UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

MAGISTRSKO DELO

ANALIZA PRIMERNOSTI UPORABE METOD PODATKOVNEGA RUDARJENJA ZA MODELIRANJE TVEGANJA PRI ZAVAROVANJU MOTORNIH VOZIL

Ljubljana, september 2014 NATAŠA ŠAJN JURŠIČ

IZJAVA O AVTORSTVU

Spodaj podpisana Nataša Šajn Juršič, študentka Ekonomske fakultete Univerze v Ljubljani, izjavljam, da sem avtorica magistrskega dela z naslovom Analiza primernosti uporabe metod podatkovnega rudarjenja za modeliranje tveganja pri zavarovanju motornih vozil, pripravljenega v sodelovanju s svetovalcem prof. dr. Jurijem Jakličem.

Izrecno izjavljam, da v skladu z določili Zakona o avtorski in sorodnih pravicah (Ur. l. RS, št. 21/1995 s spremembami) dovolim objavo magistrskega dela na fakultetnih spletnih straneh.

S svojim podpisom zagotavljam, da

- je predloženo besedilo rezultat izključno mojega lastnega raziskovalnega dela;
- je predloženo besedilo jezikovno korektno in tehnično pripravljeno v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani, kar pomeni, da sem
    - poskrbela, da so dela in mnenja drugih avtorjev oziroma avtoric, ki jih uporabljam v magistrskem delu, citirana oziroma navedena v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani, in
    - pridobila vsa dovoljenja za uporabo avtorskih del, ki so v celoti (v pisni ali grafični obliki) uporabljena v tekstu, in sem to v besedilu tudi jasno zapisala;
- se zavedam, da je plagiatorstvo – predstavljanje tujih del (v pisni ali grafični obliki) kot mojih lastnih – kaznivo po Kazenskem zakoniku (Ur. l. RS, št. 55/2008 s spremembami);
- se zavedam posledic, ki bi jih na osnovi predložene magistrskega dela dokazano plagiatorstvo lahko predstavljalo za moj status na Ekonomski fakulteti Univerze v Ljubljani v skladu z relevantnim pravilnikom.

V Ljubljani, dne 3.9.2014 Podpis avtorice:

KAZALO

KAZALO SLIK

KAZALO TABEL

UVOD

V zavarovalništvu je izračun ocene pričakovanih škod na podlagi preteklih podatkov ključnega pomena za uspešno poslovanje. Pri tem problemu se nam v okviru zavarovalnih pogodb poraja vprašanje, kako iz podatkov, ki jih imamo v trenutku sklenitve pogodbe, napovedati verjetnost škode na dani pogodbi v času njenega trajanja. Specifično se v tem delu ukvarjam z zavarovalnimi pogodbami s področja zavarovanj motornih vozil, saj so v Sloveniji po podatkih Slovenskega zavarovalnega združenja med premoženjskimi zavarovanji ta najbolj množično zastopana, med njimi pa po pričakovanjih prednjači zakonsko predpisano Zavarovanje odgovornosti pri uporabi vozil, bolj znano pod imenom avtomobilska odgovornost ali AO.

V navezi z osnovnim problemom ocenjevanja verjetnosti škod so nam pri pogodbah AO v trenutku sklenitve pogodbe znani na primer naslednji podatki: tip in znamka vozila, moč motorja, bonitetni razred zavarovalca, njegova starost, spol, morebiti tudi škodna zgodovina in podobno. Na podlagi teh podatkov želimo oceniti verjetnost škode na novi AO pogodbi v določenem časovnem obdobju. V magistrskem delu sem se osredotočila na opisani problem in skušala najti najustreznejšo rešitev.

Omenjeni problem sem reševala z metodami podatkovnega rudarjenja (angl. *data mining*), ki je po navadi definirano kot proces odkrivanja prej neznanih korelacij, vzorcev in trendov z rudarjenjem velikih količin podatkov shranjenih v podatkovnih skladiščih z uporabo statističnih metod, metod strojnega učenja in umetne inteligence ter z raznimi tehnikami predstavitve podatkov (Sumathi & Sivanandam, 2006, str. 8).

Zavarovalnice dandanes razpolagajo z velikimi količinami podatkov o svojih strankah, njihovih pogodbah, naročenih produktih, medsebojnih povezavah med strankami ... Vendar pa je iz take količine podatkov pogosto težko pridobiti koristne informacije, ki bi jih lahko učinkovito uporabili pri poslovnem odločanju. Metode podatkovnega rudarjenja nam nudijo nabor praktičnih orodij, s katerimi se lahko na učinkovit način dokopljemo do novega znanja skritega v sicer neobvladljivi količini podatkov. Možnosti uporabe je veliko, na primer:

- identifikacija skupin strank z določenim vzorcem vedenja; to nam omogoča, da lahko npr. odkrijemo tiste stranke oz. lastnosti tistih strank, ki so nagnjene k prevaram in tako zmanjšamo število prevar (IBM Corporation, 2011),
- napoved in detekcija tveganega vedenja; s tem pristopom lahko ocenimo stranke, ki predstavljajo visoko tveganje za zavarovalnico (Scism & Maremont, 2010),
- izboljšava postopkov, ki jih zavarovalnica izvaja ob prevzemu tveganja v zavarovanje (analiza tveganja, vrednotenje oz. ocena tveganja, izstavitev zavarovalne police) (angl. *underwriting*) in procesa upravljanja s tveganji (Deloitte Touche Tohmatsu, 2007),
- odkrivanje pogostih vzorcev pri sklepanju zavarovanj in s tem identifikacija strank, ki sledijo temu vzorcu; tako lahko pridobimo nove stranke oz. obdržimo stare stranke ter razvijemo nove produkte (SAS Institute Inc., 2002),

- segmentacija strank glede napovedovanja tveganja; tako pomagamo zavarovalnici, da poveča dobiček in zmanjša stroške (Devale & Kulkarni, 2012),
- odkrivanje novih trendov v škodah; to nam omogoča boljšo napoved škod in boljšo prilagoditev premij, kar posledično pomeni večjo korist tako za stranke kot tudi za zavarovalnico (Easy.Data.Mining™, b.l.).

