the tree.

At this point, you should be able to answer questions 1-5 in the "Empirical Questions" of this handout. Write your solutions in the template provided.

4.3 Written: Empirical Questions

1. **[5 Points]** Report the validation accuracy of the decision tree classifiers trained on the three included datasets (with no limitation on maximum depth and all configurations like information gain threshold etc. set to the default value):

   - `small` dataset: 0.7143
   - `heart` dataset: 0.7423
   - `education` dataset: 1.0000

2. **[5 Points]** In the box below, paste the decision tree learned on `heart_train.tsv` under max depth limitation of 4. Be sure to follow the formatting requirement in section 4.2.2.

```
(venv) PS C:\Users\dhrum\OneDrive\Desktop\ML - CS522\F25 CS522 HW1\Programming> python de
cision_tree.py .\heart_train.tsv .\heart_val.tsv 4
[102 0/98 1]
| thalassemia = 0: [81 0/22 1]
| | chest_pain = 0: [17 0/17 1]
| | | fluoroscopy = 0: [14 0/4 1]
| | | | angina = 0: [10 0/0 1]
| | | | angina = 1: [4 0/4 1]
| | | fluoroscopy = 1: [3 0/13 1]
| | | fluoroscopy = 1: [3 0/13 1]
| | | fluoroscopy = 1: [3 0/13 1]
| | | fluoroscopy = 1: [3 0/13 1]
| | | | sex = 0: [3 0/2 1]
| | | fluoroscopy = 1: [3 0/13 1]
| | | | sex = 0: [3 0/2 1]
| | | fluoroscopy = 1: [3 0/13 1]
| | | | sex = 0: [3 0/2 1]
| | | fluoroscopy = 1: [3 0/13 1]
| | | | sex = 0: [3 0/2 1]
| | | fluoroscopy = 1: [3 0/13 1]
| | | fluoroscopy = 1: [3 0/13 1]
| | | fluoroscopy = 1: [3 0/13 1]
| | | | sex = 0: [3 0/2 1]
| | | | sex = 1: [0 0/11 1]
| | chest_pain = 1: [64 0/5 1]
| | | flat_ST = 0: [14 0/3 1]
| | | | abnormal_ecg = 0: [7 0/0 1]
| | | | abnormal_ecg = 1: [7 0/3 1]
| | | flat_ST = 1: [50 0/2 1]
| | | | sex = 0: [25 0/0 1]
| | | | sex = 1: [25 0/2 1]
| thalassemia = 1: [21 0/76 1]
| | fluoroscopy = 0: [17 0/23 1]
| | | angina = 0: [13 0/8 1]
| | | | flat_ST = 0: [4 0/6 1]
| | | | flat_ST = 1: [9 0/2 1]
| | | angina = 1: [4 0/15 1]
| | | | sex = 0: [0 0/2 1]
| | | | sex = 1: [4 0/13 1]
| | fluoroscopy = 1: [4 0/53 1]
| | | abnormal_ecg = 0: [4 0/22 1]
| | | | flat_ST = 0: [1 0/15 1]
| | | | flat_ST = 1: [3 0/7 1]
| | | abnormal_ecg = 1: [0 0/31 1]
```

3. **[10 Points]** Now use `heart_val.tsv` to perform reduced error pruning on the tree you learned in the previous question; break ties in favor of shorter trees. In the box below, paste the decision tree learned after pruning. Be sure to follow the formatting requirement in section 4.2.2.

