Contents

1 Introduction 2

2 Probability-based Machine Learning 3
   2.1 Relevance .................................................. 3
   2.2 Objective .................................................. 4
   2.3 Using Bayes’ Theorem to Predict a Target Feature .......... 4

3 The Naive Bayes’ Model 7
   3.1 The ”Naive” Assumption .................................... 7
   3.2 Prediction Example .......................................... 8
   3.3 Laplace Smoothing .......................................... 9
   3.4 Naive Bayes’ Model Implementation ......................... 10
      3.4.1 Car Insurance Data Set .................................... 10
      3.4.2 Training and Testing a Naive Bayes’ Model ............ 11
      3.4.3 Cross-validation ........................................ 13

4 The Bayesian Network Model 15
   4.1 The Bayesian Network Model ............................... 15
   4.2 Markov Blankets ............................................ 16
   4.3 Bayesian Network Model Implementation .................... 17

5 Conclusion 21

References 22
Chapter 1

Introduction

The Society of Actuaries recently published a report on the new roles that actuaries fill in the financial services industry (Sondergold & Denley, 2017). Actuaries and analytics executives submitted survey responses that described how actuaries fare in markets that increasingly demand data scientists: professionals whose specialty is to "wrangle [...] data and create meaning from it". To better understand how actuaries should adapt, the authors first addressed an important question: "what do data scientists have that actuaries don't?"

It turns out that data scientists may not have much leverage over actuaries; after all, it can be said that actuaries "wrangle" data just as well. According to one survey respondent, actuaries already "have the business knowledge and the analytical foundation to be good data scientists," while "data scientists have analytical skills, but usually no business insights to make sound decisions." Nonetheless, the survey respondents suggested that actuaries continuously improve in three areas: statistics/techniques/modeling, knowledge of business processes, and flexibility in their approach to business problems.

It is hoped that this primer will help actuarial students improve in the first of these three areas by describing the probability-based approach to machine learning as well as introducing some machine learning models.
Chapter 2

Probability-based Machine Learning

2.1 Relevance

As risk management involves evaluating possible outcomes and estimating their probabilities, the machine learning models described in this primer may be based on ideas that are familiar to students of actuarial science. Still, it is important to understand that data scientists and actuaries often differ in their approach to a problem.

Arthur Charpentier, an actuary and professor at the University of Rennes, compared the machine learning (M.L.) approach to the "standard statistical/econometric approach" in a presentation he gave to actuaries (Charpentier, 2015). It was explained that M.L. models use "data to predict some variable as a function of other covariables [sic]," but often do so without giving insight on how the prediction is made. On the other hand, the explicit objective of a statistical approach is often to gain insight on the relationships between some variables involved in a problem. Thus a data scientist may prioritize solving a problem over precisely describing it in cases that an actuary may value insight on a problem over finding its best solution.

Despite their differences, the techniques involved in one approach can supplement the other. Bayesian network (B.N.) modeling, which is described in Chapter 4, is an example of an M.L. technique that may serve actuaries well. A B.N. model makes its predictions by first representing the relationships between the variables as a graph (Kelleher et al., 2015). A B.N. might then be used to model business processes just as actuaries do. According to Martina King, CEO of the analytics company "Featurespace", such models have been used to analyze problems formerly analyzed by actuaries (King, n.d.). King describes M.L. as an "assumption-less, bottom-up approach to statistics," which proves to be valuable in a world where "modern fraud attacks are evolving at a more rapid pace than humans can imagine"
or write the rules needed to block it." She admits that these new methods may displace professionals, "however, it also presents an opportunity for those actuaries who notice the trend to gain additional skills and gain a hiring advantage over their peers." Fortunately, actuarial science students have plenty of time to prepare.

2.2 Objective

The objective of a probability-based M.L. model is no different from that of any other kind; it attempts to predict some variable in a data set as a function of other variables. Throughout this primer, a variable that is predicted by a model is referred to as a target feature (Kelleher et al., 2015). The variables that inform a model's prediction of a target feature are referred to as descriptive features (Kelleher et al., 2015). For example, if the target feature in a flower phenotype data set is "flower species", then the descriptive features may be “stem length”, “petal shape”, and “petal color”. If a probability-based M.L. model is trained on this flower phenotype data, then it should be able to accurately predict the species of an unidentified flower when given the values assigned to some descriptive features.

