CSC421: Assignment 3

Student: Jordan Yu, V00727036
Date: March 9th, 2015
Instructor : George Tzantakis

Table of Contents

Question 1: Probability Theory

To begin we determine the probability of winning and losing if we

1. Always Switch
2. Always DO NOT switch

The values can be calculated using Bayes theorem [1].

$Let\ the\ doors\ be\ numbered\ 1,2,3$

$Without\ loss\ of\ generality, assume\ we\ always\ choose\ door\ 1\ for\ our\ first\ guess.$

$let\ Xi\ be\ the\ event\ that\ door\ i\ has\ the\ car.$

$let\ Yi\ be\ the\ event\ that\ door\ i\ is\ open\ by\ the\ host.$

To calculate the probability of winning a car given that we always switch:

$P(X3 \mid Y2) = P(Y2 \mid X3) * P(X3) / P(Y2)$

$P(Y2 \mid X3) = 1$

Door 3 has the car, the user has chosen door 1; therefore the host will always reveal door 2.

$P(X3) = 1/3$

The car can be behind any of the 3 doors.

$P(Y2)$

$1/3 * 1/2 + 1/3 * 0 + 1/3 * 1 = 1/2$

(1) (2) (3)

(1) If the car is behind door 1, then the host can show us either door 2, or door 3

(2) If the car is behind door 2, therefore the host will never show us door 2

(3) If the car is behind door 3, then the host will always show us door 2

$P(X3 \mid Y2) = (1 * 1/3)/(1/2)$

$P(X3 \mid Y2) = 2/3$

To calculate the probability of winning a car given that we don't switch:

$P(X1 \mid Y2) = P(Y2 \mid X1) * P(X1) / P(Y2)$

$P(Y2 \mid X1) = 1/2$

Door 1 has already been chosen by us, therefore the host only has 2 options; either door 2 or door 3

$P(X1) = 1/3$

The car can be behind any of the 3 doors

$P(Y2) = 1/2$

$1/3 * 1/2 + 1/3 * 0 + 1/3 * 1 = 1/2$

(1) (2) (3)

(1) If the car is behind door 1, then the host can show us either door 2, or door 3

(2) If the car is behind door 2, therefore the host will never show us door 2

(3) If the car is behind door 3, then the host will always show us door 2

$P(X1 \mid Y2) = (1/3 * 1/2) / 1/2$

$P(X1 \mid Y2) = 1/3$

Therefore

$Prob(Winning \mid AlwaySwitch) = 2/3$

$Prob(Losing \mid AlwaysSwitch) = 1 - 2/3 = 1/3$

$Prob(Winning \mid \sim AlwaysSwitch) = 1/3$

$Prob(Losing \mid \sim AlwaysSwitch) = 1 - 1/3 = 2/3$

Solutions to the two variants are therefore as follows:
Variant I
$P(Switch \mid Win) \;=\; P(Win, Switch) \;/\; P(Win)$
$$P(Win, Switch) = P(Switch) * P(Win \mid AlwaysSwitch)$$
$$P(Win, Switch) = \left(\frac{1}{2}\right) * \left(\frac{2}{3}\right) = \left(\frac{1}{3}\right)$$
$$P(Win) \;=\; P(Win \mid AlwaysSwitch) * P(Switch) \;+\; P(\,Win \mid \neg AlwaySwitch) * P(\neg Switch)$$
$$P(Win) = \left(\frac{2}{3}\right) * \left(\frac{1}{2}\right) + \left(\frac{1}{3}\right) * \left(\frac{1}{2}\right) = \left(\frac{1}{2}\right)$$
$P(Switch \mid Win) \;=(1/3) \;/\; (1/2) = (2/3)$

Variant II
$P(2\,wins) \;=\; P(Win \mid AlwaysSwitch) * P(Win \mid \neg AlwaysSwitch)$
$P(2\,wins) \;=\; (2/3) * (1/3) \;=\; 2/9$

$P(2\,lose) = \; P(Lose \mid AlwaysSwitch) * P(Lose \mid \neg AlwaysSwitch)$
$P(2\,lose) \;=\; P(1/3) * P(2/3) \;=\; 2/9$

Question 2: Email Categorization

Model Definition

Let C be the classification of the document.
Therefore $C \in \{c_1, c_2, c_3, \ldots, c_n\}$
Let D be the document to classify.
Each document is represented as a vector $[x_1, x_2, x_3\,, \ldots x_n]$
Each entry x_i represents the presence/absence of a keywords w_i
Therefore $x_i \in \{True, False\}$. Note the discreteness of the attributes.

The model we desire is the function
$F(x){:}\, D \,{-}{>}\; C \; which\; maps\; x \in D \; to\; a\; category\; C$

To determine this function we use a naive Bayes model to represent the problem. The naïve Bayes model allows us to assume that each keyword/attribute in the document is conditionally independent given the classification. We also make an assumption that the classification of a document is mutually exclusive.