   Note: After performing reduced error pruning, this is what the education dataset with max-depth 3 looks like:

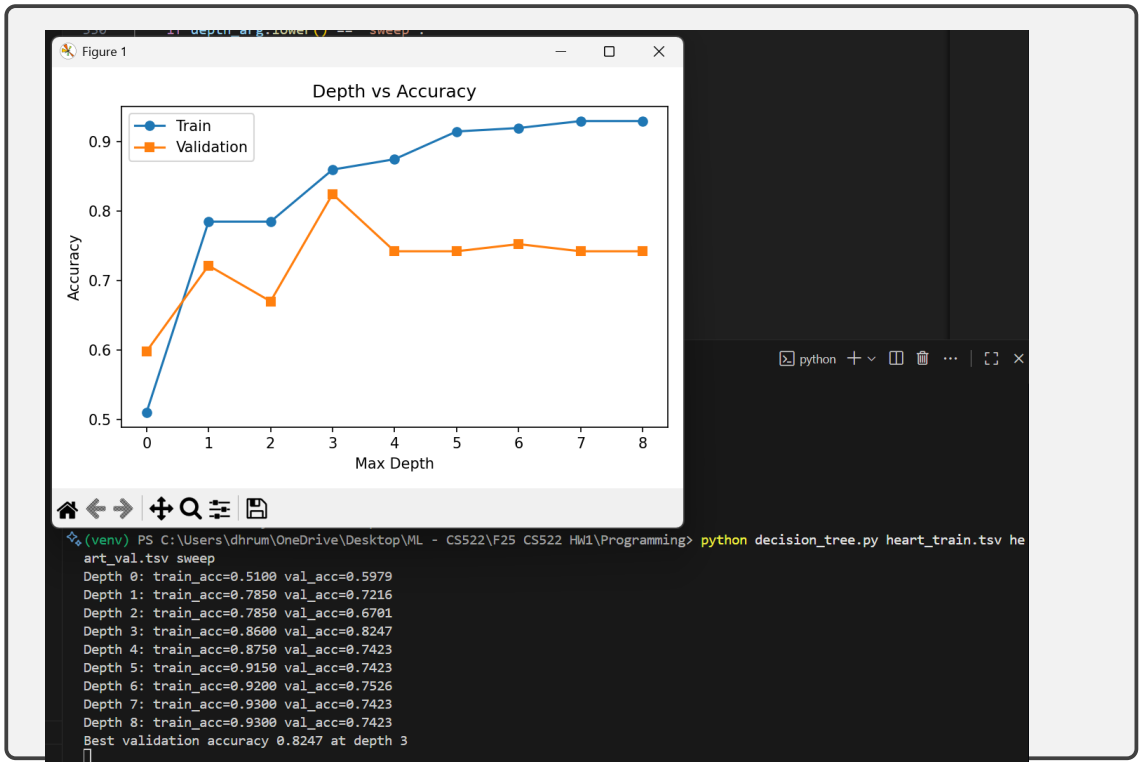
```
[65 0/135 1]
| F = 0: [42 0/16 1]
| | M2 = 0: [27 0/3 1]
| | M2 = 1: [15 0/13 1]
| | | M4 = 0: [14 0/7 1]
| | | M4 = 1: [1 0/6 1]
| F = 1: [23 0/119 1]
```

```
(venv) PS C:\Users\dhrum\OneDrive\Desktpython decision_tree.py heart_train.tsv heart_val.tsv 4
[102 0/98 1]
| thalassemia = 0: [81 0/22 1]
| | chest_pain = 0: [17 0/17 1]
| | | fluoroscopy = 0: [14 0/4 1]
| | | fluoroscopy = 1: [3 0/13 1]
| | chest_pain = 1: [64 0/5 1]
| thalassemia = 1: [21 0/76 1]
| | fluoroscopy = 0: [17 0/23 1]
| | | angina = 0: [13 0/8 1]
| | | angina = 1: [4 0/15 1]
| | fluoroscopy = 1: [4 0/53 1]
```

In the following questions, we will explore the performance of learned decision trees on validation data and introduce the problem of "overfitting". **All questions below are asked on the `heart` dataset (`heart_train.tsv` and `heart_val.tsv`).**

4. **[15 Points]** Iterate the maximum depth limitation in the range $[0, 8]$, collect the training accuracy and validation accuracy. Plot both metrics with respect to the maximum depth in a single figure. Label your axes as well as your training and validation accuracy plots.

