Ciências ULisboa | Informática

Advanced Machine Learning 2025/2026

PROJECT: Medical Diagnosis and Treatment Recommendation

1. INTRODUCTION

Healthcare is full of uncertainty. Patients present with noisy and incomplete information, diseases evolve over time in unpredictable ways, and doctors must make treatment decisions without knowing exactly how a patient will respond. Designing intelligent systems that can reason under uncertainty is, therefore, one of the most important challenges in modern AI.

In this project, you will step into the role of a data scientist working with a cardiology clinic that is following patients at risk for heart disease. Over multiple clinical visits, patients report symptoms, undergo lab tests, and sometimes receive treatments. Your task is to build models that can:

- **Diagnose** patients' hidden health states from noisy observations.
- **Model disease progression** across multiple visits.
- **Recommend treatments** that improve long-term outcomes.

The project is divided into milestones, each introducing a new modeling tool. By completing them, you will gradually assemble a full pipeline: from data exploration to probabilistic modeling, to sequential decision-making:

- **M0:** Exploratory Data Analysis (**Deadline: 30/September**)
- **M1:** Mixture Models (**Deadline: 10/October**)
- **M2:** Bayesian Networks (**Deadline: 20/October**)
- **M3:** Inference in Bayesian Networks (**Deadline: 7/November**)
- **M4:** Learning Bayesian Networks from Data (**Deadline: 21/November**)
- **M5:** Hidden Markov Models (**Deadline: 1/December**)
- **M6:** Reinforcement Learning (**Deadline: 12/December**)

By the end of this project, you will have built an end-to-end pipeline for medical decision support: starting with exploratory data analysis, uncovering hidden structure with mixture models, capturing probabilistic dependencies with Bayesian Networks, reasoning with inference, learning from data, modeling temporal dynamics with Hidden Markov Models, and finally recommending treatments with Reinforcement Learning. While the dataset is synthetic and the models simplified, the same principles underlie real clinical AI systems, where larger datasets, continuous variables, and deep learning are used to scale these ideas. The goal is not only to apply a range of machine learning techniques, but to see how they connect into a coherent story: turning data into knowledge, and knowledge into action under uncertainty.

2. Submission Guidelines

Each group should submit the report for each milestone written in Jupyter Notebook with the name **MX_GYY.ipynb**, where **X** is the number of the milestone and is the number of the milestone and **YY** is the group number registered in Moodle. The notebook should include (but not be limited to):

- The identification of the members of the group (number and name of each element)
- All the tasks described in the milestone
- The **code and corresponding outputs** obtained
- Explanations and justifications for your decisions
- Discussion of the results obtained

Beware that **I will not run the code of all groups, but I might run some randomly selected groups, so I need** explicit outputs in the report to confirm your conclusions.

Each milestone has its own deadline, and the submission of the notebook should be made by the date at **23:59 in Moodle**. The link to submit each milestone is located in the corresponding week of the deadline.

3. Project Goals

The overarching goal of this project is to give you hands-on experience with **probabilistic modeling, inference, and reinforcement learning** in the context of a realistic medical decision-making problem. By the end of the project, you should not only understand the algorithms covered in class but also appreciate how they can be applied, connected, and compared in practice.

Concretely, this project will help you:

- **Explore and understand data:** Learn how to perform exploratory analysis on a longitudinal medical dataset, identifying risk factors, disease states, and treatment patterns.
- **Discover hidden structure:** Use mixture models to cluster patient encounters and uncover latent groupings that may correspond to different disease presentations.
- **Model dependencies:** Design and learn Bayesian networks that capture the probabilistic relationships between patient features, disease states, and symptoms.
- **Reason under uncertainty:** Apply exact and approximate inference methods to answer clinically relevant queries, such as estimating the probability of disease given observed symptoms.
- **Learn from data:** Use parameter estimation and structure learning algorithms to automatically recover Bayesian networks and compare them to your own designs.
- **Capture temporal dynamics:** Fit Hidden Markov Models to patient trajectories, learning how diseases evolve across visits and how symptoms reflect hidden states.
- **Make decisions:** Apply reinforcement learning to design treatment policies that balance immediate benefits and long-term patient outcomes.

