AALBORG UNIVERSITET

Institut for *Datalogi*
Fredrik Bajers Vej 7, building E
DK-9220 Aalborg Ø

TITEL
Kreditvurdering ved brug af bayesiansk reject inference

EMNE
Datamining og reject inference

PROJEKTGRUPPEMEDLEMMER
Kim N. Jensen

VEJLEDER
Finn V. Jensen

SEMESTER
DAT 6

PROJEKTPERIODE:
1/9-2005 til 26/1-2006

ANTAL KOPIER
4

ANTAL SIDER
52 sider ekskl. bilag

Abstract

The focus of this thesis is to develop a credit scoring model that, with a higher degree of certainty than earlier, will obtain a greater share of those customers that meets the conditions defining solvency. The model is constructed through an iterative process composed of two procedures: First a selection-model is created based on information from solvent customers only. In this manner I make use of information on customers, who in the past had shown to be creditable, and argue that those have qualities that the bank in the feature should opt for. Secondly the model is modified to make sure it accepts a certain share of applicants. This must be done to avoid the reject inference problem (because the model is based on solvent customers only) and to avoid that the selection mechanism over time would accept still fewer applicants. To ensure I get the topmost share of applicants I choose a classification method that can tell how certain it is on its conclusion. This led to selecting a modified Naïve Bayes classifier, altered in such a way that it accepts a specific share of applicants. The method was tested using a virtual experiment and the result indicated that the method has some good promise. There are however issues that should be investigated more closely, e.g. the construction of the experiment and alternatives to the Naïve Bayes classifier.

Forord

Dette speciale er udarbejdet i perioden september 2005 til januar 2006 og er blevet til i forbindelse med en uddannelsesorlov fra mit arbejde hos IBM. Specialet er skrevet som afslutning på datalogi som hovedfag ved Institut for datalogi på Aalborg Universitet.

Jeg vil sige tak til IBM, som muliggjorde, at hele projektet kunne lade sig gøre. Jeg vil også benytte lejligheden til at give Finn V. Jensen en særlig stor tak for sin store støtte og vejledning under hele processen. Endelig vil jeg takke min kone for hendes moralske støtte og opbakning.

Kim N. Jensen

Indhold

Kapitel 1

Kreditvurdering

1.1 Introduktion

Banker og andre finansielle institutioner, som låner penge ud, vil sikre sig, at deres investering bliver rentabel. De har derfor brug for metoder for at vurdere, hvor stor en risiko lånansøgeren er. På baggrund af det kan det så besluttes om lånet/kreditkortet[1] skal gives.

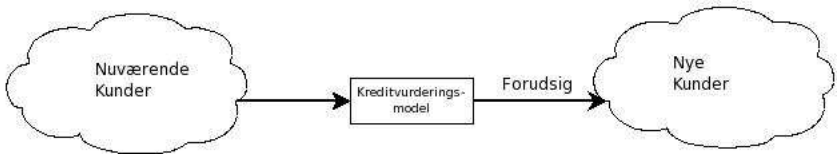
Kreditvurderingen kan foretages på flere måder. Set med historiske øjne har det tidligere udelukkende været en subjektiv vurdering, om et lån skulle bevilliges eller ej. Måske sad der en bankansat (med mange års erfaring) og vurderede, om ansøgeren var troværdig, og om han var villig til at betale lånet tilbage. I dag foregår det på en helt anden måde - selv om en subjektiv vurdering stadig spiller en rolle. Der anvendes metoder, modeller og metrikker, som er objektive, ensartede og sammenlignelige. Teknikkerne stammer fra klassisk statistik og datamining, fx lineær og logistisk regression, diskriminant analyse, beslutningstræer, neurale netværk etc. Det er modeller, der er indført for at ensarte beslutningstagningen og ikke mindst øge rentabiliteten.

Helt basalt drejer det sig om at få så god en model til kreditvurdering som muligt, og modellens primære mål er at kunne skelne mellem to grupper: De, der er kreditværdige, og de, som ikke er. Men en god model bør desuden give en større sandsynlighed (score) for de ansøgere, der performer[2] godt, og selvfølgelig lavere sandsynlighed for de, som ikke vil klare sig godt. Det betyder også, at jo højere sandsynlighed for, at ansøgeren er god, jo mindre risiko påtager långiveren sig, hvis lånet gives. Hvis et system kan dette, så giver det långiveren mulighed for at anvende en risiko-politik. Fx at acceptere

[1] Lån og kreditkort sidestilles i denne opgave, da sidstnævnte kan opfattes som et specielt lån.
[2] Performance definerer jeg her som værende betalingsevne og -vilje.

alle med en score over et vist niveau, afvise de med en score under et andet niveau og undersøge dem nærmere, som ligger derimellem.

Der foregår to processer, når man taler om kreditvurdering. Den ene er at tage stilling til, om en ansøger skal have lånet. Den anden er at følge performancen for de eksisterende låntagere. Ræsonnementet er, at risikoen for nye ansøgere kan vurderes på baggrund af den eksisterende viden, dvs. performancen for de, der har fået tildelt lån. En sådan empirisk kreditvurdering er basalt set et klassificeringsproblem. Ved at generalisere fra et kendt antal eksempler, bygger man en model, der på baggrund af opdagede mønstre i vores data kan fortælle, hvordan et ukendt emne skal klassificeres. Input til sådan en model er information om ansøgeren, fx hans indkomst, boligforhold og jobstatus. Alt sammen kendetegn, som kan antages at have en indvirkning på ansøgerens betalingsevne og -vilje.



Figur 1.1: Empirisk kreditvurdering

En vigtig pointe er, at man typisk ikke har information om de afviste ansøgeres performance. Man ved således ikke, hvordan de ville have klaret sig, hvis de fik tildelt lånet. Det medfører et potentielt dataudvælgelses-problem, da statistisk slutning (inference) kræver, at den stikprøve, som udtages, er tilfældig udvalgt og repræsentativ for hele populationen. Hvis man derfor bygger sin model udelukkende på baggrund af de ansøgere, som banken har accepteret at give et lån, risikerer man at få en model, der ikke er præcis nok - dette kaldes også for "reject inference" problemet.[3] Man er derfor nødt til at forholde sig til, hvordan de afviste ansøgere kan influere på modellen.

En måde at anskue problemet på er ved at se det som et manglende data problem. I så fald vil betydningen af den manglende viden om performancen og den effekt, hvormed det vil påvirke modellen (den statistiske konklusion), komme an på den mekanisme, som medfører de manglende data.[4] Er stikprøven fx komplet[5], og hvis den manglende datamekanisme er "missing competely at random" (MCAR) eller "missing at random" (MAR), kan man vise, at populationen af afviste ansøgere ikke vil påvirke modellen, og man kan derfor helt se bort fra dem. Men er der tale om ukomplette data, så kan man ikke se bort

[3]Feelders 2000.

[4]Sebastiani et al. 1997.

[5]Dvs. at alle kendetegn, som antages at kunne påvirke performancen, er observeret. Dvs. der er registreret en værdi for hvert kendetegn i hele datasættet.

fra de afviste ansøgere. Jeg vil beskrive disse begreber og koncepter nærmere i afsnit 2.9.

En anden pointe er, hvad man kunne kalde nåleøje-princippet. Her giver man kun kreditkort til de personer, som man er helt sikker på, vil være gode betalere. Fx ved konstant at strammer op på sin model, ved at fravælge de ansøgere med karakteristika svarende til de, der blev tildelt lån, men som performede dårligt.

1.2 Problemformulering

Hovedformålet med denne opgave er at udvikle en model med kriterier, der i højere grad øger sandsynligheden for, at man i fremtiden får øget andelen af de kunder, der opfylder kriterierne for god betalingsevne og -vilje. Det betyder også, at man i højere grad skal kunne afvise de ansøgere, der efter nuværende kriterier tildeles kreditkort, uden at burde få det.

I projektet vil jeg også undersøge i hvilken form, "reject inference" kan biddrage, når der i datamaterialet mangler information om både kendetegn og performance. Er der relevant information indeholdt i de afviste ansøgere, der legimiterer, at de skal inddrages i opbygningen af modellen, eller kan man helt vælge at se bort fra dem?

1.3 Metode

Datagrundlaget for dette speciale er kreditkortsansøgninger fra en dansk bank. Det er med udgangspunkt i dette empiriskemateriale, at jeg vil undersøge specialets to centrale antagelser: 1/ At man ved at udnytte viden om gode kunder kan bygge bedre kreditvurderingsmodeller og 2/ At man undgår "reject inference" problemet ved, at selektionsmekanismen fastholder et specifikt forhold mellem de, som accepteres, og de, som afvises.

Min antagelse er, at den viden, man har om de accepterede kunder, vil danne rammen for, hvilke fremtidige ansøgere der skal tildeles lån. Ideen er at forfine selektionsmekanismen således, at der i fremtiden vælges ansøgere, som mest ligner de gode kunder, som banken har. Jeg forudsætter med andre ord, at personer, der ligner hinanden, også vil handle ens, når det handler om betalingsevne og -vilje. Når jeg her taler om "at ligne hinanden", så mener jeg at have lignende kendetegn på forskellige målbare egenskaber, som køn, arbejde, indkomst- og boligforhold etc.

En konsekvens af, at den nye selektionsmodel udelukkende bygger på de accepterede ansøgere, er, at færre og færre ansøgere over tid bliver accepteret. For at undgå at dette nåleøje-princip (uddybes i afsnit 4.3) træder i kraft, og at for få bliver accepteret, skal jeg

have modellen til at acceptere flere ansøgere. Det er derfor vigtigt at se på præcist hvilke ansøgere, der udvælges. Her tænker jeg på, hvor god deres performance estimeres til at være. Med sådan et estimat kan modellen udvælge netop de ansøgere, der har størst sandsynlighed for at performe godt. Det er således ikke nok med en model, der kan klassificere en ansøger som værende enten god eller dårlig, modellen skal også angive en sandsynlighed for dette.

En optimal model vil kunne vælge alle de gode ansøgere og sortere alle de dårlige ansøgere fra. Men det er urealistisk, at man skulle kunne lave en model, der kan udpege samtlige gode ansøgere. Derfor sætter jeg som mål, at banken skal acceptere flere end et beregnet optimum. Derved undgår jeg nåleøje-princippet, og at "reject inference" problemet opstår. For at få modellen til at acceptere flere ansøgere, sætter jeg et lavere niveau for, hvornår en ansøger bliver accepteret. Det, som ofte sker, når et emne klassificeres, er, at det bestemmes ud fra, hvilken klasse det med størst sandsynlighed tilhører[6]. Det vil med to mulige kombinationer i så fald være den klasse, der får mere end 50% sandsynlighed. Løsningen på problemet er at sætte denne grænse så meget lavere, at modellen vil opnå det ønskede forhold mellem antallet af accepterede og afviste. Det kunne fx være, at modellen skal acceptere mellem 60-70% af alle ansøgere.
Men der skal findes et passende niveau for størrelsen på den andel, som skal accepteres. For sættes tallet for lavt, vil banken miste potentielle gode kunder, og sættes det for højt, vil banken få for mange dårlige kunder. Derfor vil jeg estimere andelen af gode kunder for hele populationen (alle lånansøgere), hvilket sker på baggrund af den registrerede performance for de accepterede kunder.

Der er også nogle praktiske problemer, der skal tages stilling til i projektet. Et problem er at få klargjort data således, at de kan anvendes til analyse. Her tænker jeg bl.a. på data-rensning og data-transformering. Et andet problem er udfordringen med mange manglende data i bankens datasæt. Noget, man kan gøre, er at estimere de manglende værdier, fx ved brug af "Expectation Maximization" (EM) algoritmen. Yderligere vil jeg skabe ansøgere, som jeg kender performancen for. Men for at kunne gøre det, må jeg have en model af, hvordan bankens ansøgere ser ud. Det er nødvendigt for at kunne sammenligne modellernes effektivitet. Jeg vil med andre ord kunne måle, om jeg opnår at skabe en model, der øger sandsynligheden for, at banken i fremtiden får en højere andel af de gode kunder.

I det følgende vil jeg beskrive strukturen for den resterende del af dette speciale:

Kapitel 2 indeholder en gennemgang af den teori, jeg anvender. Jeg redegør for maximum likelihood begrebet og gennemgår baggrunden for klassificering. Herefter følger en beskrivelse af udvalgte klassifikatorer og deres egenskaber. Jeg ser også på, hvordan

[6]De bayesianske metoder fungerer fx sådan.

boosting kan anvendes til at forbedre klassificeringen.

Et problem, man ofte står med, når empirisk materiale skal analyseres, er ukomplette data. En metode til at håndtere disse er EM-algoritmen. Denne vil jeg introducere primært med fokus på, hvordan de bayesianske metoder anvender denne.

Kapitel 3 er en kort beskrivelse af det empiriske materiale og den nødvendige datarensning og transformation, det gennemgår, før det kan anvendes til analyse. Problemet er bl.a., at data ikke er i det rette format og også indeholder støj, dvs. forkert information.

Kapitel 4 indeholder analysen. Jeg begynder kapitlet med at analysere bankens data vha. forskellige datamining principper. Resultatet af denne analyse udmunder i to modeller. En, som fremstiller bankens udvælgelsesstrategi, og en, som beskriver hhv. gode og dårlige kunder.

Jeg redegør også i detaljer for, hvordan resultaterne anvendes til at lave modelforbedringer, jf. min metode. Kapitlet slutter med to virtuelle tests af, om metoden og den tilgangsvinkel, jeg har valgt, også giver en øget andel af de ansøgere, der opfylder kriterierne for god betalingsevne og vilje.

Kapitel 5 indeholder konklusionen, hvor jeg sammenfatter specialets væsentligste resultater.

Kapitel 6 indeholder en diskussion af metodens brugbarhed, det empiriske materiale samt overvejelser om alternative løsninger.

Kapitel 2

Teori

2.1 Introduktion

I dette kapitel vil jeg gennemgå de vigtigste teorier og algoritmer, som jeg anvender i specialet. Det drejer sig i særdeleshed om metoder til håndtering af usikkerhed og manglende data samt metoder til klassificering.

Når man laver analyser og ønsker at konkludere noget generelt på baggrund af de data, man har til rådighed, så laver man en model af virkeligheden. Men sådan en model vil aldrig være præcis nok og aldrig kunne fange alle aspekter af den omfattende verden, som den forsøger at beskrive. For det første ville en universel model være alt for stor. Derfor laver man en mere simpel udgave, som har til formål at fange de vigtigste aspekter af den verden, man prøver at beskrive. For det andet vil man ikke have data nok til at kunne beskrive en universel model, man har kun en lille stikprøve. Modellen, man laver, er en generalisering af verden og vil derfor medføre tab i information og dermed tilføje en vis usikkerhed. Den usikkerhed skal man håndtere på bedste vis, og det gøres med sandsynligheder. Dvs. man giver et estimat for, hvor sikre man er på en konklusion. Sandsynlighed kan defineres sådan:

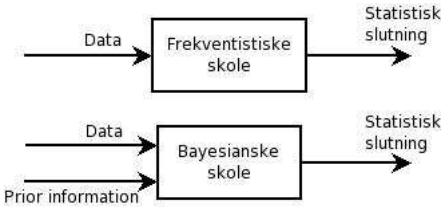
> Et kvalitativt mål for usikkerheden. Et tal som beskriver, hvor stærkt man tror, at en given hændelse vil ske.

Når man taler om forståelsen af begrebet sandsynlighed, er der to skoler (se også figur 2.1):

1. Objektiv sandsynlighed. Det er den klassiske forståelse, også kaldet frekventistiske opfattelse. Her er sandsynlighed en objektiv og målbar størrelse, og statistisk

slutning drages udelukkende på baggrund af observeret data - frekvensen hvormed en hændelse har fundet sted.

2. Subjektiv sandsynlighed. Denne opfattelse kaldes også bayesiansk, da den bygger på Bayes theorem. Her kan man anvende individuelle subjektive vurderinger af sandsynligheden for, at en given hændelse vil ske; noget frekventisten ikke tillader. Den subjektive vurdering kaldes også for prior-sandsynligheden. Med den bayesianske tilgang er det således muligt at fortælle, hvor plausibelt man mener, et udsagn er. Fx kan begge skoler tildele en sandsynlighed for, at næste stastminister bliver en kvinde. Men det er kun den bayesianske opfattelse, der tillader, at en subjektiv vurdering (personlig eller en eksperts) også anvendes i estimatet.

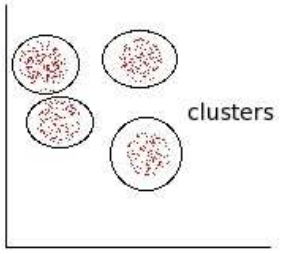


Figur 2.1: Fortolkninger af sandsynlighed

Jeg vil i specialet primært lægge mig op ad metoder og modeller baseret på den bayesianske opfattelse af sandsynlighed (nærmere begrundelse for dette følger i kapitel 4).

2.2 Klassificering

Når man taler om klassificering, handler det om, på baggrund af et elements karakteristika, at kunne tildele det en gruppe indeholdende ensartede elementer. For at lære klassifikationen, hvordan den skal virke, kan man enten anvende overvåget eller ikke overvåget læring (supervised learning eller unsupervised learning). For begge metoder gælder, at de lærer på baggrund af et træningssæt, dvs. en mængde kendte data, som man generaliserer ud fra. Fremtidige klassificeringer sker således på baggrund af den model, der opbygges vha. træningssættet.
Ved unsupervised læring kender man ikke klassen, elementerne tilhører. Her forsøger man at lære sammenhænge og strukturer i datasættet ud fra andre principper. Et eksempel på dette er "clustering", hvor datasættet partitioneres således, at hver gruppe indeholder *ensartede* elementer, men hvor elementerne ikke nødvendigvis har en fælles attributværdi. En ofte anvendt måde til bestemmelse er brugen af distance-mål. Figur 2.2 viser et eksempel på dette.



