CS37300 Homework	— Homework 1—	Last update: Jan 24, 2024
Due: Theory: Feb 5, 2024 (6pm); Programming: Feb 9, 2024 (6pm)		

Name: Derik Khoo PUID: 00370-28290

Instructions and Policy: You are allowed to study with others and use online resources for reference, however, the work you turn in must be your own. This means do not copy/paste from Stack Exchange (or from another student.) If you have worked closely with other students, provide their name(s) and a brief (at most one paragraph) description of the interaction; if we feel this oversteps the bounds, we will discuss it with you. Each student should write up their own solutions independently.

The requirements below are supposed to be followed in this and further homework assignments.

- For your theoretical submission:
  - YOU MUST INCLUDE YOUR NAME IN THE HOMEWORK.
  - The answers MUST be submitted via Gradescope.
  - Please write clearly and concisely - clarity and brevity will be rewarded. Refer to known facts as necessary.
  - Theoretical questions MUST include the intermediate steps to the final answer.
- For your programming answers:
  - The Python scripts will be submitted separately via Gradescope
  - Zero points will be given in any question where the Python code answer doesn't match the answer on Gradescope.
  - If the answer is/includes a plot, it should be added to your theoretical submission unless otherwise specified.

Your code is REQUIRED to run on Python 3.11 in an environment detailed in Homework 0 under the Python Installation section. While your code may run in other environments, any points lost due to environmental issues will not be regraded.

Please make sure you don't use any libraries that are not imported for you. If such a library is used, you will get 0 pt for the coding part of the assignment.

If your code doesn't run on Gradescope, then even if it compiles on another computer, it will still be considered not running and the respective part of the assignment will receive 0 pt.

Collaboration:
With Kartikeya Lanka, discussed about what constituted the entropy of a node in decision tree after splitting.

Theoretical Questions (24+14+16=54 pts)

Please submit your answers on Gradescope.

Q1 (24 pts): True or False Questions

Answer the following as True or False with a justification or example. Points are uniformly distributed within the questions.

1. Consider two random variables with the following joint probability density function:

$$f(X,Y) = 9X^2Y^2; \quad 0 \le X \le 1, \quad 0 \le Y \le 1.$$

Then, X and Y are independent.

True. This is because $9X^2Y^2 = (3X^2)(3Y^2)$. Then, $\int_0^1 3x^2\,dx = [x^3]_0^1 = 1$
Also, $\int_0^1 3y^2\,dy = [y^3]_0^1 = 1$
Hence, since the joint probability density function can be factored into product of 2 functions which integrates to 1 over $[0,1]$, X and Y are independent.

2. Conditional independence (CI) does not imply independence (IND) or vice-versa (if true, give two examples of both directions: CI does not imply IND and IND does not imply CI).

True. CI does not imply IND. Let $P(A) = \frac{8}{7}$, $P(B) = 0.2$, and $P(A|D) = 1$, $P(B|D) = 0.7$, for some events A, B, D. However, $P(A \cap B|D) = P(A|D)P(B|D)$. However, when B occurs, D occurs. Hence, $P(B|A) = P(B|D) = 0.7 \neq 0.2 = P(B)$. A and B are not independent, but conditionally independent given D.

3. X is a random variable with $E[X] = 100$ and $E[X^2] = 10,015$. Then $V(3X) = 135$.

True. $V(X) = E[X^2] - (E[X])^2 = 10015 - 10000 = 15$
$V(3X) = 3^2 V(X) = 9(15) = 135$

Q2. IND does not imply CI. Consider flipping two fair coins. Let A denote event coin 1 is heads, B denote event coin 2 is heads, and C is the event exactly one coin is heads. Thus, $P(A|B) = P(A)$, since A and B are independent. However, $P(A|C) = 0.5$, $P(B|C) = 0.5$, $P(A \cap B|C) = 0 \neq P(A|C)P(B|C)$.

4. (**4 pts**) If X_1 and X_2 are conditionally independent given Y, that is, $P(X_1, X_2|Y) = P(X_1|Y)P(X_2|Y)$, then $P(X_1|X_2, Y) = P(X_1|Y)$.

True. By definition of two events being conditionally independent. Suppose events A, B. $P(A \cap B|C) = P(A|C)P(B|C)$. This equivalently means $P(A|B \cap C) = P(A|C)$ or $P(B|A \cap C) = P(B|C)$.

5. K-Nearest Neighbors algorithm requires the calculation of distances between all training instances and the test instance for every prediction.

True. This is because in the algorithm, in step 2, it has to calculate the distance between the query example (test instance) and current example (training instance), and repeat for all training instances.