4. Dataset

4.1. Clinical Scenario

Imagine a cardiology clinic conducting a longitudinal study of patients at risk for cardiovascular disease. At baseline, the clinic recruits a diverse cohort of adults with varying health profiles: some are young and otherwise healthy, others carry well-known risk factors such as obesity, smoking, or family history of heart disease.

Over the course of several years, each patient is followed at regular intervals during scheduled clinical visits. At each visit, the medical team records the patient's current symptoms, performs standard laboratory tests, and prescribes treatments when necessary. The clinical aim is to monitor how underlying cardiovascular disease may progress over time, to detect deterioration at an early stage, and to evaluate how different treatments affect outcomes. This creates a **longitudinal dataset** that captures both the *static risk profile* of each patient and the *dynamic evolution* of their health status over time.

Hence, the data you are provided simulates this clinical study. Each patient has a fixed demographic and risk profile, but their disease state may evolve across visits. Symptoms and laboratory measures provide observable signals of disease activity, while treatments and outcomes reflect medical decision-making and response. Although the dataset is synthetic, it has been carefully designed to mirror the uncertainty, variability, and causal dependencies seen in real-world healthcare settings.

By analyzing this dataset, you will step into the role of a data scientist working with a clinical research team: your task is to model disease progression, infer hidden states, and evaluate treatment strategies, using probabilistic and machine learning methods.

4.2. Synthetic Medical Diagnosis and Treatment Dataset

This dataset has been constructed to simulate the type of information that might be collected in a real-world clinical study of patients at risk of cardiovascular disease. Although all data here are synthetic, they are designed to reflect plausible medical relationships between patient characteristics, disease progression, symptoms, laboratory tests, treatments, and health outcomes. The dataset allows you to explore probabilistic modeling, inference, and learning tasks in a controlled environment without relying on sensitive patient records.

The dataset consists of two main files: `patients.csv` and `encounters.csv`.

`patients.csv` contains **static patient characteristics** recorded at baseline (time of enrollment). Each patient is followed longitudinally over multiple clinical encounters, but their demographic and risk profile remain fixed across time. The available features are:

- `patient_id`: unique identifier for each patient.
- `age`: the patient's age in years at baseline.
- `sex`: biological sex of the patient (male/female).
- `bmi`: body mass index, a measure of body fat calculated from weight and height.
- `smoker`: binary indicator of whether the patient is a current smoker.

- `family_history`: binary indicator for the presence of cardiovascular disease in immediate family members.
- `hypertension`: binary indicator for whether the patient has a previous diagnosis of high blood pressure.
- `risk_score`: a synthetic index summarizing the combined effect of the risk factors above. Higher scores indicate a greater predisposition to developing cardiovascular disease.

`encounters.csv` contains **time-varying information** collected at each follow-up visit. Each patient has a sequence of follow-up visits (time steps), which allows the study of temporal progression. Each record corresponds to a patient-visit pair, and the available features are:

- `patient_id`: patient identifier linking to the static patient profile.
- `time`: the follow-up time point of the encounter (from 0 to 7).
- `state`: the underlying health state of the patient at this time point. Possible values are `Healthy`, `Early` disease, and `Advanced` disease. Although unobserved in real life, here it is provided as a latent ground truth to support teaching in hidden variable models.
- `chest_pain`, `fatigue`, `shortness_of_breath`: binary indicators of common cardiovascular symptoms, which may arise as the disease state worsens.
- `systolic_bp`, `cholesterol`, `glucose`, `troponin`: laboratory test results. These are continuous-valued and reflect common clinical measures associated with cardiovascular health. Their distributions shift as the disease state progresses.
- `treatment`: therapy administered at the time of the encounter, which includes `None` (no treatment), `DrugA`, `DrugB`, or `Lifestyle` recommendation (non-pharmacologic: diet/exercise/smoking cessation).
- `utility`: a numerical reward-like score that encodes the patient's overall quality of life given the state and treatment. Higher values correspond to more favorable clinical outcomes at that visit (e.g., patient stable, treatment effective), and lower values correspond to worse outcomes (e.g., advanced disease, poor symptom control).