Figur 2.2: Fundne clusters

Når der er tale om overvåget læring, lærer klassifikationen sammenhængen mellem observerede værdier (features) og den klasse, de tilhører. Man bruger således et træningssæt, hvor elementerne på forhånd er tildelt en klasse. Dvs. man har en funktion, der som input tager et element, beskrevet ved et sæt af attributter, og som svar giver, hvilken klasse elementet med størst sandsynlighed tilhører. Matematisk kan det beskrives således:

$$f(x): (x_1, x_2, ..., x_n) \rightarrow C = p(C|\overrightarrow{x}) \tag{2.1}$$

Hvor $C \in (c_1, c_2, ..., c_m)$ er en klasse, fx "god" eller "dårlig", og hvor
$\overrightarrow{x} = (x_1, x_2, ..., x_n)$ er en feature vektor, som beskriver elementet (ansøgerens karakteristika).

En af de vigtigste opgaver inden for dette speciale er at kunne vurdere, om en ansøger er god eller dårlig. Derfor er overvåget læring et naturligt valg, når jeg skal lave min klassifikator.

2.3 Likelihood og Maximum Likelihood

Likelihood fortæller sandsynligheden for, at en sandsynlighedsmodel har produceret et givent sæt data. Og maximum likelihood fortæller hvilken model (af flere), som har størst sandsynlighed for at producere et givent sæt data. Lad $X = \{x_1, ..., x_n\}$ definere et sæt obseverede data og Θ det sæt parametre, som ligger bag sandsynlighedsmodellen, så kan likelihood og maximum likelihood defineres således:

$$L(\Theta|X) = p(X|\Theta) = \prod_{i=1}^{N} p(x_i|\Theta) \tag{2.2}$$

$$\Theta^* = \underset{\Theta}{argmax}\, L(\Theta|X) \tag{2.3}$$

Der er således en forskel mellem sandsynligheder og likelihood. Normalt når man taler om sandsynligheden, så er modellen kendt, mens data ikke er det. Men ved likelihood er det modsatte gældende. Her er data kendt, mens modellens parametre er ukendte. Med tabel 2.1 vil jeg illustrere forskellene. Når man taler om sandsynligheder, kigger man på kolonnerne. Fx er sandsynligheden for at få to gange plat ved fire kast med en mønt 5%, hvis man har en skæv mønt. Taler man om likelihood, så ser man på rækkerne og ikke kolonnerne. Her gælder det, at to gange plat ved fire kast giver en likelihood på hhv. 5% og 38%. Den mest sandsynlige model definerer som sagt maximum likelihood'en. I eksemplet er det derfor den fair mønt, dvs. mønten med 50% skævhed.

	Skævhed	
Plat	**0,1**	**0,5**
0	0,66	0,06
1	0,29	0,25
2	0,05	0,38
3	0,00	0,25
4	0,00	0,06
sum	1	1

Tabel 2.1: Likelihood

2.4 Bayesianske klassifikatorer

I overvåget læring anvender man Bayes theorem til at klassificere med. Har man et element $\overrightarrow{x} = (x_1, x_2, ..., x_n)$, som skal klassificeres, vil Bayes regel fortælle, at sandsynligheden for, at elementet tilhører klasse c, vil være:

$$p(c|x) = \frac{p(x|c)p(c)}{p(x)} \tag{2.4}$$

$p(c|x)$= Posterior sandsynlighed.
$p(x|c)$= Likelihood (af de observerede data givet klassen).
$p(c)$= Prior sandsynlighed for klassen.
$p(x)$= Normaliserings-konstant (prior sandsynligheden for de observerede data).

Da $p(x)$ er ens for alle klasser, kan man vælge at se bort fra den. Klassifikatoren vil da se således ud:

$$p(c|x) \propto p(x|c)p(c) \tag{2.5}$$

Resultatet er en distribution og ikke en værdi. Men man ønsker at få den klasse, som man finder mest sandsynlig, at x tilhører. Derfor vælger man det resultat, som giver højeste posterior sandsynlighed (maximum a posteriori - MAP):

$$f(x) : arg\,max(p(c|x)) \tag{2.6}$$

$p(x|c)$ og $p(c)$ estimeres på baggrund af træningssættet. $p(c)$ beregnes som antal elementer med klassen c / totale antal elementer.

Udfordringen kommer ved beregning af $p(x|c)$, som vil kræve en stor mængde data, hvis klassifikationen skal virke. Den skal kende hver mulig kombination af x givet klassen, ellers vil sandsynligheden blive 0. Med andre ord, hvis man skal bestemme klassen for et nyt element, så skal samtlige værdier i x have været observeret i træningssættet for mindst en klasse, ellers kan elementet ikke klassificeres.

2.5 Naiv Bayes

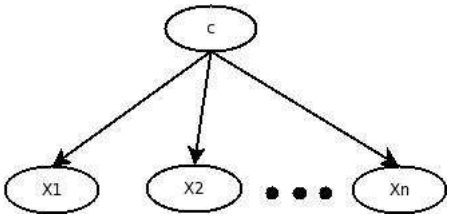
Naiv Bayes (herefter NB) løser problemet med beregningen af $p(x|c)$ ved antagelse om, at attributterne er uafhængige af hinanden givet klassen c. Derved har man, at klassifikatoren omskrives til en mere simpel model:

$$p(c|x) \propto p(x|c)p(c) = p(c) \prod_{i=0}^{n} p(x_i|c) \tag{2.7}$$

Det betyder, at vores træningssæt ikke behøver at kende alle kombinationer af værdierne i x. Det er nok, at værdierne findes, uafhængigt af hinanden, i datasættet for mindst en klasse. Grafisk kan modellen beskrives som vist i figur 2.3.

Der er stadig det problem, at hvis et element (givet en klasse) indeholder en værdi for en feature, og denne værdi ikke er kendt i træningssættet, så vil klassifikatoren ikke kunne bestemme det. Grunden er, at en af faktorene i 2.7 vil være 0, og derfor bliver hele resultatet 0.

Et andet problem er antagelsen om uafhængighed mellem attributterne givet klassen. Denne antagelse holder sjældent i virkligheden, deraf navnet “naiv” Bayes.



Figur 2.3: En NB klassifikator

Ifølge Miquélez et al. (2004) har modellen trods disse problemer vist sig meget effektiv til klassificering og kan ofte give resultater, der er sammenlignelige med mere komplekse klassifikatorer.[1] Desuden er den også beregningsmæssigt attraktiv pga. det simple design, og derfor er klassifikatoren også hurtig til at lære samt klassificere.

2.5.1 Selective Bayes

I Selective Bayes er det ikke nødvendigt at have alle attributter med i den endelige model. Det kan fx være, at en attribut ikke bidrager positivt til klassificeringen og derfor bør fjernes, hvilket vil resultere i en bedre model. Jensen (1996) argumenterer for, at redundante variable vil resultere i en generel dårligere performance af den naive klassifikator og derfor bør fjernes. Grunden er, at hvis nogle attributter ikke er betingede uafhængige (givet klassen), vil de få for stor vægt.[2]

2.6 Tree Augmented Naive Bayes - TAN

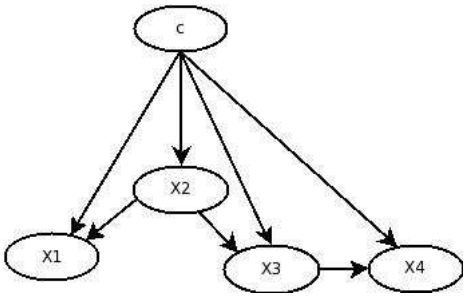
Denne bayesianske klassifikator går et skridt længere end NB ved, at modellen også tillader kanter mellem attributterne. Dermed kan man også beskrive afhængigheder mellem attributterne foruden sammenhængen mellem attributterne og klassen. TAN modeller har derfor ikke samme antagelse som NB om uafhængighed mellem attributterne givet klassen. Man kan dog ikke beskrive alle sammenhænge mellem attributterne, da TAN definitionen foreskriver, at en attribut ikke må have mere end to forældre, dvs. klassen samt en anden attribut. Denne begrænsning gør det muligt at lære det optimale sæt af kanter i polynomisk tid.[3]
Figur 2.4 viser et eksempel på en TAN model. Her ser man fx en kant fra X2 til X1, hvilket betyder, at X1's indvirkning på klassificeringen (C) også er påvirket af X2. Det giver

[1] I Miquélez et al. (2004, p. 340) henviser hun til en artikel af Domingos og Pazzani (1997), som viser dette.
[2] Jensen (1996, p. 36-37, p. 64).
[3] Friedman et al. (2001).

mulighed for mere nuancerede beskrivelser end ved NB. Lad os fx antage, at X1 har en overraskende lav værdi, men at den kan forklares ved, at X2 også er overraskende lav. Det vil betyde, at $p(x1|x2, c)$ bliver høj, selv om $p(x1|c)$ og $p(x2|c)$ er lave. TAN vil derfor bedre kunne klassificere sådanne emner. I NB er X1 og X2 uafhængige af hinanden, og derfor vil NB skulle vurdere to usandsynlige forekomster, hvilket medfører, at sådanne emner ikke vil blive klassificeret korrekt.

En TAN klassifikator defineres således:

$$p(c|x) \quad \propto \quad p(x|c)p(c) = p(c) \prod_{i=1}^{n} p(x_i|\text{parent}(x_i) \wedge c) \tag{2.8}$$



Figur 2.4: Tree Augmented Naive Bayes

Et centralt punkt i opbygningen af en TAN model er, hvordan den skal lære afhængighederne mellem attributterne. Da der kun må være en anden forælder ud over klassen, ønsker man at få de stærkeste forbindelser. En ofte anvendt metode til at finde disse er ved at beregne entropien mellem attributter givet klassen $I_P(A_i, A_j|C)$. Friedman et al. (1997) har lavet lavet en algoritme, construct-TAN, som tager udgangspunkt i denne betingede entropi.[4]

Både punkt 3 og punkt 4 kan resultere i forskellige træstrukturer, men det er blevet vist, at træerne er ækvivalente mht. klassificering.

Generelt er TAN bedre til at klassificere end NB, men prisen er en øget beregningsmæssig kompleksitet (når man taler om at lære modellen, klassificeringen er lige hurtig for begge modeller).

[4]"Construct-TAN" er en udvidelse af "Construct-tree" algoritmen defineret af Chow og Liu i 1968.

Algorithm 1 construct-TAN

1. Beregn $I_P(A_i, A_j|C)$ mellem alle attributter, $i \neq j$

   $$I_P(A_i, A_j|C) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{r=1}^{w} p(x_i y_j c_r) \log \frac{p(x_i, y_j|c_r)}{p(x_i|c_r)p(y_j|c_r)}$$

2. Lav en komplet graf, hvor knuderne svarer til attributterne ($A_1 ... A_n$).
   For hver kant i grafen tildel den vægten I_P.

3. Udtræk et "maximum weight spanning tree" :
   Vælg kanten med størst vægt og tilføj den til sættet *E*.
   For *i=2 to n* : Vælg kanten der nu har højst vægt, således det gælder, at $E^{i-1} \cup e_{max}$ ikke har en cyklisk reference

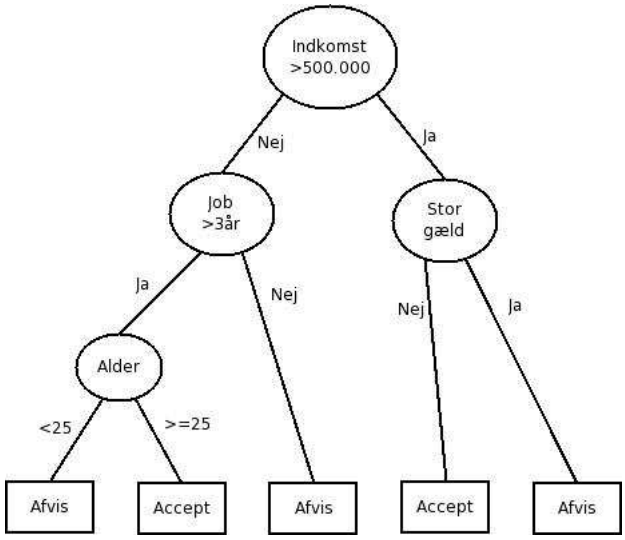
4. Retningsangiv træet. Vælg en knude og lad kantens retning gå fra denne og ud.

5. Konstruer TAN træet ved at tilføje klasse-variablen *C* med knuder fra denne til alle attributterne.

2.7 Beslutningstræer

Jeg vil i dette afsnit beskrive, hvad konceptet beslutningstræer dækker over. Den grundlæggende idé er at skabe så homogene grupper (hvad angår klasse) som muligt. Det sker ved rekursivt at partitionere sine data på sådan en måde, at i hver gruppe vil flertallet af elementer tilhøre samme klasse. Dvs. hvert split har til formål at øge ensartetheden i gruppen.
Selve klassificeringen i beslutningstræer sker top-down, hvor hver forgrening ned gennem træet svarer til en test på en attributværdi for det element, som skal klassificeres. Figur 2.5 viser et simpelt beslutningstræ for et tænkt eksempel for kreditvurdering. Skal man fx klassificere en ansøger, der er 25 år, har en indtægt på mindre end 500.000 kr. og en anciennitet på 3 år, vil træet fortælle os, at ansøgeren skal godkendes. Her ser man tydeligt en af fordelene ved beslutningstræer, nemlig at de forklarer deres valg - træerne er gennemskuelige og nemme at forstå. Den klarhed, som træerne giver, er også en af ulemperne. Hvorfor vil fx en person på 24 år og 11 måneder udgøre en større risiko end en person på 25 år?

Når man skal konstruere et beslutningstræ, vil man sigte efter at kunne klassificere et element med så få tests så muligt. Men hvordan finder man rækkefølgen på de variable, som forgreningen skal ske efter? Der findes flere forskellige metoder, men en effektiv metode har vist sig at være ved brug af entropi. Det er metoden, som C4.5 og C5.0 algoritmerne benytter sig af. Her foretages et split på den variabel, som giver den højeste informationstilgang. Lad os forestille os, at man har data som vist i tabel 2.2.



Figur 2.5: Et simpelt beslutningstræ

A_1	A_2	Klasse
A	1	X
A	0	Y
A	0	Y
B	1	X
B	0	Y
C	0	X
C	0	X
C	0	X

Tabel 2.2: Data brugt til beslutningstræ

$$\text{Entropi=Info(x)} = -\sum_{i=1}^{n} p_i \cdot log_2(p_i) \tag{2.9}$$

Man har, at 5 elementer tilhører klasse X, mens 3 tilhører klasse Y. Det betyder, at før man foretager et split, er entropien:
$-5/8 \cdot log_2(5/8) - 3/8 \cdot log_2(3/8) = 0,954$

Det split, man nu kan foretage, er enten på A_1 eller på A_2, og her vælger man den attribut, som giver den største informationsværdi. Dertil skal man bruge den forventede entropi, som er en vægtet sum af entropi over subsættet.

$$\text{Forventet entropi = Info}_X\text{(T) = } -\sum_{i=1}^{n} p_i \cdot Info(T_i) \tag{2.10}$$

Man beregner nu, hvad et split vil give:
$Info_{A_1}(T) = 3/8 \cdot Info(T) + 2/8 \cdot Info(T) + 3/8 \cdot Info(T) = 0,92$
$Info_{A_2}(T) = 6/8 \cdot Info(T) + 2/8 \cdot Info(T) = 0,81$

Slutteligt vælger man den variabel, som giver den højeste informationstilgang (største reduktion af entropi), beregnet således:

$$Gain(x) = Info(T) - Info_x(T) \tag{2.11}$$

$Gain(A_1) = 0,954 - 0,92 = 0,034$
$Gain(A_2) = 0,954 - 0,81 = 0,144$

Man vælger derfor at foretage det første split på variabelA_2.

2.7.1 Beslutningstræer og manglende data

Beslutningstræer har et særligt hensyn at skulle tage, når der er manglende data - både under træning og under brug (klassificering). Ved klassificering hvad skal algoritmen da gøre, når der mangler data på en variabel, der forgrenes på? Og under træning, hvilket subsæt skal et element da tilhøre? En nem metode er at smide elementer ud, der har manglende data - dvs. definere problemet bort. Men ofte er det ikke en brugbar metode, fx hvis der mangler data i mange af elementerne. Og under klassificering er det ikke en mulighed (svaret kan jo i så fald ikke gives).