6. Consider a decision tree T learned on a training set of n instances. Assume that there are two identical instances X and X' (i.e. they have exactly the same attributes and labels) in the training set. Removing X' out of the training set produces a different T.

True. This is because the change in entropy values due to removal of X' could possibly lead to a different split being used instead.

Q2 (14 pts): Probability Review

1. (4 pts) Independence

A pair of fair dice is rolled and the following events are defined: A: First die shows 1, 2 or 3, B: First die shows 3, 4 or 5, C: The sum of numbers shown on first and second die equals 9. Answer the following questions. Justify your answer.

$\{(3,6),(4,5),(5,4),(6,3)\}$

1. Are events A, B and C pairwise independent?
2. Are events A, B and C mutually independent?

Q1. $P(A) = \frac{1}{2}$ $P(A|B) = \frac{1}{3}$

$P(B) = \frac{1}{2}$ $P(A|C) = 0$

$P(C) = \frac{1}{9}$ $P(B|C) = \frac{3}{9}$

$P(A \cap B) = P(A|B)P(B)$

$= \frac{1}{3} \times \frac{1}{2} = \frac{1}{6}$

$\neq P(A)P(B) = \frac{1}{2}(\frac{1}{2}) = \frac{1}{4}$

$P(A \cap C) = P(A|C)P(C)$

$= 0 \neq P(A)P(C) = \frac{1}{2} \times \frac{1}{9} = \frac{1}{18}$

$P(B \cap C) = P(B|C)P(C)$

$= \frac{3}{9} \times \frac{1}{9} = \frac{1}{12} \neq P(B)P(C) = \frac{1}{2} \times \frac{1}{9} = \frac{1}{18}$

∴ A, B and C are not pairwise independent.

Q2. No. This is because since A, B and C are not pairwise independent, thus, events A, B and C are not mutually independent.

2. (6 pts) Bayes Theorem

A factory produces two types of light bulbs, Type A and Type B. It is known that 30% of the light bulbs produced are Type A and 70% are Type B. Quality control tests have shown that 8% of Type A bulbs are defective, while 15% of Type B bulbs are defective. If a randomly chosen bulb is found to be defective, what is the probability that it is a Type A bulb?

$P(A) = 0.3$

$P(B) = 0.7$

Let event D denote that bulb is defective.

$P(D|A) = 0.08$

$P(D|B) = 0.15$

$$P(A|D) = \frac{P(A)P(D|A)}{P(A)P(D|A) + P(B)P(D|B)}$$

$$= \frac{0.024}{0.024 + (0.7 \times 15)}$$

$$= 0.18607 (\text{SF})$$

$$= 0.186 (3\text{SF})$$

3. (4 pts) Probability Distribution

Consider two continuous random variables x and y with joint probability density function

$$\begin{cases} f(x,y) = \frac{3}{16}xy^2 & 0 < x < k, \quad 0 < y < k, \\ f(x,y) = 0 & \text{otherwise,} \end{cases}$$

Recall that a *density* function has the following properties:
The probability of (x, y) must be one, such that

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y) = 1$$

(a) Find the value of k so that $f(x, y)$ is a valid joint probability distribution function.

(b) (**1 pt**) Find $P(x > y)$ as a function of k.

(c) (**1 pt**) Find $P(x + y > 1)$ as a function of k.

(d) (**1 pt**) Using the k value obtained from (a), prove that x and y independent.

(a) $\int\int \frac{3}{16}xy^2\,dy\,dx = \frac{3}{16}\int \frac{1}{3}y^3 x\,dx$

$= \frac{1}{16}\left(\frac{1}{2}x^2y^3\right)$

$= \frac{1}{32}x^2y^3$

For $\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y) = 1$,
since x and y are bounded by k,

$\frac{1}{32}(k)^2(k)^3 = 1$

$k^5 = 32$

$k = 2$

∴ value of $k=2$ so that $f(x,y)$ is a valid joint probability distribution function.

(b) $P(x>y) = \int_0^k \int_0^x \frac{3}{16}xy^2\,dy\,dx$

$= \frac{3}{16}\int_0^k \frac{1}{3}x^3 \cdot x\,dx$

$= \frac{1}{16}\int_0^k x^4\,dx$

$= \frac{1}{16}\left(\frac{k^5}{5}\right)$

$= \frac{k^5}{80} =$

(c) $P(x+y>1) = P(x>1-y)$

$= \frac{3}{16}\int_0^k \int_{1-y}^k xy^2\,dx\,dy$

$= \frac{3}{16}\int_0^k \frac{k^2-(1-y)^2}{2}y^2\,dy$

$= \frac{3}{32}\int_0^k \left[k^2-(1-2y+y^2)\right]y^2\,dy$

$= \frac{3}{32}\int_0^k (k^2-1+2y-y^2)y^2\,dy$

$= \frac{3}{32}\int_0^k k^2y^2-y^2+2y^3-y^4\,dy$

$= \frac{3}{32}\left[\frac{k^5}{3}-\frac{k^2}{3}+\frac{k^4}{2}-\frac{k^5}{5}\right]$

