UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

LEARNING ENSEMBLES OF BAYESIAN NETWORK STRUCTURES USING RANDOM FOREST TECHNIQUES

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

CHRISTOPHER M. UTZ
Norman, Oklahoma
2010

LEARNING ENSEMBLES OF BAYESIAN NETWORK STRUCTURES
USING RANDOM FOREST TECHNIQUES

A THESIS APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY

Dr. Amy McGovern

Dr. Deborah A. Trytten

Dr. Susan E. Walden

Dedication

This thesis is dedicated to my fiancée Kathryn, whose endless love, support, and encouragement has kept me motivated even when there was no light at the end of the tunnel.

Acknowledgments

The work presented in Chapter 5 entitled "Case Study: Analyzing Student Data to Predict Retention" is based upon work supported by the National Science Foundation's Directorate of Undergraduate Education's STEM Talent Expansion Program Grant No. DUE-0431642. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

We would like to thank the following collaborators who have contributed to data collection: Deborah A. Trytten, Susan E. Walden, Randa L. Shehab, Teri J. Murphy, Teri Reed-Rhoads, Jeanette Davidson, Cindy E. Foor, Elaine Seymour, and Larry Schuman.

We would also like to thank Adrianna Kruger and Sean Williams for their help with cleaning and normalizing the educational data used in this work. Their work in Machine Learning (CS4033) was instrumental to our ability to analyze the data used in our engineering education case study.

Lastly, we wish to thank Andrew Fast and the Knowledge Discovery Laboratory in the Department of Computer Science at the University of Massachusetts Amherst. This work relies heavily on their PowerBayes 1.0 software package for Bayesian Network structure learning[1].

[1] `http://kdl.cs.umass.edu/powerbayes/`

Table of Contents

List Of Tables

List Of Figures

Abstract

We introduce an ensembled classifier of Bayesian Networks, EBNs, that automatically learns the structure of a set of Bayesian Networks. Our experiments demonstrate that EBNs have many of the benefits of Random Forests when compared with the classification performance of traditional Bayesian Networks. Specifically, we highlight the success of EBNs as a model for classification when trained on small datasets. Similar to Random Forests, and contrary to traditional Bayesian Networks, EBNs do not overfit. Additionally, EBNs outperform or perform as well as traditional Bayesian Networks when tasked with instance classification. EBNs also provide a measure of variable importance and can be generated efficiently on a variety of datasets. Using datasets from the standard Bayesian Network Repository we show that EBNs perform equal to or better than traditional Bayesian Networks when used for classification. We analyze a case study in engineering education to illustrate the performance benefits of EBNs on small datasets and the ability of EBNs to identify important variables within datasets. We focus on predicting retention outcomes of engineering students and prioritizing factors related to their retention in engineering education.

Chapter 1

Introduction

Bayesian Networks are probabilistic graphical models that represent variables as the nodes in a directed acyclic graph and the conditional independencies between the variables as edges (Pearl 2000, 1988). Bayesian Networks have become a powerful model for visually representing the relationships within datasets and for classification. However, when trained on small datasets, Bayesian Network structures tend to overfit the training data (Elidan and Gould 2008; Goldenberg and Moore 2004; Grossman and Domingos 2004; McGovern et al. 2008b). Random Forests developed by Breiman (2001) do not overfit. Recent work (Supinie et al. 2009; McGovern et al. 2010) has shown success when applying Random Forests techniques to classification algorithms other than the C4.5 decision tree algorithm traditionally employed by Random Forests. In this thesis, we apply the same concepts that make Random Forests successful to the task of classification in the realm of Bayesian Networks.

We begin by presenting background information in the domain of Bayesian Networks. This discussion includes related work in inference, parameter estimation and structure learning algorithms. Following the discussion of Bayesian Networks, the creation and classification algorithms for Random Forests are detailed. Subsequently, we investigate the efficacy of another algorithm that relies on Random Forest techniques, Spatiotemporal Relational Random Forests. Chapter 3 outlines our motivation for developing EBNs, as well as methods for EBN construction, prediction, variable importance identification, and visualization.

In Chapter 4, we provide an empirical evaluation of these new methods using several well-known datasets from the Bayesian Network Repository. This section also explores the algorithm parameters, and their implications on the EBNs classification performance. We also investigate the effect of training set size on EBN's ability to classify data instances correctly. In Chapter 5, we supplement our empirical evaluation with a case study in engineering education. This case study builds an EBN that predicts factors contributing to retention of minority students in the College of

Engineering at the University of Oklahoma. In conclusion, Chapter 6 summarizes our findings and suggests opportunities for future work.

Chapter 2

Background

2.1 Bayesian Networks

Given a set of variables, V, a full joint probability table can be constructed to represent probabilities over all possible outcomes for the variables in V. Inference can be performed on the table in order to ask questions of the dataset. In the case where observed values for each variable in V exists, inference becomes a basic table lookup. In the case where some variables may be unknown, inference can still be performed by calculating the marginal probability for the variable in question. The marginal probability for an event A given B is the probability of event A regardless of the value B takes. Equation (2.1) shows the probability of event A when the value of B can be $true$ or $false$.

$$p(A) = p(A, B = true) + p(A, B = false) \tag{2.1}$$

This calculation produces the $p(A)$ when the value of B is unknown. Inference can be performed on the table of multivariate distributions, but its size grows rapidly as a function of the number of variables in V and the cardinality of those variables. Equation 2.2 gives the number of rows required for a system composed of the variables in V.

$$\#ofrows = \prod_{v \in V} cardinality(v) \tag{2.2}$$

As seen in Table 2.1, which gives the full joint probability distributions for the Slippery domain described below, even for trivial datasets the size grows rapidly. This increase in growth becomes intractable for any dataset with more than a few variables. The space required for representing the full joint probability distribution for a set of variables is exponential with the number of variables and the variables' cardinalities.

Bayesian Networks (Pearl 2000, 1988) use conditional independence and Bayes' Rule to efficiently represent full joint probability distributions like those seen in Table 2.1. The compact representation of Bayesian Networks allows for inference

P(Wet)	Sprinkler	Rain	Season	Slippery
1.00	On	Yes	Su	Yes
0.99	On	Yes	Su	No
1.00	On	Yes	Sp	Yes
0.97	On	Yes	Sp	No
1.00	On	Yes	Fa	Yes
0.99	On	Yes	Fa	No
1.00	On	Yes	W	Yes
0.96	On	Yes	W	No
0.99	On	No	Su	Yes
0.58	On	No	Su	No
0.99	On	No	Sp	Yes
0.58	On	No	Sp	No
0.99	On	No	Fa	Yes
0.56	On	No	Fa	No
1.00	On	No	W	Yes
0.51	On	No	W	No
0.94	Off	Yes	Su	Yes
0.17	Off	Yes	Su	No
0.93	Off	Yes	Sp	Yes
0.13	Off	Yes	Sp	No
0.93	Off	Yes	Fa	Yes
0.13	Off	Yes	Fa	No
0.94	Off	Yes	W	Yes
0.13	Off	Yes	W	No
0.25	Off	No	Su	Yes
0.00	Off	No	Su	No
0.26	Off	No	Sp	Yes
0.00	Off	No	Sp	No
0.27	Off	No	Fa	Yes
0.00	Off	No	Fa	No
0.25	Off	No	W	Yes
0.00	Off	No	W	No

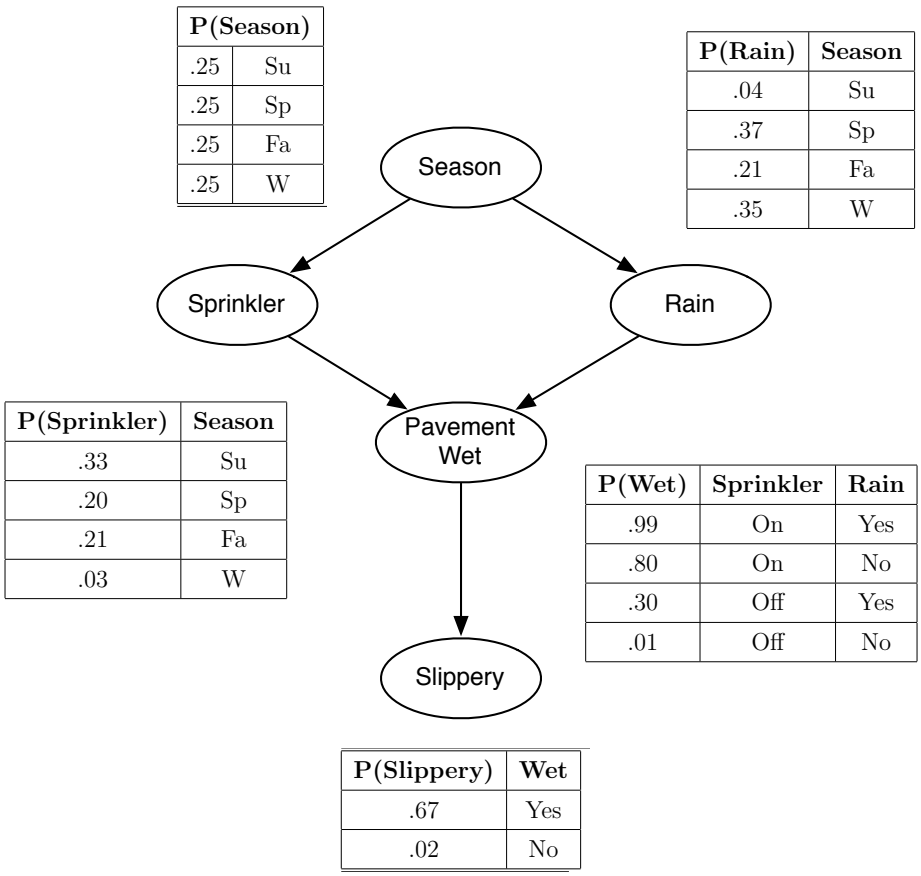
Table 2.1: Probability distributions for $p(Wet|Sprinkler, Rain, Season, Slippery)$

to be performed on datasets with large numbers of variables. Bayesian Networks also provide a graphical representation of the relationships between dataset variables. A Bayesian Network is a directed acyclic graph where each node represents a variable and edges represent relationships between variables. For each node, a conditional probability table (CPT) describes the probability distribution for the node's values given its parents. Bayes' Rule, seen in Equation (2.3), and the independence condition maintain that a variable is conditionally independent of other variables in the network given the values of its parents, children, and children's parents.

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)} \tag{2.3}$$

Each variable's conditional independence ensures that the node's CPT is of manageable size. Inference can then be performed on a variable (node) knowing only the variables required to perform a lookup based on that node's CPT. If an observed variable is unknown, inference can still be performed by recursively calculating the marginal probabilities for the missing variables.

An example Bayesian Network is given in Figure 2.1 (Pearl 2000). This network represents a system containing five variables. Each variable is encoded as one of the nodes in the graph. The variables Sprinkler On, Rain, Pavement Wet, and Slippery can all take values of Yes or No, and have a cardinality of two. Conversely the variable Season has a cardinality of four and can have a value of $Summer$, $Spring$, $Fall$, or $Winter$. Edges in the network encode relationships between variables in the dataset. The direct edges from Season to both Sprinkler On and Rain encode our understanding that the value of Season has a direct impact on the outcome of Sprinkler On and Rain. The absence of a link from Season to Slippery encodes our understanding that the influence of season on slipperiness is mediated by other factors. This coincides with the independence condition which says, knowing Wet makes Slippery independent of Sprinkler On, Season, and Rain. At each node in the network, the probability distribution for the variable's outcome is encoded as a function of its parents. This encoding is represented as a Conditional Probability Table, CPT. Each CPT is smaller than the table encoding the full joint probability distribution for the same system would be. In our example, the compact representation using Bayesian Networks, Figure 2.1, requires slightly more than half of the rows required to represent the full joint probability distribution in Table 2.1.



P(Season)	
.25	Su
.25	Sp
.25	Fa
.25	W

P(Rain)	Season
.04	Su
.37	Sp
.21	Fa
.35	W

P(Sprinkler)	Season
.33	Su
.20	Sp
.21	Fa
.03	W

P(Wet)	Sprinkler	Rain
.99	On	Yes
.80	On	No
.30	Off	Yes
.01	Off	No

P(Slippery)	Wet
.67	Yes
.02	No

Figure 2.1: Example Bayesian network for Slippery dataset

2.1.1 Inference

Given the network in Figure 2.1, questions such as "Will the pavement be wet if it is not raining but the sprinkler is on?" or "If it is raining will the pavement be wet?", can be asked. These questions are known as inference and can be a powerful tool for prediction. To answer the first question we must compute the result of equation (2.4).

$$P(Wet = Yes|Sprinkler = On, Rain = No) \tag{2.4}$$

Because the two variables that Wet is dependent upon are observed, we can perform a simple lookup in Wet's CPT. Using Figure 2.1 the solution can be found in row two of Wet's CPT. This row indicates there is an 80% chance that the pavement is wet, given the sprinkler is on but it is not raining.

In the second question, all of the necessary variables are not observed, but we can answer the question using Bayes' rule, the definition of conditional probability, and our model. As shown in equation (2.5) we marginalize over the CPT for Pavement Wet, given the probabilities for all possible values of the missing variables.

$$P(Wet = Y|Rain = Y) = \frac{P(Rain = Y, Wet = Y)}{P(Rain = Y)} \tag{2.5}$$

$$\begin{aligned} P(Rain = Y, Wet = Y) &= \sum_{sp \in \{On,Off\}} P(Rain = Y, Wet = Y, Sprinkler = sp) \\ &= \sum_{sp \in \{On,Off\}} P(Wet = Y|Rain = Y, Sprinkler = sp) * \\ &\quad P(Rain = Y)P(Sprinkler = sp) \\ P(Rain = Y) &= \sum_{s \in \{Su,Sp,Fa,W\}} P(Rain = Y, Season = s) \\ &= \sum_{s \in \{Su,Sp,Fa,W\}} P(Rain = Y|Season = s) * \\ &\quad P(Season = s) \end{aligned}$$

Following the above equations, we must sum over the first and third rows in Wet's CPT multiplying by the probability an instance exists for the criteria in each row. Using equation (2.5) and the CPTs in Figure 2.1, we can input the variables and find that:

$$
\begin{aligned}
P(Wet = Y | Rain = Y) &= \frac{P(Rain = Y, Wet = Y)}{P(Rain = Y)} \\
&= \frac{0.99 * 0.24 * 0.19 + 0.30 * 0.24 * 0.81}{0.25 * (0.04 + 0.37 + 0.21 + 0.35)} \\
&= 0.43
\end{aligned}
$$

Provided any Bayesian Network and an inference query, the probability can be calculated using the recursive algorithm in Table 2.2.

Given a Bayesian Network BN, a variable to predict V, and a set of evidence E, the following procedure can be applied. If V has no parents in BN, return the distribution from $V's$ CPT. This case is the first of two base cases in the recursive algorithm, and is encountered following an inference query such as "What is the probability it is summer?" In our example, the result of this inference query is 0.25 and is found with a table lookup in Season's CPT. The second base case occurs when V has parents, and all variables represented by these parents are included in the evidence set E. A sample inference query for this case would be "What is the probability the pavement is wet if it is raining and the sprinkler is on?" Again, this is a table lookup in V's CPT for the row that matches E. For our example query, the result 0.99 is found in the first row of Pavement Wet's CPT. The recursive case is followed for a query such as "What is the probability the pavement is wet given the sprinkler is on and it is winter?" In this case all necessary evidence is not provided. We must marginalize over the possible rows from V's CPT, querying parents for the probability we are in each row. In our query, the rows from Pavement Wet's CPT that match the data provided in E, are rows one and two. Because Rain is a parent of Wet, for which we are not provided observed data, we must determine the probability of Rain. We perform inference recursively on Rain, given the sprinkler is on and it is winter. Using the recursive case from the algorithm in Table 2.2 we find that $P(Wet|Sprinkler = On, Season = W) = 0.87$.

Table 2.2: Algorithm for inference queries

inference(Bayesian Network BN, variable to predict V, evidence E)

Let N be the node in BN representing V
If N has no parents //Base case 1
    Return lookupVarInCPT(N, V, $\emptyset$)
Let PV be the set of variables for all parents of V
If $PV \subset E$ //Base case 2
    Return lookupVarInCPT(N, V, PV)
Let $PInf = [\,]$
For each parent P of N
    $PInf[P]$=inference(BN, P, E)
Let $result = 0$
For each row R in N.CPT
    Let $PRow = 1$
    For each parent P of N
        $PRow = PRow$ * $PInf[P]$
    $result = result + PRow$ * lookupVarInCPT(N, V, E)
Return $result$

lookupVarInCPT(node N, variable V, observed variables E)

Let $table$ be N.CPT
Return $table[E][V]$

2.1.2 Learning Bayesian Networks

As described in Section 2.1, a Bayesian Network consists of a Bayesian Network structure and a set of parameters that populate the CPT for each node in the structure. Given a Bayesian Network structure with CPTs that have not been populated, it is possible to learn the parameters for each CPT from a dataset. When provided with only the structural component of a Bayesian Network, this parameter learning is required before an inference query can be performed. It is possible to learn both the CPTs parameters and the Bayesian Network structure for a provided dataset.

