Praktikum 1: Bayes Netze

Mike Siefert, Maximilian Neudert

Praktikum 1: Bayes Netze

Machen Sie sich mit der Modellierung von Bayes-Netzen mittels des R-Packages bnlearn vertraut.

Teilaufgabe a: Car Evaluation - Naive Bayes | Bayes Netze

Laden Sie den Car Evaluation Data Set von https://archive.ics.uci.edu/ml/datasets/Car+Evaluation herunter. Klassifizieren Sie mit dem R-Package bnlearn den Datensatz anhand der "car acceptability". Implementieren Sie dazu zum einen einen naiven Bayes Klassifizierer und zum anderen ein Bayes Netz, das mit Hilfe des TAN Verfahrens erstellt wurde. (jeweils mit bnlearn)

Merkmal	Beschreibung	Skalenniveau	Beispiel
buying	Kaufpreis	ordinal	"low","med","high","vhigh"
maint	Wartungskosten	ordinal	"low","med","high","vhigh"
doors	Anzahl der Türen	ordinal	"2","3","4","5more"
persons	Sitzplätze	ordinal	"2","4","more"
lug_boot	Größe des Gepäckraums	ordinal	"small","med","big"
safety	Geschätze Sicherheitseinstufung	ordinal	"high","low","med"
Target	Autoakzeptanz	ordinal	"unacc", "acc", "good", "vgood"

Table 1: Car Evaluation Data Set: Übersicht der Merkmale

Zunächst verschaffen wir uns einen Überblick über den zugrundeliegenden Datensatz und der gegeben Merkmale. Die Merkmale sind in Tab.1 zu entnehmen. Alle Merkmale weisen ein ordinales Skalenniveau auf.

```
#>   buying maint doors persons lug_boot safety Target
#> 1  vhigh vhigh     2       2    small    low  unacc
#> 2  vhigh vhigh     2       2    small    med  unacc
#> 4  vhigh vhigh     2       2      med    low  unacc
#> 5  vhigh vhigh     2       2      med    med  unacc
#> 6  vhigh vhigh     2       2      med   high  unacc
```

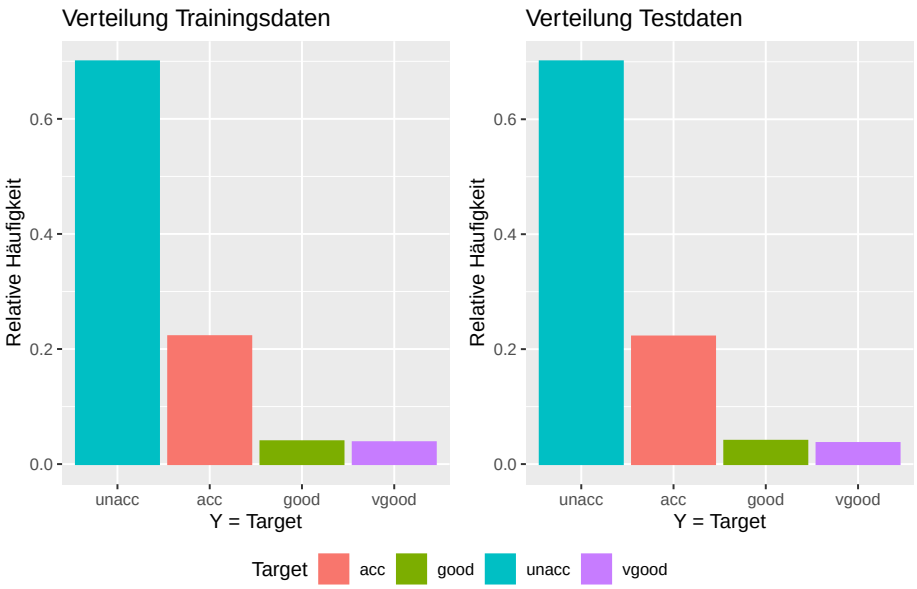
```
#> 7  vhigh vhigh     2       2      big    low  unacc
```

Das Merkmal **buying** spiegelt den Kaufpreis wieder. Hierbei wird von einem niedrigen Kaufpreis *low* bis zu einem sehr hohen Kaufpreis *vhigh* unterschieden. Neben den Anschaffungskosten (Kaufpreis) entstehen über die Zeit weitere Kosten, diese werden im Merkmal **maint** erfasst. Dabei stellt *low* geringe Wartungskosten dar und im Kontrast hierzu *vhigh* sehr hohe Wartungskosten. Hinsichtlich der mobilen Beschaffenheit des Autos lassen sich die Merkmale **doors** (Anzahl der Türen), **persons** (potenzielle Sitzplätze) und **lug_boot** (Stauraum im Kofferraum) heranziehen. Das Merkmal **safety** repräsentiert eine geschätze Sicherheitseinstufung des Autos, welche zwischen *low* und*high* liegt. Die Response Variable bzw. Zielvariable **Target** repräsentiert die Akzeptanz gegenüber diesem Auto, welche von *unacc* (nicht akzeptiert) über *acc* (akzeptiert) und *good* (gute Akzeptanz) bis schließlich *vgood* (sehr gute Akzeptanz) reicht.

Wir interessieren uns nun für die gemeinsame Verteilung der Daten. Mit Hilfe dieser können wir mittels Inferenz neue Beobachtungen klassifizieren, somit wäre es zum einen Interessant zu ermitteln, mit welcher Wahrscheinlichkeit ein bestimmtes Auto akzeptiert wird, wenn dieses bestimmte Eigenschaften erfüllt. Hierfür ziehen wir zwei Ansätze heran ***Naive Bayes*** und die ***Bayes Netze***.

Für eine nachfolgende vereinfachte Darstellung sei $buying = X_1$,$maint = X_2$,$doors = X_3$,$persons = X_4$,$lug\_boot = X_5$,$safety = X_6$ und $Target = Y$. Weiterhin splitten wir den Datensatz in einen Datensatz zum Trainieren (70%) und einen Datensatz zum Testen der Modell Güte (30%) auf.

Wir interessieren uns für die Wahrscheinlichkeit $P(h_{acc}|X_1 = a_1, ..., X_6 = a_6)$ die angibt, ob das Auto unter gegeben Beobachtungen akzeptiert wird. Hierbei orientieren wir uns an der Herangehensweise der Bayesianer, welche die gegeben Beobachtungen als fix betrachten. Wir interessieren uns somit viel mehr für die Verteilung der zugrunde liegenden Parameter unter gegeben fixen Daten $a_{1,i}, ..., a_{6,i} \in \mathcal{D}$ mit i=1,. . . ,n. Dabei bildet H den Hypothesen-Raum aller möglichen Parametern ab. Gesucht ist nun $P(h_i|a_1, ..., a_6)$, jedoch wissen wir nur $P(a_1, ..., a_6|h_i)$. Wir interessieren uns für $P(h_i|a_1, ...a_6)$, wobei für die Hypothesen h_i mit $i \in \{$(1:unacc), (2:acc), (3:good), (4:vgood)$\}$ betrachtet werden. Daher wir suchen die Parameter die, die größte ***Maximum A Posteriori Wahrscheinlichkeit*** (MAP) aufweisen. Ein erster Blick auf die relativen Häufigkeiten der Zielvariable des vorliegenden Datensatzes fällt auf, dass etwas 70 % der Beobachtungen bei gegebenem Merkmal Target *unacc* vorzeigten, somit das Auto nicht akzeptiert haben.



Mittels des Satzes von Bayes (vgl. Gl. 1) erhalten wir die ***a posteriori Wahrscheinlichkeit*** $P(h_i|a_1, ..., a_6)$. Weiterhin stellt $P(a_1, a_2, a_3, a_4, a_5, a_6|h_i)$ unseren Likelihood dar und $P(h_i)$ unsere ***a priori Wahrscheinlichkeit***.

$$P(h_i|a_1, ..., a_6) = \frac{P(a_1, a_2, a_3, a_4, a_5, a_6|h_i) \cdot P(h_i)}{P(a_1, a_2, a_3, a_4, a_5, a_6)} \tag{1}$$

Das besondere hierbei ist, dass wir zusätzliche Informationen über die a priori Wahrscheinlichkeit dem Modell beifügen können. Die gemeinsame Verteilung der Beobachtungen $P(a_1, a_2, a_3, a_4, a_5, a_6)$ im Nenner, dient als Normalisierung. Diese lässt sich mit Hilfe des Satzes der totalen Wahrscheinlichkeit, wodurch der Divisor die gewichteten Likelihood über alle möglichen Parameter abbildet, berechnen. Hierfür benötigte man alle kombinatorischen Möglichkeiten daher eine hinreichend große Menge an Daten, welche alle kombinatorischen Möglichkeiten abdecken würde. Für den binären Fall würde sich der Parametersuchraum auf 2^n ausweiten - exponentielles Wachstum! Für unseren Fall entspreche dies $4^3 \cdot 3^3 = 576$ Variationen, was zwar noch hinreichenend klein ist, jedoch in der Praxis mit hinzunahme weiterer Zufallsvariablen zu sehr langen Berechnungszeiten führen kann.

Neben der Alternativen die tatsächliche gemeinsame Verteilung zu berechnen, bilden zwei weitere Ansätze eine Möglichkeit die Verteilung zu approximieren. Die *Naive Bayes* (NB) Methode (vgl. Gl.2) geht vereinfacht davon aus, dass bestimmte Kombinationen wahrscheinlich weniger häufiger auftreten und daher ignoriert werden können, wodurch wir eine approximative Verteilung erhalten. Diese Methode nimmt also an, dass die Kovariaten $a_1, ..., a_6$ voneinander unabhängig sind (harte Annahme!).

$$h_{MAP} = \underset{h_i \in H}{\operatorname{argmax}} = \prod_{i=1}^{6} P(a_1, ..., a_6 | h_i) \cdot P(h_i) \tag{2}$$

Dieser Ansatz ist weniger rechenintensiv. Nachteilig ist hierbei, dass es sich um eine Approximation handelt. Dennoch erfreut sich dieser vereinfachte Ansatz besonderer Beliebheit in der Praxis aufgrund seiner Erklärbarkeit und guten Performanz.

Neben diesen beiden extremen Herangehensweisen, vollständige Berücksichtigung der Abhängigkeitstrukturen und den kompletten Verzicht jener, stellen die *Bayes Netze* eine besonders schönen Mittelweg dar.

Dieser Ansatz verfolgt das Ziel eine multivariate Verteilung graphisch darzustellen. Da ebenfalls wie Entscheidungsbäume, Graphen eine gute Grundlage bieten, um komplexere Sachverhalte wiederzugeben, ist auch dieser Ansatz beliebt in der Praxis. Es handelt sich hierbei um einen azyklischen gerichteten Graph (DAG), dessen Knoten Zufallsvariablen $V = \{X_1, ..., X_n\}$ repräsentieren und Kanten $E = \{(X_i, X_j) | X_i, X_j \in Vi, i \neq j\}$, welche zwischen diesen Knoten Abhängigkeitsstrukturen darstellen. Das besondere hierbei ist, dass lediglich die Annahme getroffen wird, dass der Zustand eines Knoten lediglich von seinen direkten Vorgänger (Elternknoten) abhängt (vgl. Gl. 3). Man bezeichnet diese Annahme auch als ***Global Semantics***.