$= \frac{3}{32}\left(\frac{2k^5}{15}-\frac{k^2}{3}+\frac{k^4}{2}\right) =$

Q(d) for $k=2$, $f_y(y) = \int_0^2 \frac{3}{16} xy^2 dx$

$= \frac{3}{16}\left[\frac{x^2}{2} y^2\right]_0^2$

$= \frac{3}{16}(2y^2)$

$= \frac{6}{16} y^2 = \frac{3}{8} y^2$

$f_x(x) = \int_0^2 \frac{3}{16} xy^2 dy$

$= \frac{3}{16}\left[\frac{1}{3} xy^3\right]_0^2 = \frac{1}{16}\left[xy^3\right]_0^2$

$= \frac{1}{2} x$

$\therefore f(x,y) = \frac{3}{16} xy^2$

$f_x(x) f_y(y) = \frac{6}{16} y^2 \cdot \frac{1}{2} x$

$= \frac{3}{16} xy^2$

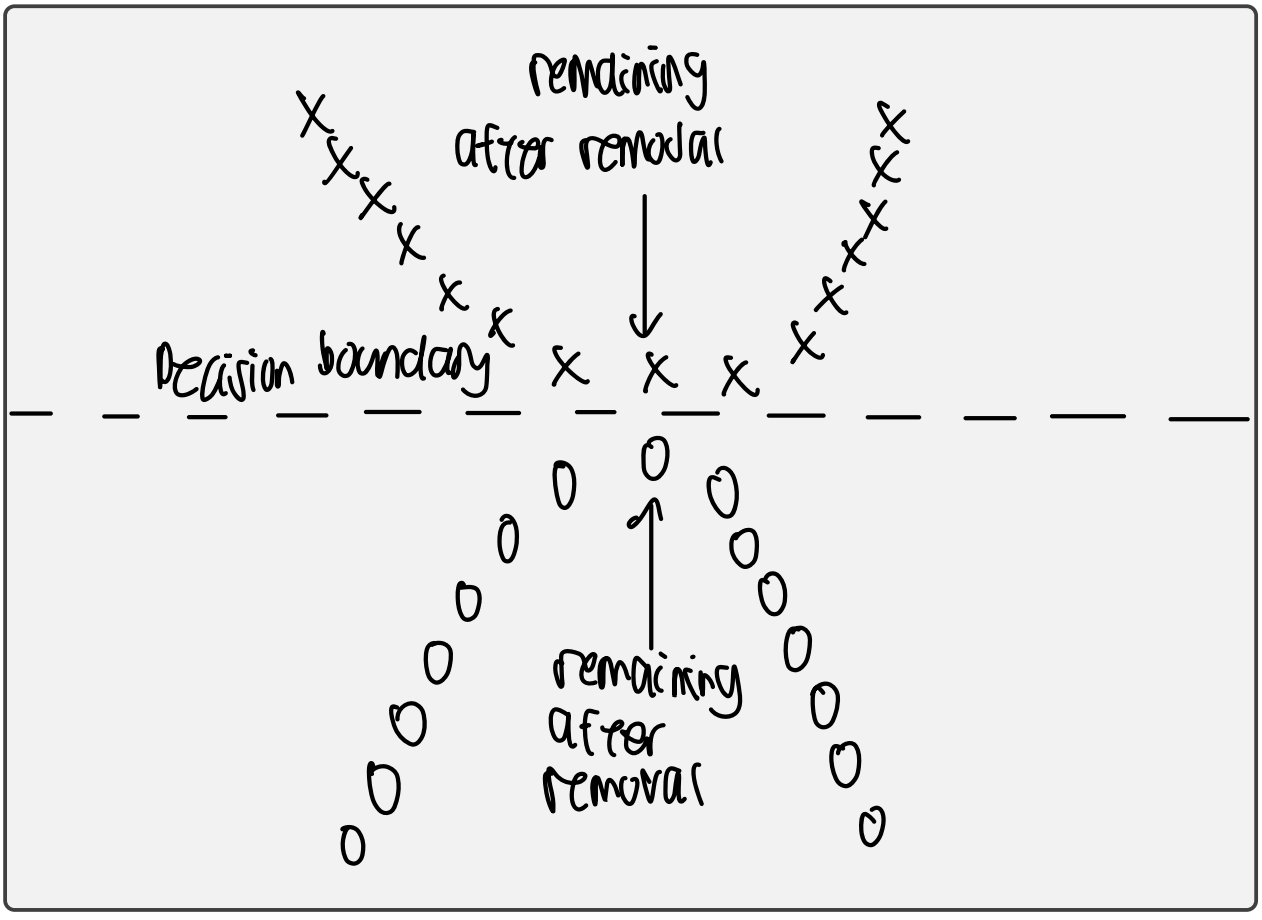
$\therefore f(x,y) = f_x(x) f_y(y)$,

Thus, x and y are independent.