Når det drejer sig om læring, så løser C4.5 problemet ved at gøre tre ting. For det første vil beregningen af $Info(S)$ og $Info_X(T)$ kun medtage de elementer, hvor der er en attributværdi. For det andet ændres $Gain(X)$ funktionen således, at den multipliceres med sandsynligheden for, at en given attribut er kendt:

$$Gain(x) = p(x) \cdot (Info(T) - Info_x(T)) \tag{2.12}$$

For det tredje tilføjes en vægt W til hvert element: Hvis en case (et element) er fuldt observeret, kan man med en sandsynlighed på 1 sige, hvilket subsæt i træet det tilhører. Men ved manglende data kan man ikke med samme sikkerhed sige, hvilket subsæt af beslutningstræet et element tilhører. Derfor bruger man vægten W til at afspejle sandsynligheden for, at det tilhører et muligt subsæt:

$$W_{new} = W_{old} \cdot p(T_i) \tag{2.13}$$

I tabel 2.3 ses en tabel, som viser resultatet af en forgrening, hvor der har været manglende data på attribut 1 (A_1) - dvs. elementet ser således ud:
$A_1 = ?$, A_2 = 40, Klasse = X

T_1:Attribut 1 = A				T_2:Attribut 1 = B		
A_2	Klasse	W		A_2	Klasse	W
20	X	1		30	Y	1
35	Y	1		40	X	1
20	X	1		40	X	3/7
45	Y	1				
40	X	4/7				

Tabel 2.3: Forgrening hvor der har været manglende data

Af tabel 2.3 kan man udlede følgende regler:

If (attribut1 = A) Then
... If (A_2<=20) Then
..... Klasse = X $(2/0)$
... Else Klasse = Y $(2,57/0,57)$

If (attribut1 = B) Then
... If (A_2<=30) Then
..... Klasse = Y $(1/0)$
... Else Klasse = X $(1,43/0)$

$(2,57/0,57)$ fortæller, at 2,57 elementer i træningssættet blev klassificeret af denne regel, hvoraf de 0,57 elementer blev klassificeret forkert.

Samme princip anvendes ved klassificering, når træet får et hidtil ukendt element, dvs. et element med attributværdier, som ikke kendes af træet. Attributværdien bliver i så fald opfattet som manglende data, og klassificeringen bliver den vej gennem træet, der giver den højeste sandsynlighed for en klasse. Det medfører en mere kompliceret beregningsmodel, da alle tilladte veje gennem træet skal undersøges, beregnes og sammenlignes.

Der findes dog forskellige andre metoder til at estimere de manglende værdier. Det vil jeg komme nærmere ind på i afsnit 2.9.

2.8 Boosting

Mange læringsproblemer er så komplekse, at det er svært at finde en model, som kan klassificere præcist nok. Men i stedet for at nøjes med en universel model, er det måske bedre at have flere modeller, der hver især er gode til at klassificere en delmængde af det samlede problem. Boosting tager udgangspunkt i denne tanke. Det er en metode til at kombinere flere forskellige klassifkatorer med det formål at opnå en bedre klassifikation, end hvad en enkelt kan formå. En analogi kunne være, at et råd af eksperter ville give bedre svar sammenlignet med, hvad en enkelt ekspert kan.

En ofte anvendt boosting metode er AdaBoost-algoritmen, som er opfundet af Freund og Schapire i 1995. Her er den grundlæggende ide at konstruere en række klassifikatorer, hvor det gælder, at hver klassifikator fokuserer mest på de fejl, som den forrige klassifikator lavede. Derved får de elementer, som er svære at klassificere, mere fokus i den efterfølgende model. Den endelige boosted model fungerer ved, at hver klassifikator stemmer om resultatet. Stemmeafgivelsen er vægtet efter, hvor godt klassifikatoren performede. Mere formelt kan stemmeafgivelsen beskrives således:

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x)) \tag{2.14}$$

hvor,
H= Den boosted klassifikator.
x= elementet der skal klassificeres (resultatet $\in \{-1, 1\}$).
T= antallet af klassifikatorer til brug for boosting.
α=klassifikatorens vægt (mål for hvor præcis den er).
h= klassifikatoren.

Algorithm 2 De grundlæggende principper i boosting

1. Giv alle elementer i træningssættet vægten $1/n$.
2. Byg en klassifikator (h_i) baseret på et subsæt af træningssættet.
3. Find de elementer som ikke blev klassificeres korrekt i det fulde træningssæt.
4. Giv dem en højere vægt (boost dem).
5. Gå til punkt 2 hvor $i = i + 1$.

En interessant mulighed med boosting er, at klassifikatorerne ikke nødvendigvis behøver at være af samme type. Fx kan man kombinere både NB og beslutningstræer.

2.9 Manglende data

Et problem, der ofte opstår, når man skal analysere data, er, at det indeholder ukomplette data. Selv hvis datagrundlaget er meget stort, kan andelen af rækker indeholdende alle oplysninger være minimal. Tag som eksempel, at datagrundlaget er skabt på baggrund af et spørgeskema. Her kan man nemt forestille sig, at nogle personer ikke svarer på alle spørgsmålene. Nogle metoder kan godt håndtere manglende data (ved fx at se bort fra dem), mens andre ikke kan. Eksempler på nogle, der kan, er NB og TAN. Men problemet kan også løses ved reparation af data. Nedenstående liste beskriver nogle af de ofte anvendte metoder til dette.

Manuelt: Hvis der er få manglende data, og hvis disse kan estimeres af fx en domæneekspert, kan man manuelt rette datasættet. Men hvis der ikke er en oplagt værdi for alle manglende data, vil denne metode tilføje uønsket støj.

Mean imputation: En manglende værdi udskiftes med middelværdien for samme feature.

Hot deck imputation: Tag værdien fra den case, som bedst ligner den, man har manglende data på.

Cold deck imputation: Her substitueres de manglende data med værdierne fra den foregående række.

Regression: Ud fra de observerede cases finder man en funktionel sammenhæng mellem featurene og den variabel, som har manglende data. Har man så en case, hvor der mangler data, indsættes denne i regressionsformlen, og resultatet er et estimat for den manglende attributværdi.

Expectation Maximization (EM metoden): Denne metode er modsat de andre iterativ. Her gennemløbes en proces, hvor manglende data erstattes med den værdi, som giver den højeste likelihood. Se også afsnit 2.10, hvor jeg uddyber denne metode.

Hver af disse metoder har fordele og ulemper, men før man vælger en af disse, skal man have klarlagt, hvorfor data er manglende. Det handler her om at forstå den mekanisme, som skaber de manglende data. Little et al. (1987) har defineret tre forskellige typer af manglende data mekanismer, som jeg vil gennemgå i det følgende.

Missing competely at random - MCAR: De manglende attributværdier optræder fuldstændig uafhængigt af de observerede og manglende data. Dvs. sandsynligheden for, at observation x_i mangler, er uafhængig af værdien af x_i og andre variable. Fx hvis alle, uanset lønindkomst, er lige tilbøjelige til ikke at oplyse husstandsindkomst. Derimod hvis det kun er folk med høj indkomst, der ikke ville oplyse husstandsindkomsten, så ville MCAR ikke gælde.

Missing at random - MAR: Her kan en manglende attributværdi afhænge af andre attributter, men ikke af attributten selv, dvs. de manglende data kan forklares alene ud fra de observerede data. Hvis fx folk med høj indkomst typisk ikke vil oplyse husstandsindkomst, er de manglende data i husstandsindkomst forklaret ved indkomstvariablen, og derfor har man MAR.

Missing not at random - MNAR: Her skyldes de manglende data selve værdien af attributten, og de kan ikke forklares på baggrund af andre variable i datasættet. Et eksempel kunne fx være, at straffede personer er tilbøjelige til ikke at fortælle det. Hvis man har MNAR, fungerer ingen af de beskrevne metoder til data-reparation, da der ikke er nok information til rådighed i datasættet til at estimere de manglende værdier. Det er derfor nødvendigt at kende den mekanisme, som skaber de manglende data.

2.9.1 Manglende data og selektionsmekanismen

I indledningen nævnte jeg, at reject inference problemet kan ses som et manglende dataproblem, hvor de manglende data findes på registreringen af, om en kunde er god eller dårlig. Grunden til, at man gerne vil kende de manglende data (performancen for ansøgerne), er en antagelse om, at man derved kan bygge en bedre model til udvælgelse. Det interessante er derfor, hvordan selektionsmekanismen skaber de manglende data. Denne viden er især vigtig, fordi den har betydning for, hvordan ens modeller skal konstrueres. Jeg vil i det følgende beskrive dette nærmere.

Den information, man har tilrådighed, er variable, som beskriver forskellige karakteristika (alder, løn etc.) for ansøgeren: $x = \{x_1, x_2, ..., x_n\}$. Man har også en selektionsvariabel $s \in \{A, R\}$, som fortæller, om ansøgeren blev accepteret (A) eller afvist (R). Desuden har man en klassifikationsvariabel $y \in \{G, B\}$, der fortæller, om kunden er klassificeret som god eller dårlig.

Under MCAR gælder det, at selektionsvariablen s er fuldstændig uafhængig af x og y. Formelt kan det beskrives således:
$s \perp x, y$ hvilket medfører, at:
a/ $p(s|x, y) = p(s)$
b/ $p(y|x, s) = p(y|x)$

MCAR vil opstå, hvis selektionen sker ved tilfældig udvælgelse, fx hvis banken accepterer hver anden ansøger, der efterspørger om lån. Ved MCAR kan man basere sin model udelukkende på information om de accepterede cases jf. punkt b.

Man har MAR, hvis oplysninger om ansøgeren er bestemmende for, om et lån gives eller ej. Det betyder, at selektionsvariablen s vil være afhængig af x, men uafhængig af y.

Matematisk kan det beskrives således:
$s \perp y|x$ hvilket medfører, at:
a/ $p(s|x,y) = p(s|x)$
b/ $p(y|x,s) = p(y|x)$

Ved MAR kan man enten antage en frekventistisk eller bayesiansk opfattelse. Vælger man den frekventistiske antagelse, er distributionen af y den samme for både de afviste og de godkendte, da man jf. b har:

$$p(y|x, s = A) = p(y|x) = p(y|x, s = R) \tag{2.15}$$

Det betyder, at modellen kan nøjes med at lære fra $p(y|x)$, hvilket medfører, at man kun ser på de accepterede ansøgere. De afviste vil ikke indeholde nogen viden om y, for som frekventisten siger, så er frekvensen, hvormed hændelsen fandt sted, lig 0.

Har man derimod en bayesiansk opfattelse, ignorerer man ikke de manglende data for y. Resultatet er, at man får man en mere generel model, der inddrager de afviste ansøgere. Her skal modellen lære:

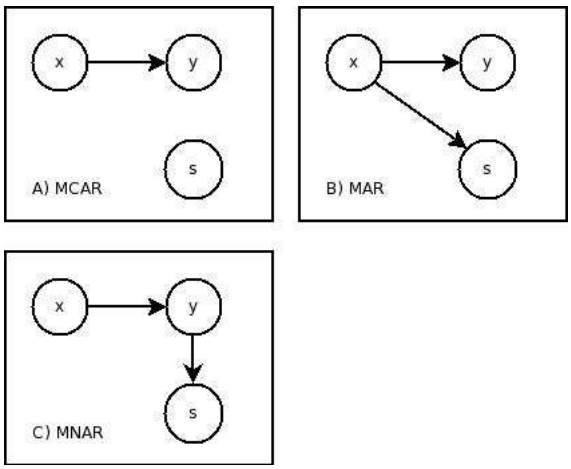
$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} \tag{2.16}$$

Men man kender hverken $p(x|y)$ eller $p(y)$ for de afviste, så derfor skal de estimeres. En ofte anvendt metode til dette er EM-algoritmen, som jeg vil beskrive nærmere i afsnit 2.10.

Under MNAR gælder det, at selektionen s er afhængig af både x og y. Dvs. $p(s|x,y) \neq p(s|x)$. Man er derfor nødt til at vide noget om, hvordan de manglende data opstår, dvs. modellen for selektionen.

Figur 2.6, viser hvordan de manglende data-mekanismer ser ud for selektionen, hvis man modellerer dem som bayesianske netværk.

Jeg forudsætter i dette speciale, at det manglende data-problem er MAR. Det er under antagelse om, at bankens selektionsmekanisme virker efter objektive kriterier, som tager udgangspunkt i de oplysninger, banken har om ansøgeren. Evt. subjektive vurderinger af, om et lån skal bevilliges eller ej, kan selvfølgelig forekomme, men jeg forudsætter, at de ikke vil have nogen signifikant indvirkning på modellen.



Figur 2.6: Forskellige bayesianske netværk, som beskriver manglende data-problemet

2.10 EM algoritmen

En af de mest brugte metoder til parameter-estimation, når man har manglende data, er EM algoritmen. Det er en generel metode til at finde maksimum likelihood for ukendte parametre. Det er en iterativ algoritme, som grundlæggende udfører to opgaver pr. iteration: Forventning (Expectation - E) og maksimering (M). M-delen maksimerer likelihooden baseret på det estimat, som er lavet i E-delen. Simplificeret ser det således ud:

1. Gentag indtil et fastsat stop-kriterium opnås.
2. Estimer de manglende værdier.
3. Estimer parametrene .
4. Gentag, hvor man i punkt 2 anvender de estimerede værdier for parametrene.

Definitioner

Hvis man har et datasæt, hvori der er ukomplette data, kan man inddele det i en observeret del og i en ukendt del. Lad da $X = \{x_1, ...x_n\}$ være en vektor med de observerede data og $Z = \{z_1, ..., z_n\}$ en vektor af de manglende data. Det fulde datasæt vil da være $Y = (X, Z)$. Hvis Θ er modellens parametre, kan tæthedsestimatet defineres som:

$$p(y|\Theta) = p(x, z|\Theta) = p(z|x, \Theta)p(x|\Theta) \tag{2.17}$$

Hvorefter likelihood-funktionen for det fuldt observerede datasæt bliver:

$$L(\Theta|Y) = L(\Theta|X,Z) = p(X,Z|\Theta) \tag{2.18}$$

Det er denne (log) likelihood, som EM vil maksimere i hvert iteration. Det sker ved, at algoritmen forsøger at forbedre estimatet for Θ^{i+1} ved at bruge informationen fra den foregående model Θ^i. Man kan nu definere en funktion Q som værende forventningsværdien af log likelihood-funktionen for det fulde datasæt *givet* de observerede data samt det nuværende parameterestimat Θ^i:

$$Q(\Theta^{i+1}|\Theta^i) \equiv E_Z[log\, p(X,Z|\,(\Theta^{i+1}|X,\Theta^i])\,) \tag{2.19}$$

Med andre ord er Θ^i det nuværende parameter-estimat, som anvendes til at beregne forventningen**,** mens Θ^{i+1} er de nye parametre, som man anvender til at øge Q.

EM iterationerne kan nu defineres mere præcist:

E-Skridt: Beregn $Q(\Theta^{i+1}|\Theta^i)$.
Estimer distributionen af de manglende data baseret på de nuværende parametre Θ^i.

M-Skridt: Find det Θ^{i+1} som maksimerer $Q(\Theta^{i+1}|\Theta^i)$.
Find de parametre, som maksimerer den forventede log likelihood af de virtuel observerede data.

Iterationerne gentages indtil et fastlagt stopkriterie er opnået: $L(\Theta^{i+1}) - L(\Theta^i) < \delta$. Det er blevet vist, at algoritmen garanterer, at hver iteration ikke mindsker log likelihooden af de observerede data. Den garanterer yderligere, at den mindst finder et lokalt maksimum[5].

2.10.1 EM-algoritmen og bayesianske netværk

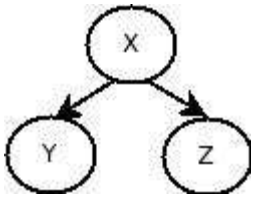
Jeg vil i dette afsnit demonstrere, hvordan EM algoritmen kan bruges i forbindelse med bayesiansk netværk.[6] Lad os antage, at man har en simpel model M, der indeholder variablene $U = \{X, Y, Z\}$ som vist i figur 2.7.

Parametrene Θ_{ijk} er de betingede sandsynligheder: $P(U_i = k|parent(U_i) = j)$. Dvs. man har følgende tre tabeller:
$P(X = k)$, $P(Y = k|X = j)$ og $P(Z = k|X = j)$

Hvis nu datasættet D er fuldt observeret, vil man kunne finde maksimum likelihooden ved blot at tælle. Men hvis der er manglende data, er det ikke muligt at gøre sådan.

[5]Af Dellaert (2002) fremgår det, at Dempster et al. (1977) giver et bevis for denne påstand. (Det er Dempster et al. som først beskriver EM algoritmen).

[6]Løst efter Jensen et al. kommende bog: "Bayesian Networks and Decision Graphs II".



Figur 2.7: Et simpelt bayesiansk netværk

Derfor beregner man det forventede antal ved at erstatte hver manglende observation med en sandsynlighedsfordeling. Det giver følgende EM skridt:

E-skridt: Beregn de forventede værdier for hver familie (variabel og dens forældre):

$$E_{\Theta^t}[N(U_i, parent(U)_i \mid D] = \sum_{d \in D} P(U_i, parent(U_i)|d, \Theta^t) \tag{2.20}$$

M-skridt: Anvend de estimerede værdier, som om de var observerede, og beregn det nye maksimum likelihood estimat for parametrene:

$$\Theta^* = \frac{E_{\Theta^i}[N(U_i = k, parent(U)_i = j \mid D]}{\sum_k E_{\Theta^i}[N(U_i = k, parent(U)_i = j \mid D]} \tag{2.21}$$

case	**X**	**Y**
1	?	Pos
2	yes	neg
3	yes	pos
4	yes	pos
5	no	neg

Tabel 2.4: En simpel database med et manglende element

Lad os antage, at man har en database, som ser ud som tabel 2.4, hvor man har en kausal sammenhæng gående fra X til Y. (Fordi X ikke har nogen forældre, så er X's familie X selv.) I denne vil man have estimeret værdien for X i case 1. Startende med en uniform a priori, og da den manglende værdi kan have to tilstande, så er bidraget fra hver 50%. De fuldt observerede rækker bidrager med 100%. Det betyder, at man nu kan finde den forventede værdi E[$N(X = yes)$] ved at lægge sandsynlighederne sammen: $0,5 + 1 + 1 + 1 = 3.5$. Det er E delen af algoritmen. Det næste er at maksimere. Det gøres ved at opdatere sandsynlighedstabellen med de værdier, man fandt i E skridtet, dvs. $p(X = yes) = 0,7$. Algoritmen gentages, indtil δ er tilpas lille. I tabel 2.5 ses, hvordan hver iteration får sandsynlighederne til at ændre sig.

