UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Marko Šmid

Uporaba metod za odkrivanje znanj iz podatkov v bančnem trženju

DIPLOMSKO DELO

Mentor:
doc. dr. Blaž Zupan

LJUBLJANA, september 2002

Povzetek

Bančna dejavnost se je v zadnjih letih močno spremenila. Tradicionalne, produktno usmerjene banke, so se morale preoblikovati v banke, usmerjene proti stranki, če so hotele ohraniti svoj položaj in biti konkurenčne. Najbolj uspešne so bile pri tem tiste banke, ki so uspele izkoristiti vse potencialne prednosti novih tehnologij. Sem sodijo predvsem sistemi za podporo odločanju, v okviru katerih se v zadnjem času uporabljajo tudi različne metode za odkrivanje znanj iz podatkov.

V okviru diplomske naloge smo preučili uporabo metod za odkrivanje znanj iz podatkov za potrebe bančnega trženja. Naloga temelji na konkretni študiji v oddelku trženja slovenske banke Abanke. Izdelana je z upoštevanjem metodologije CRISP-DM, ki predpisuje življenjski cikel projektov s področja odkrivanja znanj iz podatkov v poslovne namene. S strojnim učenjem smo gradili modele, ki so informacijsko podprli metodo ocenjevanja stranke (ang. *Customer Scoring*), ki se uporablja pri ciljno usmerjenih tržnih akcijah. Preučili smo različne metode strojnega učenja in ovrednotili kvaliteto dobljenih modelov. Rezultati so pokazali na potencialno veliko uporabnost metod za odkrivanje znanj iz podatkov na področju trženja. Izdelani model strojnega učenja bo lahko Abanka uporabila kot pripomoček pri pripravi ciljno usmerjenih tržnih akcij za produkt Abanet in kot pomoč pri razumevanju obnašanja svojih strank.

Ključne besede

odkrivanje znanj iz podatkov
strojno učenje
metodologija CRISP-DM
navzkrižno trženje
ciljno usmerjena tržna akcija
podatkovno skladišče
ocenjevanje stranke

Zahvala

Zahvaljujem se mentorju, doc. dr. Blažu Zupanu, za spodbude, nasvete in vodenje pri izdelavi diplomskega dela na tako zanimivem področju, kot je uporaba metod za odkrivanje znanj. Za zaupane podatke in pomoč pri njihovem razumevanju se zahvaljujem podjetju Abanka. Iskrena hvala pa tudi vsem bližnjim za podporo v času mojega študija.

Ljubljana, september 2002 Marko Šmid

Kazalo

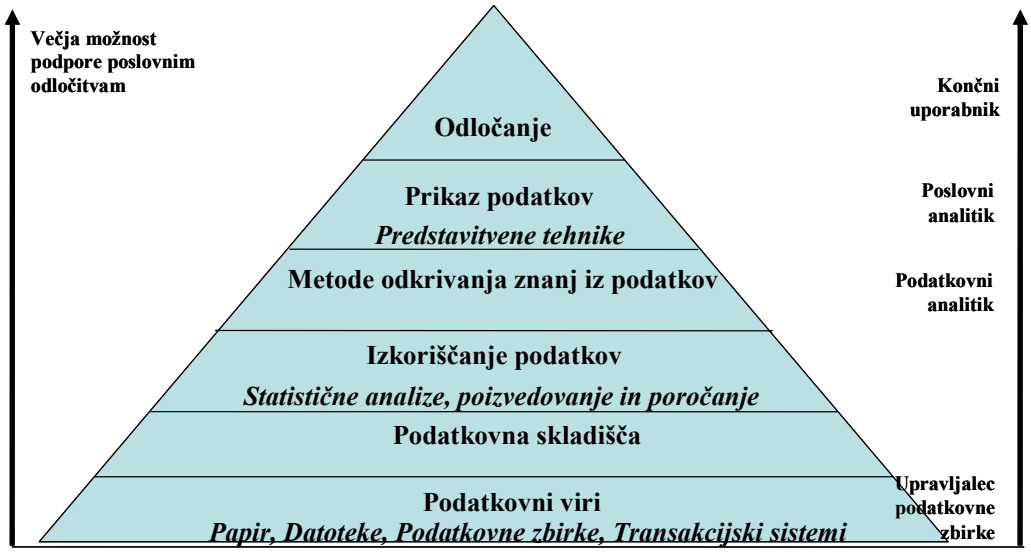
1 Uvod

Bančna dejavnost spada med tista poslovna področja, ki so se v zadnjih letih močno spremenila. Tradicionalne, produktno usmerjene banke, so se morale preoblikovati v banke, usmerjene proti stranki, če so hotele ohraniti svoj položaj in biti konkurenčne. Danes je stranka tista, ki je v središču poslovanja banke, implementiran celovit sistem upravljanja odnosov s stranko (CRM, ang. *Customer Relationship Management*) pa eden od glavnih ciljev bank za prihodnost. Vpeljava sistema za upravljanje odnosov s stranko pa zahteva nov način dela in drugačno informacijsko podporo, kot jo imajo banke danes. Banke potrebujejo veliko več podatkov, informacij in znanj, če želijo razumeti potrebe, posebnosti in obnašanje svojih strank. Le tako lahko definirajo celoten splet trženja, ki obsega razvoj produktov, prodajne poti, promocije, cenovno politiko in procese.

Informacijske rešitve morajo podpreti tudi celoten proces odločanja (slika 1.1), ki se začne z zbiranjem različnih vrst podatkov, njihovim združevanjem v obliki podatkovnega skladišča in se nadaljuje preko analiz podatkov, odkrivanja znanja iz podatkov in predstavitve podatkov do samega odločanja. Iz slike 1.1 je razvidno, da brez vpeljave metod za odkrivanje znanja iz podatkov ni možno vzpostavit popolnega procesa odločanja.

Metode za odkrivanje znanja iz podatkov se danes v bankah uporabljajo predvsem za pomoč pri navzkrižni prodaji (ang, *Cross-selling*), upravljanje rizikov, identifikacijo potencialnih novih strank, razvoj dobičkonosnih profilov strank, razumevanje navad strank, ugotavljanje zlorab kreditnih kartic in prilagajanje ciljno usmerjenih tržnih akcij.

V večini bank se največji neizkoriščen potencial skriva v obstoječih strankah. Večina jih ima samo en ali dva produkta, banke pa jih ponujajo na desetine, npr. različne vrste kreditov, vezav in hranilnih knjižic. Povečevanja števila produktov na posamezno stranko se banke danes lotevajo s pomočjo navzkrižne prodaje in ciljno usmerjenih tržnih akcij. Pri tem uporabljajo metode za odkrivanje znanj iz podatkov za napovedovanje, kako zanimiv je posamezen produkt za izbrano stranko. S pomočjo teh informacij potem izberejo ciljno populacijo, ki bo uporabljena v tržni akciji. Tovrstne rešitve poznamo pod pojmom ocenjevanje stranke (ang. *Customer Scoring*) [Thearling, 2002].



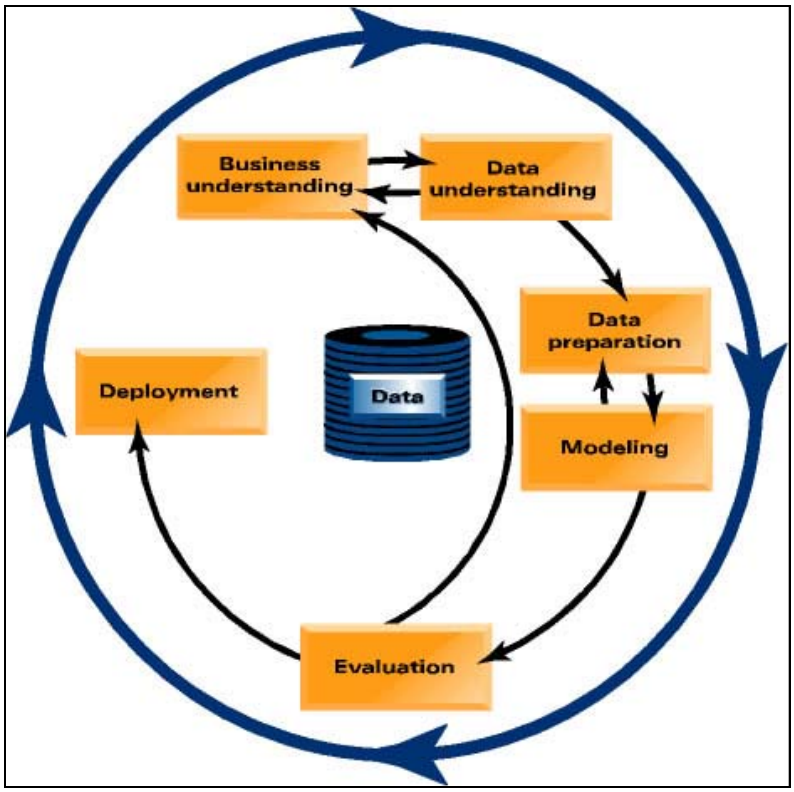
Slika 1.1 Proces odločanja v organizaciji

V diplomskem delu smo izdelali informacijsko podporo za proces ocenjevanja strank s pomočjo modelov, ki smo jih gradili s strojnim učenjem in tehnikami odkrivanja znanj iz podatkov. Ocenili in primerjali smo uspešnost dveh metod strojnega učenja (naivni Bayes in odločitveno drevo) pri napovedovanju verjetnosti, da je nek produkt zanimiv za stranko, ki tega produkta še nima. V nalogi smo se omejili na konkretno študijo v slovenski banki Abanki.

V uvodu je opisana metodologija, ki smo jo uporabili pri izdelavi naloge in podrobno so predstavljeni cilji diplomskega dela. Drugo poglavje opisuje poslovni problem, za katerega smo izdelali rešitev. Tretje poglavje opisuje zbirko podatkov, ki smo jo uporabili v eksperimentih pri študiju obnašanja metod strojnega učenja. Četrto poglavje opisuje uporabljene tehnike modeliranja in ocenjevanja modelov, ki smo jih implementirali v sistemu za strojno učenje Orange. V petem poglavju so opisani eksperimenti in predstavljeni rezultati ter ugotovitve primerjave metod strojnega učenja.

1.1 Referenčni model CRISP-DM

Celotna diplomska naloga je bila izdelana z upoštevanjem navodil in priporočil metodologije CRISP-DM (ang. *Cross-Industry Standard Process for Data Mining*) [Chapman in sod., 1999]. Glavni razlog za njeno izbiro je bil v tem, da zelo jasno opisuje, kako se spopasti s poslovnim problemom in ga nato preoblikovati v problem odkrivanja znanja iz podatkov. Metodologija predpisuje življenski cikel projekta, proces, njegove faze in pripadajoče naloge ter relacije med nalogami.



Slika 1.2 Faze procesnega modela CRISP-DM

Procesni model, ki ga predpisuje metodologija CRISP-DM je sestavljen iz šestih faz. Zaporedje faz ni striktno predpisano. Skoraj vedno se je potrebno pomikati med fazami naprej ali nazaj, glede na rezultate ob koncu posamezne faze. Na sliki 1.2 je predstavljen celoten procesni model. Zunanji krog predstavlja ciklično naravo samega procesa odkrivanja znanj iz podatkov, puščice pa kažejo najpomembnejše in najpogostejše odvisnosti med fazami.

Faze procesnega modela CRISP-DM so:

- **Razumevanje poslovnega problema** (ang. *Business Understanding*). Osnovni namen prve faze je razumevanje projektnih zahtev in ciljev s poslovnega vidika in nato spreminjanje tega znanja v definiranje možnosti uporabe metod za odkrivanje znanja iz podatkov.
- **Razumevanje podatkov** (ang. *Data Understanding*). Druga faza se začne z pridobivanjem osnovne množice podatkov in se nadaljuje z analizo zbranih podatkov, ugotavljanjem njihove kvalitete in iskanjem skritih informacij v posameznih zanimivih podmnožicah.
- **Priprava podatkov** (ang. *Data Preparation*). Tretja faza vsebuje vsa potrebna opravila za pripravo končne množice podatkov, ki bo vhod v orodje za strojno učenje. Naloge znotraj te faze vsebujejo določitev tabel, zapisov in atributov ter vse potrebne transformacije in čiščenje podatkov glede na zmožnosti izbranega orodja za strojno učenje.
- **Modeliranje** (ang. *Modeling*)**.** Faza modeliranja vključuje izbiro ustreznih tehnik modeliranja in določitev možnih optimalnih vrednosti parametrov za posamezno tehniko. Skoraj vedno obstaja več metod za odkrivanje znanja iz podatkov za enak tip problema. Nekatere metode imajo zelo specifične zahteve glede oblike podatkov, zato je večkrat potrebna vrnitev v fazo priprave podatkov.
- **Vrednotenje** (ang. *Evaluation*). V tej fazi se izdelajo vsi potrebni modeli in med njimi izberejo najboljši kandidati. Pred prehodom v zadnjo fazo je potrebno vse izbrane modele še enkrat temeljito preveriti in oceniti. Preveriti je potrebno predvsem to, ali obstajajo pomembni poslovni cilji, ki jih nismo dovolj upoštevali v modelu. Na koncu te faze se sprejme odločitev o uporabi posameznih modelov strojnega učenja.
- **Prehod v produkcijo** (ang. *Deployment*). Izdelava modela tipično ne pomeni konec projekta. Pridobljeno znanje iz modelov je potrebno zbrati in preoblikovati v obliko, ki bo primerna za uporabnika. Glede na zahteve lahko ta faza vsebuje samo pripravo ustreznih poročil ali pa vse do izdelave kompleksnega ponovljivega procesa odkrivanja znanj iz podatkov.

1.2 Motivacija in cilji

Motivacija za diplomsko dela so bile uspešne izkušnje določenih ameriških bank [Berry, 2000] pri uporabi metod odkrivanja znanja iz podatkov v procesu ciljno usmerjenega trženja svojih produktov in podatek, da si večina uspešnih bank v svetu danes že pomaga z metodami odkrivanja znanj iz podatkov kot enim od osnovnih pripomočkov pri vpeljavi novega načina upravljanja odnosov s strankami (CRM, ang. *Customer Relationship Management*).

Celotna diplomska naloga je bila omejena na konkretno študijo v oddelku trženja v slovenski banki Abanki. Cilj diplomske naloge je bil podpreti metodo ocenjevanja strank z modeli strojnega učenja. Pri tem smo se omejili na en sam produkt Abanet. Zanj smo razvili model napovedovanja potencialnih novih uporabnikov. Pri tem smo uporabili metode strojnega učenja, ki jih omogoča sistem za strojno učenje Orange. Preverili smo dva modela: model naivnega Bayesa in odločitveno drevo. Kvaliteto dobljenih modelov smo ovrednotili na podlagi klasifikacijske točnosti, površine pod krivuljo ROC in grafa, ki prikazuje krivuljo odziva (ang. *Lift Chart*). Cilj diplomske naloge je bil tudi priprava skript SQL za pripravo podatkov in skript za programski paket Orange, ki bodo uporabne v banki tudi v prihodnje.

2 Poslovna problematika in rešitev

Abanka d.d. je danes četrta največja slovenska banka glede na bilančno vsoto. Kot tržno usmerjenemu podjetju je banki eden od glavnih strateških ciljev nenehna rast in povečevanje dobička. To pa je mogoče doseči v banki na dva načina: s pridobivanjem novih strank ali pa preko povečanja števila in uporabe produktov, ki jih posamezna stranka že ima. Glede na zasičenost trga je prvi način precej težji in dražji. Večji izziv predstavlja drugi način, ker gre za že obstoječe strank, ki jih banka dobro pozna in ima o njih zbranih veliko podatkov, ki jih lahko koristno uporabi. Poglavje opisuje poslovni problem, ki smo ga reševali, proces ocenjevanja strank, ki smo ga podprli in rešitev, ki smo jo izdelali.

2.1 Poslovni problem

Problem, s katerim se banka danes sooča je sledeč: ne glede na to, da ima banka v svoji ponudbi za fizične osebe več deset produktov, jih ima večina strank samo enega ali dva. Zato se v oddelku za trženje nenehno trudijo povečevati število produktov na posamezno obstoječo stranko. Glede na omejena finančna sredstva za tržne akcije so se odločili, da bodo poskušali za vsak produkt poiskati tisti segment obstoječih strank, za katere obstaja velika verjetnost, da se bodo v prihodnosti za ta produkt odločile.

