Bayesian network classifiers for the German credit data

Scott A. Zonneveldt, Kevin B. Korb and Ann E. Nicholson
{scottzonneveldt@gmail.com, kbkorb@gmail.com,
ann.nicholson@infotech.monash.edu.au}

Abstract

Accurately predicting whether a loan will be repaid (credit scoring) is an important task for any bank. High accuracy benefits both the banks and the loan applicants. This report applies Bayesian Networks to freely available German credit data (Asuncion and Newman, 2007) and attempts to classify the sample data into default and non-default categories. A hand crafted network and several different TANs are developed and compared to naive Bayes. The optimal cut off for giving a loan is calculated and used in a cost sensitive analysis of each network. Predictive accuracy results show that even when compared to more complicated TANs and causal networks, the naive Bayes classifier performs strongly; however, when looking at the expected value of loans, none of the classifiers significantly outperform the default behavior of simply granting every loan application.

Introduction

In this report we develop and compare several different Bayesian Networks which attempt to predict whether loan applicants will default or not. The networks are tailored to the freely available German credit data (Asuncion and Newman, 2007) and are created using the Netica application and Java API. We compare naive Bayes (NB) models to different augmented NB models and a handcrafted causal nework. For each network we determine the accuracy of its predictions and perform a cost sensitive analysis using a cost matrix provided with the data, as well as other matrices.

The German credit dataset used within this report contains 1,000 records, with 20 predictive variables in each record.

Handcrafted Causal Network

The first network we created was designed for a generic credit scoring scenario. Nodes within the net were based upon intuition. Since the network was not tailored to any existing problem, we took the intersection of it and the variables in the German credit dataset to give us the possibility of testing it. The resultant network had 12 evidence variables, feeding into 6 inner nodes, which, in turn, feed into the outcome node, LoanRepaid.

