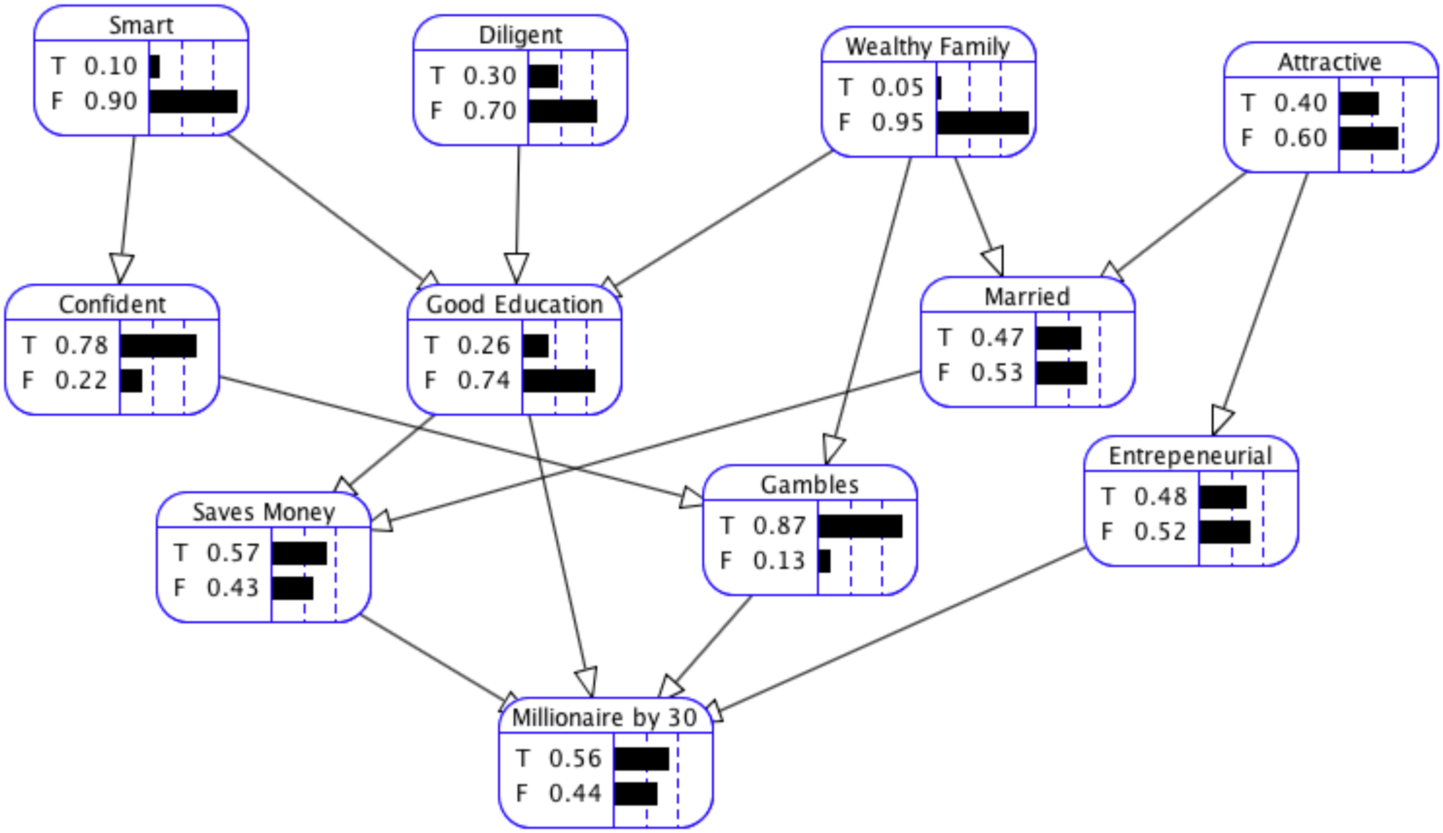
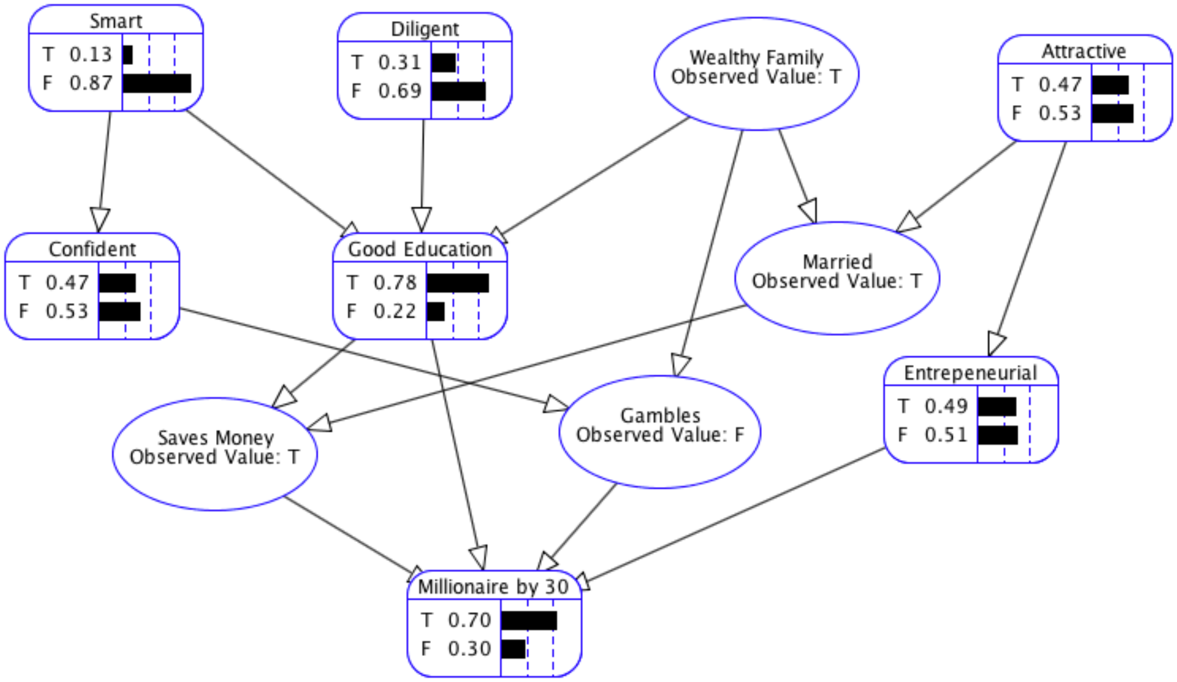
Haig Nalbandian
CSCI 360, Koenig
Project 3 May 1, 2015