This dataset is intended to mimic the type of decision-making environment faced in clinical practice, where physicians must integrate patient history, risk factors, current symptoms, and lab results to decide on appropriate treatment strategies. The probabilistic structure embedded in the data allows for experimentation with machine learning methods such as clustering, Bayesian networks, hidden Markov models, and reinforcement learning, while still remaining simple enough for educational purposes.

WHY DOES THE DATASET INCLUDE `UTILITY`?

In this dataset, each encounter also contains a variable called `utility`, which is intended to summarize the patient's progress after the visit. Utility is not something a doctor measures directly in real life; no machine outputs a “utility score”. Instead, utility is a **synthetic proxy** for patient outcomes (e.g., quality of life, survival). It allows us to model treatments as affecting long-term health, and later serves as the reward signal in reinforcement learning.

4.3. Limitations of the Dataset

While this dataset is designed to resemble a real-world cardiology cohort, it is important to recognize its limitations:

- **Synthetic generation:** All patient records are simulated. This means that while the dataset reflects plausible medical patterns (e.g., chest pain being more common in advanced disease), it does not capture the full biological and social complexity of real populations.
- **Simplified disease progression:** Health states (`Healthy`, `Early`, `Advanced`) are modeled as a small number of discrete categories. In reality, cardiovascular disease develops along a continuum with many intermediate conditions.
- **Restricted treatment options:** Only a few treatments are available (`None`, `DrugA`, `DrugB`, `Lifestyle`). Real clinicians have a much richer set of interventions (medications, procedures, behavioral counseling) and often combine them.
- **Utility as a proxy:** The `utility` variable condenses multiple aspects of patient well-being into a single number. In practice, outcomes are multidimensional and difficult to quantify so neatly.
- **No external validity:** The dataset should not be used to draw medical conclusions about real patients or treatments. It is strictly a teaching tool to illustrate methods for probabilistic reasoning, inference, and decision-making.

Despite these limitations, the dataset is rich enough to support the main goals of this project: exploring how machine learning and probabilistic models can help reason under uncertainty, model disease progression, and optimize treatment strategies in a controlled, safe environment.

4.4. Load the Dataset

You should use `pandas` to read each one of the CSV files containing the dataset. In the encounters file, if the entries with `None` in the variable `treatment` appear as `NaN` (missing value), you should do the following:

```python
import pandas as pd

patients = pd.read_csv('patients.csv', sep=';')
encounters = pd.read_csv('encounters.csv', sep=';')

# treatment "None" is not a missing value
encounters['treatment'] = encounters['treatment'].fillna('None')
```

5. M0: Exploratory Data Analysis (Deadline: 30/September 23:59)

Before we can build probabilistic models, it is important to familiarize ourselves with the dataset and the type of information it contains. In this milestone, your task is to conduct a brief exploratory data analysis (EDA) to gain a deeper understanding of the data structure, detect potential issues such as missing values, and formulate hypotheses about the relationships between patient characteristics, disease states, symptoms, and treatments.

You will work with the two provided files:

- `patients.csv` (static baseline features: demographics and risk factors)
- `encounters.csv` (longitudinal follow-up data for each patient, including state, symptoms, labs, treatments, and outcomes)

For this milestone, treat each encounter (row in `encounters.csv`) as an independent observation when analyzing disease states, symptoms, labs, treatments, and utility. Use `patients.csv` only for static baseline summaries (age, sex, risk factors).

Your analysis should include both summary statistics and visualizations. **You do not need to be exhaustive**; the goal is to uncover patterns that will motivate the probabilistic models in later milestones.

The focus here is **interpretation**: try to reason about what the patterns you observe might mean in a real medical context, even if the data is synthetic. This will help you later when building Bayesian networks and hidden Markov models.

Concretely, you should address the following 6 tasks:

5.1. Dataset overview
- How many patients are included? How many encounters in total?
- How many encounters per patient on average?

5.2. Baseline characteristics
- What is the distribution of age, sex, BMI, smoking status and family history?
- How does the computed risk score distribute across the population?