V delu sem preko zavarovalne pogodbe napovedovala tudi tveganje določenega zavarovalca oz., ali bo določen zavarovalec povzročil škodo (tu privzamem, da je zavarovalec tudi voznik vozila, čeprav v praksi to ni nujno). V magistrskem delu bom besedi zavarovalec in zavarovanec obravnavala kot sinonima, čeprav to ni vedno res, saj je zavarovalec oseba, ki sklene pogodbo z zavarovalnico, zavarovanec pa oseba, katere premoženjski interes je zavarovan.

Modeli, ki sem jih izdelala, napovedujejo verjetnost škode na AO pogodbi. Škode so naključni dogodki, kljub temu pa obstajajo zakonitosti, ki nam pomagajo določiti tiste voznike oz. pogodbe, ki so bolj tvegane (pri tem je težko določiti, kakšen odstotek predstavlja naključje in kakšen odstotek je možno napovedati). Izdelani modeli sicer ne napovedujejo verjetnosti škode s 100 % natančnostjo, vendar pa so napovedi pridobljene z metodami podatkovnega rudarjenja dobro korelirane z dejanskimi nesrečami. S tem so se metode podatkovnega rudarjenja v mojem delu izkazale za primeren pristop k omenjeni problematiki.

Problem, ki sem ga reševala v magistrskem delu, lahko obravnavamo tako z metodami klasifikacije kot z metodami regresije. Temu primerno sem med metodami podatkovnega rudarjenja iskala tiste, ki so primerne za klasifikacijo ali regresijo: odločitvena drevesa, odločitvena pravila, metoda Naivnega Bayesa, nevronske mreže, linearne metode in podobne.

Metode podatkovnega rudarjenja niso univerzalna rešitev za vse tipe analiz podatkov. Lahko se zgodi, da v določenih primerih ne dajo zadovoljivih rezultatov. Metode se med seboj precej razlikujejo. Nekatere so primerne za določen tip problemov, spet druge za drugačen tip. Poglavitna naloga tega magistrskega dela je tako bila preizkus različnih metod ter na podlagi rezultatov izbrati tiste, ki so najprimernejše za reševanje predstavljenih zavarovalniških problemov.

Namen magistrskega dela je bil analizirati primernost uporabe metod podatkovnega rudarjenja pri problemu tveganja zavarovalne pogodbe ter izdelati najprimernejši model na podlagi dejanskih podatkov iz baze kompozitne zavarovalnice, ki trži zavarovanja motornih vozil. Magistrsko delo je zasnovano na predpostavki, da so znani osnovni pojmi zavarovalništva ter verjetnostnega računa in statistike.

Cilji magistrskega dela so bili naslednji:

- pregled standardnih pristopov in postavitev osnovnih gradnikov metodologije dela podatkovnega rudarjenja ter prilagoditev le-teh za aktuarske potrebe,

- predstavitev priprave podatkov, ki vključuje tudi čiščenje podatkov ter generiranje novih, izpeljanih atributov,
- izbira najustreznejših metod podatkovnega rudarjenja za predstavljen problem,
- določitev najboljših parametrov za posamezno metodo,
- izbira najpomembnejših atributov za posamezno metodo,
- evalvacija vseh izdelanih modelov in izbor najboljšega oz. najprimernejšega med njimi,
- ovrednotenje primernosti uporabe metod podatkovnega rudarjenja za reševanje omenjene problematike.

Pri izdelavi magistrskega dela sem uporabila teoretične podlage s področja podatkovnega rudarjenja tujih in domačih avtorjev. Veliko snovi sem zajela iz elektronskih virov, prispevkov in člankov z najnovejšimi spoznanji iz tega hitro razvijajočega se področja. Poleg tega sem uporabila tudi teoretično podlago iz zavarovalništva, saj je pri reševanju zastavljenega problema ključnega pomena razumevanje podatkov iz področja zavarovanj motornih vozil, ki sem jih uporabila pri praktičnem delu. Poleg omenjenih virov sem se naslonila tudi na lastno znanje, ki sem ga pridobila med študijem aktuarstva, ter na svoje delovne izkušnje, ki jih pridobivam kot aktuarka, zaposlena v kompozitni zavarovalnici.

Konceptualno sem pri reševanju problema sledila CRISP-DM pristopu (angl. *Cross-Industry Standard Process for Data Mining*), ki je postal de-facto standard pri procesu podatkovnega rudarjenja v poslovni praksi, saj opisuje naraven tok tega procesa. Proces je cikličen, velikokrat se je potrebno vrniti na prejšnje stopnje in določene naloge se lahko ob spremenjenih pogojih kar nekajkrat ponovijo. CRISP-DM pristop opredeljuje reševanje nekega problema z metodami podatkovnega rudarjenja v šestih stopnjah (Putler & Krider, 2012, str. 19):

1. razumevanje problema (angl. *Business Understanding*),
2. razumevanje podatkov (angl. *Data Understanding*),
3. priprava podatkov (angl. *Data Preparation*),
4. modeliranje (angl. *Modeling*),
5. ovrednotenje kakovosti modela (angl. *Evaluation*),
6. vključitev modela v poslovno okolje (angl. *Deployment*).

Magistrsko delo je sestavljeno iz sedmih glavnih poglavij, tematika pa je dodatno razdeljena v podpoglavja. Tako kot je potek dela sledil CRISP-DM pristopu, je tudi magistrsko delo zgrajeno tako, da večina poglavij predstavlja eno ali več stopenj tega pristopa.

Uvodnemu poglavju, v katerem predstavim problem, namen, cilje in metodologijo magistrskega dela, sledi poglavje o predstavitvi domene, ki zajema kratke opise zavarovanj motornih vozil v Sloveniji ter aktuarskih metod, ki se dandanes uporabljajo v zavarovalništvu za določanje tveganja. Ta del predstavlja prvo stopnjo CRISP-DM pristopa.