Največji izziv danes v banki predstavljajo novi produkti, ki jih prinaša internet oz. elektronsko poslovanje. Tako banka ponuja svojim strankam novo storitev Abanet (slika 2.1), elektronsko banko za fizične osebe in samostojne podjetnike, ki omogoča opravljanje bančnih ter borznih poslov. Stranke lahko z uporabo sistema Abanet plačujejo položnice, nakazujejo denar, spremljajo promet in stanje na vseh svojih računih ter upravljajo in spremljajo svoje vrednostne papirje. Poslovna vrednost Abaneta je predvsem v nižji ceni storitve za uporabnika in nižjih stroških banke za opravljeno transakcijo. Izkušnje tudi kažejo, da ima uporabnik ponavadi samo eno elektronsko banko. V banki, kjer jo uporablja, bo verjetno ostal, tudi, če trenutno uporablja več bank.

Slika 2.1 Abanet

Abanka je začela s trženjem produkta Abanet v začetku leta 2001. V prvih mesecih je število uporabnikov hitro naraščalo, potem pa se je proti koncu leta začelo umirjati. Ob koncu leta je imelo približno 10% imetnikov tekočega računa novo storitev Abanet. To pa je precej manjši delež, kot je število strank, ki je že imelo sredi leta 2001 dostop do interneta. Teh je bilo približno 55%. Zato si je banka zadala za cilj povečati število uporabnikov storitve Abanet. Zaradi omejenih finančnih sredstev so se odločiti za trženje produkta na omejeni množici strank. Med več kot 70.000 obstoječimi strankami, ki Abaneta še nimajo, Abanka želi poiskati tiste stranke, za katere bi bil ta produkt potencialno zelo zanimiv. V pričujoči nalogi smo za ta namen s strojnim učenjem izdelali model, ki za dano stranko lahko napove, kako zanimiv bi bil zanjo dani produkt. Na osnovi modela je moč stranke urediti v skladu z njihovim potencialnim zanimanjem za produkt, ter na ta način poiskati ciljno skupino strank. Za te izbrane stranke bodo pripravili ciljno usmerjeno tržno akcijo, pri kateri želijo doseči bistveno večji odziv, kot če bi stranke izbirali naključno. Do sedaj so za izbor ciljne množice uporabljali razne statistične metode in izkušnje tržnih specialistov, za prihodnost pa so želeli preveri nove pristope, med katerimi so izbrali tudi uporabo metod za odkrivanje znanj iz podatkov.

2.2 Ocenjevanje stranke

Proces ocenjevanja stranke (ang. *Customer Scoring*) [Thearling, 2002] se uporablja za prepoznavanje ciljne množice strank, ki bo sodelovala v usmerjeni tržni akciji. Modeli strojnega učenja se v procesu uporabljajo za napovedovanje obnašanja stranke v prihodnosti. Izhod modela je neka ocena oz. napoved, ki je lahko predstavljena kot število, niz znakov ali pa celo kot podatkovna struktura. Največkrat pa jo zasledimo v obliki številke, npr. kot verjetnost, da se bo izbrana oseba odzvala na določeno oglaševalsko akcijo. Proces ocenjevanja, če ga opazujemo kot samostojno aplikacijo, poteka navadno v naslednjem vrstnem redu:

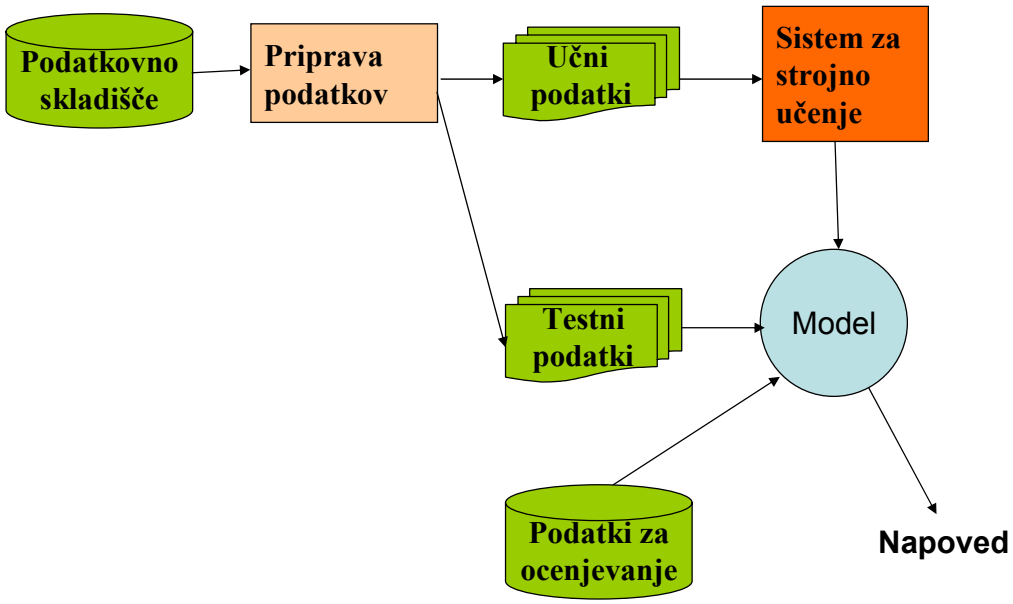
- Proces se prične z izbiro podatkovne zbirke, ki vsebuje podatke o obstoječih ali potencialnih strankah. Ponavadi je ta zbirka del obsežnejšega podatkovnega skladišča, včasih pa tudi samo interna podatkovna zbirka oddelka za trženje.
- Strokovnjak za trženje izbere segment strank, ki ga zanimajo (npr: stranke, ki imajo določen produkt in so stare več kot 40 let).
- Za izbrano skupino strank se izvede s pomočjo modela strojnega učenja za napovedovanje izračun napovedi potencialnega zanimanja za določen produkt. Navadno je to neka vrednost med 0 in 1, ki predstavlja verjetnost, da se bo npr. stranka odločila za nek produkt, če ji pošljemo ponudbo.
- Po opravljenem izračunu je potrebno stranke urediti glede na napovedano vrednost v padajočem vrstnem redu. Najboljših nekaj odstotkov iz prioritetne lestvice se izbere za sodelovanje v tržni akciji.

Ena glavnih prednosti takega pristopa je predvsem za, da nam omogoča izvajanje zelo pogostih, učinkovitih in strogo ciljno usmerjenih tržnih akcij, ki zahtevajo manjša finančna sredstva. Pri tem lahko dosežemo bistveno boljše rezultate glede na vložena sredstva, kot s starim pristopom, ki je temeljil na dragih, redkih in masovnih akcijah.

2.3 Opis rešitve

Za rešitev poslovnega problema smo uporabiti metode za odkrivanje znanja iz podatkov. Z modelom strojnega učenja smo podprli zgoraj opisani proces ocenjevanja strank. Cilj nam je bil izdelati model, ki temelji na tehnikah razvrščanja in bo namenjen napovedovanju verjetnosti, da bo določena stranka postala v bljižni prihodnosti imetnik produkta Abanet. S pomočjo modela bi potem ocenili vse stranke, ki Abaneta še nimajo

in jih razvrstili glede na napovedano oceno od tiste z največjo do tiste z najmanjšo verjetnostjo. Prvih *N* strank iz prioritetne lestvice predstavlja najbolj perspektivno množico potencialnih novih kandidatov za ta produkt.



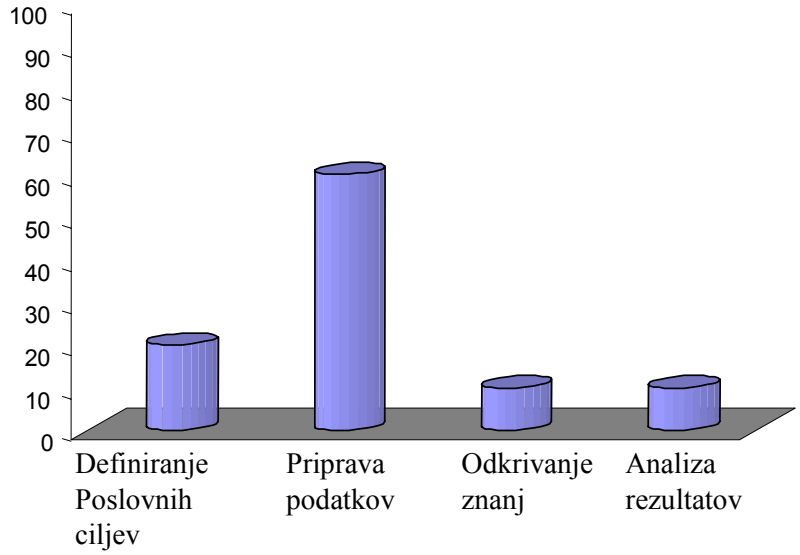
Slika 2.2 Koncept rešitve za ocenjevanje strank

Koncept rešitve je predstavljen na sliki 2.2 in vsebuje:

1. Pripravo osnovne množice podatkov, ki jih pridobimo iz podatkovne skladišča, ki se nahaja v podatkovni zbirki Oracle.
2. Razdelitev osnovne množice na učno in testno množico.
3. Izdelavo modela za napovedovanje z uporabo učne množice podatkov in metod strojnega učenja v sistemu Orange.
4. Preverjanje stabilnosti modela s pomočjo testne množice podatkov.
5. Uporabo izbranega modela za napovedovanje na množici podatkov, ki jih želimo oceniti.

3 Podatki

Priprava kvalitetne množice podatkov predstavlja eno od najbolj zahtevnih faz v projektu odkrivanja znanj iz podatkov. Hkrati je to ponavadi tudi časovno najdaljša faza. Glede na podatke [Kimball, 1998] predstavlja faza priprave podatkov tudi do 70% porabljenega časa (slika 3.1) na projektu. V poglavju so opisani postopki priprave osnovne množice podatkov, definiranje potrebnih atributov in rezultati osnovne analize podatkov.



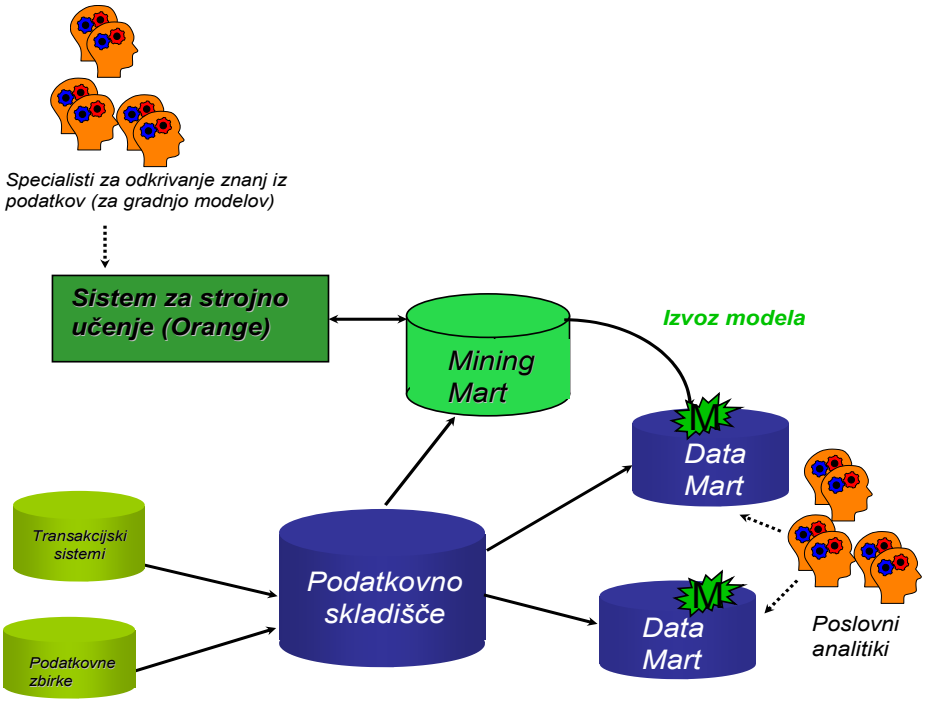
Slika 3.1 Razmerje porabe časa po korakih procesa odkrivanja znanj iz podatkov

3.1 Pridobivanje osnovne množice podatkov

Na podlagi analize poslovnih zahtev in ciljev smo ugotovili, da mora osnovni nabor podatkov vsebovati matične podatke o stranki (podatki o prebivališču, starosti, spolu in zaposlitvi, itd.), podatke o vseh produktih, ki jih stranka že ima v banki in podrobne podatke o produktih tekoči račun (TR) in Visa kartica. Pri matičnih podatkih smo naleteli na problem kvalitete podatkov. Prvi razlog je, da banke zaradi zakonskih ovir ne smejo zbirati vseh želenih podatkov in drugi, da v preteklosti ni bilo potrebe po zbiranju podatkov zaradi drugačnega poslovnega modela. Pri podatkih o posameznem produktu smo naleteli na problem, ker imajo različni produkti različno, največkrat zelo težko združljivo informacijsko podporo.

Zaradi zgoraj opisanih problemov smo se odločili, da za osnovni vir podatkov izberemo podatkovno skladišče, ki ga banka že ima. Podatkovno skladišče po definiciji

[Kimball, 1998] predstavlja celovito podatkovno zbirko, ki na enem mestu vsebuje združene tekoče in zgodovinske podatke o poslovanju organizacije, namenjeno predvsem različnim oblikam podpore odločanju. Njegova glavna prednost je predvsem v tem, da lahko dobimo vse informacije na enem mestu. S tem se izognemo večini problemov, ki so povezani z integracijo podatkov iz različnih podatkovnih virov. Problem uporabe podatkovnega skladišča za potrebe odkrivanja znanj iz podatkov je v tem, da je skoraj vedno zaradi velike količine podatkov normaliziran [Berry and Linoff, 2000]. To pa pomeni, da je potrebno za pripravo podatkov izvajati časovno zelo zahtevne poizvedbe, ki lahko trajajo ure ali dneve. Druga past pa se navadno skriva v agregaciji in sumarizaciji, ki se uporabljata za zmanjšanje velikosti podatkovnega skladišča in povečanje hitrosti poizvedovanja. Pri tem pa se največkrat izgubi osnovna informacija o transakciji, ki pa je zelo pomembna pri odkrivanju znanj iz podatkov.



Slika 3.2 Arhitektura sistema za podporo odločanju

Abanka ima v podatkovnem skladišču zbrane na enem mestu vse podatke o poslovanju fizičnih oseb v preteklih letih. Zaradi posebnih zahtev so že od samega začetka skladišče gradili na atomarnem nivoju. Ugotovili smo, da se vsi podatki, ki jih želimo uporabiti in jih banka ima, nahajajo v njem. Do podatkov v podatkovnem skladišču lahko uporabniki v banki prihajajo na različne načine: od namenskih aplikacij do posebnega orodja za poizvedovanje in poročanje. Po testiranju zmožnosti vseh

obstoječih orodij smo se odločili, da za potrebe pridobivanja, čiščenja in preoblikovanja podatkov iz podatkovnega skladišča za naš konkreten primer izdelamo lastne skripte SQL (priloga 7.2) in jih izvajamo v grafičnem okolju programskega paketa TOAD. Glavni razlog je bil v časovni kompleksnosti poizvedb, ki so lahko trajale tudi po 40 in več ur. Vzrok je bil v tem, da je bilo potrebno pri nekaterih izvedenih atributih za vsak zapis izvajati poizvedbe na več deset milijonski tabeli bančnih transakcij. Rezultat izvajanja skript SQL je bil »*Mining Mart*«, ki nadgrajuje obstoječo arhitekturo sistema za podporo odločanju (slika 3.2) in bo uporaben tudi v bodoče.

Za osnovno množico smo izbrali imetnike produkta tekoči račun (TR), ki so imeli 31.12.2001 v banki podpisano pogodbo za ta produkt. Dobili smo množico podatkov z več kot 80000 zapisi, v kateri vsak posamezen zapis vsebuje podrobne podatke o enem imetniku: matične podatke, podatke o produktih, ki jih ima oz. nima, podrobne podatke o tekočem računu in podatke vezane na njegove kreditne kartice. Zapis vsebuje tudi klasifikacijsko oznako, ki označuje ali je stranka na izbrani dan imela produkt Abanet ali ne.

Za celotno osnovno množico smo izdelali eno samo tabelo z vsemi izbranimi atributi (tabela 3.1). Polnjenje tabele s podatki iz podatkovnega skladišča smo izvajali inkrementalno. Vrednosti za posamezni atribut smo dobili na enega od dve načinov:

- z direktno preslikavo iz modela podatkovnega skladišča v našo tabelo ali
- s pomočjo podprograma, napisanega v jeziku SQL, s katerim smo implementirali postopke za polnjenje vrednosti atributa.