Iteration	yes	no	Δ
0	0.5	0.5	
1	0.7	0.3	0,2
2	0.74	0.26	0,04
3	0.748	0.252	0,008
4	0.7496	0.2504	0.0016
5	0,74992	0,25008	0,00032

Tabel 2.5: De betingede sandsynligheder under EM processeringen

Man kan også beregne, hvad algoritmen konvergerer til. Nævneren i M-skridtet er 5, og tælleren er 3 + x, hvor 3 er antallet af "yes", mens x definerer den ukendte andel. Dvs.:

$$\frac{3+x}{5} = x \Rightarrow 3 + x = 5x \Rightarrow x = 3/4$$

Kapitel 3

Beskrivelse af empirisk materiale

Det datamateriale, som jeg har fået stillet til rådighed af en bank vha. min vejleder, stammer fra ansøgninger om kreditkort med mulighed for lån for op til 30.000 kr. Datasættet har 20.466 rækker med 36 attributter, som beskriver karakteristika for lånansøgeren, fx årsindkomst og antal børn. Desuden har banken tilføjet 3 "speciale" attributter: Om det er en god eller dårlig kunde, om ansøgeren blev accepteret eller ej samt kreditmaksimum på kortet.

Dataene er udtrukket fra et operationelt system, og det indeholder "beskidte data". Derfor er det nødvendigt først at foretage en data-rensning. Formålet er at rette/slette data, som er forkerte og derfor vil generere støj i analysen, fx datoer som er formateret forkert/forskelligt. Et andet formål er at skabe konsistens mellem attributværdier. Fx er der naturligt nok den funktionelle sammenhæng mellem alder og anciennitet, at sidstnævnte ikke kan være højst. I de tilfælde, hvor der eksisterer data, som er forkerte, vil de blive renset. En rensning kan være flg.:

- Sletning af attribut-værdi.
- Modificering af attribut-værdi.
- Sletning af hele rækken.

Men der er også problemer med, hvorledes data er struktureret. Strukturen i dataene fra banken har ikke en form, der er optimal for at kunne foretage analyser. Fx er tidsmålinger opdelt i to kolonner: en for år og en for måneder. Det er derfor hensigtsmæssigt at lade de udtrukne data gennemløbe en række transformationer for at få dem omstruktureret således, at de får den struktur, der bedst understøtter de analyser, jeg skal foretage.

Det er dog umuligt at foretage denne transformation automatisk, uden først at have undersøgt attributterne og deres domæne. Inden for datawarehousing kaldes hele denne proces for ETL: Extraction, Transformation and Loading.

Datasættet indeholder både diskrete og kontinuerte variable, men da de fleste datamining metoder og værktøjer har det bedst med diskrete variable, er de kontinuerte variable blevet diskretiseret. En anden grund til at anvende diskrete variable er, at man kan repræsentere værdier, som man ikke kender i sit datamateriale. Derved kan man under læring af modellen fx repræsentere alle indkomster mellem 0-20.000 kr., ved at give dem en kategori. Det er at foretrække frem for kun at kunne repræsentere de eksakte værdier, som der måtte findes i datamaterialet.

Det sidste, jeg har gjort, er at slette kolonner, som jeg antog for at være uvæsentlige. Fx hvis der har været meget dårlige svarprocenter, eller hvis jeg vurderede attributtens relevans for værende begrænset. Et eksempel er spørgsmålet om, hvilke telefonnumre ansøgeren oplyste (hjemmenummer, arbejdsnummer etc.). Det er en ren subjektiv vurdering, jeg har anlagt til udvælgelse, og en mere analytisk gennemgang ville have været at foretrække. Dette valg har jeg truffet af tidsmæssige årsager.

Tabel 3.1 giver et overblik over de attributter, jeg fortsætter med. En mere detaljeret beskrivelse findes i bilag 2.

	Attribut	**Værdi**	**Beskrivelse**
1	BOERN	“”,A,B,C,D	Antal børn
2	BOLIGFORM	“”,C,F,H,K,L	Fx Hus eller lejlighed
3	BOLIGTYPE	“”,A,E,L	Fx leje
4	BESKAEFTIGELSE	“”,F,H,I,L,P,S,T,U	Fx ledig eller funktionær
5	ANCINNITET	“”,A,B,C,D,E,F	Tid på nuværende job i måneder
6	AKASSE	“”,J,N	Medlem af A-kasse
7	AKASSE_TID	“”,A,B,C,D,E,F	Medlemskabs-periode i måneder
8	BRUTTO_AAR	“”,A,B,C,D,E,F,G,H,I	Årlig bruttoløn
9	NETTO_MM	“”,A,B,C,D,E,F,G,H,I,J,K	Månedlig nettoløn
10	PI_TID	“”,A,B,C,D,E,F	Hvor længe har ansøgeren været i banken
11	MEDANS	“”,J,N	Om der er en medansøger
12	CIVILSTAND	“”,A,E,G,K,N,O,S	Fx gift, skilt og alene
13	KOEN	M,K	Køn
14	ALDER	“”,A,B,C,D,E,F,G,H,I,K	Alder i måneder
15	LAANE_MAX	A,B,C,D,E,F	Kreditmaksimum i kr.
16	RKI_HIT	“”,J,N	Registeret i RKI
17	DCODE	A,R	Accepteret eller afvist
18	GB	“”,B,G	God eller dårlig kunde

Tabel 3.1: Beskrivelse af data

Kapitel 4

Analyse

Frem til nu har jeg beskrevet den generelle problemstilling, der har med kreditvurdering at gøre. Jeg har også præsenteret den værktøjskasse, i form af teorier og modeller, jeg vil anvende til at efterprøve min ide om, hvordan en bedre model til selektion kan skabes. I dette kapitel vil jeg blive mere konkret. Jeg vil anskue problemet set med bankens øjne. Hvad kan banken gøre for at finde ud af noget om dens kunder og systemer? Og hvordan kan den viden omsættes til brug for konstruktion af bedre og mere rentable modeller til udvælgelse af fremtidige kunder?

4.1 Datamining

Datamining er en proces, der har til formål at finde implicit, tidligere ukendt og potentielt brugbar information fra data.[1] Til det anvendes forskellige værktøjer til at identificere interessante mønstre og sammenhænge i data, som kan anvendes til forudsigelse, klassificering eller beslutningstagning. I traditionel (statistisk) analyse får man sin konklusion ved at efterprøve en hypotese, dvs. man på forhånd har en ide om, hvad man leder efter. I datamining går man den anden vej rundt og spørger, om man ikke kan få en konklusion på baggrund af de data, man har. Man har således ikke på forhånd en hypotese. Nogle af de problemer, man står over for, når man skal lave datamining, er: Store datamængder, manglende/ukomplette data, fejl i data og komplekse datastrukturer.

4.2 Datamining af bankens data

Jeg vil i dette afsnit analysere bankens data med henblik på at få viden om, hvordan bankens selektionsmodel virker. Dvs. jeg bygger en model, som beskriver bankens ud-

[1] Frawley et al. (1992, p. 213-228)

vælgelsespraksis. Modellen skal anvendes til sammenligning med den model, jeg senere får lavet - simpelthen for at se, om den model, jeg får konstrueret, er bedre end den nuværende. Et andet vigtigt punkt, som jeg vil opnå med analysen, er at finde forskellene mellem de gode og dårlige kunder. Den viden skal bruges til at lave min nye model med.

4.2.1 Manglende data

I bankens datasæt er der en stor andel af manglende data, og derfor er det at foretrække at estimere værdierne frem for at slette rækker i datasættet. Hvis modellerne er bayesianske, anvendes EM-algoritmen, mens beslutningstræerne beregner en sandsynlighedsfordeling som vist i afsnit 2.7.1.

Jeg vil i det næste kort beskrive inputtet til EM-algoritmen.
Datasættet $\mathcal{D}$ indeholder to populationer. De, som er accepteret af banken ($\mathcal{A}$), og de, som er afvist ($\mathcal{R}$). Dvs. $\mathcal{D} = \{\mathcal{A}, \mathcal{R}\}$. For begge populationer gælder, at der er manglende data på features, mens det for $\mathcal{R}$ også gælder, at der ingen oplysninger er om variablen, der beskriver kundens performance. Spørgsmålet er nu, hvordan EM skal anvendes for at estimere de manglende data. Skal det ske på baggrund af population $\mathcal{D}$,$\mathcal{A}$ eller $\mathcal{R}$? Svaret er, at det kommer an på formålet:

- Selektions-modellen: Her estimeres på baggrund af hele datasættet $\mathcal{D}$, da det er en model, som beskriver alle kreditkort-ansøgerne.
- Performance-modellen: Her vælger jeg at køre EM på $\mathcal{A}$ alene, fordi jeg her kun vil se, hvordan den accepterede gruppe arter sig. Derfor er de afviste ikke interessante.

4.2.2 Selektionsmodellen

Jeg har ikke kendskab til, hvorledes bankens selektion foregår, dvs. efter hvilke kriterier den udvælger og fravælger ansøgere. Derfor er det første skridt for at forstå denne mekanisme, at analysere bankens data. Opgaven er et typisk klassificeringsproblem, hvor man skal lære at skelne mellem to typer: De som accepteres, og de som afvises. Inden jeg starter med selve analysen, inddeler jeg mit datasæt i to[2]: et træningssæt og et testsæt. Træningssættet anvendes til at bygge modellerne med, mens testsættet anvendes til at evaluere, hvor god klassifikationen er. Datasættet er inddelt således, at 2/3 anvendes til træning og 1/3 til test.

[2]Denne metode kaldes for "holdout".

Til at identificere selektionsmodellen anvender jeg forskellige former for klassifikatorer[3] for at se, hvilken der i min situation performer bedst. Dem, jeg vil bruge, er: NB, Selective Naive Bayes (SNB), TAN, C5-træer samt boostet C5-træer. I tabel 4.1 ses resultatet - tallene angiver fejlraten i procent. Tallene 10 og 50 fortæller, hvor mange gange boosting er foretaget.

Navn	ε	**Placering**
NB	11,79	5
SNB	7,66	4
TAN	13,53	6
C5	1,92	3
$C5_{10}$	1,78	2
$C5_{50}$	1,74	1

Tabel 4.1: Resultat af klassificering

Det er tankevækkende at se, hvor god klassifikationen er. Jeg havde selvfølgelig en ide om, at selektionsmodellen ville performe godt, fordi den bygger på regler defineret af banken. Men en fejlrate på <2% er næsten for godt til at være sandt. Forklaringen får man, når man undersøger den model nærmere, som SNB danner. Den indeholder kun en attribut ud over klassifikationen, nemlig RKI_HIT. Forklaringen må være, at banken kun indhenter RKI oplysninger for de ansøgere, som er accepteret af deres model. Dvs. at der ikke er RKI oplysninger om de ansøgere, de vidste, de *ikke* ville have. Derfor er RKI_HIT variablen blot en indikator på et afslag. Modellen er med andre ord god til at give et afslag baseret på et afslag. Jeg fjerner derfor RKI_HIT attributten fra datasættet og laver nogle nye modeller. Resultatet fremgår af tabel 4.2.

Navn	ε	**Placering**
NB	18,08	6
SNB	17,75	4
TAN	17,95	5
C5	15,27	3
$C5_{10}$	13,62	2
$C5_{50}$	13,17	1

Tabel 4.2: Resultat af klassificering

Det er her overraskende at se, hvor stor forskel der er på C5 og de bayesianske metoder. Det ses også tydeligt, hvor meget boosting betyder. Det skal dog siges, at jeg ikke har fundet noget software, som kan booste de bayesianske net. Men ud fra resultaterne

[3]Til dette formål har jeg anvendt flg. software: Poulin-Hugin, Clementine samt bci/bcx skrevet af Christian Borgelt fra University of Magdeburg.

kunne det tyde på, at boosting hjælper meget på klassificeringen.

Det sidste jeg undersøgte, og som kunne have indflydelse på performancen, var forekomsten af tvetydig data. Dvs. om der er rækker med ens værdier, men med forskellig konklusion på klassificeringen. I hele datasættet har jeg kun fundet 38 rækker (<0,2%), så tvetydighed kan udelukkes.

4.2.3 Performancemodellen

Ligesom for selektionsmodellen har jeg valgt at bruge NB, SNB, TAN, C5-træer samt boostet C5-træer. Jeg arbejder her kun på den del af datasættet, hvor en ansøger er accepteret. Datasættet er også her inddelt sådan, at 2/3 anvendes til træning og 1/3 til test. Resultaterne fremgår af tabel 4.3.

Navn	ε	**Placering**
NB	22,01	6
SNB	21,28	5
TAN	20,75	4
C5	16,29	3
$C5_{10}$	9,62	2
$C5_{50}$	7,8	1

Tabel 4.3: Resultat af klassificering

Man ser ligeledes her, at der er en stor forskel mellem C5-træerne og de bayesianske modeller, men forklaringen skyldes sikkert boosting.

Det, at C5 klarer sig så godt, fik mig til at tænke på, om en partitionering af dataene ville gøre de bayesianske metoder bedre. Det, som beslutningstræerne gør, er at dele træningssættet i mindre bidder, hvor hver bid, dvs. forgrening, kan opfattes som en selvstændig model. Den tanke bruger jeg til at lave forskellige bayesianske net med. C5 træets første split er på attributten AKASSE. Her bliver træet delt i tre:

1. AKASSE = J.
2. AKASSE = N.
3. *AKASSE = null.*

Jeg laver derfor en model for hver af disse og måler deres performance.[4] Resultatet er, at det ikke kan betale sig at partitionere dataene (efter denne metode) - et boostet C5 træ er stadig bedst.

[4]Trænings- og test-sættene deles med andre ord i tre.

	null	JA	Nej
Antal	1657	1760	149
NB	5,07	35,74	32,21
SNB	3,8	35,11	30,11
$\mathbf{C5}_{50}$	3,74	13,12	11,41

Tabel 4.4: Klassificeringsevne ved partitionering på medlemskab i AKasse

En lille kuriositet er, at hvis der ingen oplysninger findes om A-kasse, så performer modellen forbløffende godt. Jeg kan ikke komme med en fornuftig forklaring på dette, og af tidsmæssige årsager har jeg ikke foretaget en nærmere analyse af dette fænomen. Jeg kan dog sige, at det ikke skyldes, at alle i denne gruppe enten accepteres eller afvises.

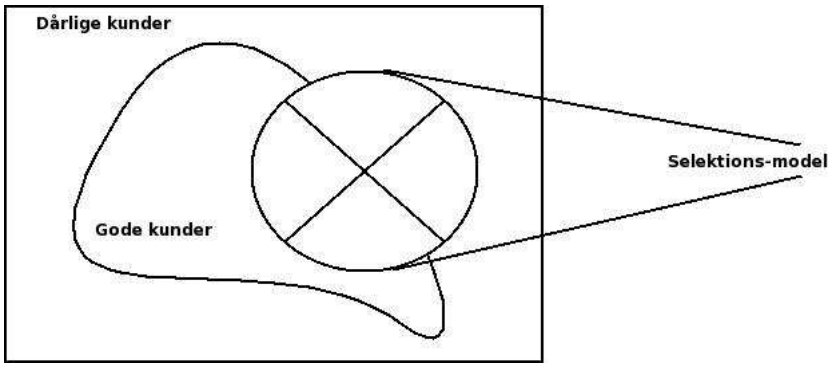
Jeg har også undersøgt for tvetydig data, men har kun fundet 15 elementer (0,11%). Så heller ikke her er der tale om, at det er tvetydig data, som gør klassificeringen dårligere.

4.3 Modelforbedring

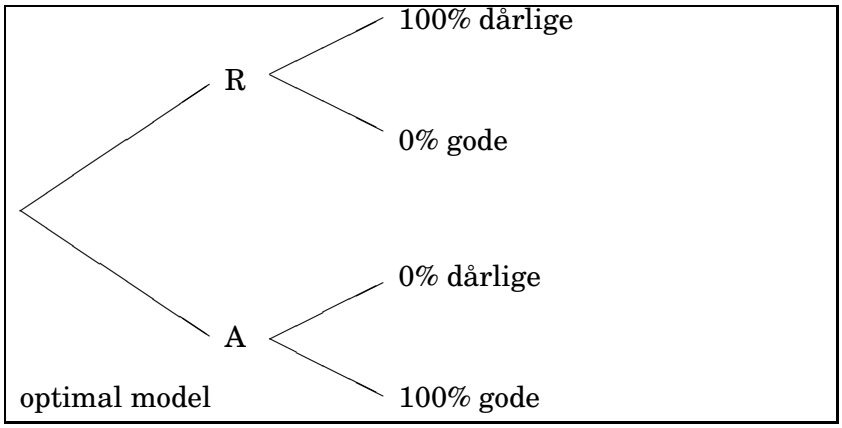
Jeg har nu analyseret bankens data, og næste skridt bliver at tage denne viden og omsætte den, således at jeg kan få udviklet en ny og bedre model til udvælgelse af de bedste kreditkort-ansøgere. Denne viden samt min antagelse om, at reject-inference problemet løses ved at fastholde et specifikt forhold mellem de, som accepteres, og de som afvises[5], benyttes til at konstruere den nye model. Men hvad betyder det i praksis, at man fastholder et specifikt A/R-forhold? Jo det betyder, at modellen kun kan forbedres, hvis andelen af gode kunder kan øges. Hvis A/R-forholdet fx er 75/25 (75% accepteres, og 25% afvises), skal jeg forsøge at udvælge 75% af ansøgerne, som er bedre end de, som den nuværende model finder. Jeg har lavet et lille illustrativt eksempel i figur 4.1. Sigtekornet viser hvilke ansøgere, der accepteres af modellen, og det handler derfor om at få flyttet det, således at flere af de gode kunder tages med.