Q3 (16 pts): KNN and Decision Trees

1. (8 pts) Explain your answer within several sentences:

(a) (**4 pts**) Consider a 1-NN classification task with two class labels (X and O) and two-dimensional features (i.e., we want to classify points in a 2D space). Draw a training dataset with 15 elements for each class (30 in total), such that we could remove 14 points from class X and 14 points from class O, (*i.e.* leaving a single point for each class) and the decision boundary computed before and after removal would be the same. Indicate these 2 remaining points left after removal.



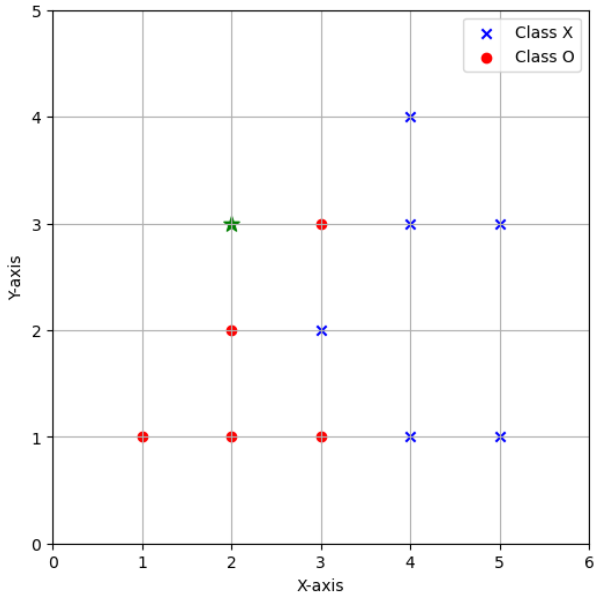


Figure 1: KNN classifier

(b) (**4 pts**) Figure 1 shows a set of points classified as 'X' or 'O'. Consider the k-nearest neighbor algorithm using Euclidean distance to classify new points. Given k=3, answer the following:

(a) What is the class of '*'? Justify.

(b) Identify the points in the training set that would be misclassified (Don't count '*' in the training set).

[6][a] Class of "*" is 0. Since distance metric is Euclidean distance, and k=3, the 3 closest points are (2,2), (3,2) and (3,3) with Euclidean distances of 1, $\sqrt{2}$ and 1 respectively. Taking majority vote, since 2 out of the 3 points are labelled class 0, class of "*" is thus 0.

[6][b] Points (3,2) and (3,3). This is because using k-NN where k=3, the majority vote would classify both of these 2 points as the opposite label instead.

2. (8 pts) Build a decision tree

We are trying to build a classifier to figure out if a patient is likely to have heart disease. We gathered data about 15 different patients, including their age group (35-50,51-65), gender (male, female), blood pressure(high, normal, low) and cholesterol(high, normal). The data is reported in the table below:

Table 1: Patient Dataset

ID	Age group	Gender	Blood Pressure	Cholesterol	Heart Disease
1	51-65	Male	High	High	Yes
2	51-65	Female	Normal	Normal	No
3	51-65	Male	High	High	Yes
4	51-65	Female	Low	Normal	No
5	36-50	Male	High	Normal	No
6	51-65	Female	Low	High	Yes
7	36-50	Female	Low	High	No
8	51-65	Male	High	Normal	Yes
9	36-50	Female	Normal	Normal	No
10	51-65	Male	Low	High	Yes
11	36-50	Female	High	Normal	Yes
12	51-65	Male	Normal	High	Yes
13	51-65	Male	Normal	Normal	No
14	36-50	Female	High	Normal	No
15	36-50	Male	Low	High	No

(a) (**6 pts**) Using this data build a decision tree (use **information gain**) to decide whether a patient would have heart disease or not, showing at each level how you decided which attribute to expand next. Stop when the training set error (i.e. the fraction of points in the training set that it misclassified) drops below 0.15 **In case of a tie, choose the attributes in alphabetical order.** For logarithmic calculations, use **2** as the base.