However, probability-based M.L. models have a distinct approach to such problems. They predict a target feature by learning and exploiting its probabilistic relationship with some descriptive features. By doing so, probability-based M.L. models have some notable strengths (Sullivan & Maki, 2013). For one, they are more powerful than deterministic methods (e.g. information-based decision trees) when provided sparse training data. Moreover, as probability-based M.L. models tend to be well-defined mathematically, their predictions can be easily interpreted (Charpentier, 2015).

2.3 Using Bayes’ Theorem to Predict a Target Feature

Bayes’ theorem is fundamental to the probability-based machine learning approach (Kelleher et al., 2015). The theorem can be defined as follows

\[ P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)} \]

Bayes' theorem is meant to describe how the post-processing probability may be defined in terms of other relevant probabilities. \( P(A \mid B) \) is the posterior probability of an event, \( A \), given some evidence, \( B \). \( P(B \mid A) \) is the conditional probability of observing \( B \) given \( A \). \( P(A) \) is the prior probability of
To continue the flower example from the past section, the theorem can be used to calculate the probability of a white-petaled flower being a daisy as follows

$$P(\text{spec} = \text{daisy} | \text{petalCol} = \text{white}) = \frac{P(\text{petalCol} = \text{white} | \text{spec} = \text{daisy}) \times P(\text{spec} = \text{daisy})}{P(\text{petalCol} = \text{white})}$$

This is to say that the probability of a white-petaled flower being a daisy is equal to the product of the probability of a daisy being white-petaled, and the probability of a flower being a daisy, all divided by the probability of a flower being white-petaled.

However, this definition of the theorem can only relate a target feature to a single descriptive feature. To investigate the relationship between multiple descriptive features and a target feature, it may be helpful to redefine Bayes’ theorem as follows (Kelleher et al., 2015)

$$P(t = v | d[1], \ldots, d[m]) = \frac{P(d[1], \ldots, d[m] | t = v) \times P(t = v)}{P(d[1], \ldots, d[m])}$$

On the left side of the equation, we have $P(t = v | d[1], \ldots, d[m])$, or the posterior probability of a target feature, $t$, being assigned a value, $v$, when given the values assigned to a set of $m$ descriptive features, $d[1], \ldots, d[m]$. On the right side of the equation, we have the product of $P(d[1], \ldots, d[m] | t = v)$, which is the conditional probability of observing $d[1], \ldots, d[m]$ given that $t = v$, and $P(t = v)$, which is the prior probability of $t = v$. This is then divided by $P(d[1], \ldots, d[m])$, or the prior probability of $d[1], \ldots, d[m]$ occurring.

By using this definition of Bayes’ theorem, the probability of a long-stemmed, white-petaled flower being a daisy can be calculated as follows

$$P(\text{spec} = \text{daisy} | \text{stemLength} = \text{long}, \text{petalCol} = \text{white}) = \frac{P(\text{stemLength} = \text{long}, \text{petalCol} = \text{white} | \text{spec} = \text{daisy}) \times P(\text{spec} = \text{daisy})}{P(\text{stemLength} = \text{long}, \text{petalCol} = \text{white})}$$

This is to say that the probability of a long-stemmed, white-petaled flower being a daisy is equal to the product of the probability of a daisy having both a long stem and white petals, and the probability of a flower being a daisy, all divided by the probability of a flower having a long stem and white petals.

The most straightforward method of using Bayes’ theorem to predict the value assigned to a target feature is to find the value in the domain of the target feature that maximizes its posterior probability. This is referred to as a MAP (maximum a posteriori) prediction (Kelleher et al., 2015). When given $d[1], \ldots, d[m]$, the MAP prediction of $t$ is just the $v$ in
domain(t) that maximizes \( P(t = v \mid d[1], \ldots, d[m]) \). A general definition for a function that returns this kind of prediction is as follows

\[
T_{MAP}(d[1], \ldots, d[m]) = \arg \max_{\nu \in \text{domain}(t)} P(t = \nu \mid d[1], \ldots, d[m])
\]

\[
= \arg \max_{\nu \in \text{domain}(t)} P(d[1], \ldots, d[m] \mid t = \nu) \times P(t = \nu)
\]

Each model described in this primer makes some form of a MAP prediction.
Chapter 3

The Naive Bayes’ Model

3.1 The "Naive" Assumption

Recall from Section 2.2 that a probability-based M.L. model "attempts to predict a target feature by learning and exploiting its probabilistic relationship with some descriptive features." In the section that followed, a MAP prediction function, which seemed to do exactly that, was defined. If a data set can be used to estimate each probability required by that function, then what is there to gain by making a prediction using a naive Bayes' model of the same data set?