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i | Parents(X_i)) \tag{3}$$

$$P(X_i | Parents(X_i)) = P(X_i | X_1, ..., X_{i-1}) \tag{4}$$

Die gemeinsame Verteilung lässt sich mittels der Kettenregel (engl. chain rule) ermitteln, indem man die Wahrscheinlichkeiten vom Wurzelknoten bis zum Zielknoten multipliziert (vgl. Gl. 4). Dies hat zwei wesentliche Vorteile. Zum einen ermöglicht dies uns bestimmte Abhängigkeiten in unser Modell fließen zu lassen, zum anderen aber auch, dass wir hierdurch eine Reduzierung der vorherige Laufzeitkomplexität von $O(2^n)$ zu $O(n \cdot 2^k)$ gelangen. Hierbei steht k für die Anzahl der Elternknoten, da die meisten Domänen nur bestimmte Abhängigkeitsstrukturen aufzeigen gilt in der Regel $k < n$. Die Modellierung hängt hierbei ganz vom Interessenstandpunkt ab. Neben der kausalen Wissensmodellierung (engl. causal knowledge), welche oftmals durch klare Abhängigkeitsstrukturen über Domänen Experten bestimmt werden. Gibt es auch die diagnotische Wissensmodellierung (engl. diagnostic Knowledge), welche Abhängigkeitsstrukturen berücksichtigt, welche das Modell ziemlich genau machen, jedoch schwieriger zu erfassen sind, da diese Strukturen oftmals nicht direkt ersichtlich sind.

$$\mathbf{P}(\mathcal{M}|\mathcal{D}) = \mathbf{P}(\mathcal{G}, \Theta|\mathcal{D}) = \mathrm{P}(\mathcal{G}|\mathcal{D}) \quad \cdot \quad \mathrm{P}(\Theta|\mathcal{G}, \mathcal{D}) \tag{5}$$

Das Modell $\mathbf{P}(\mathcal{M}|\mathcal{D})$ wird somit über zwei Parameter $\mathcal{G}$ und Θ definiert (vgl. Gl.5). Die graphische Struktur $\mathcal{G}$ ergibt sich durch die a posteriori Wahrscheinlichkeit $\mathrm{P}(\mathcal{G}|\mathcal{D})$ (engl. structure learning). Diese Struktur wird benötigt, um letztlich die lokalen Wahrscheinlichkeitsverteilungen Θ zu ermitteln, welche sich über die Daten und die Struktur des Netzes, $\mathrm{P}(\Theta|\mathcal{G}, \mathcal{D})$, ergeben. Neben den ***Global Semantics*** existieren noch die ***Local Semantics***. Während die global Semantics, das gesamte Netz als eine gemeinsame Verteilung betrachten, welche über das Produkt lokaler bedingter Verteilungen im Netz ermittelt wird, so gewährleisten die locale Semantics, dass diese bedingt Unabhängig von einader sind.

Hierbei stellen die Markov Blankets eine äußerst hilfreich Möglichkeit bereit, indem Sie eine Menge von Knoten definieren, welche die zu betrachtende Zufallsvariable als eine Folge bedingt unabhänger Zustände modelliert. So wird die Wahrscheinlichkeitsverteilunge eines einzelnen Knoten lediglich durch sein Markov Blanket bestimmt. Diese hängen neben der topologischen Struktur des Modells (Reihenfolge der gewählten Zufallsvariablen im Graph) auch von dem Interessenstandpunkt (causal vs. diagnostic knowledge) ab, da dies die Anzahl der Kanten bestimmt. Das Lernen des Modells erfolgt somit über zwei Schritte (1) Finde eine geeignte Netz-Struktur $\mathcal{G}$, (2) ermittle lokale Wahrscheinlichkeitsverteilungen Θ mittels der Netz-Struktur $\mathcal{G}$.

Das R-Package *bnlearn: "Bayesian network structure learning, parameter learning and inference"* stellt das passende Werkzeug für unser weitere Vorgehen bereit. Das strukturelle Lernen des Bayes Netzes erfordert einen Trade-Off, welcher Abhängig vom Interessenstandpunkt ist. Das Lernen eines optimalen Bayes Netzes ist NP-schwer (nicht deterministische Polynomialzeit). Wir werden hierfür bestimmte Varianten des strukturellen Lernen beobachten und vergleichen. NB wird uns eine äußerst schlichte Netz-Struktur bieten. Eine weitere Variante, welche weniger rechenintensiv ist, ist der sogenannte ***Tree-Augmented Naive Bayes*** (TAN). Diese kann als eine Erweiterung des Naive Bayes Ansatzes gesehen werden. Es wird zunächst ein Naive Bayes Modell erzeugt (Grundgerüst des Netzes). Die Kovariaten sind untereinander unabhängig. Nun wird für das strukturelle Lernen des Netzes basiert auf den *Chow-Liu* Ansatz zurückgeriffen, welcher einen ***Maximum Weight Spanning Tree*** (MWST) erzeugt. Hierbei wird *jeweils* ein DAG für die Kovariaten erzeugt, welcher sich immer für die größtmögliche gewichtete Kante zwischen diesen entscheidet und den Graph dabei azyklisch lässt. Die Approximation veranlasst, dass ausgegangen vom Klassen Knoten (Wurzelknoten) alle direkten Knoten (Kinder) maximal ein Elternknoten der anderen Kinderknoten erhalten. Das bedeutet, dass wir lediglich zwei Kovariaten zu lassen, welche einen direkten Einfluss auf den betrachteten Knoten haben (Wurzelknoten + 1 Nachbar) - wodurch maximal zwei Elternknoten aus dem Modell folgen. In *bnlearn* ist der TAN Ansatz implementiert in der Klasse **tree.bayes**. Besonders hervorzuheben sind die Parameter ***whitelist*** und ***black-***

list. Diese ermöglichen uns Domänen-Wissen zu strukturellen Abhängigkeiten mit ins Modell fließen zu lassen. Über die *whitelist* werden Beziehungen definiert, welche im Modell vorkommen müssen. Das Pendant ist die *blacklist*, welche gezielt bestimmte Beziehungen ausschließt. Die a priori Wahrscheinlichkeit wird über den Parameter ***prior*** gesteuert. Neben diesen beiden eher approximativen Varianten fürs strukturelle Lernen des Netzes, implementiert *bnlearn* viele weitere Algorithmen, welche sich hauptsächlich in 3 Kategorien zusammenfassen lassen.

- **Constraint-Based Learning**: Nutzen bedingte Unabhängigkeitstests und nehmen an, dass die Daten den perfekten DAG widerspiegeln.
- **Score-Based Learning**: Erzeuge mehrere DAG. Jeder DAG wird bewertet. Optimierungsproblem: Ziel maximiere Bewertungsfunktion f(x) und wähle DAG, welcher f(x) maximiert.
- **Hybrid Learning**: Nutzt bedingte Unabhängigkeitstests, um kleinere Menge an bedingten Unabhängigkeiten (Kanten) zu ermitteln. Dies reduziert den Suchraum für eine Score-based Variante.
- **Pairwise Mutual Information**: Approximieren ein Netz, indem Sie paarweise den Informationsgewinn ermitteln.

Constraint-Based Learning ist demnach stark abhängig von der Wahl des bedingten Unabhängigkeitstest. Diese konvergieren meist langsamer als Score-based und Hybrid Verfahren, da Sie alle Variablen Kombinationen durchgehen müssen. Das ständige betrachten einzelner Teilmengen des Datensatz macht diese Alogrithmen dennoch sehr Memory-Effizient. Darüberhinaus lässt sich diese Prozedur gut parallelisieren, womit eine gute Skalierbarkeit einhergeht. Das **Score-Based Learning** konvergiert schneller, erfordert jedoch Heuristiken, da Suchraum meist zu groß ist. Dadurch ist ein globales Maximum nicht garantiert. **Hybrid Learning** nutzt Methoden beider Vorgänger, so werden bedingte Unabhängigkeitstests gewählt, um die Grundlage für das Netzwerk (d.h. Restriktion der Anzahl möglicher Kandidaten) zu schaffen. Dann wird eine Score-Funktion definiert, welche es zu maximieren gilt. In die letzte Kategorie der **Pairwise Mutual Information** Algorithmen fällt auch der TAN Algorithmus. Hier werden paarweise Vergleiche der Kanten herangezogen mit dem Ziel den Informationsgewinn zu maximieren.

Nachdem wir die Struktur erlernt haben, können wir die lokalen bedingten Wahrscheinlichkeitsverteilungen schätzen. Hierfür bietet *bnlearn* die **bn.fit** Methode, welche diese dann aus den gegebenen Daten und der ermittelten Abhängigkeitsstruktur schätzt. Der Parameter *method* gibt uns die Möglichkeit die lokalen bedingten Wahrscheinlichkeiten mittels **Maximum Likelihood** (ML) festzulegen oder **Bayes**, welches die Schätzung auf Grundlage mehrer Parameterschätzungen festlegt. Wir werden den bayesianischen Weg gehen und uns auf die **Maximum A Posteriori** Wahrscheinlichkeit (MAP) verlassen. Die allgemeine Vorgehensweise hierfür wäre wie folgt.

1. Lege a priori Wahrscheinlichkeit des Parameters $P(\theta)$ fest

2. Wähle einen beliebigen Parameter θ (z.B. Hypothese {acc, unacc})
3. Füge Parameter in generatives Modell ein (z.B. Likelihood $P(D|\theta_h = acc)$)
4. Simuliere Daten und schätze Verteilung des Parameters
5. Wähle die Hypothese, welche a posterori Wahrscheinlichkeit maximiert (MAP)

Je mehr Beobachtungen wir haben desto weniger wichtig wird die vorherige Information (prior) im Modell. Umgekehrt je weniger Beobachtungen wir haben, desto stärker fällt die a priori Wahrscheinlichkeit ins Gewicht.

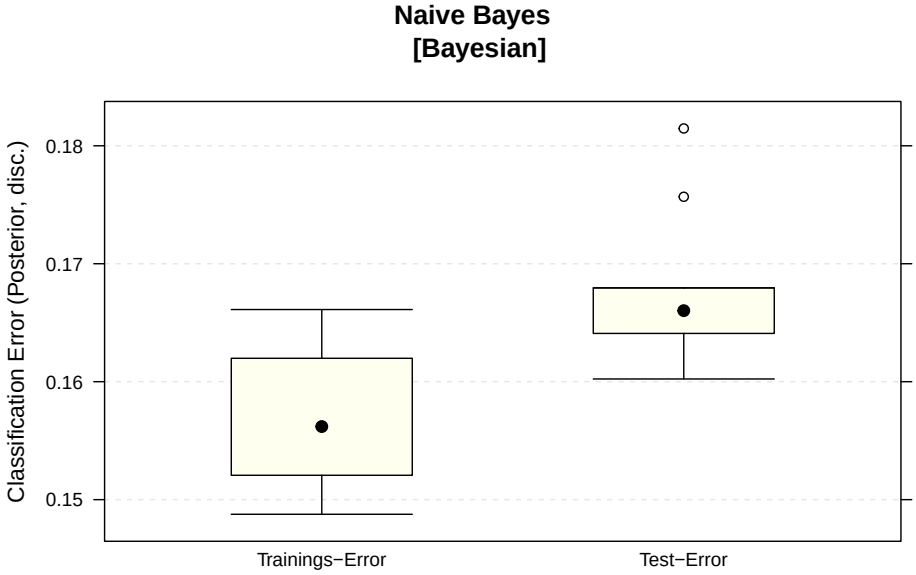
Experimentell werden wir nun diese beiden erläuterten Modelle NB und TAN evaluieren. Dabei werden wir einmal mit und einmal ohne den Einfluss von Domänen Wissen das TAN evaluieren[1].

Um eine relativ genaue Fehlerabschätzung zu erhalten wenden wir die *K-Fold Cross-Validation*(CV) an. Wir wählen für $K = 10$ Teilmengen, wobei wir diese 10 mal durchführen, um eine gute approximative Verteilung des Fehlers zu erhalten. Wir verwenden hierbei die Loss-Funktion *Posterior Classification Error*.

