Lab 2: The Naïve Bayes Classifier

OBJECTIVES

The objective of this lab is to gain practical experience adapting classification code in the naïve Bayes style for a different data set and problem.

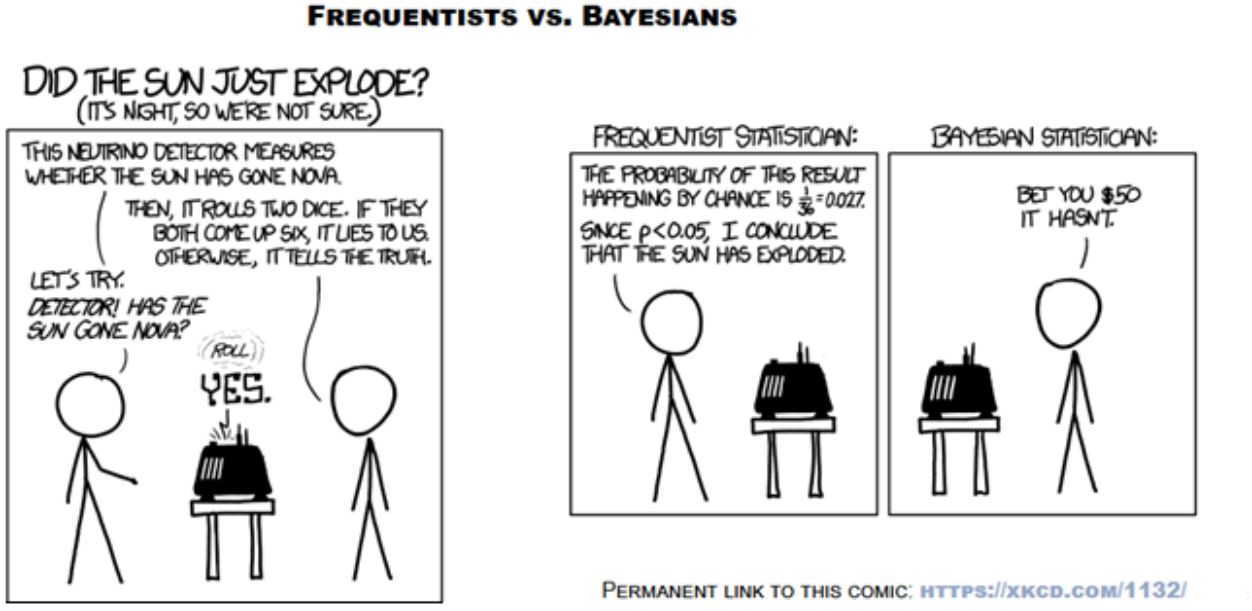
EQUIPMENT AND MATERIALS

For this lab, you will need access to a Python environment and the data files on Canvas.

THEORY (BACKGROUND)

Bayes' theorem is a direct consequence of the total-probability theorem and the definition of conditional probability (hence, the axioms of probability). The "naïve" Bayes classifier (NBC/NB) is a simpler version of the Bayes optimal classifier and it can work on small data sets. (Not everyone has access to the types of data sets that Google *et al.* can store.)

The naïve assumption is that the features that make up a vector can be considered independent of one another. We say "naïve" because we know this is hardly ever true—we would be naïve to think it's true in practical cases. Yet, even when it's not valid, this assumption simplifies computation a great deal while reducing accuracy tolerably little.



The field of *Bayesian Statistics* was developed before *frequentist Statistics*, but the latter ruled statistical practice right up to the 21st century when large datasets and fast processors like GPUs became available. NB is an initial step into the vast subset of machine learning called Bayesian learning. Bayesian statistics allows probability without repeatability. It follows from, but is much more than, Bayes' theorem (which points the way toward updating one's predictions as more data become available). Bayesian statistics was not developed by Thomas Bayes, but mostly by Laplace, Jeffreys, Pearl, and others in the statistics and machine-learning communities. Bayesian philosophy is (at least somewhat) in opposition to the frequentist philosophy of probability and statistics.

PROCEDURE

Part 0 (5 pts.)

1. Note that you must work with at least one and no more than two teammates.

2. Start a Jupyter notebook with a filename that includes all team members' initials and an introductory markdown cell that includes all team members' names along with today's date, and the title 'Naïve Bayes'.

Part 1 (25 pts.)

1. Download the files *carDataNL.csv*, *carDataNL_train.csv*, *carDataNL_test.csv*, and *motor_vehicles.csv* from Canvas. (If you want to change any formatting, you can save a file as XLSX, make the changes, and resave as a CSV file.)

   Do not move on to coding until you have shown me you've completed the following:

2. Use the training set *carDataNL_train.csv* to learn the priors and class conditionals ***by hand***. Record the probabilities you calculate in a markdown cell.

3. Classify your five test vectors ***by hand***. Show me your results and enter your results in your report (what you got for each of the test vectors and the model accuracy.).

4. In your report, include any false starts you or your teammates had and how you worked through them. If you did anything that turned out wrong, make sure to explain what it was, what the fix was, and the reasons.

   Once checked off:

5. You have a piece of sample code that implements a naïve Bayes classifier. Use it to start understanding naïve Bayes better and then modify it to build your own naïve Bayes classifiers that will work with the *motor_vehicles.csv* and *carDataNL.csv* data sets. (This could be one file or two files.)

6. Test your *carDataNL.csv*-based classifier on *carDataNL_test.csv*, checking the accuracy of your code. If your code results match your handwritten results, continue to the next step. If not, find and fix the problem.

   In your report, explain what initially went wrong. (Something usually does.) Include each team member's challenges and contributions at each stage of the process.

7. Report the classification of the five vectors with 0/1 loss and a hand-made confusion matrix.

8. Generate the same confusion matrix in Python. Are all your answers the same? (If not, explain why not and fix the faulty classifier.)

Part 3 (20 pts.)

9. Adapt your well-tested classifier to the more realistically sized dataset *motor_vehicles.csv*.

   You should read the data in, shuffle it, and set aside a quarter of it as the test set, all in pretty much the same way (with some customization) as before.

   The latter half of the code requires that you know the features, how many values they take, and the output classes for the new dataset so you can correctly modify the code.

10. If you haven't already, go back and clearly state each partner's contributions at every stage.

11. Upload your working Jupyter notebook to Canvas. Add comments if your changes to the code require additional steps for it to work.