The advantages follow from the naive Bayes' model's assumption about how descriptive features relate to each other. This "naive" assumption is that the value assigned to a descriptive feature can only be directly influenced by the value assigned to the target feature, and therefore a descriptive feature can only influence another by virtue of its influence on the target feature. Thus any apparent relationships between the values assigned to descriptive features are ignored when the value of the target feature is known (Kelleher et al., 2015). This is to say that descriptive features are conditionally independent given the value of the target feature.

This assumption has an impact on how a naive Bayes' model makes a MAP prediction.

\[
T_{NB}(d[1], \ldots, d[m]) = \arg \max_{v \in \text{domain}(t)} P(d[1], \ldots, d[m] \mid t = v) \times P(t = v)
\]

\[
= \arg \max_{v \in \text{values}(t)} \left( \prod_{i=1}^{m} P(d[i] \mid t = v) \times P(t = v) \right)
\]

The assumption allows for \( P(d[1], \ldots, d[m] \mid t = v) \) to be defined as a product of multiple
conditional probabilities of single events instead of a single conditional probability of a joint
event. Estimating \( m \) probabilities of the form \( P(d[i] \mid t = v) \) instead of estimating a single
probability of the form \( P(d[1], \ldots, d[m] \mid t = v) \) allows the naive Bayes’ model an advantage in
dealing with sparse and fragmented data. For example, if \( P(d[1], \ldots, d[m] \mid t = v) \) were to be
estimated without first factoring it, then the data set which informs the estimation would
require at least one instance of \( d[1], \ldots, d[m] \) and \( t = v \) occurring in conjuction for each \( v \) in
\( domain(t) \). This may not be a catastrophic issue if \( m \) is low, but it should be noted that an
increase in \( m \) translates to an exponential decrease in the number of representative samples.
This phenomenon is referred to as data fragmentation (Kelleher et al., 2015).

3.2 Prediction Example

The table below stores a data set which was fabricated for the purpose of this example. A naive Bayes’
model of this data set can MAP predict the value of \( A \) given \( B = F, C = F, D = F, \) and \( E = T \) by returning the value \( v \) in \( domain(A) \) that maximizes \( P(A = v \mid B = F, C = F, D = F, E = T) \) (while under the naive assumption). The calculation and comparison of
\( P(A = v \mid B = F, C = F, D = F, E = T) \) across \( domain(A) = \{T, F\} \) is displayed in the table at the
beginning of the next page. As \( v = F \) is found to maximize the posterior probability, \( T_{NB} \)
returns \( A = F \) as the prediction.

\[
T_{NB}(B = F, C = F, D = F, E = T) = \arg \max_{v \in domain(A)} \prod_{l=B}^{E} P(d[l] \mid A = v) \times P(A = v)
\]

\[
= \arg \max_{v \in \{T,F\}} P(B = F \mid A = v) \times P(C = F \mid A = v) \times P(D = F \mid A = v) \times P(E = F \mid A = v) \times P(A = v)
\]

<table>
<thead>
<tr>
<th>Index</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>5</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>8</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Though the assumption of conditional independence of descriptive features allows the naive Bayes’ model to withstand data fragmentation to some degree, in no way is the model immune (Kelleher et al., 2015). For instance, if the data set from the last example had contained no instance of $B = F$ and $A = F$ occurring in conjunction, the model would have estimated $P(B = F | A = F)$ to be 0. As a consequence, any product of which $P(B = F | A = F)$ was a factor would be 0, and thus the naive Bayes’ model would have predicted that $A = T$. In cases like this one, it is recommended that the probabilities used by the model are smoothed.

To smooth a probability is to perform a function on the data that is used to calculate the probability such that it is prevented from catastrophically influencing the performance of a model (Kelleher et al., 2015).