V drugem poglavju sem predstavila osnovne koncepte podatkovnega rudarjenja. Najprej sem navedla osnovne definicije, sledila je predstavitev klasifikacijskih in regresijskih metod,

procedur za izbor atributov ter mer evalvacije, ki so primerne za ta problem. V tem poglavju sem predstavila tudi teorijo neuravnoteženih problemov, kamor se uvršča problem tega magistrskega dela. Poglavje sem zaključila z opisom uporabljenih orodij.

Sledijo poglavja z mojim prispevkom k reševanju predstavljene problematike. Tretje poglavje je namenjeno obdelavi podatkov. V tem poglavju sem predstavila dve stopnji CRISP-DM procesnega modela: razumevanje podatkov in priprava podatkov. Opisala sem zajem podatkov, pripravo vseh poizvedb ter najbolj zahteven del - čiščenje podatkov. Predstavila sem tudi postopek izpeljave atributov. Opisani stopnji sta zelo povezani, zato nisem strogo ločevala, kateri del pripada eni in kateri drugi stopnji.

Četrto poglavje vsebuje naslednji dve stopnji CRISP-DM pristopa: izdelavo modelov in evalvacijo le-teh. Tudi tu sem združila dve stopnji, saj sta precej prepleteni: ko sem izdelovala določen model, je bilo potrebno le-tega tudi ovrednotiti. Stopnjo izdelave modelov sem razdelila na štiri podstopnje: izbira ustrezne metode, vzorčenje, optimizacija parametrov za posamezno metodo ter izbira pomembnih atributov.

V petem poglavju sem predstavila rezultate analize ter podala odgovor na vprašanje, ali so metode podatkovnega rudarjenja primerne za reševanje omenjene problematike. Skušala sem prikazati tako dobre kot slabe strani rudarjenja podatkov s fokusom na aplikaciji zavarovanj motornih vozil. To poglavje delno vključuje zadnjo stopnjo CRISP-DM pristopa, t. j. vključitev modela v poslovno okolje, saj sem podala tudi nekaj primerov uporabe modelov ter ostalih rezultatov v zavarovalniškem poslovnem okolju.

V sklepu sem predstavila najpomembnejše ugotovitve predstavljene raziskave in podala zaključno misel.

1 ZAVAROVANJA VOZIL TER AKTUARSKI IZRAČUNI

Dobro razumevanje področja je ključnega pomena pri izdelavi kakovostnega modela. Področje, s katerim se ukvarjam v tem delu, je zavarovanje motornih vozil v Sloveniji, zato ga bom bolj podrobno predstavila. Največ poudarka dajem zavarovanju AO, ker tudi v praktičnem delu napovedujem verjetnost škode na AO pogodbi, na kratko pa se bo dotaknila tudi ostalih zavarovanj vozil. Nekaj besed namenjam aktuarskim metodam, ki se trenutno najbolj uporabljajo v zavarovalništvu za določanje tveganj: tu se bom osredotočila na izračun zavarovalne premije, ki je cena za prevzem tveganja in s tem deloma izraža tudi verjetnost škode na pogodbi, kar je fokus tega dela. Predstavila pa bom tudi kriterije za določanje AO premije, ki so trenutno v uporabi.

1.1 Delitev zavarovanj motornih vozil

Avtomobilska zavarovanja so ena izmed najbolj množičnih zavarovanj v Sloveniji. Razlogi za to so različni. Eden izmed glavnih razlogov je v zakonskih predpisih, saj 2. člen Zakona o

obveznih zavarovanjih v prometu (Ur.l. RS, št. 93/2007-UPB3, v nadaljevanju ZOZP) predpisuje, da mora vsak lastnik vozila zavarovati svoje vozilo proti odgovornosti za škodo, povzročeno tretjim osebam. To zavarovanje poznamo pod imenom zavarovanje avtomobilske odgovornosti ali AO. Po Zakonu o varnosti cestnega prometa (Ur.l. RS, št. 56/2008-UPB5, v nadaljevanju ZVCP) je vozilo vsako prevozno sredstvo, namenjeno vožnji po cesti (razen posebnih prevoznih sredstev). Med vozila se po 16. členu ZOZP štejejo vozila za prevoz oseb, za prevoz stvari in za vleko ter delovna vozila in traktorji, ki morajo imeti po predpisih o registraciji prometno dovoljenje.

Po zavarovanju AO pa določene škode niso krite. Po tem zavarovanju pravice do odškodnine nima npr. voznik vozila, s katerim je bila povzročena škoda, za škodo na stvareh ter oškodovanci v nekaterih posebnih pogojih. Zato obstaja še vrsta drugih zavarovanj, ki se navadno sklenejo istočasno in na isto zavarovalno pogodbo kot AO, krijejo pa te izključitve.

Zaradi prepletenosti AO z različnimi tipi dodatnih zavarovanj vozil predstavljam poleg AO še ostala zavarovanja vozil v Sloveniji. Posamezne lastnosti teh zavarovanj (npr. premijski razred AK), ki se pojavljajo na AO pogodbah, uporabljam kot atribute pri podatkovnem rudarjenju, zato je predstavitev dodatnih zavarovanj na tem mestu primerna. Po Zakonu o zavarovalništvu (Ur.l. RS, št. 99/2010-UPB7, v nadaljevanju ZZavar) se med zavarovanja vozil uvršča:

- **Zavarovanje odgovornosti pri uporabi vozil** vključuje zavarovanje lastnikov cestnih motornih vozil proti odgovornosti za škodo, povzročeno tretjim osebam ter zavarovanje prevozniške odgovornosti za blago v cestnem prometu.
- **Zavarovanje kopenskih vozil**, ki ga bolj poznamo pod imenom avtomobilski kasko (v nadaljevanju AK), krije škodo na lastnem vozilu zavarovanca. Sem so štete škode zaradi uničenja, poškodovanja ali izginotja zavarovanih stvari, ki nastanejo kot posledica nenadnih in od zavarovančeve volje neodvisnih dogodkov. Obseg in kritje se precej razlikujejo od posamezne zavarovalnice. Ne glede na to pa ločimo:
  - popolno zavarovanje AK, ki vključuje najširši obseg zavarovalnih kritij,
  - delno zavarovanje AK, ki krije le posamezne nevarnosti; te nevarnosti so razvrščene v skupine oz. kombinacije, ki pa se od zavarovalnice do zavarovalnice precej razlikujejo.