5.3. Disease states
- What proportion of encounters fall into each state?
- Do some patients remain stable while others progress? Provide a few illustrative examples.

5.4. Symptoms and laboratory values
- For binary symptoms (chest pain, fatigue, shortness of breath), what are the overall frequencies?
- For lab values (blood pressure, cholesterol, glucose, troponin), what are the means, medians, and ranges?
- Do these variables appear to differ by disease state?

5.5. Treatments and outcomes
- What is the distribution of treatments across encounters? How do treatments vary across states?
- What is the distribution of the utility values across encounters? Do utilities differ systematically by state or treatment?

5.6. Data quality check
- Are there any missing values? If so, where do they appear, and how frequently are they?
- Suggest one or two simple strategies to handle missing values. Later (in milestone M1), you will need to implement one strategy to handle missing values.

6. M1: MIXTURE MODELS (DEADLINE: 10/OCTOBER 23:59)

In this milestone, you will use **mixture models** to uncover hidden structure in the population of patient encounters. **Each encounter (row in `encounters.csv`) should be treated as an independent observation**, since symptoms and lab values can change across visits even for the same patient.

Here, you will use **a broad set of observed variables (symptoms and lab values)** to cluster encounters, since the goal is to explore the overall structure of the data and see whether latent groups correspond to different disease states. Later milestones will focus on other aspects of the problem — for example, in M5 we will restrict to fewer variables to capture **temporal dynamics** using HMMs.

By fitting a **Gaussian Mixture Model (GMM)**, we can explore whether such latent groups capture patterns similar to the disease states or reveal new insights. The goal of this milestone is not to achieve perfect alignment between clusters and disease states, but to practice fitting mixture models and interpreting latent group structure in medical data. The clusters may or may not match the labeled states; this is part of the analysis.

Concretely, you should address the following 4 tasks:

6.1. Feature selection and preprocessing

- Choose the features from `encounters.csv` to use for clustering. Remember that each encounter (row in the file) should be treated as an independent observation.
- Standardize or normalize continuous variables so that features with different scales are comparable.
- Implement and apply the imputation method that you described in milestone M0.

6.2. Model fitting

- Fit a **Gaussian mixture model (GMM)** with a different number of clusters (e.g., 2-8). **Note:** You can treat binary features as numeric 0/1 and include them in the GMM (if desired), even though this is an approximation. More advanced models can handle mixed data types explicitly, but are beyond our scope.
- Use information criteria (AIC or BIC) to select a reasonable number of clusters.
- Assign each encounter to its most likely cluster.

6.3. Cluster characterization

- Analyze the distribution of symptoms and lab features within each cluster.
- Compare these cluster profiles to the true disease states (Healthy, Early, Advanced). Do the clusters align with the states, or do they capture different groupings?

6.4. Interpretation

- What clinical interpretation can you give to each cluster? For example, do some clusters represent “mostly healthy” patients with normal labs”?
- Do treatments or utility values vary across clusters? Provide a simple table or plot to illustrate.

7. M2: BAYESIAN NETWORKS (DEADLINE: 20/OCTOBER 23:59)

In the previous milestone, you used mixture models to explore hidden groupings of patient encounters based on symptoms and lab values. This revealed a possible structure in the data, but it did not explicitly model how different variables influence each other.

In this milestone, you will design a **Bayesian network (BN)** to represent probabilistic dependencies among patient characteristics, hidden disease states, observable features, and outcomes. Unlike clustering, a BN makes assumptions about *causal direction* (e.g., state → symptoms) and allows you to reason about conditional independencies.

At a minimum, your BN should include `risk_score`, `state`, `treatment`, and `utility`, along with at least a subset of observed variables (symptoms and/or lab values). This ensures that your BN can represent how risk factors influence disease, how disease generates evidence, and how treatments affect outcomes.

The goal here is not to build a perfect medical BN, but to practice **formalizing modeling assumptions** and to see how probabilistic dependencies emerge from a structured graphical representation. In later milestones, you will refine these models with inference and parameter learning.

