Project 3

Due July 18

1 Description

`house-votes-84.data` is a comma-separated values file with 17 fields and 435 records. The first is that U.S. House of Representatives member's party affiliation. The rest are that representative's vote on piece of legislation identified by the Congressional Quarterly Almanac.

```
handicapped-infants
water-project-cost-sharing
adoption-of-the-budget-resolution
physician-fee-freeze
el-salvador-aid
religious-groups-in-schools
anti-satellite-test-ban
aid-to-nicaraguan-contras
mx-missile
immigration
synfuels-corporation-cutback
education-spending
superfund-right-to-sue
crime
duty-free-exports
export-administration-act-south-africa
```

The party affiliation is either `democrat` or `republican`. From each representative, each piece of legislation received either a vote for (`y`), a vote against (`n`) or abstention (`?`).
Your task is to build a naive Bayesian classifier 'from scratch'. (In practice, it would be very unusual to have to implement a naive Bayesian classifier, much like how it would be very unusual to have to implement a sorting algorithm. But it's a good exercise to get a better understanding of how they work.) Provide the executable file `bayes` that takes two arguments:

- `./bayes train.data classify.data`

Where `train.data` is the training set (data to learn from) and `classify.data` is a set of data that the trained classifier should attempt to classify. `classify.data` will be of the same format as training data (CSV with 17 fields), except that the first field (party affiliation) is ignored.
Your `bayes` program should output the estimated *probability* that each record in `classify.data` is a `democrat` representative, with each estimated probability appearing on a separate line of standard output. For example, the following contents of a `classify.data` file, fed into your `bayes` program:

```
party,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n
party,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y
party,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
```

Should result in 3 probabilities being printed to 3 separate lines of standard output (the first being the probability that a representative who voted 'no' on every piece of legislation is a democrat, the second, the probability that a representative who voted 'yes' on every piece of legislation is a democrat, etc...)

2 Other details

- **Start early. Do not put this project off until the last minute.**
- You don't have to implement Laplace smoothing.
- You may assume that all `train.data` files are CSV with exactly 17 fields. You may *not* 'hard code' probabilities. (We reserve the right to test your code on different training data sets.)
- All code submitted must be your own. You may use standard library functions.
- Your files must be executable; if you're using a scripting language (recommended), the first line should be an appropriate shebang (`https://en.wikipedia.org/wiki/Shebang_(Unix)`), e.g.

```
#!/usr/bin/env python3
```

- The following programming languages are 'officially sanctioned'. Talk to us before using anything different (remember, we do need to be able to execute your code on our machines):
  - Java
  - Python 2 or 3 (make sure you use the correct shebang)
  - Ruby
  - OCaml (make sure `$ ./solution ...` works)
  - Bash (to 'wrap' your code files, if necessary)

3 Acknowledgment

The Congressional Voting Records Data Set was provided by the UCI Machine Learning Repository.

Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.