To begin we investigate the simpler case of identify a document d given only a single classificationc_i.
$P(c_i \mid d) \;=\; (P(\,d\mid c_i) \;*\; P(c_i)) \;/\; P(d)$

This represents the probability that given document d the classification of the document is c_i.

To classify for the more general case we calculate the probability that document D for all the classifications in C and choose the classification which has the highest probability.

$$\operatorname*{Max}_{c_i \in C}(\,P(c_i \mid d))$$

Therefore the final function is:

$$\underset{c_i \in C}{\text{Max}}\left(\frac{P(d\,|c_i) * P(c_i)}{P(d)}\right)$$

We can drop $P(d)$ because it is constant and does not affect the relative probabilities between the classifications.

$$\underset{c_i \in C}{\text{Max}}(P(d\,|c_i) * P(c_i))$$

We expand the document d into its vector form

$$\underset{c_i \in C}{\text{Max}}(P(x_1, x_2, x_3, \dots, x_n|c_i) * P(c_i))$$

We note that $P(x_1, x_2, x_3, \dots, x_n\,|c_i)$ given our naive bayes assumption can be easily calculated

$$P(x_1, x_2, x_3, \dots, x_n\,|c_i) = \prod P(x_i|c_i)$$

Therefore the model we wish to construct is the conditional probability table for each x_i given c_i
1. $foreach\ x_i, c_i\ P(\ x_i\,|c_i)$
2. $foreach\ x_i, c_i\ P(\neg x_i\,|c_i)$
3. $P(c_i)$

These can easily be calculated using the training data set.
Let n be the number of classified documents in the data set.
Let m_i be the number of documents classified as c_i in the data set.
$P(c_i) = m_i\,/\,n$

$Let\ r_i\ be\ the\ number\ of\ documents\ classified\ as\ ci\ and\ contains\ the\ word\ x_i$
$P(x_i\,|c_i) = \dfrac{r_i}{m_i}$
$P(\neg x_i\,|c_i) = 1 - P(x_i\,|c_i)$

Implementation
The training data was created using my personal emails as samples.

The three categories chosen are:
School – Emails which relate to course work, or team projects. Tuitions and school events are also included in this category.
The keywords use for school email: [connex, uvic, assignment, student]

Personal – Emails which originate from family members, or are dealing which personal finances and services attached to the email.
The keywords used for personal email: [steam, programming, lily, peter]

Professional (Prof) – The category of emails which belong to correspondence with employers or businesses. This may include recruiter emails, interviews, and professional profiles.

The keywords used for professional email: [linkedin, coop, interview, university]

The main issues faced during the implementation:

1. How to deal with zero frequency keywords when calculating $P(x_i \mid c_i)$
   The solution.
   Add an extra factor when calculating the $P(x_i \mid c_i)$
   Therefore $P(x_i|c_i) \ = \frac{NumberOf(x_i)}{NumberOf(c_i)}$ becomes

   $$P(x_i|c_i) \ = \frac{NumberOf(x_i)+1}{NumberOf(c_i)+NumberOf(keywords)}$$

   This ensures that the weights from each probability will at-least greater than zero and the relative weights of each keyword does not change.

2. What do you multiply if the keyword is not present?
   When calculating the $P(x_1, x_2, x_3, \dots, x_n| \ c_i)$ if the keyword is not present in the document then we ignore the weight from the probability.

3. Tedious to classify all emails for the training set.
   There was no clean solution for this problem. I just went through every email in my inbox, and extracted out the relevant keywords and classified into one of the three categories.

To use the classifier

```
b = Bayes();
b.setParameters({
    "school" : ["connex","uvic","assignment","student"],
    "personal" : ["steam","programming","lily","peter"],
    "prof" : ["linkedin","coop","interview","university"]
});
b.train("training_data.txt")
print(b.predict("email"))
```

An example usage of the model, the source code and the training data used can be found under the code listings section of this report.

Sample Emails

Five sample emails for each classification was generated using the model and keywords from the previous section. The resulting emails are listed here.