   Is the validation accuracy monotonically increasing as the maximum depth limit increases? What about the training accuracy? Report the maximum validation accuracy observed and the maximum depth limit that at which it is achieved.



The phenomenon you have (hopefully) witnessed, wherein the model struggles to extrapolate effectively from the training dataset to the validation dataset, is widely referred to as *overfitting*. One approach to mitigating the overfitting dilemma involves constraining the complexity of the classifier, and this can be done during the construction of the decision tree (recall the diverse `Base Conditions` in the lecture slides).

Constraining the maximum depth of the decision tree is perhaps the most straightforward method for addressing the issue of overfitting but there are many alternatives.

5. **[10 Points]** Among these alternatives is the imposition of a restriction on the size of splitting nodes. Specifically, if a node comprises *fewer than* M data points during the training process, further splitting is prohibitted.

   Train a decision tree under different minimum splitting sizes, with C ranging from 0 to the number of data points in the `heart` dataset: what is the optimal minimum splitting size, and what is the validation accuracy under this value?

   Note: If there are multiple minimum splitting sizes that achieve the optimal validation accuracy, report the smallest of them.

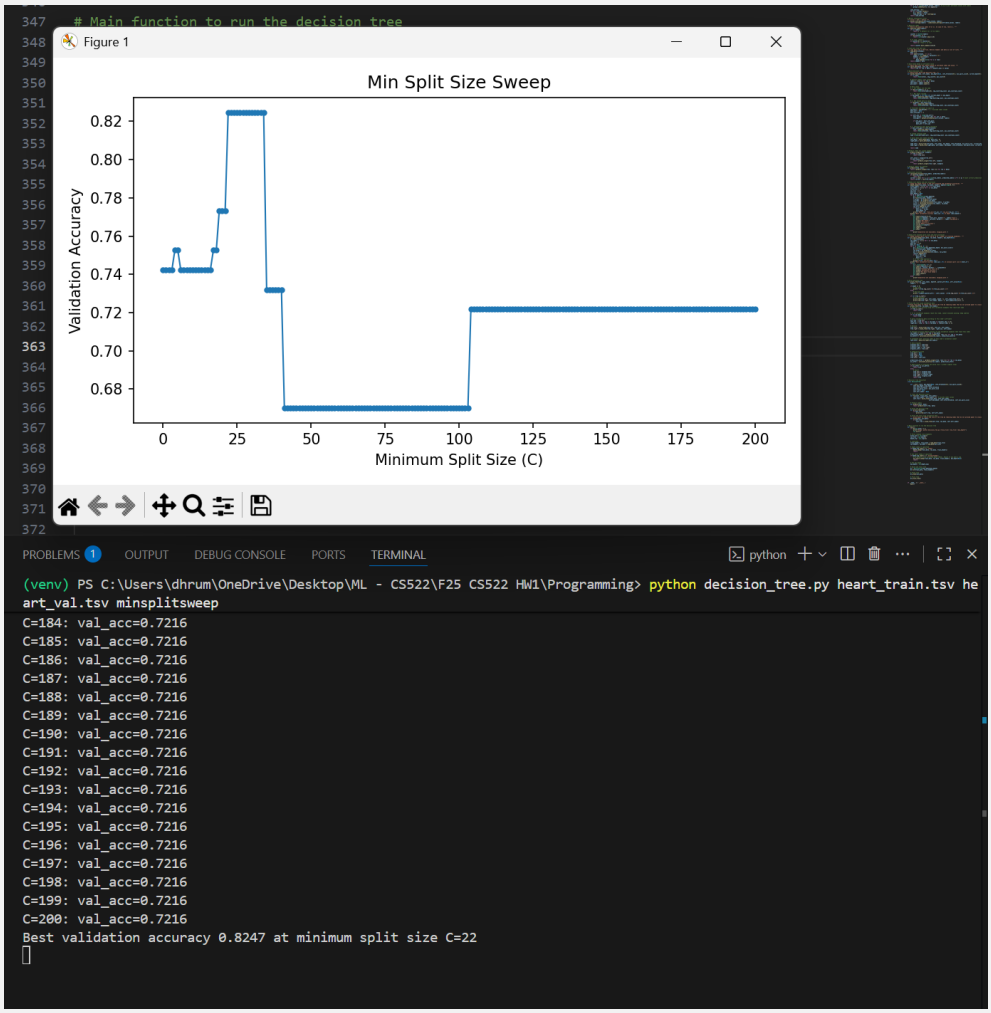
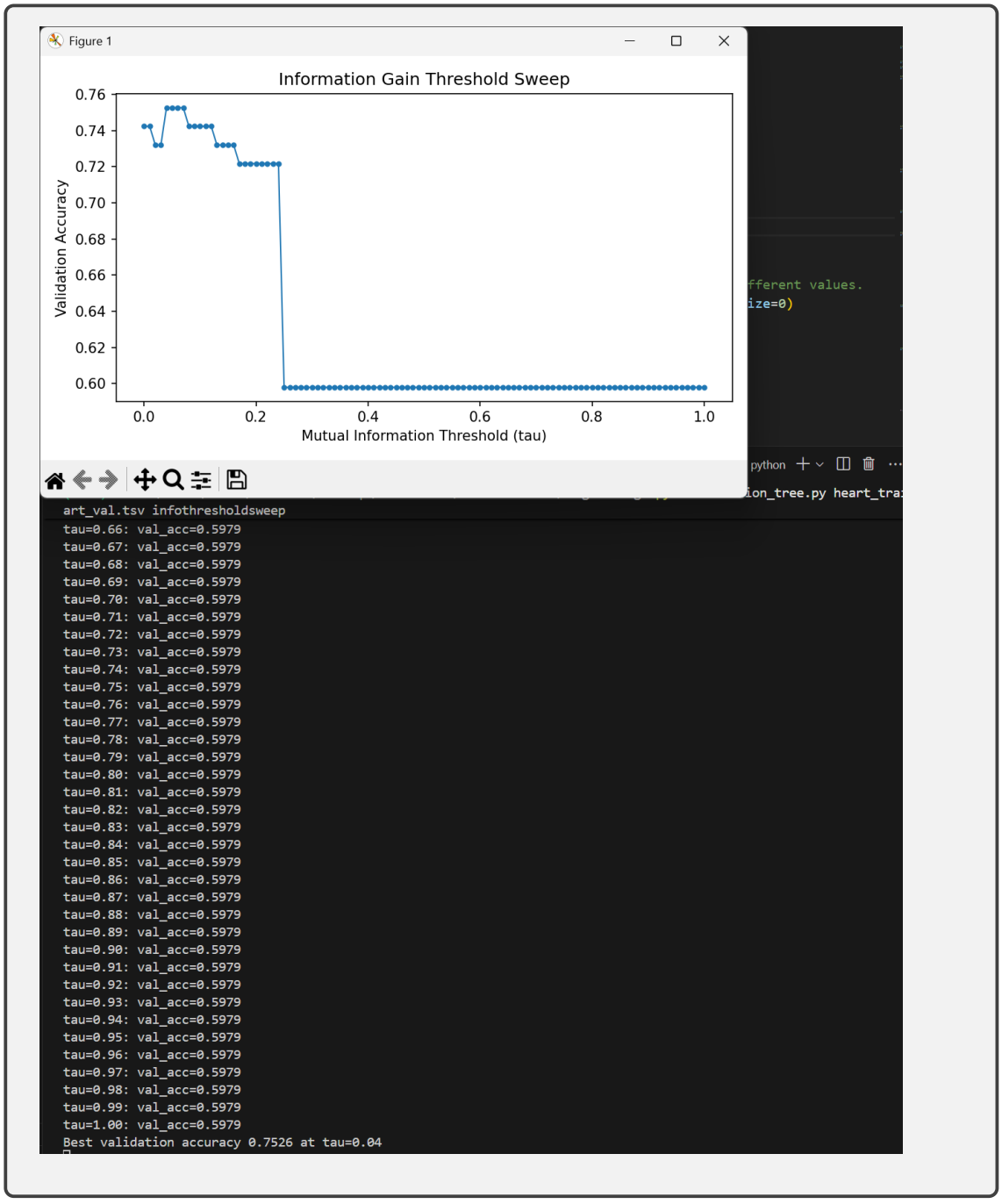


