

# CSE 4309 - Assignments - Assignment 2

## List of assignment due dates.

The assignment should be submitted via [Canvas](#). Submit a file called assignment2.zip, containing the following two files:

- answers.pdf, for your answers to the written tasks, and for the output that the programming task asks you to include. Only PDF files will be accepted. All text should be typed, and if any figures are present they should be computer-generated. Scans of handwritten answers will NOT be accepted.
- naive\_bayes.py, containing your Python code for the programming part. Your Python code should run on the versions specified in the syllabus, unless permission is obtained via e-mail from the instructor or the teaching assistant. Also submit any other files that are needed in order to document or run your code (for example, additional source code files, but NOT data files used as input).

These naming conventions are mandatory, non-adherence to these specifications can incur a penalty of up to 20 points.

Your name and UTA ID number should appear on the top line of both documents.

---

### **Task 1 (20 points, written)**

You are a meteorologist that places temperature sensors all over the world, and you set them up so that they automatically e-mail you, each day, the high temperature for that day. Unfortunately, you have forgotten whether you placed a certain sensor  $S$  in Maine or in the Sahara desert (but you are sure you placed it in one of those two places). The probability that you placed sensor  $S$  in Maine is 5%. The probability of getting a daily high temperature of 80 degrees or more is 20% in Maine and 90% in Sahara. Assume that probability of a daily high for any day is conditionally independent of the daily high for the previous day, given the location of the sensor.

**Part a:** If the first e-mail you got from sensor  $S$  indicates a daily high under 80 degrees, what is the probability that the sensor is placed in Maine?

**Part b:** If the first e-mail you got from sensor  $S$  indicates a daily high under 80 degrees, what is the probability that the second e-mail also indicates a daily high under 80 degrees?

**Part c:** What is the probability that the first three e-mails all indicate daily highs under 80 degrees?

---

### **Task 2 (10 points, written)**

Function  $P$  is a function defined on a set of samples  $S = \{A, B, C, D\}$ . We do not know the value of  $P$  for all samples, but we know that  $P(A) = 0.3$  and  $P(B) = 0.6$ . What can you say about whether  $P$  is a

valid probability function? Is  $P$  definitely a probability function, possibly a probability function, or definitely not a probability function? Justify your answer.

---

### Task 3 (10 points, written)

Function  $P$  is a function defined on the set of real numbers. We do not know the value of  $P$  for all cases, but we know that  $P(x) = 0.3$  when  $0 \leq x \leq 10$ . What can you say about whether  $P$  is a valid probability density function? Is  $P$  definitely a probability density function, possibly a probability density function, or definitely not a probability density function? Justify your answer.

---

### Task 4 (10 points, written)

This problem refers to the Bayes Classifier as applied to the problem of "boxes and fruits" that we covered in class. A description of that example is in slides 13-19 from the [Bayes Classifier presentation](#). Suppose that the training stage computed the correct probabilities:

- $p(B = r) = 0.4$
- $p(B = b) = 0.6$
- $p(F = a \mid B = r) = 0.25$
- $p(F = o \mid B = r) = 0.75$
- $p(F = a \mid B = b) = 0.75$
- $p(F = o \mid B = b) = 0.25$

Suppose that we get a test object  $x$  by following the experiment protocol from slide 13 (we randomly pick a box, with the given priors  $p(B = r) = 0.4$  and  $p(B = b) = 0.6$ , and then we randomly pick a fruit from the box). What is the probability that the classifier will give the correct output for  $x$ ? Justify your answer.

---

### Task 5 (50 points, programming)

In this task you will implement naive Bayes classifiers based on Gaussians.

#### Arguments

File [naive\\_bayes\\_main.py](#) contains incomplete code that aims to learn a naive Bayes classifier given some training data, and then evaluate this classifier on test data.

To complete that code, you must create a file called `naive_bayes.py`, where you implement the following Python function:

```
naive_bayes(training_file, test_file)
```

- The first argument, `training_file`, is a string specifying the path name of the training file, where the training data is stored. The path name can specify any file stored on the local computer.
- The second argument, `test_file`, is a string specifying the path name of the test file, where the test data is stored. The path name can specify any file stored on the local computer.