Classification: School
Email 1: assignment uvic connex steam student connex uvic programming connex connex
Email 2: assignment university student student assignment programming lily peter connex student
Email 3: university steam coop programming assignment programming connex connex uvic university
Email 4: connex student coop steam lily student uvic student university coop
Email 5: peter coop uvic assignment connex linkedin connex student uvic student

Classification: Personal

Email 1: university linkedin interview linkedin steam programming coop lily connex lily
Email 2: linkedin uvic lily steam interview interview lily programming connex programming
Email 3: uvic peter steam lily connex coop student steam programming lily
Email 4: uvic coop coop programming peter university lily coop steam peter
Email 5: programming linkedin peter lily lily steam connex steam steam student

Classification: Professional
Email 1: coop coop university coop coop connex interview linkedin assignment coop
Email 2: linkedin coop student lily university linkedin student coop student student
Email 3: uvic coop linkedin peter linkedin student peter uvic university linkedin
Email 4: linkedin interview coop linkedin linkedin interview interview student interview assignment
Email 5: lily university uvic connex coop coop coop linkedin linkedin coop

Question 3: Bayesian Network

Application and Data

The application area chosen for the Bayesian network:
Cooling down oneself if one is overheated.

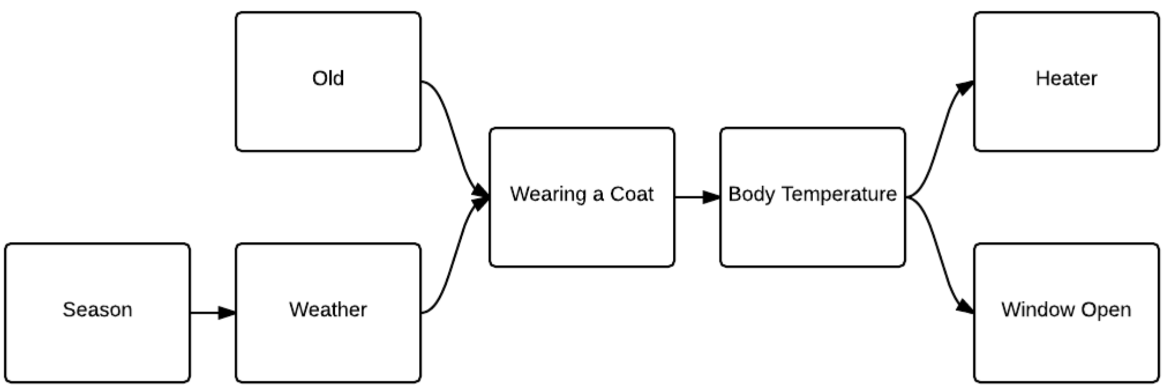


FIGURE 1: TOPOLOGY OF THE BAYESIAN NETWORK

Season				
P(Winter)	P(Spring)	P(Summer)	P(Fall)	
0.25	0.25	0.25	0.25	
Weather				
Season	P(Sunny | Season)	P(Cloudy | Season)	P(Raining | Season)	P(Windy | Season)
Winter	0.2	0.4	0.2	0.2
Spring	0.35	0.12	0.3	0.23
Summer	0.7	0.1	0.1	0.1
Fall	0.1	0.2	0.6	0.1
Old				
P(Old)	P(~Old)			
0.45	0.55			
Coat				
Weather	Old	P(Coat | Weather,Old)	P(~Coat | Weather, ~Old)	
Winter	T	0.78	0.22	
Winter	F	0.78	0.22	
Spring	T	0.65	0.35	
Spring	F	0.54	0.46	
Summer	T	0.3	0.7	
Summer	F	0.22	0.78	
Fall	T	0.55	0.45	
Fall	F	0.44	0.66	
Body Temp				
Coat	P(Hot | Coat)	P(Warm | Coat)	P(Cold | Coat)	
T	0.68	0.22	0.1	
F	0.4	0.3	0.3	
Window				
BodyTemp	P(Window)	P(~Window)		
Hot	0.7	0.3		
Warm	0.5	0.5		
Cold	0.2	0.8		
Heater				
BodyTemp	P(Heater)	P(~Heater)		
Hot	0.1	0.9		
Warm	0.55	0.45		
Cold	0.76	0.24		

TABLE 1: CONDITIONAL PROBABILITIES OF THE DISCRETE VARIABLES

This is a discrete Bayesian network with the following random variables
$Season \in \{Winter, Spring, Summer, Fall\}$
$Weather \in \{Sunny, Cloudy, Raining, Windy\}$

$isOld \in \{T, F\}$
$Coat \in \{T, F\}$
$Body\ Temperature \in \{Hot, Warm, Cold\}$
$Window \in \{T, F\}$
$Heater \in \{T, F\}$

The probabilities could be calculated by questionnaire.
For instance, sampling the weather given the season can be done through observation of past weather data. Samples for old or not old can be found by questioning young individuals, and asking how often they wear a coat in doors, and whether or not they turn on the heat, or open the window.