Part 1

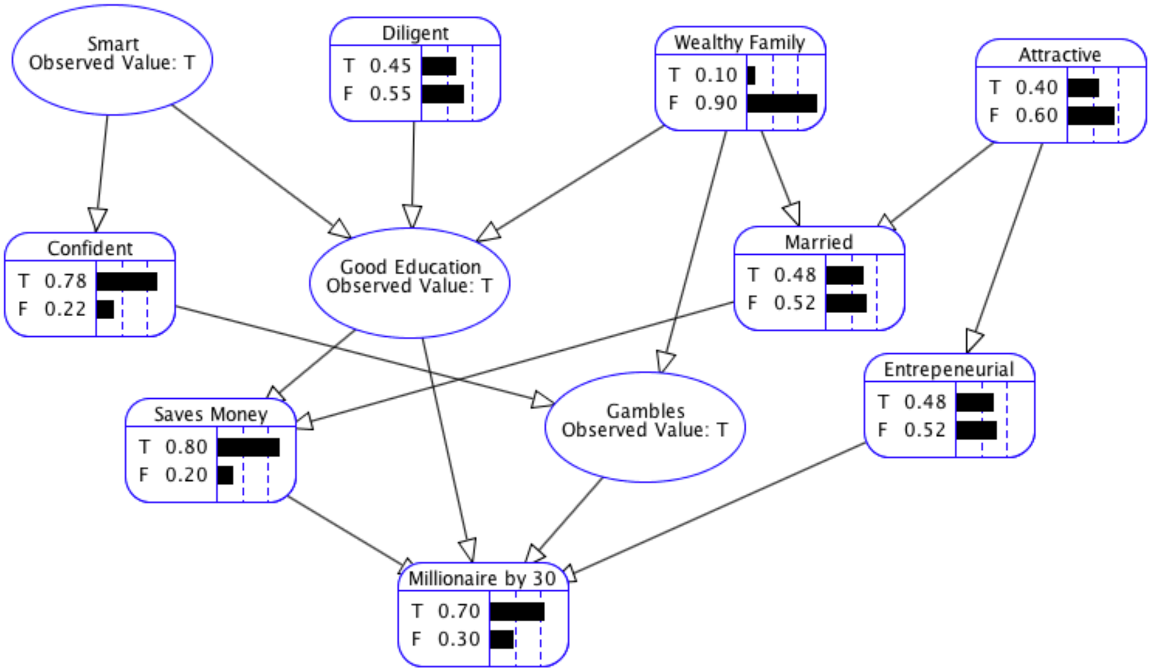
1A:



1B:



Here we observe a person who comes from a wealthy family, is known to save money and not gamble, and is married. Based on these known variables, the probability of being a millionaire goes up by ~14%. This makes sense, as these are all advantageous positions to hold (having money, being conservative with it, and earning tax benefits from being married).



In this example, we only consider that we know the person to be a smart USC student and a gambler. Say we heard of them at a Vegas alumni event. We have not even seen them. Knowing these things, we know they are more likely to diligent, and a millionaire. We also observe their likelihood of being from a wealthy family has doubled from 5% to 10%, among other changes.

1C: In my second scenario in 1B, I will argue that “Entrepreneurial” and “Confident” are conditionally independent in the graph, but dependent in real life.

Consider the “pipe” argument and the three traditional conditional independencies in a directed graph. Starting from “Confident”, we first go to “Gambles.” The route is not blocked to “Wealthy Family” since “Gambles” is given. However, the route to “Attractive” through “Married” is blocked, and thus we cannot go that way. Say from “Gambles” we try to get to “Entrepreneurial” directly. That is also blocked since “Millionaire” is not given. Thus, they are conditionally independent since these are the only two ways to get to “Entrepreneurial” (All paths must go through “Attractive or “Millionaire.”)

Of course, this is not necessarily representative of real life. One can imagine that given someone smart, well-educated, and fond of gambling, the person’s entrepreneurial fervor would depend on their diligence. What’s likely not modeled in the graph is Diligence’s direct effect on Entrepreneurship which likely plays a factor in real life.

PART 2

EXECUTION INSTRUCTIONS FOR CODE

Compile the single cpp file using gcc 4.8 or higher, and be sure to include C++11 in the compilation.

Run the resulting executable in the same directory as the data folder. That is, both the data folder and the executable show up in the same directory.

I have already included the data folder in the right location in my zip file, so feel free to compile it and run it exactly where it is.

DISCUSSION

What is the difference of the evaluation results (precisions and recalls) on the training set and the test set? Can you provide some possible reasons?

Testing:

Precision: 0.895238
Recall: 0.723077

Training:

Precision: 0.946154
Recall: 1

Because the probabilities are calculated using the training set, the filter is adapted to that set of examples. The error will be smaller there because the results are based on themselves.

The testing set will generate more errors because the probabilities used are generated from a different set, and thus have some differences in word choice, etc.

Based on the probabilities learned in your naive Bayesian classifier, which 3 words do you think are most likely to occur in Spam emails? Which 3 words are most likely to occur in Ham emails?

Spam: http, more, email
Ham: enron, please, attached

These are the words that appear most often in the spam and ham emails respectively, and thus makes theirs probabilities the highest to occur in their respective email types.

This makes sense, since it is likely that spam contains links and ham contains insider information about oil futures and/or attachments. (Note: subject was also a high candidate for ham, but was ignored due to “subject” being a standard part of every email.)

There must be some Ham emails misclassified as Spam. Take one as an example, think about why it is misclassified, and provide at least one suggestion to improve the precision of the native Bayesian classifier.

One such email is data/test/ham/0320.2000-02-03.farmer.ham.txt.

From reading the email and analyzing the results from question 2, I feel this email was classified as spam because of the “get your private, free email at http://www.hotmail.com” signature.

To improve the Bayesian classifier, we can expand the quantity of words we consider.

Better yet, we can consider where the words appear in the email. Because keeping track of the exact word location would make little sense, we can instead keep track of individual sentences and treat those as bags of words. Therefore, we can rank emails by how many “spammy” sentences they have instead of how “spammy” they are overall.

We could also expand the dictionary.