Concretely, you should address the following 4 tasks:

7.1. Structure design

- Propose a BN structure that connects baseline risk to disease state, and disease state to observation variables. At a minimum, your BN should include `risk_score`, `state`, `treatment`, and `utility`. You should also add at least a subset of the observed variables (e.g., a few symptoms and/or lab values) so the BN can represent how the state generates observable evidence.
- You are encouraged to justify why each edge makes sense clinically. For example, the disease state influences symptoms and lab values; a higher risk increases the likelihood of progressing from Healthy to Diseased.
- Implement the network in `pgmpy`.

7.2. Discretization

- Continuous variables cannot be used directly in simple BNs. Select appropriate thresholds to discretize them into categories. **Note:** You should work with the imputed dataset, but not the normalized one.
- Document your discretization rules and explain why they are reasonable.

7.3. Estimating CPTs by counting

- Use the dataset to approximate CPTs by **relative frequencies** (counting occurrences). **Note 1:** Treat each encounter (row in `encounters.csv`) as an independent observation, so that probabilities reflect visit-level behavior rather than patient-level summaries. **Note 2:** Use the full dataset to design your BN and estimate CPTs by counts. At this stage, we are more interested in structure than predictive accuracy.
- For variables with multiple parents, compute counts conditioned on all parent values.
- Draw the BN graph (e.g., `pgmpy` plotting utilities or `network`).

- Show at least one CPT as a table.

7.4. Analysis of independencies

- Use `pgmpy`'s `get_independencies()` function to identify conditional independencies implied by your BN.
- Select a few that you find clinically interesting or surprising, and explain whether they seem reasonable in the context of the dataset. **Example** (you don't need to use these exact variables in your BN): "Chest pain is independent of cholesterol given disease state." Does that make sense? (Yes: both are caused by a disease state, not directly by each other.)

8. M3: Inference in Bayesian Networks (Deadline: 7/November 23:59)

In the previous milestone, you designed a Bayesian Network (BN) to capture dependencies between risk factors, disease states, symptoms, treatments, and outcomes. Defining the structure and estimating CPTs gave you a static model of how variables are related.

In this milestone, you will take the next step: **reasoning with the BN by performing inference**. The goal is to use your model to answer clinically meaningful questions, for example, estimating the probability of disease given symptoms, predicting which treatment is likely to be prescribed, or evaluating the chances of a good outcome under a specific therapy.

You will begin with **exact inference** (Variable Elimination and Belief Propagation), then move to **approximate inference** using sampling. This will allow you to compare methods in terms of accuracy and computational efficiency, and to reflect on how probabilistic reasoning supports clinical decision-making under uncertainty.

You will work primarily with the **Bayesian network built in M**2 and with the **imputed and discretized dataset from M2** (but not the normalized one).

Concretely, you should address the following 3 tasks:

8.1. Exact inference

- Using `pgmpy`, compute posterior probabilities for variables of interest.
  a) $P(state = Healthy | chest\_pain = 1, fatigue = 1)$ – Given a patient with chest pain and fatigue, what is the probability that the patient is Healthy?
  b) $P(treatment | chest\_pain = 1, fatigue = 1)$ – What treatment is most likely to be prescribed for patients with chest pain and fatigue?
  c) $P(utility = high | risk\_score = 4, treatment = DrugA)$ – If we treat a patient with DrugA, what are their chances of a good outcome?

  Note: If your BN structure differs, you may adapt the queries to your chosen variables, but make sure to include at least one query about diagnosis (a), one about treatment (q), and one about outcomes (c).
- Use **Variable Elimination** and **Belief Propagation** for exact computation.
- For each query, run both Variable Elimination and Belief Propagation. The posterior distributions should be nearly identical. If they differ, discuss possible reasons.

8.2. Approximate inference

- Implement **one sampling-based approach** for the same queries.
- For each query, compare the posterior probability distribution obtained from exact inference with the distribution obtained from your sampling-based method. How close are the results? How does the accuracy change as you increase the number of samples? Discuss whether the sampling method is computationally efficient compared to exact inference. **Note:** For query b), choose one value of `treatment` to compare the posterior distribution obtained from exact inference and your sampling method.