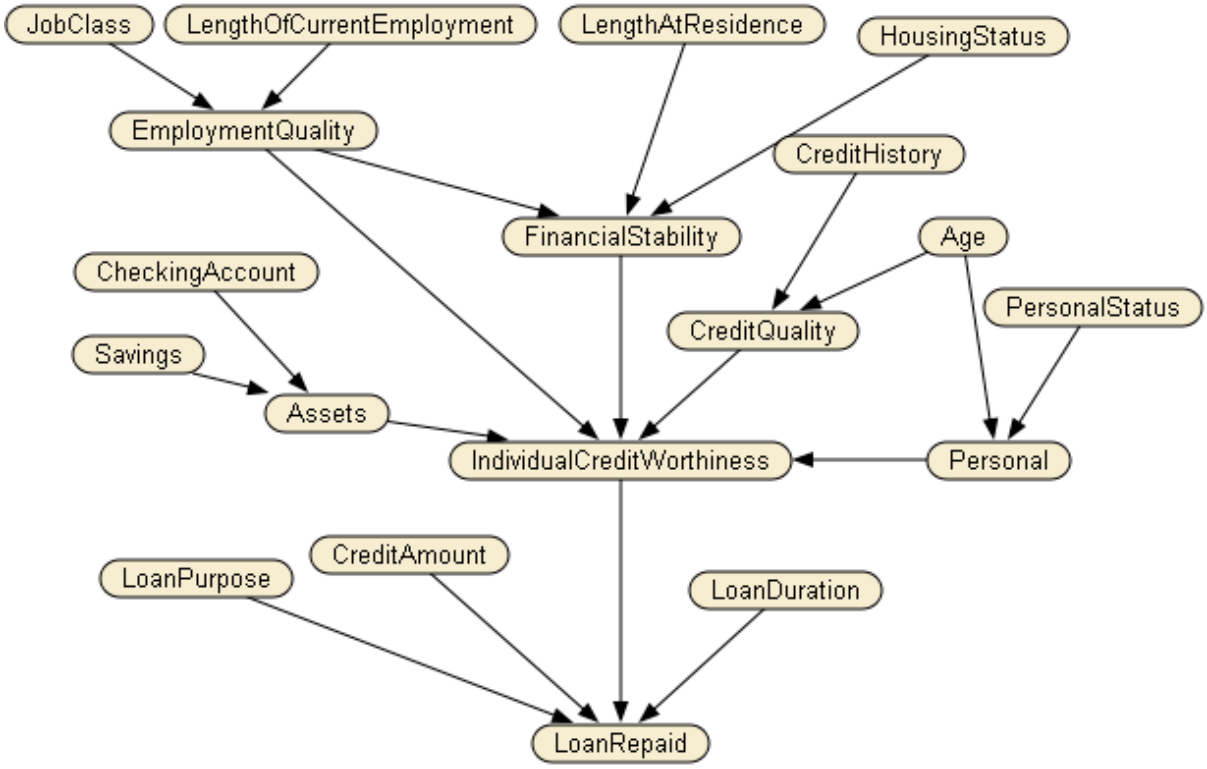

Figure 1 - The first handcrafted net (GermanCreditwithEquations07.neta)

The inner nodes serve two purposes: they group data into qualitative conceptual categories, making interpretation of the BN easier, and they reduce the size of the CPTs in the network (via "divorcing"). However, even when using these inner nodes, the CPT was still too large to fill in manually. So, we developed equations for each inner node which were used by Netica to generate the CPTs. The node, IndividualCreditworthiness, used a weighted average of its parents, which in turn used logistic equations. These equations were not created to fit any existing data, and thus were not expected to perform very well.

Naive Bayes

Naive Bayes has been shown to perform well on many classification tasks despite its simplicity and strong assumptions (Domingos & Pazzani 1997). However, its performance can suffer when many of the input variables are highly correlated. Another drawback is that it is anti-causal, making the interpretation of naive Bayesian networks less intuitive.

We created two NB networks, one containing the thirteen variables used in the original network and the one shown in Figure 2, which uses all twenty variables.

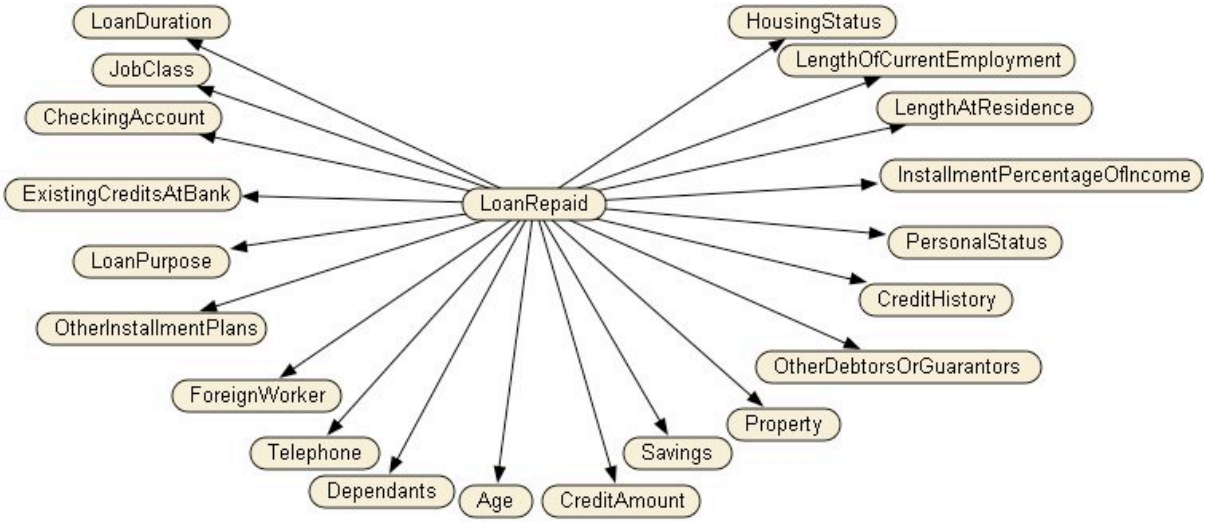

Figure 2 – The 20 node naive Bayes network (GermanCreditNBall02.neta)

Learning was done using basic count learning, which is fast and performs well on nets with no hidden variables, such as naive Bayesian networks.

TANs

Tree Augmented Naive Bayes (TANs), NB models with an additional tree structure between attributes independent of the target variable, have been shown to outperform NB models in many cases (Friedman et al., 1997). We tested the performance of different super-parent TANs, that is, TANs with a single attribute as a second parent of all other attributes.