Learning a full Bayesian Network, which includes learning both the parameters and structure, is multistep process. Thinking logically, the structure for the Bayesian Network is learned first and the parameters for the CPTs are then populated. However, in practice, algorithms for learning Bayesian Network structures include a component that must simultaneously learn the values for the CPTs. Both parameter and structure learning for Bayesian Networks are discussed below.

2.1.2.1 Parameter Learning

Given a directed acyclic graph (DAG) representing the structural component of a Bayesian Network, the parameters for each node's CPT must be learned prior to performing inference. For each node in the structure, the CPT must be populated with the distributions for the variable, given the variables represented by the node's parents. A generic algorithm for this process can be found in Table 2.3. This algorithm requires all variables in the network to be present in the dataset. In certain situations all variables required for the network structure may not be present in the dataset.

For an example of this situation, consider a Bayesian Network that has been learned by a group of researchers in the domain of engineering education whose research has been shown to successfully model the factors leading to retention in engineering programs. Because of their success, researchers from other institutions wish to model their datasets using the developed Bayesian Network. In order to perform inference queries on the Bayesian Network using their institutions data they must first learn the parameters for the CPTs from their data. Although they may have collected some of the same data that the original researches used to develop the Bayesian Network, they may not have data for all of the variables in the network structure. This results in a dataset that is missing values required for learning the CPTs' parameters. Even with these missing values, inference can still be performed after estimating the unknown variables.

Table 2.3: Parameter learning algorithm

learnParameters(Bayesian Network BN, dataset D)
    For each node N in BN
        Let V be the variable represented by N
        Let PS be the set of variables represented by N's parents
        N.CPT = generateCPT(V, PS, D)
    Return BN
generateCPT(variable V, variables PS, dataset D)
    Let CPT = [ ]
    For each permutation of parent values Row in PS
        Let *count* be the number of instances in D that match Row
            for each possible value v in V
                Let *portion* = number of instances in D that match Row and $V = v$
                $CPT[Row][v]$=*portion*/*count*
    Return CPT

Unknown variables are referred to as latent or hidden nodes and their CPTs can be estimated using the dataset. The expectation maximization (EM) algorithm (Dempster et al. 1977) can be used for populating the CPTs of a network's latent nodes. Shown in Table 2.4, the EM algorithm involves alternating between two steps. The first step calculates the expected value for the unobserved variables given the observed data. In the second step, the network is updated to maximize likelihood, assuming the values calculated in the first step are correct. The algorithm continues alternating these steps until the CPTs for the latent nodes converge. Dempster et al. (1977) has proven convergence of the EM algorithm and has shown empirically that the EM algorithm often converges quickly. We have empirically verified these properties to be true in our experiments as well.

2.1.2.2 Structure Learning

Structure learning for Bayesian Networks is an unsolved problem. The large search space and tendency for learned models to overfit make structure learning complicated.

Table 2.4: Parameter learning for networks with latent nodes

learnParametersWithLatentNodes(Bayesian Network BN, dataset D)

Let HD be dataset containing the values for the latent nodes

For each $latentNode$ in BN

$HD[latentNode]$=randomDistribution($latentNode$)

While likelihood increases

Let $FullData = D \cup HD$

learnParameters(BN, $FullData$)

For each $latentNode$ in BN

HD = updateHiddenNode(BN, $latentNode$, HD)

learnParameters(BN, $FullData$)

Return BN

updateHiddenNode(Bayesian Network BN, node $latentNode$, dataset HD)

For instance I in HD

Let criteria C = markovBlanket($latentNode$)

Let observed variables $O = HD[I][C]$

$HD[I][C]$=inference(BN, $latentNode$, C)

Return HD

However, several heuristic and statistical based algorithms have been developed for structure learning. These methods fit into two main categories. The first group of algorithms are search and score based. This approach involves searching over possible Bayesian Network structures in an attempt to maximize a scoring function. Scoring functions are generally a variation on likelihood penalized to discourage overly complex network structures. Akaike Information Criterion (AIC) (Akaike 1978), Bayesian Information Criterion (BIC) (Schwarz 1978), and Bayesian Dirichlet equivalence uniform (BDeu) (Buntine 1991) are three common penalized likelihood metrics used in search and score based algorithms. Although model likelihood is maximized, the search problem grows exponentially with the size of the dataset. Due to the large size of the problem space, search algorithms are generally coupled with heuristics that limit the size of the problem e.g.(Buntine 1996; Heckerman et al. 1995; Cooper and

Dietterich 1992; McGovern et al. 2008b). The search and score approach is highly flexible, but has a tendency to incorrectly reproduce the generating structure because conditional independence relationships are not enforced.

The second group of algorithms are constraint based structure learning algorithms (Pearl 2000; Cheng et al. 2002; Spirtes et al. 2000). Constraint based algorithms use local statistical tests to identify a dependency model representing the conditional independencies present in the dataset. The final structure selected must adhere to, and is limited by, the conditional independencies contained in the learned dependency model. Constraint based algorithms generally have smaller likelihood scores than search and score algorithms; however, constraint based algorithms are more efficient and create structures more accurately representing the conditional independencies of the original dataset.

Recently, a hybrid approach for structure learning algorithms has been identified. To generate Bayesian Network structures, hybrid algorithms rely on the strengths of both search and score and constraint based algorithms. For our experiments we rely on a variation of the canonical hybrid approach Min - Max Hill Climbing, MMHC, (Tsamardinos et al. 2006). The first step of MMHC uses constraint identification to create an undirected skeleton graph. Each pair of dependent variables is linked by an edge in the skeleton. The second stage uses this skeleton to constrain a greedy hill climbing search for the final structure. In this search, states are Bayesian Network structures, and operators are the addition, removal, and redirection of edges. MMHC uses BDeu as the scoring metric. The Bayesian Network output by the MMHC algorithm meets the dependency constraints of the skeleton, and has been maximized locally in terms of BDeu score. Fast et al. (2008) present a variation on MMHC that uses the POWER correction (Fast 2009) to improve the skeleton generation step. Provided as part of the PowerBayes[1] package (Fast 2009), "mmhc-cv-power" is the structure learning algorithm we use throughout this thesis.

With the success of ensemble approaches in other facets of Machine Learning, new ensemble based algorithms for Bayesian Network structure learning are also being developed. Liu et al. (2007) present an ensemble technique for generating a Bayesian Network structure from an ensemble of learned structures. Unlike the method we present, which is more closely related to Random Forests, their research constructs a single Bayesian Network structure from an ensemble of learned structures. This

[1] `http://kdl.cs.umass.edu/powerbayes/`

difference in output is a key distinction between the methods introduced by Liu et al. (2007) and the methods we introduce in this thesis. Their methods introduce a new procedure for sampling the data used in the generation of each component network, known as Root Node Sampling. Root Node Sampling is used to ensure that conditional independencies within the sampled data match the conditional independencies of the original dataset. Additionally, they present a method for aggregating component Bayesian Networks into one ensembled network structure. The consistency enforced by Root Node Sampling ensures the mathematical requirements of Bayesian Networks are upheld when combined into a final network structure.

2.2 Random Forests

Random Forests are ensemble classifiers developed by Breiman (2001). Random Forests are made up of a collection of individual decision trees learned independently from a subset of the training data. Given an instance for classification, the Random Forests allows each component tree to vote on a class. The class receiving the majority of votes is output as the result of classification using Random Forests. For decision tree construction of each tree T^i, Random Forests use a modified C4.5 decision tree algorithm without pruning.

The C4.5 algorithm (Quinlan 1993) generates decision trees which are used for instance classification. Two key features of C4.5 trees are their ability to handle continuous variables and missing values. The algorithm for generating C4.5 decision trees handles continuous variables by first establishing a threshold. The continuous variable is then categorized by the values above the threshold and the values less than or equal to the threshold. Missing values are allowed by ignoring attributes with missing values in the calculation of information gain.

The algorithm for generating C4.5 decision trees takes a set of training instances and produces a classification model. For each attribute a, the normalized information gain is calculated and the attribute with the highest information gain is selected as the decision node n. For each split in n, the algorithm is applied recursively by partitioning the training instances by their value n. The recursion is terminated when all instances provided are in the same class. At this point a leaf node is created specifying the classification value for the branch of the tree.

Table 2.5: Algorithm for Random Forest generation

generateForest(dataset D, int features, int components)

Let $forest$=[ ]

While $i < components$

D^i = sampleWithReplacement(D.instances, sizeof(D.instances))

$forests[i]$ = learnTree($null$,D^i, $features$)

Return $forest$

learnTree(Tree T, dataset D, int $features$)

Let $vars$ = sampleWithoutReplacement($D.variables$, $features$)

Let attribute a = findAttributeWithHighestGain($vars$)

Let datasets $splits$ = split(D, a)

If $splits$.size = 1 //Base case

Return T.addLeaf(a)

T.addNode(a)

For each $split$ in $splits$

T.addChildTree(learnTree(T, $split$, $features$)

Return T

Using the C4.5 algorithm, each tree in the forest is grown on a set of instances selected randomly with replacement from the dataset. In addition, at each split the tree construction algorithm considers only a subset of variables for node selection. The algorithm outlining forest construction can be seen in Table 2.5. This approach has been shown to provide numerous benefits over traditional decision trees. Random Forests do not overfit. This property is particularly useful for classifiers built from small training sets, because traditional methods require careful consideration for termination before overfitting. Random Forests also provide methods to balance error in datasets with rare events, and offer insight into which variables are important for classification. In addition, the algorithm for constructing Random Forests is forgiving with respect to parameter selection. These beneficial features have established Random Forests as a successful ensemble classifier in machine learning (Bosch et al. 2007; Fislason et al. 2006; Segal 2004).

With the success of Random Forests, researchers have incorporated principals of Random Forests into other facets of machine learning. Supinie et al. (2009) extended traditional Random Forests to work with spatiotemporal relational probability trees, SRPTs, (McGovern et al. 2008a) for classification problems in which both space and time are critical elements of classification. Supinie et al. (2009) illustrate that Random Forest techniques can be successfully applied to classification algorithms other than the traditional C4.5 decision trees to increase performance. Many principals of traditional Random Forests hold true when used with other algorithms. An empirical verification that performance is asymptotic as both the size of the forest and the amount of training data increases (Supinie et al. 2009) suggests that Random Forests techniques can improve performance of Bayesian Network classification on small datasets.

The background presented in Chapter 2 identifies key features of two different models commonly used in machine learning. The strengths and weaknesses of both Random Forests, an ensemble tree based classifier, and Bayesian Networks, a probabilistic graphical model, have been outlined. The next chapter presents Ensembled Bayesian Networks, a hybrid machine learning technique that builds on the strengths of the Random Forests and Bayesian Networks. We introduce algorithms for model construction and methods for classifying instances. Additionally, we provide techniques for visualizing relationships within datasets and for identifying variables most important to the task of classification.

Chapter 3

Ensembled Bayesian Networks

Our motivation for developing Ensembled Bayesian Networks, EBNs, stems from our past work using Machine Learning techniques to analyze student data in engineering education (McGovern et al. 2008b). Stricken by unbalanced class data, complex interactions between variables, and a small number of instances from which to train models, our past work left room for improvement. Our domain experts desired a model that could accurately classify instances from unbalanced data, but also provide visualization of factors that influence the outcome of the classification task. From these factors and our past experiences, we developed the following criteria to be used in the search for a new algorithm.

- The model produced should be able to classify instances from unbalanced data successfully.
- The algorithm should construct a model with good classification performance when trained on small datasets. A dataset's size is a function of not only the number of observed instances but also the number of variables that represent each instance. As the number of variables, or features, used to represent instances in a dataset increases, the number of instances required to accurately model the dataset also increases. This fact makes modeling complex datasets, those with large numbers of features, difficult to model. Training models on complex datasets with too few instances leads to overfitting and a performance degradation on testing sets. We desire a model that is not subject to this phenomenon even when trained from small datasets.
- The generated model should provide an understanding of the interaction between variables in a dataset. The resulting model should identify not only the relationships between variables but which variables are most important for classification.

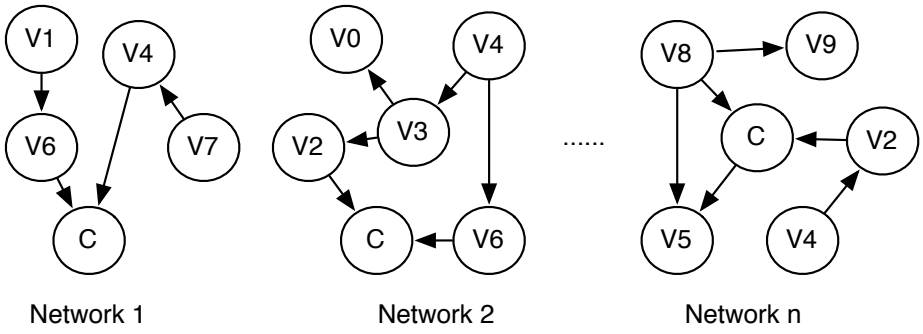


Figure 3.1: Hypothetical EBN of size n

The desire for domain experts to visualize the relationships within the data was a key factor in the decision to use Bayesian Networks in our first work (McGovern et al. 2008b). As discussed in Section 2.1, the ability for Bayesian Networks to represent the relationships within complex datasets and provide methods for inference make them a model satisfying some of the above criteria. Random Forests, discussed in Section 2.2, is another attractive algorithm that meets our criteria; specifically, its ability to not overfit and provide a measure of variable importance. Given Bayesian Networks and Random Forests, two models each having strengths and meeting portions of our criteria, the next step is to combine the two algorithms to form a new method that satisfies the requirements of our research. Our marriage of Bayesian Networks and Random Forests has resulted in a new ensemble based Bayesian Network classifier.

Ensembled Bayesian Networks (EBNs) consist of several independently learned Bayesian Network structures that are ensembled together to form one classifier. Unlike traditional Bayesian Networks, component networks of an EBN are learned from a subset of the available training data. Borrowing from component construction in Random Forests (Breiman 2001), each component network is learned using a subset of the training data and a subset of the available variables. The component construction results in several unique models learned from different parts of the original dataset. Since the construction of each component network is independent, construction can be executed in parallel and then aggregated to form an EBN. An example EBN for the task of classifying a variable C given other observed variables $V0$ - $V9$ is shown in Figure 3.1. Continuing the example, an instance can then be provided as input for the EBN, and a probability the instance is in a particular class of C is subsequently produced. The remainder of Chapter 3 will provide the necessary details for

understanding EBN construction and instance classification. We will also present a method for variable importance based largely on the algorithm for determining variable importance in Random Forests. Lastly, we will introduce the idea of a composite structure used to identify the relationships that exist across all component network structures.

3.1 EBN Construction

The procedure for developing EBNs, given in Table 3.1, is based largely on the algorithm for generating Random Forests. The EBN construction algorithm takes as input: a training dataset D, a Bayesian Network structure learning algorithm A, the EBN size n, the component size m, and a classification variable V. As a result of this input, the algorithm returns an EBN that can be used to classify instances for variable V. For each component of the EBN, a random subset of variables, whose size is much smaller than the number of variables in D, is selected. A training set whose size is equal to that of the original dataset size is selected at random by sampling with replacement from D. The Bayesian Network structure learning algorithm, A, is then used to create a Bayesian Network using the sampled subset of variables and instances. This procedure is repeated until n component structures have been generated. The result is an EBN consisting of n Bayesian Network structures.

Table 3.1: Algorithm for constructing EBNs

learnEBN(dataset D, algorithm A, int n, int m, variable V)

$Components$=[ ]
While numberOfComponents $< n$
D^i = sampleWithReplacement(D.instances, sizeof(D.instances))
V^i = sampleWithoutReplacement($D.variables$, $m-1$) + V
$Components[i] = A(D^i, V^i)$
return $Components$

3.1.1 Variable Selection

As seen in Table 3.1, each component of an EBN is constructed on a subset of variables that have been sampled randomly from the dataset's available variables. Since inference is performed on all components of the EBN for prediction, each network must contain the variable about which inference questions will be asked. This requirement has the unfortunate side effect of necessitating that a new model be constructed for each variable for which instances will be classified. In practice, we have found this requirement is not a significant issue. The dataset for our case study, focusing on retention prediction, along with several datasets from the Bayesian Network Repository have only a small subset of the variables for which classification is desired. In addition, as the size of the component networks is limited, runtime of the underlying Bayesian Network structure learning algorithm decreases. Quantifying this improvement relies heavily on the underlying structure learning algorithm; however, due to the exponential component of many structure learning algorithms, this improvement can be significant.