8.3. Query design

- Pose 2 or 3 queries of your own interest and compute the posterior distributions using both exact and approximate methods. **Example:** "Given a patient with high cholesterol and shortness of breath, what is the probability of having Advanced disease?"

9. M4: Learning Bayesian Networks from Data (Deadline: 21/November 23:59)

In the previous milestone, you used your Bayesian Network (BN) to perform inference and answer clinical queries. That exercise showed how a hand-designed model can be used for reasoning under uncertainty.

In this milestone, you will explore an alternative approach: **learning Bayesian Networks directly from data.** Instead of relying only on expert knowledge to define the structure and parameters, you will let algorithms estimate them from the dataset. This will allow you to compare:

- How well a **human-designed BN** (from M2) captures clinical reasoning, versus
- How well a **data-driven BN** (learned automatically) fits the observed encounters.

You will start by learning parameters for your hand-designed BN, then move to **structure learning** with score-based search. Finally, you will evaluate and compare the models on held-out patients, reflecting on the trade-off between **interpretability** (expert-designed networks) and **fit to data** (learned networks).

You will work primarily with the **Bayesian network built in M**2 and with the **imputed and discretized dataset from M2** (but not the normalized one).

Concretely, you should address the following 4 tasks:

9.1. Parameter learning

- Start by splitting your data into train and test data. You need to split your data at the patient-level:
  - Assign 70% of patients to the training set and 30% to the test set.
  - Use "all encounters" from each patient consistently in either the train or test. This prevents information leakage (a patient's baseline risk factors in training and their encounters in testing). This is more realistic because models must generalize to new patients.
- Start with the BN structure designed in M2.
- Use a parameter estimator of your choice (e.g., maximum likelihood estimation, Bayesian estimation) to learn the CPTs from the encounter-level training data.

- Compare the learned CPTs to the relative-frequency CPTs you computed in M2. Do the probabilities look similar? Are there differences where the data was sparse?

9.2. Structure learning

- Use a **score-based search algorithm** to learn a BN structure directly from the training data. **Note:** Use the same subset of variables as in your hand-designed BN, so you can compare the learned structure fairly.
- Visualize the learned network and compare it to your designed structure:
  - Which edges are the same?
  - Which edges are missing or reversed?
  - Did the algorithm discover dependencies you had not included?

9.3. Model comparison

- Evaluate both your designed BN and the learned BN on the test set of encounters. Use log-likelihood, BIC, or other Bayesian score as a scoring function. You may compute multiple scores, but choose one to guide your final model comparison
- Which network explains the data better?
- Discuss the trade-off between interpretability (designed BN) and fit to data (learned BN).

9.4. Reflection

- In practice, we could reduce variance in evaluation by using cross-validation. What would be the advantages and challenges of doing that in the context of Bayesian networks?
- In medicine, we rarely rely only on data-driven structure learning, because learned edges may reflect statistical associations rather than true causal links. Write a short discussion of how you would combine expert knowledge with data-driven methods in practice.

10. M5: Hidden Markov Models (Deadline: 1/December 23:59)

So far, we have modeled each encounter as an independent observation. In reality, however, patients are followed over time, and their underlying health condition may evolve gradually across visits. To capture this temporal dependence, we will use **Hidden Markov Models (HMMs)**.

In an HMM, we assume that:

- Each patient has a **hidden disease state** that evolves over time according to Markov dynamics (e.g., transitions between Healthy → Early → Advanced).
- At each visit, we observe certain **symptoms or lab measurements** that depend probabilistically on the current hidden state.

In principle, HMMs could include many observable variables. However, for this milestone, we restrict to just troponin and one additional symptom. This keeps the model computationally manageable and easier to interpret, while still capturing meaningful temporal dynamics of disease progression. In real applications, larger HMMs or more advanced temporal models could incorporate richer sets of features.

You will work with the **train and test split from M4**, which was obtained from the **imputed and discretized dataset from M2** (but not the normalized one). The discretization of continuous variables will make HMM fitting easier.

his milestone introduces temporal models of disease progression, showing how to perform inference in sequential data and learn model parameters from patient trajectories.