Podatkovni tip	Opis
Splošni podatki o stranki	
NUMBER	številka stranke v banki
NUMBER	identifikator fizične osebe
CHAR	poštna številka
CHAR	država stalnega prebivališča
NUMBER	starost stranke v letih
NUMBER	mesec rojstva
CHAR	spol stranke
CHAR	zaposlen na dan opazovanja
CHAR	trenutna boniteta na dan opazovanja
CHAR	minimalna boniteta v opazovanem letu
CHAR	maksimalna boniteta v opazovanem letu
NUMBER	koliko mesecev je že stranka banke
NUMBER	število aktivnih računov odprtih v banki
NUMBER	število aktivnih produktov odprtih v banki

CHAR	stranka ima žiro račun
CHAR	stranka ima devizni račun
CHAR	stranka ima devizno knjižico
CHAR	stranka ima hranilno knjižico
CHAR	stranka ima tolarski depozit
CHAR	stranka ima devizni depozit
CHAR	stranka ima rentno varčevanje
CHAR	stranka ima namensko varčevanje
CHAR	stranka ima varčevalno knjižico
CHAR	stranka ima kredit
CHAR	stranka ima katerokoli visa kartico
CHAR	stranka ima nacionalno varčevalno shemo
CHAR	stranka ima abanet
CHAR	stranka ima trgovalni račun
Podatki o tekočem računu stranke	
NUMBER	koliko mesecev ima odprt tekoči račun
CHAR	trenutna boniteta za tekoči račun na dan opazovanja
NUMBER	trenutno stanje na tekočem računu
NUMBER	povprečno stanje na tekočem računu v opazovanem letu
NUMBER	trenutni limit na tekočem računu na dan opazovanja
NUMBER	koliko dni je bil prekoračen limit v opazovanem letu
NUMBER	število vseh prilivov na tekoči račun v opazovanem letu
NUMBER	število vseh odlivov na tekoči račun v opazovanem letu
NUMBER	število dvigov na bankomatu v opazovanem letu
NUMBER	celoten promet v dobro v opazovanem letu
NUMBER	celoten promet v breme v opazovanem letu
NUMBER	povprečen osebni dohodek v opazovanem letu
NUMBER	število pooblaščencev, ki jih ima stranka
CHAR	stranka ima trajnik na tekočem računu
Podatki o Visa karticah stranke	
NUMBER	število visa kartic, ki jih ima stranka
CHAR	stranka ima visa klasik kartico
CHAR	stranka ima visa gold kartico
CHAR	stranka ima visa elektron kartico
CHAR	stranka ima visa kreditno kartico
CHAR	stranka ima visa poslovno kartico
NUMBER	koliko mesec ima najstarejšo visa kartico
NUMBER	celoten promet z visa karticami v opazovanem letu

Tabela 3.1: Izbrani atributi osnovne množice

Pri pripravi podatkov smo se odločili, da izberemo podatke, ki so veljali na dan 31.12.2001. Pravilno bi bilo, da bi za vsako stranko zbrali podatke na zadnji dan, preden je postala imetnik opazovanega produkta. Razlog, da nismo izbrali zadnje

rešitve je v tem, da zahteve priprava take množice veliko količino računalniških virov in časa.

3.2 Osnovna analiza podatkov

Končni rezultat polnjenja podatkov iz podatkovnega skladišča je bila ena sama tabela oz. osnovna množica podatkov. Za njeno analizo smo uporabili orodje Orange [Demšar in Zupan, 2002], s pomočjo katerega smo izdelali skripte (priloga 7.1) za analizo karakteristik osnovne množice podatkov: število razredov, število atributov, informacije o manjkajočih vrednostih v podatkih in distribucije razredov.

Osnovna množica se sestoji iz 47 atributov, med katerimi je 14 zveznih in 33 diskretnih. Opazovani atribut je diskreten in ima dva razreda: uporabnik ima produkt Abanet (razred 1) in uporabnik nima produkta Abanet (razred 0). Njegova distribucija je predstavljena v tabeli 3.2.

	Število primerov	Odstotek
Razred 1 (ima produkt)	8685	10.7 %
Razred 0 (nima produkta)	72183	89.3 %
Skupaj	80868	100 %

Tabela 3.2 Distribucija opazovanega atributa

Analiza je pokazala, da je atributov z manjkajočimi vrednostmi malo. Večina atributov je brez manjkajočih vrednosti, med tistimi, pri katerih vrednosti manjkajo (tabela 3.3) pa smo ugotovili, da atributa z največjim deležem manjkajočih vrednosti VISA_PROMET in VISA_TRAJANJE nista problematična, ker manjkajoče vrednosti predstavljajo tiste stranke, ki nimajo Visa kartice, kar je pravilno. Pri ostalih štirih atributih pa vrednosti resnično manjkajo.

Odstotek manjkajočih vrednosti	Atribut
70.5%	VISA_PROMET
68.5%	VISA_TRAJANJE
36.1%	SPOL
15.5%	ZAPOSLEN
4.5%	STAROST
4.5%	MESEC_ROJSTVA

Tabela 3.3. Atributi z manjkajočimi vrednostmi

4 Metode modeliranja in ocenjevanja uspešnosti učenja

V namene rangiranja bančnih strank smo uporabili dve metodi strojnega učenja: naivnega Bayesa in odločitveno drevo. Za izbiro podmnožice atributov (FSS, ang. *Feature Subset Selection*) smo uporabili metodo, ki uporablja algoritem ReliefF [Kononenko,1997]. Ker naivni Bayes zahteva diskretne atribute smo diskretizacijo zveznih atributov opravili z metodo Fayyad-a in Irani-ja [Fayyad in Irani, 1992]. Za vrednotenje uspešnosti učenje smo uporabili standardne ocene, ki temeljijo na klasifikacijski točnosti, površini pod krivuljo ROC in graf, ki prikazuje krivuljo odziva (ang. *Lift chart*). Poglavje opisuje uporabljene metode učenja in izbire atributov, tehnike diskretizacije atributov in načine vrednotenja modelov.

4.1 Metodi strojnega učenja

Pri naši analizi podatkov smo uporabili dve nadzorovani metodi strojnega učenja: naivnega Bayesa in odločitveno drevo. Glavni razlogi za uporabo teh dveh metod so bili v tem, da sta obe med najprimernejšimi metodami za klasifikacijske probleme in napovedovanje. Poleg tega odločitvena drevesa generirajo pravila, ki jih je enostavno izvoziti v druga programska orodja ali prevesti v naravni jezik. Pravila lahko tudi interpretiramo oz. jih skušamo razumeti. To pa je navadno lahko zanimivo za strokovnjaka iz problemske domene, saj lahko iz drevesa razbere določene zakonitosti in strukturo. Metodi znata obravnavati tudi manjkajoče vrednosti v podatkih, kar je zelo pomembno, saj je zelo težko najti popolne podatke.

4.1.1 Naivni Bayesov klasifikator

Naivni Bayesov klasifikator predpostavlja pogojno neodvisnost vrednosti različnih atributov pri danem razredu. Osnovna formula Bayesovega pravila je [Kononenko,1997]:

$$P(r_k|V) = P(r_k)\prod_{i=1}^{a}\frac{P(r_k|v_i)}{P(r_k)} \tag{4.1}$$

Naloga učnega algoritma je s pomočjo učne množice podatkov oceniti apriorne verjetnosti razredov $P(r_k), k = 1...n_0$ in pogojne verjetnosti razredov r_k , $k = 1...n_0$ pri dani vrednosti v_i atributa A_i, $i = 1..a : P(r_k | v_i)$. Za ocenjevanje apriornih verjetnosti se navadno uporablja Laplaceov zakon zaporednosti [Kononenko,1997]:

$$P(r_k) = \frac{N_k + 1}{N + n_0} \tag{4.2}$$

kjer je N_k število učnih primerov iz razreda r_k in N število vseh učnih primerov. Za ocenjevane pogojnih verjetnosti se uporablja *m*-ocena [Kononenko,1997]:

$$P(r_k | v_i) = \frac{N_{k,i} + mP(r_k)}{N_i + m} \tag{4.3}$$

kjer je $N_{k,i}$ število učnih primerov iz razreda r_k in z vrednostjo *i*-tega atributa v_i ter N_i število vseh učnih primerov z vrednostjo *i*-tega atributa v_i [Kononenko,1997].

V naši konkretni nalogi smo za vrednost parametra *m*-ocene uporabili m=0, kar je povsem primerno, ker je podatkov veliko. Naivni Bayesov klasifikator, ki smo ga uporabili, uporablja samo diskretne atribute. Ker smo imeli v nalogi opravka z zveznimi atributi, smo jih morali diskretizirati. Pri tem smo uporabili metodo, opisano v poglavju 4.3.

4.1.2 Odločitvena drevesa

Odločitvena drevesa spadajo v kategorijo metod simboličnega učenja. Odločitveno drevo [Kononenko,1997] sestavljajo notranja vozlišča, ki predstavljajo atribute, veje, ki predstavljajo podmnožice vrednosti atributov in listi, ki ustrezajo razredom. Glede na algoritem se lahko vsako vozlišče razdeli na dve ali več vej. Vsaka pot v drevesu od korena do lista ustreza enemu odločitvenemu pravilu. Odločitvena drevesa, ki se uporabljajo za napovedovanje kategoričnih spremenljivk imenujemo klasifikacijska drevesa.

Proces učenja, katerega rezultat je odločitveno drevo, imenujemo indukcija in zahteva majhno število prehodov preko učne množice. Večina algoritmov za odločitvena drevesa vsebuje dve fazi: fazo gradnjo drevesa (ang. splitting), ki ji sledi faza rezanja (ang. prunning). Gradnja drevesa je iterativen proces, ki vključuje delitev

podatkov v vedno manjše podmnožice. Algoritmi za gradnjo dreves imajo ponavadi več pravil, kdaj se izgradnja odločitvenega drevesa ustavi. Pravila temeljijo na različnih pogojih, kot so npr. največja globina drevesa ali pa minimalno število elementov v vozlišču, ki so potrebna za delitev vozlišča. Večina orodij omogoča spreminjanje parametrov, ki določajo te pogoje. Nekateri algoritmi omogočajo tudi gradnjo drevesa do maksimalne globine. Taka drevesa omogočajo točno napoved za vse učne primere, razen navzkrižnih. Problem takih dreves je v tem, da zaradi prevelike prilagojenosti učnim podatkom ne dosežejo želenih rezultatov na drugih neodvisnih testnih množicah. Zato algoritmi, ki gradijo drevesa do maksimalne globine uporabljajo avtomatsko tudi rezanje drevesa.

Zgrajeno drevo lahko uporabimo za klasifikacijo novih primerov. Od korena potujemo po ustreznih vejah do lista. List vsebuje informacijo o številu učnih primerov iz posameznih razredov. Iz frekvenc učnih primerov se oceni verjetnostna porazdelitev razredov. Koristen podatek je tudi število vseh učnih primerov v listu, ki kaže na zanesljivost ocene verjetnostne porazdelitve razredov.

Danes v svetu obstaja veliko različnih modelov odločitvenih dreves. V naši nalogi smo uporabili odločitveno drevo C4.5 [Quinlan,1993].

4.2 Izbira podmnožice atributov

Kadar imamo veliko število atributov v opazovani množici, moramo ugotoviti, kateri atributi so resnično koristni za modeliranje cilja. Osnovna ideja je v tem, da nekatere metode strojnega učenja delujejo bolje, če se učenje izvaja s samo »najboljšimi« atributi. Obstaja več različnih načinov za izbiro koristnih atributov v strojnem učenju [Kononenko in Hong, 2002].

Pri danem konkretnem problemu odkrivanja znanj iz podatkov so lahko atributi koristni ali nekoristni. Kadar imamo veliko število atributov, so lahko celo nekateri koristni atributi ob prisotnosti drugih atributov odvečni. Atribut je lahko koristen sam zase ali pa kot element neke skrite podmnožice atributov. Izbira primerne podmnožice atributov iz množice razpoložljivih predstavlja velik izziv, ker z zmanjšanjem števila atributov, ne samo pohitrimo proces učenja, ampak tudi preprečimo, da bi učni algoritmi izgradili neustrezne modele, ki bi uporabljali nekoristne in odvečne atribute.

Izbira koristnih atributov ni možna brez ugotavljanja kvalitete posameznega atributa za izbrano nalogo modeliranja. Kvaliteta atributa naj bi izražala uporabno informacijo, ki jo nosi atribut. Obstajata dva pristopa za ocenjevanje kvalitete atributov:

- kvaliteta atributa se oceni brez upoštevanja ostalih atributov,
- kvaliteta atributa se oceni v kontekstu ostalih atributov.

Prvi pristop predpostavlja apriorno in pogojno neodvisnost atributov pri danem razredu. Njegova prednost je v tem, da je možno kvaliteto atributa lažje in hitreje izračunati. Drugi pristop je računsko precej bolj zahteven, a lahko, v nasprotju s prvim pristopom, odkrije visoke odvisnosti med atributi.

V diplomski nalogi smo uporabili metode za izbiro najboljših atributov za klasifikacijske probleme, ki jih podpira orodje Orange [Demšar in Zupan, 2002]. Implementiran je t.i. pristop filtriranja, kjer se pomembnost atributa oceni brez vedenja, katero metodo strojnega učenja bomo uporabili v nadaljevanju. Za potrebe našega konkretnega problema smo se odločili, da preverimo dve funkciji izračuna kvalitete atributa. Prva računa informacijski prispevek in predpostavlja neodvisnost atributov, druga pa uporablja algoritem Relief [Kononenko, 1997] oz. njegovo izpopolnjeno različico RelifF in učinkovito rešuje problem odvisnosti atributov. Osnovna ideja algoritma Relif je, da za vsak učni primer poišče najbližji primer iz istega razreda (najbližji zadetek) in najbližji primer iz nasprotnega razreda (najbližji pogrešek). Na ta način lahko oceni kvaliteto atributa glede na lokalne značilnosti razločevanja razredov. Lokalnost je tista, ki vključuje v oceno tudi ostale atribute. Z metodo Relief je možno ocenjevati tako diskretne, kot zvezne atribute, ki so med seboj močno odvisni. Izpopolnjena različica algoritma ReliefF [Kononenko, 1997], ki je implementirana v sistemu za strojno učenje Orange, vsebuje še razširitve za delo z manjkajočimi vrednostmi atributov, šumnimi podatki in večrazrednimi problemi.

4.3 Diskretizacija zveznih atributov

Mnogi algoritmi za strojno učenje lahko uporabljajo samo diskretne atribute. Da bi tudi tovrstne algoritme lahko uporabljali na poljubnih vhodnih podatkih, je potrebno vse zvezne atribute najprej diskretizirati v majhno število intervalov. Diskretizacija se lahko izvede vnaprej ali med samim učenjem.

V diplomski nalogi smo uporabili tudi metodo strojnega učenja naivni Bayes, ki zahteva na vhodu samo diskretne atribute. Podatki, na katerih smo izvajali poizkuse, vsebujejo več zveznih atributov (različni zneski, časovni termini, itd), ki jih je bilo potrebno najprej diskretizirati. Uporabili smo diskretizacijo Fayyad-a in Irani-ja [Fayyad in Irani, 1992], ki jo uporablja tudi Orange. Deluje tako, da s pristopom od zgoraj navzdol (ang. top-down) razdeli začetni obseg vrednosti atributa v manjše intervale. Delitev lokalno poveča informacijsko vsebino diskretnega atributa (ang. informativity) in se ustavi, ko je dolžina opisa (ang. description length) večja od pridobljene informacije. Število intervalov ne more biti vnaprej določeno. Zgodi se lahko, da je atribut diskretiziran v eno samo vrednost. V takem primeru ga obravnavamo kot redundantnega [Demšar in Zupan, 2002].

4.4 Ocenjevanje uspešnosti učenja

Uspešnost metod strojnega učenja smo ocenjevali z 10–kratnim prečnim preverjanjem (ang. *10-fold Cross Validation*), s katerim smo množico razpoložljivih učnih primerov razdelili na 10 približno enako močnih podmnožic. Učenje in ocenjevanje smo tako izvajali 10–krat. V *i*–tem izvajanju smo za ocenjevanje vzeli *i*–to podmnožico, za učenje pa preostalih devet. Poleg klasifikacijske točnosti smo uspešnost učenja merili še s površino pod krivuljo ROC in grafom, ki prikazuje krivuljo odziva (ang. *Lift chart*).

4.4.1 Klasifikacijska točnost

Uspešnost reševanja dvorazrednih klasifikacijskih problemov ocenjujemo s klasifikacijsko točnostjo (CA, ang. *Classification Accuracy*) [Kononenko, 1997]. Klasifikacijska točnost je definirana z:

$$T = \frac{N^{(p)}}{N} \times 100\% \qquad (4.4)$$

kjer je N število vseh možnih primerov problemov na danem področju in $N^{(p)}$ število pravilnih rešitev primerov. Klasifikacijsko točnost lahko označimo kot verjetnost, da bo naključno izbran primer pravilno klasificiran.