3.1.2 Training Set Selection

Similar to Random Forests, each component of an Ensembled Bayesian Network is learned on not only a subset of the dataset's variables but also on a subset of the dataset's instances. The training set used for learning each component is sampled with replacement from the original dataset. This new instance set, which is the same size as the original training set, is referred to as the in-bag instances. This in-bag set becomes the instances provided to the structure learning algorithm. It has been shown this in-bag set contains approximately two-thirds of the instances found in the original training set (Breiman 1996). The remaining one-third of the instances not selected form the out-of-bag set. This set of instances may be used as a validation set, for error analysis, or to calculate a measure for variable importance as shown in Section 3.3.

3.2 Prediction

Once constructed, an EBN can be used to predict the probability that an instance has a particular value of the classification variable. The algorithm for classification

takes as input: an EBN, a Bayesian Network inference algorithm A, the value of the classification variable to predict, c, and the set of observed evidence for the instance to be classified, e. As a result, the algorithm produces a probability that the instance's classification is the value supplied to the inference algorithm. This prediction, combined with a cutoff value, can then be used for classification of the test instance. The algorithm for prediction can be seen in Table 3.2. For each component

Table 3.2: Algorithm for prediction using EBNs

predict(EBN E, algorithm A, evidence e, classification variable c)

sum =0

For each *component* in E.components

sum+ =A(*component*, e, c)

Return sum / E.size

network of EBN, perform inference by using the inference algorithm A given e and c. To compute the value returned by the EBN prediction algorithm, average the individual component predictions. This equal weighting of the components' results, is the method employed by Random Forests for ensembling each decision tree's results. We will use this method of combination throughout the remainder of our work. It is feasible that individual component predictions could be aggregated in other ways; however, this investigation is outside the scope of this thesis and will be left for future investigation.

3.3 Variable Importance

As briefly mentioned in Section 3.1.2, out-of-bag instances can be used to measure variable importance in EBNs. The procedure provides a confidence score which can be used to rank the included attributes by their effect on the EBN's classification decision. The method for producing variable importance scores is straightforward. For each component network in the EBN, classify the out-of-bag instances and then count the number of instances classified correctly. To calculate the importance score for variable v permute the value of v in the out-of-bag instances. Next, reclassify

the out-of-bag instances and count the number of instances classified correctly. The mean of the difference of these two counts across all component networks is variable v's raw importance score. Provided with a variable's raw importance score, we can calculate a standard score, z-score, for v. The z-score is calculated by dividing the raw score by the standard error. Assuming normality, the z-score can be used to compute a confidence value and determine which variables are most important in terms of classification performance (Breiman 2001).

Variable importance scores, calculated using the method detailed above, may be used to eliminate unnecessary variables from future runs. The removal of unnecessary variables results in training data that contains less noise. In addition, with the extra variables removed fewer training instances are required to produce a well performing classification model. An EBN can first be generated from the full dataset and the variable importance scores can be calculated for all the included variables. The variables identified as having low importance scores are filtered from the dataset. The resulting data can then be used to construct a new model, EBN or other classification model, that includes only the most important variables as related to the task of classification.

3.4 Composite Structure

Since EBNs are built from Bayesian Networks, inspection of an individual component yields information about the relationships among the component's variables. Each of the networks in Figure 3.1 is an example component network from an EBN. Looking at the edges present in this component, we can identify several relationships that exist between the variables. While an analysis provides some indication of the relationships between variables, not all of the necessary data is gleaned. Recall that each component of an EBN is generated on a random subset of instances and possible variables. Therefore, in order to understand the relationships within a dataset, every component of the Ensembled Bayesian Network must be investigated. A full investigation of the EBN's component networks is not feasible. In order to obtain an aggregate view of the relationships between variables using our generated EBN, we utilize the concept of composite structures introduced in our first work (McGovern et al. 2008b).

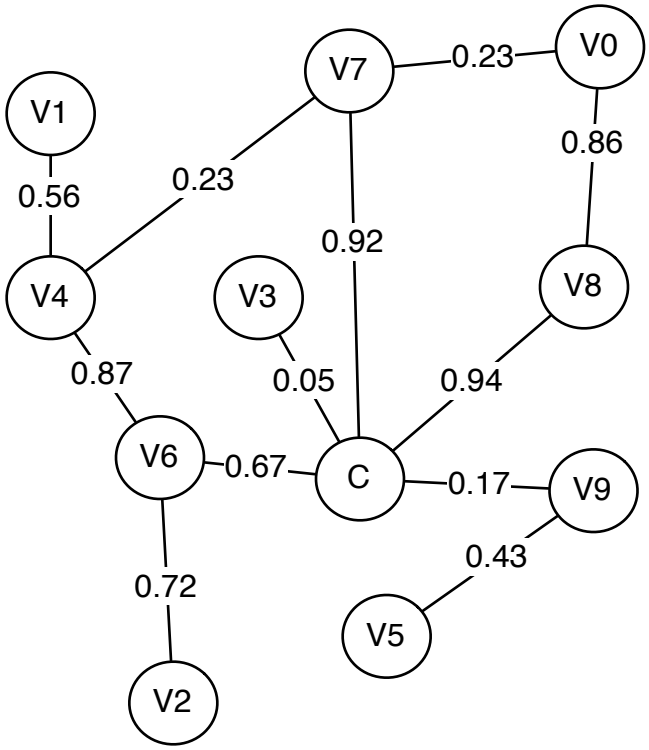


Figure 3.2: Hypothetical composite for an EBN

A composite structure is an undirected graph whose edges are an aggregation of edges in some number of generating structures. It is important to recognize that because the composite structure is an undirected graph, edges in the structure represent correlation between variables rather than causation. For EBNs, the composite structure contains an edge from variable A to variable B if there exists an edge in at least one of the component Bayesian Networks from A to B or B to A. Edges are then weighted to show the frequency with which they occur. Figure 3.2 shows an example composite network generated from the hypothetical EBN given in Figure 3.1. The existence of the edge in the composite structure from $V0$ to $V8$ with a weight of 0.86 indicates that an edge from $V0$ to $V8$ or $V8$ to $V0$ was present in 86% of the component networks containing both variables $V0$ and $V8$.

With methods for EBN construction, instance classification, variable importance calculation, and variable relationship identification outlined, we shift our focus to an empirical evaluation of our model. Chapter 4 concentrates on an analysis of EBNs utilizing a variety of public datasets. We present a description of the datasets

employed, methods for gauging performance and comparing EBNs with traditional Bayesian Networks, an evaluation of the two main parameters for construction, and an analysis of classification performance when trained on small datasets.

Chapter 4

Empirical Evaluation

We now concentrate on an empirical evaluation of the EBN model we introduced in Chapter 3. We analyze various experiments to improve our understanding of the two main parameters driving EBN construction. The EBN size and component network size parameters are the most important factors influencing the shape of the EBNs. Another focus of this evaluation is to show that EBNs are a successful model for classification when trained on small datasets. Additionally, we examine the EBNs' ability to outperform, or perform as well as, classification using traditional Bayesian Networks.

4.1 Scoring Metrics

Throughout our experiments, we utilize area under the receiver operator curve as a performance measure of an EBN's classification abilities. We could have used classification accuracy to evaluate our model's efficacy; however, accuracy can misrepresent the classification performance under unknown classification costs and unequal class distributions. For example, using accuracy to judge classification of models trained from datasets in which there are far fewer instances with a negative class value than a positive class value, models that always predict the positive class will have very high accuracy. This high accuracy score is not a good measure of the model's classification performance because it is safer for the model to always predict the more likely class label rather than try to predict the correct class label. Area Under the receiver operator Curve (AUC) is a more precise measurement of an algorithm's performance under unknown misclassification costs and unequal class distributions (Provost and Fawcett 2001). These restrictions were encountered in our data, and therefore AUC was the logical choice.

The receiver operator curve (ROC) allows for visualization of the ratio of the true positive rate to false positive rate for various classification cutoffs. A point on the

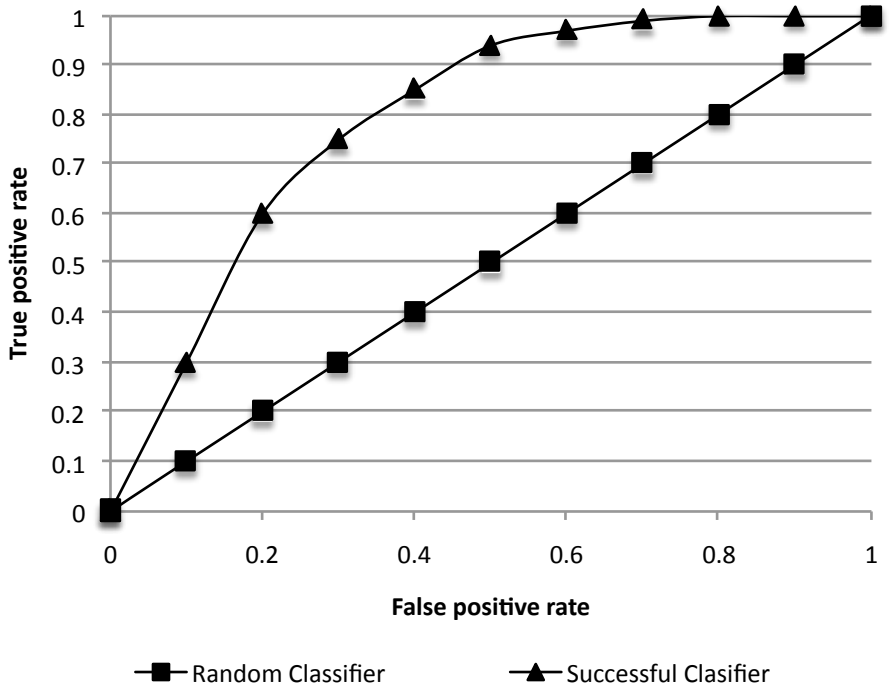


Figure 4.1: Two receiver operator curves showing the performance of a random classifier and a succesful classifier.

receiver operator curve is the ratio of a classification model's true positive rate to the false positive rate for a specific probability threshold. For models producing a class label, this ratio is the percentage of positive instances classified correctly divided by the percentage of negative instances classified incorrectly. On the other hand, some classification models, such as Bayesian Networks, produce a probability of class membership rather than a class label. For these specific models to produce a class label, a cutoff must be provided. Instances with probabilities above this cutoff are classified as positive, and instances with probabilities below the cutoff are classified as negative.

When given a model that outputs a probability for class membership, assigning a class label requires a cutoff value. This cutoff varies drastically across different datasets and it is often difficult to select accurate cutoff values. For datasets with highly unbalanced class labels the cutoff value could be greater than 0.80 or 0.90. In contrast, cutoffs for datasets with a more even distribution of class labels are closer to 0.50. AUC provides a measure of the robustness of a model across varying

classification cutoffs. This metric measures the probability that given one positive and one negative instance, a classification model will assign a higher probability to the positive instance than it does for the negative instance. Two example receiver operator curves can be seen in Figure 4.1. The AUC is measured from zero to one, with a perfect algorithm scoring 1.0 and a random algorithm earning a score of 0.50. A ROC for a random model and a successful classification model can be seen in Figure 4.1.

4.2 Datasets

For empirical evaluation, experiments were performed using six standard Bayesian Networks obtained from the Bayesian Network Repository, BNR[1]. This repository was formed to encourage the use of standard networks in empirical research, allowing for comparison and replication of results across several different problems (Friedman et al. 1997). These well-known networks range in size from nineteen to fifty-six variables with varying degrees of cardinality, and provide a method to benchmark EBNs before moving on to our case study. These particular networks are an acceptable base for our evaluation because they were also used in the evaluation of the constraint based structure learning algorithm POWER (Fast et al. 2008). Recall we have selected POWER as the internal Bayesian Network structure learning algorithm for EBNs. We now present a set of metrics describing the six networks we have chosen for our empirical evaluation: Alarm, Hailfinder, Insurance, Water, Diabetes0, and Child.

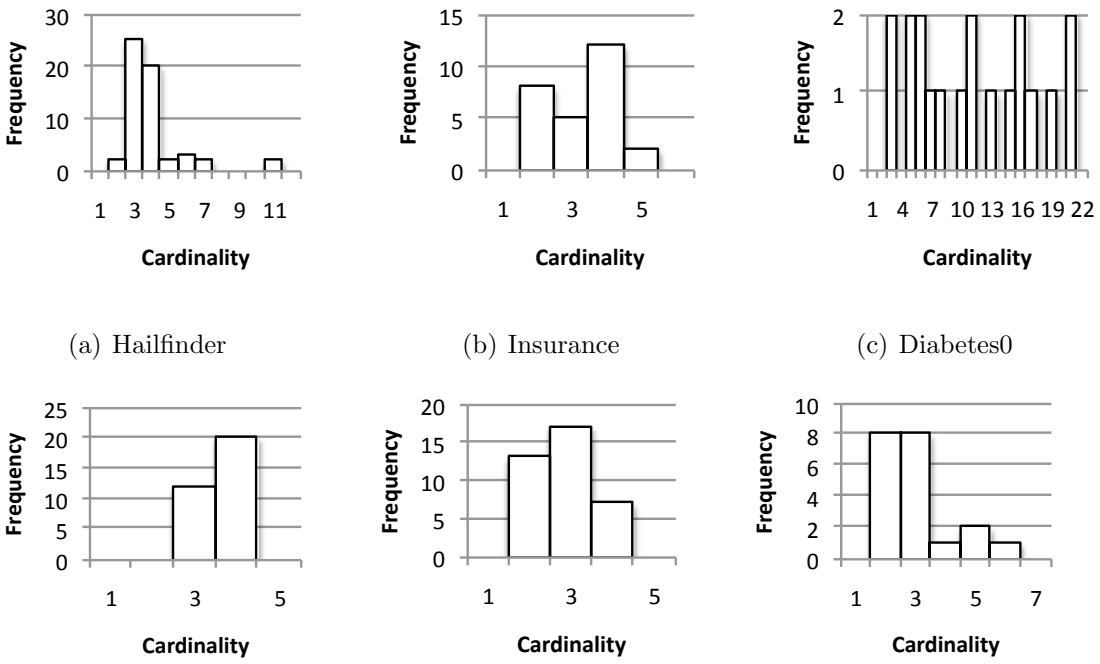
Table 4.1 shows various metrics for each of the six datasets used in our empirical analysis. These metrics include the total number of nodes and edges in each network. In addition, the mean degree and mean cardinality are reported. A node's degree is the total number of parents and children of that node. The mean degree provides a measure of the density of the generating Bayesian Network structure. A node's cardinality is a measure of the number of values the node's variable can take on. In addition to reporting the mean cardinality in Table 4.1, the frequency of nodes with particular cardinalities for the various datasets can be seen in Figure 4.2. Each of the six datasets are described in more detail below.

[1] `http://www.cs.huji.ac.il/~galel/Repository/`

Network	# Nodes	# Edges	Mean Degree	Mean Cardinality
Alarm	37	46	1.24	2.84
Hailfinder	56	66	1.18	3.98
Insurance	27	52	1.93	3.30
Water	32	66	2.06	3.63
Diabetes0	19	23	1.21	11.21
Child	20	25	1.25	2.90

Table 4.1: Metrics for the six datasets used in our empirical evaluation.

- **Alarm**: A network designed by medical professionals as part of ALARM (A Logical Alarm Reduction Mechanism) for use in patient monitoring (Beinlich et al. 1989). Our classification models were trained to predict the variable BP.
- **Hailfinder**: A Bayesian Network developed as a model for predicting severe weather in Northeast Colorado (Edwards 1998). Instances were classified based on the attribute PlainsFcst.
- **Insurance**: A Bayesian Network that models expected claim cost for car insurance policy holders. The network has three outputs: medical coverage, liability coverage, and property coverage costs (Binder et al. 1997). Models were used to predict the value of PropCost.
- **Water**: A timeseries based model used to represent the complex biological processes that take place in water purification (Jensen et al. 1989). Variables were classified as positive if the value of CNON_12_45.
- **Diabetes0**: A model used as a guide for practitioners when determining insulin dosage amounts for diabetes patients (Andreassen et al. 1991). Classification models were trained to predict the variable basal_bal_0.
- **Child**: A network used for diagnosing children with a particular ailment given some known symptoms (Cowell et al. 2007). Instances were classified as positive if the model predicted the variable GRUNTING.



(a) Hailfinder (b) Insurance (c) Diabetes0

(d) Water (e) Alarm (f) Child

Figure 4.2: Frequency of nodes with various cardinalities for each of the six datasets used in our empirical evaluation

4.3 Analysis of Component Size

The first of two necessary parameters required for EBN construction is component size, m. Recall from Chapter 3 the value for m is the total number of variables, including the variable required for classification, considered during component network construction. The m variables will be considered by the Bayesian Network structure learning algorithm during the creation of each component Bayesian Network. Naturally, the range of possible values varies with each dataset D with $1 <= m <= vs$; where vs is the total number of variables in the dataset. Breiman (2001) suggests values of $m << vs$, specifically $m \approx \log_2(vs) + 1$.