Both the training file and the test file are text files, containing data in tabular format. Each value is a number, and values are separated by white space. The  $i$ -th row and  $j$ -th column contain the value for the  $j$ -th feature of the  $i$ -th object. The only exception is the LAST column, that stores the class label for each object. **Make sure you do not use data from the last column (i.e., the class labels) as attributes (features).**

The training and test files will follow the same format as the text files in the [UCI datasets](#) directory. A description of the datasets and the file format can be found [on this link](#). For each dataset, a training file and a test file are provided. The name of each file indicates what dataset the file belongs to, and whether the file contains training or test data. Your code should also work with ANY OTHER training and test files using the same format as the files in the [UCI datasets](#) directory.

As the [description](#) states, **do NOT use data from the last column (i.e., the class labels) as features**. In these files, all columns except for the last one contain example inputs. The last column contains the class label.

The numpy library has a predefined constant `numpy.pi` for  $\pi$ . You should use this constant in your formulas, instead of hardcoding a different value for  $\pi$ .

### Training:

You should model  $P(x \mid \text{class})$  as a Gaussian SEPARATELY for each dimension of the data and (obviously) for each class. The output of the training phase should be a sequence of lines like this:

```
Class %d, attribute %d, mean = %.2f, std = %.2f
```

The output lines should be sorted in increasing order of class number. Within the same class, lines should be sorted in increasing order of attribute number. Attributes should be numbered starting from 1, not from 0.

In certain cases, it is possible that value computed for the standard deviation is equal to zero. Your code should make sure that the variance of the Gaussian is NEVER smaller than 0.0001. Since the variance is the square of the standard deviation, this means that the standard deviation should never be smaller than  $\sqrt{0.0001} = 0.01$ . Any time the value for the standard deviation is computed to be smaller than 0.01, your code should replace that value with 0.01.

In your answers.pdf document, provide the output produced by the training stage of your program when given [yeast\\_training.txt](#) as the input file.

## Classification

For each test object  $x$  and each class  $C$ , your program should compute  $P(C | x)$ , so as to identify the class  $C$  that maximizes  $P(C | x)$ . To compute  $P(C | x)$ , you can use Bayes rule, which means that you need to first compute  $P(x | C)$ ,  $p(C)$ , and  $P(X)$ . Note that  $P$  denotes probability densities, and  $p$  denotes probabilities.

$P(x | C)$  should be computed using the Naive Bayes assumption, that each dimension is independent of all other dimensions. Thus,  $p(x | C)$  is a product of as many terms as the number of dimensions of  $x$ . Each term is the output of a Gaussian, that is computed based on the parameters you learned during the training stage. For  $p(C)$ , you should use the fraction of training objects that belong to class  $C$ .  $P(x)$  can be computed using the sum rule.

For each test object you should print a line containing the following info:

- object ID. This is the line number where that object occurs in the test file. Start with 1 in numbering the objects, not with 0.
- predicted class (the result of the classification). If your classification result is a tie among two or more classes, choose one of them randomly.
- probability of the predicted class given the data.
- true class (from the last column of the test file).
- accuracy. This is defined as follows:
  - If there were no ties in your classification result, and the predicted class is correct, the accuracy is 1.
  - If there were no ties in your classification result, and the predicted class is incorrect, the accuracy is 0.
  - If there were ties in your classification result, and the correct class was one of the classes that tied for best, the accuracy is 1 divided by the number of classes that tied for best.
  - If there were ties in your classification result, and the correct class was NOT one of the classes that tied for best, the accuracy is 0.

The output of the classification phase should be a sequence of lines like this:

```
ID=%5d, predicted=%3d, probability = %.4f, true=%3d, accuracy=%4.2f\n
```

Object IDs should be numbered starting from 1, not 0. Lines should appear sorted in increasing order of object ID.

After you have printed the results for all test objects, you should print the overall classification accuracy, which is defined as the average of the classification accuracies you printed out for each test object. The classification accuracy should appear in a line following this format:

```
classification accuracy=%6.4f
```

In your answers.pdf document, provide ONLY THE LAST LINE (the line printing the classification accuracy) of the output by the test stage of your program, when given [yeast\\_training.txt](#) and [yeast\\_test.txt](#) as the training and test files.

## Example Output

Students usually ask for an example of correct output for this program. In the real world, usually you will not be able to obtain such examples. To best prepare for the real world, I strongly recommend that you try to develop and test your solution without using such an example output. Nonetheless, if you decide that you need to see such an output, file [results\\_yeast.txt](#) shows the output of my program when run on the yeast dataset.

## Grading

- 20 points: Correct estimation of Gaussians, correct mean and standard deviation estimated for each Gaussian.
- 20 points: Correct application of a naive Bayes classifier based on Gaussians for classification of test data.
- 10 points: Following the specifications in producing the required output.