## SENG474 Assignment 1. Fall 2017 (100pts)

`Student Name:` `Student Number:`
`Instructor George Tzanetakis`

| Question | SENG474 | CSC578D |
|---|---|---|
| 1 | 25 | 15 |
| 2 | 20 | 15 |
| 3 | 55 | 70 |
| Total | 100 | 100 |

# 1 Overview

The goal of this assignment is to explore decision trees and the Naive Bayes classifier. Different marking schemes will be used for undergrads (SENG474) and graduate students. Undergraduate students do not have to answer the grad questions.

All code questions use Python and the scikit-Learn library. You may install it on your own computer or alternatively use the computers in the lab. Submissions in other programming languages or environments are welcome but you will need to find the appropriate libraries yourself or write the code from scratch. Don't hesitate to contact the instructor via email or utilize the mattermost channels for any questions/clarifications you might need.

**Submit a PDF for questions 1-2 and code for question 3 through Connex**

# Question 1 (SENG474: 25 points/ CSC578D: 15 points)

- **Q1.1 (SENG474: 20 points, CSC578D: 10 points** By hand, construct the root and the first level of a decision tree for the contact lenses data (attached to this assignment on connex) using the ID3 algorithm. Show the details of your construction and all your calculations; no points will be given for solutions only.

- **Q1.2 (SENG474: 5 poitns, CSC578D: 5 points)**

  Using the tree.DecisionTreeClassifier module from pythons sckit-learn, fit a tree using the contact-lenses data using criterion=entropy. Compare the entropy values obtain in part a) with the ones calculated by the sklearn.tree module. Explain in detail why the trees are not the same. You may find the documentation for decision trees helpful: `http://scikit-learn.org/stable/modules/tree.html`

  Note: You can import the data directly from the contact-lenses.arff file using the Arff2Skl() converter from util2.py provided with this assignment, using these lines of code:

```
from util2 import Arff2Skl
cvt = Arff2Skl('contact-lenses.arff')
label = cvt.meta.names()[-1]
X, y = cvt.transform(label)
```

# Question 2 (SENG 474: 20 points; CSC578D: 15 points

Calculate the probabilities needed for Nave Bayes using the contact lens dataset. Classify: prepresbyopic, hypermetrope, yes, reduced, ? using your calculated probabilities. Use additive smoothing, as described in `https://en.wikipedia.org/wiki/Additive_smoothing`. Show all the details.

# Question 3 (SENG474: 55 points; CSC578D: 70 points)

- **Q3.1 (SENG474: 40 points, CSC578D: 30 points)**

  Implement Nave Bayes, assuming that each feature is binary (can only take on values 0 or 1). This is called Bernoulli Nave Bayes, because each feature is a Bernoulli random variable. Use the skeleton code provided, as we will be using the class for testing. (If you are not using Python you will have to write a similar skeleton code).

  Recall that for Nave Bayes, the classification decision is:

  $$\hat{y} = \text{argmax}_y(P(y)\prod_{i=1}^{p} P(x_i|y))$$

  where $y$ is the class, $p$ is the total number of attributes (features) in $x$, and $x_i$ is the $i$-th feature of the vector $x$. Using the training data, you must calculate $P(y)$ for each of the classes ($y$), as well as $P(x_i = 0|y)$ and for each $y$ and every feature $x_i$ . Note that $P(x_i = 1|y) = 1-P(x_i = 0|y)$, so you dont need to explicitly calculate that value. Consult the class notes for information on how to calculate these values.

  Implement additive smoothing with $\alpha = 1$. We will be testing your code by creating an instance of MyBayesClassifier and passing it a new binary dataset, so dont assume that you know the number of features or the number of classes.

- **Q3.2 (SENG474: 15 points, CSC578D: 10 points)**

  Implement additive smoothing `https://en.wikipedia.org/wiki/Additive_smoothing` so that you can vary $\alpha$ in your code. Create (and hand in) a plot of the accuracy on the test set as a function of $\alpha$, varying from 0.01 to 1 (use steps of 0.01). Based on your experiment, what is the optimal value for alpha?

- **Q3.3 (SENG474: 0 points, CSC578D: 30 points)**

  Implement a multinomial version of the Nave Bayes classifier that can accept non-binary input vectors (see skeleton in code). We will rewrite the probability of a feature given the class as:

$$P(x_i|y) = \frac{N_{yi} + \alpha}{N_y + \alpha p}$$

  where $N_{yi}$ is the total number of times feature $i$ appeared for any instance with label $y$, and $N_y = \sum_{i=1}^{p} N_{yi}$ is the total number of times any feature appeared in an instance label $y$, $p$ is the total number of features, and $\alpha$ is the additive smoothing parameter. Note that in this implementation, if a feature doesnt appear in a test instance, we ignore it (that is, we dont model $P(x_i|y) = 0$, we just model the existence of features.

  Now you should be able to change the CountVectorizer to have parameter binary=false. Which performs better on the test data, the (binary + Bernoulli) setup or (non-binary + multinomial)? Report your accuracies with each setup.