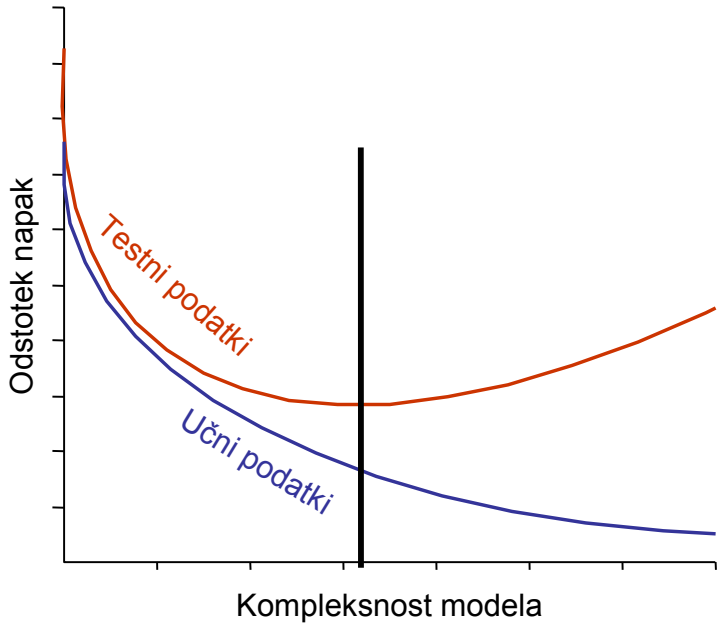
V realnih primerih je praktično nemogoče izračunati T, ker ponavadi ne poznamo vse pravilnih rešitev in tudi N je lahko zelo velik ali neskončen. Zato klasifikacijsko točnost ocenjujemo na neodvisni testni množici primerov. Definiramo jo kot:

$$T_t = \frac{N_t^{(p)}}{N_t} \times 100\% \tag{4.5}$$

kjer je N_t število vseh testnih primerov in $N_t^{(p)}$ število pravilnih rešitev testnih primerov. Testna množica mora biti neodvisna od učne množice primerov, ki jo algoritem uporablja pri učenju. Klasifikacijska točnost na N_u učnih primerih je podana z:

$$T_u = \frac{N_u^{(p)}}{N_u} \times 100\% \tag{4.6}$$

in predstavlja zgornjo mejo klasifikacijske točnosti. Teoretično je možno doseči T_u=100%, vendar se v takem primeru algoritem preveč prilagodi učni množici. Zato imajo nekateri algoritmi za strojno učenje posebne mehanizme, ki preprečujejo preveliko prileganje učni množici (ang. overfitting). Problem prevelikega prilagajanja je prikazan na sliki 4.1. Želimo dobiti model, ki je označen z debelo črto.



Slika 4.1 Problem prilagajanja učni množici

4.4.2 Cena napačne klasifikacije

Pri vrednotenju posameznih klasifikatorjev moramo velikokrat upoštevati tudi ceno napačnih klasifikacij (ang. *misclassification cost*). Največkrat se zgodi, da je napačna klasifikacija primerov iz nekaterih razredov hujša kot napačna klasifikacija primerov iz drugih. Optimizacija klasifikacijskih rezultatov brez upoštevanja cen napak lahko vodi v zelo nerazumljive rezultate.

Če so cene znane, jih je enostavno upoštevati v finančnih analizah pri procesu odločanja. V našem primeru smo se omejili na primer, ki ima samo dva razreda: Da in Ne. Tako dobimo štiri vrste možnih rezultatov za posamezno napoved: pravilno klasificirani pozitivni primeri in pravilno klasificirani negativni primeri predstavljajo pravilno klasifikacijo, nepravilno klasificirani pozitivni primeri (ko je napoved Da, v resnici pa je Ne) in nepravilno klasificirani negativni primeri (ko je napoved Ne, v resnici pa je Da) predstavljajo nepravilno klasifikacijo. Najlažje rezultate ocenjevanja nekega klasifikatorja za dvorazreden problem prikažemo v matriki (tabela 4.1).

		Napovedan razred (ang. predicted class)	
		DA	NE
Resnični Razred	DA	pravilno klasificirani pozitivni primeri	nepravilno klasificirani pozitivni primeri
	NE	Nepravilno klasificirani negativni primeri	pravilno klasificirani negativni primeri

Tabela 4.1 Matrika rezultatov za dvorazredni problem

Nepravilno klasificirani pozitivni primeri in nepravilno klasificirani negativni primeri imajo ponavadi različno ceno. Prav tako je tudi različna korist med pravilno klasificirani pozitivni primeri in pravilno klasificirani negativni primeri. Namesto klasifikacijske točnosti, ki ne upošteva cen napačne klasifikacije lahko, če cene poznamo, računamo povprečno ceno napačne klasifikacije. Povprečna cena napačne klasifikacije je definirana z:

$$C_t = \frac{\sum_{i,j}(C_{ij}N_t^{(ij)})}{N_t} \tag{4.7}$$

kjer je $N_t^{(ij)}$ število testnih primerov, ki pripadajo *i*-temu razredu in jih dana teorija klasificira v *j*-ti razred. C_{ij}, kjer je C_{ii}=0, i=1... M_R, predstavlja matriko cen napačnih klasifikacij. N_t pa je definiran z:

$$N_t = \sum_{i=1}^{M_R} N_t^{(ii)} \tag{4.8}$$

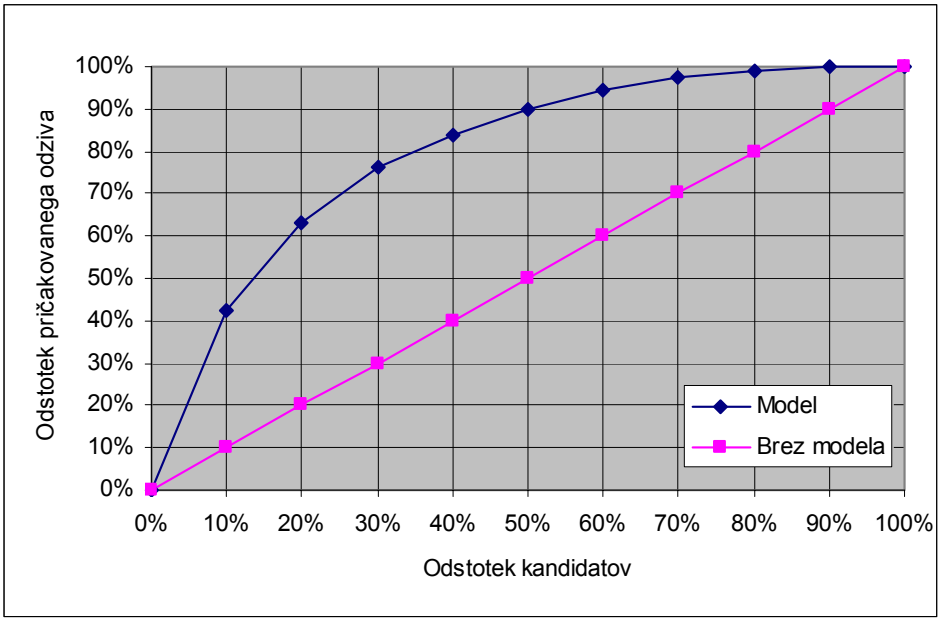
Cene napačne klasifikacije je najbolje upoštevati že pri izdelavi klasifikatorjev, kjer se cene upoštevajo med samim strojnim učenjem.

4.4.3 Krivulja odziva

Krivulja odziva se največ uporablja v oddelkih trženja, ker omogoča grafično predstavitev in merjenje uspešnosti nekega klasifikacijskega modela zgrajenega s strojnim učenjem. V grafu, ki prikazuje krivuljo odziva (LC, ang. *Lift Chart*) [Witten in Frank], se skrivata odgovora na dva zelo pogosta vprašanja, ko razvijamo različne modele za klasificiranje ali napovedovanje:

- Kateri model strojnega učenja med potencialnimi kandidati je najboljši za naš problem?
- Kateri od razvitih modelov omogoča uspešno klasificiranje ali napovedovanje tudi izven učne množice, s katero je bil razvit ?

LC prikazuje, kako velik delež populacije moramo izbrati za tržno akcijo, če želimo doseči željen odstotek od pričakovanega celotnega odziva. Na sliki 4.2 vidimo, da je za 62% odziv potrebno samo 20% pravilno izbrane množice strank. Kvocient 62/20 imenujemo »lift« in pove, da bomo dobili 3.1-krat večji odziv, če sodelujoče v akciji izberemo na podlagi rezultatov modela strojnega učenja namesto naključno.



Slika 4.2 Hipotetični »Lift chart«

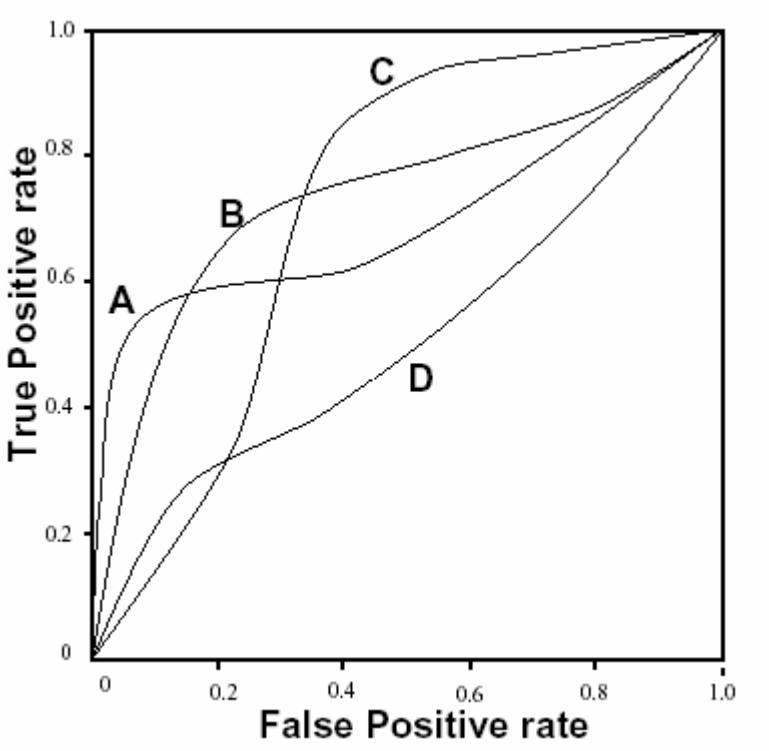
Hipotetični kumulativni LC je prikazan na sliki 4.2. Horizontalna os predstavlja velikost vzorca v odstotkih, ki je predmet tržne akcije. Vertikalna os predstavlja odstotek odziva na tržno akcijo glede na celoten odziv, ki bi ga dobili, če bi akcijo izvajali na vsej populaciji.. Graf LC ima dve krivulji. Prva krivulja, v grafu 4.2 označena z »Brez modela« predstavlja odstotek od celotnega odziva, ki ga dobimo pri naključno izbranem deležu od celotne populacije. Druga krivulja, v grafu označena z »Model« pa predstavlja odstotek od celotnega odziva, ki ga dobimo, če izberemo ciljno populacijo z uporabo modela strojnega učenja. Krivulja je tem boljša, bolj je blizu zgornjega levega kota. Najmanj kar pričakujemo od uspešnega modela je to, da je krivulja nad diagonalo. Graf LC ima nekaj tipičnih točk. Spodnja leva točka pomeni nič primerov v trži akciji in odzivom 0, zgornja desna točka pa pomeni tržno akcijo na vseh primerih in 100% napovedanim odzivom.

Pri merjenju uspešnosti modelov z grafom LC moramo upoštevati, da je vrednotenje uspešnosti posameznega modela potrebno gledati v odvisnosti od zahtevnosti posamezne napovedi. Veliko bolje se izkaže uporaba graf LC za primerjavo uspešnosti učnih algoritmov za isti problem.

4.4.4 Površina pod krivuljo ROC

Pri ocenjevanju in primerjavi uspešnosti učenja različnih klasifikatorjev na podlagi cene napačne klasifikacije je pomembno vedeti v kakšnih razmerah bodo le–ti uporabljeni. Distribucije razredov in cene napačne klasifikacije v večini realnih problemov ne moremo točno določiti oziroma se le–ti spreminjata [Provost in Fawcett, 1997].

Uporaba klasifikacijske točnosti včasih ni ustrezna, ker z njo privzete predpostavke o relativno uravnoteženi distribuciji razredov in ceni napačne klasifikacije redkokdaj držijo [Provost in Fawcett, 1997]. V naši problemski domeni je namreč cena napačne pozitivne klasifikacije (ang. false positive) lahko dosti manjša od cene napačne negativne klasifikacije (ang. false negative). Tudi distribucija razredov je zelo neuravnotežena.



Slika 4.3 Graf prostora ROC štirih klasifikatorjev [Provost in Fawcett, 1997].

Graf ROC izhaja iz teorije signalov, od tod tudi ime (ang. *Receiver Operating Characteristics*). ROC je grafični prikaz klasifikacijske zmožnosti enega ali več klasifikatorjev (slika 4.3), ki je neodvisen od distribucije razredov in cene napačne

klasifikacije. Prikazuje kompromis med pravilno pozitivno klasifikacijo (TP, ang. true positive) in napačno pozitivno klasifikacijo (FP, ang. false positive), katerih posteriorne verjetnosti približno ocenimo:

$$TP \approx \frac{\text{pravilno klasificirani pozitivni primeri}}{\text{vsi pozitivni primeri}}$$

$$FP \approx \frac{\text{nepravilno klasificirani pozitivni primeri}}{\text{vsi negativni primeri}}$$

Graf ROC ima nekaj tipičnih točk. Točka (0, 0) predstavlja strategijo, da nobenega primera ne klasificiramo za pozitivnega, točka (1, 1) pa strategijo, da vse primere klasificiramo za pozitivne. Diagonala x = y ustreza naključnemu uganjevanju razreda. Neformalno je ena točka boljša od druge, če je bolj severozahodno (večji TP, manjši FP), kar omogoča neformalno vizualno primerjavo klasifikatorjev [Provost in Fawcett, 1997].

Čeprav je prostor ROC uporabno vizualno orodje, nam sam po sebi ne olajša izbire najboljšega klasifikatorja, razen v primeru, ko je en klasifikator očitno boljši. Primer je slika 4.3, kjer ne moremo enostavno določiti najboljšega klasifikatorja. Vidimo le, da je klasifikator D nesporno najslabši. Odgovor na vprašanje o najboljšem klasifikatorju je odvisen od cen obeh tipov napačne klasifikacije in od distribucije razredov v pogojih, ko bodo klasifikatorji uporabljeni.

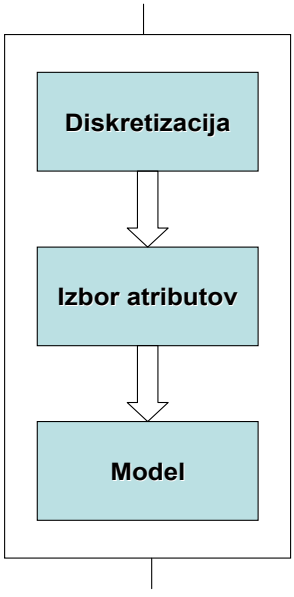
V našem konkretnem primeru smo želeli izbrati samo en klasifikator, ki bo dovolj dober v vseh pogojih delovanja, čeprav bo v mnogih primerih neoptimalen. Pri izboru smo si pomagali z eno izmed standardnih in razširjenih metod, ki računajo povprečno uspešnost klasifikatorja v celotnem prostoru ROC. Odločili smo se, da bomo uspešnost klasifikatorjev ocenjevali na podlagi celotne površine pod krivuljo ROC.

5 Poskusi in rezultati učenja

Primerjavo uspešnosti metod strojnega učenja smo izvajali na izbrani osnovni množici podatkov. Ocenjevanje uspešnosti modelov smo izvajali z 10-kratnim prečnim preverjanjem. Pri modelih, ki zahtevajo na vhodu diskretne atribute smo naredili diskretizacijo vseh zveznih atributov. Glede na veliko število atributov smo preverili tudi metodo izbora podmnožice atributov (FSS, ang. Feature Subset Selection). Pri vseh modelih strojnega učenja smo uporabili isto razbitje na učno in testno množico. Uspešnost učenja smo merili s:

- klasifikacijsko točnostjo,
- površino pod krivuljo ROC in
- krivuljo odziva.