  Pri sklepanju popolnega AK se lahko sklene tudi dogovor o soudeležbi zavarovanca pri škodi - to je odbitna franšiza. Pri popolnem AK se podobno kot pri AO uporablja bonus-malus sistem. Po navadi je razpon tega sistema od 50 do 200 odstotkov temeljne zavarovalne premije, premijski razredi in odstotki od temeljne zavarovalne premije pa so določeni malce drugače kot pri AO. K popolnemu AK lahko priključimo tudi eno ali več kombinacij delnega AK (Fric & Prijanovič, 2010, str. 31 - 33).
- **Zavarovanje prevoza blaga** je po 2. členu ZZavar zavarovanje, ki krije škode na oz. izgubo blaga, vključno s prtljago, ne glede na vrsto prevoza.

- **Nezgodno zavarovanje** krije škodo ob nastanku nezgode med vožnjo ali upravljanjem vozila za osebe, ki so nezgodno zavarovane v vozilu ne glede na to, ali vozilo upravljajo ali se v njem prevažajo. Zavaruje se nevarnosti (Fric & Prijanovič, 2010, str. 30):
  - smrt zavarovanca zaradi nezgode: v tem primeru zavarovalnica izplača zavarovalno vsoto svojcem zavarovanca,
  - dnevna odškodnina za čas, ko zavarovanec ni sposoben opravljati svojega rednega dela oz. največ do nekega limita števila dni,
  - invalidnost kot posledica nezgode, kjer zavarovalnica izplača dogovorjeno zavarovalno vsoto pri stoodstotni invalidnosti, oz. izplača del zavarovalne vsote glede na odstotek invalidnosti, ki se ga ugotavlja po tabelah invalidnosti,
  - stroški zdravljenja do dogovorjenega zneska, ki jih ima zavarovanec zaradi nezgode in jih mora plačati sam.

Kot dodatna zavarovanja se k zgoraj omenjenim lahko sklene še naslednja zavarovanja:

- **Zavarovanje voznika za telesne poškodbe** (v nadaljevanju AO plus) je nastalo kot dopolnitev zavarovanja AO, saj voznik vozila, s katerim je bila povzročena škoda, nima pravice do odškodnine iz zavarovanja AO. S tem zavarovanjem je krita škoda zaradi telesnih poškodb, ki jih v prometni nesreči utrpi voznik vozila, pa tudi škoda v primeru smrti voznika, ki jo zakon priznava svojcem ob smrti bližnjega. To zavarovanje je mogoče skleniti le skupaj z zavarovanjem AO in pri isti zavarovalnici kot AO (Fric & Prijanovič, 2010, str. 29).
- **Zavarovanje stroškov postopka** je zavarovanje pravne zaščite zaradi uporabe motornih vozil. Zavarovanje krije stroške odvetnikov v kazenskih postopkih in postopkih o prekrških zaradi prometne nesreče, ki je nastala z uporabo zavarovanega motornega vozila ter druge stroške postopka. V Sloveniji je to zavarovanje manj razširjeno kot v tujini. Obseg in vrsta zavarovanih nevarnosti se v posameznih zavarovalnicah razlikujeta, prav tako tudi višina zavarovalne vsote, do katere zavarovalnica krije stroške (Fric & Prijanovič, 2010, str. 36).
- **Zavarovanje avtomobilske asistence** je zavarovanje pomoči, ki krije pomoč osebam, ki zaidejo v težave na potovanju oziroma v drugih primerih odsotnosti od njihovega doma oziroma stalnega prebivališča (ZZavar, 2. člen, točka 18). Zavarovalnica nudi zavarovancu 24 ur na dan pomoč in kritje stroškov v primeru, ko je zavarovano vozilo nevozno ali neprimerno za vožnjo zaradi okvare, poškodbe, uničenja ali izginotja. Tudi pri tem zavarovanju se zavarovalni pogoji slovenskih zavarovalnic precej razlikujejo in s tem seveda tudi kritje.

Od naštetih oblik zavarovanj je le zavarovanje AO obvezno zavarovanje, ostala so prostovoljna. V nadaljevanju predstavljam še zavarovanje AO oz. njegove lastnosti in posebnosti.

Zavarovanje AO spada med zavarovanja odgovornosti, saj je tu predmet zavarovanja odgovornost lastnika vozila. Po tem zavarovanju ni krita škoda na vozilu, ampak je krita

nevarnost, da bo ogroženo premoženje lastnika vozila zaradi plačila odškodnine tretji osebi, kateri je odgovoren za škodo (Ristin, Korbar & Simoniti, 2008, str. 105). AO zavarovanje krije premoženjske in nepremoženjske zahtevke. Premoženjska škoda se običajno relativno hitro prijavi in reši, nepremoženjsko škodo pa je bistveno težje oceniti, prav tako traja precej časa, da poškodovanec okreva. Takšne škode so običajno prijavljene in tudi izplačane kasneje kot premoženjske, in sicer v enkratnem znesku ali v obliki rente.

Po znesku zbrane zavarovalne premije vseh slovenskih zavarovalnic se uvršča v sam vrh za življenjskimi in zdravstvenimi zavarovanji, po znesku izplačanih škod pa na prav tako na tretje mesto, za zdravstvenimi zavarovanji in AK (Slovensko zavarovalno združenje, 2013, str. 53).

Obveznost zavarovalnice iz zavarovanja AO je omejena z **zavarovalno vsoto**, veljavno na dan škodnega dogodka, če z zavarovalno pogodbo ni dogovorjena višja vsota (zavarovalec ima možnost, da izbere višjo zavarovalno vsoto, ki je določena kot večkratnik najnižje zavarovalne vsote). Najnižja zavarovalna vsota, na katero mora biti sklenjeno zavarovanje avtomobilske odgovornosti, znaša (19. člen ZOZP):

- za škodo zaradi smrti, telesne poškodbe in prizadetega zdravja, ki izvira iz posameznega škodnega dogodka ne glede na število oškodovancev, 5.000.000 evrov,
- za škodo zaradi uničenja ali poškodovanja stvari, ki izvira iz posameznega škodnega dogodka ne glede na število oškodovancev, 1.000.000 evrov.