10.1. Model setup

- Treat the **true state variable** in `encounters.csv` as hidden (ignore it during training).
- Select **two observable variables**:
  - One must be `troponin` (already discretized in your dataset).
  - The second should be an **informative feature** that reflects disease progression (e.g., a symptom such as `shortness_of_breath`). Avoid features with little variation, since they won't help the HMM learn meaningful emissions.
- Because `hmmlearn`'s `CategoricalHMM` requires a **single categorical observation per time** step, you must encode your two features into a single symbol. **Example:** if troponin was previously discretized into 0/1, then you should

  `(troponin=0, shortness_of_breath=0)` → `0`,
  `(troponin=0, shortness_of_breath=1)` → `1`,
  `(troponin=1, shortness_of_breath=0)` → `2`,
  `(troponin=1, shortness_of_breath=1)` → `3`.

  After this encoding, each patient's sequence is just a series of integers in `{0,1,2,3}`, one symbol per encounter.
- Organize your data so that each patient corresponds to one sequence of encoded symbols (sorted by time), and fit the HMM using `CategoricalHMM` in `hmmlearn`.

10.2. Parameter learning

- Use the **Baum–Welch algorithm** (implemented in `hmmlearn`) to estimate:
  - Transition probabilities between hidden states.
  - Emission probabilities for the observable symbols.
- Fit an HMM with **3 hidden states** (to mirror Healthy, Early, and Advanced). Run the HMM multiple times with different initializations and select the best model using the training data.
- **Important:** The learned hidden states are *not labeled*; the algorithm might call them "State 0, State 1, State 2" in any order. To interpret them, you must inspect the emission probabilities. **Example:**
  - Compute $P(troponin = 1|state = k)$ and $P(shortness\_of\_breath = 1|state = k)$ for each hidden state k.
  - The state with the highest troponin probability and symptom probability is most likely Advanced, the state with the lowest troponin and symptom probability is Healthy, and the remaining one is Early.
- After mapping states, examine the **learned transition matrix**. Does it reflect plausible disease progression (e.g., more likely to move from Healthy → Early than Early → Healthy; do Advanced patients tend to stay in Advanced, with only rare improvement)?
- Inspect the **emission probabilities**. Do they capture meaningful associations (e.g., high troponin and shortness of breath are more likely in the Advanced state)?

- Finally, discuss: why **shouldn't we select the best model only based on training likelihood**? What alternative evaluation strategies could we use?

10.3. Inference

- Apply the **Viterbi algorithm** to decode the most likely sequence of hidden states for each patient.
- Compare the decoded states to the **true** `state` **column** (kept hidden during training).
- Evaluate performance:
  - Compute a confusion matrix to see where the model makes errors.
  - Plot decoded vs true state sequences for 1–2 patients to visualize disease trajectories.
- Reflect on the errors:
  - Which states are most often confused (e.g., Early vs Advanced)?
  - What does this tell us about the dataset or the model's limitations?

11. M6: Reinforcement Learning (Deadline: 12/December 23:59)

Up to this point, you have built models to describe and predict patient health:

- Mixture models revealed hidden structure in encounters.
- Bayesian Networks captured probabilistic dependencies.
- Hidden Markov Models described disease progression over time.

In practice, however, clinicians are not only interested in understanding disease; they must also **decide how to treat patients** at each visit. This brings us to the final step of the project: **Reinforcement Learning (RL)**.

In this milestone, you will view treatment as a sequential decision-making problem:

- The **environment** is the patient trajectory.
- The **state** is the health status of the patient (hidden in reality, but available in our synthetic dataset).
- The **actions** are the possible treatments (`None`, `DrugA`, `DrugB`, `Lifestyle`).
- The **reward** can be derived from the `utility` score, which summarizes patient outcomes.

In this milestone, you will implement **tabular Q-learning** to learn treatment policies that maximize long-term patient outcomes. You will compare the learned policy to simple baselines (random and heuristic) and reflect on whether the RL agent's choices align with clinical intuition or exploit artifacts of the synthetic data.