In reality, the lower bound of the aforementioned range is too small to be practical. Selecting m as 1 results in an EBN where each component is a probability distribution for the classification variable. In our experiments, we found values of $m < 4$ to be impractical. The structure learning algorithm utilized for the creation of component structures was unable to terminate with such a small number of variables. On the other end of the spectrum, it has been shown that selecting a value for m which

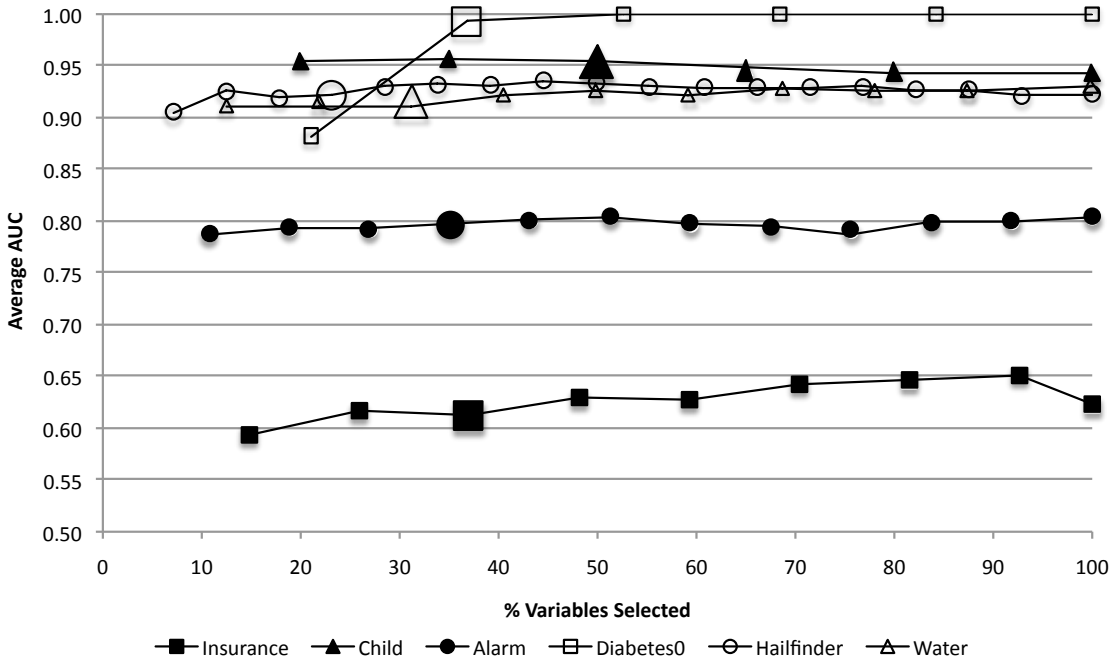


Figure 4.3: Network Size: % of variables and AUC. The larger marker represents the percent of variables used for each dataset for the remainder of this chapter.

is too large results in increased correlation between components. This increase in correlation impacts performance of the ensemble in a negative way (Breiman 2001).

As part of our parameter exploration, we investigate the impact of m on classification performance for various datasets. For each dataset we generate five EBNs of size 500 for each of the possible values of $4 <= m <= vs$ in increments of three. An AUC value is then produced for each of the five EBNs using a test set of size 100. These AUCs are averaged to produce a score for the EBN with component size m. Figure 4.3 reports the mean AUC for each dataset as a function of the percentage of total variables used in component network construction. Like Random Forests, the effect of varying m on the models' performance is small.

Of the six standard datasets used for our experiments, we only observed a large increase in performance, as a function of m, on Diabetes0. Smaller AUC increases were detected in Insurance, Alarm, Water, and Hailfinder. Also notable is the slight decrease in AUC as m approaches the upper limit of its range. This phenomena is attributed to the increase in correlation between the individual network components

Dataset	Number of variables	m
Alarm	37	7
Hailfinder	56	9
Insurance	27	7
Water	32	7
Diabetes0	19	6
Child	20	6

Table 4.2: Component size parameter: m

of the EBN. The impact of correlation is most prevalent on Insurance but can also be seen on Child and Hailfinder.

Beyond the increased correlation among component structures, another factor that discourages large values of m is the runtime needed to construct individual component Bayesian Networks. Since the search space can increase exponentially with each additional variable, large values of m can dramatically increase runtimes. While our experiments suggest there is leeway in the selection of m, our remaining analysis of EBNs will use values of m equivalent to $\lceil \log_2(vs) \rceil + 2$. This is one more than the number suggested by Breiman (2001). We add an additional variable because of the requirement that each component contain the classification variable. Table 4.2 shows the values of m we have selected for each dataset in our empirical evaluation.

4.4 Analysis of EBN Size

In conjunction with the parameter for component size, discussed in Section 4.3, EBN construction requires an additional sizing parameter. This second parameter is EBN size, or number of component networks, n as referenced in Table 3.1. The value of n can range from one to positive infinity and has a direct impact on the running time of both EBN construction and classification. At the smallest extreme, an EBN with $n = 1$ is a standard Bayesian Network that has been trained on a subset of data instances and variables. EBNs of larger size lead to more consistent classification performance, but result in longer runtimes for both construction and classification. The increase in EBN size is beneficial and the increased runtime is easy to overcome. The independent construction of an EBNs' component networks allows for highly

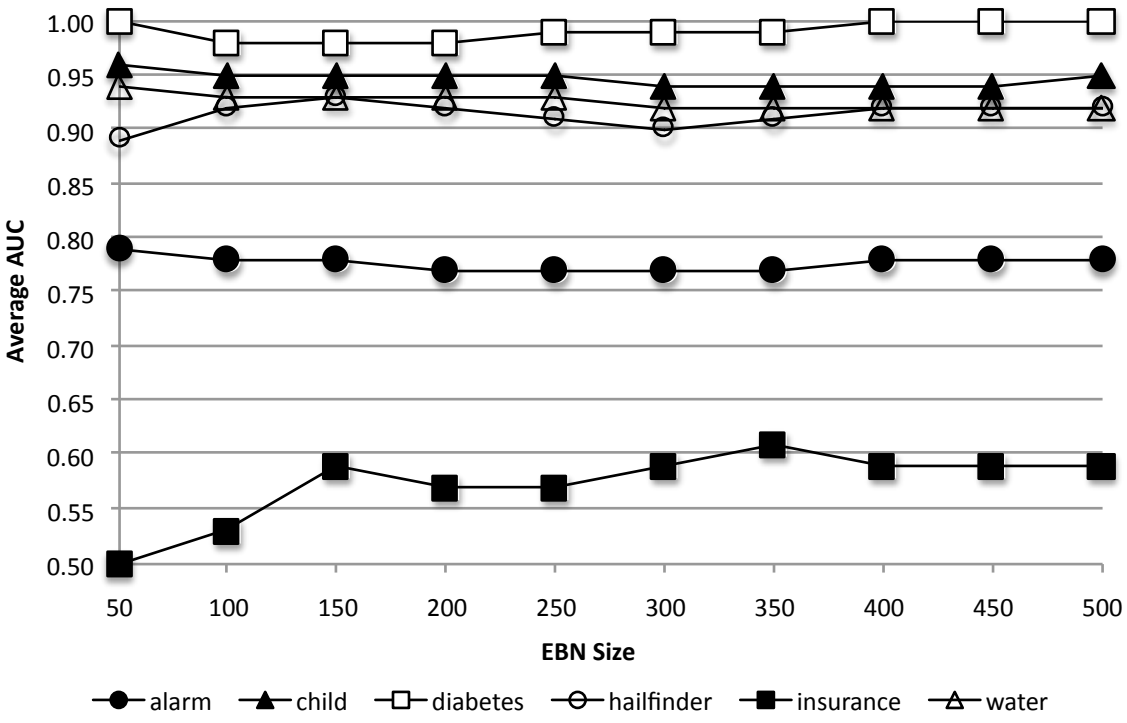


Figure 4.4: EBN size and AUC.

parallelized generation that may be used to minimize the additional runtime needed for construction.

Selecting an appropriate EBN size is a balance between performance and runtime. The classification performance of Random Forests has been shown to converge rapidly when varying the number of trees in the forest. Our empirical evaluation suggests the same holds true for EBNs. A parameter search for EBN size was run on the empirical datasets detailed in Section 4.2. Five training datasets of size 500 were generated for each empirical dataset. Using these training datasets, we generated 500 component Bayesian Networks, and formed five EBNs of varying size ranging from 50 to 500 in increments of 50. A test set of size 100 was then generated for each of the five training datasets and used to test the classification ability of each EBN. The mean AUC across the five datasets for each EBN size were then compared to observe the effect of model size on classification performance. The resulting AUC for each classification can be seen Figure 4.4.

For the datasets used in our analysis, classification performance increases as the number of Bayesian Networks considered by the EBN increases. However, the increase in AUC becomes asymptotic as the size of the EBN reaches about 500 networks. This

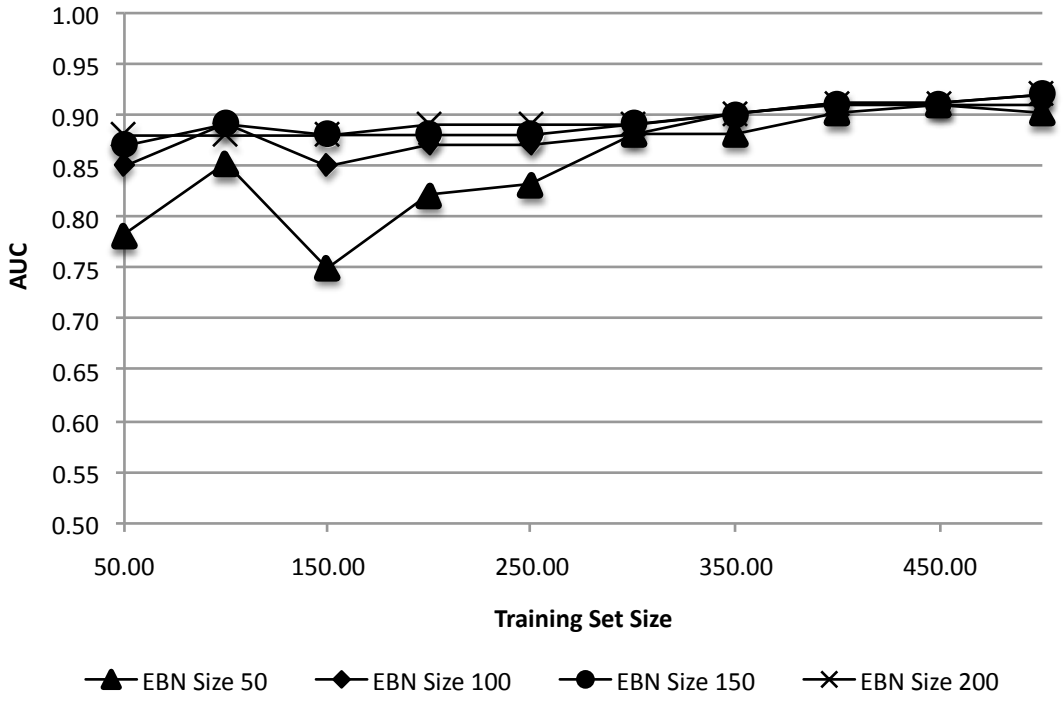


Figure 4.5: Hailfinder: Analysis of web size

quick convergence is consistent with traditional Random Forests (Breiman 2001) and was also verified by Supinie et al. (2009).

The effect dataset size may have on the relationship between EBN size and classification performance is also significant to our work. Figure 4.5 and Figure 4.6 are an analysis of two datasets showing the performance implications of EBN size on EBNs generated from datasets of varying training set size. When trained on small datasets an increase in the number of component networks constructed as part of the EBN can increase performance. However, as the dataset size increases to approximately 500 training instances, the performance of EBNs of various sizes converge. The four other datasets in our analysis show similar behavior with respect to the relationship of training set size and EBN size on classification performance.

4.5 Analysis of Dataset Size

Large datasets are considered a luxury in machine learning, and many real world problems are backed by datasets traditional machine learning algorithms would consider

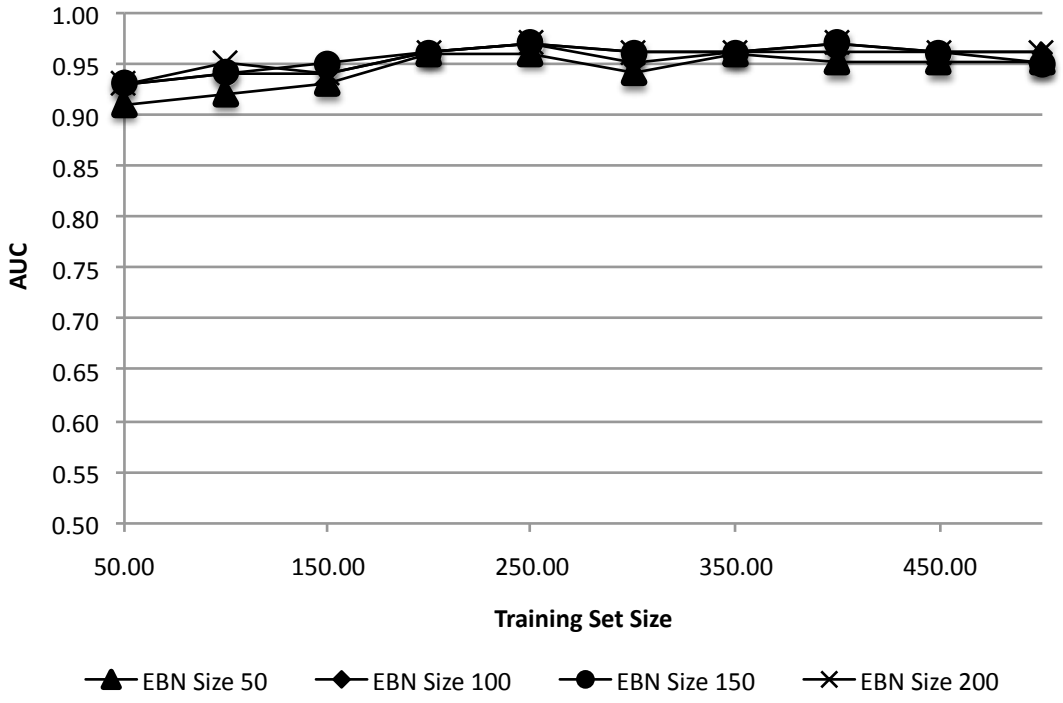


Figure 4.6: Child: Analysis of web size

small. Large datasets may be too expensive or time consuming to collect. Similarly, there may not be enough observations to produce large amounts of data. These factors can limit the amount of data collected, resulting in the need for analysis of small sized datasets. Even though machine learning techniques excel in finding patterns in large datasets, domain experts and computer scientists alike still desire the use of Machine Learning techniques in the analysis of small datasets. Analysts prefer Machine Learning techniques because they are unbiased and can identify relationships within data not easily discovered by humans. It is possible to generate data models from small datasets using Machine Learning techniques, but the models typically suffer from overfitting. The resulting models are tied so tightly to the original training data that they often underperform when provided with new instances for classification. This phenomenon has showed up in our past work (McGovern et al. 2008b) and our desire to eliminate overfitting was a leading factor in the creation of EBNs.

To understand the effect of the training set size used for EBN construction on classification performance, we set up a series of dataset size experiments. Considering our research in educational data was constrained by small datasets, an upper bound

on training set size was established at 500 instances. We generated synthetic training datasets of sizes 50, 100, 150, 200, 250, 300, 350, 400, 450, and 500 from each of the networks discussed in Section 4.2. It was possible that not all cases would be represented and that two datasets generated from the same Bayesian Network could be drastically different. Recognizing these restrictions we generated five datasets of each training size. Performing each experiment five times allowed us to observe performance variations on EBNs generated from training datasets of the same size. In addition to the generated training sets, five test sets, of size 100, were constructed for evaluating the models learned on each training dataset. These five sets were used to test classification performance across the EBNs for each of the five synthetic datasets generated.

Our dataset size experiments used the following procedure. First, five Bayesian Networks were generated using the POWER (Fast et al. 2008) structure learning algorithm for each training set size. The five networks (Full PBNs) generated from each dataset were then used to classify the test instances from their corresponding test set. The output of this test set classification was used to calculate an AUC score. The mean AUC was calculated across the five Bayesian Networks generated for each training set size. Each computed mean was used as the baseline score for its training set size.

With a baseline performance metric established, we constructed EBNs for comparison. Following a procedure similar to determining the baseline, five EBNs were learned for each training set size. In our experiments, we fixed the size of each EBN to 200 components, and used the values in Table 4.2 as the component size for each dataset. The constructed EBNs were scored in the same manner as the baseline Full PBNs. This procedure produced an AUC for each EBN, in each training size. For each training size the AUCs were averaged across the five training datasets and compared with the baseline scores. Figure 4.7 through Figure 4.12 compare the performance measured by AUC, as a function of the training set size for the generated Full PBNs and EBNs on the six standard datasets from the BNR.