Lahko pa se zgodi, da je zavarovanec kršil zavarovalno pogodbo, zavarovalnica pa je izplačala odškodnino oškodovancu. Po 7. členu ZOZP zavarovalnica nasproti oškodovancu ne more uveljavljati ugovorov, ki jih ima proti svojemu zavarovancu, ker ta ni ravnal po zakonu, zavarovalni pogodbi ali zavarovalnih pogojih. V tem primeru ima zavarovalnica pravico uveljavljati povračilo izplačanih zneskov skupaj z obrestmi in stroški od zavarovanca. Te zahtevke imenujemo **regresi**. V 7. členu so navedeni tudi primeri, kdaj ima zavarovalnica pravico do regresnih zahtevkov, vendar to presega okvir tega dela, zato jih podrobneje ne bom navajala. V teh primerih ima zavarovalnica pravico do regresa v višini 12.000 evrov, razen v primeru, ko je voznik povzročil škodo namenoma: v tem primeru lahko zavarovalnica uveljavlja regres v celotnem znesku odškodnine.

Vsebinski opis zaključujem z navezavo na problem, ki sem ga reševala v tem delu. Za napovedovanje škod potrebujem jasno specificirano definicijo, kaj smatram kot škoda na pogodbi – torej katere so tiste pogodbe, ki so zame pozitivni primeri škod. To definira, kaj bom napovedovala, oz. kakšen tip problema bom reševala z metodami podatkovnega rudarjenja. Po začetnem vpogledu in poskusih na podatkih, ki so mi bili na voljo, sem prišla do spoznanja, da lahko definiram naslednje tri poglavitne koncepte škodnih pogodb:

- tiste pogodbe, pri katerih je prišlo do nesreče,
- tiste pogodbe, ki imajo odprt škodni spis, torej pri katerih je prišlo do prijave škode,
- tiste pogodbe, katerim je bila škoda dejansko izplačana.

Čeprav je velika večina pogodb takih, da spadajo ali v vse tri zgornje kategorije ali pa v nobeno, pa obstaja nezanemarljiv delež tistih, za katere to ne velja. S tem pride do bistvene razlike pri interpretaciji modelov in rezultatov na podlagi rudarjenja podatkov:

- Prva možnost je zelo zanimiva s stališča prometne varnosti, saj bi lahko bolje ocenjevali, kakšne lastnosti veljajo za varne oz. nevarne pogodbe/voznike in bi lahko aktivno vplivali na prometno varnost. Žal podatkov o vseh dejanskih nesrečah nimamo v bazi zavarovalnice, ampak imamo le tiste, ki so bile prijavljene.
- Druga možnost je tesno navezana na prvo (enaka, v kolikor bi bile prijavljene vse AO škode), vendar je bolj realna, saj te podatke imamo. Izjema so primeri manjših nesreč in potencialnih majhnih odškodnin, saj se vpleteni pogosto pogodijo brez prijave škode.
- Tretja možnost pa je vsebinsko bližje perspektivi zavarovalnice pri ocenjevanju tveganja. Tu nas ne zanima, koliko je dejanskih nesreč, oz. koliko je prijavljenih nesreč, temveč kakšne so lastnosti tistih pogodb, katerim je morala zavarovalnica izplačati škodo.

Ker je zame najbolj zanimiva interpretacija s stališča zavarovalniškega ocenjevanja tveganja, sem izbrala tretjo možnost, zato zame škode predstavljajo zgolj tiste pogodbe, pri katerih je prišlo do izplačila. Zgoraj predstavljeni regresi so tako izključeni, v kolikor jih je morala stranka vrniti. Prav tako ni pomembno, ali je prišlo do izplačila materialne ali nematerialne škode. V tem delu predstavlja škodo vsako utemeljeno izplačilo ne glede na višino zneska.

1.2 Aktualne metode za določanje tveganja v zavarovalništvu

Eden glavnih problemov, s katerim se ukvarjajo zavarovalnice, je ocenjevanje tveganj, ki jih prevzemajo. Zavarovalna premija je cena za prevzem tega tveganja, sprejemljiva pa mora biti tako za zavarovalnico kot za sklenitelja zavarovanja (Komelj, 2012). To pomeni, da morajo biti v premiji zajeti vsi znani dejavniki tveganja, ki odražajo stranko, po drugi strani pa mora biti določena premija poštena do vseh strank, kar pomeni, da ne diskriminira določenega segmenta populacije (Chapados, 2010). Aktuarske analize zavarovalnih pogodb in škod so tako ključnega pomena pri identifikaciji, analizi in ocenjevanju tveganj premoženjskih, in v tem okviru avtomobilskih zavarovanj (Apte et al., 1998).

Tradicionalne metode, ki jih uporabljajo aktuarji za konstrukcijo modelov tveganja, vključujejo segmentacijo populacije zavarovalcev v nevarnostne skupine, ki temeljijo na dejavnikih, kot npr. spol, starost … (Apte et al., 1998). Parametri tveganja se za vsako skupino ocenijo na podlagi podatkov s preteklih polic in škodnih spisov. Za vsako tako skupino se oceni povprečno škodo, s katero se določi nevarnostno premijo (to je tisti del premije, ki v povprečju zadošča za kritje vseh škod iz sprejetih tveganj), ki velja za vse zavarovalce, ki spadajo v določeno skupino (Chapados, 2010) – tu se torej predvideva, da so vsi posamezniki na nek način enaki. Pomembno je tudi, da te skupine niso niti prevelike niti premajhne: prevelike skupine so lahko premalo homogene, premajhne skupine pa nimajo zadostnih statističnih značilnosti (Kahane, Levin, Meiri & Zahavi, 2005). Idealno so te skupine homogene glede na določeno tveganje, kar pomeni, da nam z dodajanjem spremenljivk nadaljnja delitev na podskupine da enake parametre

tveganja. Aktuarji za identifikacijo ustreznih dejavnikov tveganja po navadi uporabijo kombinacijo intuicije, ugibanj in testiranja hipotez. Tak proces zahteva veliko napora in več let razvoja in izboljšav, da dobimo dober model tveganja (Apte et al., 1998).

