1. [8 points] Please select the most suitable type of machine learning approaches from "supervised learning", "unsupervised learning", "semi-supervised learning", "reinforcement learning", and "not a learning problem" which is best fit for the following using scenarios. Please first choose one of the five answers and explain your answer by specifying the input data, labels, and your reasons.

(a) A model is trained using patient medical records, including genetic information and family history, to predict the likelihood of developing diseases like Alzheimer's or diabetes in the future. The dataset includes past cases labeled with whether the patient developed the disease or not. ← labels

Supervised learning (past cases labeled with prescription)
Input: patient medical records

(b) A system is designed to optimize drug dosage recommendations for diabetes patients by continuously adjusting medication levels and monitoring blood sugar responses. It learns through trial and error, aiming to maintain stable glucose levels while minimizing side effects. The system receives feedback based on patient outcomes and refines its dosage recommendations over time.

Reinforcement learning
(Receives feedback, learns through trial and error)
Input: feedback based on patient outcomes

(c) A hospital's billing system calculates patient invoices based on standardized rates for treatments, medications, and services used. The system follows a predefined set of rules and does not improve over time based on data.

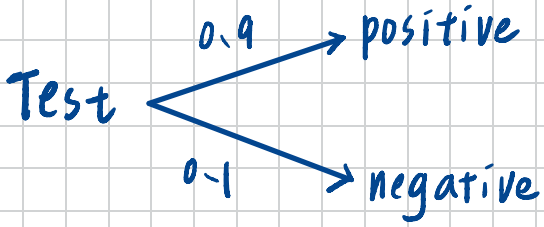
Not a learning problem
(The system doesn't improve over time based on data)

(d) A wearable heart monitor records patients' heartbeats and detects unusual patterns that could indicate potential heart conditions. The system does not have predefined examples of anomalies but is learned by identifying patterns that deviate from normal heart activity.

unsupervised learning (learned by different patterns)
No predefined examples
⇒ No labels, The model may learn by itself to identify abnormal patterns.

2. [10 points] Assume a disease is so rare that it is seen in only one person out of every million. Assume also that we have a test that is useful in that if a person has the disease, there is a 90 percent chance that the test result will be positive; however, the test is not perfect, and there is a one in a thousand chance that the test result will be positive on a healthy person. Assume that a new patient arrives, and the test result is positive. What is the probability that the patient has the disease? Please write down the procedure to show how you derive your answer.

$P_D = \frac{1}{10^6}$ $\quad P(A|B) = \frac{P(B|A)\cdot P(A)}{P(B)}$, $P(B|A) = \frac{P(A\cap B)}{P(A)}$



false positive : 0.001

$$P(\text{Disease}\mid\text{Positive}) = \frac{P(\text{Positive}\mid\text{Disease})\cdot P(\text{Disease})}{P(\text{Positive})}$$

($P(\text{Positive}\mid\text{Disease}) = 0.9$, $P(\text{Disease}) = \frac{1}{10^6}$)

$$P(\text{Positive}) = P(\text{Positive}\mid\text{Disease})\cdot P(\text{Disease}) + P(\text{Positive}\mid\text{No Disease})\cdot P(\text{No Disease})$$

$$= 0.9\times\frac{1}{10^6} + 0.001\times\left(1-\frac{1}{10^6}\right) \approx 0.001$$

$$P(\text{Disease}\mid\text{Positive}) = \frac{0.9\times\frac{1}{10^6}}{0.001} = 9\times10^{-4} \#$$

Weight Loss	Headache	Fever	Cough	Prescription
Obvious	Yes	Yes	Yes	Yes
Obvious	Yes	Yes	No	Yes
No	Yes	Yes	Yes	No
Mild	No	Yes	Yes	No
Mild	No	No	Yes	No
Mild	No	No	No	Yes
No	No	No	No	No
Obvious	No	Yes	Yes	Yes
Obvious	No	No	Yes	No
No	No	Yes	No	Yes
Mild	No	No	Yes	Yes
Obvious	No	Yes	No	Yes
Obvious	Yes	No	No	Yes
Mild	No	No	No	No

(a) [10 points] Given that the features (Weight Loss, Headache, Fever, Cough) of a patient are (Obvious, Yes, No, No), please predict whether to use the medicine (prescription) using maximum a-posterior approach (naïve bayes). Derive the final answer by calculating prior, likelihood, and posterior. Please provide the procedure and details calculated by hand.