To determine if the difference between performance curves is significant, we use a combination of randomized analysis of variance (Randomized ANOVA) (Piater and Cohen 1998) and t-tests. Randomized ANOVA is a statistical method for comparing sets of performance curves and can be used even if points on the curve are not independent. Using randomized ANOVA, we produce a confidence value for rejecting the

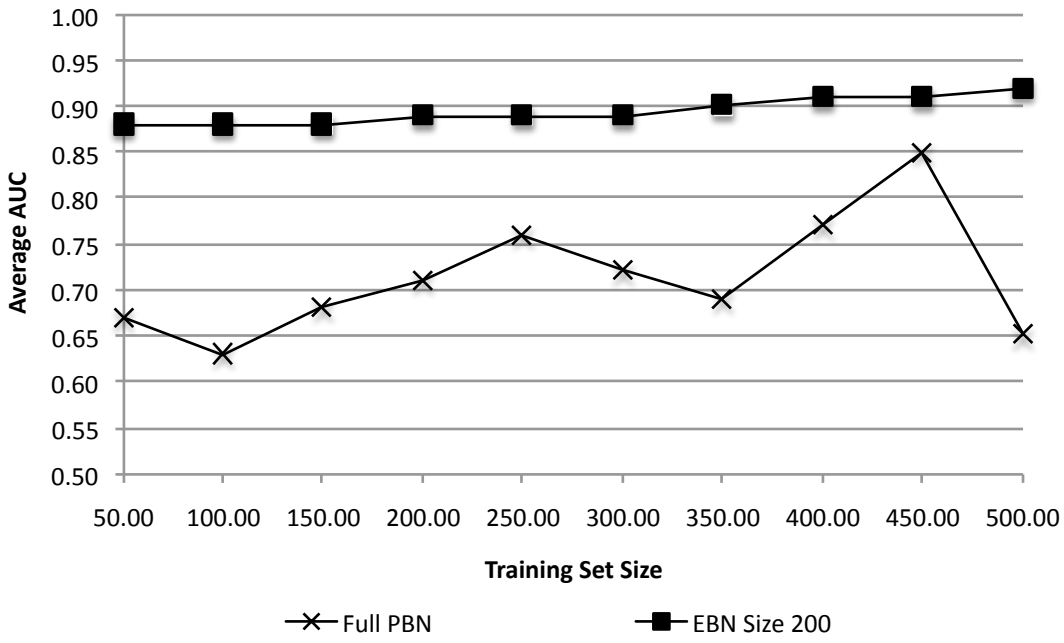


Figure 4.7: Hailfinder: Training set size analysis

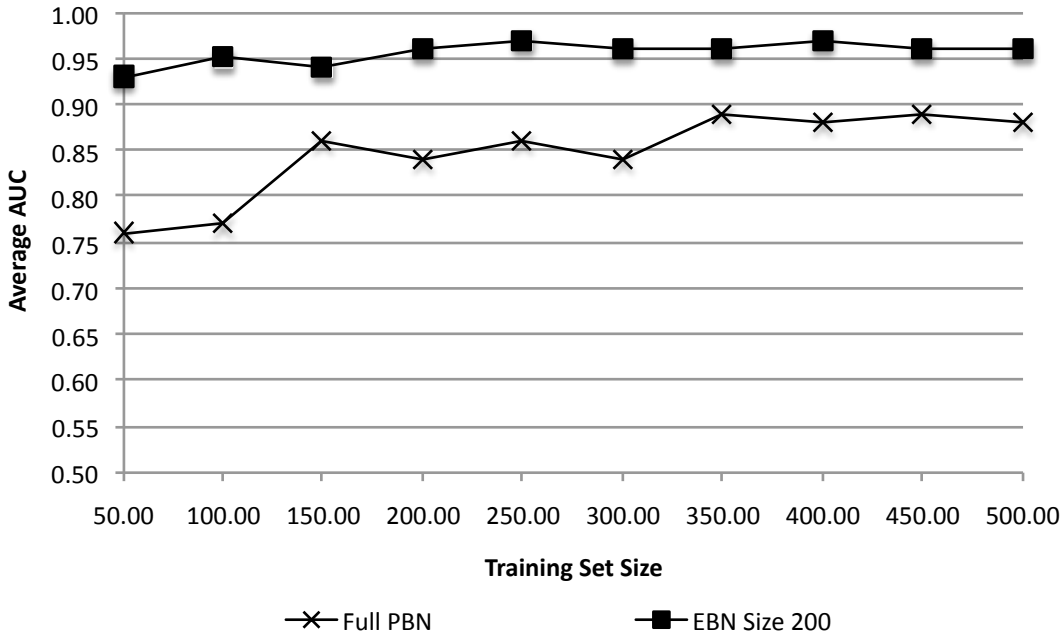


Figure 4.8: Child: Training set size analysis

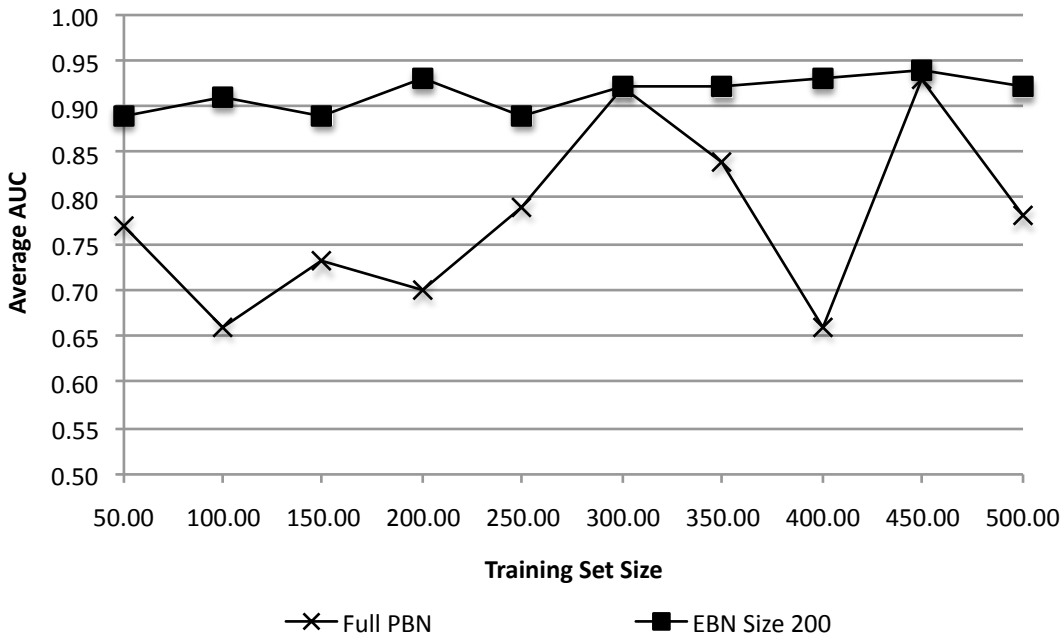


Figure 4.9: Water: Training set size analysis

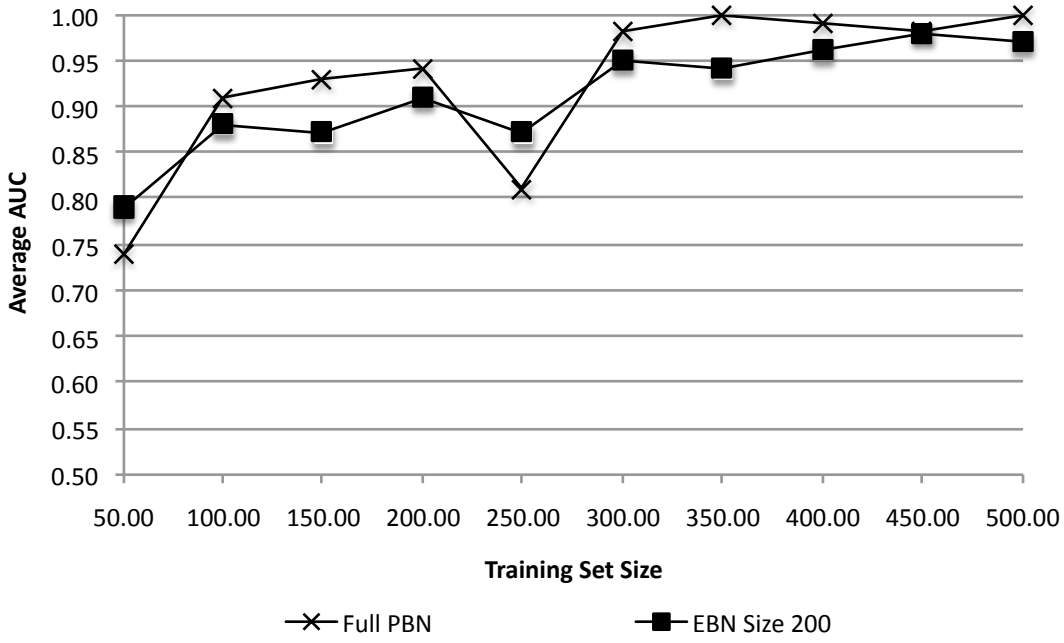


Figure 4.10: Diabetes0: Training set size analysis

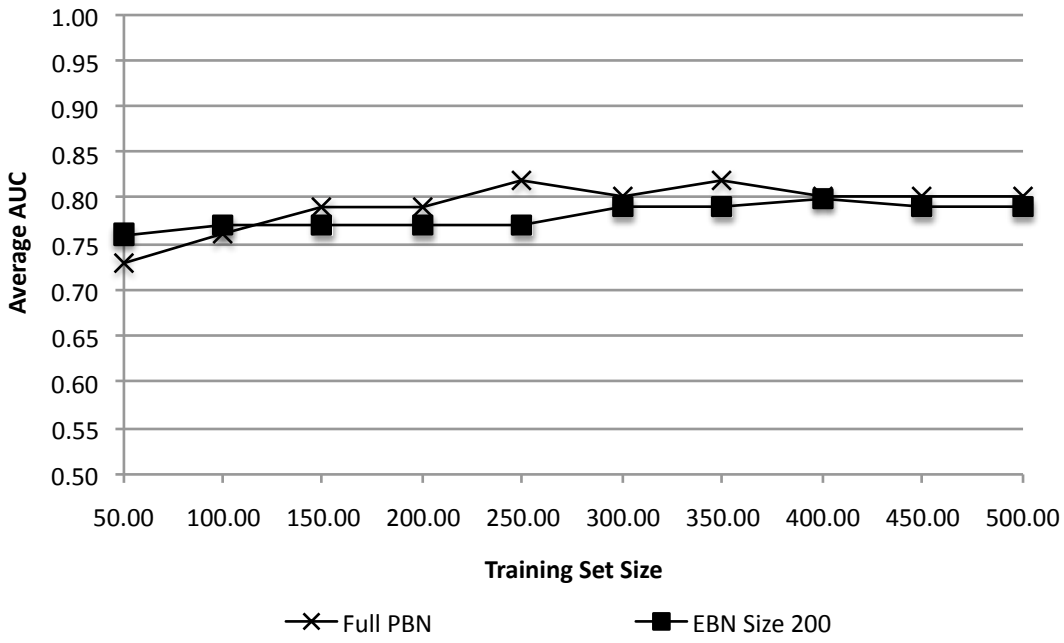


Figure 4.11: Alarm: Training set size analysis

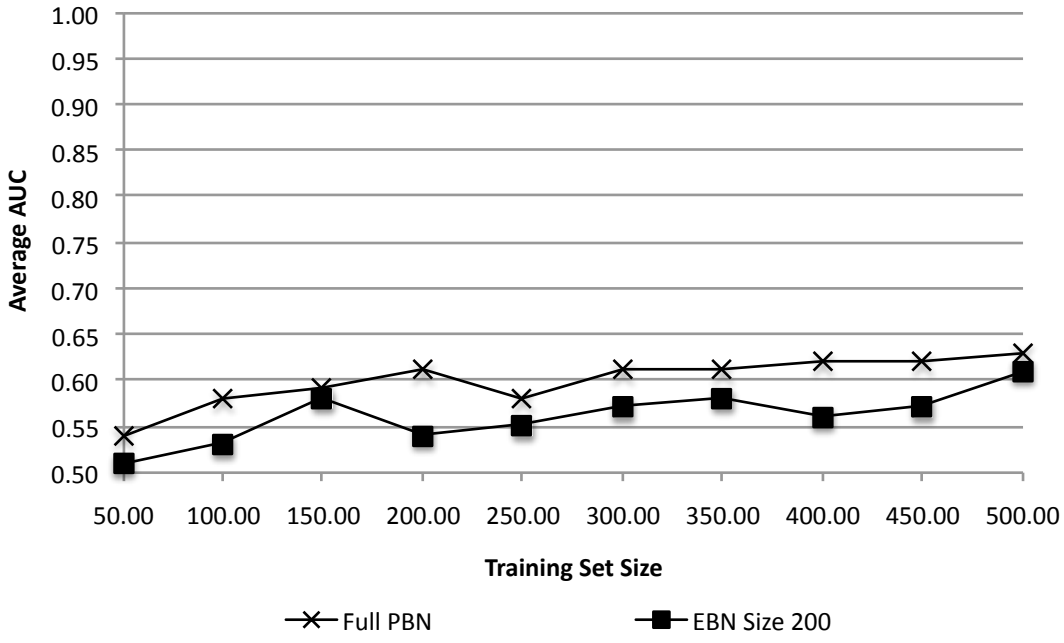


Figure 4.12: Insurance: Training set size analysis

Hailfinder	Child	Water	Diabetes0	Alarm	Insurance
0.002	**0.009**	**0.013**	0.390	0.126	**0.006**

Table 4.3: Randomized ANOVA: p-value for rejecting the null hypothesis stating "The mean performance of EBNs and Bayesian Networks are the same." Values that are statistically significant ($\alpha < 0.05$) are shown in bold.

null hypothesis that the mean performance of EBNs and Bayesian Networks are the same. Paired t-tests may be used to provide a p-value at various points, such as the endpoints on the performance curves. Table 4.3 details the p-values for rejecting the null hypothesis determined for our training set size experiments.

For the first three datasets, Hailfinder, Child, and Water respectively, EBNs outperform their corresponding Full PBNs in the task of classification. Given the p-values in Table 4.3 we reject the null hypothesis using a cutoff of 0.050 for these three datasets. We hypothesize the performance improvement of EBNs is present because the traditional Bayesian Networks used as a baseline have been overfit to the small size of our training sets. An analysis of the standard deviation across the five models at each training set size for Hailfinder supports this theory. As seen in Figure 4.13 the standard deviation for the Full PBNs are larger than the standard deviation for their corresponding EBNs. The high value of standard deviation indicates volatility in the five Full PBNs generated at each training set size. The decrease in volatility for the Full PBNs as the training set size increases implies the amount of overfitting is also decreasing. This is confirmed by the closing of the AUC gap between the Full PBNs and EBNs as the training set size increases. In contrast to the Full PBNs, the standard deviation for the generated EBNs indicate less volatility between models; as such, the performance of EBNs remain stable as the training set size increases. The analysis of stability for both Alarm and Child show similar results to Hailfinder.

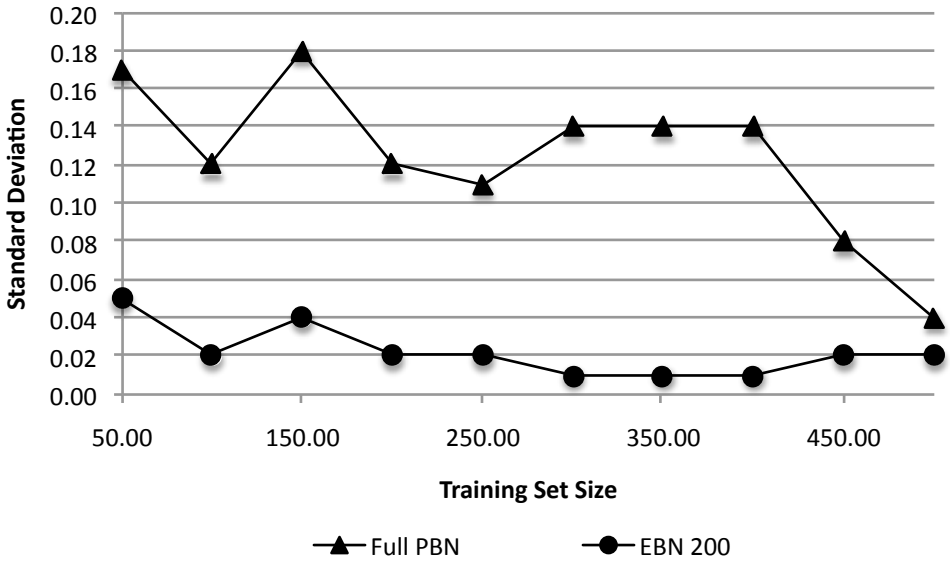


Figure 4.13: Hailfinder: Standard deviation for Full PBNs and EBNs as a function of training set size

These results are in line with our expectation that the Random Forests techniques utilized in EBN construction dramatically decrease the amount of overfitting seen in models trained on small datasets. Since EBNs do not overfit, they are an effective model for classification when trained on datasets with many features but few training instances. The capability to be trained from small datasets distinguishes EBNs from other classification models prone to overfitting.