Tudi vsi postopki, ki jih zavarovalnica izvaja pred prevzemom individualnega tveganja v zavarovanje in zajemajo analizo tveganja in vrednotenje tveganja, v glavnem temeljijo na pristopu s segmentacijo, poleg tega pa so pri teh postopkih prisotne tudi subjektivne komponente, kar posledično vodi do velike variabilnosti v oceni tveganja (Deloitte Touche Tohmatsu, 2007).

Pristop s segmentacijo lahko privede do napačnih rezultatov zato, ker običajno ne upošteva interakcije med posameznimi atributi, ki pa je v zavarovalniških podatkih zelo pogosta (Kahane et al., 2005). Segmentacija odpove tudi, ko število spremenljivk naraste – to imenujemo "prekletstvo dimenzionalnosti" (angl. *curse of dimensionality*) (Chapados, 2010).

Pomanjkljivost konvencionalnih statističnih modelov je torej ta, da so v glavnem aditivni v smislu, da se vsak posamezen dejavnik oz. atribut, ki vpliva na višino premije, obravnava individualno in ne v povezavi z ostalimi atributi. S temi modeli je zato težko najti morebitne povezave med spremenljivkami (Steinberg, 2008). Take enosmerne analize so namreč precej občutljive na morebitne korelacije med spremenljivkami. Primer: mladi vozniki v splošnem vozijo starejše avtomobile (Anderson et al., 2007, str. 5). Enosmerna analiza starosti avtomobila pokaže veliko število škod za starejše avtomobile, kar pa je v glavnem posledica tega, da te avtomobile vozijo mlajši vozniki, ki so precej bolj tvegani kot pa starejši vozniki. Če bi povezali enosmerno analizo na starosti avtomobila in enosmerno analizo na starosti voznika, bi se učinek starosti voznika podvojil.

Ta problem je rešljiv z uporabo multivariantne analize, kamor se uvrščajo generalizirani linearni modeli (v nadaljevanju GLM), ki so prilagojeni za morebitne korelacije med atributi in omogočajo obravnavanje kombinacij atributov. GLM so danes splošno priznani na področju aktuarstva kot standardna metoda za določanje premij (angl. *pricing*) zavarovanj motornih vozil ter še nekaterih premoženjskih zavarovanj v večini držav Evropske Unije. Poleg tega se uporabljajo tudi na področju marketinga ter pri procesu sprejemanja določenega tveganja v zavarovanje. Metoda je robustna, transparentna in lahko razumljiva, kar se je v praksi izkazalo za zelo dragoceno. GLM temelji na statističnem izračunu učinka, ki ga imajo atributi na razred (Anderson et al., 2007, str. 4, 92).

V nadaljevanju navajam še nekaj pomanjkljivosti omenjenih konvencionalnih metod v primerjavi z metodami podatkovnega rudarjenja (Steinberg, 2008):

- Pri tradicionalnih metodah, kamor se uvršča tudi GLM, običajno potrebujemo veliko časa za razvoj, poleg tega se zahteva tudi precej ekspertize oz. ustrezne strokovnjake. Raziskave kažejo, da precej hitreje zgradimo modele podatkovnega rudarjenja, poleg tega pa ti modeli pogosto dosegajo boljše rezultate.

- S klasičnimi pristopi velikokrat ne zmoremo obdelati zelo velike količine podatkov. Določene metode podatkovnega rudarjenja pa zelo učinkovito obdelajo velike in kompleksne nabore podatkov ter najdejo tudi zapletene interakcije med atributi.
- Pri tradicionalnih metodah je potrebno manjkajoče vrednosti obravnavati ročno, večina metod podatkovnega rudarjenja pa le-te obravnava samodejno.
- Velik problem za konvencionalne metode predstavljajo tudi nelinearnost, osamelci in morebitni šumi v podatkih, ki lahko zelo pokvarijo rezultate. Te posebnosti je potrebno predhodno preučiti in izločiti iz podatkov. Metode podatkovnega rudarjenja pa so večinoma že zasnovane tako, da opisane posebnosti nimajo velikega vpliva na rezultat.
- Če izbiramo atribute ročno preko tradicionalnih pristopov, je to lahko zelo dolgotrajen proces, poleg tega lahko kak pomemben atribut izpustimo. To se pri metodah podatkovnega rudarjenja s samodejno izbiro atributov ne more zgoditi.

Modeliranje kompleksnejših struktur premije je s tradicionalnimi metodami zelo težko oz. sploh ni možno, saj to zahteva upoštevanje tudi večdimenzionalnih nemonotonih relacij med spremenljivkami (Christmann, 2004). Z metodami podatkovnega rudarjenja pa lahko pogosto v hitrejšem času zgradimo kakovostnejše modele (Steinberg, 2008).

1.3 Določanje cene tveganja pri zavarovanju AO

Zavarovalna premija je cena za prevzem tveganja. Zavarovalec zavarovalnici plača kosmato premijo, ki naj bi bila izračunan tako, da na daljši rok zadošča za izplačilo odškodnin, kritje stroškov zavarovalnice in dobiček (Komelj, 2012). Zavarovalne premije so tesno korelirane z oceno tveganja pri zavarovalnem poslu, zato v tem podpoglavju na kratko predstavljam tudi postopek in dejavnike, ki vplivajo na določitve premij.