Det er hensigten, at selektionsmekanismen skal undgå at fravælge gode ansøgere og medtage dårlige. I en optimal - men urealistisk - situation skal modellen vælge alle de gode ansøgere og fravælge samtlige dårlige. Dette er illustreret i figur 4.2. Det er dog ikke muligt at lave en optimal model, pga. hvad man normalt kalder den menneskelige faktor. Man kan fx forestille sig, at alle kendetegn for en ansøger peger i retning af, at han er i stand til at vedligeholde sit lån. Men alligevel sker det ikke. Man kan også forestille sig, at en kundes situation kan ændre sig under lånets løbetid. Fx hvis han

[5]Herefter kaldes dette forhold for "A/R-forholdet".



Figur 4.1: Et eksempel på en model, der kan forbedres



Figur 4.2: Optimal selektionsmodel

mister sit arbejde, hvilket kan gøre ham ude af stand til fortsat at klare lånet.[6] Den menneskelige faktor betyder, at modellen aldrig vil kunne blive perfekt. Men dermed ikke sagt, at den ikke kan forbedres.

Ideen med, at selektionsmekanismen skal fastholde et specifikt forhold mellem de, som accepteres, og de som afvises, medfører, at det helt centrale spørgsmål bliver, hvorvidt det er muligt (billedligt talt) at udskifte de dårlige kunder fra population $\mathcal{A}$ med de gode kunder fra population $\mathcal{R}$. Det åbner for to vigtige spørgsmål:

1. Hvordan skal det specifikke A/R-forhold bestemmes?

2. Hvordan måler jeg, om den nye model er bedre end den gamle?

[6]Denne form kunne give anledning til at differentiere mellem de dårlige kunder, ment på den måde, at ikke alle dårlige kunder er lige dårlige. Nogle betaler måske intet tilbage, mens andre betaler en andel tilbage. Af de dårlige er sidstnævnte selvfølgelig at foretrække. Men da informationen ikke er tilgængelig i datamaterialet fra banken, kan sådan en sondring p.t. ikke finde sted.

Forholdsbestemmelse. Det A/R-forhold, som selektionsmekanismen skal sigte mod, bestemmes på bagrund af en vurdering af, hvor stort et potentiale af gode kunder, der befinder sig i $\mathcal{R}$ gruppen. Men jeg har kun den fulde information om de kunder, banken har accepteret at give et lån. Og denne gruppe er ikke repræsentativ for hele populationen $\mathcal{D}$. Derfor ved jeg ikke, hvordan en ansøger fra $\mathcal{R}$ ville have performet, hvis han blev accepteret. En løsning på dette problem er, at man i en periode giver lån til alle ansøgere, men det er en dyr metode til at indsamle data på og næppe en, man vil forfølge. En anden metode er, hvis man kan købe sig til den manglende information. Jeg anser det dog for at være praktisk umuligt, for det ville kræve, at banker og andre finansielle institutioner udveksler information om deres kunder. Den form, jeg har valgt til vurdering af potentialet af gode kunder i $\mathcal{R}$, er EM-algoritmen. Oplysninger om performance er jo ukendt i denne gruppe, men EM-algoritmen kan netop bruges til at estimere manglende data. Algoritmen skal dog kende til performancen for nogle elementer, før den kan estimere de manglende data. Derfor udføres algoritmen på det fulde datasæt $\mathcal{D} = \{\mathcal{A}, \mathcal{R}\}$, da det er $\mathcal{A}$, som har den nødvendige information. Resultatet bliver, at der er 62% gode kunder i $\mathcal{D}$.

Estimeret andel af gode kunder: 62%

Jeg vil i denne forbindelse kort nævne to ekstremer. Det første er, hvis der findes flest gode kunder i den afviste gruppe. I så fald kan det være en indikation af, at den eksisterende metode til kreditvurdering ikke virker efter hensigten, da den fravælger flere gode ansøgere, end den accepterer. Det andet ekstrem er, hvis der ikke forekommer gode kunder i $\mathcal{R}$, hvilket kan være et udtryk for, at for mange ansøgere bliver godkendt. Ekstremerne vil dog sjældent forekomme, så jeg vil ikke bruge mere tid på dem her.

Den optimale model vil som sagt acceptere samtlige 62% af de gode kunder og ikke tage nogle af de resterende dårlige 38% med. Men da det ikke er realistisk at lave en perfekt model, er det nødvendigt at slække på kravet. Derfor definerer jeg, at modellens målsætning skal være at få de bedste 70% af alle ansøgere. Dette tal er næsten identisk med bankens system, der også accepterer knap 70%. Når jeg vælger et tal højere end 62%, betyder det, at modellen vil godkende nogle dårlige ansøgere. Men man kan, som jeg tidligere har nævnt, argumentere for, at ikke alle dårlige kunder er lige dårlige, selv om man ikke kan se det i vores data. Jeg har desuden en antagelse om, at man skal vælge en procentsats højere end det estimerede for at undgå nåleøje-princippet. Hvis jeg fx valgte 62%, ville andelen af gode kunder i $\mathcal{A}$ gruppen alt andet lige falde over tid. Hvis jeg så efter et par år igen laver tilsvarende analyse, vil EM-algoritmen estimere andelen af gode kunder i den fulde population lavere end første gang. Det skyldes, at estimatet baserer sig på andelen af gode kunder i $\mathcal{A}$ (som nu er mindre).

Model målsætning: 70% af de bedste ansøgere

Det er nu muligt at bestemme det teoretiske minimum for, hvor godt modellen vil kunne performe til 70-62%=8%.

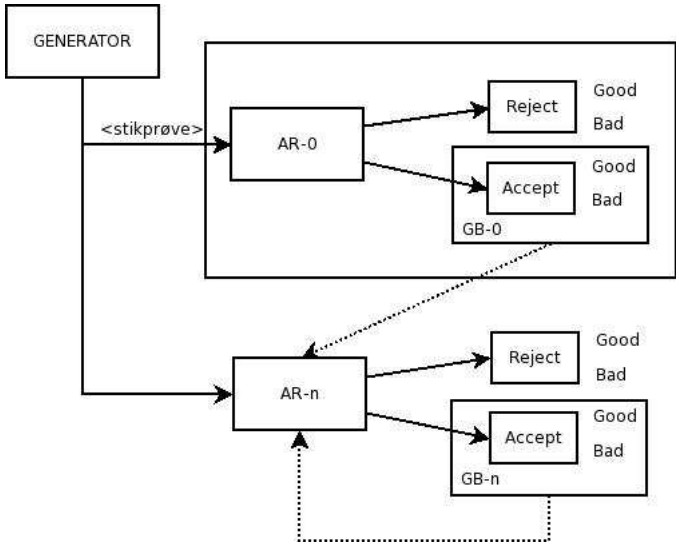
Modelsammenligning. En enkel måde til at måle, om en model er bedre end en anden, er ved at sammenligne deres evne til at klassificere. Her kan man enten vælge at sige, at alle fejlklassifikationer er lige dyre, eller man kan vælge at differentiere på typen af fejl. Sidstnævnte kan fx gøres ved, at prisen for at acceptere en dårlig ansøger er tre gange højere end prisen for at afvise en god. Mit datasæt indeholder ingen information om omkostningerne ved forskellige fejl, og derfor kan jeg ikke differentiere på fejltypen. Mine sammenligninger sker derfor på baggrund af antallet af fejl og ikke typen af fejl. Men før jeg overhovedet kan foretage nogle sammenligninger, må jeg kende performancen for alle ansøgere - også for dem i den afviste gruppe. Men da jeg ikke har den oplysning i datamaterialet, laver jeg et virtuelt eksperiment. Grundlaget for eksperimentet er en case-generator, hvis formål er at kunne danne ansøgere, hvorom jeg kender deres performance. Jeg vil i afsnit 4.6.1 komme nærmere ind på, hvordan generatoren bliver konstrueret.

4.4 Strategien

I dette afsnit vil jeg konkretisere min metode, dvs. gå i detaljer med, hvordan konkrete problemstillinger i opgaven løses.

Min metode til at lave en bedre model omhandler en iterativ proces, hvor selektionsmodellen forbedres ved at udnytte den viden, som ligger i gruppen af accepterede ansøgere. Konkret handler det om, at performancemodellen bliver opgraderet til den nye selektionsmodel. Herefter bliver de genererede data igen kørt gennem klassifikatoren, hvilket danner grundlaget for en ny selektionsmodel. Min forventning er, at ved at gentage denne proces et antal gange, vil man opnå at finde en model, som er bedre til at udvælge de gode ansøgere. Processen er illustreret i figur 4.3. AR-0 er den model, banken anvender i dag, mens AR-n er de nye selektionsmodeller, der konstrueres på baggrund af de accepterede ansøgeres performance.

Denne fremgangsmåde vil føre to problemer med sig. Det første problem drejer sig om prior sandsynligheder for klassifikationen i de nye AR-n modeller. Fordi modellen er bygget på de accepterede ansøgere, vil disse vægte meget højere end de afviste. Løsningen på problemet er indbygget i min metode, da jeg fastholder et specifikt A/R-forhold. Det andet problem er risikoen for, at nogle elementer ikke kan klassificeres. Det opstår,



Figur 4.3: Opbygningen af en ny, og måske bedre, model.

fordi AR-n bygges på et subsæt af alle data. Derfor kan der være elementer, som modellen simpelthen ikke ved, hvordan den skal håndtere, fordi de ikke var at finde i den forrige iteration. Da det er vigtigt, at alle elementer bliver klassificeret, skal der findes en løsning på dette problem. Løsningen er betinget af valget af klassifikator og vil blive behandlet i næste afsnit.

Valg af klassifikator

Næste skridt bliver at bestemme, hvilken type af klassifikator jeg vil anvende. Fordi målet er at få de bedste 70% af alle ansøgere, betyder det, at modellen skal udvælge netop de ansøgere, der har størst sandsynlighed for at performe godt. Modellen skal derfor kunne angive en sandsynlighed for dette, og ikke blot klassificere en ansøger som værende enten god eller dårlig. Derfor er beslutningstræer som fx C5 udelukket, da de giver et enten-eller svar. Man kan ikke (på en fornuftig måde) få disse til at acceptere flere eller færre ansøgere. Det kan til gengæld de bayesianske metoder, som giver en distribution og ikke en enkelt værdi som resultat. Mine resultater fra analysen af selektionsmekanismen viser, at der ikke er den store forskel mellem TAN og NB klassifikatorene (under 2%-point), så derfor vælger jeg den mest simple af dem, som er NB.
For at sikre, at alle elementer kan klassificeres med NB, må ingen sandsynligheder være lig nul. Grunden er, at hvis et element indeholder en værdi for en feature, og denne værdi ikke er kendt i træningssættet, så vil klassifikatoren ikke kunne bestemme det. Gælder

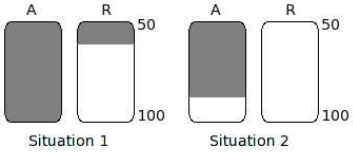
det for samtlige klasser, vil resultatet af produktet vil blive 0 jf. 4.1.

$$p(c|x) \propto p(x|c)p(c) = p(c) \prod_{i=0}^{n} p(x_i|c) \tag{4.1}$$

Der er umiddelbart to forskellige løsninger. Enten ignorerer man de ukendte attributværdier, eller også sættes sandsynligheden meget lavt, fx 10^{-5}. Vælger man sidstnævnte, kan alle elementer klassificeres, omend nogle sandsynligheder kan være meget små. En anden fordel er, at en efterfølgende analyse kan afdække hvilke elementer, der er svære at klassificere - dvs. give viden om, hvor modellen mangler information. Den anden løsning er at ignorere de attributværdier, der ikke er registreret information om. Det gøres ved at sætte sandsynligheden til 1 (dvs. $p(x_i|c) = 1$) for ikke klassificerbare attributværdier. Det medfører selvfølgelig den lidt kunstige situation, at jo *mindre* information, der er til rådighed, jo højere bliver likelihooden.
De to metoder medfører ingen forskel i klassificeringen - differencen er <0,1%.

Hvordan sikres modelmålsætningen?

Hvordan opnår jeg, at modellen får det ønskede 70/30-forhold mellem accepterede og afviste ansøgere? Hvis GB-0 opgraderes, og hvis generatoren ellers virker fornuftigt, så vil AR-1 acceptere ca. 53%, hvilket er for lidt (nåleøje-princippet er sat i kraft). Derfor skal jeg have modellen til at acceptere flere ansøgere, hvilket gøres ved at sætte et lavere niveau for, hvornår en ansøger er god. Denne grænse er for NB som standard sat ved 50% - modellen tager blot den klasse, som den er mest sikker på er den rigtige. Løsningen på problemet er at sætte grænsen så meget lavere, at den nye model stadig vil acceptere ca. 70% af ansøgerne (det kan man teste på de genererede data). Men problemet kan også være det omvendte, nemlig at for mange accepteres. Figur 4.4 viser de to tilstande. Situation 1 beskriver, når der er for få accepterede, mens situation 2 beskriver, når der er for mange accepterede. Tallene angiver sandsynligheden for at tilhøre klassen. Det grå område illustrerer, hvor mange der skal accepteres for at opnå det ønskede forhold.



Figur 4.4: Opretholdelse af ønsket A/R-forhold

En vigtig pointe er, at den nye grænse kun skal gælde for enten at acceptere *eller* for at afvise, men aldrig begge dele samtidigt. Dvs. hvis for få accepteres, skal kun grænsen for at afvise sættes lavere. Omvendt hvis for mange accepteres, er det godkendelsesniveauet, som skal sættes højere. Tabel 4.5 giver et eksempel på en klassifikator, som har accepteret for mange ansøgere. Her ses det, at rækkerne 1201-1266 skal ændres fra en accept til en afvisning for, at A/R forholdet kan blive de ønskede 70%.

n	**p(C)**	**Klasse**	**C**	**F(x)**
1200	0.55	A	A	69,96
1201	*0,57*	*A*	*A→R*	*70,01*
	...	...	*A→R*	...
1266	*0.999*	*A*	*A→R*	*80.52*
1267	0.503	R	R	81,04
1268	0.567	R	R	81,44

Tabel 4.5: Resultatet af en klassifikation, hvor der korrigeres for A/R forholdet

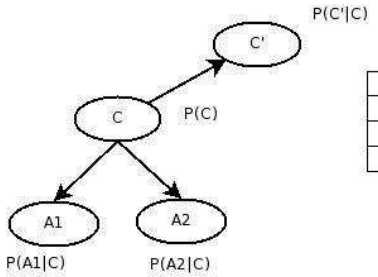
Problemet med at få klassifikatoren til at opretholde et specifikt A/R forhold kan løses på to måder. Resultatet bliver det samme, så valget kommer an på, hvordan det nemmest laves med den software, API eller kode, man har til rådighed. Programmerer man selv sin klassifikator, sætter man blot en ny grænse for, hvornår en ansøger skal afvises. Det skulle være ligetil, da man jo har beregnet distributionen for $\mathcal{AR}$. Tager man udgangspunkt i tabel 4.5, ses det, at den nye grænse for accept bliver 57%.
Alternativt, hvis man ønsker at have en klassifikator, der virker efter princippet om, at det er den mest sikre klasse, som vælges, så kan man tilføje en ny knude til sin NB klassifikator, hvori man definerer den nye grænse. Denne nye knude vil således være "påvirket" af den oprindelige beslutningsvariabel. Hvis man kalder den nye knude $\mathcal{C}'$ og den oprindelige $\mathcal{C}$, så har man, at beslutningen tages på baggrund af: $f(x) = p(\mathcal{C}'|\mathcal{C})$. Jeg vil give et eksempel på, hvordan det helt konkret fungerer, når modellen har accepteret for få: Hvis $\mathcal{C}$ har accepteret, så accepterer $\mathcal{C}'$ selvfølgelig også. Men hvor $\mathcal{C}$ afviser, når værdien er under 50%, vil $\mathcal{C}'$ kun afvise ved en lavere sandsynlighed. Figur 4.5 viser et eksempel på dette. Af figuren ses det, at givet at $\mathcal{C}$ afviser, vil $\mathcal{C}'$ gøre det med 90% sikkerhed. Det betyder, at den reelle acceptprocent bliver 44,44%. Dvs. beslutningsknude $\mathcal{C}$ vil med en acceptprocent på 44,44% medføre, at $\mathcal{C}'$ bliver 50%, altså en accept. Den nye grænse kan beregnes med formel 4.2, hvor $p(\mathcal{C}' = Reject)$ i dette eksempel er 90%.

$$f(x) \quad = 1 - (0,5 \,/\, p(\mathcal{C}' = Reject)) \tag{4.2}$$

Princippet vil være det samme, hvis der er accepteret for mange af modellen. Men her er procenterne faste for $\mathcal{C}$="Reject". Ved brug af denne metode handler det så om at

finde den procentsats, der gør, at modellen får et 70/30 forhold mellem de accepterede og afviste. Og her er der ikke andet at gøre end at prøve sig frem.