Pierre-Simon Laplace, a French scholar, ran into a similar problem when he was tasked with estimating the probability of the sun rising the next day. He was troubled by the fact that, even after observing many consecutive days of sunrises, he was unable to estimate the probability that it would fail to do so without first observing a day without a sunrise. Nonetheless, he invented the Laplace smoothing technique to come to an answer. A naive Bayes’ model that applies Laplace smoothing can be defined as follows (Kelleher et al., 2015)

$$T_L(d[1], ..., d[m]) = \arg \max_{v \in \text{values}(t)} \left( \prod_{i=1}^{m} \frac{\text{count}(d[i] | t = v)}{\text{count}(i | t = v) + (k \times | \text{Domain}(d) |)} \right) \times P(t = v)$$

$\text{count}(d[i] | t = v)$ is how often $d[i]$ occurs in conjunction with $t = v$. $k$ is a smoothing constant, which is usually a low integer (e.g. 1, 2, etc.) $\text{count}(i | t = v)$ is just the number of instances where the descriptive feature, $i$, is assigned any value in conjunction with $t = v$. $|\text{Domain}(i)|$ is the number of values in the domain of the descriptive feature.

Let $T$ be assigned to $D$ in the second row of the data set from the last example. In this case, both $P(D = F | A = T)$ and $P(D = F | A = F)$ would be 0, and the naive Bayes’ model would be unable to make an informed prediction. However, if it were to use Laplace smoothing (with a

$$P(B = F | A = T) \quad P(B = F | A = F)$$

$$P(C = F | A = T) \quad P(C = F | A = F)$$

$$P(D = F | A = T) \quad P(D = F | A = F)$$

$$P(E = T | A = T) \quad P(E = T | A = F)$$

$$P(A = T) \quad P(A = F)$$

$$P(A = T | B = F, C = F, D = F, E = T) \quad P(A = F | B = F, C = F, D = F, E = T)$$

### 3.3 Laplace Smoothing
smoothing factor of 1) to estimate $P(D = F \mid A = F)$, then it would be calculated as follows

$$\frac{\text{count}(D = F \mid A = F) + k}{\text{count}(D = \text{Tor} D = F \mid A = F) + (k \times |\text{Domain}(D)|)} = \frac{0 + 1}{5 + (1 \times |4|)} = \frac{1}{9}$$

Note that the new estimate is defined by values that are relevant to the descriptive feature (i.e. the size of the feature’s domain, the number of times the feature is assigned to any value, etc.). By applying Laplace smoothing, the probabilities considered by the model are guaranteed to be non-zero without disrupting any important statistical trends, as changes to non-zero probabilities will be relatively tiny, and changes to zero probabilities will only be as large as they are required to prevent model failure.

### 3.4 Naive Bayes’ Model Implementation

#### 3.4.1 Car Insurance Data Set

The models implemented in Section 3.5 and Section 4.3 are trained on a car insurance data set from Piet de Jong’s, “Generalized Linear Models for Insurance Data.” It stores values of ten variables that describe one-year vehicle insurance policies taken out in 2004 and 2005. Piet de Jong’s variable descriptions are as follows

- "veh_value": The value of vehicle in 10,000s
- "exposure": 0-1
- "clm": The occurrence of claim (0 = no, 1 = yes)
- "numclaims": The number of claims
- "claimcst0": The claim amount (0 if no claim)
- "veh_body": vehicle body, coded as "BUS", "CONVT" = convertible, "COUPE", "HBACK" = hatchback, "HDTOP" = hardtop, "MCARA" = motorized caravan, "MIBUS" = minibus, "PANVN" = panel van, "RDSTR" = roadster, "SEDAN", "STNWG" = station wagon, "TRUCK", "UTE" = utility
- "veh_age": The age of vehicle: 1 (youngest), 2, 3, 4
- "gender": The gender of driver: M (male), F(female)
- "area": The driver’s area of residence: A, B, C, D, E, F
• "agecat": The driver's age category: 1 (youngest), 2, 3, 4, 5, 6

It should be noted that it is not necessary to classify every feature of a data set as either a target feature or a descriptive feature; the distinction of a target feature and its descriptive features from each other, and any other features of a data set, is based on a value judgment. For example, as claims filed by policy owners are potentially costly for car insurers, an actuary employed by a car insurance provider should consider classifying “clm” as the target feature. And if he or she decides to do so, then the descriptive features should be classified with respect to that decision. The models implemented in this section, and in Section 4.4, are built to predict "clm" as a function of all other features except for "numclaims" and "claimcst0." These two features are removed from consideration because they are tightly coupled to "clm" by definition.