Figure 1
Min Split Size Sweep
Validation Accuracy
0.82
0.80
0.78
0.76
0.74
0.72
0.70
0.68
0
25
50
75
100
125
150
175
200
Minimum Split Size (C)
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
PORTS
TERMINAL
python
(venv) PS C:\Users\dhrum\OneDrive\Desktop\ML - CS522\F25 CS522 HW1\Programming> python decision_tree.py heart_train.tsv heart_val.tsv minsplitsweep
C=184: val_acc=0.7216
C=185: val_acc=0.7216
C=186: val_acc=0.7216
C=187: val_acc=0.7216
C=188: val_acc=0.7216
C=189: val_acc=0.7216
C=190: val_acc=0.7216
C=191: val_acc=0.7216
C=192: val_acc=0.7216
C=193: val_acc=0.7216
C=194: val_acc=0.7216
C=195: val_acc=0.7216
C=196: val_acc=0.7216
C=197: val_acc=0.7216
C=198: val_acc=0.7216
C=199: val_acc=0.7216
C=200: val_acc=0.7216
Best validation accuracy 0.8247 at minimum split size C=22

6. **[10 Points]** Another technique for mitigating overfitting involves changing the mutual information threshold. In this approach, at each node, if the best possible mutual information from a subsequent split falls below a designated threshold τ, the node is not divided any further.

   Train a decision tree with mutual information thresholds of $\{0.00, 0.01, \cdots, 0.99, 1.00\}$: what is the optimal information gain threshold, and what is the validation accuracy under this value?

   Note: If there are multiple mutual information thresholds that achieve the optimal validation accuracy, report the smallest of them.



4.4 Submission Instructions

Programming Please ensure you have completed the following file(s) for submission.

`decision_tree.py`

When submitting your solution, make sure to select and upload the file(s) shown above. **Any other files will be deleted.** Ensure the files have the exact same spelling and letter casing as above. We may manually grade your code for the purposes of assigning partial credit if your solutions to the empirical questions differ from our reference solutions.

Written Questions Make sure you have completed all questions from Written component (including the collaboration policy questions) in the template provided. When you have done so, please submit your document in **PDF format** to the corresponding assignment slot on Gradescope.

5 Collaboration Questions

1. (a) Did you receive any help whatsoever from anyone in solving this assignment? **Solution** No.

   (b) If you answered 'yes', give full details (e.g. "Jane Doe explained to me what is asked in Question 3.4")

   Solution

2. (a) Did you give any help whatsoever to anyone in solving this assignment? **Solution** No.

   (b) If you answered 'yes', give full details (e.g. "I pointed Joe Smith to section 2.3 since he didn't know how to proceed with Question 2")

   Solution

3. (a) Did you find or come across code that implements any part of this assignment? **Solution** Yes.

   (b) If you answered 'yes', give full details (book & page, URL & location within the page, etc.).

   - GeeksforGeeks (ID3 implementation concepts, entropy, information gain, pruning ideas)
     - https://www.geeksforgeeks.org/machine-learning/iterative-dichotomiser-3-id3-algorithm-from-scratch/
     - https://www.geeksforgeeks.org/machine-learning/pruning-decision-trees/
   - Python docs (API usage)
     - collections.Counter
     - csv module for TSV reading
   - Course materials: Lecture 1 (Introduction), Lecture 2 (Decision Trees), Lecture 3 (KNN) slides.
   - Tools: ChatGPT and GitHub Copilot for coding assistance (refactoring, helper function structure, sweep utilities).