Lab 4

Due: Friday 11/14/2025 @ 11:59pm EST

The purpose of labs is to practice the concepts that we learn in class. To that end you will be writing java code that uses a game engine called Sepia to develop agents that solve specific problems. In this lab we will be playing a zombie survival game called Zombayes, where we have to classify between human units (pretending to be zombies so they don't get eaten) and actual zombies. Zombies will attack you given the chance, but humans will not. Your job will be to use a Naive Bayes classifier to learn how to distinguish between zombies and humans: you want to attack zombies and not humans!

1. Copy Files

Please, copy the files from the downloaded lab directory to your cs440 directory. You can just drag and drop them in your file explorer.

- Copy `Downloads/lab4/lib/zombayes.jar` to `cs440/lib/zombayes.jar`.
  This file is the custom jarfile that I created for you.
- Copy `Downloads/lab4/data/labs/zombayes` to `cs440/data/labs/zombayes`.
  This directory contains a game configuration and map files.
- Copy `Downloads/lab4/src/labs` to `cs440/src/labs`.
  This directory contains our source code `.java` files.
- Copy `Downloads/lab4/zombayes.srcs` to `cs440/zombayes.srcs`.
  This file contains the paths to the `.java` files we are working with in this lab. Just like last lab, files like these are used to speed up the compilation process by preventing you from listing all source files you want to compile manually.
- Copy `Downloads/lab4/doc/labs` to `cs440/doc/labs`. This is the documentation generated from `zombayes.jar` and will be extremely useful in this assignment. After copying, if you double-click on `cs440/doc/labs/zombayes/index.html`, the documentation should open in your browser.

2. Test run

If your setup is correct, you should be able to compile and execute the given template code. You should see the Sepia window appear.

```
# Mac, Linux. Run from the cs440 directory.
javac -cp "./lib/*:." @zombayes.srcs
java -cp "./lib/*:." edu.cwru.sepia.Main2 data/labs/zombayes/easy/tunnel.xml

# Windows. Run from the cs440 directory.
javac -cp "./lib/*;." @zombayes.srcs
java -cp "./lib/*;." edu.cwru.sepia.Main2 data/labs/zombayes/easy/tunnel.xml
```

3. Survival Rules & Information

In this game, you and a partner are trying to collect some gold. However, the gold is guarded by a horde of zombies. Sprinkled amongst the zombies are humans whoe are pretending to be zombies (so they don't get eaten). There is also a special zombie unit who will continuously spawn new units whenever another unit is killed (this special zombie can spawn other zombies and also other humans, dont ask why). If you or your partner start collecting gold, the horde will notice and attack! True zombies will try to attack you while humans pretending to be zombies will take the opportunity to pass harmlessly by and escape. Since the horde is infinite, you will eventually lose the game. The purpose of this assignment is not for you to win but to see how good of a classifier you can make. We are interested in how many zombies you can take down with you before you are killed (and also how many humans you choose to kill as well).

The game will start off with your partner going to collect gold. Once your partner is adjacent to the gold, all units in the horde will rush them. As mentioned previously, humans will take advantage of the confusion and harmlessly pass around your partner (and you) while escaping from the horde. The zombies will seek to kill your partner (and you when it is your turn). Your partner has a lot of health but isn't very smart (or capable with their weapon), so will eventually succumb to the horde.

While your partner interacts with the horde, your unit has been paying attention and keeping a detailed record of which units attacked (and therefore are known to be zombies), and which units did not (and therefore are known to be humans). You have recorded a feature description of each unit as well as its ground truth status (human or zombie). When your partner eventually dies, you will use this record as training data to a classifier to learn how to distinguish between human and zombie units given their feature description. Unfortunately this feature description is not visible in the game rendering, so to make it easier when watching the game unfold, I have annotated human units with a "h" character and zombie units with "z" character. The "S" character is the zombie spawning unit, and there is only one of them in the game. Your unit does not get to see these annotations, they are solely a rendering trick.

Once the classifier is done training, its your turn to go get the gold! When the horde attacks you, your unit will use this classifier to predict which units are zombies (class 1) and which units are humans (class 0). Given predictions, my code will take over and kill any units classified as zombies, and will not kill any units classified as humans.

There are three difficulties in this game: `EASY`, `MEDIUM`, and `HARD`. There are two differences between difficulties: how much health your partner has (which controls the size of the training data you get) and how difficult it is to distinguish between humans and zombies. My code is responsible for generating the feature values of each unit in the horde, and I maintain a "human distribution" f and a "zombie distribution" g. When a new unit is generated, my code first decides whether that new unit is a zombie or a human, and depending on which class it is the feature values of that unit are generated by sampling from the corresponding distribution. In `EASY` difficulty, f and g are pretty far apart so it is easy to classify. In `MEDIUM` difficulty f and g are closer together so it is harder to classify, and in `HARD` f and g have significant overlap so it will be challenging to classify.

Task 1: Naive Bayes (100 points)

In this task, I want you to complete `src.labs.zombayes.models.NaiveBayes`. You will need to finish the implementation for a `NaiveBayes` type, specifically the `fit` and `predict` method. The `fit` method should train a Naive Bayes model on the provided training data, and `predict` should make a prediction using that trained model on the provided test point. I have hidden most of the Sepia-ness from you, so please focus entirely on making a good naive bayes model!

Note: I will test the correctness of your code not on how well your agent performs, but on how correct of a Naive Bayes model you have implemented. Your model should be generic enough to process different types of features (e.g. discrete and continuous)

Task 2: Extra Credit (50 points)

In this task, I want you to tune your model. There are many ways of doing this (for instance generating a ROC curve, etc.) and you are free to pick which method(s) you want. The way we tune Naive Bayes is through smoothing: there are many choices you can make regarding what kind of smoothing you do, and then even more choices on how you implement that flavor of smoothing. I want you to try some methods of smoothing and see how well they perform! Your goal here is to make the best **performing** model you can for the survival task.

The reason we need to tune our model is that when our model is wrong, the consequences of being wrong depends on **how** our model was wrong. For instance, in our survival task, if we misclassify a human as a zombie, we will kill that human (and presumably feel bad), but you will still be alive. If you misclassify a zombie as a human, you will not kill that zombie and it will stand next to you continually attacking you until you die. So to recap: you will survive misclassifying humans while you will not survive misclassifying a zombie. You might be tempted to engineer a model that predicts everything as a zombie. That is certainly a (rather grim) solution to this problem. However, I have defined the objective you want to optimize to care about human life so there are more optimal solutions than predict everything as a zombie.

I will award extra credit by evaluating your model on the `HARD` difficulty. Your implementation will earn credit proportional to the number of zombies you kill, and will lose credit for every human that you kill. For full extra credit, you must kill at least 50 zombies and not kill any humans.

Task 3: Submitting your lab

Please submit `NaiveBayes.java` on gradescope (just drag and drop the file). If you complete the extra credit, please also submit your `NaiveBayes.java` to the extra credit entry.