# CPS363: Introduction to Bioinformatics
# Homework 4: Naive Bayes Algorithm
**(30 Points)**

**Files to turn in:** A zip file which contains a coversheet (in PDF), source code files, output files, screenshots if necessary, and a readme file explaining how to run your program.

In this assignment you will implement the Naive Bayes algorithm and evaluate its effectiveness by experimenting with the contact-lenses fitting problem.

You are provided with a text file called `contact-lenses.arff` which gives you data samples for deciding which contact lenses should be prescribed to a particular patient. Lines started with '%' are only for documentation and are not part of the dataset. The line started with @relation tells the name of the dataset. Lines started with @attribute give the name of each feature and list all its possible values. Lines after @data list all the data samples. For each sample, the last attribute corresponds to the class of the sample. For example, 'young,myope,no,normal,soft' tells that "age = young, spectacle-prescrip = myope, astigmatism = no, tear-prod-rate = normal, and the contact-lenses should be soft".

1. Write a Python script called `naivebayes.py` which trains a Naive Bayes classifier from a training file and makes predictions on an input file which contains independent data samples with the same set of features. For example, you should be able to run your script using the provided sample files (`sampletraining.arff` and `sampleinput.arff`) as below:

   *$python3 naivebayes.py sampletraining.arff sampleinput.arff predictions.txt*

   Your program should use all samples in `sampletraining.arff` as the training dataset and predict on each test sample in `sampleinput.arff`. The output will be saved in `predictions.txt` with each line corresponding to the prediction result of each sample. Your program should not be restricted to be runnable only on the provided sample files. For any training file and test file provided, your program should automatically process the files, read the number of features, figure out the set of possible values for each feature, the set of classes, and output probabilities of each sample belonging to different classes.

   For the provided sample files, each line of the output file should contain the predicted contact-lenses type (i.e., soft, hard, or none) followed by probabilities of being prescribed contact-lenses of soft, hard and none type (i.e., remember that the posterior "possibilities" calculated by Naive Classifier are not the actual probabilities. In order to get the real probabilities you need to divide each "pseudo-probability' by the sum of all "pseudo-probabilities".).

   To make sure your program work correctly, please calculate these probabilities by hand and compare them with the output of your program.

2. Leave-one-outcross-validation.
   *K*-fold cross-validation is normally used to evaluate the performance of a classifier. The dataset is divided into *k* subsets. In each round of the experiment, *k*-1 subsets are combined and used as

the training set, and the remaining subset is used as the test set. However, for small datasets leave-one-out cross-validation is normally used to avoid insufficient training. In each round of experiment, only one sample is used in the test set, and the remaining samples together as the training set. Write a Python script called `evaluate.py` which can perform leave-one-out cross-validation by calling `naivebayes.py` with various input files. In each round of the experiment, your Python script should create a test file which contains only one sample and a training file which contains all the remaining samples. Doing it *N* (*N* is the number of samples in the whole dataset) times and you will have each sample in the dataset as the test sample once. Your script should be able to compare your predicted contact-lenses type of each sample with the actual contact-lenses type and report the performances such as the confusion matrix and the overall accuracy. Report the performance in your PDF file. You should run your program as below:

```
$python3 evaluate.py contact-lenses.arff
```

Hint: You can use the `os.system(command)` to execute a command in the terminal. So here is a structure you may use to perform leave-one-out cross-validation.

Repeat the following **`n`** times:

(a) Generate a test file (i.e., `test.txt`) with one sample
(b) Generate a training file (i.e., `training.txt`) with the rest `n-1` samples
(c) Train Naive Bayes classifier on training file and make predictions on the validation file by running
  - `os.system('Python3 naivebayes.py training.txt test.txt result.txt') #need to import os module first`
  - Process predicted results from `result.txt` and compare with the actual class of each sample.

Summarize all predictions and evaluate the performance using confusion matrix, and overall accuracy. Output your confusion matrix and overall accuracy to the terminal.