Sample Queries

For simplification let

Season = S	Weather = WE	Old = O	
Coat = C	Body Temp = B	Window = WI	Heater = H

The following are four sample queries that can be made using the network.
$P(S\ |c = F, o = T, b = Hot, h = F)$
$P(WI\ |s = summer, we = rain, o = F, c = T)$
$P(\ H\ |wi = f, c = t, we = cloudy, o = f)$
$P(H\ \ |\ wi = T, b = Hot, we = wind, o = T)$

Calculations using exact inference by enumeration.
1.)
$$P(\ S\ \ |c\ =\ F, o\ =\ T, b =\ Hot, h = F)$$
$$= \alpha \sum_{WE} \sum_{WI} P(h = F \mid b = Hot) \cdot P(WI\ |b = Hot) \cdot P(b = Hot|c = F) \cdot P(c = F|o = T, WE) \cdot P(o = T) \cdot P(WE\ |S) \cdot P(S)$$
$$= \alpha P(\ h = F\ |b = Hot) \cdot P(\ b = Hot\ |c = F) \cdot P(\ o = T) \cdot P(S) \cdot \sum_{WE} P(c = F|o = T, WE) \cdot P(WE|S) \cdot \sum_{WI} P(WI\ |b = Hot)$$
$$= \alpha(0.1) \cdot (0.4) \cdot (0.55) \cdot$$
$$(P(c = F\ |o = T, we = Sunny) \cdot\ (P(wi = True\ |b = Hot) \cdot P(wi = False\ |b = Hot)) +$$
$$P(c = F\ |o = T, we = Cloudy) \cdot\ (P(wi = True\ |b = Hot) \cdot P(wi = False\ |b = Hot)) +$$
$$P(c = F\ |o = T, we = Rain) \cdot\ (P(wi = True\ |b = Hot) \cdot P(wi = False\ |b = Hot)) +$$
$$P(c = F\ |o = T, we = Windy) \cdot\ (P(wi = True\ |b = Hot) \cdot P(wi = False\ |b = Hot)))$$

2.)
$$P(WI\ \ |s = Summer, we = Raining, o = F, c = T)$$
$$= \alpha \sum_{H} \sum_{B} P(WI|B) \cdot P(H\ |B) \cdot P(B \mid c = T) \cdot P(c = T\ |o = F, we = rainy) \cdot P(o = F) \cdot P(we = rainy \mid s = summer) \cdot P(s = summer)$$
$$= \alpha \cdot P(c = T\ |o = F\ , we = rain) \cdot P(o = F) \cdot P(we = rain\ |s = summer) \cdot P(\ s = summer) \cdot \sum_{H} \sum_{B} P(WI|B) \cdot P(H\ |B) \cdot P(B \mid c = T)$$

Calculations using variable elimination

3.)

$P(H \mid wi = F, c = T, we = Cloudy, o = F)$

$= \alpha \sum_S \sum_B P(H|B) \cdot P(wi = F|B) \cdot P(B \mid c = T) \cdot P(c = T|o = F, we = cloud) \cdot P(o = F) \cdot P(we = cloud \mid S) \cdot P(S)$

$= \alpha P(c = T \mid o = F, we = cloud) \cdot$

$P(o = F) \cdot \sum_S P(we = cloud\,|S) \cdot P(S) \cdot \sum_B P\,(wi = F|B) \cdot P(B|c = T) \cdot P(H \mid B)$

$= \alpha f_1(\ \ ) \times \sum_S f_2(S) \times f_3(S) \times \sum_B f_4(B) \times f_5(B) \times f_6(H, B)$

$= \alpha f_1(\ \ ) \times \sum_S f_2(S) \times f_3(S) \times \sum_B f_7(H, B)$

where $\mathrm{f_8(H) = (f_4(b = hot) \times f_5(b = hot) \times f_6(H, b = hot) +}$
$\mathrm{f_4(b = hot) \times f_5(b = hot) \times f_6(H, b = hot) +}$
$\mathrm{f_4(b = hot) \times f_5(b = hot) \times f_6(H, b = hot))}$

$= \alpha f_1(\ \ ) \times \sum_S f_2(S) \times f_3(S) \times f_8(H)$

$= \alpha f_1(\ \ ) \times f_8(H) \sum_S f_2(S) \times f_3(S)$

where $\mathrm{f_{10}(\ \ ) = ((f_2}(s = winter) \times \mathrm{f_3}(s = winter)) +$
$(\mathrm{f_2}(s = spring) \times \mathrm{f_3}(s = spring)) +$
$(\mathrm{f_2}(\ s = fall) \times \mathrm{f_3}(s = fall)) +$
$(\mathrm{f_2}(s = summer) \times \mathrm{f_3}(s = summer))\ )$

$= \alpha \mathrm{f_1}(\ \ ) \times f_8(H) \times f_{10}(\ \ )$

4.)