[1]Es sei angemerkt, dass wir keine Experten Wissen besitzen und es sich hier lediglich um eine simulierte Gegenüberstellung handelt.

Modellierung: Naive Bayes

Wir trainieren nun erstmal den *Naive Bayes* Klassifizierer, welcher in *bnlearn* in **naive.bayes** implementiert ist.




```
#>
#>   k-fold cross-validation for Bayesian networks
#>
#>   target network structure:
#>    [Naive Bayes Classifier]
#>   number of folds:                        10
#>   loss function:                          Classification Error (Posterior, disc.)
#>   training node:                          Target
#>   number of runs:                         10
#>   average loss over the runs:             0.1566942
#>   standard deviation of the loss:         0.005661811
```

Wir sehen, dass der Trainingsfehler im Mittel bei 0.1566942 ($\pm$ 0.005661811).
Das Modell hat sich somit gut an die Trainingsdaten angepasst.

```
#>
#>   k-fold cross-validation for Bayesian networks
#>
#>   target network structure:
#>    [Naive Bayes Classifier]
#>   number of folds:                        10
#>   loss function:                          Classification Error (Posterior, disc.)
#>   training node:                          Target
#>   number of runs:                         10
#>   average loss over the runs:             0.1671815
#>   standard deviation of the loss:         0.006637733
```

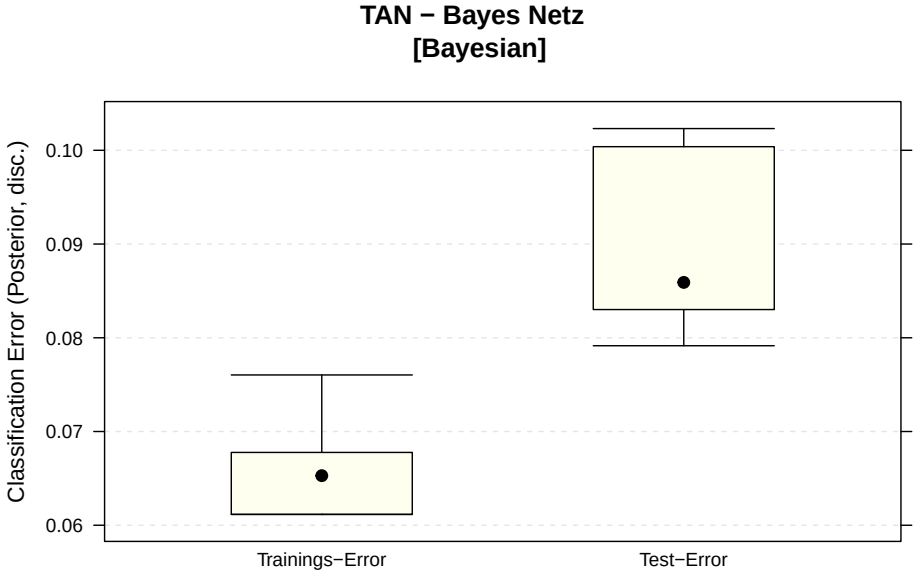
Bei Betrachtung des Test-Fehlers sehen wir einen etwas höheres Mittel von 0.1671815 ($\pm$ 0.006637733).

```
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction acc good unacc vgood
#>      acc    88   13    12    12
#>      good    5    8     0     0
#>      unacc  22    0   351     0
#>      vgood   0    0     0     7
#>
#> Overall Statistics
#>
#>                Accuracy : 0.8764
#>                  95% CI : (0.845, 0.9035)
#>     No Information Rate : 0.7008
#>     P-Value [Acc > NIR] : < 2.2e-16
#>
#>                   Kappa : 0.7194
#>
#>  Mcnemar's Test P-Value : NA
#>
#> Statistics by Class:
#>
#>                      Class: acc Class: good Class: unacc Class: vgood
#> Sensitivity              0.7652     0.38095       0.9669      0.36842
#> Specificity              0.9082     0.98994       0.8581      1.00000
#> Pos Pred Value           0.7040     0.61538       0.9410      1.00000
#> Neg Pred Value           0.9313     0.97426       0.9172      0.97652
#> Prevalence               0.2220     0.04054       0.7008      0.03668
#> Detection Rate           0.1699     0.01544       0.6776      0.01351
#> Detection Prevalence     0.2413     0.02510       0.7201      0.01351
#> Balanced Accuracy        0.8367     0.68545       0.9125      0.68421
```

Die finale Klassifizierung unser trainierten NB Modell zeigt eine besonders hohe sensivität gegenüber unacc auf. Was aufgrund der einhergehenden Betrachtung der Verteilung zu erwarten war. Die Klassen *good* und *vgood* werden weniger gut erkannt. Eine denkbare Überlegung wäre es die Daten zusammenzufassen. Unter Berücksichtigung eines wirtschaftlichen Kontext, könnte man lediglich daran interessiert sein, ob ein potenzieller Kunde ein Auto akzeptiert oder nicht akzeptiert. Man könnte somit die Klassen *good* und *vgood* mit der Klasse *acc* vereinigen. Obwohl die Accuracy bei 0.8764 liegt, könnte diese Vereinigung das Modell robuster machen.

Bayes Netz: TAN Verfahren

Nun trainieren wir den Tree-Augmented Naive Bayes Klassifizierer.



Die Berücksichtigung mancher Abhängigekitsstrukturen zahlt sich aus. Wir können im obigen Plot bereits eine deutliche Verbesserung erkennen.

```
#>
#>   k-fold cross-validation for Bayesian networks
#>
#>   target network structure:
#>    [Target][buying|Target][maint|Target:buying][safety|Target:buying]
#>    [persons|Target:safety][lug_boot|Target:safety][doors|Target:lug_boot]
#>   number of folds:                        10
#>   loss function:                          Classification Error (Posterior, disc.)
#>   training node:                          Target
#>   number of runs:                         10
#>   average loss over the runs:             0.06636364
#>   standard deviation of the loss:         0.005242128
```

Der mittlere Trainingsfehler-Fehler liegt bei 0.06636364 ($\pm$0.005242128).

```
#>
#>   k-fold cross-validation for Bayesian networks
#>
#>   target network structure:
#>    [Target][buying|Target][maint|Target:buying][safety|Target:buying]
#>    [persons|Target:safety][lug_boot|Target:safety][doors|Target:lug_boot]
#>   number of folds:                        10
#>   loss function:                          Classification Error (Posterior, disc.)
#>   training node:                          Target
#>   number of runs:                         10
#>   average loss over the runs:             0.08996139
#>   standard deviation of the loss:         0.009244934
```

Bei Betrachtung des Test-Fehlers sehen wir einen etwas niedrigeren mittleren Wert von 0.08996139 ($\pm$ 0.009244934). Wir erreichen somit eine mittlere Genauigkeit von 0.9100386 auf den Testdaten.

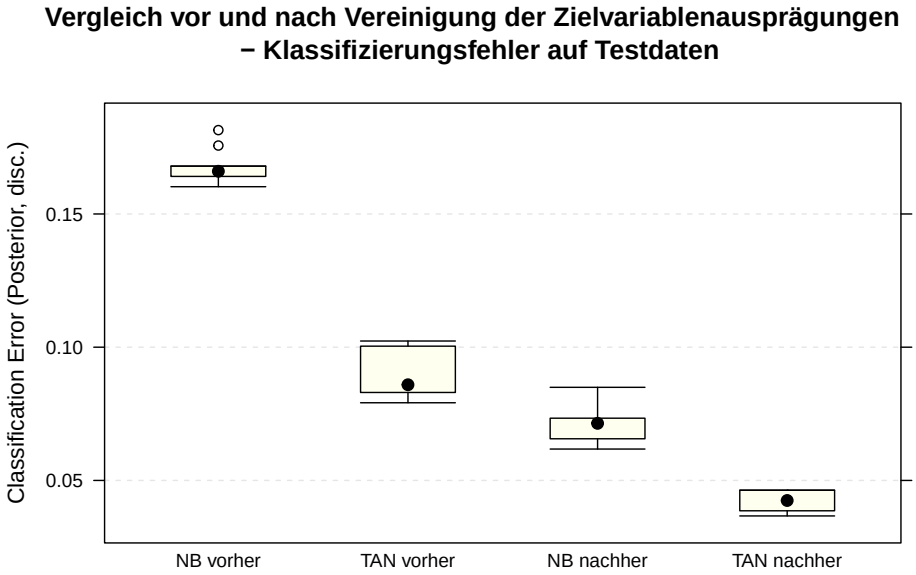
```
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction acc good unacc vgood
#>      acc   105    0    12     1
#>      good    1   20     0     2
#>      unacc   9    1   351     0
#>      vgood   0    0     0    16
#>
#> Overall Statistics
#>
#>                Accuracy : 0.9498
#>                  95% CI : (0.9273, 0.967)
#>     No Information Rate : 0.7008
#>     P-Value [Acc > NIR] : < 2.2e-16
#>
#>                   Kappa : 0.8904
#>
#>  Mcnemar's Test P-Value : NA
#>
#> Statistics by Class:
#>
#>                      Class: acc Class: good Class: unacc Class: vgood
#> Sensitivity              0.9130     0.95238       0.9669      0.84211
#> Specificity              0.9677     0.99396       0.9355      1.00000
#> Pos Pred Value           0.8898     0.86957       0.9723      1.00000
#> Neg Pred Value           0.9750     0.99798       0.9236      0.99402
#> Prevalence               0.2220     0.04054       0.7008      0.03668
#> Detection Rate           0.2027     0.03861       0.6776      0.03089
#> Detection Prevalence     0.2278     0.04440       0.6969      0.03089
#> Balanced Accuracy        0.9404     0.97317       0.9512      0.92105
```

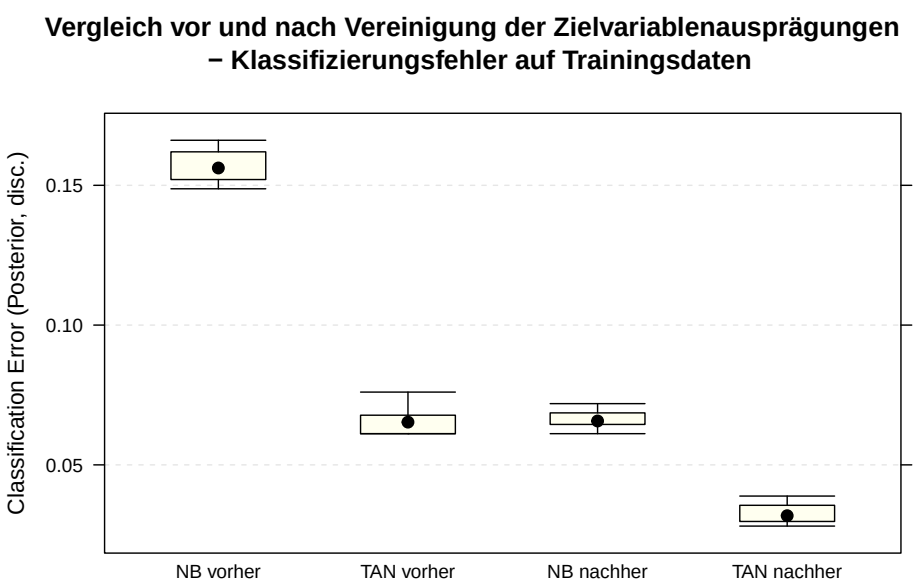
Die Modell Genauigkeit liegt bei 0.9498. Wir hatten zu Beginn des Experiments den *method* Parameter nicht entdeckt. Wir erlangten zuvor eine finale Genauigkeit den Testdaten von 0.3243. Es fiel besonders auf, dass *unacc* eine Sensivität von 0.2948 aufwies. Das war ziemlich niedrig. Unser vorherigen Überlegungen zu einem binären Klassifizierungsmodell überzugehen, wurden hier nochmal verstärkt gestützt. Betrachtete man die vorherige Konfusionsmatrix, so sah man sehr gut, dass *acc* eine hohe Varibilität aufzeigte. Das Modell konnte hier nicht gut *acc*,*good* und *vgood* differenzieren.