Den første metode, jeg beskrev, er at foretrække i de fleste tilfælde, da den i kun to gennemløb af dataene vil resultere i det ønskede A/R-forhold.



C	Accept	Reject
C'		
Accept	1	0.1
Reject	0	0.9

Figur 4.5: Ny beslutningsvariabel

4.5 Eksperiment I

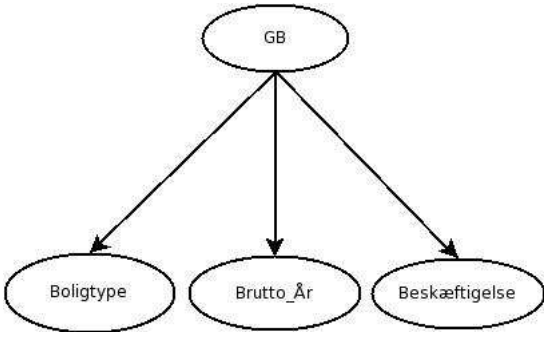
Jeg vil i det følgende lave et lille eksperiment, som vil vise, at det er teoretisk muligt at lave en ny og bedre model efter de principper og ideer, jeg har fremlagt. Formålet med eksperimentet er at vise, at man kan forbedre modellen ved iterativt at lave nye modeller baseret primært på de godkendte ansøgere. Lad os derfor antage, at man har en selektionsmodel, AR-0, som ikke performer nær det optimale. Den kan fx have et A/R-forhold på 30/70, selv om potentialet af gode kunder er langt højere. Det skulle så resultere i, hvis min ide holder, en model, som er bedre end den nuværende.

Selektionsmodel AR-0

Første skridt bliver at skabe et datagrundlag for eksperimentet. Her laver jeg en simpel NB model, som jeg kan generere træningsdata med. Modellen indeholder tre features samt en klassifikation af ansøgerens performance (se figur 4.6). Parametrene til modellen kommer fra bankens datasæt, hvor EM-algoritmen er kørt på det fulde datasæt. De genererede cases renses for dubletter[7], og en NB klassifikator fremstilles, nemlig AR-0.

A/R-forholdet for træningsdataene er 77/23

[7]En dublet er defineret på værdierne i samtlige attributter. Dvs. en feature vektor definerer netop en klasse.



Figur 4.6: Simpel NB klassifikator

Test data

For at kunne teste modellernes evne til at klassificere, og for at kunne sammenligne dem, skal jeg have nogle testdata. Disse genereres også ud fra modellen vist i figur 4.6. Også her har jeg renset for dubletter og ender med 2171 rækker, hvoraf de 1552 er gode.

A/R forholdet testdataene er 71,5/28,5

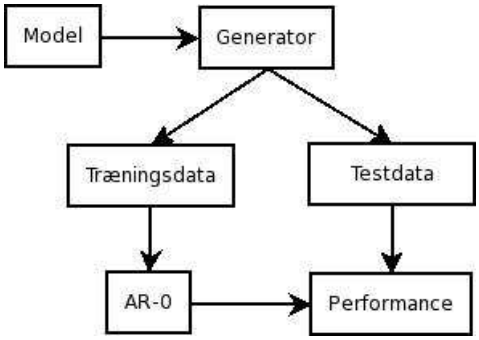
Forskellene på A/R-forholdene mellem trænings- og testdata kan forklares ud fra tilfældigheder, da det er samme model, de er skabt på baggrund af.

Performance for AR-0

Det er nu muligt at måle, hvor effektiv klassifikatoren er. Resultatet er en total fejlrate på 16,44%. Ser man udelukkende på den accepterede gruppe, er fejlraten 14,5%. Nu handler det om, hvorvidt det er muligt, når man har et "forkert" A/R-forhold i den initiale model, at lave en ny og bedre model. Og modellen er bedre, hvis fejlraten for den *accepterede* gruppe er lavere end 14,5%.

Nu er grundlaget for det simple eksperiment dannet, og processen frem til nu er vist i figur 4.7.

Nu mangler jeg kun to trin, før jeg kan udføre eksperimentet. Jeg skal have fastlagt modelmålsætningen samt rette A/R-forholdet i AR-0 modellen - herefter kaldet AR-0'. Målsætningen sætter jeg til at være 75% af de bedste ansøgere, og det "forkerte" A/R-forhold sættes til 30/70.
Med målsætningen på 75% kan jeg beregne det teoretiske minimum for antal fejl i den accepterede gruppe. Modellen vil acceptere ca. $2171 \times 0,75 = 1628$, og minimum er da $(1628 - 1552) \times 100/1628 = 4,67$ %



Figur 4.7: Grundlag for eksperimentet

Min ide kan nu efterprøves efter princippet vist i figur 4.3, og håbet er, at efter et antal iterationer vil jeg finde en model, som performer bedre end AR-0'. Hvis det kan lade sig gøre, er det en indikation af, at min ide holder. Til at udføre denne proces har jeg skrevet nogle programmer, og det er resultaterne fra disse, som jeg viser her. Programmerne er gennemgået i bilag 1.

Algorithm 3 Eksperiment

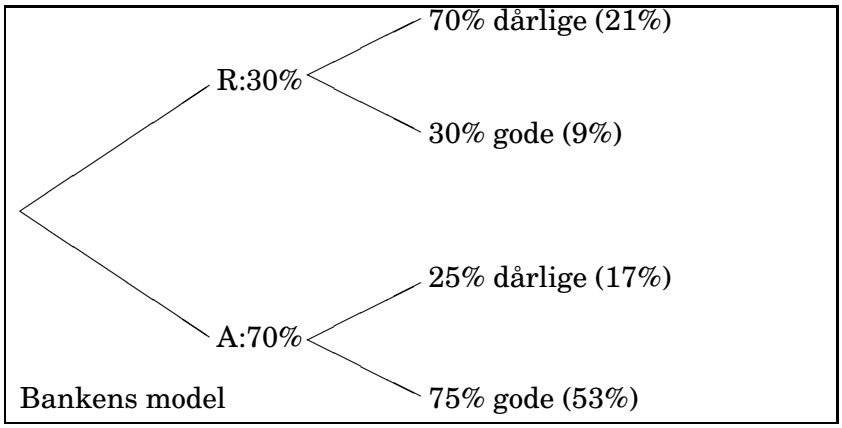
1. Lav AR-0 modellen.
2. Konstruer AR-0' modellen: Ret de marginale sandsynligheder for A/R til noget forkert (30/70).
3. Påbegynd iterationerne.
4. Lav en AR-n model baseret på de accepterede ansøgere.
5. Definer en ny accept eller reject grænse, således det ønskede A/R-forhold opnås. Det giver AR-n' modellen.
6. Mål performancen.
   Er performancen acceptabel, så stop, ellers gå til punkt 4, hvor n=n+1.

Resultaterne fremgår af tabel 4.6, hvor "Accept" er grænsen for, hvornår der accepteres, for at det ønskede A/R-forhold opnås. "Fejlrate total" er, hvor mange fejl den givne model laver, mens "A Fejlrate" kun måler fejlraten i den accepterede gruppe.

Det fremgår af tabellen, at det lykkes at finde frem til en ny og bedre model, og eksperimentet peger således i retning af, at metoden til modelforbedring holder.
AR-0' modellen er i dette eksempel en ubrugelig model, da den accepterer alle ansøgere. Derfor er det ikke noget problem at skabe en model, som er bedre end denne. Men det lykkes også at finde en model, fx AR-1', som er bedre end AR-0, så konklusionen er stadig den samme: At metoden antageligt virker.

	Accept (%)	**Fejl total (%)**	**A Fejlrate (%)**
AR-0	50	16,44	14,5
AR-0'	50	28,51	28,51
AR-1	50	13,59	9,99
AR-1'	46,5	*12,11*	*10,24*
AR-2'	73,7	17,83	14,68
AR-3'	*69,8*	*17,6*	*14,42*
AR-22'	*74,4*	*12,53*	*10,74*
AR-24'	*74,6*	*12,53*	*10,74*

Tabel 4.6: Resultater



Figur 4.8: Bankens selektionsmodel

4.6 Eksperiment II

Næste skridt er at forsøge metoden på et mere realistisk datagrundlag. Men inden jeg kommer dertil, er det relevant at undersøge, hvor meget der kan vindes, hvis en bedre model kan konstrueres. Bankens system (som man ikke kender) godkender ca. 70% af alle ansøgere, og af disse viser det sig, at ca. 75% er gode kunder. Dvs. bankens model får 70*75%=53% af de mulige 62% gode ansøgere. Det betyder også, at der ligger et potentiale gemt på ca. 9% gode ansøgere, og det er disse, man skal forsøge at få fat i. Af figur 4.8 fremgår det, hvorledes bankens model fordeler ansøgerne. Tallene i parentes er procent af totalen.

Fremgangsmåden er den samme, som jeg brugte i det første eksperiment, men der er nogle forskelle. Eksempelvis var både modeller og datagrundlag meget forsimplede i forhold til den virkelige verden. Det interessante bliver derfor at se, om metoden også holder, når der arbejdes på et mere realistisk fundament. Jeg vil kort opsummere grundlaget for eksperimentet (se også afsnit 4.3):

- Målsætningen for modellen er et A/R-forhold på 70/30.
- Klassifikatoren er NB.
- Potentialet af gode ansøgere er 9%.
- Fejlrate kan ikke blive lavere end 8%.

Udgangspunktet for testen er den selektionsmodel, jeg konstruerede på baggrund af bankens data, og det er gennem denne model, jeg vil lade mine kunstigt skabte ansøgere blive klassificeret første gang. Det svarer til, hvor banken er i dag, dog med den ekstra viden, at jeg kender performancen for ansøgerne. Det er som sagt den viden, jeg vil anvende til at forbedre selektionen. Algoritmen ligner den, jeg brugte under første eksperiment.

Algorithm 4 Eksperiment II

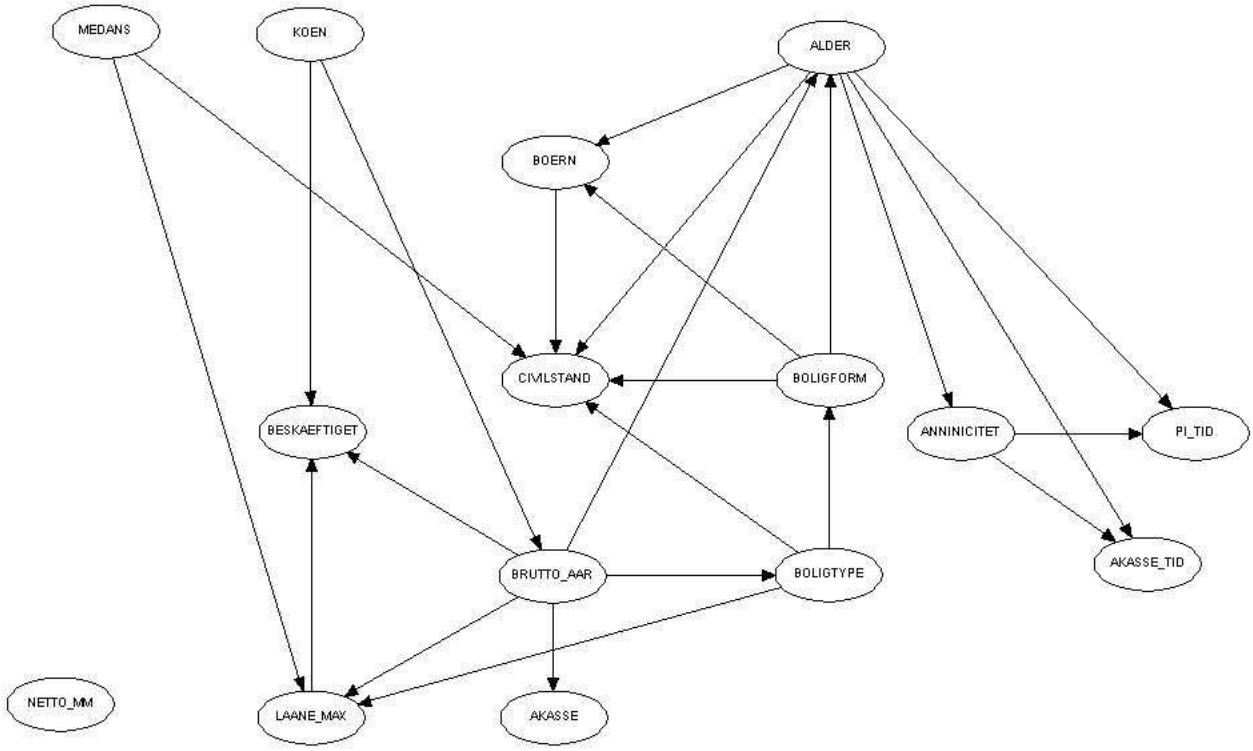
1. Skab en mængde lånansøgere.
2. Lad dem klassificere af bankens selektionsmodel (AR-0).
3. Påbegynd iterationen.
4. Lav en AR-n model baseret på de accepterede ansøgere.
5. Definer en ny accept eller reject grænse, således det ønskede A/R-forhold opnås. Det giver AR-n' modellen.
6. Mål performancen.
   Er performancen acceptabel, så stop, ellers gå til punkt 4, hvor n=n+1.

Et af de centrale punkter for dette eksperiment er case-generatoren. Hvordan skaber jeg en mængde kunstige ansøgere med samme kendetegn, som rigtige ville have haft? Det vil jeg undersøge i det næste afsnit.

4.6.1 Generator

Formålet med generatoren er at kunne skabe ansøgere, som jeg kender performancen for. Derved kan jeg nemt sammenligne modellernes evne til at klassificere. Men for at kunne lave en generator, må jeg vide noget om, hvordan en ansøger kan se ud. Til det anvender jeg den information, jeg har fra banken - en database med lånansøgere. Ideen er nu at lave en model, som beskriver disse, og anvende den til at generere cases med. Metoden, jeg bruger for at finde en model, er at finde sammenhængene mellem variablene. Det kan fx gøres med en NB eller en TAN model. Men da det ikke er en klassifikation,

jeg skal lave, mener jeg ikke, at disse typer er det bedste valg. Derfor vælger jeg at bruge et bayesiansk net i stedet, eftersom dette kan beskrive flere aspekter, dvs. kausale sammenhænge, end de andre to typer. Til at lære strukturen af et bayesiansk net kan man fx anvende PC-algoritmen.[8] Kort fortalt finder man betingede uafhængigheder i datasættet og d-separerer mellem disse. Startende med en komplet graf, stiller man spørgsmålet, om der er uafhængighed mellem to variable givet et sæt af andre variable. Hvis der er, fjernes forbindelsen mellem dem. Software-programmet Hugin gør fx dette. Mit første forsøg på at finde en model gav et resultat, der mest mindede om et spindelvæv, hvor samtlige variable var forbundet på kryds og tværs. Med andre ord en for kompleks og ubrugelig model. Derfor skulle jeg have reduceret antallet af forbindelser, hvilket jeg gjorde i to trin. For det første medtog jeg kun de stærkeste forbindelser, og for det andet slettede jeg forbindelser, som jeg vurderede var forkerte. Det gav et mere brugbart resultat, som er vist i figur 4.9.



Figur 4.9: Bayesiansk model for kreditkortansøgere

Nu er strukturen lært, og det næste er at bestemme modellens parametre. Her anvender jeg EM-algoritmen, hvor jeg bruger det fulde datasæt $\mathcal{D}$ til input. Det gør jeg, fordi jeg ønsker at fremstille ansøgere, som ligner de, der har ansøgt om lån. Derfor kan jeg ikke

[8]Path Consistency algorithm.

nøjes med kun at kigge på $\mathcal{A}$, da feature distributionen for $\mathcal{A}$ med stor sandsynlighed vil se anderledes ud end den for de afviste ($\mathcal{R}$). Fx kunne man sikkert antage, at indkomstforhold ser bedre ud for dem i population $\mathcal{A}$ end for dem i population $\mathcal{R}$. En anden vigtig pointe er distributionen af gode og dårlige kunder i de to populationer. Hvis man kun ser på $\mathcal{A}$, gælder det, at andelen af gode og dårlige er hhv. 75% og 25%. Men dette tal er ikke repræsentativt for $\mathcal{D}$ (under antagelse om, at bankens model til sortering af ansøgere virker blot nogenlunde, dvs. selektions-mekanismen *ikke* er tilfældig). Man kan således ikke forvente, at der også vil ligge 75% gode kunder skjult i den afviste gruppe. Derfor vil EM algoritmen estimere, at distributionen af gode kunder vil være noget lavere for $\mathcal{D}$ end for $\mathcal{A}$. (Andelen beregner jeg til 62% i afsnit 4.3).

Nu har jeg information nok til at kunne generere cases. Med Hugins hjælp laver jeg 300.000 ansøgere, som jeg skal have klassificeret. Mine analyser har vist, at et boostet C5 træ er signifikant bedre til at klassificere sammenlignet med de bayesianske metoder TAN og NB. Derfor vælger jeg C5 til at klassificere de genererede data med. Resultatet af klassificeringen bliver, at ca. 88% er gode ansøgere i datasættet. Jeg havde forventet at få en fordeling tæt på de 62/38, som jeg havde estimeret ud fra de oprindelige data. Når det ikke sker, kan problemet skyldes, at det bayesianske net ikke præcist nok får beskrevet, hvordan ansøgerne ser ud. Den umiddelbare løsning, jeg har valgt, er at fjerne - tilfældigt - så mange af de gode ansøgere, at jeg opnår den rette fordeling.