V praksi kosmato premijo (to je premija, ki jo plača zavarovalec) določimo z naslednjimi koraki (Komelj, 2012):

- določimo dovolj velike in dovolj homogene nevarnostne skupine, za katere bomo računali zavarovalno premijo (opisano kot segmentacija v prejšnjem poglavju),
- analiziramo zgodovinske podatke o pogostosti in višini odškodnin,
- določimo željeno končno razmerje med odškodninami in premijami,
- upoštevamo trende odškodnin,
- upoštevamo spremembe tveganj in zavarovalnih pogojev ter spremembe kritij,
- upoštevamo pričakovani delež stroškov in načrtovane prihodke iz finančnih naložb,
- dobljeno premijo primerjamo s premijami konkurence in trenutnimi premijami,
- pri končni odločitvi upoštevamo komercialne momente.

Premijo se lahko znotraj posamezne nevarnostne skupine še dodatno prilagodi na podlagi preteklih izkušenj (upoštevanje škodnega dogajanja preteklega leta pri določanju premije za

naslednje leto, sistemi bonusov/malusov …), vendar je potrebno paziti, da ne pride do pojava antiselekcije. Pogosto tudi dejanski škodni količnik prilagajamo ciljnemu (Komelj, 2012).

Na stopnjo tveganja pri AO pogodbah oz. posledično na višino zavarovalne premije vplivajo določeni elementi. Le-te delimo na objektivne (nanašajo se na vozilo) in na subjektivne (nanašajo se na osebo) in so pri AO naslednji (Fric & Prijanovič, 2010, str. 21):

- vrsta vozila (osebna, tovorna, avtobusi, vlačilci, motorna kolesa …),
- tehnične lastnosti vozila (moč motorja, nosilnost, prostornina motorja, masa vozila …),
- namembnost vozila (gasilsko, rent-a-car …),
- lastništvo vozila (pravna ali fizična oseba),
- zavarovalni kraj,
- škodno dogajanje (število škodnih dogodkov v zgodovini, škodno razmerje),
- osnovni podatki o vozniku (spol, starost …),
- vozniške izkušnje zavarovanca,
- oblika zavarovanja,
- višina kritja (minimalne ali povečane zavarovalne vsote),
- obseg kritja,
- trajanje zavarovanja,
- druga tveganja opredeljena v premijskem sistemu (lastna hiša, alarm, garaža, opravljen tečaj varne vožnje …).

Zavarovalna polica mora vsebovati vse podatke, ki vplivajo na izračun premije. V nekaterih državah Evrope poleg zgoraj naštetih upoštevajo še druge elemente tveganja, npr. stopnja izobrazbe zavarovalca, poklic zavarovalca, vrsta zavor, pospeški, pri določenih vrstah vozil celo barva, ipd. (Komelj, 2012). Vsak element tveganja je predstavljen kot atribut v bazi podatkov in lahko bistveno vpliva na višino premije.

2 ORODJA IN METODE PODATKOVNEGA RUDARJENJA

V tem poglavju predstavljam orodja, metode in teoretične osnove podatkovnega rudarjenja, ki so predmet analize primernosti v tem delu. V prvem podpoglavju sem navedla osnovne pojme podatkovnega rudarjenja, jedro tega poglavja pa predstavlja podpoglavje 2.2, kjer sem opisala metode učenja zakonitosti v podatkih, ki sem jih uporabila pri praktičnem delu. V naslednjih podpoglavjih sem prikazala teoretično podlago za izbor atributov (podpoglavje 2.3), teorijo neuravnoteženih podatkov (podpoglavje 2.4), načine vrednotenja uspešnosti določene metode (podpoglavje 2.5) ter uporabljena orodja (podpoglavje 2.6).

2.1 Osnovni pojmi podatkovnega rudarjenja

Podatkovno rudarjenje nima skupne, splošno sprejete, definicije. Berry & Linoff (2000, str. 7) ga opredelita kot raziskovanje in analiziranje velikih količin podatkov z namenom odkrivanja smiselnih vzorcev in pravil. Ideja rudarjenja podatkov je tudi v tem (Witten & Frank, 2005, str.

xxiii, 4), da se zgradijo računalniški programi, ki po bazah podatkov avtomatsko ali vsaj pol-avtomatsko iščejo smiselne vzorce. Proces je kompleksen, saj zahteva ustrezno razumevanje podatkov, ki jih analiziramo, ustrezno izbiro metod in orodij, smiselno izvedbo izbranih metod z ustreznimi parametri in objektivno evalvacijo rezultatov. Za začetek bom predstavila osnovne koncepte.

Vhod v podatkovno rudarjenje je **nabor podatkov**, ki ga sestavlja množica primerov (angl. *instance*), ki so med seboj neodvisni. Vsak primer je predstavljen z vrednostmi fiksne vnaprej definirane množice atributov (Witten & Frank, 2005, str. 48). **Atribut** (angl. *attribute, feature*) je spremenljivka, ki ima določeno množico možnih vrednosti (Kononenko, 1997, str. 62). Nabor podatkov je torej podatkovna matrika, kjer vrstice predstavljajo vhodne primere, stolpci pa predstavljajo atribute. Vsak primer lahko opišemo kot vektor $x = (x_1, x_2, \dots, x_p)$, kjer so x_i spremenljivke vektorja x, p pa dimenzija vektorja x. Za boljšo ponazoritev: v mojem delu je vhodni primer oz. vektor enak zavarovalni pogodbi, njegove spremenljivke oz. vrednosti atributov pa so lastnosti pogodbe, npr. tip vozila, starost vozila. Atributi so lahko (Hand, Mannila & Smyth, 2001, str. 6):