Anmerkung: Wir werden nachfolgend die Zielklasse zusammengefassen. Wir nehmen an, dass die mehrwertschöpfende Fragestellung sich an den Absatz richtet. Weiterhin nehmen wir an, dass man ein Auto akzeptiert oder nicht akzeptiert. Hierdurch fassen wir nun folgende Ausprägungen zusammen $\{acc, good, vgood\} \Rrightarrow \{acc\}$**, wodurch sich nun der Hypothesenraum auf** $\{unacc, acc\}$ **reduziert hat. Folglich**

liegt nun ein binäres Klassifikationsproblem vor.

Wir haben zwar die Zielvariable angepasst, dennoch dominiert noch *unacc*.





In der Grafik sind die neuen Modelle mit den alten Modellen gegenübergestellt. Wir können eine deutiche Verbesserung sehen, alle Boxplots auf der rechten Seite sind unterhalb ihrer Vorgänger.

```
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction acc unacc
#>      acc   148     9
#>      unacc   7   354
#>
#>                Accuracy : 0.9691
#>                  95% CI : (0.9503, 0.9822)
#>     No Information Rate : 0.7008
#>     P-Value [Acc > NIR] : <2e-16
#>
#>                   Kappa : 0.9266
#>
#>  Mcnemar's Test P-Value : 0.8026
#>
#>             Sensitivity : 0.9548
#>             Specificity : 0.9752
#>          Pos Pred Value : 0.9427
#>          Neg Pred Value : 0.9806
#>              Prevalence : 0.2992
#>          Detection Rate : 0.2857
#>    Detection Prevalence : 0.3031
```

```
#>       Balanced Accuracy : 0.9650
#>
#>        'Positive' Class : acc
#>
```

Wir sehen, dass der NB sich leicht gegenüber dem vorherigen Modell verbessert hat. Der Vorgänger hatte insbesondere Probleme *vgood*, *good* und *acc* auseinander zuhalten (hinsichtlich der Sensitivität). Die Modell Vereinfachung zeigt nun deutlich, dass die Sensitivität gestiegen ist.

```
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction acc unacc
#>      acc   154    11
#>      unacc   1   352
#>
#>                Accuracy : 0.9768
#>                  95% CI : (0.9599, 0.988)
#>     No Information Rate : 0.7008
#>     P-Value [Acc > NIR] : < 2.2e-16
#>
#>                   Kappa : 0.9458
#>
#>  Mcnemar's Test P-Value : 0.009375
#>
#>             Sensitivity : 0.9935
#>             Specificity : 0.9697
#>          Pos Pred Value : 0.9333
#>          Neg Pred Value : 0.9972
#>              Prevalence : 0.2992
#>          Detection Rate : 0.2973
#>    Detection Prevalence : 0.3185
#>       Balanced Accuracy : 0.9816
#>
#>        'Positive' Class : acc
#>
```

Das angesprochene Problem von NB, existierte beim TAN nicht. Wir sehen durch die Modellvereinfachung dennoch eine leichte Verbesserung in der Modellgüte. Nachfolgend eine Auflistung der Accuracy des multiplen und binären Klassifikationsproblem. Da der Parameter *method* von *bn.fit* zu spät gesichtet wurde, haben wir neben des bayesischen Ansatzes (Bayes) auch die zuvor frequentistischen Modellierungsvarianten (ML) mit protokolliert.

In Tab.2 sehen wir, dass die Zusammenlegung der Variablen das Modell vereinfacht. Wodurch wir für NB und TAN bessere Werte erzielen konnten. Interessant ist zu sehen, dass der einfachere Klassifizierer von beiden, daher NB, durch die

Modell	4-Klassen	2-Klassen	Verbesserung
Naive Bayes (ML)	0.8764	0.9691	0.0927
TAN (ML)	0.3243	0.7027	0.3784
Naive Bayes (Bayes)	0.8764	0.9691	0.0927
TAN (Bayes)	0.9498	0.9768	0.027

Table 2: Überblick Accuracy: Multiples Klassifikationsproblem vs. Binäre Klassifikationsproblem. Gegenüberstellung Maximum Likelihood vs. Maximum A Posteriori

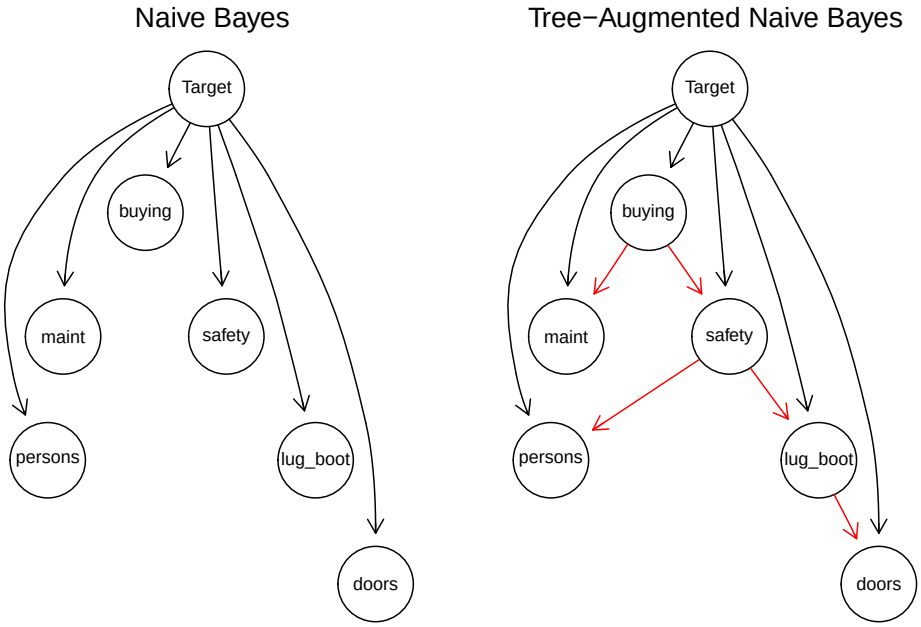
Datenvereinigung deutliche Verbesserungen erzielt, wohingegen das Bayes Netz beide Fälle vergleichbar gut ist.

Weiterhin sehen wir aber auch, dass die Bayes Methode zu einer wesentlich besseren Modellgüte geführt hat.

Teilaufgabe b: Modell Visualisierung als Graph

Visualisieren Sie die beiden Ansätze. (ebenfalls mit bnlearn)

Nun wollen wir uns die gelernten Strukturen einmal genauer betrachten. Hierfür beziehen wir uns wieder auf die zuvor erlernten Multi-Klassen Klassifizierer und vernachlässigen die Netze des binären Klassifizierer.



In der obigen Grafik sind beide erlernte Netze gegenübergestellt. Wir sehen im linken Plot den Naive Bayes basierenden Ansatz. Target hängt ab von

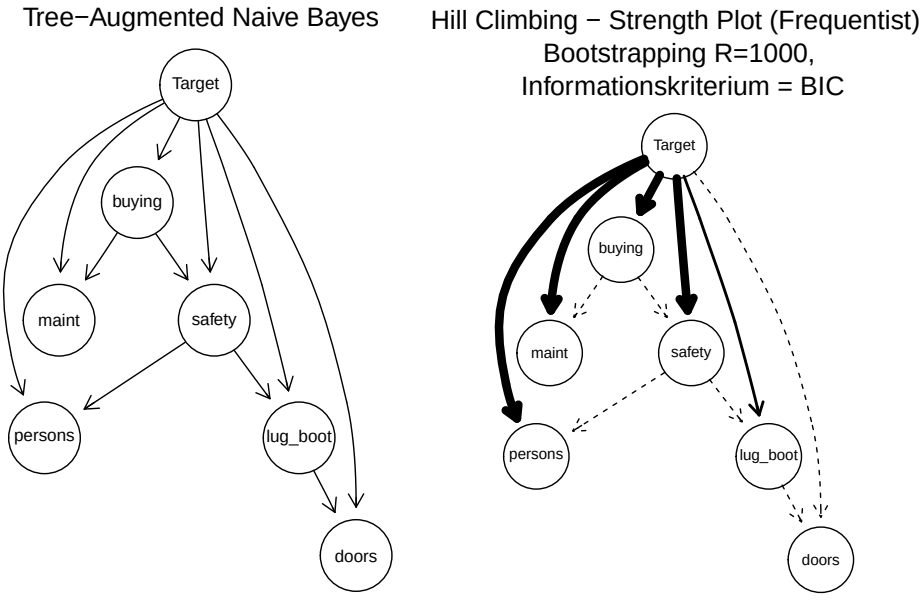
seinen Kinderknotenab. **Die Pfeilrichtung ist hier zu interpretieren als *ist abhängig von***. Wir sehen, wie zu erwarten für den NB, dass keine Abhängigkeiten zwischen den Kovariaten vorliegen. Rechts daneben sehen wir TAN. Die rot eingefärbten Pfeile zeigen die zusätzlichen Abhängigkeiten zwischen den Kovariaten auf, welche sich durch den Chow Liu Algorithmus ergeben.

Gegenüberstellung eines frequentistischen Modells

Wir wollen nun einmal ein BN heranziehen, welchen einem frequentischen Ansatzes verfolgt. Dabei führen wir eine klassisch frequentistische Herangehensweise an. Hierzu nutzen wir Bootstapping und schätzen mittels der Methode *Hill Climbing* ein approximatives Modell. Der Algorithmus *Hill Climbing* gehört zu der Familie der score-based Learning Algorithmen. Es handelt sich hierbei um einen *Greedy* Algorithmus, welcher keine Garantie für eine optimale Suche darstellt. Der Algorithmus ist sehr primitiv und betrachtet nur die rechts und linksseitigen Anstieg, so wandert er in ein Maximum und terminiert dort, falls links und rechts keine Verbesserung vorliegen sollte ("Bergspitze").

$$BIC = -2\ell\left(\tilde{\boldsymbol{\theta}}_{ML}\right) + p\ln(n) \tag{6}$$

Als Informationskriterium wird BIC herangezogen, welche minimal wird genau dann wenn, der Maxiumum Likelihood maximal wird (vgl. Gl. 6). Dabei repräsentiert $\ell(\tilde{\boldsymbol{\theta}}_{ML})$ die Log-Likelihood Funktion über den Maximum-Likelihood Schätzer. Dabei stellt p die Anzahl der Parameter dar, welche mit zunehmender Anzahl den Strafterm vergrößern.