Det sidste, jeg gør, er at tilføje, hvad man kan kalde den menneskelige faktor til datasættet. Det er tilfælde, som man ikke kan forklare ud fra modellen - ansøgere som handler stik modsat af det forventede. Måden at gøre det på er ved at tilføje støj til de genererede data. Det har jeg valgt at gøre i 5% af alle tilfælde, hvor jeg ændrer klassificeringen til at være det modsatte.

Det endelige resultat af de genererede data ser ud som vist i tabel 4.7. Dvs. jeg har 117.199 ansøgere, hvor jeg kender deres performance.

Klassificering	**Antal**	**Procent**
G	79.549	67,88%
B	37.650	32,12%
Total	117.199	100%

Tabel 4.7: De generede data

4.6.2 Resultater

Som det fremgår af tabel 4.8, lykkes det at finde en ny og bedre model efter blot 1 iteration (AR-1'). Eksperimentet peger således i retning af, at metoden til modelforbedring

holder.
Første række af tabellen viser, hvordan banken faktisk performer - tallene er fundet ved at analysere det originale datamateriale. Det er således *ikke* performancen ved brug af mine modeller og kunstige data. Anden række derimod er resultatet af de kunstige data, når de bliver klassificeret af bankens selektionsmodel. Og her er der en vigtig pointe at bemærke: Hvis generatoren skaber lånansøgere, som ligner bankens lånansøgere, så skulle de to resultater minde om hinanden. Det gør de ikke. Faktisk er resultatet af klassificeringen på de kunstige data ikke meget bedre end ved tilfældig udvælgelse (når man ser på det totale antal fejl).

	Accept (%)	**Fejl total (%)**	**A Fejlrate (%)**
Banken	N/A	26	17
AR-0	50	39,44	26,95
AR-1'	52,3	27,17	20,95
AR-2'	66,9	30,34	23,25
AR-3'	64,9	28,78	22,08
AR-10'	68,1	32,22	24,57
AR-11'	66,2	29,14	22,36
AR-12'	67,8	31,74	24,22
AR-13'	62,3	29,47	22,6

Tabel 4.8: Resultater

Konklusionen er, at generatoren ikke er præcis nok, da de ansøgere, den får skabt, er for afvigende sammenlignet med de, som banken kender til. Det betyder, at man ikke umiddelbart kan sammenligne bankens model med den, jeg får skabt. Men det, man kan sige noget om, er, hvordan metoden virker på de kunstige data. Dvs. under antagelse om, at generatoren skaber lånansøgere, som ligner rigtige ansøgere, så viser resultatet, at der er noget at vinde ved at lave en ny model baseret på de principper, jeg har fremlagt i dette speciale. Startende med en fejlrate i den accepterede gruppe på 26,95%, så lykkes det at finde en model med en fejlrate i samme gruppe på 20,95% - en forbedring på 22,3%. Ser man på det totale antal fejl, er forbedringen mere markant, hvor jeg går fra 39,44% til 27,17%, dvs. en forbedring på 31,1%.

	AR-0	**AR-1'**	**Forbedring**
A Fejlrate	26,95	20,95	22,26
Fejl total	39,44	27,17	31,11

Tabel 4.9: Resultater

En anke kan dog være, at fordi den initiale model er dårlig, så skal der ikke meget til at

forbedre den. Jeg har derfor også forsøgt med en TAN model som basis for generatoren. Klassificeringen af performancen sker stadig gennem et boosted C5 træ. Men resultatet var ikke bedre end det, hvor et bayesiansk net blev anvendt, se også tabel 4.10.

	AR-0	**AR-1'**	**Forbedring**
A Fejlrate	36,86	28,59	22,44
Fejl total	43,81	30,65	30,04

Tabel 4.10: Resultater

Sidste forsøg, jeg har gjort, er at modificere det bayesianske net, som jeg anvendte i generatoren under første forsøg (se figur 4.9). Jeg har tilføjet klassifikationsvariablen til modellen og ladet den pege på samtlige features - præcist som i NB. Desuden har jeg tildelt klassifikationsvariablen en lav prior sandsynlighed for accept (62/38). Med dette håber jeg, at model-parametrene vil blive mere realistiske.
Men A/R-forholdet, efter det boosted C5 træ har klassificeret elementerne, er stadig for højt: 88/12, så derfor fjernede jeg de "overskydende" gode. Men som det fremgår af tabel 4.11 så løser det ikke problemet, og konklusionen er, at heller ikke den model er bedre end den første.

	AR-0	**AR-1'**	**Forbedring**
A Fejlrate	32,76	28,09	14,3
Fejl total	40,49	26,2	35,3

Tabel 4.11: Resultater

Kapitel 5

Konklusion

Hovedformålet med denne opgave var at udvikle en model, der i højere grad øgede sandsynligheden for, at banken i fremtiden ville få en større andel af de ansøgere, der opfylder kriterierne for god betalingsevne og -vilje. Til det formål udviklede og afprøvede jeg en iterativ metode til at finde bedre modeller til kreditvurdering. Tanken var, at selektionsmodellerne blev bygget på baggrund af viden om de ansøgere, som var accepteret, og som havde performet godt. Min antagelse var, at modelforbedringen kunne skabes ved at udnytte viden fra denne gruppe af accepterede. Jeg viste i den forbindelse, at modeller bygget udelukkende på baggrund af de accepterede gode kunder ville medføre, at nåleøje-princippet trådte i kraft. Dvs. at færre og færre ansøgere med tiden ville blive accepteret af modellen. Derfor foreslog jeg, at man fastholdt et specifikt A/R-forhold. Det betyder, at modellen (oftest) skal tvinges til at acceptere nogle ansøgere, den ellers ville have afvist. For at undgå, at man valgte de dårligste ansøgere, konstruerede jeg metoden således, at den altid ville tage de bedste af alle ansøgere. Det betød, at jeg måtte vælge en model til kreditvurdering, som kunne give et mål for, hvor sandsynligt vurderingen var. Valget faldt på Naiv Bayes, hvor jeg muliggjorde, at accept-procenten kunne flyttes, for at opnå det rette A/R-forhold.

Med henblik på at besvare det første spørgsmål i problemformuleringen, om min metode var bedre til kreditvurdering end bankens, måtte jeg sammenligne modellernes evne til at klassificere korrekt. For at kunne dette, skulle jeg for det første skabe en case-generator for at få ansøgere, hvorom jeg kendte deres performance. For det andet måtte jeg finde en model for den selektionsmekanisme, som banken anvender. Til det formål undersøgte jeg forskellige datamining-metoder og fandt, at C5-træer var de bedste til at foretage klassificeringer. Lidt overraskende fandt jeg også, at de bayesianske metoder, jeg undersøgte, ikke performede godt på det foreliggende datamateriale. Selve testen blev udført som et virtuelt eksperiment, hvor jeg lod de kunstigt skabte data blive klassificeret af både bankens model, og den model min metode fandt frem til. Resulta-

tet pegede i retning af, at der var noget at vinde ved at bruge den iterative metode, jeg foreslog. Det er dog ikke en entydig konklusion, for bankens model performer så dårligt på de kunstige data, at der ikke skal meget til for at forbedre den. Problemet ligger i case-generatoren, som ikke præcist nok får skabt ansøgere, der ligner de, som banken har.

Det andet spørgsmål jeg stillede i problemformuleringen, var at undersøge i hvilken form "reject inference" kunne bidrage, når der var manglende data. Problematikken var, at banken kun har en faktuel viden om betalingsevne og -vilje for de ansøgere, den har accepteret, men den mangler information for gruppen af afviste ansøgere. Den manglende viden medfører et problem med den statistiske konklusion; for hvordan skal man vurdere de afviste ansøgeres performance? Jeg argumenterede for, at problemet var at sidestille med et manglende data problem. Det betød, at jeg skulle identificere den mekanisme, som medførte de manglende data. Min antagelse var, at mekanismen, dvs. kriterierne bag selektionen, var objektive, ensartede og bestemt ud fra oplysninger om ansøgeren. Derfor sluttede jeg, at den manglende datamekanisme var MAR. I forbindelse hermed valgte jeg den bayesianske skoles fortolkning af sandsynligheder. Det gjorde jeg, fordi jeg havde en stærk antagelse om, at der lå viden gemt i den afviste gruppe, som ikke fandtes i den godkendte. På den baggrund kunne jeg ikke ignorere den afviste gruppe og inddrog den i analysen. Herunder bestemte jeg andelen af gode ansøgere i den fulde population til at være 62% - et tal noget lavere end de 75%, som den frekventistiske opfattelse ville have givet.

Kapitel 6

Diskussion

Jeg vil i diskussionen komme omkring de områder, som jeg mener har influeret på undersøgelsens resultat og konklusion.

Fokus i opgaven var at undersøge, hvorvidt det var muligt at udskifte nogle af de gode kunder fra den afviste gruppe med de dårlige kunder i den accepterede gruppe. Resultaterne fra den iterative metode, jeg præsenterede, indikerede, at det kunne lade sig gøre at skabe en bedre kreditvurderingsmodel. Men der var ikke noget entydigt svar, hvilket kan skyldes flere faktorer.
For det første var generatoren, jeg anvendte i de virtuelle eksperimenter, ikke præcis nok, og man skulle derfor overveje, om det ikke var muligt at skabe en generator, der er bedre til at lave ansøgere. Og her tyder meget på, at de bayesianske modeller, jeg undersøgte, ikke er gode nok på det foreliggende datamateriale. Et alternativ kunne være at skabe data på baggrund af et C5-træ. Men da der ikke er sandsynligheder tilknyttet disse træer, er det ikke muligt at skabe den rette distribution af data. En løsning kunne dog være at tilføje sandsynligheder til hvert split i træet og anvende disse til at generere en vægtet distribution.
For det andet var evnen til at klassificere ikke blevet så meget bedre, at metoden reelt ville kunne konkurrere med et boosted C5-træ. En ukendt faktor i denne sammenhæng er dog boosting, som jeg ikke havde tid til at undersøge. Det ville derfor være interessant at se nærmere på, hvor meget boosting ville betyde, hvis det blev anvendt på de bayesianske modeller. Men også andre metoder, som kan give et sandsynlighedsmål for sikkerheden på klassifikationen, kan overvejes.

NB's afhængighedsantagelse kan måske forklare, hvorfor denne klassifikationstype ikke klarer sig godt sammenlignet med C5-træerne. I kapitel 4 fandt jeg, da jeg skulle lære strukturen på en model over kreditkortansøgere, at der var mange afhængigheder mellem attributterne. Men det står i kontrast til NB's antagelse om uafhængighed. TAN

modellerne forsøger at rette op på NB begrænsning, og de performer da også i mit tilfælde lidt bedre, omend det stadig ikke er godt nok. Derfor tror jeg, at et bedre valg til klassificering, er modeller, der ligesom C5 har en træstruktur. Alternativt kan man vælge at se populationen af kreditkortsansøgere som værende sammensat af forskellige grupper.[1] Hver gruppe indeholder ansøgere med "ens" kendetegn, som kan analyseres forskelligt. Derved opnår man, alt andet lige, at variansen på vores estimatorer (modelparametre), bliver reduceret. Det er principielt det, som beslutningstræerne gør, hvor et split kan sammenlignes med et *stratum*.

Det væsentligste grundlag for denne undersøgelse har været datamaterialet leveret fra en bank. I den forbindelse har det vist sig utrolig vigtigt at have et kendskab til det domæne, som man analyserer. Det ideelle ville selvfølgelig have været, hvis analytikeren selv havde denne viden. Men det er ofte ikke realistisk, og derfor vil adgang til en domæneekspert ofte være nødvendig. Domæneeskperten kan bidrage med forretningsforståelse, dataforståelse og i nogen grad validering af modellerne. Uden denne viden må man på bedste vis lave nogle antagelser i tvivlsspørgsmål, men med stor risiko for, at de er forkerte. Eksempelvis antog jeg, at det manglende data-problem var MAR, uden at kende bankens politik for selektion af kreditkortsansøgere. Et andet eksempel var, da jeg foretog datarensning og -transformation, hvor jeg ligeledes måtte lave antagelser om attributters betydning, indhold og funktionelle sammenhænge til andre attributter. Det sidste, meget tydelige eksempel på, at adgang til domæneviden er vigtig, var, da jeg lavede datamining på bankens selektionsmekanisme. Her fandt jeg en NB baseret klassificeringsmodel med en fejlrate på knap 12%. Denne kunne jeg principielt have stillet mig tilfreds med. Men grunden til, at modellen performede så (relativt) godt var, at svaret lå indkodet i RKI_HIT attributten. En viden som en domæneekspert med det samme ville have kunnet give.
Fejlen med RKI_HIT attributten blev fundet, fordi jeg brugte forskellige metoder til datamining. Dette er en vigtig pointe. Målet med datamining er som tidligere nævnt at finde implicit, tidligere ukendt og potentielt brugbar information fra data. Dvs. man på forhånd ikke kan angive en specifik metode til at være den bedste, og derfor er det vigtigt at prøve forskellige metoder. De har alle fordele og ulemper og kan grave forskelig information ud af de data, som analyseres.

[1] Inden for statistik kaldes disse grupper for *strata.*

Litteraturliste

Aczel, A. D. 1999: “Complete Business Statistics”.
Singapore: Irwin/McGraw-Hill.

Chow, C. et al. 1968: “Approximating Discrete Probability Distributions with Dependence Trees”.
IEEE Transactions on Information Theory, Vol.14, P. 462-467.

Dellaert, F. 2002: “The Expectation Maximization Algorithm”.
Georgia: Georgia Institute of Technology.

Dunham, M. H. 2003: “Data Mining - Introductory and Advanced Topics”.
Prentice Hall.

Feelders A. J. 2000: “Credit Scoring and Reject Inference With Mixture Models”.
International Journal of Intelligent Systems in Accounting, Finance & Management.

Feelders A. J. 2003: “An Overview of model Based Reject Inference for Credit Scoring”.
Utrecht: Utrecht University - Institute for Information and Computing Sciences.

Frawley, W. et al. 1992: “Knowledge Discovery in Databases: An Overview”.
AI Magazine

Friedman, N. et al. 1997: “Bayesian Network Classifiers”.
Machine Learning, 29:131-163.

Gongyue, C. G. et al. 2003: “Bound and Collapse Bayesian Reject Inference When data are MNAR”.
University of Waterloo & University of Toronto.

Hand, D. et al. 2001: “Principles of Data Mining”.
London: The MIT Press.

Jensen, F. V. 1996: “An Introduction to Bayesian Networks”.
London: UCL Press.

Jensen, F. V. & Nielsen, T. 2005: “Bayesian Networks and Decision Graphs II”.
Aalborg: Udkast til endnu ikke udgivet bog. (Venligst udlånt af F. V. Jensen).

Kantardzic, M. 2003: “Data Mining: Concepts, Models, Methods, and Algorithms”.
IEEE: John Wiley & Sons.

Little, R.J.A. et al. 1987: “Statistical Analysis with Missing Data”.
New York: John Wiley & Sons, Inc.

Mester, L. J. 1997: “What’s the Point of Credit Scoring?”.
Philadelphia: Federal Reserve Bank of Philadelphia’s Business Review.

Miquélez, T. et al. 2004: “Evolutionary computation based on Bayesian classifiers”.
Int. J. Appl. math. Comput. Sci., Vol. 14, No. 3, p.335-249.

Schapire, R. E. 2001: “The Boosting Approach to Machine Learning - An Overview”.
New York: AT&T Labs - Research.

Smith, A. et al. 2004: “A Bayesian Network Framework for reject Inference”.
San Diego: University of California.

Tan, P. et al. 2006: “Introduction to Data Mining”.
Addison-Wesley.

Bilag

Bilag 1
NBC softwaren

Systemkrav

Programmerne er skrevet i Java og kan således afvikles på hvilken som helst platform, der har en Java virtuel maskine installeret (version 1.4 eller 5.0).

Installation

CD'en som er vedlagt projektet som bilag 3 indeholder en zipfil med programkode, programfiler og datafiler. Zipfilen skal blot udpakkes på din harddisk, hvor du måtte ønske det, fx. c:\ (windows) eller ~/ (unix).

Filstruktur

NBC/
... XXXX.jar
... classify.sh
... classify.bat
... purify.sh
... purify.bat
... genmodel.sh
... genmodel.bat
... config.cfg

NBC/models
... bad.nbc

NBC/models/iterations

NBC/data
... train.dat
... test.dat

NBC/results

Anvendelse

genmodel [datafil] [modelfil]

Dette program skaber en NB klassifikator.

- *datafil* er træningsdataene. Det skal være en kommasepareret fil, hvor sidste kolonne indeholder klassen.

- *modelfil* er det NB netværk, som genereres på basis af *datafil.* Filen bliver som standard lagt i biblioteket NBC/models og kan læses med en teksteditor.

classify [modelfil] [datafil]

Dette program måler klassifikatorens performance.

- *modelfil* er det naive bayesiankse netværk, som skal testes.

- *datafil* er testdataene. Det skal være en kommasepareret fil, hvor sidste kolonne indeholder klassen.

Resultatet bliver skrevet til standard output (skærmen) og kan se således ud:

> Errors: 357/2171 = 16.44%
> AErrors: 244/1683 = 14.5%

Hvor Errors betegner det totale antal fejl i testsættet, og AErrors betegner antallet af fejl i den accepterede gruppe alene.

purify

Dette program udfører forsøget på den iterative modelforbedring, som jeg har beskrevet i dette speciale. Programmet konfigureres i filen config.cfg.