Entropy (Heart Disease) $= -\frac{7}{15}\log_2\frac{7}{15} - \frac{8}{15}\log_2\frac{8}{15}$

$\simeq 0.99679$ (5sf)

IG (Age group) $= 0.99679 - \frac{9}{15}\left[-\frac{6}{9}\log_2\frac{6}{9} - \frac{3}{9}\log_2\frac{3}{9}\right]$

$- \frac{6}{15}\left[-\frac{1}{6}\log_2\frac{1}{6} - \frac{5}{6}\log_2\frac{5}{6}\right]$

$= 0.18581$ (5sf)

IG (Gender) $= 0.99679 - \frac{8}{15}\left[-\frac{5}{8}\log_2\frac{5}{8} - \frac{3}{8}\log_2\frac{3}{8}\right]$

$- \frac{7}{15}\left[-\frac{2}{7}\log_2\frac{2}{7} - \frac{5}{7}\log_2\frac{5}{7}\right]$

$= 0.08997$ (5sf)

IG (Blood Pressure) $= 0.99679 - \frac{6}{15}\left[-\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6}\right]$

$- \frac{4}{15}\left[-\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4}\right]$

$- \frac{5}{15}\left[-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right]$

$= 0.089982$ (5sf)

IG (Cholesterol) $= 0.99679 - \frac{7}{15}\left[-\frac{5}{7}\log_2\frac{5}{7} - \frac{2}{7}\log_2\frac{2}{7}\right]$

$- \frac{8}{15}\left(-\frac{2}{8}\log_2\frac{2}{8} - \frac{6}{8}\log_2\frac{6}{8}\right)$

$= 0.16132$ (5sf)

Training error $= \frac{(3sf)}{15} = 0.267$ (3sf)

Thus, since IG(Age Group) has highest value, the first split in the decision tree is by Age Group.

$$\text{Entropy(Age Group 51-65)} = -\frac{6}{9}\log_2\frac{6}{9} - \frac{3}{9}\log_2\frac{3}{9}$$
$$= 0.918830 \text{ (5sf)}$$

$$\text{Entropy(Age Group 36-50)} = -\frac{1}{6}\log_2\frac{1}{6} - \frac{5}{6}\log_2\frac{5}{6}$$
$$= 0.65002 \text{ (5sf)}$$

Considering Age Group 51-65,

$$IG(\text{Gender}) = 0.918830 - \frac{6}{9}\left[-\frac{5}{6}\log_2\frac{5}{6} - \frac{1}{6}\log_2\frac{1}{6}\right] - \frac{3}{9}\left[-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right]$$
$$= 0.079 \text{ (3sf)}$$

$$IG(\text{Blood Pressure}) = 0.918830 - \frac{3}{9}[0] - \frac{3}{9}\left[-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right] - \frac{3}{9}\left[-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right] = 0.306 \text{ (3sf)}$$

$$IG(\text{Cholesterol}) = 0.918830 - \frac{5}{9}[0] - \frac{4}{9}\left[-\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4}\right] = 0.557 \text{ (3sf)}$$

∴ Since IG(Cholesterol) has highest value, next split is by Cholesterol for age group 51-65.

Considering age group 36-50,

$$IG(\text{Gender}) = 0.65002 - \frac{2}{6}[0] - \frac{4}{6}\left[-\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4}\right] = 0.109 \text{ (3sf)}$$

$$IG(\text{Blood Pressure}) = 0.65002 - \frac{3}{6}\left[-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right] - \frac{1}{6}[0] - \frac{2}{6}[0]$$
$$= 0.191 \text{ (3sf)}$$

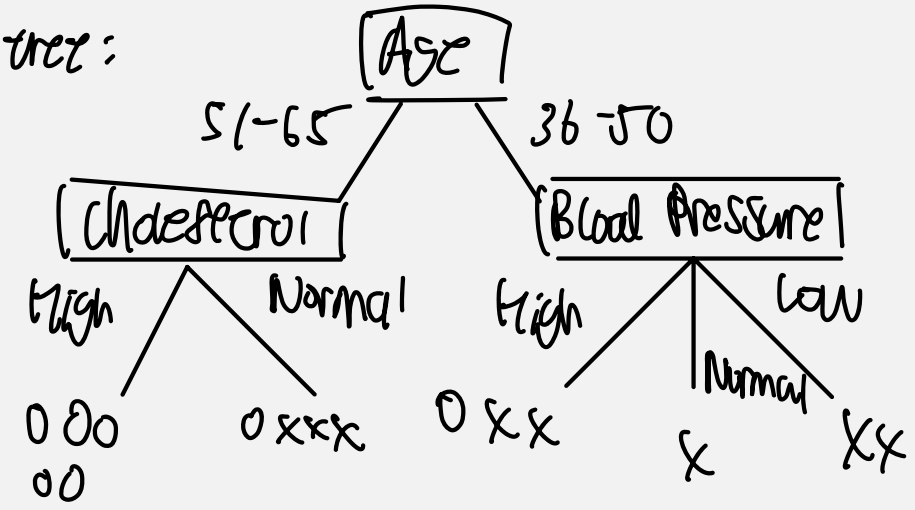
$$IG(\text{Cholesterol}) = 0.65002 - \frac{2}{6}[0] - \frac{4}{6}\left[\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4}\right]$$
$$= 0.109 \text{ (3sf)}$$