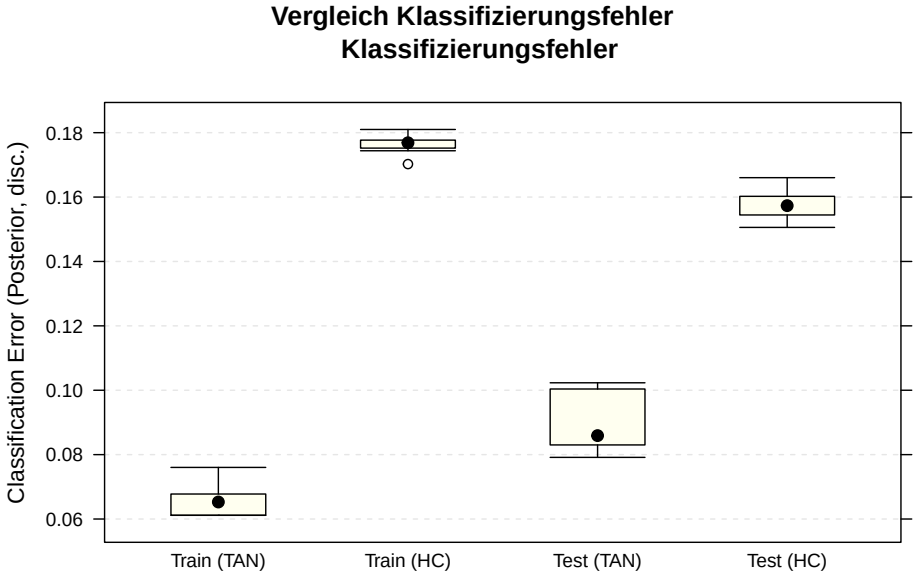
Wir sehen im Vergleich die TAN erzeugte Netzstruktur (links) und Hill-Climbing Modell (HC) des Frequentisten (rechts). Gestrichelte Linien zeigen auf, wo sich das HC vom TAN unterschieden hat. Die Stärke der Linien gibt Aufschluss über die Signifikanz der einzelnen Beziehungen. Dabei ist zu erkennen, dass die Akzeptanz stark von *persons*,*maint*,*buying* und *safety* beeinflusst wird. Die Beziehung zu *lug_boot* wurde hingegen seltener beobachtet und nimmt somit weniger Einfluss[2]. Wir bilden nun einen Klassifizierer, welcher die ermittelten Kanten des HC Verfahren nun nutzt.

```
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction acc good unacc vgood
#>      acc   115   21    71    19
#>      good    0    0     0     0
#>      unacc   0    0   292     0
#>      vgood   0    0     0     0
#>
#> Overall Statistics
#>
#>                Accuracy : 0.7857
#>                  95% CI : (0.7478, 0.8203)
#>     No Information Rate : 0.7008
#>     P-Value [Acc > NIR] : 8.511e-06
#>
#>                   Kappa : 0.5783
#>
#>  Mcnemar's Test P-Value : NA
#>
#> Statistics by Class:
#>
#>                      Class: acc Class: good Class: unacc Class: vgood
#> Sensitivity              1.0000     0.00000       0.8044      0.00000
#> Specificity              0.7246     1.00000       1.0000      1.00000
#> Pos Pred Value           0.5088         NaN       1.0000          NaN
#> Neg Pred Value           1.0000     0.95946       0.6858      0.96332
#> Prevalence               0.2220     0.04054       0.7008      0.03668
#> Detection Rate           0.2220     0.00000       0.5637      0.00000
#> Detection Prevalence     0.4363     0.00000       0.5637      0.00000
#> Balanced Accuracy        0.8623     0.50000       0.9022      0.50000
```

Wir haben mit dem HC deutlich schlechtere Werte hinsichtlich der Modellgüte erzielt. Die Klassen *acc* und *unacc* haben bezüglich ihrer Sensitivität hohe Werte erreicht, jedoch konnte *good* und *vgood* nicht erkannt werden. Die Spezifität hingegen liegt bei beiden sehr hoch, was jedoch auf Grund ihrer relativ

[2]Der Schwellwert für die Berücksichtigung einer Kante lag bei 0.962‘

betrachteten Häufigkeiten im Datensatz nicht verwunderlich ist. Nachträglich betrachten wir noch einmal den Fehler der Kreuz-Validierung.



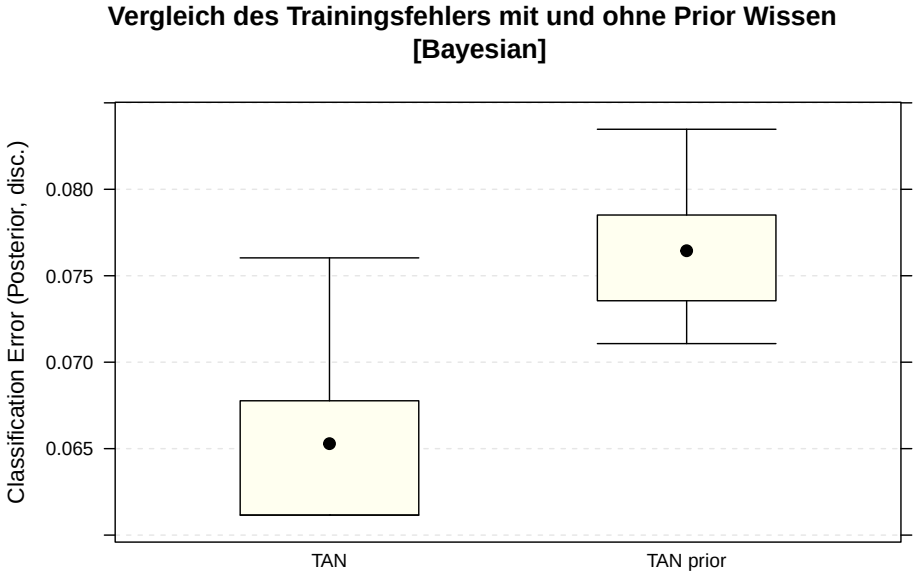
Wir sehen, dass unser TAN Modell, das frequentitische ermittelte Modell auch hinsichtlich seines mittleren Fehlers schlagen konnte.

TAN - Bayes Netz mit Prior - Information

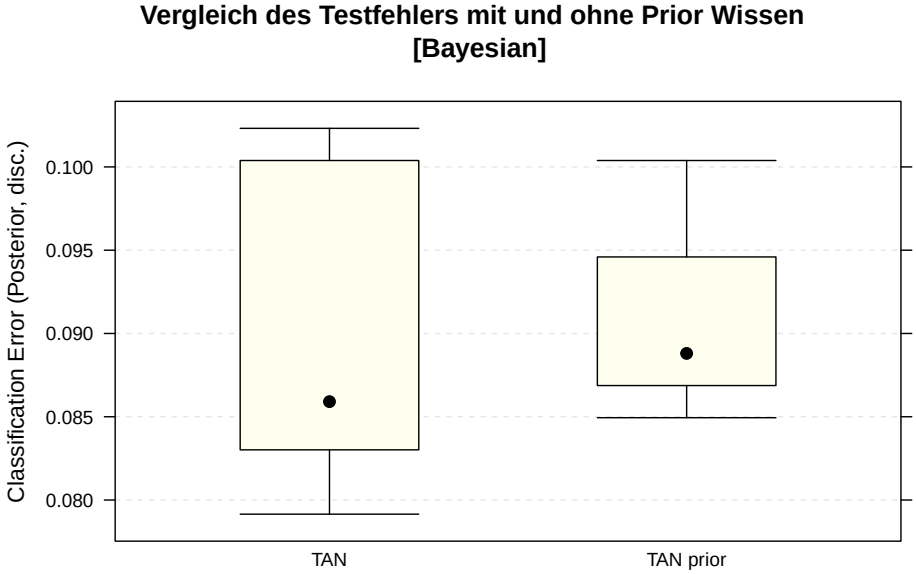
Wir nehmen nun an, dass wir von einem Experten folgende Aussagen erhalten haben:

- Sicherheit wird beeinflusst durch doors, maint und lug_boot (Annahme: erhöht die Knautschzone)
- Die Wartungskosten sind meist teurer, wenn das Auto einen hohen Anschaffungswert besitzt (Annahme: Qualität spiegelt sich im Preis wieder)
- Anzahl Türen beeinflusst die Anzahl der Personen, jedoch nicht anders herum (z.B. 4 Sitzer mit 2 Türen).
- Der Kaufpreis ist unabhängig von der Anzahl Personen (z.B. Sportwagen).

Wir werden nun nachfolgend das Domänen Wissen im Modell umsetzen.



Wir können sehen, dass der Trainingsfehler sich erhöht hat. Unsere Modell passt sich somit noch schlechter an die Daten an.



Wir sehen, dass das die Verteilung des Fehlers für das TAN prior Modell weniger Varibilität aufzeigt, als unser Modell ohne Vorwissen. Erstaunlich, auch der tatsächliche Fehler konnten wir mit der Hinzunahme des Experten Wissen verringern. Nachfolgend betrachten wir noch die finale Klassifikation auf den Testdaten.

```
#> Confusion Matrix and Statistics
```

```
#>
#>           Reference
#> Prediction acc good unacc vgood
#>      acc    96    0     7     1
#>      good    0   21     0     3
#>      unacc  17    0   356     0
#>      vgood   2    0     0    15
#>
#> Overall Statistics
#>
#>                Accuracy : 0.9421
#>                  95% CI : (0.9184, 0.9606)
#>     No Information Rate : 0.7008
#>     P-Value [Acc > NIR] : < 2.2e-16
#>
#>                   Kappa : 0.8706
#>
#>  Mcnemar's Test P-Value : NA
#>
#> Statistics by Class:
#>
#>                      Class: acc Class: good Class: unacc Class: vgood
#> Sensitivity              0.8348     1.00000       0.9807      0.78947
#> Specificity              0.9801     0.99396       0.8903      0.99599
#> Pos Pred Value           0.9231     0.87500       0.9544      0.88235
#> Neg Pred Value           0.9541     1.00000       0.9517      0.99202
#> Prevalence               0.2220     0.04054       0.7008      0.03668
#> Detection Rate           0.1853     0.04054       0.6873      0.02896
#> Detection Prevalence     0.2008     0.04633       0.7201      0.03282
#> Balanced Accuracy        0.9075     0.99698       0.9355      0.89273
```

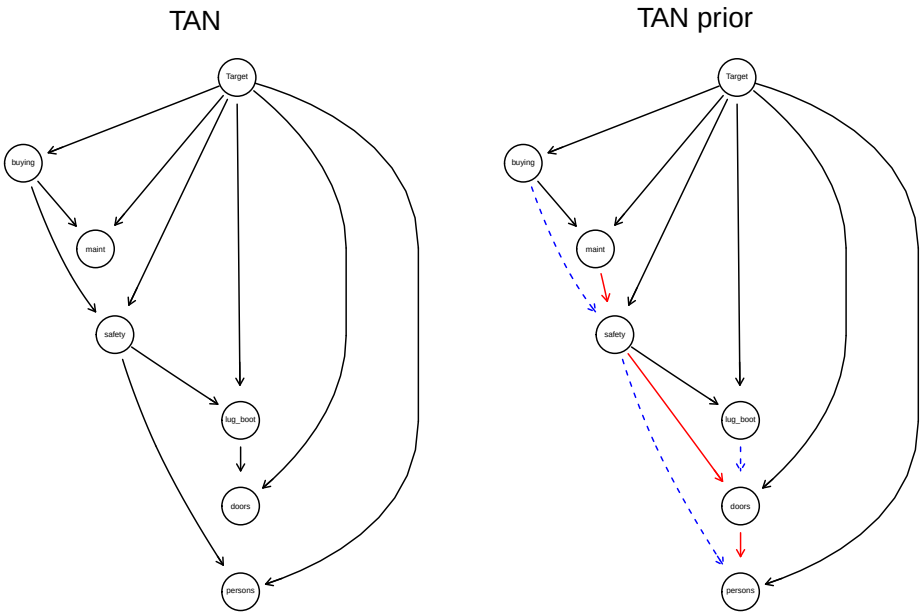
Die obige Konfusion Matrix zeigt die finale Evaluierung auf den Daten.