Results from the remaining three datasets, Diabetes0, Alarm, and Insurance can be seen in Figure 4.10, Figure 4.11, and Figure 4.12 respectively. Although the performance gap between traditional Bayesian Networks and EBNs is not present in these datasets, important observations can be made.

1. Although EBNs do not outperform Bayesian Networks on datasets Diabetes0 and Alarm, our analysis shows them to be statistically the same (Table 4.3).

2. As with Full PBNs, EBNs benefit from increased quantities of training data. Notably, both Diabetes0 and Insurance show upward trends in performance as the training set size increases. Paired t-tests show these changes are significant with a p-values of 0.014 and 0.000 respectively.

Training Set Size	p-value
50	0.202
100	0.085
150	0.238

Table 4.4: Paired t-test at smallest training sizes for Insurance: p-value for rejecting the null hypothesis stating "The mean performance of EBNs and Bayesian Networks are the same"

3. Although Figure 4.12 and Table 4.3 indicate Bayesian Networks outperform EBNs on the Insurance dataset, a paired t-test shows the points on the performance curve for the lowest training set sizes are statistically the same. Detailed p-values for these test can be seen in Table 4.4.

Items 2 and 3 suggest that when trained on larger datasets, EBNs can perform at least as well as their corresponding Full PBNs. To verify our hypothesis, additional experiments should be run with more training data.

Our dataset size experiments have shown that EBNs are a robust model for classification, even when learned from small datasets. Our analysis has shown the performance gap can be widely attributed to a lack of overfitting when using EBNs, compared to traditional Bayesian Networks. Although large performance improvements in all datasets would have been ideal, EBNs should not be judged solely on classification performance. EBNs provide variable importance metrics and visualization of data relationships, which add to their effectiveness. Understanding the implication of training set size on model performance is important given the criteria we set forth for EBNs. In addition the understanding of the parameters required for algorithm construction is also imperative. Recognizing EBNs meet the criteria set for our model we focus on the original dataset we desired to model. Chapter 5 presents a case study in engineering education. We use EBNs to model educational data in an effort to identify relationships within the dataset and also to build a classifier for predicting retention.

Chapter 5

Case Study: Analyzing Student Data to Predict Retention

Following the empirical evaluation of EBNs discussed in Chapter 4, we apply EBNs to the original data that we wished to analyze. This involves evaluating student data in an effort to identify traits associated with retention. As institutions strive to increase retention and graduation rates in their engineering programs, considerable effort has been put into understanding the factors which contribute to student retention. An evaluation of factors affecting retention and attrition in college engineering programs is critical for identifying those students who may be at risk for leaving engineering programs. Conceivably, this research will aid in developing resources to increase the success rates of students in engineering programs. Identifying these factors is complicated by the fact that no single factor is the primary cause for retention or attrition. Rather, it is the interaction of multiple factors, some not easily measured, which influence retention rates of engineering and computer science students. Researchers in the fields of engineering education and machine learning are drawn to this complex issue.

5.1 Background

Machine Learning techniques have been previously utilized in the analysis of engineering educational data. Mendez et al. (2008) present an analysis of the factors influencing persistence in STEM and engineering majors. Data were collected on freshmen students entering Arizona State University in the 1999-2000 school year. Using Random Forests and classification trees, researchers were able to identify complex relationships within the datasets that were not identified by statistical models traditionally employed on educational data. The use of Random Forests' variable importance scores by Mendez et al. (2008) suggests analysts can identify important variables without specifying a model. In a separate study performed at a large

Midwestern university, researchers analyzed the effect of cognitive and non-cognitive factors on retention of more than 4,900 first-year engineering students from three freshman cohorts (Imbrie et al. 2008). Their results showed the prediction powers of artificial neural networks trained from only cognitive or non-cognitive features were nearly identical; however, the use of both cognitive and non-cognitive features improves the prediction power of the learned artificial neural networks. In addition to the analysis of retention, Machine Learning has also been applied to assessment of student learning (Conati et al. 1997; Vanlehn and Martin 1998).

The data used in our case study is part of a larger project approved by the Institutional Review Board (IRB 10563) describing the differential strategies used by members of four minority groups (Native Americans, Hispanic Americans, Asian Americans, and African Americans) to be academically successful engineering undergraduates at the University of Oklahoma. The primary data sources include one-time and longitudinal interviews with minority engineering students, academic transcripts, and a survey instrument based on the work of Besterfield-Sacre et al. (1997, 1998). Traditionally these data have been evaluated using a qualitative analysis of the interview data with quantitative aspects used for triangulation (Do et al. 2006; Foor et al. 2007; Walden and Foor 2008a,b; Shehab et al. 2007; Foor and Shehab 2009; Wong Lowe et al. 2010; Trytten et al. 2009; Walden and Shehab 2009). This work is unique because it addresses the disaggregation of minority groups, includes a minority group that is not underrepresented (Asian Americans), and focuses on those students who succeed academically. Our recent work (McGovern et al. 2008b) used Bayesian Networks to identify relationships within the quantitative data. The aim of this thesis is to use EBNs as a tool for improving the models built with our previous machine learning techniques.

5.2 Dataset

The data used for this thesis are derived from a modification of the Pittsburgh Junior Engineering Learning and Curriculum Evaluation Instrument (PJEAS) (Besterfield-Sacre et al. 1997, 1998). The modified instrument was administered as supplemental data for a qualitative study of persistence in engineering and computer science. Our data also include demographic statistics for the participants and information excerpted from the participants' academic transcripts. Responses were collected from

150 students across 9 engineering disciplines and computer science. The questions in the original PJEAS instrument were grouped into 13 constructs. Given the small number of participants in this pilot study and the resulting need to limit the number of variables, we grouped the survey questions into the following eight categories: (1) Preparedness, (2) Attitudes about engineering professional career, (3) Attitudes about engineering education and study, (4) Confidence in problem solving skills, (5) Confidence in engineering skills, (6) Confidence in communication skills, (7) Confidence in engineering's impact on society, and (8) Engineering related work experience.

The questions contained in each grouping for both the Computer Science and Engineering surveys can be seen in Table 5.1 through Table 5.8. "Preparedness", shown in Table 5.1, contains questions designed to gauge students perceptions of how prior courses and experiences have prepared them for the skills required in engineering and computer science related roles. Table 5.2 lists the questions in Group 2, "Attitudes about engineering as a professional career." These questions are written to gauge students' perceptions of their future careers. "Attitudes about engineering education and study," Table 5.3, includes questions aimed at judging students' feelings regarding their current major, and the students' perceived abilities to fit into their major. Questions used to determine students' perceptions of their problem solving skills, Group 4, can be found in Table 5.4. Group 5, "Confidence in engineering skills", found in Table 5.5 identifies students' levels of confidence concerning the skills required in an Engineering or Computer science profession. Questions written to gauge students' confidence in his or her oral and written communication skills, Group 6, can be found in Table 5.6. The last of the confidence related groups, "Confidence in engineering's impact on society," can be found in Group 7. These questions are designed to measure students' views of the professions' future impact on society. Finally, "Engineering related work experience," Group 8, was used to determine the impact of past work experience in the engineering or computer science fields such as internships and co-ops.

For the survey questions, students' Likert responses for the questions assigned to each category were averaged. This calculation resulted in an answer of NA if none of the questions in the groups were answered or a numeric score indicating their average response between 0 and 5. Demographic and academic transcript variables included ethnicity, gender, major (divided by enrollment managed discipline or not), hours of transfer credit, high school type (rural or urban), number of hours of advanced

Computer Science	**Engineering**
Ability to apply discrete math concepts to solve problems.	Ability to apply math concepts to solve engineering problems.
	Ability to apply chemistry concepts to solve engineering problems.
	Ability to apply physics concepts to help solve engineering problems.
Ability to solve unstructured problems.	Ability to solve unstructured engineering problems.
Ability to analyze data.	Ability to analyze engineering data.
Ability to design a process.	Ability to design a device or process.
Ability to use proper software engineering processes.	Ability to use proper laboratory procedures.
Computer programming skills.	Computer programming skills.
Ability to use software packages to solve problems.	Ability to use software packages to solve engineering problems.
Technical writing ability; i.e., prepare reports and papers.	Technical writing ability; i.e., prepare engineering reports and papers.
Oral communication skills.	Oral communication skills.
Ability to function effectively in different team roles.	Ability to function effectively in different team roles.
Ability to set goals and achieve them on time.	Ability to set goals and achieve them on time.
Ability to learn new things on my own.	Ability to learn new things on my own.

Table 5.1: Preparedness

Computer Science	**Engineering**
I expect computer science will be a rewarding career.	I expect engineering will be a rewarding career.
I don't care for this career.	I don't care for this career.
From what I know, computer science is boring.	From what I know, engineering is boring.
Computer Science is an exact science.	Engineering is an exact science.
Computer scientists are in occupations that are respected by other people.	Engineering is an occupation that is respected by other people.
I like the professionalism that goes with being a computer scientist.	I like the professionalism that goes with being an engineer.
Computer Scientists have contributed greatly to solving society's problems.	Engineers have contributed greatly to solving society's problems.
I feel I know what a computer scientist does.	I feel I know what an engineer does.

Table 5.2: Attitudes about engineering professional career

Computer Science	**Engineering**
I have no desire to change to another major (ex. Biology, English, Chemistry, Art, History, etc.).	I have no desire to change to another major (ex. Biology, English, Chemistry, Art, History, etc.).
I enjoy mathematics the most of all my subjects.	I enjoy the subjects of science and mathematics the most of all my subjects.
Creative thinking is one of my strengths.	Creative thinking is one of my strengths.
I feel confident in my ability to succeed in computer science.	I feel confident in my ability to succeed in engineering.
I prefer studying/working alone.	I prefer studying/working alone.
I am good at designing things.	I am good at designing things.
I consider myself technically inclined.	I consider myself technically inclined.
I enjoy solving open-ended problems.	I enjoy solving open-ended problems.

Table 5.3: Attitudes about engineering education and study

Computer Science	**Engineering**
Using discrete mathematical concepts to solve problems.	Using mathematical concepts to solve engineering problems.
Using computer science concepts to solve relevant problems.	Using chemistry concepts to solve engineering problems.
	Using physics concepts to solve relevant engineering problems.
	Designing an experiment to obtain measurements or gain additional knowledge about a process.
	Analyzing a set of data to find underlying meaning(s).

Table 5.4: Confidence in problem solving skills

Computer Science	**Engineering**
Designing a process or program when given a set of specifications.	Designing a device or process when given a set of specifications.
Functioning as an accountable member of a computer science team.	Functioning as an accountable member of an engineering team.
Solving unstructured problems.	Formulating unstructured engineering problems.
Using appropriate programming techniques and tools for problem solving.	Using appropriate engineering techniques and tools including software and/or lab equipment for problem solving.
Recognizing the limitations of my computer science knowledge and abilities and knowing when to seek additional information.	Recognizing the limitations of my engineering knowledge and abilities and knowing when to seek additional information.

Table 5.5: Confidence in engineering skills

Computer Science	**Engineering**
Writing effectively.	Writing effectively.
Making professional presentations.	Making professional presentations.
Effectively communicating computer science-related ideas to others.	Effectively communicating engineering-related ideas to others.
	Listening to and impartially interpreting different viewpoints.

Table 5.6: Confidence in communication skills

Computer Science	**Engineering**
Understanding the professional and ethical responsibilities of a computer scientist.	Understanding the professional and ethical responsibilities of an engineer.
Understanding the potential risks (to the public) and impacts that a computer science solution or design may have.	Understanding the potential risks (to the public) and impacts that an engineering solution or design may have.
Applying knowledge about current issues (economic, environmental, political, societal, etc.) to computer science related problems.	Applying knowledge about current issues (economic, environmental, political, societal, etc.) to engineering related problems.

Table 5.7: Confidence in engineering's impact on society

Computer Science	**Engineering**
Increased my ability to succeed in my computer science classes.	Increased my ability to succeed in my engineering classes.
Were related to computer science.	Were related to my specific field of engineering.
Provided me with the opportunity to pursue learning on my own.	Provided me with the opportunity to pursue learning on my own.
Allowed me to improve on my computer skills.	Allowed me to improve on my computer skills.
Provided me with the opportunity to work in a team environment.	Provided me with the opportunity to work in a team environment.
Allowed me to work on "real-world" problems.	Allowed me to work on "real-world" engineering problems.
Allowed me to be a more creative problem solver.	Allowed me to be a more creative problem solver.
	Allowed me to work in a laboratory environment.
Helped me to better understand what computer scientists do.	Helped me to better understand what engineers do.
Helped me to develop my written communication skills.	Helped me to develop my written communication skills.
Helped me to develop my oral communication skills.	Helped me to develop my oral communication skills.

Table 5.8: Engineering related work experience

Variable	**Score Ranges**
Gender	Male, Female
Enrollment Management	Yes, No
Ethnicity	p, q, r, s
High School Type	Urban, Rural
Advanced Standing Hours	$<= 12$, > 12
Transfer Credit Hours	$<= 24$, > 24
Engineering Related Work Experience	NA, < 3.59, $3.58 - 4.17$, > 4.17
Preparedness	NA, < 3.65, $3.65 - 4.13$, > 4.13
Confidence in problem solving skills	NA, < 3.57, $3.57 - 4.09$, > 4.09
Confidence in engineering skills	NA, < 3.72, $3.72 - 4.21$, > 4.21
Confidence in communication skills	NA, < 3.57, $3.57 - 4.27$, > 4.27
Confidence in engineering's impact on society	NA, < 3.84, $3.84 - 4.37$, > 4.37
Attitudes about engineering as a professional career	NA, < 3.6, $3.6 - 3.94$, > 3.94
Attitudes about engineering education and study	NA, < 3.5, $3.5 - 3.96$, > 3.96
GPA	NA, < 2.65, $2.65 - 3.25$, > 3.25
Retained	Yes, No

Table 5.9: Discrete variables

standing credit, and grade point average up to date of the interview in Science, Technology, Engineering, and Math (STEM) classes. The final variable, Retained, was defined as positive if a student had graduated with an engineering or computer science degree, or was continuing in good standing towards graduation and negative otherwise. Ultimately, each student was represented by 16 variables. Many of these variables are continuous and with so few students to train on, we discretized the variables by examining the mean of each variable. Breakpoints were identified at the mean plus and minus one half of the standard deviation for each continuous variable. These breakpoints can be seen in Table 5.9. Of our 150 students, 22 were not retained. This lack of balance in class labels supports our use of AUC, as discussed in Section 4.1, for measuring classification performance rather than accuracy.

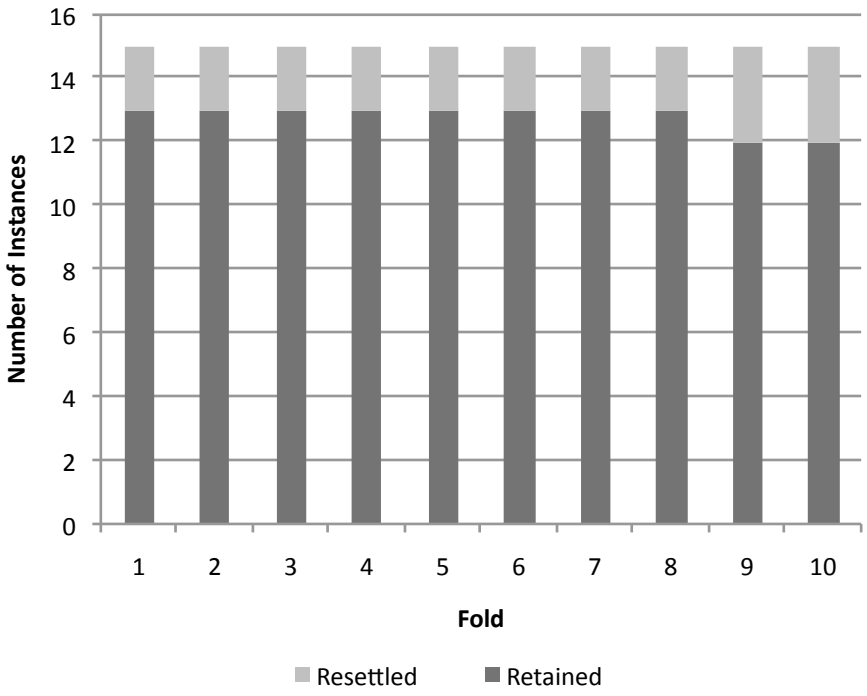


Figure 5.1: Distribution of class labels across the folds used for 10-fold cross validation.