∴ IG(Blood Pressure) has highest value, next split is by Blood Pressure for age group 36-50.

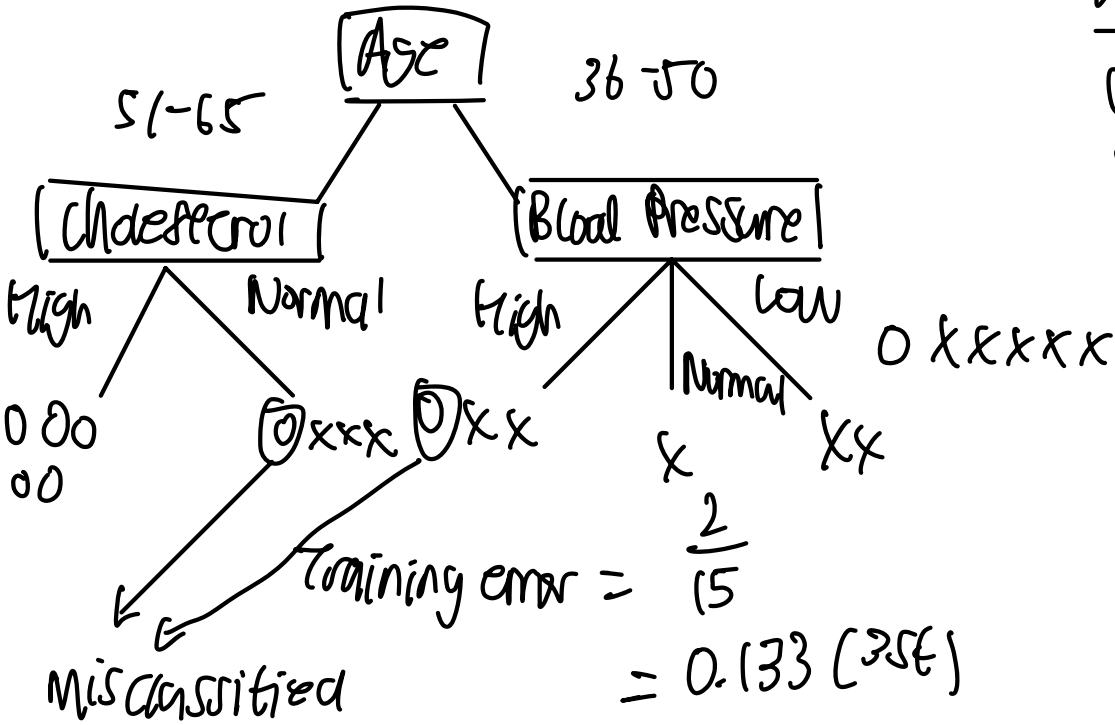
(b) (**2 pts**) State the IDs of the patients who are most likely to have heart disease by referring to the obtained decision tree.

IDs are: 1, 3, 6, 10, 12

Decision tree:



(a) Continuation



Legend:

0 — Heart Disease

1 — No heart disease

(Ground truth labels)

Training error = $\frac{2}{15}$

= 0.133 (3SF)

Misclassified

Programming Part (12 + 23 + 11 = 46 pts)

General Instructions

For this assignment, please download the provided `hw1.zip` file on **Brightspace**.

Overview

In this assignment, you will learn and implement a KNN classifier and also a decision tree classifier.

Files

You will fill in functions in knn.py, scorer.py, and dt.py. After you finish, you will submit these as well as any other files mentioned in the assignment to Gradescope.

Asides from these three files, you will also find utils.py in the same folder. You do not need to modify these two files. utils.py contains some helper functions.

Packages

You will need the following packages for this assignment:

- `numpy`
- `pandas`

If you want, you can install `tqdm` to see a progress bar when running the autograder. This is not required for this class and might not be heavily focused on in future assignments, but it is a useful package to know if you are doing any machine learning work in Python.

These packages should be installed if you have installed Anaconda. If you are not using Anaconda, you can install them using `pip` or `conda`. For example, you can run `pip install numpy` in the terminal to install `numpy`.