This final milestone highlights how probabilistic modeling and sequential decision-making connect in a real-world inspired setting: building not just predictive models, but systems that can **recommend actions under uncertainty**. In real-world settings, RL is applied with much larger state/action spaces, often requiring function approximation (e.g., deep RL). Our toy environment will illustrate the principles in a controlled way.

11.1. Environment setup

- Use the **true** `state` from encounters.csv as the environment state. Map it to integers: `{Healthy=0, Early=1, Advanced=2}`.
- Map treatments: `{None=0, DrugA=1, DrugB=2, Lifestyle=3}`.

- Define the reward at each step from the `utility` column. You may scale utility values to `[0,1]` using min-max normalization for stability. This avoids very large or negative rewards dominating the Q-learning updates.
- Construct the `transition` matrix by building the next-state and next utility. The matrix should contain $(s_t, a_t, s'_{t+1}, utility_{t+1})$. **Note:** Keep only rows with a valid transition to the next visit.
- Build empirical **transition dynamics** (`trans_mat`: matrix of shape $(n\_states, n\_actions, n\_states)$) from the dataset

$$\hat{P}(s'|s,a) = \frac{\#\{(s_t = s, a_t = a, s_{t+1} = s'\}}{\#\{(s_t = s, a_t = a)\}}$$

  Note: Use Laplace smoothing to prevent zero probabilities.
- Build a table `r_sa` of shape $(n\_states, n\_actions)$ where each entry is the **average reward you get when taking action a in state s**. **Note:** Compute the global mean across all transitions; this will serve as a fallback if some (s,a) never occurs in the dataset (sparse problem).
- Estimate the initial state distribution π (matrix of shape $(n\_states,)$) from the first visits in the data.
- Generate the environment from the following code. This is an environment that will allow us to sample the next state s' from $\hat{P}(s'|s,a)$.

```
# environment simulator for Q-learning
rng = np.random.default_rng(0)

def sample_next_state(s, a):
    return rng.choice(n_states, p=trans_mat[s, a])

def reward_of(s, a):
    return r_sa[s, a]

class PatientEnv:
    def __init__(self, pi, trans_mat, r_sa, horizon=6, rng=None):
        self.pi = pi
        self.P = trans_mat
        self.R = r_sa
        self.horizon = horizon
        self.rng = np.random.default_rng() if rng is None else rng
        self.reset()

    def reset(self):
        self.t = 0
        self.s = self.rng.choice(n_states, p=self.pi)
        return self.s

    def step(self, a):
        r = float(self.R[self.s, a])
        s_next = self.rng.choice(n_states, p=self.P[self.s, a])
        self.s = s_next
        self.t += 1
        done = (self.t >= self.horizon)
        return s_next, r, done, {}

env = PatientEnv(pi, trans_mat, r_sa, horizon=7, rng=rng)
```

11.2. Tabular Q-learning

- Implement the Q-learning algorithm for the previous environment.

- Train an agent with ϵ-greedy to select treatments that maximize cumulative reward.
- Compare learned policy to a baseline. Use the following function to evaluate any policy. This function returns the rewards after 1000 episodes; you should then average those rewards.
  - Random policy: choose any action uniformly.
  - Heuristic policy: rule-based, of your choice, you can invent any reasonable mapping. **Example:** Healthy → None or Lifestyle; Early → DrugA; Advanced → DrugB

```
# --- Generic simulator for any policy ---
def evaluate_policy(env, policy_fn, Q=None, episodes=1000, seed=0):
    rng = np.random.default_rng(seed)
    returns = []
    for _ in range(episodes):
        s = env.reset()
        G = 0.0
        for t in range(env.horizon):
            a = policy_fn(s, Q=Q, rng=rng)
            s, r, done, _ = env.step(a)
            G += r
            if done: break
        returns.append(G)
    return np.array(returns)
```

11.3. Policy evaluation

- Plot the average cumulative reward over training episodes.
- Inspect the learned Q-table: which treatments does the agent prefer for each state?
- Does the learned policy align with clinical intuition, or is it an artifact of the synthetic data and how rewards/transitions were estimated?