5.3 Experiments

For our experiments we used k-fold cross validation as a method for analyzing how well our model would generalize to an independent dataset. For k-fold cross validation, a dataset is partitioned into k folds, buckets, and a model is trained and tested for each of the k folds. For fold i's model, the instances in fold i are removed and used as a test set and the remainder of the folds are combined into a training set. The results for each of the folds can then be aggregated to provide an analysis of the model in which every instance has been used as both a training and test instance.

In our experiments we used 10-fold cross validation. Instances from our data were assigned to folds pseudo randomly with an equal class label distribution. This assignment resulted in 10 random pairs of training and test sets that were used to ensure consistent results even though the original dataset is highly unbalanced. The distribution of class labels across our 10 folds can be seen in Figure 5.1. For each fold a model is trained on 135 observed instances and tested on the remaining 15 instances.

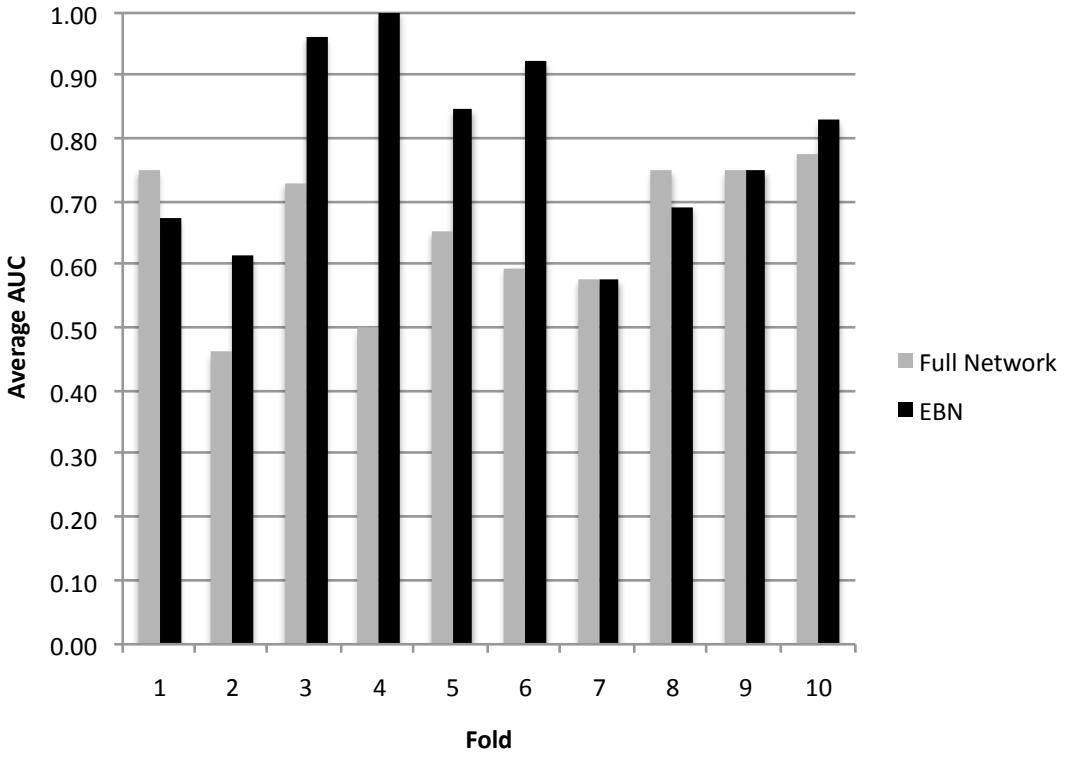


Figure 5.2: Comparison of classification performance, using AUC, for full Bayesian Networks and EBNs

To establish a baseline for comparison we first generated a full Bayesian Network on each of the 10 folds' training sets using the POWER (Fast et al. 2008) structure learning algorithm utilized in our empirical evaluation. With baseline models constructed, we then classified each instance of the folds' corresponding test set to produce baseline AUC scores. These scores can be seen in the Figure 5.2. With an average AUC of only 0.65 and a standard deviation of 0.13, our baseline is an example of the struggles with classification models learned from small, unbalanced datasets. Not only is the average AUC for these baseline models a cause for concern, the high variability across the folds is also unacceptable to domain experts. In an attempt to address both of these issues and provide a measure of variable importance, we explore the use of EBNs.

Ten EBNs were generated and scored using the same folds generated for our baseline tests. Each EBN was composed of 1,000 component Bayesian Networks generated from six total variables including the classification variable, Retained ($n = 1000$, $m = 6$). Using the corresponding test set for evaluation, EBNs outperformed their

Table 5.10: Algorithm for producing aggregate importance scores from multiple EBNs

generateAggregateScores(EBNs *ebns*, **Variables** *vs*)

```
rawScores = [ ]
For ebn in ebns
    rawScores[ebn] =getImportanceScores(vs)
aggregateScores = [ ]
For variable in vs
    For ebn in ebns
        aggregateScores[variable]+= 1-rawScores[ebn][variable]
Return aggregateScores
```

corresponding baseline traditional Bayesian Network on average. Our EBNs had an average AUC of 0.79 with a standard deviation of 0.14. The AUC for our EBN classifier for each fold can be seen alongside its baseline Bayesian Network in Figure 5.2. A paired t-test states the null hypothesis, the performance of Bayesian Networks and EBNs are equal, can be safely rejected with a p-value of 0.050. These results are in line with our empirical analysis of EBNs. Although a notable improvement in classification was documented, domain experts are also interested in identifying which factors influence retention. For this we look at the variable importance measures for those attributes in our EBNs.

Although providing importance measures for an individual fold in our experiment clarifies those variables which are important within that fold, we are most concerned with variables important to classification across all ten folds. Because it is not appropriate to average a variables' z-score or p-value across the different folds, we used the procedure in Table 5.10 to produce a hierarchy of the most important variables across the models generated for each fold. The procedure outlined in Section 3.3 was used to analyze each fold and produce a z-score and a p-value for each variable. The aggregate importance score for a variable was then calculated as the sum over all folds of 1 - the variable's p-value. This procedure produces a score in the range of 0, least important variable, to the theoretical maximum of the number of folds

used to calculate the aggregate importance score. The variable importance rankings determined in our experiments using EBNs can be found in Table 5.11.

Variable	**Aggregate Importance Score**
Confidence in engineering skills	9.98
STEM GPA	9.8
Confidence in communication skills	9.68
Gender	9.52
Enrollment Management	9.48
Preparedness	9.42
Confidence in engineering's impact on society	9.33
Advanced Standing	9.18
Attitude about engineering as a professional career	8.76
HighSchool Type	8.59
Confidence in problem solving skills	8.44
Ethnicity	8.43
Attitude about engineering education a study	8.17
Engineering related work experience	7.34
Transfer Credit	4.82

Table 5.11: Variable importance for STEM Experiment.

Interpreting these results we find students' confidence in engineering skills and their STEM GPA are highly influential when classifying students as Retained. This suggests a correlation between each of the aforementioned factors and retention. It is logical that students who are confident with their engineering skills are more likely to continue on the path towards graduation, whereas those who are struggling with the skills required for a profession in engineering or computer science may pursue other opportunities. STEM GPA also has a direct impact on the models' ability to predict a student as being retained. It makes sense that students who cannot maintain the required GPA for graduation in engineering programs may be forced out of their program. Our previous analysis of this data (McGovern et al. 2008b) also indicated that students who were reluctant to provide access to their STEM grades or had a GPA of < 2.65 were more likely to resettle into different university programs or leave university all together. A high GPA is not always a determining factor in students'

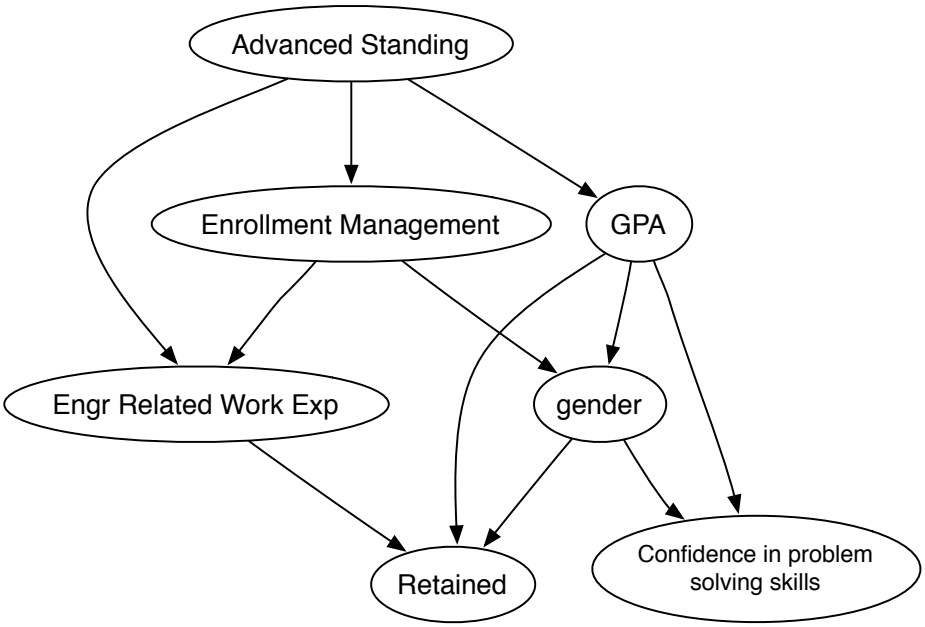


Figure 5.3: Example component network showing relationships between variables

abilities to perform well in engineering programs; however, it can be viewed as proxy for many other variables such as a students' confidence, preparedness, and abilities.

Understanding which variables are important in the task of classifying students as retained is important; however, the factors contributing to successful outcomes are highly complex. Domain experts are not only interested in variables deemed important, but also the relationships between those variables. Figure 5.3 shows the Bayesian Network for an example EBN component network. Although this component structure provides one example of some of the relationships in our dataset, as discussed in Section 3.4, component networks provide only a small piece of the necessary picture.

To obtain a more complete view of the relationships in our data, we generate composite structures for each fold.We then aggregate the 10 folds' composite structures to produce an aggregate composite showing the relationships across all EBNs generated for our data. An edge from A to B exists in the aggregate composite if there exists an undirected edge in one of the folds' composite structures from A to B. The edge label for this structure is the mean weight and standard deviation for a particular edge across the folds' composite structures. As a final step, in an effort to increase readability and highlight those relationships with high frequency, edges with

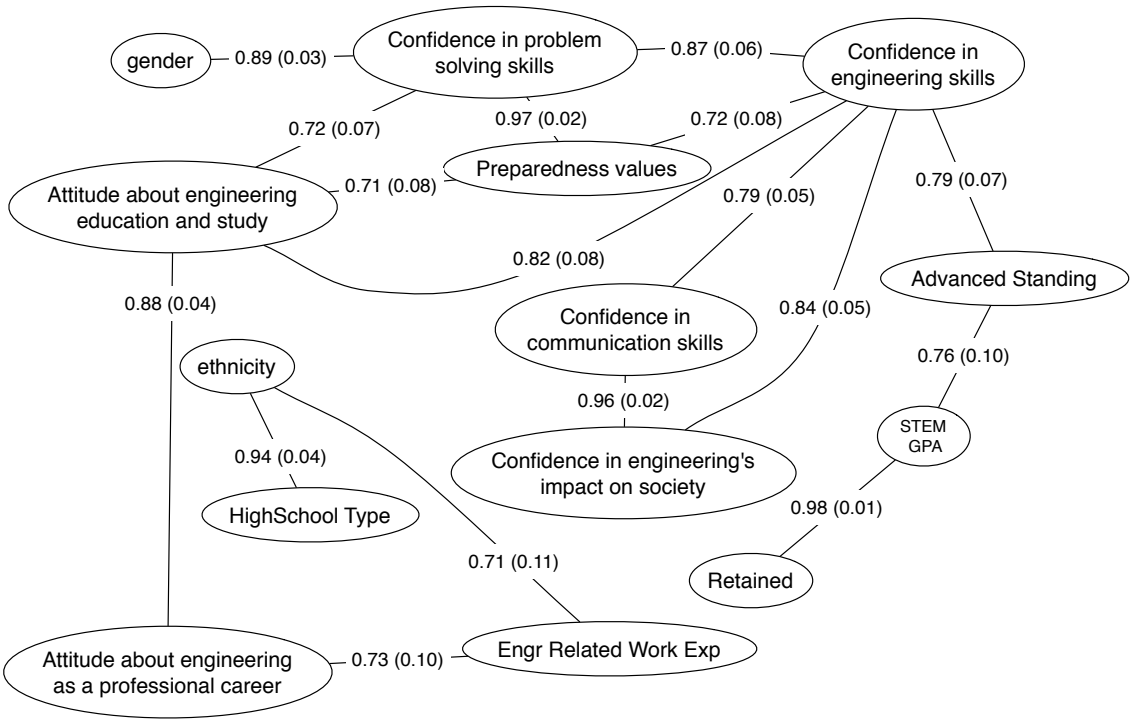


Figure 5.4: Example aggregate composite structure generated from 10 EBN composite structures

an average of less than 0.70 are pruned from the aggregate composite structure. The aggregate composite from our experiments can be seen in Figure 5.4.

As seen in our previous work (McGovern et al. 2008b), the relationship occurring with the most frequency is between STEM GPA and Retained. Recall that students' GPA was also identified as one of the most important variables related to the task of classification. Another relationship of interest is the correlation between Confidence in solving engineering problems and Preparedness. This link seems logical, since students who feel prepared in the aspects required for a career in engineering education are likely to have confidence in their ability to solve engineering problems. Similarly, students who feel ill-prepared are found to have less confidence in their problem solving skills. Also noteworthy are the relationships seen among all of the confidence related question groupings. It stands to reason that a student's level of confidence in one area could affect their perception of confidence in other areas. Missing from the aggregate composite structure is the variable for encoding Transfer Credit. Given its

Model	**Full Mean AUC**	**Trimmed Mean AUC**	**Change**	**p-value**
Bayesian Network	0.65	0.63	-0.02	0.561
EBN	0.79	0.79	0	0.731

Table 5.12: Change in performance between full and trimmed STEM data

low ranking in our variable importance results we are not concerned with Transfer Credit's absence.

With an analysis of our full dataset complete, we were interested in the effect on our results of removing variables identified as least important. We hypothesized that upon removing the noise-creating, least important variables, our classification results would improve and the relationships within our aggregate composite structure would remain largely unchanged. To test this hypothesis, we removed those variables whose importance scores were less than the average importance score minus one standard deviation. This cutoff eliminated both Engineering Related Work Experience and Transfer Credit from our dataset. Using the same procedure as above we reran our experiments on the trimmed dataset. The resulting AUCs across all ten folds can be seen in Figure 5.5 for the full Bayesian Networks and EBNs. Table 5.12 details the change in the performance of EBNs and Bayesian Networks on the trimmed data. Although the performance for full Bayesian Networks decreased to 0.63, the standard deviation across folds increased to 0.14. Conversely, the average AUC for our EBNs remained the same but the standard deviation across folds decreased to 0.11. A paired t-test shows the performance difference between Bayesian Networks and EBNs on the trimmed data is statistically significant with a p-value of 0.003.