Metrik	**TAN**	**TAN prior**
Accuracy	0.9498	0.9421
Train-Error	0.0663 ($\pm$ 0.0052)	0.0765 ($\pm$ 0.0040)
Test-Error	0.0899 ($\pm$ 0.0092)	0.0903 ($\pm$ 0.0050)

Table 3: Gegenüberstellung: TAN vs. TAN prior

In Tab.3 haben wir das vorherige Modell mit dem a priori angereichertem Modell verglichen. Die Genauigkeit hat sich im TAN prior Modell leicht verschlechtert, jedoch sehen wir, dass der Fehler der Kreuz-Validierung auf den Test-Daten wie auch auf den Trainingsdaten weniger streut. Man erkennt, dass wir durch das hinzufügen unserer Wissen einen größeren Bias erhalten. Man vergleiche hierzu den Trainingsfehler auf, welcher minimal größer ist als zum Vorgänger Modell

ohne Vorwissen, jedoch die einhergehende Streuung auf den Testdaten hierdurch reduziert werden konnte.



Stellt man beide Modelle graphisch gegenüber, so sieht man sehr schön unsere, durch Experten-Wissen angereicherten, Modelle. Die roten Pfeile heben hierbei Beziehungen hervor, welche das Modell auf Grund seiner Zielfunktion vernachlässigt hätte und durch unser Vorwissen hinzugefügt wurde. Die blauen Pfeile spiegeln hingegen stellen das ursprüngliche Modell ohne Vorwissen dar.

Teilaufgabe c: Markov Blanket im Bayes Netz

Bestimmen Sie den Markov Blanket der einzelnen Knoten im Bayes Netz.

Zunächst erläutern wir was man unter einem Markov Blanket versteht. Hierfür betrachten wir die zuvor eingeführten **local semantics**, welche das Netz als eine Sammlung bedingt unabhängiger Ausdrücke betrachtet. Als Markov Blanket bezeichnet man zu einem gegeben Knoten X_i, seine Elternknoten U_i, Kinderknoten Y_i und die Elternknoten der Kinderknoten Zi. Es gelten insbesondere zwei Aussagen.

1. Jeder Knoten ist bedingt unabhängig von seinen Nicht-Nachfolgern, falls der Zustand der Elternknoten bekannt ist.
2. Jeder Knoten ist bedingt unabhängig für alle anderen Knoten, wenn sein Markov Blanket gegeben ist.

Daraus ergeben sich für die *local semantics* folgende bedingte Unabhängigkeiten:

$$\mathbf{P}\left(X_i|U_{1,\cdots},\ U_m, Z_{1j},\ldots,Z_{nj}\right) = \\ = \mathbf{P}\left(X_i|U_1,\ldots,U_m\right) \tag{7}$$

$$\mathbf{P}\left(X_i|U_{1,\cdots},U_m, Y_{1,\cdots},Y_n, Z_{1j},\ldots,Z_{nj}\right) = \\ = \mathbf{P}(X_i|\text{ alle Variablen }) \tag{8}$$

Diese Bedingungen müssen für jeden Knoten im Netz erfüllt sein, damit die global Semantics erhalten bleiben.

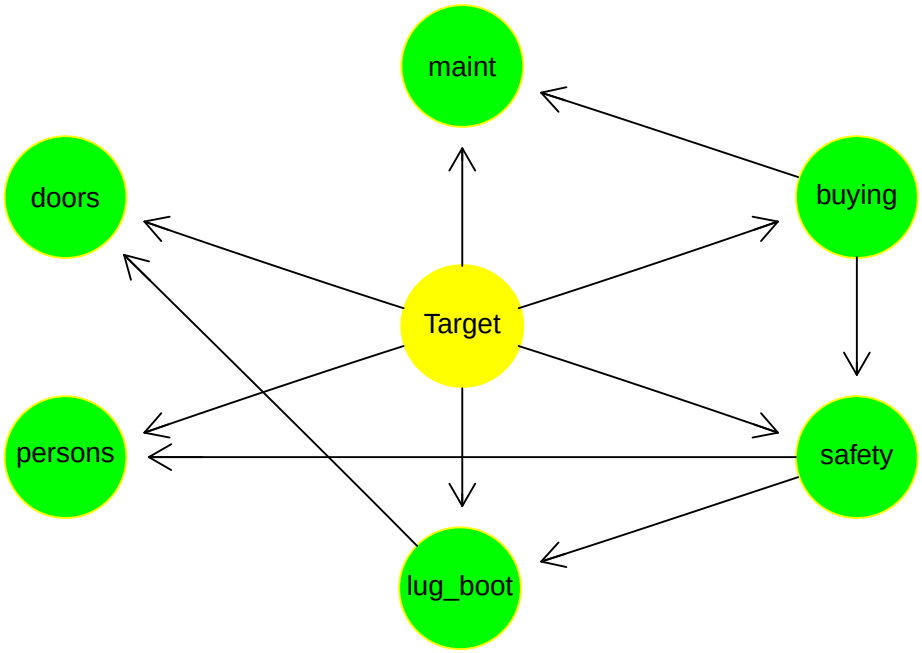
Wir werden diese Knoten nun visualisieren. Hierbei ist der Knoten von Interesse X_i gelb eingefärbt. Wir bezeichnen U_i als X_i Elternknoten und Y_i die Kinderknoten und als Z_i die Eltern der direkten Nachfolgern (Kinderknoten), welche grün eingefärbt sind.

Anmerkung: Die Pfeilrichtung von *bnlearn* repräsentiert die Relation X_i ist abhängig von X_j. Daher es existiert die Kante $X_i \to X_j$. In der Vorlesung haben wir jedoch die Relation X_j beeinflusst X_i kenenngelernt, daher $X_j \to X_i$.

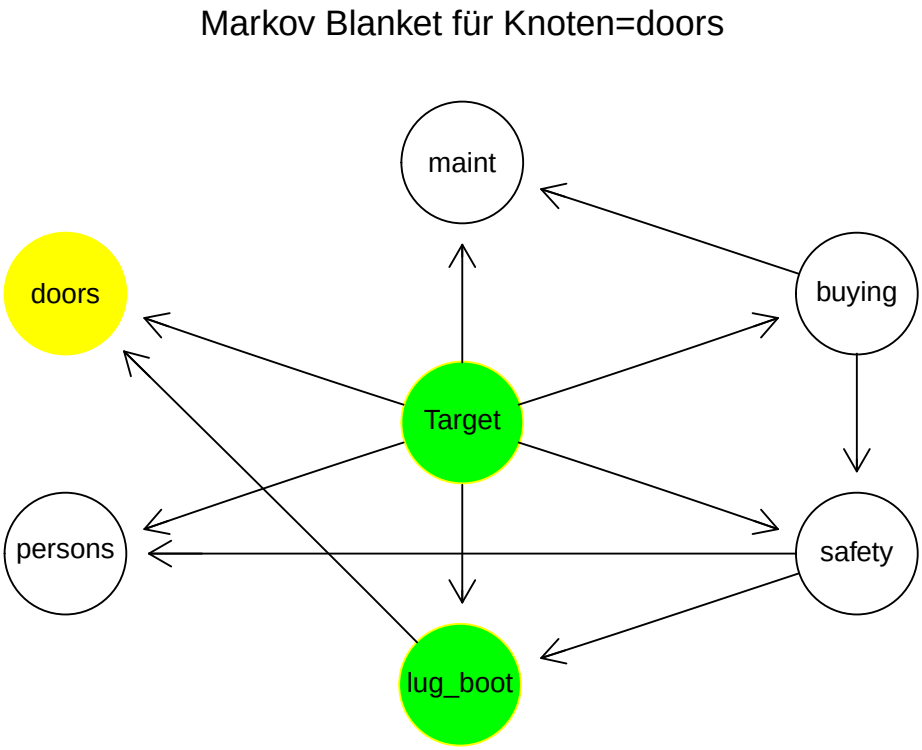
X_i	**U**	**Y**	**Z**	**Markov Blanket**
Target	buying, maint, safety, lug_boot,persons, dooors			buying, maint, safety, lug_boot,persons, doors
doors		persons, Target	persons	persons, Target
maint		buying, Target	buying	buying, Target
buying	maint, safety	Target	safety, maint	maint, safety,Target
safety	lug_boot, persons	buying,Target	persons, lug_boot	lug_boot, persons,buying,Target
lug_boot		safety, Target	safety	safety, Target
persons	doors	safety,Target	safety	doors,safety,Target

Table 4: Übersicht der identifizierten Markov Blankets des TAN

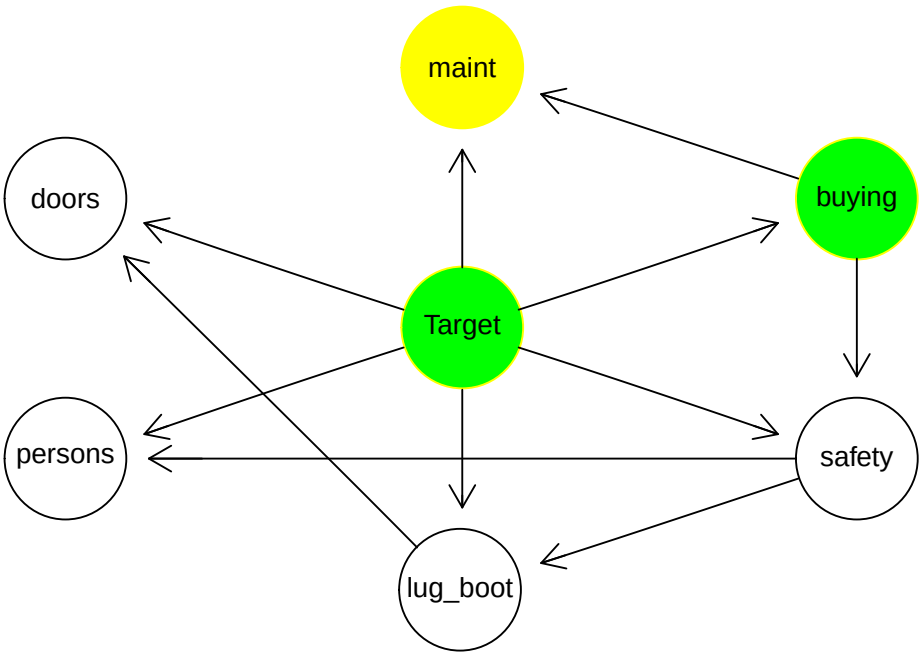
In Tab.4 ist eine Zusammenfassung der erfassten Markov Blankets. Nachfolgend wollen wir diese dennoch mal genauer betrachten.



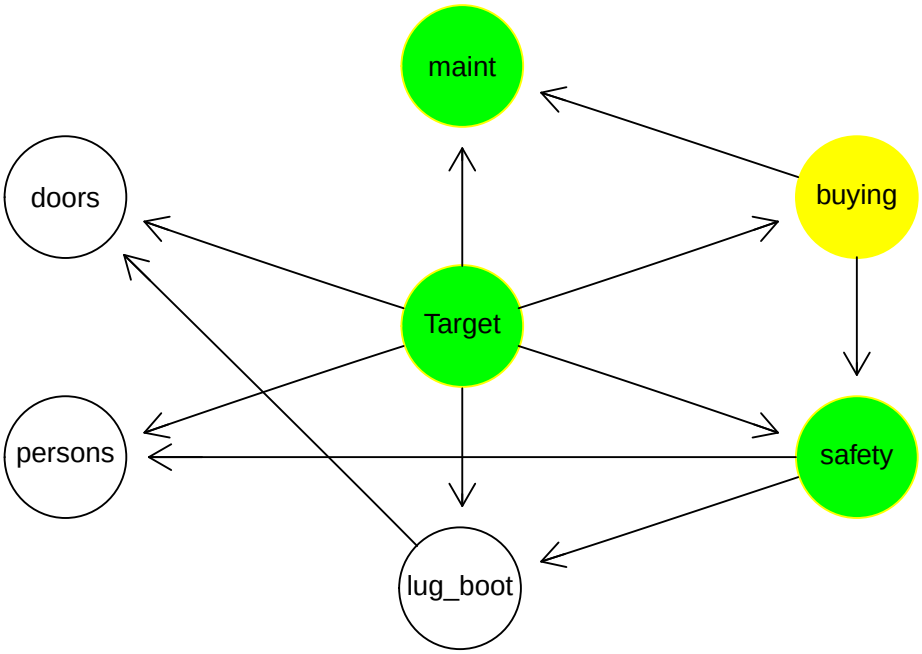
Durch das TAN Verfahren wurde der NB Klassifizier als Grundlage gelegt. Hierdurch ergibt sich, dass *Target* abhängig von all seinen Elternknoten ist $U = \{buying, maint, safety, lugboot, persons, doors\}$.