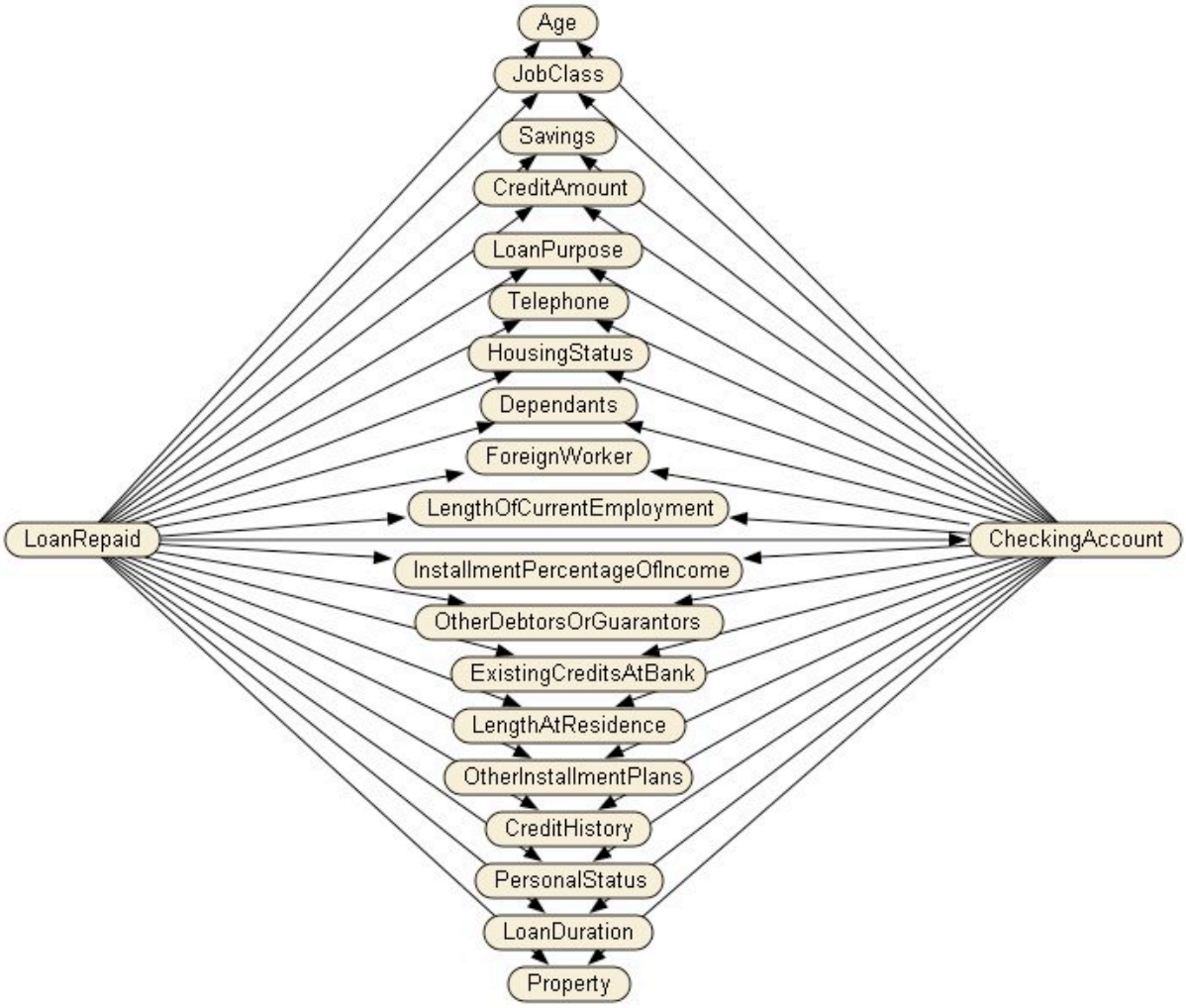

Figure 3 – A super-parent TAN created from checking account (GermanCreditNBallTAN.neta)

Using Netica's Java API, we generated super-parent TANs for every attribute in the dataset. Figure 3 shows an example of a super-parent TAN created by making CheckingAccount a parent of all other attribute nodes.