Ocenjevali smo dve metodi strojnega učenja: naivni Bayes in odločitveno drevo. Za vsako od metod smo definirali poseben postopek izgradnje modela, ki pa je na najvišjem nivoju enak tistemu na sliki 5.1. Upoštevali smo dejstvo, da se modela razlikujeta v parametrih, ki jih lahko nastavljamo za vsakega od njiju in v uporabi zveznih atributov. Naivni Bayes, ki smo ga uporabili, zahteva na vhodu diskretne atribute, zato smo morali pred začetkom učenja vse zvezne atribute diskretizirati. V poglavju je opisan natančen postopek gradnje modelov in rezultati ovrednotenja modelov.



Slika 5.1 Postopek izgradnje modela strojnega učenja

5.1 Priprava učne in testne množice

Vhodne podatke smo naključno razdelili na dve disjuktivni množici v razmerju 70:30. Prva se imenuje učna množica in je namenjena strojnemu učenju modelov, ki jih želimo ocenjevati, druga pa predstavlja testno množico in se uporablja za preverjanje uspešnosti izgrajenih modelov. V tabeli 5.1 so podani podatki o velikosti in strukturi učne in testne množice. Pri vseh poizkusih smo uporabili isto učno in testno množico.

	Število zapisov	Delež	Število pozitivnih primerov	Delež pozitivnih primerov
Učna množica	56605	70 %	6072	10.72 %
Testna množica	24260	30 %	2613	10.79 %
Skupaj:	80868	100 %	8685	10.74 %

Tabela 5.1 Podatki o testni, učni in osnovni množici

5.2 Diskretizacija atributov

Uporabljena metoda strojnega učenja naivni Bayes zahteva na vhodu samo diskretne atribute. Uporabljena osnovna množica podatkov vsebuje 14 zveznih atributov, kot so npr. povprečno stanje na računu v preteklem letu, celoten promet v dobro v enem letu ali število mesecev od odprtja računa. Zato smo morali vsakega od zveznih atributov diskretizirati v majhno število intervalov. Uporabili smo diskretizacijo Fayyad-a in Irani-ja [Fayyad in Irani, 1992], ki jo uporablja tudi Orange [Demšar in Zupan, 2002] in uporablja entropijo in MDL, da poišče najboljše meje intervalov. V tabeli 5.2 so predstavljeni rezultati diskretizacije zveznih atributov učne množice.

Atribut	Intervali
D_STRANKA_MESECEV	≤52, (52, 86], (86, 277], (277, 296], >296
D_TR_MESECEV	≤44, (44, 294], >294
D_TR_STANJE	≤-419057, (-419057, -272656], (-272656, -209798], (-209798, -16167], (-16167, -52], (-52, 1717], (1717, 7397], (7397, 77631], (77631, 1711651], >1711651
D_TR_STANJE_AVG	≤-342508, (-342508, -207272], (-207272, -19785], (-19785, -91], (-91, 2474], (2474, 17082], (17082, 97963], (97963, 1971106], >1971106
D_TR_LIMIT	≤61000, (61000, 80000], (80000, 99000], (99000, 100000], (100000, 399000], (399000, 700000], >700000
D_TR_LIMIT_PREKORACEN_DNI	≤0, (0, 98], (98, 269], >269
D_TR_PROMET_DOBRO	≤62282, (62282, 1126253], (1126253, 1623052], (1623052, 1772092], (1772092, 2215153], (2215153, 3297221], (3297221, 5007137], >5007137

D_TR_PROMET_BREME	≤60295, (60295, 1400866], (1400866, 1710927], (1710927, 2176314], (2176314, 3005973], (3005973, 4828540], >4828540
D_TR_DOHODEK_AVG	≤12695, (12695, 111872], (111872, 135117], (135117, 173387], (173387, 246235], (246235, 334242], >334242
D_VISA_TRAJANJE	≤21, (21, 81], (81, 111], (111, 113], >113
D_VISA_PROMET	≤5250, (5250, 391739], (391739, 1170106], >1170106
D_TR_ST_PRILIVOV	≤16, (16, 26], (26, 28], (28, 29], (29, 39], (39, 41], (41, 44], (44, 46], (46, 48], (48, 51], >51
D_TR_ST_ODLIVOV	≤2, (2, 24], (24, 71], (71, 73], (73, 75], (75, 127], (127, 170], (170, 173], (173, 210], (210, 273], (273, 304], (304, 467], (467, 481], >481
D_TR_ST_BANKOMAT	≤0, (0, 13], (13, 47], (47, 100], (100, 256], >256

Tabela 5.2 Diskretizacija zveznih atributov učne množice

Analiza je pokazala, da so pri večini atributov intervali s poslovnega vidika smiselni. Problematična sta bila samo atributa za povprečno mesečno stanje (D_TR_STANJE_AVG) in letni promet z visa kartico (D_VISA_PROMET). Pri prvem atributu je bil sporen predzadnji interval, ker je prevelik. Pri drugem atributu pa so strokovnjaki za trženje ocenili, da bi moralo biti več intervalov.

5.3 Izbor podmnožice atributov (FSS)

Glede na veliko število atributov, ki jih ima osnovna množica, smo se odločili, da preverimo ali so vsi atributi resnično potrebni za gradnjo modelov strojnega učenja. Zanimala nas je pomembnost oz. kvaliteta posameznega atributa.

Za izbor podmnožice atributov (FSS, ang. Feature Subset Selection) smo uporabili algoritem ReliefF, ki rešuje tudi problem odvisnosti atributov. Privzeli smo standardne parametre za algoritem ReliefF (k=20, m=50). Pri tem pomeni k število najbližjih zadetkov in pogreškov, ki jih za vsak učni primer poišče algoritem ReliefF in m število iteracij v algoritmu ReliefF. Za vrednost praga, nad katerim mora biti vrednost za posamezni atribut, da je izbran, smo izbrali vrednost 0,01. Izdelali smo skripto (priloga 7.1) za orodje Orange, s katero smo preverili razlike v izračunanih ocenah kvalitete posameznih atributov brez in z uporabo metode izbora podmnožice atributov. Rezultati so zbrani v tabeli 5.3.

Lestvica atributov brez FSS	Lestvica atributov s FSS
0.072 D_TR_ST_ODLIVOV	0.077 D_TR_ST_ODLIVOV
0.056 D_TR_PROMET_DOBRO	0.071 D_TR_STANJE_AVG
0.053 D_TR_STANJE	0.069 D_TR_STANJE
0.051 D_TR_ST_BANKOMAT	0.065 D_STRANKA_MESECEV
0.049 D_TR_ST_PRILIVOV	0.063 D_TR_ST_BANKOMAT
0.044 D_TR_DOHODEK_AVG	0.053 BONITETA_MIN
0.038 D_TR_STANJE_AVG	0.051 D_TR_LIMIT
0.035 MESEC_ROJSTVA	0.049 D_TR_DOHODEK_AVG
0.034 D_TR_PROMET_BREME	0.047 AKTIVNIH_PRODUKTOV
0.034 STAROST	0.041 AKTIVNIH_RACUNOV
0.032 AKTIVNIH_PRODUKTOV	0.040 IMA_TR_TRAJNIK
0.032 TR_POOBLASCENCEV	0.040 BONITETA
0.032 AKTIVNIH_RACUNOV	0.036 MESEC_ROJSTVA
0.031 D_TR_LIMIT_PREKORACEN_DNI	0.034 STAROST
0.030 TR_BONITETA	0.030 ZAPOSLEN
0.029 D_TR_LIMIT	0.029 D_TR_PROMET_BREME
0.028 IMA_TR_TRAJNIK	0.029 D_TR_MESECEV
0.025 D_STRANKA_MESECEV	0.027 D_TR_ST_PRILIVOV
0.023 BONITETA	0.025 D_TR_PROMET_DOBRO
0.023 D_VISA_TRAJANJE	0.025 TR_BONITETA
0.023 IMA_DR	0.025 TR_POOBLASCENCEV
0.021 ZAPOSLEN	0.025 D_TR_LIMIT_PREKORACEN_DNI
0.019 VISA_ST_KARTIC	0.025 D_VISA_TRAJANJE
0.018 IMA_ZR	0.024 BONITETA_MAX
0.018 D_VISA_PROMET	0.021 IMA_ZR
0.015 IMA_VISA	0.016 D_VISA_PROMET
0.014 BONITETA_MIN	0.007 IMA_VISA
0.014 D_TR_MESECEV	0.004 IMA_HV
0.012 SPOL	
0.012 IMA_VISA_KLASIK	
0.011 BONITETA_MAX	
0.010 IMA_DV	
0.010 IMA_KRED	
0.009 IMA_VARKNJ	
0.008 IMA_TOLDEP	
0.006 IMA_HV	
0.006 IMA_DEVDEP	
0.005 IMA_VISA_ELEKTRON	
0.002 IMA_NSVS	
0.002 IMA_RENVAR	
0.002 IMA_NAMVAR	
0.001 IMA_VISA_GOLD	
0.001 IMA_VISA_KREDITNA	
0.001 IMA_VISA_POSLOVNA	
0.000 IMA_IBI	

Tabela 5.3 Ocene atributov brez in po izboru atributov

Ugotovili smo, da se število atributov po izboru zmanjša iz 45 na 28, zelo pa se je spremenila tudi pomembnost oz. rang posameznih atributov. Če primerjamo rezultate v tabeli 5.3 vidimo, da sta med prvimi desetimi atributi samo prvi in tretji na istih mestih in da je samo pet atributov med prvimi desetimi istih na obeh lestvicah.

Skupna analiza s strokovnjaki za trženje prvih desetih atributov na obeh lestvicah v tabeli 5.3 je pokazala, da je večina atributov pričakovanih. Na prvi lestvici je bil nepričakovano visoko mesec rojstva. Preverili smo distribucijo razredov po mesecih in ugotovili, da je skoraj enaka za vse mesece. Razloga za tako visoko mesto nismo znali pojasniti. Na drugi lestvici je bil nepričakovano visoko atribut za minimalno boniteto (BONITETA_MIN), dokaj nizko pa se je odrezal atribut za letni promet v dobro (D_TR_PROMET_DOBRO), ki bi po mnenju tržnikov moral biti precej višje.

Metodo izbora podmnožice atributov smo uporabili pri gradnji enega od klasifikacijskih modelov, ki uporablja učni algoritem naivni Bayes.

5.4 Uporabljene metode strojnega učenja

Za izdelavo modela za napovedovanje potencialnega zanimanja bančne stranke za produkt Abanet smo preverili več različnih klasifikacijskih modelov, ki uporabljajo dve različni metodi strojnega učenja: naivni Bayes in odločitveno drevo. Izdelali smo naslednje štiri klasifikatorje:

- naivnega Bayesa s privzetimi vrednostmi parametrov in atributi, ki smo jih disktretizirali s pomočjo metode, ki uporablja entropijo in MDL.
- naivnega Bayesa s privzetimi vrednostmi parametrov in atributi, ki smo jih disktretizirali s pomočjo metode, ki uporablja entropijo in MDL. Poleg tega smo uporabili samo izbrano podmnožico atributov, ki smo jo dobili s pomočjo algoritma ReliefF.
- odločitveno drevo C4.5 s privzetimi vrednostmi parametrov.
- odločitveno drevo C4.5 z vsemi privzetimi vrednostmi parametrov, razen parametra *m*, ki določa minimalno število učnih primerov v vozišču za zanesljivo nadaljevanje gradnje drevesa. V našem primeru smo izbrali, glede na velikost množice, vrednost *m*=50.

5.5 Ocenjevanje uspešnosti učenja

Vse zgoraj definirane modele smo ocenili po kriterijih klasifikacijske točnosti, površine pod krivuljo ROC (AUC) in krivulje odziva. Najprej smo ocenjevanje vseh metod strojnega učenja izvajali z 10-kratnim prečnim preverjanjem na učni množici, nato pa smo izbrani model še preverili na testni množici.

5.5.1 Primerjava metod strojnega učenja z 10-kratnim prečnim preverjanjem

Metode strojnega učenja smo ocenjevali z metodo 10-kratnega prečnega preverjanja na učni množici. Kvaliteta te metode je v tem, da se navadno z njo izognemo problemu prevelikega prileganja učnega algoritma učnim podatkov (ang. *overfitting*). Učno množico smo razdelili na 10 približno enako močnih podmnožic. V vsakem izvajanju smo ocenjevali i-to podmnožico, za učenje pa smo uporabili preostalih devet. Pri gradnji obeh modelov naivnega Bayesa smo znotraj vsakega izvajanja naredili še

diskretizacijo zveznih atributov, pri zadnji različici pa še izbor podmnožice atributov. Učenje in ocenjevanje smo tako izvajali 10-krat. Ocenjevali smo klasifikacijsko točnost in površino pod krivuljo ROC. Rezultati so v tabeli 5.4.

Učni algoritem	Klasifikacijska točnost	Površina pod krivuljo ROC
Major	0.893	0.500
Naivni Bayes	0.780	0.797
Naivni Bayes +FSS	0.831	0.770
Odločitveno drevo	0.865	0.595
Odločitveno drevo (m=50)	**0.895**	**0.799**

Tabela 5.4 Primerjava uspešnosti učnih algoritmov

Najboljše rezultate je dosegel učni algoritem odločitvenega drevesa z *m*=50. Rezultat, ki ga je dosegel pri izračunu površine pod krivuljo ROC je dober. Ta algoritem smo tudi izbrali in z njim s pomočjo sistema za strojno učenje Orange in učne množice izdelali model odločitvenega drevesa za napovedovanje.