Resultatet bliver skrevet til standard output (skærmen) og kan se ud som i figur 6.1.

Errors betegner det totale antal fejl, mens AErrors betegner antallet af fejl i den accepterede gruppe alene. Linje 5 fortæller, at AR1-modellen accepterede for mange jf. den definerede værdi i config.cfg. Den fortæller også, hvad den nye acceptgrænse bliver sat til, for at det specificerede A/R-forhold opnås.

Modellerne, som skabes, kan findes i NBC/models/iterations.

```
1.      AR0:    Errors: 590/2171 = 27.18%
2.              AErrors: 316/1594 = 19.82%
3.      AR1:    Errors: 251/2171 = 11.56%
4.              AErrors: 215/1731 = 12.42%
5.      AR1-1:  Too many accepted - New accept limit:0.714      Errors: 257/2171 = 11.84%
6.                                                              AErrors: 142/1579 = 8.99%
```

Figur 6.1: Resultat fra kørsel af purify programmet

Konfiguration

I config.cfg filen kan man bl.a. sætte nogle parametre for, hvordan programmerne skal afvikles. Disse vil jeg gennemgå i det følgende.

Parametre gældende for alle programmerne:

ACCEPT_RATIO: Hvor stor en andel, der skal accepteres.
NA_VALUE: Hvilken værdi skal *classify* bruge, hvis et element ikke kendes. Værdierne skal være mellem 0 og 1.

Parametre som kun *purify* anvender:

HEADER: Attributnavnene på datafilen.
INITIAL_MODEL: Modellen som processen begynder med (den som skal forbedres).
SAMPLE_DATA: Input til *purify*.
ITERATIONS: Antallet af gennemløb.

Eksempel

Lav en model:

Fra en kommandoprompt gå til biblioteket NBC og skriv flg.:

> ./genmodel.sh data/train.dat models/myModel.nbc

Resultatet er en NB model, som bliver gemt i NBC/models/myTest.nbc

Test modellen:

> ./classify.sh models/myTest.nbc data/test.dat

Resultatet skrives til skærmen

Kør iterationerne:

> ./purify.sh

Iterationerne køres 10 gange, og resultatet skrives til skærmen. Modellerne bliver gemt som NBC/models/iterations/ARn.nbc.

Bilag 2
Databeskrivelse & rensning

Dette bilag vil kort gennemgå de forskellige attributter i bankens datasæt samt en nærmere beskrivelse af attributter, transformationsregler samt hvorvidt attributten blev medtaget eller ej. Det gælder at alle tidsperioder er i måneder, fx alder og anciennitet.

BØRN

Hvor mange børn har ansøgeren?

Range [0-3]

Datakvalitet:

Blanke: 7442 eller 35,4%

Kategorisering

0 = A
1 = B
2 = C
3 = D

BOLIGFORM

Hvor bor ansøgeren?

Set {"",C,F,H,K,L}

C=C/O adresse, F=Hos Forældre, H=Hus, K=Kollegie, L=Lejlighed

Blanke: 1036 eller 5,06%

BOLIGTYPE

Hvordan bor ansøgeren?

Set {"", A, E, L}

A=Andelslejlighed, E=Ejer bolig, L=Leje

Datakvalitet:

Blanke: 1773 eller 8,66%

Jeg antager, at der er en vis sammenhæng mellem "BOLIGFORM" og "BOLIGTYPE". Flg. sammenhæng antages:

1. Boligform = "Hos Forældre" $\Rightarrow$ Boligtype = {"", L}

Ad.1: Blank skyldes sikkert, at man enten ikke har svaret, eller at man bor gratis. Er der angivet anden boligtype end blank eller "L", skyldes det sikkert, at ansøgeren har svaret på, hvordan forældrene bor. I disse tilfælde konverteres svaret til blank:
12 stk. A

71 stk. E

Jeg har også overvejet flg. regel: Boligform = "Kollegie" $\Rightarrow$ Boligtype = {"", L}.
Men det ville nok være for stram en antagelse. En person kunne fx eje en lejlighed i en by, men studere og bo på kollegium i en anden.

EGEN TELEFON

Har ansøgeren telefon?

Set {"", J, N}

Datakvalitet:

Blanke: 10497 eller 51,29%

Attributten er *fravalgt*.

FLYTTET N6M

Er ansøgeren flyttet inden for de sidste 6 måneder?

Set {"", J, N}

Datakvalitet:

Blanke: 9838 eller 48,07%

Attributten er *fravalgt*.

BILLED ID

Hvilken form for billede-legimitation har ansøgeren fremvist?

Set {"", D, I, K P, X, Z}
""= Ingen information
D= Dankort
I= ID Kort
K= Kørekort
P= Pas
X= Ikke vist
Z= Ikke vist

Datakvalitet:

Blanke: 123 eller 0,6%

De blanke, X og Z er sammen type.

Overvejelse: Kan attributten overhovedet bruges - fortæller fremvist legimitation noget om ansøgeren?

Attributten er *fravalgt*.

ADRESSE ID

Hvilken form for adresse-id har ansøgeren fremvist?

Set {"", B, L, S, X, Z}
""= Ingen information
B= Bopælsattest
L= lejekontrakt
S= Sygesikring
X= Ikke vist
Z= Ikke vist

Datakvalitet:

Blanke: 85 eller 0,42%

De blanke, X og Z er sammen type.

Overvejelse: Kan attributten overhovedet bruges - fortæller fremvist legimitation noget om ansøgeren?

Attributten er *fravalgt*.

DOB

I datasættet fremgår fødseldatoerne, men det er ansøgerens alder og ikke fødselsdag, der giver mest information. Derfor ignoreres DOB attributten.

Attributten er *fravalgt*.

BESKÆFTIGELSE

Hvilken beskæftigelse har ansøgeren?

Set {"", F, H, I, L, P, S, T, U}

F=Funktionær, I=Ikke funktionær, T=Tjenestemand, S=Selvstændig, L=Ledig, P=Pensionist, H=Hjemmegående, U=Under uddannelse

Datakvalitet:

Blanke: 384 eller 1,88%

ANCIENNITET

I hvor mange måneder har ansøgeren været i nuværende stilling?
Resultatet kommer fra data-sættets 2 kolonner: TID_JOB_AA og TID_JOB_MM (hhv. antal år og måneder på nuværende job).

Range [0-1044]

Datakvalitet:

Blanke total: 5788 eller 28,28%

Blanke, hvor BESKÆFTIGELSE er "F ∣ I ∣ T ∣ S" = 689+197+464+342 = 1692 eller 8,27%.

Jeg antager, at der er en vis sammenhæng mellem "BESKÆFTIGELSE" og "ANCIENNITET". Flg. sammenhæng antages:

1. Beskæftigelse = "F ∣ I ∣ T ∣ S" $\Rightarrow$ Anciennitet = {"", m}
2. Beskæftigelse = "L ∣ P ∣ H" $\Rightarrow$ Anciennitet = {"", 0}

Man kunne diskutere, hvorvidt det giver mening, at en person under uddannelse har nogen anciennitet. Jeg antager dog, at i sådan tilfælde, betyder det, at ansøgeren har et studiejob. Dvs. ancienniteten afspejler, hvor lang tid vedkommende har haft dette job.

Ad. 1: Reglen er overholdt, så der sker ingen konvertering.

Ad. 2: Hvis reglen ikke er overholdt, konverteres det forkerte svar til blank.

0 stk. L

0 stk. P

0 stk. H

Der må også være flg. sammenhæng mellem ANCIENNITET og ALDER:

1. ALDER + *y* > ANCIENNITET

Det er svært at sætte en præcis værdi for *y*. Er y på 6 år, 10 år eller måske 14 år acceptable, og hvor afhængig skal *y* være af beskæftigelse? Fx bliver man næppe funktionær i en alder af 12 år, men derfor kunne man godt være i arbejde allerede i den alder.

Ad. 1: ALDER + *Y* skal være højere end 11 år, ellers konverteres ANCIENNITET til blank.

5 stk. ANCIENNITET

Kategorisering

Fl.g. grupper er konstrueret:
Blanke: 5830
[0-24 [= A
[24-48 [= B
[48-72 [= C
[72-96 [= D
[96 -120 [= E
[120-] = F

AKASSE

Er ansøgeren medlem af en a-kasse?

Set {"", J, N}

Datakvalitet:

Blanke: 7899 eller 38,6%

AKASSE_TID

I hvor mange måneder har ansøgeren været medlem af en a-kasse?

Resultatet kommer fra datasættets 2 kolonner: TID_AKA_AA og TID_AKA_MM (hhv. antal år og måneder).

Range [0-600]

Datakvalitet:

Blanke: 13069 eller 63,86%

Blanke hvor AKASSE =”J” = 3163 (Måske betegner disse, at man har været medlem af en a-kasse, men ikke længere eer det. Der foretages ingen datarensning i dette tilfælde).

Jeg antager samme sammenhæng, som under ANCIENNITET:

1. ALDER + y > AKASSE_TID

Ad. 1: ALDER + Y skal være højere end 11, ellers konverteres AKASSE_TID til blank.

Denne antagelse holder.

Kategorisering

Fl.g. grupper er konstrueret:
[0-24 [= A
[24-48 [= B
[48-72 [= C
[72-96 [= D
[96 -120 [= E
[120-] = F

BRUTTO_ÅR

Brutto årsindtægt

Range [1-500000]

Datakvalitet

Blanke: 32 eller 0.16%

En række indeholder værdien “87878787”, to rækker indeholder “54545454”, og tre indeholder “21212121”. Disse værdier sletter jeg under antagelse om, at det enten er en fejl eller en kode.

Følgende sammenhæng mellem "BRUTTO_ÅR" og "BRUTTO_HUS" antages:

1. BRUTTO_ÅR $\leq$ BRUTTO_HUS

BRUTTO_ÅR	BRUTTO_HUS	Ny brutto
57300028	815000	573000

Desuden indeholder datasættet 2 meget høje værdier: 32565815 og 24209513, som er slettet.

Kategorisering

Flg. grupper er konstrueret:
[0-100K [= A
[100K-150K [= B
[150K-200K [= C
[200K-250K [= D
[250K -300K [= E
[300K -350K [= F
[350K -400K [= G
[400K -450K [= H
[450K -500K [= I
[500K-] = K

NETTO_MM

Ansøgerens månedlige nettoindkomst - eller måske månedlige netto rådighedsbeløb.

Range [1-450000]

Datakvalitet:

Blanke: 41 eller 0,2%

Jeg har overvejet flg. sammenhæng mellem "NETTO_MM" og "BRUTTO_ÅR":

1. NETTO_MM * 12 $\leq$ BRUTTO_ÅR

Men BRUTTO_ÅR kan stamme fra ansøgerens årsopgørelse, dvs. sidste års indkomst, og NETTO_MM kan stamme fra de foregående tre måneders lønsedler. Derfor kan tallene ikke sammenlignes, og jeg bruger ikke samenhængen.

Følgende værdier er slettet fra denne kolonne: 878788, 545454 og 21212. Det er rækker svarende til de fra BRUTTO_ÅR, som også blev slettet.

Kategorisering

Flg. grupper er konstrueret:
[0-3000 [= A
[3000-6000 [= B
[6000-9000 [= C
[9000-12000 [= D
[12000 -15000 [= E
[15000 -18000 [= F
[18000 -21000 [= G
[21000 -24000 [= H
[24000 -27000 [= I
[27000 -30000 [= J
[30000-] = K

BRUTTO_HUS

Denne attribut beskriver husstandens bruttoløn.

Range [2-11500000]

Blanke: 13484 eller 65,88%

Attributten er *fravalgt*.

PI_TID

I hvor mange måneder har ansøgeren været i nuværende pengeinstitut?

Resultatet kommer fra data-sættets 2 kolonner: TID_PI_AA og TID_PI_MM (hhv. antal år og måneder).

Range [0-1008]

Datakvalitet:

Blanke: 2664 eller 13,02%

399 af kolonnerne TID_PI_MM indeholdt forurenede data og er blevet slettet.

Jeg antager desuden samme sammenhæng som under ANCIENNITET:

1. ALDER + y > PI_TID

Ad. 1: ALDER + Y skal være højere end 11 år, ellers konverteres PI_TID til blank.

1 stk. PI_TID

Kategorisering

Flg. grupper er konstrueret:
[0-24 [= A
[24-48 [= B
[48-72 [= C
[72-96 [= D
[96 -120 [= E
[120-] = F

BETAL_MET

Betalingsmetode

Set {"", K, P}
K = Girokort, P=PBS

Datakvalitet:

Blanke: 9488 eller 46,36%

Attributten er *fravalgt*.

DANKORT

Har ansøgeren et Dankort?

Set {"", J, N}

Datakvalitet:

Blanke: 13781 eller 67,34%

Attributten er *fravalgt*.

VISADK

Har ansøgeren et Visa/Dankort

Set {"", J, N}

Blanke: 12418 eller 60,68%

Attributten er *fravalgt*.

BENZINKORT

Har ansøgeren et benzinkort?

Set {"", J, N}

Datakvalitet:

Blanke: 13307 eller 65,026%

Attributten er *fravalgt*.

KREDITKORT

Har ansøgeren et kreditkort?

Set {"", J, N}

Datakvalitet:

Blanke: 18615 eller 90.96%

Attributten er *fravalgt*.

RKI_HIT

Er ansøgeren registreret i RKI?

Set {"", J, N}

Datakvalitet:

Blanke: 6485 eller 31,69%

MEDANS

Er der en medansøger?

Set { J, N}

ARBG_FL

Denne attribut bruges til at samle information om arbejdsforhold. Det drejer sig hhv. om, hvorvidt arbejdsgivers navn og ansættelsesperiode er registreret.

Set {J, N, X, Z}
J = Arbejdsgiver er oplyst.

N = Arbejdsgiver er ikke oplyst, men ansættelsesperiode er
X = Ingen er oplyst (uden arbejde)
Z = Arbejdsgiver er oplyst men ansættelsesperiode er ikke

Overvejelse: Skal attributten medtages, når der er tale om infomation, der allerede er tilgængelig i datamaterialet?

Attributten er *fravalgt*.

CIVILSTAND

Ansøgerens civilstand.

Set {"", A, E, G, K, N, O, S}
blank = Ingen information
A = Alene
E = Separereret
G = Gift
K = Skilt
N = Enke/Enkemand
O = ? (data-/indtastnings -fejl?)
S = Samlever

Datakvalitet:

Blanke: 282 eller 1,38%

KØN

Ansøgerens køn.

Set {K, M}

TLF_FLAG

Denne attribut fortæller, hvilke telefonoplysninger ansøgeren har givet.

Set {1,2,3,4,5,6,7,8,9}
1= Kun privatnummer
2= Kun privat mobilnummer
3= Kun arbejdsnummer
4= Kun firma mobilnummer
5= Både privat og arbejdsnummer

6= Både privat og firma mobilnummer
7= Både privat mobil og arbejdsnummer
8= Både privat mobil og firma mobilnummer
9= Ingen oplysninger

Overvejelse: Skal attributten medtages - har den relevans?

Attributten er *fravalgt*.

ANS_DATO

Ansøgningsdato.

Range [8/Apr/1998 - 8/Sep/2003]

ALDER

Hvor gammel er ansøgeren.

Range [18-90]

Datakvalitet

Der er en ansøger, der er blevet registreret med en alder på 0 år. Det må være en fejl, og derfor er værdien slettet.

Kategorisering

Flg. grupper er konstrueret:
[0-23 [= A
[23-28 [= B
[28-33 [= C
[33-38 [= D
[38 -43 [= E
[43 -48 [= F
[48 -53 [= G
[53 -58 [= H
[58 -63 [= I
[63-] = K

ALDER_ANS2

Måske hvor gammel ansøger 2 er.

Range [0-99]

Datakvalitet

Jeg ved ikke, hvad denne attribut skal bruges til.

Tilsyneladende er det ikke medansøgerens alder, der her er tale om, for hvis jeg sletter alle de værdier, hvor MEDANS=N, bliver det til 18202 stk.

Måske kan man være to ansøgere til et kreditkort, men i så fald er der værdier, der ikke giver mening (måske koder?), fx en alder på 0, 10, 98 og 99.

Attributten er *fravalgt*.

DCODE

Konklusion på lånansøgningen.

Set {"", Afs, Bev, Ref}
""= Ingen information (giver det mening her?)
Afs= Afslag
Bev= Bevilliget
Ref= Afventer information

Datakvalitet:

Blanke: 3 eller 0,01%

GB

Var/er ansøgeren beskrevet som værende god eller dårlig?

Set {0,1,99}
0= God
1= Dårlig
99= Afslag

Datakvalitet

I datasættet findes der også en kode 2, som også betyder "dårlig". Da der tilsyneladende ingen forskel er på kode 1 og 2, er de lagt sammen.
114 stk. med kode 2 er rettet til kode 1.

Kode 99 kan man se bort fra. Det er ansøgere, der på forhånd fik afslag, så derfor ved man ikke, om de ville være gode betalere eller ej.

LÅNE_MAX

Hvor højt kreditmaksimum blev bevilliget?

Range [0-30000]

Kategorisering

Flg. grupper er konstrueret:

[0-5000] = A
]5000-10.000] = B
]10.000-15.000] = C
]15.000-20.000] = D
]20.000-25.000] = E
]25.000-30.000] = F

Bilag 3
CD