Greedy FAN

In addition to the TANs above, we found Forest Augmented Networks (FANs) using Hill Climbing search. FANs generalize TANs by allowing multiple trees between the attribute variables. The greedy FANFinder algorithm is as follows:

1. Load the naive Bayes Network.
2. Measure accuracy of the current network.
3. Repeat until accuracy no longer improves:
    a. Measure the accuracy of all possible links between leaf nodes (while maintaining the FAN structure).

b. Add the link that increases accuracy the most to the network.

4. Return the final network.

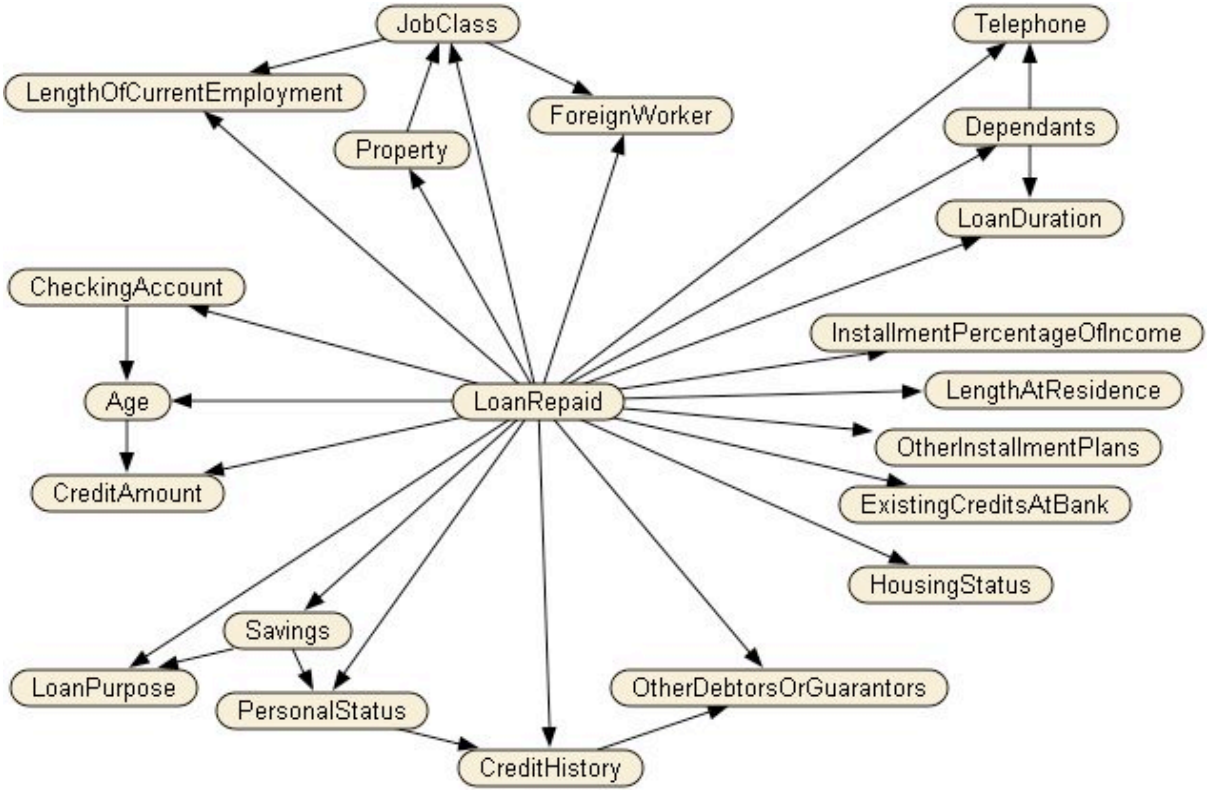

Figure 4 - The FAN created by FANFinder (GermanCreditGreedyTANcompact.neta)

Although this FAN may improve accuracy, it is not without drawbacks. The extra connections on the FAN increase the complexity of the network and there are usually no obvious causal reasons for the connections. As a result, some of the benefit of using Bayesian networks has been lost.

Furthermore, the time to create these greedy FANs can be significant. The naive algorithm used to create this FAN runs in polynomial time. Specifically, it is $O(n^3)$, where n is the number of variables. Others have done better with similar Bayesian net representations (Keogh & Pazzani 1999), so this can be improved.