3.4.2 Training and Testing a Naive Bayes’ Model

This section describes the R commands that can be used to build a naive Bayes’ model of the car insurance data set. The commands in each section may be compiled and run from a single “.rmd” file, or performed individually in the console provided by RStudio. RStudio is an open source development environment which includes a console, code editor, debugging support, and visualization tools.

A great resource for those that need an introduction to using R for M.L. purposes is listed in the Conclusion.

```
install.packages("e1071")
install.packages("caret")
library(e1071)
library(caret)
```

First, the R program installs and imports two packages made available by RStudio. The “e1071” package provides access to functions that will be used to build the model. The “caret” package provides a function that will be used at the beginning of the cross-validation process, which is explained in the next subsection.

```
car = read.csv("car.csv")
```

In the line of code above, "read.csv()" reads the content of "car.csv" into a data frame – R's built-in, table-like data structure – named "car".

By placing the file that contains the car insurance data into the same folder as the program, its name can be directly passed to “read.csv().” Otherwise, its file path must be passed.
"numclaims" and "claimcst0" are removed from consideration by executing the above command, which produces a copy of "car" with columns 4 and 5 omitted. The same can be done to remove any other features.

Execution of this code ensures that "clm", "veh_age", "veh_body", "gender", "agecat", and "area" are treated as categorical features by R.

As "veh_value" and "exposure" can take a wide range of values, they cannot be immediately considered as categorical features by R. However, "cut()" can be used to divide the domain of a feature into intervals that serve as a basis for categorizing it. For example, this program separates the domain of "veh_value" into four intervals and labels each interval as "carcat1", "carcat2", "carcat3", and "carcat4."

Execution of the above lines of code will split the car insurance data set into a training set and a testing set. As their names suggest, a training set is used to build a model while a testing set is used to evaluate its performance. This is a form of cross-validation, which will be defined in the next section (Schneider 1997).
sets. A training set can be reproduced by passing the same parameter (e.g. 42) to "set.seed()" before sampling from the original set. On the second line of code above, "sample()" uses R's built-in random number generator to designate 3/4 of the rows in "cardat" as members of the training set.

On the third line, the rows that correspond to the numbers in "train_index" are copied into "car_trn". The fourth line assigns the remainder to "car_tst".

```
1 nb = naiveBayes(clm ~ ., data = car_trn, laplace = 1)
2 nb_pred_tst = predict(nb, car_tst)
```

Here a naive Bayes’ model is trained and then assigned to “car_nb”. The first argument that is passed to “naiveBayes()” is of the form “<target feature> <descriptive features>". Passing "clm ." to the function tells the naive Bayes’ model that “clm” is the target feature and that all other features should be considered as descriptive features ("." stands for all other features). The second argument names the data table on which the model should train. The third argument is the smoothing factor (usually 1).

The second line produces a set of predictions of the values assigned to "clm" in the testing data set and then stores them in "car_nb_test_pred".

```
1 accuracy = function(actual, predicted) 
2 {
3     mean(actual == predicted)
4 }
```

The above code defines a function which will be used to assess the performance of the model in the next step.

```
1 accuracy(actual = car_tst$clm, predicted = nb_pred_tst, )
```

The accuracy function that was defined in the last step can now be used to evaluate the model’s performance (our trained model made the correct prediction 93.16% of time).

### 3.4.3 Cross-validation

Cross-validation describes any model evaluation method that involves removing some portion of a data set for the purpose of training a model (Schneider 1997). Separating the original data set into a training and a testing set (i.e. the hold-out method) is one form of cross-validation. Though this method can yield a fair measurement, there is an alternative which may do a better job.

The k-fold cross-validation method improves on the hold-out method by performing it multiple times (k-times), and then returning the average of the results (Schneider, 2017). This
is to make sure that the model's measured accuracy is independent of how its training and testing sets were sampled.

```r
folds = caret::createFolds(cardat$clm, k = 5)

nb_fold_acc = rep(0, length(folds))
```

First, the “createFolds()” function partitions the rows of "cardat" into five parts, and stores each part as a list of indices in “folds”.