$P(H \mid wi = T, b = Hot, we = Windy, o = T)$

$= \alpha \sum_S \sum_C P(H|b = hot) \cdot P(wi = T\,|b = Hot) \cdot P(b = Hot \mid C) \cdot P(C\,|we = wind, o = T) \cdot P(o = T) \cdot P(\ we = wind \mid S) \cdot P(S)$

$= \alpha P(H|\ b = hot) \cdot P(wi = T\,|b = Hot) \cdot P(o = T) \cdot \sum_S P(S) \cdot P(we = wind\,|S) \cdot \sum_C P(b = Hot\,|C) \cdot P(C\,|we = wind, o = T)$

$= \alpha \times f_1(H) \times f_2(\ \ ) \times f_3(\ \ ) \times \sum_S f_4(S) \times f_5(S) \times \sum_C f_6(\ C) \times f_7(C)$

$where\ f_7(\ \ ) = \big(f_6(c = T) \times f_7(c = T)\big) + \big(f_6(c = F) \times f_7(c = F)\big)$

$= \alpha \times f_1(H) \times f_2(\ \ ) \times f_3(\ \ ) \times \sum_S f_4(S) \times f_5(S) \times f_7(\ \ )$

$= \alpha \times f_1(H) \times f_2(\ \ ) \times f_3(\ \ ) \times f_7(\ \ ) \times \sum_S f_4(S) \times f_5(S)$

$where\ f_8(\ \ ) = \big(f_4(s = winter) \times f_5(s = winter)\big) + \big(f_4(s = spring) \times f_5(s = spring)\big) +$
$\big(f_4(s = summer) \times f_5(s = summer)\big) + \big(f_4(s = fall) \times f_5(s = fall)\big)$

$= \alpha \times f_1(H) \times f_2(\ \ ) \times f_3(\ \ ) \times f_8(\ \ )$

Question 4: Learning and Decision Trees

The application area chosen is described as such:

Goal Predicate: Should I attend class today?

Attributes: Type: Lecture, Lab, Tutorial

Midterm: T, F

Time of class: Early, Afternoon, Late

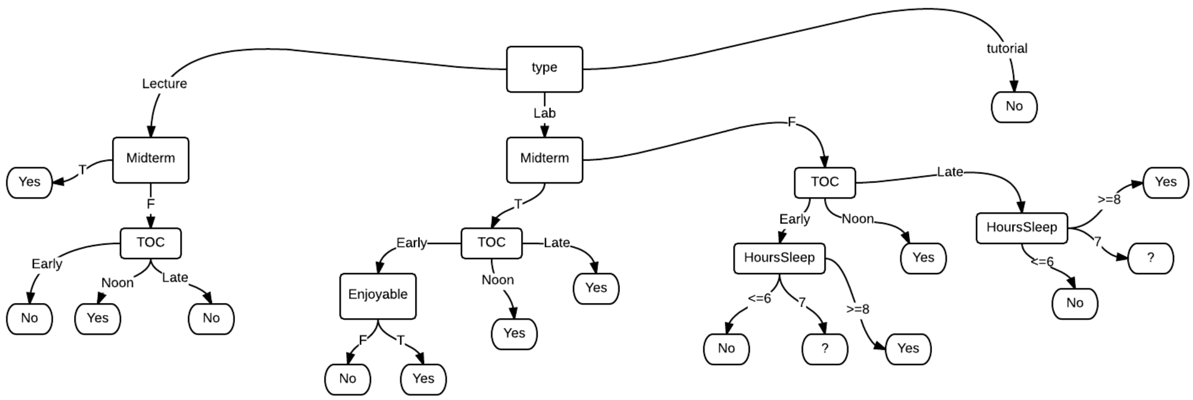
Enjoyable : Yes, No

Hours of Sleep Last Night : <=6, 7, >= 8

The test data used for the decision tree is show below.

Type	Midterm	Time Of Class	Hours of Sleep	Enjoyable	Goal
Lab	F	Early	7	F	T
Lecture	F	Early	7	F	F
Tutorial	F	Late	7	F	F
Lecture	T	Late	7	T	T
Lab	F	Noon	7	F	T
Lecture	F	Noon	7	F	T
Tutorial	F	Noon	7	F	F
Lab	T	Noon	7	F	T
Lecture	T	Noon	7	F	T
Lab	F	Early	<=6	F	F
Lecture	F	Early	<=6	F	F
Tutorial	F	Early	<=6	F	F
Lab	T	Early	<=6	F	F
Lab	T	Early	<=6	T	T
Tutorial	T	Early	<=6	F	F
Lab	F	Late	<=6	F	F
Tutorial	F	Late	<=6	F	F
Tutorial	T	Late	<=6	T	F
Lab	F	Noon	<=6	F	T
Tutorial	T	Noon	<=6	F	F
Tutorial	T	Noon	<=6	T	F
Lecture	T	Early	>=8	F	T
Lecture	T	Early	>=8	T	T
Lab	F	Late	>=8	F	T
Lecture	F	Late	>=8	T	F
Lab	T	Late	>=8	F	T
Lab	T	Late	>=8	T	T
Lecture	F	Noon	>=8	F	T
Tutorial	F	Noon	>=8	F	F
Lecture	T	Noon	>=8	T	T

Full Simple Decision Tree



Information Gain Heuristic

Using the information gain heuristic we apply an algorithm to choose the best attribute to use for each level of the decision tree. The heuristic informs us of the amount of entropy gained by choosing each specific attribute.