Note that for this assignment, you should **not** use any other packages. You may see some other packages in the skeleton code, but they are only used for testing and grading. If you use any other packages, you may lose points. If you are not sure whether you can use a package, please ask the course staff.

Evaluation

Unless otherwise specified, your code will be evaluated by an autograder on Gradescope using the same environment as detailed below. You should make sure your code runs without errors in this environment. Otherwise, you may lose points. You can submit your code to Gradescope as many times as you want. We will only grade the latest submission.

There will be two types of test cases in the autograder: public (local) and hidden. Public test cases are visible to you, and you can see the results after you submit your code. You can also run the public test cases

locally by running `python <filename>.py` in the same folder as your code. Passing all public test cases **does not** guarantee you will get full credit for the assignment.

Hidden test cases, however, are not visible to you, and you will not be able to see the results until your grade is published. The final score you get for this assignment is the sum of the scores of all public and hidden test cases.

Getting Help

If you have any questions about this assignment, please contact the course staff for help, preferably during office hours. You can also post your questions on the course forum. However, please do not post any code publicly. When asking questions, you should describe your problem in words and post the relevant code snippet. Please avoid showing a screenshot of your code to TAs and ask "why my code is not working".

1 KNN (1 + 1 + 2 + 2 + 6 = 12 pts)

In this part, you will implement a KNN classifier. From our lecture, you should know that KNN is a non-parametric classifier, which means it does not make any assumption about the underlying distribution of the data. Instead, it uses the training data directly to make predictions. In this assignment, you will implement the KNN classifier using the Manhattan distance as the distance metric.

You will write your code in knn.py.

1.1 (1 pts) KNN - Constructor

Under `class KNearestNeighbor`, fill in the constructor (`__init__`) so that it takes in the number of neighbors `k` and stores it as an attribute `self.k`. You should also initialize variables `self.X_train` and `self.y_train` to be `None`. These two attributes will be set in the `fit` method.

1.2 (1 pts) KNN - Fit

Under `class KNearestNeighbor`, fill in the method fit so that it takes in the training data `X_train` and `y_train` and stores them as attributes `self.X_train` and `self.y_train`. Note that `X_train` is a 2D array of shape (`n_samples`, `n_features`), and `y_train` is a 1D array of shape (`n_samples`,).

The two `assert` statements are there to help you debug. The first one checks whether the number of training samples is the same as the number of training labels. The second one checks whether the number of training samples is greater than or equal to the number of neighbors `k`. If either of them fails, you should raise a `ValueError` with an appropriate error message.

1.3 (2 pts) KNN - Manhattan Distance

Under `class KNearestNeighbor`, fill in the method `calc_distance` so that it takes in a single test sample `x` and returns a 1D array of shape `(n_samples,)` containing the Manhattan distance between `x` and each training sample in `self.X_train`.

1.4 (2 pts) KNN - Get Neighbors

Under `class KNearestNeighbor`, fill in the method `get_top_k` so that it takes in a 1D array `distances` and returns a 1D array of shape `(self.k,)` containing the indices of the top `self.k` smallest elements in `distances`. You may find the function `np.argsort` useful.

1.5 (6 pts) KNN - Predict

Under `class KNearestNeighbor`, fill in the method predict so that it takes in a 2D array `X_predict` of shape (`n_samples`, `n_features`) and returns a 1D array of shape (`n_samples`,) containing the predicted labels for each sample in `X_predict`. You should use the `get_top_k` method you implemented in Q1.4 to get the indices of the top `self.k` smallest elements in `distances`. Then, you can use these indices to get the corresponding labels from `self.y_train` and return them as the predicted labels. You may find the function `np.bincount` and the function `np.argmax` useful.

2 Scorer (1 + 4 + 4 + 3 + 3 + 4 + 4 = 23 pts)

In this part, you will implement a decision tree classifier. From our lecture, you should know that a decision tree classifier is a tree-structured classifier that makes decisions based on the values of features. In this assignment, you will first implement a scorer that measures the quality of a split in terms of both information gain and Gini impurity. Then, you will implement a decision tree classifier that uses the scorer to make decisions.

You will write your code in `dt.py`.

2.1 (1 pts) Scorer - Constructor

Under `class Scorer`, fill in the constructor (`__init__`) so that it takes in a string type and stores it as an attribute `self.type`. The string type can be either `"information"` or `"gini"`. The function also takes in a 1D array `class_labels` containing the class labels of the training data and `alpha` which is the parameter used for Laplace smoothing. You should store the unique class labels in `self.class_labels` and the parameter `alpha` in `self.alpha`.