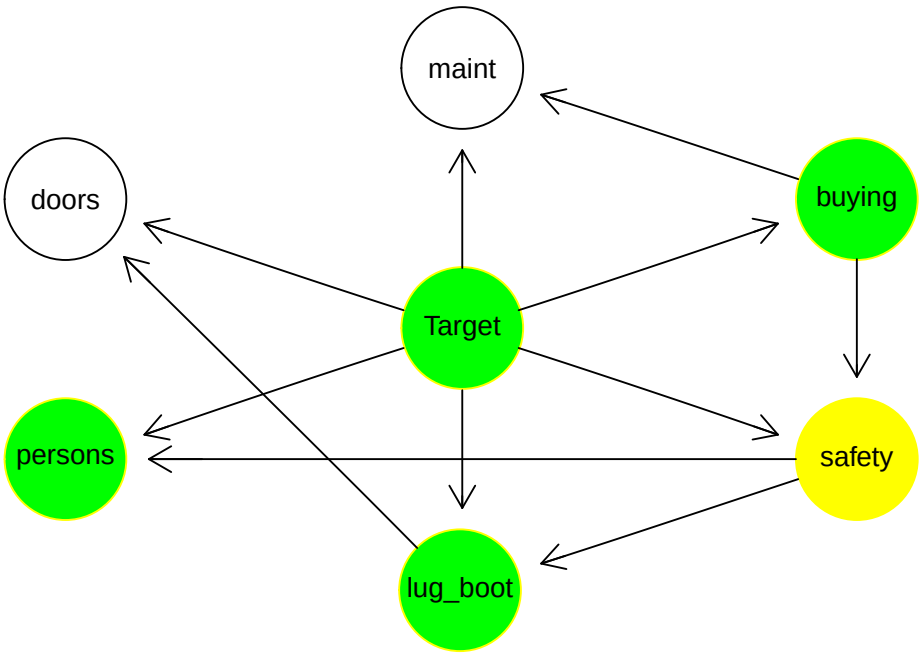
Das Markov Blanket für *doors* beinhaltet zwei Kinderknoten. Dabei stellt *doors* den Elternknoten zu *persons* und *Target* dar. Zugleich ist jedoch auch *persons* der Elternknoten des Kindes *Target*.



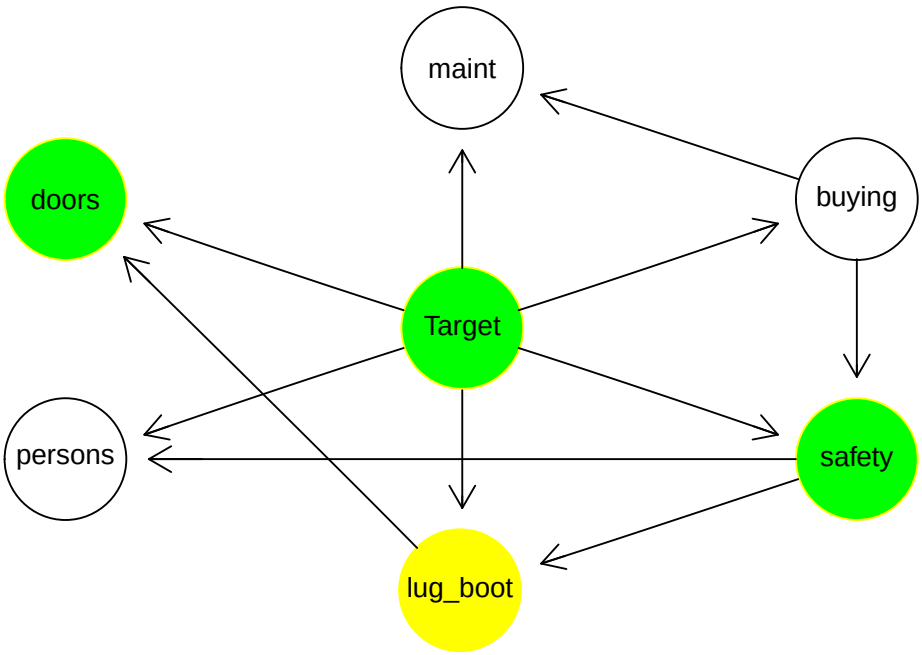
Das Markov Blanket für *maint* beinhaltet auch nur zwei Knoten. Dabei ist *maint* der Elternknoten mit *Target* und *buying* als seine Kinderknoten.



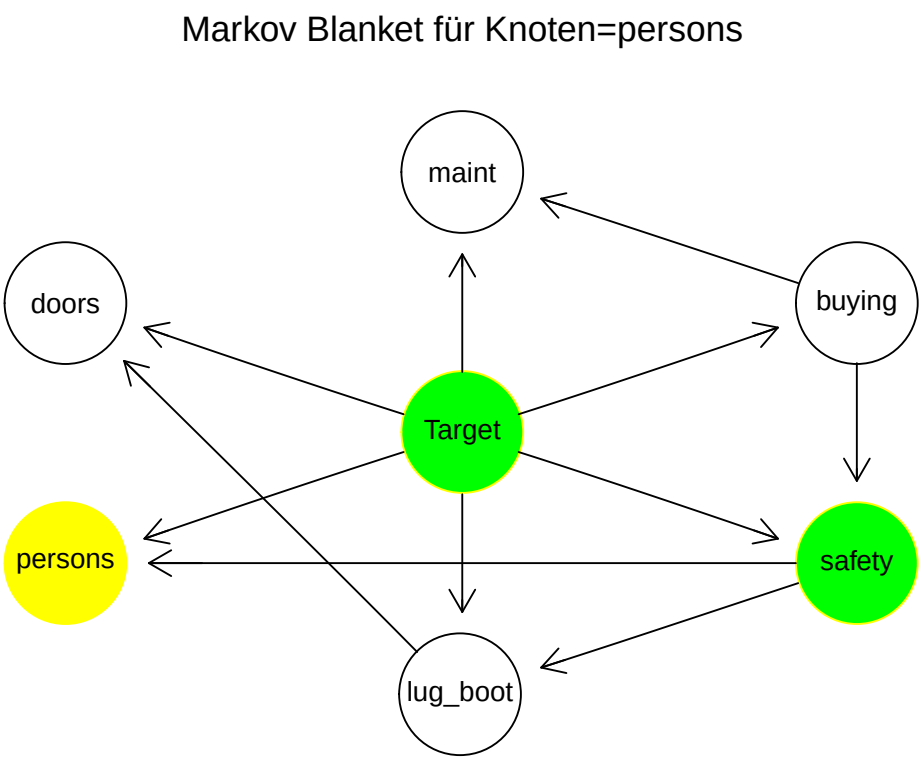
Das Markov Blanket für *buying* beinhaltet 3 Knoten. Dabei ist *buying* hier sowohl zweimal Kind - als auch einmal Elternknoten. Die Knoten *maint* und *safety* sind hierbei die Elternknoten von *buying*. Der Knoten *Target* ist das Kind von *buying*, welches zudem *maint* und *safety* als Elternknoten hat.



Betrachten wir nun *safety* hier haben wir nun 4 Knoten im Markov Blanket. Zunächst identifizieren wir die Elternknoten von *safety*. Diese sind *lugboot* und *persons*. Kinderknoten von *safety* sind *Target* und *buying*, wobei *buying*, *lugboot* und *persons* auch Elternknoten des Kindes *Target* widerspiegeln.



Das Markov Blanket für *lugboot* besteht aus zwei Knoten *safety* (Kind, Elternknoten von *Target*) und *Target* (Kind).



Letztlich bleibt noch das Markov Blanket zu *persons* zu betrachten. Der Elternknoten ist *doors*, welcher jedoch auch Elternknoten von *Target* ist. Des Weiteren ist *safety* der Elternknoten von Target, wobei *Target* und *safety* die Kinderknoten von *persons* sind.

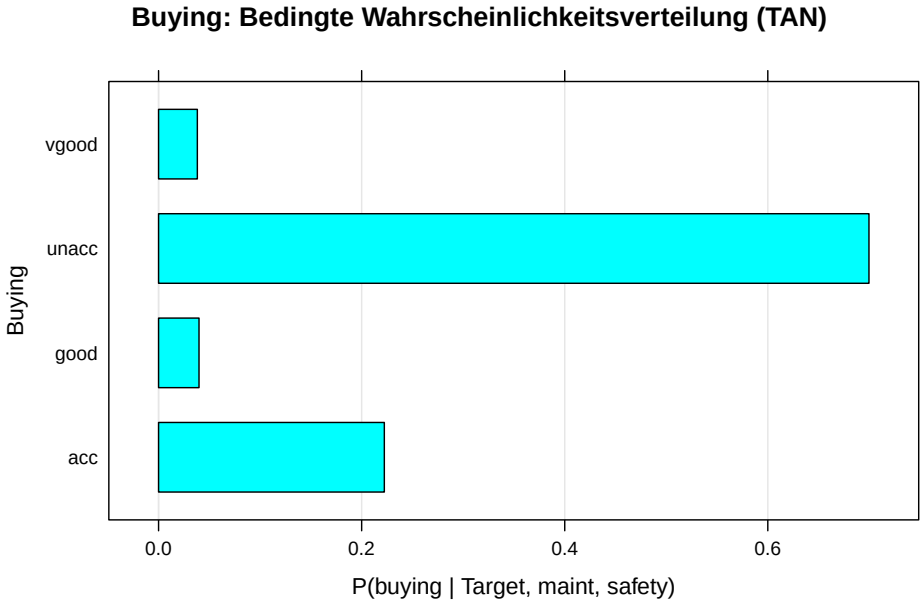
Teilaufgabe d: Interpretation der Ergebnisse

Interpretieren Sie ihre Ergebnisse.

Wir konnten feststellen, dass die erlernte Struktur des Bayes Netz, neben der Klassifikation, uns auch Einblicke in die erlernten Abhängigkeiten geben konnte, welche jedoch vom jeweiligen Verfahren abhängig waren. Dabei zeigen Bayes Netze eine sehr gute Möglichkeit Zustände innerhalb eines Systems zu modellieren. Wir haben die gemeinsame Verteilung des Ausgangsszenarios modellieren können und wollen uns nun im letzten Teil nochmal auf die resultierenden bedingten Wahrscheinlichkeitsverteilungen einzelner Variablen konzentrieren. Da eine Variable lediglich von ihren Elternknoten abhängig ist, werden wir zudem nochmal die Elternknoten in die Dokumentation aufnehmen.