In the second line, the “rep()” function creates a vector of the same size as “folds” and stores it in a variable “nb_fold_acc”.

```r
for (i in 1:length(folds))
{
  # split for fold i
  train_fold = cardat[ folds[[i]],]
  validate_fold = cardat[ folds[[i]],]

  # accuracy for fold i
  nb_fold = naiveBayes(clm ~ ., data = train_fold, laplace = 1)
  nb_fold_pred = predict(nb_fold, data = validate_fold, type = "class")
  nb_fold_acc[i] = accuracy(actual = validate_fold$clm, predicted = nb_pred)
}
```

It is recommended that readers become familiar with "for loops" to understand the code above.

The for loop above executes the code within it's brackets "length(folds)"-times (5-times in this case). The following operations are performed over the course of each loop. First, a training fold that stores the data contained by "cardat", except for one of the five partitions (the i-th fold in "folds"), is generated. This is stored in “train_fold”. Second, the remaining partition is stored in “validate_fold”. Third, a naive Bayes’ model is trained on “train_fold”. Fourth, that model is tested on “validate_fold”. Finally, its accuracy is stored in the i-th cell in “nb_fold_acc”.

```r
mean(nb_fold_acc)
```

Finally, the mean of the accuracies stored in “nb_fold_acc” is calculated and displayed (our mean success rate came out to be 93.18%).
Chapter 4

The Bayesian Network Model

4.1 The Bayesian Network Model

A B.N. model can assume the conditional independence of descriptive features just as a naive Bayes’ model does. However, unlike a naive Bayes’ model, a B.N. model can be selective in its assumptions. It accounts for structural relationships (i.e. direct influence and conditional independence) between features by embedding them in a tree-like, graphical representation such as the one pictured below (Kelleher et al., 2015).

Each node in a B.N. represents a feature of the data it models. Each edge represents the supposed influence of one feature on another. To easily discuss the hierarchy of influence in a B.N., terms such as "parent" and "child" are used. For example, in the B.N. above, node A is said to be a parent of node B (and node B a child of node A), because the value of feature B is supposed to depend on the value of feature A (as indicated by the direction of the edge which connects the two) (Kelleher et al., 2015). It is not within the scope of this primer to discuss methods of generating a B.N.’s structure, but common methods include the K2 algorithm,
force naive Bayes', the genetic algorithm, and the Hill-Climbing algorithm (Heaton, 2013). The algorithms are listed in no particular order, but the K2 algorithm happens to be most commonly used.

A B.N. defines these relationships by associating each node with a **conditional probability table** (C.P.T.) (Kelleher et al., 2015) For example, the C.P.T. associated with node $D$ might look like the table below.

<table>
<thead>
<tr>
<th>$B$</th>
<th>$C$</th>
<th>$P(D = T \mid B, C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>0.4</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
<td>0.4</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
<td>0.2</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Methods such as Bayes' model averaging, multi-nominal Bayes' model averaging, and simple probability estimation can be used estimate the conditional probability distributions associated with each node (Heaton, 2013).

### 4.2 Markov Blankets

The way in which conditional independence is encoded in the structure of a B.N. can be understood by considering an example of a **Markov blanket** (Kelleher et al., 2015). A node's Markov Blanket is the set composed of its parents, children, and parents of its children (other than itself). In the network pictured below, if the red-colored node represents the target feature, and the orange-colored nodes make up the red-colored node's Markov blanket, then the remaining gray-colored nodes represent features that are conditionally independent of the target feature given the values assigned to the Markov blanket.

![Markov Blanket Diagram](image-url)
As it does for a naive Bayes’ model, accounting for conditional independence has an impact on how a B.N. makes its predictions.

\[
T_{BN}(d[1],...,d[m]) = \arg \max_{v \in \text{domain}(t)} P(t = v \mid d[1],...,d[m])
\]

\[
= \arg \max_{v \in \text{domain}(t)} P(t = v \mid \text{Parents}(t)) \times \prod_{j \in \text{Children}(t)} P(d[j] \mid \text{Parents}(j))
\]

Similar to \(\text{Parents}(t)\), \(\text{Children}(t)\) is just the set containing the children of \(t\). By defining \(P(t = v \mid d[1],...,d[m])\) in terms of the values assigned to \(t\)'s Markov blanket, a B.N. model is compact like the naive Bayes' model. This can be demonstrated by exploiting a Markov blanket in the B.N. depicted at the beginning of the chapter. As the Markov blanket of node \(D\) is the set of nodes containing \(D\)'s parents (\(B\) and \(C\)) and \(D\)'s child (\(E\)), then \(P(D = T \mid A = T, B = F, C = F, E = T)\) can be defined as follows

\[
P(D = T \mid A = T, B = F, C = F, E = T) = P(D = T \mid \text{Parents}(D)) \times \prod_{j \in \text{Children}(D)} P(j \mid \text{Parents}(j))
\]

Note that the final calculation did not require knowledge of the value assigned to \(A\). This is because \(A\) and \(D\) are assumed to be conditionally independent given the value assigned to \(B\), a member of \(D\)'s Markov blanket. As \(A\) can only influence \(D\) by virtue of its influence on \(B\), \(A\)'s value can be ignored.