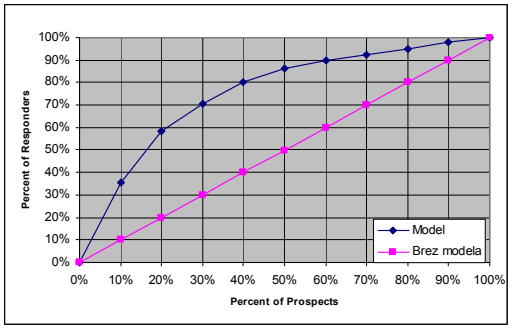
Zgrajeno odločitveno drevo je imelo 483 vozlišč pred rezanjem in 83 po rezanju. Obrezano drevo je predstavljeno na sliki 5.2. Pri analizi zgrajenega odločitvenega drevesa s poslovnega vidika smo ugotovili, da so izbrani atributi na najvišjih nivojih smiselni. Edina izjema je atribut IMA_VISA_GOLD, ki predstavlja imetje zelo redkega produkta. Razloga nismo znali pojasniti, zato smo predlagali rešitev ponovne gradnje odločitvenega drevesa brez uporabe tega atributa. Če primerjamo rezultate ocenjevanja atributov z algoritmom ReliefF (tabela 5.3, Lestvica atributov brez FSS) in atribute na zgornjih nivojih drevesa, ugotovimo, da se večina atributov ujema. V korenu drevesa je atribut TR_ST_ODLIVOV, ki je tudi najbolje ocenjeni atribut po algoritmu ReliefF. Na tretjem nivoju v drevesu imamo atribut TR_PROMET_DOBRO, ki je drugi najbolje ocenjeni atribut po metodi RelieF.

```
TR_ST_ODLIVOV ≤ 173 : 0 (42502.0/2700.3)
TR_ST_ODLIVOV > 173 :
|   IMA_VISA_GOLD = 0:
|   |   TR_PROMET_DOBRO ≤ 2.69218e+06 : 0 (7715.0/1135.0)
|   |   TR_PROMET_DOBRO > 2.69218e+06 :
|   |   |   ZAPOSLEN = 0: 0 (378.0/57.3)
|   |   |   ZAPOSLEN = 1:
|   |   |   |   TR_ST_ODLIVOV ≤ 273 :
|   |   |   |   |   TR_ST_BANKOMAT > 94 : 0 (1872.0/399.7)
|   |   |   |   |   TR_ST_BANKOMAT ≤ 94 :
|   |   |   |   |   |   IMA_NSVS = 0:
|   |   |   |   |   |   |   IMA_TR_TRAJNIK = 1:
|   |   |   |   |   |   |   |   IMA_DV = 1: 0 (314.5/91.9)
|   |   |   |   |   |   |   |   IMA_DV = 0:
|   |   |   |   |   |   |   |   |   AKTIVNIH_PRODUKTOV ≤ 2 : 0 (267.5/76.5)
|   |   |   |   |   |   |   |   |   AKTIVNIH_PRODUKTOV > 2 :
|   |   |   |   |   |   |   |   |   |   AKTIVNIH_RACUNOV > 6 : 1 (51.9/23.9)
|   |   |   |   |   |   |   |   |   |   AKTIVNIH_RACUNOV ≤ 6 :
|   |   |   |   |   |   |   |   |   |   |   SPOL = Z: 0 (256.6/100.7)
|   |   |   |   |   |   |   |   |   |   |   SPOL = M:
|   |   |   |   |   |   |   |   |   |   |   |   AKTIVNIH_RACUNOV ≤ 3 : 0 (100.1/43.2)
|   |   |   |   |   |   |   |   |   |   |   |   AKTIVNIH_RACUNOV > 3 :[S1]
|   |   |   |   |   |   |   IMA_TR_TRAJNIK = 0:
|   |   |   |   |   |   |   |   AKTIVNIH_PRODUKTOV ≤ 3 : 0 (100.5/48.8)
|   |   |   |   |   |   |   |   AKTIVNIH_PRODUKTOV > 3 : 1 (50.6/17.7)
|   |   |   |   |   |   IMA_NSVS = 1:
|   |   |   |   |   |   |   AKTIVNIH_RACUNOV ≤ 5 : 0 (58.9/31.2)
|   |   |   |   |   |   |   AKTIVNIH_RACUNOV > 5 : 1 (54.7/18.9)
|   |   |   |   TR_ST_ODLIVOV > 273 :
|   |   |   |   |   TR_ST_BANKOMAT ≤ 182 :
|   |   |   |   |   |   AKTIVNIH_RACUNOV ≤ 2 :
|   |   |   |   |   |   |   TR_BONITETA = A: 0 (38.6/20.3)
|   |   |   |   |   |   |   TR_BONITETA = AA: 0 (143.4/49.4)
|   |   |   |   |   |   |   TR_BONITETA = AAA: 1 (73.2/34.7)
|   |   |   |   |   |   |   TR_BONITETA = B: 1 (6.0/3.4)
|   |   |   |   |   |   |   TR_BONITETA = E: 0 (0.0)
|   |   |   |   |   |   |   TR_BONITETA = C: 0 (0.0)
|   |   |   |   |   |   |   TR_BONITETA = D: 0 (0.0)
|   |   |   |   |   |   AKTIVNIH_RACUNOV > 2 :
|   |   |   |   |   |   |   IMA_TR_TRAJNIK = 0: 1 (62.8/19.9)
|   |   |   |   |   |   |   IMA_TR_TRAJNIK = 1:
|   |   |   |   |   |   |   |   IMA_NSVS = 1: 1 (158.8/55.5)
|   |   |   |   |   |   |   |   IMA_NSVS = 0:
|   |   |   |   |   |   |   |   |   TR_ST_PRILIVOV ≤ 36 :
|   |   |   |   |   |   |   |   |   |   BONITETA_MAX = AAA: 0 (115.8/44.0)
|   |   |   |   |   |   |   |   |   |   BONITETA_MAX = A: 1 (28.9/12.3)
|   |   |   |   |   |   |   |   |   |   BONITETA_MAX = B: 0 (0.0)
|   |   |   |   |   |   |   |   |   |   BONITETA_MAX = E: 0 (0.0)
|   |   |   |   |   |   |   |   |   |   BONITETA_MAX = D: 0 (0.0)
|   |   |   |   |   |   |   |   |   |   BONITETA_MAX = C: 0 (0.0)
|   |   |   |   |   |   |   |   |   |   BONITETA_MAX = AA:
|   |   |   |   |   |   |   |   |   |   |   TR_ST_BANKOMAT ≤ 118 : 1 (56.7/28.0)
|   |   |   |   |   |   |   |   |   |   |   TR_ST_BANKOMAT > 118 : 0 (50.9/19.9)
|   |   |   |   |   |   |   |   |   TR_ST_PRILIVOV > 36 :
|   |   |   |   |   |   |   |   |   |   SPOL = M: 1 (333.0/123.1)
|   |   |   |   |   |   |   |   |   |   SPOL = Z:
|   |   |   |   |   |   |   |   |   |   |   BONITETA_MIN = AA: 0 (89.6/47.3)
|   |   |   |   |   |   |   |   |   |   |   BONITETA_MIN = A: 1 (97.4/45.4)
|   |   |   |   |   |   |   |   |   |   |   BONITETA_MIN = AAA: 0 (44.4/19.7)
|   |   |   |   |   |   |   |   |   |   |   BONITETA_MIN = C: 1 (0.0)
|   |   |   |   |   |   |   |   |   |   |   BONITETA_MIN = E: 1 (5.0/1.2)
|   |   |   |   |   |   |   |   |   |   |   BONITETA_MIN = D: 1 (0.0)
|   |   |   |   |   |   |   |   |   |   |   BONITETA_MIN = B:
|   |   |   |   |   |   |   |   |   |   |   |   TR_MESECEV ≤ 111 : 1 (52.9/19.9)
|   |   |   |   |   |   |   |   |   |   |   |   TR_MESECEV > 111 : 0 (50.5/25.4)
|   |   |   |   |   TR_ST_BANKOMAT > 182 :
|   |   |   |   |   |   AKTIVNIH_PRODUKTOV ≤ 5 : 0 (818.4/246.9)
|   |   |   |   |   |   AKTIVNIH_PRODUKTOV > 5 : 1 (58.0/23.1)
|   IMA_VISA_GOLD = 1:
|   |   TR_ST_PRILIVOV ≤ 40 :
|   |   |   BONITETA = AA: 1 (53.0/25.0)
|   |   |   BONITETA = A: 0 (30.0/17.4)
|   |   |   BONITETA = AAA: 0 (88.0/31.6)
|   |   |   BONITETA = B: 0 (3.0/1.1)
|   |   |   BONITETA = E: 0 (1.0/0.8)
|   |   |   BONITETA = C: 0 (0.0)
|   |   |   BONITETA = D: 0 (0.0)
|   |   TR_ST_PRILIVOV > 40 :
|   |   |   TR_DOHODEK_AVG ≤ 505961 : 1 (231.0/68.3)
|   |   |   TR_DOHODEK_AVG > 505961 : 0 (50.0/24.9)
```

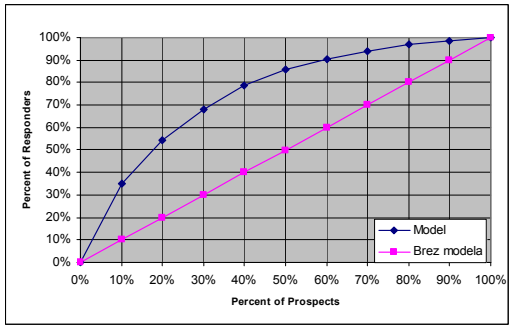
Slika 5.2 Generirano drevo

5.5.2 Primerjava metod strojnega učenja na učni množici

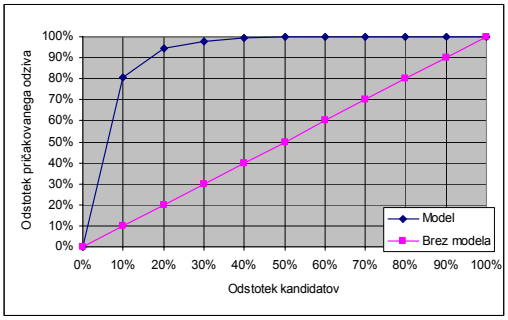
V primeru, da izbiramo učni algoritem s pomočjo 10-kratnega prečnega preverjanja, navadno ne izvajamo še enkrat primerjanja uspešnosti algoritmov strojnega učenja na učni množici. Mi smo ga izvedli zato, da preverimo grafično tehniko prikazovanja uspešnosti modelov strojnega učenja, ki se uporablja v oddelkih trženja. Na slikah 5.3, 5.4, 5.5 in 5.6 so prikazani grafi, ki prikazujejo krivuljo odziva (LC, ang. *Lift Chart*), ki prikazuje učinkovitost posameznih modelov na učni množici. Med njimi izstopa graf na sliki 5.5. Če bi upoštevali kriterije po katerih merimo uspešnost, bi bil ta model najboljši. Vendar pa se navadno za takim grafom skriva povezanost med vhodnimi podatki in napovedanim izhodom. V našem primeru se je model odločitvenega drevesa popolnoma prilagodil učnim podatkom. Isti model se je po preverjanju na testni množici izkazal za najslabšega.



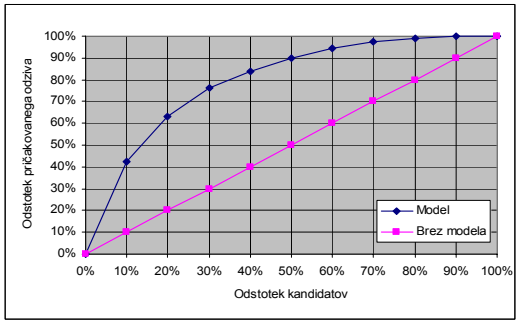
Slika 5.3 LC naivnega Bayesa



Slika 5.4 LC naivnega Bayes s FSS



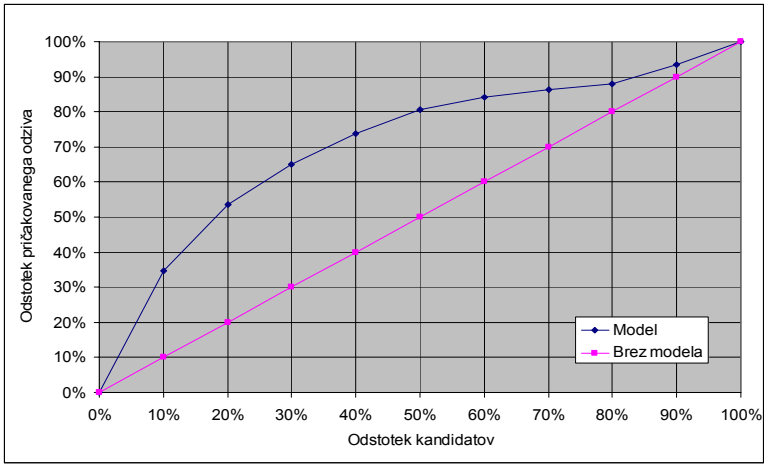
Slika 5.5 LC odločitvenega drevesa



Slika 5.6 LC odločitvenega drevesa z m=50

5.5.3 Vrednotenje izbranega modela na testni množici

Izbrani model odločitvenega drevesa z *m*=50 smo preverili še na testni množici podatkov. S tem smo želeli še enkrat preverili obnašanje modela na neznanih podatkih. Model je dosegel klasifikacijsko točnost 0.891 in površino pod krivuljo ROC 0.726, kar je v primerjavi z rezultati, ki jih je dosegel pri 10-kratnem prečnem preverjanju (klasifikacijska točnost 0,895 in površina pod krivuljo ROC 0.799), zelo dobro. Učinkovitost modela, podana z grafom LC, je prikazana na sliki 5.7. Če primerjamo krivuljo odziva na sliki 5.7 s krivuljo odziva na sliki 5.6 vidimo, da je učinkovitost modela na testni množici samo malo slabša od rezultatov na učni množici.



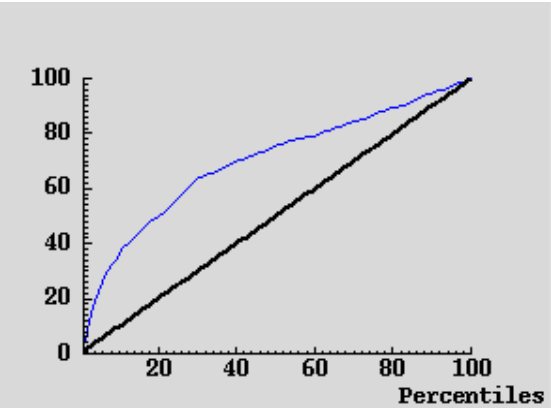
Slika 5.7 LC odločitvenega drevesa z *m*=50 na testni množici

Eden od problemov, ki smo ga zaznali pri analizi uporabljenih algoritmov strojnega učenja je bil ta, da noben algoritem ne upošteva pri gradnji modela cen za napačno klasifikacijo. Predpostavili smo, da bi bil izbran algoritem še uspešnejši, če bi upošteval cene napačnih klasifikacij. S strokovnjaki v banki smo definirali približne cene za oba tipa napačnih klasifikacij (tabela 5.5). Vidi se, da ceni napačnih klasifikacij še zdaleč nista enaki.

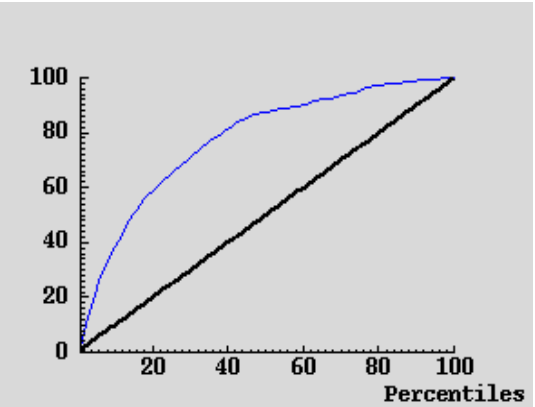
		Napovedan razred	
		Ima produkt	Nima produkta
Resnični Razred	Ima produkt	0	15
	Nima produkta	2	0

Tabela 5.5 Matrika cen napačnih klasifikacij

Hipotezo smo preveri s pomočjo orodja SPSS Clementine, ki omogoča pri gradnji odločitvenega drevesa tudi upoštevanje cen napačnih klasifikacij. Uporabili smo obstoječo učno in testno množico podatkov in enake parametre, kot jih je imel naš izbrani model. Rezultati primerjave na testni množici so v obliki grafa LC na slikah 5.8 in 5.9. Naša hipoteza se je izkazala za pravilno. Na sliki 5.9 se vidi pri modelu, ki upošteva cene napačne klasifikacije, precej bolj strma krivulja, ki pomeni boljši model.



Slika 5.8 LC za odločitveno drevo brez upoštevanja cen napačnih klasifikacij



Slika 5.9 LC za odločitveno drevo z upoštevanjem cen napačnih klasifikacij

5.6 Poslovna vrednost modela

Z izbranim model strojnega učenja si bo banka pomagala pri pripravi ciljno usmerjenih tržnih akcij za produkt Abanet. Celotno množico potencialnih kandidatov za tržno akcijo bo lahko z modelom ocenila, jih razvrstila glede na doseženo napoved in med njimi izbrala samo tiste, pri katerih je napovedana verjetnost, da je produkt Abanet za njih zanimiv, nad določeno mejo.

Poslovna vrednost uporabljenega modela je najbolje razvidna iz slike 5.6. Graf LC nam pove, da če npr. banka izbere z modelom samo 10% celotne množice potencialnih kandidatov za tržno akcijo, lahko pričakuje odziv 35% od celotnega pričakovanega odziva. To pa pomeni 3.5-krat več, kot če bi banka izbrala 10% kandidatov naključno.

Drugo uporabno vrednost za banko pa predstavljajo odločitvena drevesa. Odločitveno drevo predstavi rezultat učenja kot drevo oz. množico pravil (slika 5.7), ki so enostavno razumljiva poznavalcu problemske domene (v banki je to strokovnjak za trženje). Odločitveno drevo razdeli eno veliko množico vhodnih podatkov v veliko število manjših. Vsi primeri v danem listu drevesa ustrezajo istemu pravilu. Verjetnost pozitivnega primera v vsakem listu se uporablja kot ocena za vsak primer, ki je dosegel ta list. Za vse primere v listu velja, da imajo precej stvari skupnih, ne glede na to, kateremu razredu pripadajo. Če je v nekem listu npr. 50 primerov, med katerimi jih je 40 v razredu 1 in 10 v razredu 0, prestavlja teh deset zelo dobre kandidate za tržno akcijo, ker imajo veliko skupnih elementov z preostalimi štiridesetimi, pri katerih je bila uspešna ena od preteklih tržnih akcij.

6 Zaključek

V uvodu diplomskega dela smo opisali izzive, s katerimi se banke danes soočajo in možnosti uporabe metod za odkrivanje znanja iz podatkov za namene bančnega trženja. Diplomska naloga je bila usmerjena k rešitvi konkretnega primera v oddelku trženja v Abanki. Preveriti smo želeli ali je možno s strojnim učenjem izdelati tak model, ki bo uspešno prepoznaval imetnike bančne storitve Abanet. S pomočjo izdelanega modela bi potem poiskali tiste bančne stranke, ki tega produkta še nimajo, je pa njihovo obnašanje zelo podobno tistim, ki ga imajo. Najboljše kandidate bo banka uporabila v ciljno usmerjeni tržni akciji.

Uporabili smo eno samo osnovno množico podatkov, ki smo jo pridobili iz podatkovnega skladišča. Z uporabo podatkovnega skladišča smo se izognili problemom dostopa in integracije podatkov iz različnih sistemov. Glede na veliko količino podatkov smo vse potrebne preračune za posamezne atribute izvedli v času polnjenja podatkov iz podatkovnega skladišča.