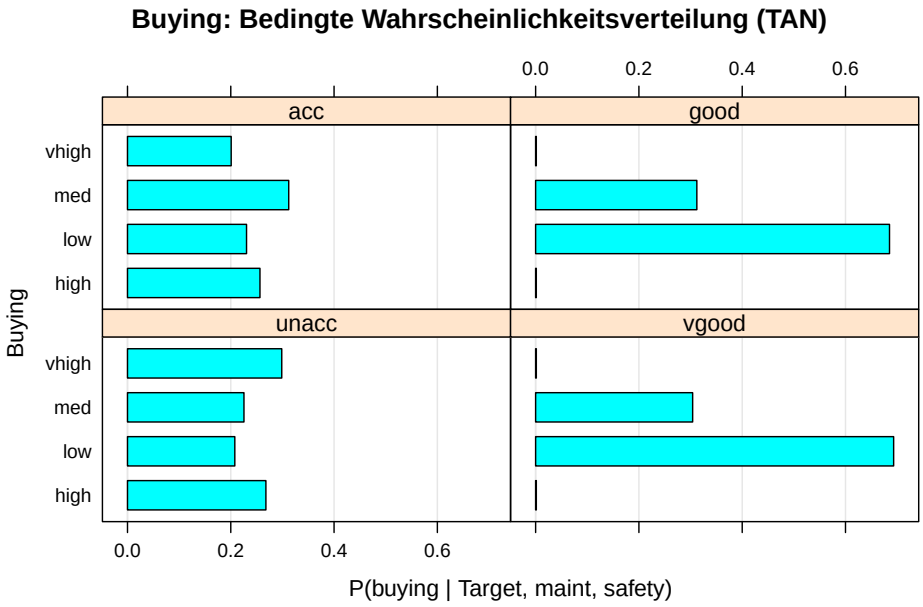
“Bayesian network is a directed graph in which each node is annotated with quantita- tive probability information. The full specification is as follows: 1. Each node corresponds to a random variable, which may be discrete or continuous.

2. A set of directed links or arrows connects pairs of nodes. If there is an arrow from node X to node Y , X is said to be a parent of Y. The graph has no directed cycles (and hence is a directed acyclic graph, or DAG. 3. Each node X i has a conditional probability distribution P(X i | P arents(X i)) that quan- tifies the effect of the parents on the node." - S.511 Russell Norvig - AI A mordern Approach




```
#> Error: 'grain.CPTspec' is not an exported object from 'namespace:gRain'
#> [1] "buying"   "maint"    "doors"    "persons"  "lug_boot" "safety"
```

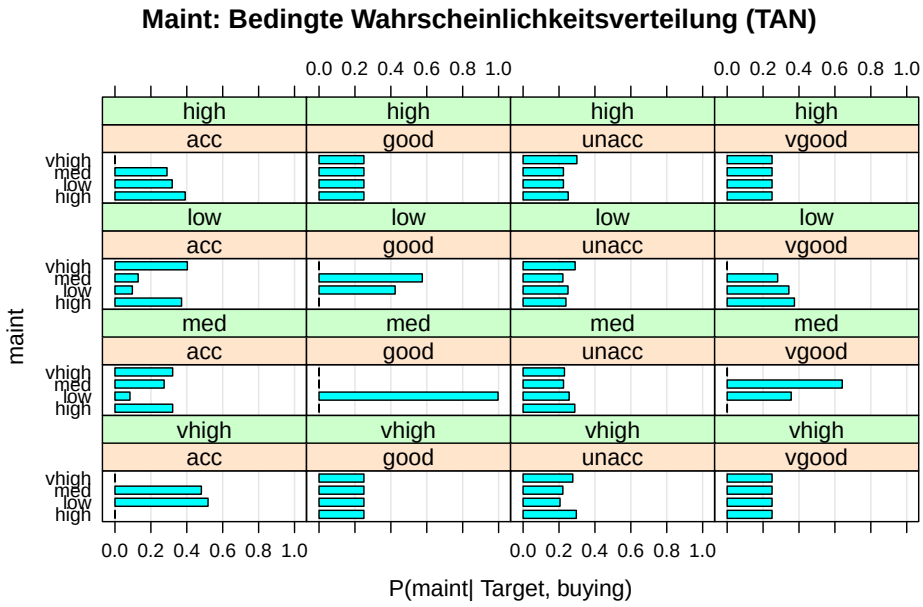
Interpretation Buying




```
#> [1] "maint"  "safety"
```

```
#> [1] "Target" "maint"  "safety"
```

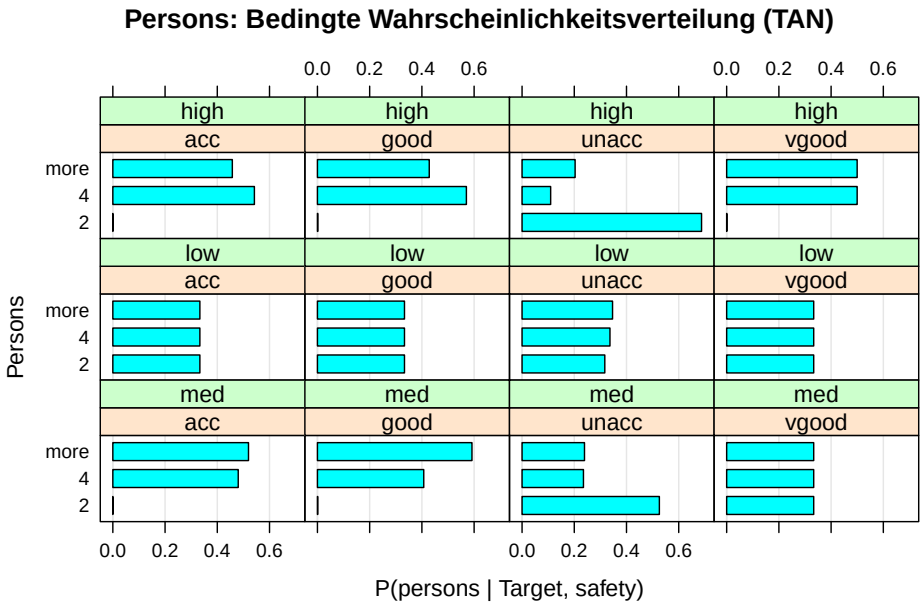
Interpretation maint




```
#> [1] "Target" "buying"
```

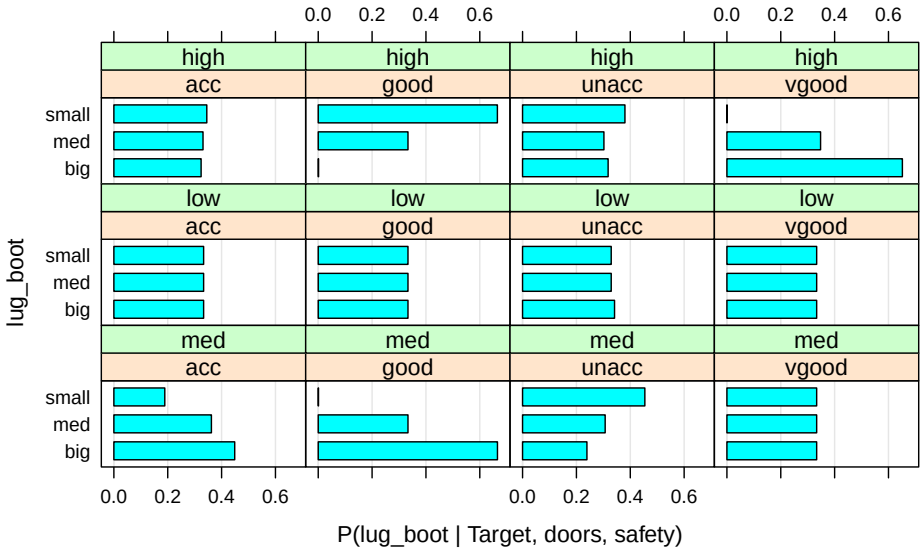
```
#> [1] "Target" "buying"
```

Interpretation Persons



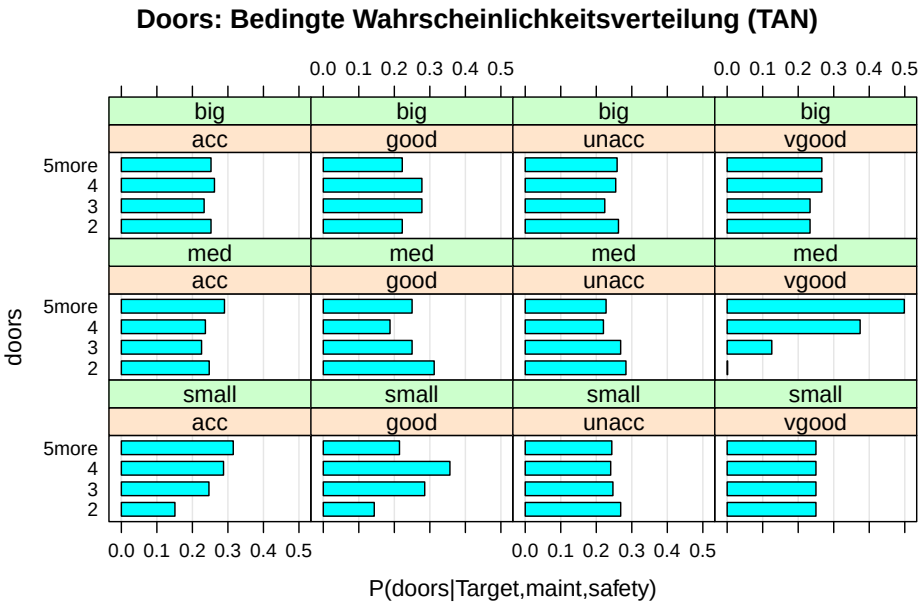

```
#> [1] "Target" "safety"
```

Interpretation Lug_boot



```
#> [1] "Target" "safety"
```

Interpretation doors




```
#> [1] "Target"   "lug_boot"
```

Hohe Sicherheit erhöht die Akzeptanz

TBD———————————————

Unter der Bedingung, dass es sich um ein viertüriges Auto handelt, einen großen Gepäckraum besitzt und mehr als 4 Leute Platz in diesem Auto haben, so liegt die bedingten Wahrscheinlichkeit, dass das Auto mit einer hohen Sicherheitseinstufung versehen wird bei 0.3494898. Das bedeutet im Umkehrschluss, dass es zu einer 0.6505102 nicht als Sicherheitseinstufung nicht als *low* eingestuft wird. Es fällt jedoch auf, dass die Wahrscheinlichkeiten relativ gleichverteilt sind. Wir wissen, dass die A posterori Warhscheinlichkeit sich immer mehr der a priori Wahrscheinlichkeit annährt je weniger Beobachtungen vorliegen.

- $P(safety = low | persons = 4, lugboot = big) = 0.3069728$
- $P(safety = med | persons = 4, lugboot = big) = 0.3435374$
- $P(safety = high | persons = 4, lugboot = big) = 0.3494898$

Offene Punkte:

- Erwartungshaltung der Interpretation? CP-Query? CPT? CPT Chart?
- graphviz.chart -> funktioniert nicht? Falls gewünscht, wie bringe ich es zum laufen?
- Pfeilrichtungen sind umgedreht (laut Anmerkung Kommilitone und Prof), wenn dem so sei, was sagen mir dann bn.fit.barchart(): 'plot the probabilities in the conditional probability table associated with each node.'. Das entspricht doch P(X| Parents(X)). Implementierung zeigt von Target auf leere Menge. Widerspruch zur Annahme (umgekehrte Richtung). Ggf. vgl. Oxford Slides Literatur.
- Verständnis Frage: TAN lediglich Algorithmus für MWST. Max vom IG. Das strukturelle Lernen ist somit weder frequentist noch bayesian, korrekt? Wir haben weder Konfidenzintervalle noch Kredibilität. Paramterschätzung gegeben des Graphes jedoch wieder Bayesian Style (wie kann ich mir das generative Modell hier vorstellen?)

Anmerkungen/Korrektur