$P(y|x) \propto P(y) \cdot p(x_1|y) \cdot p(x_2|y) \cdot p(x_3|y) \cdot p(x_4|y)$

$P(\text{Yes}) = \frac{8}{14}$

$P(\text{No}) = \frac{6}{14}$

$p(\text{Weight loss} = \text{Obvious} \mid \text{Yes}) = \frac{5}{8}$

$p(\text{Headache} = \text{Yes} \mid \text{Yes}) = \frac{3}{8}$

$p(\text{Fever} = \text{No} \mid \text{Yes}) = \frac{3}{8}$

$p(\text{Cough} = \text{No} \mid \text{Yes}) = \frac{5}{8}$

$p(\text{Weight loss} = \text{Obvious} \mid \text{No}) = \frac{1}{6}$

$p(\text{Headache} = \text{Yes} \mid \text{No}) = \frac{1}{6}$

$p(\text{Fever} = \text{No} \mid \text{No}) = \frac{4}{6}$

$p(\text{Cough} = \text{No} \mid \text{No}) = \frac{2}{6}$

Posterior for Yes:

$P(\text{Yes} \mid \text{features}) = \frac{8}{14} \cdot \frac{5}{8} \cdot \frac{3}{8} \cdot \frac{3}{8} \cdot \frac{5}{8} = 0.031$

$>$

Posterior for No:

$p(\text{No} \mid \text{features}) = \frac{6}{14} \cdot \frac{1}{6} \cdot \frac{1}{6} \cdot \frac{4}{6} \cdot \frac{2}{6} = 0.002$

$\Rightarrow$ Prescription = Yes

(b) [2 points] What is the difference when we have features in discrete and continuous values (the previous coding problem versus this coding problem)?

Hint: If the features are continuous values, how do we compute likelihood?

Discrete values : We can compute the likelihood by counting the frequency

Continuous values : It's unable to count the frequency

→ Instead, we assume the features follow a Gaussian distribution, and compute the likelihood using the probability density function.

(g) [5 points] Compare the above tree-based approaches with the naïve bayes classifier. Please state which classifier performs better and explain with reason. Your answer should include explanations about the differences between classifiers and the reason that the classifier is suitable for the data.

1. **Decision Tree**:
   - Simple, interpretable model that divides the feature space using a tree structure
   - Prone to overfitting, especially with deep trees
   - Limited in capturing complex relationships without proper pruning
2. **Random Forest**:
   - Combination of multiple decision trees, each trained on bootstrap samples
   - Reduces overfitting through averaging and random feature selection
   - Handles high-dimensional data well without feature scaling
   - Robust to outliers and non-linear relationships
3. **Gradient Boosting**:
   - Sequential ensemble that builds trees to correct errors from previous trees
   - Often achieves high accuracy but can overfit without careful tuning
   - More sensitive to noisy data and outliers than Random Forest
   - Requires more careful parameter tuning
4. **Naive Bayes**:
   - Probabilistic classifier based on Bayes' theorem
   - Assumes features are conditionally independent given the class
   - Performs well with limited training data and high-dimensional spaces
   - Less computationally intensive than tree-based methods

In this case, the dataset contains complex and non-linear relationships between features that aren't well-suited for the independence assumption of Naive Bayes. Tree-based models are more capable of handling diverse training samples. Among the other three models, I found that the random forest model performed the best in test accuracy, class 0, and class 1 F-1 scores after parameter tuning and feature selection. Due to the ensemble mechanism, the random forest model can concentrate more on the relationship between each feature after parameter tuning and feature selection. The decision tree model is subject to overfitting and may be too simple for the complex patterns in this dataset. Gradient boosting already performed well before tuning and feature selection, showing that it might reach its performance limit for this dataset.