Pri ocenjevanju in primerjavi uspešnosti metod strojnega učenja smo prišli do zanimivih ugotovitev. Najboljše rezultate smo dosegli z modelom odločitvenega drevesa. Na osnovi primerjanja krivulj odziva se je izbrani model izkazal za dokaj stabilnega. Rezultati na testni množici so bili le za malenkost slabši od rezultatov na učni množici. Ugotovili smo, da bi se dalo doseči še boljše rezultate, če bi algoritem za gradnjo odločitvenega drevesa upošteval še cene napačnih klasifikacij.

Glavni prispevek diplomskega dela s tehničnega vidika je implementacija postopkov priprave podatkov in polnjenja osnovne množice podatkov iz podatkovnega skladišča ter izdelava skript za ocenjevanje in preverjanje različnih modelov strojnega učenja. Vse rezultate diplomske naloge bodo lahko tržniki v banki uporabili pri svojem delu.

Nadaljnje delo vključuje več možnosti. Na podlagi rešitve, ki smo jo izdelali za produkt Abanet je možno razviti podobne rešitve tudi za ostale bančne produkte. Če vsakemu produktu določimo še ustrezno utež, lahko izdelamo rešitev, ki bo za vsako stranko vrnila produkt, ki je v določenem trenutku najbolj zanimiv zanjo. Glede na to, da je odločitveno drevo možno izvozit v druga programska okolja, lahko pridobljena znanja iz modelov vključimo tudi v že obstoječe bančne rešitve.

Literatura

Berry MJA and Linoff GS. Mastering Data Mining. Wiley, 2000.

Berry MJA and Linoff GS. Data Mining Techniques.Wiley, 1997.

Berson A, Smith S, Thearling K. Building Data Mining Applications for CRM. Osborne McGraw-Hill, 1999.

Bounsaythip C and Rinta-Runshala E. Overview of Data Mining for Customer Behavior Modeling. VTT, 2001.

Chapman P, Clinton J, Keber R, Khabaza T, Reinartz T, Shererer C and Weith R. CRISP-DM 1.0. NCR Systems , DeimlerChrysler AG, SPSS Inc., OHRA, 1999.

Coppock DS. Data Modeling and Mining: Why Lift?. http://www.dmreview.com, 2002.

Demšar J in Zupan B. Programski paket za strojno učenje Orange. http://magix.fri.uni-lj.si/orange, 2002.

Fayyad UM and Irani KB. The attribute selection problem in decision tree generation. In Proc. of AAAI-92, pages 104–110, San Jose, CA, 1992.

Kimball R, Reeves L, Ross M, Thormwaite W. The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses. John Wiley & Sons, 1998.

Kimball R, Ross M. The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (Second Edition). John Wiley & Sons, 2002 .

Kohavi R and John G: Wrappers for Feature Subset Selection, Artificial Intelligence, 97 (1-2), pages 273-324, 1996.

Kononenko I. Strojno učenje. Založba FE in FRI, Ljubljana, 1997.

Kononenko I and Hong SJ. Atribute selection for modeling. http://www.research.ibm.com/dar/papers, 1997.

Provost F in Fawcett T. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), 1977.

Scoot RI, Svinterrikou S, Tjortjis C, Keane JA. Experience of using Data Mining in a Banking Application. http://citeseer.nj.nec.com/95606.html, 1998.

Spiegel MR. Probability and Statistics, Second Edition. Schaum's, 2000.

Thearling K. Scoring Your Customers, http://www.thearling.com, 2002.

Thearling K. Data Mining and Customer Relationships, http://www.thearling.com, 2002.

Witten IH and Frank E. Data mining. Academic Press, 2000.

Two Crows Corporation. Introduction to Data Mining and Knowledge Discovery, Third Edition. http://www.twocrows.com, 2001.

Priloge

Priloženih je večina skript, ki smo jih napisali in uporabili pri delu s programskim paketom za strojno učenje Orange (http://magix.fri.uni-lj.si/orange) in skripte SQL, ki smo jih napisali in uporabili za delo s podatkovno zbirko Oracle, v kateri se nahaja podatkovno skladišče.

A. Skripte za Orange

Osnovna analiza podatkov

Program prebere osnovno množico podatkov in vrne število razredov opazovanega atributa, število vseh atributov, število zveznih in diskretnih atributov in distribucijo razredov.

```
import orange
data = orange.ExampleTable("d:\dip\del100")
print "Razredov:", len(data.domain.classVar.values)
print "Vseh atributov:", len(data.domain.attributes), "\n",

# preštej število zveznih in diskretnih atributov
ncont=0; ndisc=0
for a in data.domain.attributes:
    if a.varType == orange.VarTypes.Discrete:
        ndisc = ndisc + 1
    else:
        ncont = ncont + 1
print "   ", ncont, "zveznih"
print "   ", ndisc, "diskretnih"

# pridobi distribucijo razredov
c = [0] * len(data.domain.classVar.values)
for e in data:
    c[int(e.getclass())] += 1
print "velikost množice: ", len(data), "vseh\n",
r = [0.] * len(c)
for i in range(len(c)):
    r[i] = c[i]*100./len(data)
for i in range(len(data.domain.classVar.values)):
    print "      %d(%4.1f%s) v razredu %s\n" % (c[i], r[i], '%',
data.domain.classVar.values[i]),
print
```

Preverjanje manjkajočih vrednosti za posamezen atribut

Program prebere osnovno množico podatkov in za vsak atribut izpiše odstotek manjkajočih vrednosti.

```
import orange
data = orange.ExampleTable("d:\dip\del100")

natt = len(data.domain.attributes)
missing = [0.] * natt
for i in data:
    for j in range(natt):
        if i[j].isSpecial():
            missing[j] += 1
missing = map(lambda x:x/len(data)*100., missing)

print "Odstotek manjkajocih vrednosti za atribut:"
atts = data.domain.attributes
for i in range(natt):
    print "%5.1f%s%s" % (missing[i], '%,', atts[i].name)
```

Diskretizacija atributov

Program izvede disktretizacijo z entropijo vseh zveznih atributov in za vsak zvezni atribut izpiše intervale in njihove meje.

```
import orange

def show_values(data, heading):
    print heading
    for a in data.domain.attributes:
        print "%s: %s" % (a.name, reduce(lambda x,y: x+', '+y, [i for i in
a.values]))

data = orange.ExampleTable("d:\dip\del70.tab")

data_ent = orange.Preprocessor_discretize(data,
method=orange.EntropyDiscretization())
show_values(data_ent, "Diskretizacija z entropijo in MDL")
print
```

Pomembnost atributov

Program izračuna pomembnost posameznega atributa v učni množici po metodah Info in RefiefF (k=20, m=50). Učno množico najprej diskretizirano.

```
import orange
LearnSet = orange.ExampleTable("d:\dip\del70")
data = orange.Preprocessor_discretize(LearnSet,
method=orange.EntropyDiscretization())

info = orange.MeasureAttribute_info()
relief = orange.MeasureAttribute_relief(k=20, m=50)

# izračunaj in zapiši rezultate še v datoteko
f=open('d:\dip\dip_pa.txt', 'w')
print "%15s %6s %6s" % ("name", "info", "Relief")
f.write("name, info, Relief,,,\n")
for attr in data.domain.attributes:
```

```
    print "%15s %6.3f %6.3f" % (attr.name, info(attr, data)[0], relief(attr,
data)[0])
    f.write(attr.name +","+str(info(attr, data)[0])+","+str(relief(attr,
data)[0])+"\n")
f.close()
```

Izbira podmnožice atributov z uporabo metode ReliefF

Program izpiše vrednosti za kvaliteto atributov izračunano s pomočjo algoritma ReliefF pred in po FSS s pragom 0.01. Vse zvezne atribute najprej diskretiziramo.

```
import orange, orngFSS

def izpisi_pomembnost_atributa(data):
  m = orngFSS.attMeasure(data)
  for i in m:
    print "%5.3f %s" % (i[1],i[0])

LearnSet = orange.ExampleTable("d:\dip\del70")
data = orange.Preprocessor_discretize(LearnSet,
method=orange.EntropyDiscretization())
print "Brez FSS:";
izpisi_pomembnost_atributa(data)

marg = 0.01
filter = orngFSS.FilterRelief(margin=marg)
ndata = filter(data)
print "\nZ FSS s pragom: %5.3f:" % marg
izpisi_pomembnost_atributa(ndata)
```

Ocenjevanje klasifikatorjev

Program najprej razdeli osnovno množico podatkov na učno in testno množico (razmerje 70:30). Na učni množici izvede z 10-kratnim prečnim preverjanjem ocenjevanje izbranih modelov strojnega učenja. Na testni množici potem modele še preveri. Za vsak model izpiše klasifikacijsko točnost in površino pod krivuljo ROC.

```
import orange, orngDisc, orngEval, orngFSS

dataall = orange.ExampleTable("d:\dip\del100.tab")
# razdeli vse podatke na ucno in testno množico v razmerju 70:30
selection = orange.MakeRandomIndices2(dataall,0.7)
data = dataall.select(selection, 0)
testdata = dataall.select(selection, 1)

major = orange.MajorityLearner()
major.name = 'majority'
bayes = orange.BayesLearner()
dBayes = orngDisc.DiscretizedLearner(bayes, name='disc bayes')
fss = orngFSS.FilterAttsAboveThresh(threshold=0.01)
fBayes = orngFSS.FilteredLearner(dBayes, filter=fss, name='bayes+fss')
c45 = orange.C45Learner()
c45.name = 'c4.5'
c45M = orange.C45Learner()
c45M.name = 'c4.5m50'
```

```
c45M.commandline('-m 50')

learners = [major, dBayes, fBayes, c45, c45M]

# 10-kratno prečno preverjanje
results = orngEval.CrossValidation(learners, data)
cdt = orngEval.computeCDT(results)
print "\nLearner         Accuracy     AUC        10FCV"
for i in range(len(learners)):
  print "%-15s %5.3f     %5.3f" % (learners[i].name, \
                                   orngEval.CA(results)[i], \
                                   orngEval.aROC(cdt[i])[7])

#Test modelov strojnega učenja na testni množici
resT= orngEval.LearnAndTestWithTestData(learners, data, testdata)
cdtT = orngEval.computeCDT(resT)
print "\nLearner         Accuracy     AUC        "
for i in range(len(learners)):
  print "%-15s %5.3f     %5.3f" % (learners[i].name, \
                                   orngEval.CA(resT)[i], \
                                   orngEval.aROC(cdtT[i])[7])
```

B. Skripte SQL

Kreiranje tabele za osnovno množico podatkov

```
CREATE TABLE FIOS2001 (
  KOMI_ID                  NUMBER,
  FIOS_ID                  NUMBER,
  PTT                      VARCHAR2 (5),
  DRZAVA                   VARCHAR2 (3),
  STAROST                  NUMBER,
  MESEC_ROJSTVA            NUMBER (2),
  SPOL                     VARCHAR2 (1),
  ZAPOSLEN                 VARCHAR2 (1),
  BONITETA                 VARCHAR2 (3),
  BONITETA_MIN             VARCHAR2 (3),
  BONITETA_MAX             VARCHAR2 (3),
  STRANKA_MESECEV         NUMBER,
  AKTIVNIH_RACUNOV         NUMBER,
  AKTIVNIH_PRODUKTOV       NUMBER,
  TR_MESECEV               NUMBER,
  TR_BONITETA              VARCHAR2 (3),
  TR_STANJE                NUMBER,
  TR_STANJE_AVG            NUMBER,
  TR_LIMIT                 NUMBER,
  TR_LIMIT_PREKORACEN_DNI  NUMBER,
  TR_PROMET_DOBRO          NUMBER,
  TR_PROMET_BREME          NUMBER,
  TR_DOHODEK_AVG           NUMBER,
  TR_POOBLASCENCEV         NUMBER (1),
  IMA_ZR                   VARCHAR2 (1),
  IMA_DR                   VARCHAR2 (1),
  IMA_DV                   VARCHAR2 (1),
  IMA_HV                   VARCHAR2 (1),
  IMA_TOLDEP               VARCHAR2 (1),
  IMA_DEVDEP               VARCHAR2 (1),
  IMA_RENVAR               VARCHAR2 (1),
  IMA_NAMVAR               VARCHAR2 (1),
  IMA_VARKNJ               VARCHAR2 (1),
  IMA_KRED                 VARCHAR2 (1),
  IMA_VISA                 VARCHAR2 (1),
  IMA_NSVS                 VARCHAR2 (1),
  IMA_ABANET               VARCHAR2 (1),
  IMA_TR_TRAJNIK           VARCHAR2 (1),
  VISA_ST_KARTIC           NUMBER,
  IMA_VISA_KLASIK          VARCHAR2 (1),
  IMA_VISA_GOLD            VARCHAR2 (1),
  IMA_VISA_ELEKTRON        VARCHAR2 (1),
  IMA_VISA_KREDITNA        VARCHAR2 (1),
  IMA_VISA_POSLOVNA        VARCHAR2 (1),
  VISA_TRAJANJE            NUMBER,
  VISA_PROMET              NUMBER,
  TR_ST_PRILIVOV           NUMBER,
  TR_ST_ODLIVOV            NUMBER,
  TR_ST_BANKOMAT           NUMBER,
  IMA_IBI                  VARCHAR2 (1))
```