Testing Methodology

To test these networks, the German credit data were used. The variables were renamed from their original names (A91, A92, A93 etc) to reflect the states they actually represented. Though this is strictly unnecessary, it is desirable because it improves the clarity of the Bayesian network.

As there were only 1000 records in the dataset, we did 90-10 splits into training and test cases, as is common in the machine learning literature. I.e., each net was parameterised using a random selection of 900 records and then tested on the remaining 100. This was repeated 30 times to obtain a reasonable sample size. The reported accuracy is the mean of these 30 tests. Statistical significance

was calculated using a z-test with standard errors of the mean. In the tables below the highest achieving network is bolded and underlined. Results that were not significantly different from the highest performing network (at the 95% confidence level, using a one-tailed test) are also bolded. Standard error is included in brackets; the 95% confidence intervals are obtained by adding and subtracting 1.96 times these standard errors. The repeated resampling performed on the single data set is know to compromise the accuracy of the results and, in particular, it reduces the power of the tests (the probability of finding statistically significant results). As this is preliminary work, we did not perform more accurate tests, such as that devised by Nadeau and Bengio (2003).

We also measured expected cost of using each network for deciding on loan applications. Costs were calculated according to the cost matrix provided with the German data, as well as two other custom matrices, as detailed below.

To calculate costs we used a Java function that entered the values of the current record into the network as findings, then used the network to calculate probability of default. If this probability was below the supplied threshold, the loan was considered granted and the cost was calculated by using the appropriate entry of the cost matrix. The data used to calculate mean costs was gathered in the same way as the accuracy data; 30 tests, using the same random 900/100 splits.

Results

TAN accuracy

TAN	**Accuracy (std err)**	TAN	**Accuracy (std err)**
LoanPurpose	70.9000 (5.55)	**Age**	72.4667 (3.15)
LoanDuration	**73.4333 (3.70)**	**PersonalStatus**	71.9333 (4.59)
CreditAmount	**74.5000 (4.39)**	**OtherDebtorsOrGuarantors**	**73.0333 (5.07)**
Savings	**73.9667 (4.72)**	**InstallmentPercentageOfIncome**	**73.1333 (3.92)**
CheckingAccount	**74.5000 (4.10)**	**Property**	72.6667 (4.17)
JobClass	72.5333 (4.52)	**OtherInstallmentPlans**	71.8333 (4.15)
LengthOfCurrentEmployment	70.7667 (4.55)	**ExistingCreditsAtBank**	**73.4333 (3.87)**
LengthAtResidence	**72.8333 (4.71)**	**Dependants**	**73.2000 (4.01)**
HousingStatus	**72.8000 (4.35)**	**Telephone**	**73.7333 (4.11)**
CreditHistory	**73.2333 (4.46)**	**ForeignWorker**	**74.3000 (4.63)**

Figure 5 – Super-parent TAN accuracy results. The best result is underlined, results that are not significantly different from the best result are bolded.

Although there are some differences in accuracy amongst the super-parent TANs, even the best ones are not significantly better than naive Bayes. For future comparisons between networks, we include only the TAN with Checking Account as the super-parent, since this performed the best.

Network Accuracy

The results for the accuracy of the other networks after testing are as follows:

Network	Accuracy % (std err)
Network1 Equations	54.9000 (2.94)
Network 1 Learned	70.9667 (3.57)
Naive Bayes (13 variables)	74.3000 (5.06)
Naive Bayes (20 variables)	**75.4667 (4.34)**
Greedy FAN	**77.1000 (4.19)**
Checking Account TAN	75.1000 (5.23)

Figure 6 - Network accuracy

Network1, which was the original handcrafted network, performed very poorly, especially when the CPTs were generated by equations. Higher accuracy can be achieved simply by classifying nobody as a defaulter. Doing so achieves 70%, since 700 out of the 1,000 records list the loan as being repaid. Even removing the equations and using Netica's EM learning capability to learn the CPTs resulted in only 70.9667% accuracy in classifying, which again is not significantly better than the default strategy of classifying everybody into the not defaulting group.