Language Models Usage Policy for This Homework

We allow students to use language models for coding problems, i.e., the delivery of Python code. Other problems are not allowed to use language models.
If students use any language model for their homework, they must:

1. **Describe how the language model was used** and provide the prompts (e.g., "I asked GPT to write the posterior calculation function of Bayes' Theorem with the prompt, ..., then I refactor the code.").
2. **Show how they modified the response** (e.g., highlighting changes, adding their thoughts, or expanding explanations).
3. **Reflect on what they learned** (e.g., "GPT helped me understand reinforcement learning, but I also researched additional sources to confirm the explanation").

If students do not clearly explain their use of language models, the problem will receive a zero score.

For most of the coding parts, I asked Chatgpt and Claude to guide me through the coding and analyze the results. Most of the prompts: "Guide me through the decision tree model from scikit-learn step by step," "Why did the gradient boosting model improve so little after tuning parameters? Is it because of the sequential process?", "How can I implement randomly downsample? What does it mean?", "here is my conclusion: check grammar mistakes and misspellings, and maintain a logical flow." "from the results, I think 'max_depth' and 'min_samples_leaf' are the most influential parameters. Does it make sense?", "for the feature selection part, should I use the model with default parameters to train or the one with best parameters?" and so on. I mostly use generative AI to check my logic and answer my concerns.

Claude and ChatGPT helped me understand the difference between these models: naive Bayes, decision trees, random forests, and gradient-boosting models. They also used simple explanations to help me understand the algorithm behind each model, such as the decision tree. It uses the 'information gain' method to choose which feature to split to minimize the entropy overall. I was confused; I first thought that as the information gain decreased, the overall entropy decreased. But it turned out it was the opposite: information gain measures how much entropy decreases after splitting. So, as the information increases, the entropy decreases. Claude and ChatGPT also helped me a lot in coding, such as the grid search method for tuning parameters. I know the technique, but I'm unsure which parameters are more crucial to be tuned since I can't tune all the parameters. So I asked Claude, and it gave me the most tuned parameters due to their effect on the model. For example, for the trees model, the depth, the number of trees, and the leaves of the trees are critical since they are related to the overfitting problem. That is the reason I selected to modify those parameters.

妳的理解大致是對的，但我們可以再稍微釐清一下細節來幫助妳更精準掌握 decision tree 的原理：

🌳 Decision Tree 運作邏輯簡述：

1. **核心目標：**
   希望每次分裂（split）後，**讓分類結果越「純」越好**，也就是子節點的 entropy 越低。
2. **Entropy 是什麼？**
   - 一種衡量「雜亂程度」的指標。
   - 如果一個節點中所有資料都是同一類（例如全是貓），那 entropy = 0。
   - 如果一個節點中有 50% 貓、50% 狗，這就是最雜亂的情況，entropy 最大。
3. **Information Gain 是什麼？**
   - 衡量「使用某個 feature 來分裂」後，entropy **下降了多少**。
   - 公式：

$$\text{Information Gain} = \text{Entropy}(\text{原始}) - \text{加權後的子節點 entropy}$$

   - 所以：**Information Gain 越大，代表這個 feature 分得越好！**

Overall, generative AI helped me understand the algorithm behind each model and provided me with simple explanations. They also assisted me in writing the code, with explanations for each line. Now, I have a basic understanding of naive Bayes, decision trees, random forests, and gradient boosting models, as well as the whole training, validation, and testing process in machine learning. I also know how to import useful packages from scikit-learn, and I decided to explore more.