A program was written to calculate the Gain() from each attribute. This entropy gain is used to inform the decision tree which attribute would be most beneficial to check next. In this way a smaller decision tree can be created.

The following table shows the entropy gain on each attribute for the initial root node.

Attribute Name	Gain
Enjoyable	0.016529
Time of Class	0.032285
Type	0.396274
Amount of Sleep	0.235241
Midterm	0.052168

The highest entropy gain in this case comes from choosing the ‘type’ attribute.
This process is then repeated for each sub-tree of the graph.
Therefore the gain will need to calculated for the remaining attributes

Type = Lecture	Type = Lab	Type = Tutorial
Enjoyable	Enjoyable	Enjoyable
Time of class	Time of class	Time of class

Amount of Sleep	Amount of Sleep	Amount of Sleep
Midterm	Midterm	Midterm

We can observe that *Type* and *Amount of Sleep* provide the most gain when creating the decision tree. The full program used to generate this output can be seen under the Code Listings section, under Question 4 Code.

Weka

Having run the training set through Weka, the following decision tree and evaluation output was created.

```
=== Run information ===
Scheme:weka.classifiers.trees.Id3
Relation:     ATTENDCLASS
Instances:    30
Attributes:   6
              Type
              Midterm
              Time
              HoursSleep
              Enjoyable
              GoToClass
Test mode:10-fold cross-validation

=== Classifier model (full training set) ===

Id3


Type = Lab
|  HoursSleep = <=6
|  |  Time = Early
|  |  |  Enjoyable = T: T
|  |  |  Enjoyable = F: F
|  |  Time = Noon: T
|  |  Time = Late: F
|  HoursSleep = 7: T
|  HoursSleep = >=8: T
Type = Lecture
|  Midterm = T: T
|  Midterm = F
|  |  Time = Early: F
|  |  Time = Noon: T
|  |  Time = Late: F
Type = Tutorial: F

Time taken to build model: 0.02 seconds
```

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        24               80      %
Incorrectly Classified Instances       6               20      %
Kappa statistic                        0.6
Mean absolute error                    0.2
Root mean squared error                0.4472
Relative absolute error               39.5455 %
Root relative squared error           88.3807 %
Total Number of Instances             30

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.8       0.2       0.8       0.8       0.8        0.8      T
                 0.8       0.2       0.8       0.8       0.8        0.8      F
Weighted Avg.    0.8       0.2       0.8       0.8       0.8        0.8

=== Confusion Matrix ===

  a  b   <-- classified as
 12  3 |  a = T
  3 12 |  b = F
```

Important observations from the classifier includes:

1. Classifier exhibited an 80% correctness percentage. This is a modest result given the small dataset. Looking at the confusion matrix we can see that for both Boolean cases the classifier was equally bad.
2. Looking at the decision tree we see that the top 2 levels of the tree use the attributes “Type”, and “HoursSleep” which conforms to the our expected attributes as measured by the entropy gains.

Code Listing

Question 2 Code

```
import sys
import random
from pprint import pprint,pformat

class Classification:
    def __init__(self,classifier_name,words):
        self.classifier_name = classifier_name
        self.prob_classifier = 0
        self.prob_keyword = {}
```

```
        for w in words:
            self.prob_keyword[w] = 0

    def __str__(self):
        return "'{}':{}\n{}\n".format(self.classifier_name,self.prob_classifier,
                              pformat(self.prob_keyword))
    def __repr__(self):
        return self.__str__()


class Bayes:
    def __init__(self):
        self.classifiers = {"classification": ['keywords']}
        self.probs = { 'classification': {'keyword':0.4}}
        self.words = set()

    """
    @param d - dictionary containing the classifications and keywords
        for this naive bayes model.
        Assumption is everything is lower case.
    """
    def setParameters(self, d):
        self.classifiers = d

        # create a set of the keywords
        self.words = set()
        for c in d:
            self.words = self.words.union(d[c])

        # create Classification objects
        self.probs = {}
        for c in self.classifiers:
            self.probs[c] = Classification(c,self.words)

    """
    @param training_data (string) - filename of training data.
        the format is the following
        <classification> <word>,...,<word>
    """
    def train(self, training_data):
        f = open(training_data,"r")