The greedy FAN achieved the highest accuracy out of all the networks we tried and is comparable to the best reported in the literature. It was significantly better than the handcrafted network, the super-parent TANs, and the naive Bayes network using only 13 variables. Despite its good performance, it was not significantly better than naive Bayes.

Expected Cost Evaluation

Mean costs were calculated from the matrices in Figure 7. The first matrix is distributed with the data. Since it gives to value to a repaid loan, identically with a refused loan that would not have been repaid, it is clearly inadequate. The second matrix fixes this problem by identifying the magnitude of repaid loans with refused loans, identifying the opportunity cost of foregoing good loans with the amount of profit they would bring. Since the cost of defaults is much larger than the lost opportunity, we think Matrix 3 is actually a far more appropriate cost matrix.

Matrix 1

	GiveLoan	
Repaid	**No**	**Yes**
No	0	5
Yes	1	0

Matrix 2

	GiveLoan	
Repaid	**No**	**Yes**
No	0	5
Yes	1	-1

Matrix 3

	GiveLoan	
Repaid	**No**	**Yes**
No	0	10
Yes	1	-1

Figure 7- The different cost matrices used

Figure 8 shows the mean cost for each matrix if everybody who has a probability of greater than 50% of not defaulting is given a loan. The greater accuracy of the Greedy FAN and naive Bayes network directly translates into lower costs.

	Matrix 1 mean Cost	Matrix 2 mean Cost	Matrix 3 mean Cost
Network1 Learned	1.3357 (0.19)	0.6650 (0.22)	1.9717 (0.41)
Naïve Bayes (13 variables)	0.9517 (0.20)	0.3353 (0.24)	1.2037 (0.44)
Naïve Bayes (all variables)	**0.8733 (0.17)**	**0.2620 (0.26)**	**1.0470 (0.36)**
Greedy FAN	**0.8517 (0.15)**	**0.2253 (0.23)**	**1.0037 (0.34)**
Checking Account TAN	**0.8663 (0.13)**	**0.2613 (0.15)**	**1.0330 (0.26)**

Figure 8 – Mean costs and standard error.

These costs are, however, not optimised. A cut-off of 50% is very arbitrary, and can be adjusted to minimise expected costs. The optimal cut-off for granting a loan was calculated as follows:

Let $Pr(Repaid)$ be the probability that the loan is repaid. We then have:

$$U(GiveLoan = Yes) = Pr(Repaid) * 1 + (1 - Pr(Repaid)) * -5$$

and

$$U(GiveLoan = No) = Pr(Repaid) * -1 + (1 - Pr(Repaid)) * 0$$

To find out when to give the loan, we want to know when

$$U(GiveLoan = Yes) > U(GiveLoan = No)$$

So,

$$Pr(Repaid) - 5(1 - Pr(Repaid)) > -Pr(Repaid)$$
$$Pr(Repaid) > 5/7$$

So the utility of giving a loan is higher than not giving the loan when the probability of repaying the loan is greater than 71.4286%.

We calculated this threshold for each cost matrix and used it to determine when to give applicants loans. This yielded the cost table shown in Figure 9. Accuracy data using the new threshold is shown in Figure 10. Accuracy for the original matrix and matrix 3 is the same because both matrices have the same optimal cutoff. Using this optimal cost threshold, the greedy FAN is no longer the best performer in predictive accuracy, although it is still not significantly worse than the best.

From Figure 9 we can see that there was no statistically significant difference between *any* of these models and the default prediction of approving credit, when assessing performance using either matrix 1 or matrix 3. Matrix 2 allows a statistically significant difference between all four of the better models and the default, however this comes at the price of an implausibly low penalty for defaulted loans. It is interesting to note that a plausible cost matrix (matrix 3) completely overturns the evaluative verdict from simple predictive accuracy: no predictor does bettern than default behavior!