2.2 (4 pts) Scorer - Class Probabilities

Under `class Scorer`, fill in the method `compute_class_probabilities` so that it takes in a 1D array `labels` containing the class labels of the training data and returns a dictionary containing the class probabilities. The keys of the dictionary are the unique class labels and the values are the corresponding class probabilities. You should use Laplace smoothing with parameter `self.alpha` to compute the class probabilities. Make sure your code handles the case where the input array `labels` is empty and returns a non-empty dictionary with all class probabilities.

2.3 (4 pts) Scorer - Data Subsets

Under `class Scorer`, fill in the method `subset_data` so that it takes in a 2D array `data` containing the training data, a 1D array `labels` containing the class labels of the training data, an integer `split_attribute` indicating the index of the feature to split on, and a value `split_value` indicating the value of the feature to split

on. The method should return a tuple of two arrays (`data_subsets`, `labels_subsets`) where `data_subsets` is a 2D array and `labels_subsets` is a 1D array. Each row in `data_subsets` and `labels_subsets` should correspond to a subset of the training data and the corresponding class labels that originally have the feature value `split_value` at the feature index `split_attribute`. You may find the function `np.where` useful.

2.4 (3 pts) Scorer - Information Score

Under `class Scorer`, fill in the method so that it takes in a 1D array `labels` containing the class labels of the training data and returns the information score (entropy) of the class labels. You should use the class probabilities computed in Q2.2 to compute the information score. We use `log2` as the base of the logarithm.

2.5 (3 pts) Scorer - Scorer - Gini Score

Under `class Scorer`, fill in the method `gini_score` so that it takes in a 1D array `labels` containing the class labels of the training data and returns the Gini score of the class labels. You should use the class probabilities computed in Q2.2 to compute the Gini score.

2.6 (4 pts) Scorer - Information Gain

Under `class Scorer`, fill in the method `information_gain` so that it takes in a 2D array `data`, a 1D array `labels`, and an integer `split_attribute` indicating the index of the feature to split on. The method should return the information gain of splitting on the feature at index `split_attribute`. You should use the information score (entropy) computed in Q2.4 to compute the information gain.

2.7 (4 pts) Scorer - Gini Gain

Under `class Scorer`, fill in the method `gini_gain` so that it takes in a 1D array `labels`, and an integer `split_attribute` indicating the index of the feature to split on. The method should return the Gini gain of splitting on the feature at index `split_attribute`. You should use the Gini score computed in Q2.5 to compute the Gini gain.

3 Decision Tree (2 + 2 + 7 = 11 pts)

In this part, you will implement a decision tree classifier using the scorer you implemented in Part 2. You will write your code in `dt.py`.

You will see four classes in `dt.py`:

- `class Node(ABC)`, which is an abstract class that represents a node in the decision tree. You may ignore this class.
- `class Leaf(Node)`, which is a subclass of Node that represents a leaf node in the decision tree.
- `class Split(Node)`, which is a subclass of Node that represents a split node in the decision tree.
- `class DecisionTree`, which is a class that represents a decision tree classifier.

3.1 (2 pts) Decision Tree - Predicting Class Probabilities

Under `class Leaf`, fill in the method `predict_class_probabilities` so that it takes in a 2D array `X` containing the data and returns a 2D array containing the predicted class probabilities for each data point in `X`.

Note that since this is a leaf node, the predicted class probabilities should be the same for all data points in `X`. You should use `self.class_probabilities` and `self.class_labels` to compute the predicted class probabilities, as they represent the class probabilities of the training data that have reached this leaf node.

3.2 (2 pts) Decision Tree - Predicting Leaf Node Class

Under `class Leaf`, fill in the method `predict` so that it takes in a 2D array `X` containing the data and returns a 1D array containing the predicted class for each data point in `X`. The probabilities of each class for each data point in `X` are already computed in `probabilities` in the skeleton code. You should use `self.class_labels` to compute the predicted class for each data point in `X` based on the probabilities.

3.3 (7 pts) Decision Tree - Building the Tree

Under `class DecisionTree`, fill in the method `build_tree` so that it takes in a 2D array `data` containing the training data, a 1D array `labels` containing the class labels of the training data, an integer `max_depth` indicating, and a set `exclude` containing the indices of the features to exclude. The method should return a `Node` object representing the node in the decision tree. You should use the scorer you implemented in Part 2 to compute the gain of each feature and split the feature with the highest gain. You should use the `Split` and `Leaf` classes to represent the split and leaf nodes in the decision tree. Do not modify the code that is not guarded by `YOUR CODE HERE` and `END OF YOUR CODE`.

Submission

You will submit the following files to Gradescope:

- `knn.py`
- `scorer.py`
- `dt.py`