        # temp data structure to hold the number of occurences of each
        # classification as well the count on the number of words
        data = {}
        for x in self.classifiers:
            data[x] = {
                "count":0,
```

```
            "words":{}
        }
    num_entries = 0

    # read through the file and fill in the counts
    for line in f:
        line = line.rstrip('\n')
        words = line.split()

        # error checking
        if(len(words) <= 0):
            continue

        class_type = words[0].lower()
        words = words[1:]

        num_entries += 1

        # increment a count on a class_type
        data[class_type]["count"] += 1;

        # go through all the words for this line and keep a count on them
        for w in words:
            w = w.lower()
            if w in data[class_type]["words"]:
                data[class_type]["words"][w] += 1
            else:
                data[class_type]["words"][w] = 1
    f.close()

    # parse through data and calculate the probabilties
    alpha = 1
    for k,c in self.probs.items():
        c.prob_classifier = (float(data[k]['count']) + alpha)/(num_entries + alpha*len(self.words))
        for w in c.prob_keyword:
            word_count = 0 if not w in data[k]['words'] else (data[k]['words'][w])
            c.prob_keyword[w] = float(word_count + alpha)/(data[k]['count'] + alpha*len(self.words))

"""
    @param doc(string)- the filename of the document to classify
    @return (string) - string representing the classifcation of the document.
"""
def predict(self, doc):
    # read in the file and determine the words that are present
    f = open(doc,"r")
    words = set()
    for line in f:
        line = line.rstrip('\n')
```

```
            words = words.union(line.split())
        f.close()

        # For all the words present in the document
        # take the product of all the probabilties for each classifcation
        # and take the classification with the highest probability
        max_c = None
        max_prob = 0
        for k,c in self.probs.items():
            prob = c.prob_classifier
            for w in c.prob_keyword:
                if( w in words):
                    # word is present, multiply against the probability
                    prob *= c.prob_keyword[w]

            # if the final probablity is larger than everything, then record it
            if(max_c == None or prob > max_prob):
                max_c = k

        return max_c

    def generate(self,class_type,doc_len=10):
        # extract the array of prob,keyword pairs and sort
        # by from lowest to greatest prob
        vals = self.probs[class_type].prob_keyword.items()
        vals = map(lambda x : x[::-1], vals)
        vals.sort()

        #normalize the probabilities nto the range 0,1
        prob_sum = reduce(lambda x,y : (x[0] + y[0],0),vals)[0]
        vals = map(lambda x : (x[0]/prob_sum,x[1]), vals)

        # construct the string from the keywords based on the prob distribution
        rs = ""
        for i in xrange(doc_len):

            # randomly choose a keyword
            r = random.random()
            for v in vals:
                if r <= v[0]:
                    rs += v[1] + " "
                    break
                else:
                    r -= v[0]
        return rs

b = Bayes();
b.setParameters({
```

```
    "school" : ["connex","uvic","assignment","student"],
    "personal" : ["steam","programming","lily","peter"],
    "prof" : ["linkedin","coop","interview","university"]
});

train_file = "q2_training_data.txt"
predict_file = "q2_training_data.txt"
b.train(train_file);
print("predict '{}' => {}\n".format(train_file,b.predict(predict_file)))

def gen(num,type):
    print(type)
    for x in xrange(num):
        print(b.generate(type,10))
    print("")

gen(5,"school")
gen(5,"personal")
gen(5,"prof")
```

Question 4 Code

```
import sys
import math
from pprint import pprint


f = open('q4_data.txt',"r")


attrs = {
    'type':["lecture","lab","tutorial"],
    'midterm':["t","f"],
    'toc':["early","noon","late"],
    'sleep':["<=6","7", ">=8"],
    'enjoyable':["t","f"],
}

data = []
neg = 0
pos = 0
for line in f:
    line.rstrip('\n')
    words = line.split()
    words = map(lambda x: x.lower(), words)
    a = {
        'type':words[0],
        'midterm':words[1],
```

```
        'toc':words[2],
        'sleep':words[3],
        'enjoyable':words[4],
        'goal':words[5]
    };
    if( a['goal'] == 't'):
        pos += 1
    else:
        neg += 1
    data.append(a)


def parse_for_pk_nk(data):
    rs = {}
    for i in attrs:
        rs[i] = {}
        for j in attrs[i]:
            rs[i][j] = {
                'pk': 0 ,
                'nk' : 0
            }

    for row in data:
        for attr in attrs:
            val = row[attr]
            if row['goal'] == 't':
                rs[attr][val]['pk'] += 1
            else:
                rs[attr][val]['nk'] += 1

    return rs

data_pk_nk = parse_for_pk_nk(data)

def B(q):
    if(q == 0):
        return 0
    else:
        return -1*(q*math.log(q,2) + (1-q)*math.log(1-q,2))

def Remainder(attr):
    rs = 0
    for k in attrs[attr]:
        pk = data_pk_nk[attr][k]['pk']
        nk = data_pk_nk[attr][k]['nk']
        rs += (float((pk + nk))/( pos + neg ))*B( float(pk)/(pk + nk))
    return rs
```