### 4.3 Bayesian Network Model Implementation

```r
install.packages("bnlearn")
source("https://bioconductor.org/biocLite.R")
library(bnlearn)
biocLite(Rgraphviz)
```

First, the program installs and imports two packages. "bnlearn" provides functions that fit the B.N. model to the insurance data. "Rgraphviz" provides functions that are used to visualize the model.
In this case, the data is prepared as it was in the naive Bayes' model implementation. Remember that it is crucial to set the same seed value before producing a training set.

```
res = hc(car_trn)
```

```
fittedbn = bn.fit(res, data = car_trn)
```

“`car_trn`” is passed to “`hc()`,” which solves for the structure of the B.N. The result is stored in “`res`”.

Using the structure stored in “`res`” as a guide, the function “`bn.fit()`” solves for the C.P.T. associated with each node. Once this process is finished, the completed B.N. is stored in "`fittedbn`." If a console is being used, "`fittedbn`" can be entered to take a look at the C.P.T. for each node in the B.N.

```
bn_pred_tst = predict(fittedbn, "clm", car_tst)
```

As it did with the naive Bayes' model, "`predict()`" uses "`fittedbn`" to predict the values assigned to "`clm`" in "`car_tst`.”

```
accuracy(actual = car_tst$clm, predicted = bn_pred_tst)
```

The model's success rate is calculated according to the accuracy function defined in the naive Bayes' model implementation (our B.N. model happened to have the same success rate as our naive Bayes' model).

```
graphviz.plot(fittedbn, highlight = NULL, layout = "dot",
shape = "circle", main = NULL, sub = NULL)
```

Finally, "`graphviz.plot()`" visualizes the B.N.'s structure.
The structure of this B.N. suggests that the probability of a claim being filed by a policy owner ("clm") is directly influenced by the insured vehicle's exposure to loss ("exposure"). Though this is certainly accurate, it is not surprising. To gain more subtle insight on a data set, it may be best to remove a descriptive feature from consideration in cases where it is found to be so tightly coupled with the target feature. By doing so, the model is forced to relate the target feature to descriptive features it had previously neglected.

The body type of a policy owner's vehicle ("veh_body") seems to have an influence on the vehicle's age ("veh_age"), the gender of its driver ("gender"), the driver's area of residence ("area"), as well as his or her age category ("agecat"). Despite their being a correlation between these features, it would be too much to say that an insured vehicle's body type has a causal relationship with its age or any of its driver's qualities.
By removing "exposure" from consideration, another B.N. structure can be produced. This B.N. suggests that the occurrence of a claim ("clm") is dependent on the age category of the driver ("agecat"). This insight is more subtle than that of the relationship between exposure to loss and occurrence of claim but should be just as familiar to academics (Kelly & Nielson, 200) and car insurers alike. Considering that the rate of accident-related deaths per mile among teens is much higher than that of adults, this B.N. may be on to something ("Why Some Drivers Pay Less", n.d.).
Chapter 5

Conclusion

This primer was intended to acquaint actuarial science students with the probability-based approach to M.L. However, in no way does it exhaustively describe its topic. Fortunately, recent movements to make M.L. instruction more accessible have produced countless free and paid resources that are meant to do so.

- While working on this primer, we consulted a textbook published by MIT Press, "Fundamentals of Machine Learning for Predictive Data Analytics" by Kelleher et al.

- MIT generously provides free course material from "Prediction: Machine Learning and Statistics" (as taught by Prof. Cynthia Rudin in the spring of 2012) through its web-based publication, MIT OpenCourseWare. (https://ocw.mit.edu/). The notes for Lecture #2 are a great introduction to using R for machine learning purposes.

- An important but perhaps understated component of data analysis is the data preparation process. We looked to "An introduction to data cleaning in R" by Jonge et al. for help in this area.
Bibliography