Programi za polnjenje podatkov v tabelo iz podatkovnega skladišča

```
-----------------------------------------------------------------
--
-- Procedura za inicialno polnjenje podatkov o strankah, ki
-- imajo tekoče račune (TR)
--
-----------------------------------------------------------------
CREATE OR REPLACE PROCEDURE inicialno_polnenje
  (p_datum IN DATE DEFAULT TO_DATE('31.12.2001','DD.MM.YYYY'))
AS
```

```
v_date        CONSTANT DATE := p_datum;
rec           fios2001%ROWTYPE;
-- Kazalec za vse stranke, ki imajo TR
CURSOR c_komi IS
  SELECT *
    FROM ab_dim_komi
    WHERE komi_dat_rac_p <= v_date
      AND (dat_velj_kon IS NULL OR dat_velj_kon > v_date)
      AND EXISTS (SELECT NULL FROM ab_dim_racun
                    WHERE komi_id = ab_dim_komi.komi_id
                      AND produkt_id = 'TR';
-- Kazalec za podatke o fizični osebi - stranki
CURSOR c_fios (p_fios_id IN NUMBER) IS
  SELECT TRUNC(MONTHS_BETWEEN(v_date, f.fios_dat_roj)/12) AS starost,
         TO_CHAR(f.fios_dat_roj, 'MM') AS mesec_rojstva,
         f.fios_spol AS spol,
         n.ptt_id AS ptt,
         NVL(n.drz_id, '705') AS drzava,
         DECODE(f.status_komi_id,
                NULL, NULL, 1, 1, 7, 1, 99, NULL, 0
               ) AS zaposlen
    FROM ab_dim_fios f,
         ab_rel_fios_nasl rfn,
         ab_naslov n
    WHERE f.fios_id = p_fios_id
      AND f.fios_id = rfn.fios_id
      AND rfn.naslov_tip = 'S'
      AND rfn.naslov_id = n.naslov_id;
-----------------------------------------------------------------
-- Podprocedura poišče najnižjo in najvišjo bonitetno oceno stranke
-- v opazovanem letu
-----------------------------------------------------------------
PROCEDURE get_komi_boniteta (p_komi_id IN NUMBER,
                             p_boniteta OUT VARCHAR2,
                             p_boniteta_min OUT VARCHAR2,
                             p_boniteta_max OUT VARCHAR2)
IS
BEGIN
  BEGIN
    SELECT ocena INTO p_boniteta FROM ab_boni_komi
      WHERE komi_id = p_komi_id AND dat_id = v_date;
  EXCEPTION
    WHEN OTHERS THEN NULL;
  END;
  BEGIN
    SELECT RTRIM(MIN(RPAD(ocena,3,'X')),'X'),
           RTRIM(MAX(RPAD(ocena,3,'X')),'X')
      INTO p_boniteta_max,
           p_boniteta_min
      FROM ab_boni_komi
      WHERE komi_id = p_komi_id
        AND dat_id BETWEEN ADD_MONTHS(v_date, -11) AND v_date;
  EXCEPTION
    WHEN OTHERS THEN NULL;
  END;
END get_komi_boniteta;
-----------------------------------------------------------------
-- Podprocedura za podatke o tekočem računu stranke
-----------------------------------------------------------------
PROCEDURE get_tr (p_komi_id IN NUMBER,
                  p_tr_mesecev OUT NUMBER,
                  p_tr_boniteta OUT VARCHAR2,
                  p_tr_stanje OUT NUMBER,
                  p_tr_stanje_avg OUT NUMBER,
                  p_tr_limit OUT NUMBER,
                  p_tr_limit_prekoracen_dni OUT NUMBER,
                  p_tr_limit_prekoracitev_avg OUT NUMBER,
                  p_tr_promet_dobro OUT NUMBER,
                  p_tr_promet_breme OUT NUMBER,
                  p_tr_dohodek_avg OUT NUMBER,
                  p_tr_pooblascencev OUT NUMBER)
IS
  -- Kazalec za osnovne podatke o strankinem tekočem računu
  CURSOR c_racun IS SELECT * FROM ab_dim_racun
                      WHERE produkt_id || '' = 'TR'
```

```
                      AND komi_id = p_komi_id
                      AND dat_odp <= v_date
                      AND (dat_zap IS NULL OR dat_zap > v_date)
                    ORDER BY dat_odp;
  -- Kazalec za stanje, boniteto, limit,... na računu za opazovani datum
  CURSOR c_cpo_tr (p_racun_id IN NUMBER) IS
    SELECT * FROM cpo_fios_dan_tr
      WHERE racun_id = p_racun_id AND dat_id = v_date;
BEGIN
  FOR r_racun IN c_racun LOOP
    -- Koliko mesecev je TR odprt
    p_tr_mesecev := ROUND(MONTHS_BETWEEN(v_date, r_racun.dat_odp));
    -- Boniteta, stanje, povprečni dohodek, limit TR
    FOR r_cpo_tr IN c_cpo_tr(r_racun.racun_id) LOOP
      p_tr_boniteta := r_cpo_tr.boniteta;
      p_tr_stanje := r_cpo_tr.stanje_rac;
      SELECT ROUND(AVG(povpr_stanje)), ROUND(AVG(redni_priliv))
        INTO p_tr_stanje_avg, p_tr_dohodek_avg
        FROM cpo_fios_dan_tr
        WHERE racun_id = r_racun.racun_id
          AND dat_id BETWEEN ADD_MONTHS(v_date, -11) AND v_date;
      p_tr_limit := r_cpo_tr.znesek_lim;
    END LOOP;
    -- povprečno stanje na TR
    SELECT ROUND(AVG(povpr_stanje))
      INTO p_tr_stanje_avg
      FROM cpo_fios_dan_tr
      WHERE racun_id = r_racun.racun_id
        AND dat_id BETWEEN ADD_MONTHS(v_date, -11) AND v_date;
    -- Število pooblaščencev za ta TR
    SELECT COUNT(*)
      INTO p_tr_pooblascencev
      FROM ab_dim_racun
      WHERE racun_ido = r_racun.racun_ido
        AND produkt_id || '' = 'POOBL'
        AND dat_odp <= v_date
        AND (dat_zap IS NULL OR dat_zap > v_date);
  END LOOP;
EXCEPTION
  WHEN OTHERS THEN NULL;
END get_tr;
-----------------------------------------------------------------
-- Podprocedura za vnos dobljenih podatkov v končno tabelo
-----------------------------------------------------------------
PROCEDURE insert_record (p_rec IN fios2001%ROWTYPE)
IS
BEGIN
  INSERT INTO fios2001(komi_id,
                       fios_id,
                       ptt,
                       drzava,
                       starost,
                       mesec_rojstva,
                       spol,
                       zaposlen,
                       boniteta,
                       boniteta_min,
                       boniteta_max,
                       stranka_mesecev,
                       aktivnih_racunov,
                       aktivnih_produktov,
                       tr_mesecev,
                       tr_boniteta,
                       tr_stanje,
                       tr_stanje_avg,
                       tr_limit,
                       tr_limit_prekoracen_dni,
                       tr_limit_prekoracitev_avg,
                       tr_promet_dobro,
                       tr_promet_breme,
                       tr_dohodek_avg,
                       tr_pooblascencev)
    VALUES (p_rec.komi_id,
            p_rec.fios_id,
            p_rec.ptt,
```

```
              p_rec.drzava,
              p_rec.starost,
              p_rec.mesec_rojstva,
              p_rec.spol,
              p_rec.zaposlen,
              p_rec.boniteta,
              p_rec.boniteta_min,
              p_rec.boniteta_max,
              p_rec.stranka_mesecev,
              p_rec.aktivnih_racunov,
              p_rec.aktivnih_produktov,
              p_rec.tr_mesecev,
              p_rec.tr_boniteta,
              p_rec.tr_stanje,
              p_rec.tr_stanje_avg,
              p_rec.tr_limit,
              p_rec.tr_limit_prekoracen_dni,
              p_rec.tr_limit_prekoracitev_avg,
              p_rec.tr_promet_dobro,
              p_rec.tr_promet_breme,
              p_rec.tr_dohodek_avg,
              p_rec.tr_pooblascencev);
  END insert_record;
BEGIN
  FOR r_komi IN c_komi LOOP
    -- Število aktivnih računov in aktivnih produktov te stranke
    SELECT COUNT(racun_id), COUNT(DISTINCT produkt_id)
      INTO rec.aktivnih_racunov, rec.aktivnih_produktov
      FROM ab_dim_racun
      WHERE komi_id = r_komi.komi_id
        AND dat_odp <= v_date
        AND (dat_zap IS NULL OR dat_zap > v_date)
        AND (produkt_id NOT IN ('POOBL') AND racun_ido_dodatek IS NULL);
    IF rec.aktivnih_racunov > 0 THEN  -- obstajajo aktivni računi
      rec.komi_id := r_komi.komi_id;
      rec.fios_id := r_komi.fios_id;
      -- Podatki o fizični osebi - stranki
      FOR r_fios IN c_fios(r_komi.fios_id) LOOP
        rec.ptt := r_fios.ptt;
        rec.drzava := r_fios.drzava;
        rec.starost := r_fios.starost;
        rec.mesec_rojstva := r_fios.mesec_rojstva;
        rec.spol := r_fios.spol;
        rec.zaposlen := r_fios.zaposlen;
      END LOOP;
      rec.stranka_mesecev := ROUND(MONTHS_BETWEEN(v_date, r_komi.komi_dat_rac_p));
      -- Najnižja in najvišja bonitetna ocena stranke
      get_komi_boniteta (rec.komi_id,
                         rec.boniteta,
                         rec.boniteta_min,
                         rec.boniteta_max);
      -- Podatki o TR
      get_tr (rec.komi_id, rec.tr_oe, rec.tr_mesecev,
              rec.tr_boniteta, rec.tr_ba_kart,
              rec.tr_visa_el, rec.tr_trajnal,
              rec.tr_stanje, rec.tr_stanje_avg,
              rec.tr_limit, rec.tr_limit_prekoracen_dni,
              rec.tr_limit_prekoracitev_avg,
              rec.tr_promet_dobro, rec.tr_promet_breme,
              rec.tr_dohodek_avg, rec.tr_pooblascencev);

      -- Shranim zbrane podatke
      insert_record (rec);

    END IF;
  END LOOP;
END inicialno_polnenje;
BEGIN
  inicialno_polnenje (TO_DATE('31.12.2001','DD.MM.YYYY'));
  COMMIT;
END;
-- Polnjenje posamičnih atributov, ki niso zajeti v
-- proceduri INICIALNO_POLNENJE
-- Ima žiro račun
UPDATE fios2001 x SET ima_zr = '1' WHERE EXISTS
```

```
  (SELECT 1 FROM ab_dim_racun
     WHERE komi_id = x.komi_id
       AND produkt_id = 'ZR'
       AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
       AND (dat_zap IS NULL OR
            dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY')));
COMMIT;
-- Ima devizni račun
UPDATE fios2001 x SET ima_dr = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_racun
    WHERE komi_id = x.komi_id
      AND produkt_id= 'DR'
      AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
      AND (dat_zap IS NULL OR dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY')));
COMMIT;
-- Ima devizno knjižico
UPDATE fios2001 x SET ima_dv = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_racun
    WHERE komi_id = x.komi_id
      AND produkt_id = 'DV'
      AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
      AND (dat_zap IS NULL OR dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY')));
COMMIT;
-- Ima hranilno knjižico
UPDATE fios2001 x SET ima_hv = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_racun
    WHERE komi_id = x.komi_id
      AND produkt_id = 'HV'
      AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
     AND (dat_zap IS NULL OR dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY'))
  );
COMMIT;
-- Ima tolarski depozit
UPDATE fios2001 x SET ima_toldep = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_racun
    WHERE komi_id = x.komi_id
      AND produkt_id = 'TOLDEP'
      AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
      AND (dat_zap IS NULL OR dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY')));
COMMIT;
-- Ima devizni depozit
UPDATE fios2001 x SET ima_devdep = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_racun
    WHERE komi_id = x.komi_id
      AND produkt_id = 'DEVDEP'
      AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
      AND (dat_zap IS NULL OR dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY')));
COMMIT;
-- Ima rentno varčevanje
UPDATE fios2001 x SET ima_renvar = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_racun
    WHERE komi_id = x.komi_id
      AND produkt_id = 'RENVAR'
      AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
      AND (dat_zap IS NULL OR dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY')));
COMMIT;
-- Ima namensko varčevanje
UPDATE fios2001 x SET ima_namvar = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_racun
    WHERE komi_id = x.komi_id
      AND produkt_id = 'NAMVAR'
      AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
      AND (dat_zap IS NULL OR dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY')));
COMMIT;
-- Ima varčevalno knjižico
UPDATE fios2001 x SET ima_varknj = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_racun
    WHERE komi_id = x.komi_id
      AND produkt_id = 'VARKNJ'
      AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
      AND (dat_zap IS NULL OR dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY')));
COMMIT;
-- Ima kredit
UPDATE fios2001 x SET ima_kred = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_racun
```

```
    WHERE komi_id = x.komi_id
    AND produkt_id = 'KRED'
    AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
    AND (dat_zap IS NULL OR dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY')));
COMMIT;
-- Ima kakršnokoli VISA kartico
UPDATE fios2001 x SET ima_visa = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_racun
    WHERE komi_id = x.komi_id
      AND produkt_id = 'VISA'
      AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
      AND (dat_zap IS NULL OR dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY')));
COMMIT;
-- Varčuje v nacionalni varčevalni shemi
UPDATE fios2001 x SET ima_nsvs = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_racun
    WHERE komi_id = x.komi_id
      AND produkt_id='NSVS'
      AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
      AND (dat_zap IS NULL OR dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY')));
COMMIT;
-- Ima Abanet
UPDATE fios2001 x SET ima_abanet = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_fios
     WHERE fios_id = x.fios_id
       AND abanet_id IS NOT NULL);
COMMIT;
-- Ima trajnik na tekočem računu
UPDATE fios2001 x SET ima_tr_trajnik = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_tn t, ab_dim_racun r
    WHERE r.komi_id = x.komi_id
      AND r.produkt_id='TR'
      AND r.dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
      AND (r.dat_zap IS NULL OR
           r.dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY'))
      AND r.racun_id = t.racun_id);
COMMIT;
-- Število VISA kartic, ki jih ima stranka
UPDATE fios2001 x SET visa_st_kartic =
  (SELECT COUNT(1) FROM ab_dim_racun
    WHERE komi_id = x.komi_id
      AND produkt_id = 'VISA'
      AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
      AND (dat_zap IS NULL OR dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY')))
  WHERE ima_visa IS NOT NULL;
COMMIT;
-- Ima VISA Klasik kartico
UPDATE fios2001 x SET ima_visa_klasik = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_racun r, ab_dim_racun_visa v
    WHERE komi_id = x.komi_id
      AND produkt_id = 'VISA'
      AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
      AND (dat_zap IS NULL OR dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY'))
      AND r.racun_id = v.racun_id
      AND v.visa_tip = 'klasična')
  AND ima_visa IS NOT NULL;
COMMIT;
-- Ima VISA Gold kartico
UPDATE fios2001 x SET ima_visa_gold = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_racun r, ab_dim_racun_visa v
    WHERE komi_id = x.komi_id
      AND produkt_id = 'VISA'
      AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
      AND (dat_zap IS NULL OR dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY'))
      AND r.racun_id = v.racun_id
      AND v.visa_tip = 'zlata')
  AND ima_visa IS NOT NULL;
COMMIT;
-- Ima VISA Elektron kartico
UPDATE fios2001 x SET ima_visa_elektron = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_racun r, ab_dim_racun_visa v
     WHERE komi_id = x.komi_id
       AND produkt_id = 'VISA'
       AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
       AND (dat_zap IS NULL OR
```

```
            dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY'))
       AND r.racun_id = v.racun_id
       AND v.visa_tip = 'electron')
  AND ima_visa IS NOT NULL;
COMMIT;
-- Ima VISA Kreditno kartico
UPDATE fios2001 x SET ima_visa_kreditna = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_racun r, ab_dim_racun_visa v
    WHERE komi_id = x.komi_id
      AND produkt_id = 'VISA'
      AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
      AND (dat_zap IS NULL OR dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY'))
      AND r.racun_id = v.racun_id
      AND v.visa_tip = 'kreditna')
  AND ima_visa IS NOT NULL;
COMMIT;
-- Ima VISA Poslovno kartico
UPDATE fios2001 x SET ima_visa_poslovna = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_ivisa_kart v
    WHERE fios_id = x.fios_id
      AND dat_pri <= TO_DATE('31.12.2001','DD.MM.YYYY')
      AND (dat_kon IS NULL OR dat_kon > TO_DATE('31.12.2001','DD.MM.YYYY')) )
  AND ima_visa IS NOT NULL;
COMMIT;
-- Trajanje (število mesecev) najstarejše Visa kartice
UPDATE fios2001 x SET visa_trajanje =
  (SELECT ROUND(MONTHS_BETWEEN (TO_DATE('31.12.2001','DD.MM.YYYY'), MIN(dat_odp)))
     FROM ab_dim_racun
     WHERE komi_id = x.komi_id
       AND produkt_id='VISA'
       AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
       AND (dat_zap IS NULL OR
            dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY')))
  WHERE ima_visa IS NOT NULL;
COMMIT;
-- Celoten promet z VISA karticami
UPDATE fios2001 x SET visa_promet =
  (SELECT SUM(v.znesek_p_sit)
     FROM ab_trans_visa v, ab_dim_racun r
     WHERE r.komi_id = x.komi_id
       AND r.produkt_id = 'VISA'
       AND r.dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
       AND (r.dat_zap IS NULL OR
            r.dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY'))
       AND r.racun_id = v.racun_id
       AND v.dat_brem BETWEEN TO_DATE('01.01.2001','DD.MM.YYYY') AND
                              TO_DATE('31.12.2001','DD.MM.YYYY'))
  WHERE ima_visa IS NOT NULL;
COMMIT;
-- Ima trgovalni račun
UPDATE fios2001 x SET ima_ibi = '1' WHERE EXISTS
  (SELECT 1 FROM ab_dim_racun
    WHERE komi_id = x.komi_id
      AND produkt_id = 'IB'
      AND dat_odp <= TO_DATE('31.12.2001','DD.MM.YYYY')
      AND (dat_zap IS NULL OR dat_zap > TO_DATE('31.12.2001','DD.MM.YYYY')));
COMMIT;
-- Število dvigov na bankomatih
UPDATE fios2001 x SET tr_st_bankomat =
 (SELECT COUNT(*) FROM ab_trans_tr
   WHERE racun_id = x.komi_id
     AND trans_id = 6093);  -- oznaka transakcij na bankomatih
COMMIT;
-- Število vseh prilivov in odlivov na TR
UPDATE fios2001 X SET (tr_st_odlivov, tr_st_prilivov) =
  (SELECT SUM(DECODE(DOBRO_BREME,'-',1,0)),
          SUM(DECODE(DOBRO_BREME,'+',1,0))
     FROM ab_trans_tr
     WHERE racun_id = x.komi_id);
COMMIT;
-- Promet v dobro in v breme na TR
UPDATE fios2001 X SET (tr_promet_dobro, tr_promet_breme) =
  (SELECT SUM(DECODE(dobro_breme,'+',znesek,0)),
          SUM(DECODE(dobro_breme,'-',znesek,0))
     FROM ab_trans_tr
```

```
     WHERE racun_id = x.komi_id);
COMMIT;
-- Število dni s prekoračenim limitom
UPDATE fios2001 x SET tr_limit_prekoracen_dni =
  abf_lim_prekoracitev_dni
    (komi_id => x.komi_id,
     start_date => TO_DATE('01.01.2001','dd.mm.yyyy'),
     end_date => TO_DATE('31.12.2001','dd.mm.yyyy'));
COMMIT;
```

Izjava

Izjavljam, da sem diplomsko delo izdelal samostojno pod vodstvom mentorja doc. dr. Blaža Zupana. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Marko Šmid