```
def Gain(attr):
   return B(float(pos)/ ( pos + neg)) - Remainder(attr)


for attr_name in data_pk_nk:
   print(attr_name)
   for attr_type in data_pk_nk[attr_name]:
      print("," + attr_type + "," + str(data_pk_nk[attr_name][attr_type]['pk']) + "," +
str(data_pk_nk[attr_name][attr_type]['nk']))

for attr_name in attrs:
   print(str(attr_name) +  "," + str(Gain(attr_name)))
```

Data (Spreadsheets + ARFF)

Training Data for Question 2

school connex assignment uvic seng student students
school uvss uvic students
school student uvic survey university
school graduating grad
school announcement connex class
school workterm reports coursespaces coop university victoria
school undergraduate university victoria v00727036 graduation
school worklog report team project
school uvic course survey instructor university victoria
school project office email ELW
school project lol ELW
school project group class team
school meeting project
school student university victoria
school midterm announcement connex
school connex announcement assignment
school connex announcement midterm grades
school midterm assignment grade
school assignment connex lecture
school assignment grade submission connex
school assignment submission grade v00727036 connex
school exam announcement connex
school grade report submitted
school connex assignment submission
prof linkedin coop engn undergrad connect
prof automatically
prof dear t4 employer regards
prof google recruiter coordinator sincerely
prof interview coordinators onsite questions
prof t4 payroll
prof engineers engn-ugrad student survey regards canada
prof phone interview candidacy hiring feedback

prof regards technical interviews
prof recruit recruiting student graduates software enigneering resume transcript
prof graduation application tuition uvic mypage student
prof workterm logbooks worksite
prof invitations linkedin profile
prof sin canadian passport coop intern employment
prof coop apartment
prof coop intern
prof shortlist interview intern student
prof coop intern engrcoop learninginmotion interview
prof bell bill e-bill
prof engn-ugrad university
prof reset password account
prof engn-ugrad engineering student uvic students
prof linkedin profile skill endorse
prof survey
prof linkedin congratulate network
personal stymphalian steam games game
personal family peter
personal steam game gift
personal programming
personal programming
personal lily
personal peter fatboy
personal steam game
personal school uvic tuition

ARFF File for Question 4

```
% Weka data file for Question 4

@RELATION ATTENDCLASS
@ATTRIBUTE Type {Lab,Lecture,Tutorial}
@ATTRIBUTE Midterm {T,F}
@ATTRIBUTE Time    {Early,Noon,Late}
@ATTRIBUTE HoursSleep {<=6,7,>=8}
@ATTRIBUTE Enjoyable {T,F}
@ATTRIBUTE GoToClass {T,F}

@DATA
Lab,F,Early,7,F,T
Lecture,F,Early,7,F,F
Tutorial,F,Late,7,F,F
Lecture,T,Late,7,T,T
Lab,F,Noon,7,F,T
Lecture,F,Noon,7,F,T
Tutorial,F,Noon,7,F,F
Lab,T,Noon,7,F,T
Lecture,T,Noon,7,F,T
```

```
Lab,F,Early,<=6,F,F
Lecture,F,Early,<=6,F,F
Tutorial,F,Early,<=6,F,F
Lab,T,Early,<=6,F,F
Lab,T,Early,<=6,T,T
Tutorial,T,Early,<=6,F,F
Lab,F,Late,<=6,F,F
Tutorial,F,Late,<=6,F,F
Tutorial,T,Late,<=6,T,F
Lab,F,Noon,<=6,F,T
Tutorial,T,Noon,<=6,F,F
Tutorial,T,Noon,<=6,T,F
Lecture,T,Early,>=8,F,T
Lecture,T,Early,>=8,T,T
Lab,F,Late,>=8,F,T
Lecture,F,Late,>=8,T,F
Lab,T,Late,>=8,F,T
Lab,T,Late,>=8,T,T
Lecture,F,Noon,>=8,F,T
Tutorial,F,Noon,>=8,F,F
Lecture,T,Noon,>=8,T,T
```

References

[1] C. Long, "The Bayes Solution to Monty Hall," 28 June 2012. [Online]. Available: http://angrystatistician.blogspot.ca/2012/06/bayes-solution-to-monty-hall.html. [Accessed 4 March 2015].

[2] P. N. Stuart Russel, Artificial Intelligence : A Modern Approach Third Edition, Upper Saddle River, New Jersey: Prentice Hall, 2010.