	Matrix 1 mean cost	Matrix 2 mean cost	Matrix 3 mean cost
Network1 learned	0.6823 (0.05)	0.6163 (0.24)	0.6803 (0.05)
Naive Bayes (13 variables)	**0.6453 (0.11)**	**0.2047 (0.18)**	**0.6063 (0.24)**
Naive Bayes (all variables)	**0.6370 (0.13)**	**0.2090 (0.16)**	**0.5897 (0.27)**
Greedy FAN	**0.6517 (0.65)**	**0.2627 (0.19)**	**0.6190 (0.28)**
Checking Account TAN	**0.6403 (0.13)**	**0.2173 (0.15)**	**0.5963 (0.26)**

Figure 9 – Mean costs using optimal cut-off

	Matrix 1 Accuracy	Matrix 2 Accuracy	Matrix 3 Accuracy
Network1 learned	31.90 (4.57)	58.66 (8.97)	31.90 (4.57)
Naive Bayes (13 variables)	**56.66 (5.44)**	**69.50 (5.81)**	**56.66 (5.44)**
Naive Bayes (all variables)	**60.16 (4.89)**	**70.03 (4.57)**	**60.16 (4.89)**
Greedy FAN	**61.10 (4.83)**	**68.90 (4.80)**	**61.10 (4.83)**
Checking Account TAN	**63.43 (5.23)**	**63.43 (5.24)**	**63.43 (5.23)**

Figure 10 – Accuracy using optimal cut-off

Conclusion

In no test did naive Bayes significantly under-perform the other networks. Strategies that extend NB have resulted in minor gains in accuracy, usually at the cost of some simplicity. However, when we apply a plausible evaluation based upon expected cost there turns out to be no benefit to any of these models, including NB models. There nevertheless are advantages to using NB models, even for a domain well characterized by the German credit data. Simple Bayesian net models are highly intuitive for people to use; they are easily modified to incorporate new variables or subnetworks; and they are simple to tune, for example, by modifying (or optimizing) the probability threshold with which loans are approved. More complex networks for this problem have yet to show any substantial advantages which might compensate for their complexity.

References

Asuncion, A., & D. Newman (2007). UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html

Domingos, P. & Pazzani, M. (1997) On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29, 2-3, pp. 103-130.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997) Bayesian Network Classifiers. *Machine Learning* 29, 2-3, pp. 131-163.

Keogh E. & Pazzani M. (1999) Learning augmented Bayesian classifiers: a comparison of distributions-based and classification-based approaches, *Uncertainty 99: The 7th international workshop on artificial intelligence and Statistics*, pp. 225-30.

Nadeau, C. & Y. Bengio. (2003). Inference for the generalization error. *Machine Learning, 52,* 239-281.

Ratanamahatana, C. & Gunopulos, D. (2003) Feature Selection for the Naive Bayesian Classifier using Decision Trees. *Applied Artificial Intelligence,* 17, 5-6, pp. 475-487.

Appendix

Explanation of code

There are 3 Java files which were used for automating the creation and testing of nets:

BatchTest.java
TANmaker.java
GCredTester.java

First, a warning- This code was not written with reuse in mind! It desperately needs tidying up and fixing. There are undoubtedly many bugs!

The Netica Java API is required. It can be tricky to get working; check the Netica site for help.

BatchTest.java
This file contains methods for testing networks. It can learn CPTs from data and test against data.
The method, batchTest does both of these things and returns and prints the result. The code should be able to be used on other networks and data sets.
The other useful method is testCustomCutoff, which does what batchTest does, but you can supply a threshold for the yes/no decision (giving a loan in this case).
Also contains some util methods for calculating variance, stat sig, etc.

Read the comments in the file for details on each method.

TANmaker.java
This code is used to automate the creation of super-parent TANs. It is specific to my network, GermanCreditNBall.neta.

It also has code for creating a greedy FAN in makeBestTAN(), again, specific to my network. This method takes several hours to run.

GcredTester.java
Uses TANmaker and BatchTester to output results. Specific to German Credit data and my nets. Try to avoid using this, its very messy and I would just write new tests in there as I needed them.

Other files

The code makes use of these other files:

ALLout.cas - Netica style case file. A translation of the german credit data file into more descriptive language. BNs are learnt from and tested against this data.

GermanCreditGreedyTAN.net - the greedy FAN drawn in netica. It was created algorithmically and then redrawn in Netica so I could have a picture. Also can now be tested again without having to recreate it in java.

GermanCreditNB02.java - NB network with 13 vars

GermanCreditNBall01.java - NB network with all 20 vars

GermanCreditWithEquations07.neta - Original causal network