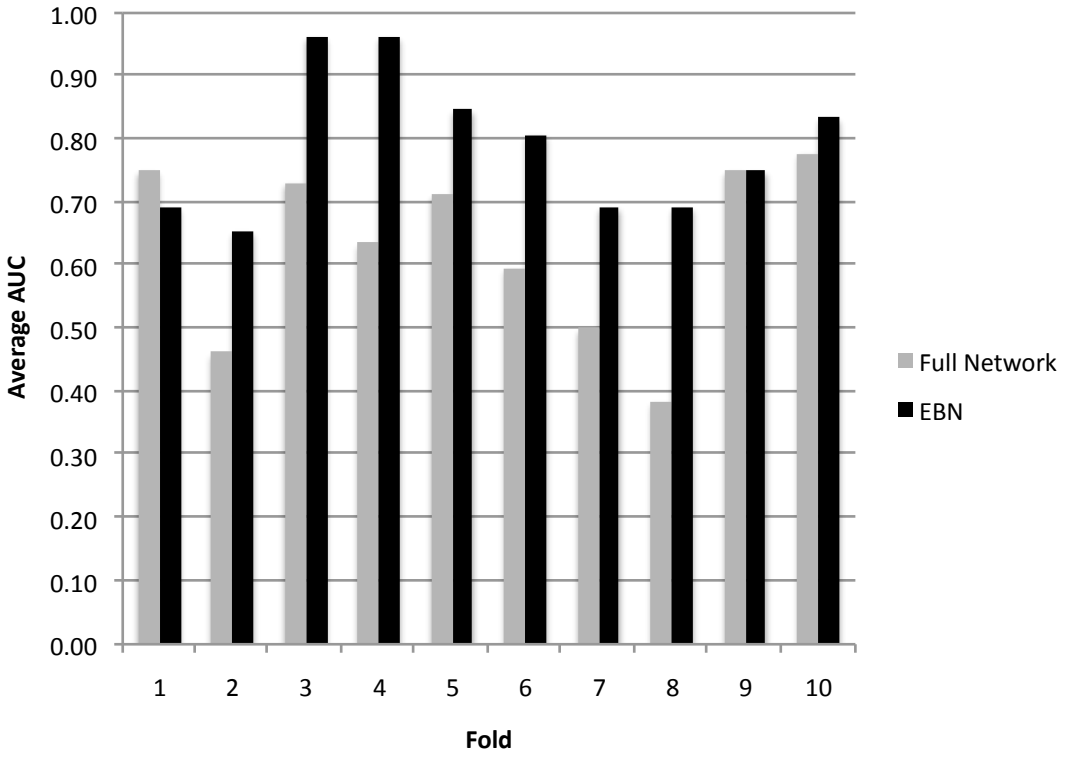


Figure 5.5: Comparison of classification performance, using AUC, for full Bayesian Networks and EBNs on trimmed data

These results support our hypothesis that removing variables with the lowest importance scores would have little to no impact on the performance of EBNs. The aggregate variable importance scores from our experiments on the trimmed data can be seen in Table 5.13. While the ordering of the variables remained largely unchanged from the previous rankings, a few local changes between pairs of variables were noted.

The aggregate composite structure for our experiments using the trimmed data can be seen in Figure 5.6. As expected, the relationships between the variables shown in the composite remain similar. The relationships between the confidence related variables, STEM GPA, and Retained all remain intact. Our results on trimmed data confirm the findings of Breiman (2001), that variable importance scores may be used as a method of trimming data.

Our analysis of this case study,"Analyzing Student Data to Predict Retention", illustrates the efficacy of EBNs as a machine learning technique for small datasets. In conjunction with our empirical evaluation, our research demonstrates EBNs can be a successful model for instance classification when learned from small datasets. In

Variable	**Aggregate Importance Score**
Confidence in engineering skills	10
Confidence in problem solving skills	9.96
STEM GPA	9.95
Confidence in communication skills	9.93
Gender	9.89
Preparedness	9.83
Enrollment Management	9.67
Advanced Standing	9.51
Attitude about engineering as a professional career	9.24
HighSchool Type	9.2
Confidence in engineering's impact on society	9.15
Attitude about engineering education and study	8.86
Ethnicity	7.89

Table 5.13: Variable importance on trimmed dataset.

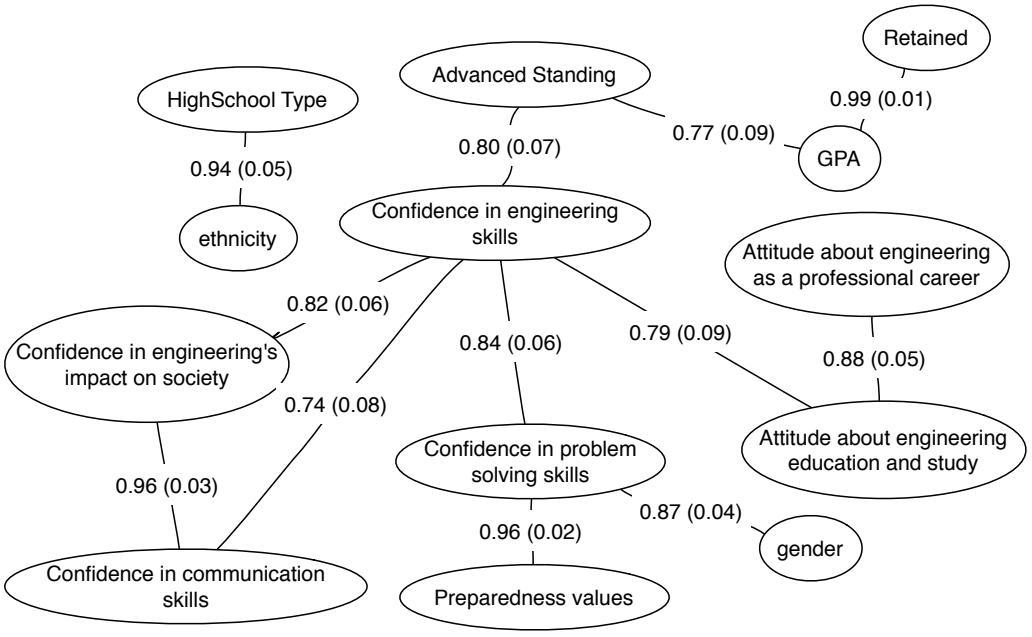


Figure 5.6: Example aggregate composite structure generated from the trimmed dataset

addition, this case study highlights the ability for EBNs to show relationships within datasets and to identify variables important to the task of classification. Identifying factors related to retention in college engineering programs is a complex task. Specifically, retention is influenced by the interaction of many factors, some of which are difficult to measure. The difficulties modeling this data are compounded by the fact that human beings often make decisions independent of the measurable data. Based on EBN's ability to be a useful classifier for retention and identify the complex relationships contributing to retention, we believe that EBNs can be successful in other domains when provided with limited amounts of training data.

Chapter 6

Conclusions and Future Work

Our research has shown Ensembled Bayesian Networks, EBNs, can be used as a model for instance classification that outperforms classification using traditional Bayesian Networks. Specifically, EBNs trained from small datasets produced statistically better results when compared with traditional Bayesian Networks learned using the POWER structure learning algorithm. EBNs' ability to not overfit on small datasets supports their use in real world problems. Because EBNs are constructed using Bayesian Networks, they can graphically represent the relationships between variables. EBNs also posses the capacity to identify variables most important to the task of classification. The ability of EBNs to identify important variables and graphically represent relationships in datasets make EBNs as a more attractive option than black box approaches like Neural Networks. With the success of our case study, which evaluated the role of various factors on retention rates in engineering education at the University of Oklahoma, we plan to pursue further research in the domain of engineering education using EBNs. We have several datasets from different institutions which we plan to analyze and compare with our findings in this paper and other literature in the field.

Our work is an example of the successful application of Random Forests principles to different underlying models. As such, we believe there exist other techniques in Machine Learning that could benefit from the same concepts that made EBNs successful. The ability to model and analyze small datasets using many of the leading algorithms in Machine Learning is often limited; however, EBNs have improved performance on small datasets. Our work, along with other similar works that build on the successes of Random Forests, will hopefully encourage similar research in other areas of Machine Learning in which the analysis of small datasets is needed.

In an effort to encourage future work and the reproduction of results, the code developed for this thesis is released under the GNU General Public License[1] and can be found on the University of Oklahoma School of Computer Science IDEA lab's

[1] `http://www.gnu.org/licenses/gpl.html`

website[2]. While outside the scope of this thesis, more analysis is required on the effect of the Bayesian Network structure learning and inference algorithms used by EBNs. Our algorithm for classification using EBNs relied on an evenly weighted combination of component scores to produce the ensembled prediction. While this approach has been shown to work well for EBNs and traditional Random Forests, other techniques for aggregation should be explored. Our experiments have also shown EBNs perform well when coupled with the POWER structure learning algorithm and standard inference. However, other algorithms for structure learning and inference could be substituted for comparison. Specifically, in this research we left out latent nodes because the algorithm used for Bayesian Network structure learning did not support them. For the addition of hidden nodes to EBNs, another structure learning algorithm that handles latent nodes could be used. We hypothesize that EBNs are a successful framework for classification using Bayesian Networks given any inference or structure learning algorithms. As advances are made in the area of Bayesian Networks, the performance of EBNs should continue to improve.

[2] `http://idea.cs.ou.edu/theses/cutz/`

Reference List

Akaike, H., 1978: A Bayesian analysis of the minimum AIC procedure. *Annals of the Institute of Statistical Mathematics*, volume 30, 9–14.

Andreassen, S., R. Hovorka, J. Benn, K. Olesen, and E. Carson, 1991: A model-based approach to insulin adjustment. *Proceedings of the Third Conference on Artificial Intelligence in Medicine*, 239–248.

Beinlich, I., G. Suermondt, R. Chavez, and G. Cooper, 1989: The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. *Proceedings of the 2nd European Conference on AI in Medicine*, Springer-Verlag, 247-256, 247–256.

Besterfield-Sacre, M., C. Atman, and L. Shuman, 1997: Characteristics of freshman engineering students: Models for determining student attrition in engineering. *Journal of Engineering Education*, volume 86, 139–149.

—, 1998: Engineering student attitudes assessment. *Journal of Engineering Education*, volume 87, 133–141.

Binder, J., D. Koller, S. Russell, and K. Kanazawa, 1997: Adaptive probabilistic networks with hidden variables. *Machine Learning*, Kluwer Academic Publishers, Hingham, MA, USA, volume 29, 213–244.

Bosch, A., A. Zisserman, and X. Munoz, 2007: Image classification using Random Forests and ferns. *Proceedings of the 11th International Conference on Computer Vision*, 1–8.

Breiman, L., 1996: Bagging predictors. *Machine Learning*, volume 24, 123–140.

—, 2001: Random Forests. *Machine Learning*, volume 45, 5–32.

Buntine, W., 1991: Theory refinement on Bayesian Networks. *Proceedings on the Seventh Conference on Uncertainty in Artificial intelligence (UAI)*, Morgan Kaufmann, Los Angeles, CA, 52–60.

—, 1996: A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, Los Alamitos, CA, USA, volume 8, 195–210.

Cheng, J., R. Greiner, J. Kelly, D. Bell, and W. Liu, 2002: Learning Bayesian Networks from data: An information-theory based approach. *Artificial Intelligence*, volume 137, 43–90.

Conati, C., A. Gertner, K. Vanlehn, and M. Druzdzel, 1997: On-line student modeling for coached problem solving using Bayesian networks. *User Modeling: Procedings of the Sixth International Conference, UM97*, Springer, 231–242.

Cooper, G. and T. Dietterich, 1992: A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, volume 9, 309–347.

Cowell, R. G., A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter, 2007: *Probabilistic Networks and Expert Systems: Exact Computational Methods for Bayesian Networks*. Springer Publishing Company.

Dempster, A. P., N. Laird, and D. Rubin, 1977: Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, volume 39 (Series B), 205–247.

Do, B., Y. Zhao, D. Trytten, and A. W. Lowe, 2006: 'Getting an internship. . . I'm still trying to find that:' Asian American student experiences obtaining engineering internships. *Proceedings of the Asian Pacific Educational Research Association*.

Edwards, W., 1998: Hailfinder: Tools for and experiences with bayesian normative modeling. *American Psychologist*, volume 53, 416–428.

Elidan, G. and S. Gould, 2008: Learning bounded treewidth Bayesian networks. *Journal of Machine Learning*.

Fast, A., 2009: *Learning the Structure of Bayesian Networks with Constraint Satisfaction*. Ph.D. thesis, University of Massachusetts Amherst.

Fast, A., M. Hay, and D. Jensen, 2008: Statistical power analysis for improved learning bayesian network structure.

Fislason, P., J. Benediktsson, and J. Sveinsson, 2006: Random Forests for land cover classification. *Pattern Recognition Letters*, volume 27, 294–300.

Foor, C. and R. Shehab, 2009: 'I feel like Forest Gump': Mixed-race Native American students find community in a college of engineering. *Proceedings of the 2009 American Society for Engineering Education Annual Conference and Exposition*, Austin, TX.

Foor, C., S. Walden, and D. Trytten, 2007: 'I wish I belonged more to this whole engineering group': Achieving individual diversity. *Journal of Engineering Education*, volume 96, 103–115.

Friedman, N., M. Goldszmidt, and D. Heckerman, 1997: Challenge: Where is the impact of Bayesian Networks in learning? *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 10–15.

Goldenberg, A. and A. Moore, 2004: Tractable learning of large Bayes net structures from sparse data. *Proceedings of the 21st International Conference of Machine Learning*.

Grossman, D. and P. Domingos, 2004: Learning Bayesian network classifiers by maximizing conditional likelihood. *Proceedings of the 21st International Conference of Machine Learning*, volume 69, 361–368.

Heckerman, D., D. Geiger, and D. Chickering, 1995: Learning Bayesian Networks: The combination of knowledge and statistical data. *Machine Learning*, volume 20, 197–243.

Imbrie, P. K., J. J. Lin, and A. Malyscheff, 2008: Artificial intelligence methods to forecast engineering students' retention based on cognitive and non-cognitive factors. *Annual Conference of American Society for Engineering Education,*.

Jensen, F. V., U. Kjærulff, K. G. Olesen, and J. Pedersen, 1989: An expert system for control of waste water treatment - a pilot project. Technical report, Judex Datasystemer A/S, Aslborg, Denmark.

Liu, F., F. Tian, and Q. Zhu, 2007: Ensembling Bayesian network structure learning on limited data. *CIKM '07: Proceedings of the sixteenth ACM conference on information and knowledge management*, Association of Computing Machinary, New York, NY, USA, 927–930.

McGovern, A., N. Hiers, M. Collier, D. Gagne II, and R. Brown, 2008a: Spatiotemporal relational probability trees. *Proceedings of the 2008 IEEE International Conference on Data Mining*, Pisa, Italy, 935–940.

McGovern, A., T. Supinie, D. J. Gagne II, N. Troutman, M. Collier, R. Brown, J. Basara, and J. Williams, 2010: Understanding severe weather processes through spatiotemporal relational random forests. *To appear at the 2010 NASA Conference on Intelligent Data Understanding*.

McGovern, A., C. Utz, S. Walden, and D. Trytten, 2008b: Learning the structure of retention data using Bayesian networks. *Proceedings of the 2008 Frontiers in Education Conference*.

Mendez, G., T. Buskirk, S. Lohr, and S. Haag, 2008: Factors associated with persistence in science and engineering majors: An exploratory study using classification trees and random forests. *Journal of Engineering Education*, volume 97, 57–70.

Pearl, J., 1988: *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers.

—, 2000: *Causality. Models, Reasoning, and Inference*. Cambridge University Press.

Piater, J. and P. Cohen, 1998: A randomized ANOVA procedure for comparing performance curves. *Proceedings of the Fifteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers, 430–438.

Provost, F. and T. Fawcett, 2001: Robust classification for imprecise environments. volume 42, 203–231.

Quinlan, R., 1993: *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.

Schwarz, G., 1978: Estimating the dimension of a model. *The Annals of Statistics*, volume 6, 461–464.

Segal, M., 2004: Machine learning benchmarks and Random Forest regression. Technical report, Center for Bioinformatics and Molecular Biostatistics.

Shehab, R., T. Murphy, J. Davidson, C. Foor, T. Reed-Rhoads, D. Trytten, and S. Walden, 2007: Academic struggles and strategies: How minority students persist. *Proceedings of the 2007 American Society for Engineering Education Annual Conference and Exposition*, Honolulu, HI.

Spirtes, P., C. Glymour, and R. Scheines, 2000: *Causation, Prediction, and Search*. The MIT Press, 2 edition.

Supinie, T., A. Mcgovern, J. Williams, and J. Abernethy, 2009: Spatiotemporal relational random forests. *Proceedings of the 2009 IEEE International Conference on Data Mining (ICDM) workshop on Spatiotemporal Data Mining.*.

Trytten, D., A. W. Lowe, and S. Walden, 2009: Racial inequality exists in spite of over-representation: The case of Asian-American students in engineering education. *Proceedings of the 2009 American Society for Engineering Education Annual Conference and Exposition*, Austin, TX.

Tsamardinos, I., L. Brown, and C. Aliferis, 2006: The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, Kluwer Academic Publishers, Hingham, MA, USA, volume 65, 31–78.

Vanlehn, K. and J. Martin, 1998: Evaluation of an assessment system based on Bayesian student modeling. *International Journal of Artificial Intelligence in Education*, volume 8, 179–221.

Walden, S. and C. Foor, 2008a: Is transfer credit a strategy for success or a prescription for failure? *American Society for Engineering Education Annual Conference*, Pittsburgh, PA.

—, 2008b: 'What's to keep you from dropping out?' – student immigration into and within engineering. *Journal of Engineering Education*, volume 97.

Walden, S. and R. Shehab, 2009: Where successful latino/a engineering undergraduates find community at a predominately white research university. *Proceedings of the 2009 American Society for Engineering Education Annual Conference and Exposition*, Austin, TX.

Wong Lowe, A., M. Flippin, J. Rogers, C. Foor, and S. Walden, 2010: *Grabbing from my Racial Toolkit: Ethnic-Racial Socialization Between and Among African-American, Asian-American, Hispanic American, and Native American Students*. Kendall/Hunt.