- **Nominalni** so predstavljeni s kvalitativnimi vrednostmi, ki jih ni mogoče urediti po velikosti. Tudi uporaba logičnih oz. matematičnih operacij je za take atribute nesmiselna. Primer je atribut, ki opisuje vrsto zavarovalca, kjer poimenujemo kategorije fizična oseba, pravna oseba in samostojni podjetnik. Poseben primer so **binarni** ali **Boolovi atributi** (angl. *Boolean*), ki imajo smo dve vrednosti: pravilno in nepravilno oz. da ali ne.
- **Vrstilni** so predstavljeni s kvalitativnimi vrednostmi, vendar jih je mogoče urediti po velikosti, uporaba matematičnih operacije pa ni smiselna. Primer je atribut, ki opisuje moč motorja, kjer imamo kategorije do 30 KW, do 40 KW, …, do 130KW, nad 130KW.
- **Intervalni** imajo za osnovo neko numerično skalo, primere pa lahko na podlagi takega atributa primerjamo z odštevanjem. Primer je atribut leto izdelave avtomobila, kjer je smiselno računati npr. razliko v letih (razlika med leti 2001 in 2007 je 6 let), nesmiselno pa je npr. deljenje let (kvocient let 2001 in 2007 je 100,3 %).
- **Racionalni** atributi imajo poleg numerične skale opredeljeno tudi smiselno ničelno točko. Primere lahko na podlagi tega atributa primerjamo tudi z kvocientom. Primer je atribut masa avtomobila, kjer je kvocient med masama dveh avtomobilov 1000 kg in 1200 kg enak 1,2 oz. drugi avto ima maso 120 % mase prvega avtomobila.

Nominalne in vrstilne atribute imenujemo tudi s skupnim imenom **kategorični** oz. **kvalitativni** atributi, intervalni in racionalni pa predstavljajo **numerične** oz. **kvantitativne** atribute.

Pri podatkovnem rudarjenju poznamo različne **tipe nalog**. Larose (2005, str. 11-17) definira šest osnovnih tipov, ki so večinoma tudi v skladu z ostalimi avtorji:

- **Opis** (angl. *description*) je tip naloge, pri kateri želimo najti dober opis vzorcev in trendov v podatkih. Od metod so primerna npr. odločitvena drevesa, saj omogočajo intuitivno interpretacijo in razlago.

- **Klasifikacija** (angl. *classification*) je naloga, pri kateri se želimo iz množice že klasificiranih primerov naučiti, kako klasificirati še nevidene primere. Tipične metode so *k*-najbližjih sosedov, odločitvena drevesa in nevronske mreže.
- **Regresija** oz. **ocena** (angl. *regression, estimation*) je podobna klasifikaciji, le da je tu ciljna spremenljivka, ki jo napovedujemo, numerična in ne kategorična.
- **Napoved** (angl. *prediction*) je tip naloge podoben klasifikaciji in regresiji, le da se rezultati napovedi nanašajo na prihodnost (npr. napoved cene delnice čez tri mesece).
- Pri **razvrščanju v skupine** (angl. *clustering*) je naloga razvrstiti primere v čim bolj homogene skupine, tako da se maksimira podobnost med zapisi v eni skupini in minimizira podobnost z zapisi v drugi skupini.
- Naloga **asociacijskih pravil** (angl. a*ssociation rule learning*) je iskanje zakonitosti med atributi. Pravila so oblike "če pogoj potem posledica". Tipičen primer je primer napovedovanja nakupovalne košarice (kateri izdelki v trgovini se kupujejo skupaj in kateri se nikoli ne kupujejo skupaj).

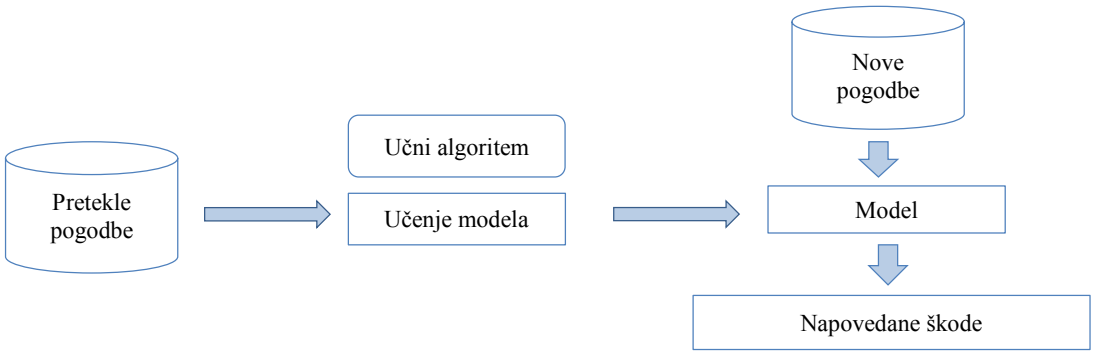
Tip naloge, ki ga rešujem v magistrskem delu, se lahko uvršča tako v klasifikacijo kot v regresijo, zato omenjena tipa nalog predstavljam bolj podrobno.

Klasifikacija je ena najbolj pogosto uporabljenih nalog podatkovnega rudarjenja. Definiramo jo kot diskretno funkcijo f, ki slika iz množice atributov v enega od vnaprej definiranih razredov. Funkcija f je **klasifikacijski model** (Tan, Steinbach & Kumar, 2014, str. 146).

Regresija je zelo podobna klasifikaciji, le da je v tem primeru funkcija f zvezna in slika iz množice atributov v eno od vrednosti, ki jih napovedujemo. V obeh primerih rezultat napovedi imenujemo ciljna spremenljivka, ki je pri klasifikaciji **razred** (angl. *class*), pri regresiji pa **numerična vrednost**. Pri obeh konceptih se iz množice primerov z že znano ciljno spremenljivko naučimo, kako ciljno spremenljivko določiti za nove primere. V postopku učenja zgradimo klasifikacijski oz. regresijski model. Model se najprej nauči, katere kombinacije atributov in na kakšen način določajo vrednost ciljne spremenljivke. Te podatke imenujemo **učni podatki**. V drugi fazi naučen model pregleda nove še nevidene primere, ki nimajo podatka o ciljni spremenljivki (oz. jim ta podatek za namene testiranja odvzamemo), in jim na osnovi naučenega določi vrednost (Slika 1). To množico podatkov imenujemo **testna množica**. Pri klasifikaciji je ciljna spremenljivka kategoričnega tipa (v mojem primeru razreda da oz. realizirana škoda na pogodbi in ne oz. na pogodbi ni bilo škode), pri regresiji pa je ciljna spremenljivka numerična vrednost (v mojem primeru sem naredila preslikavo razredov da in ne v vrednosti 1 in 0).

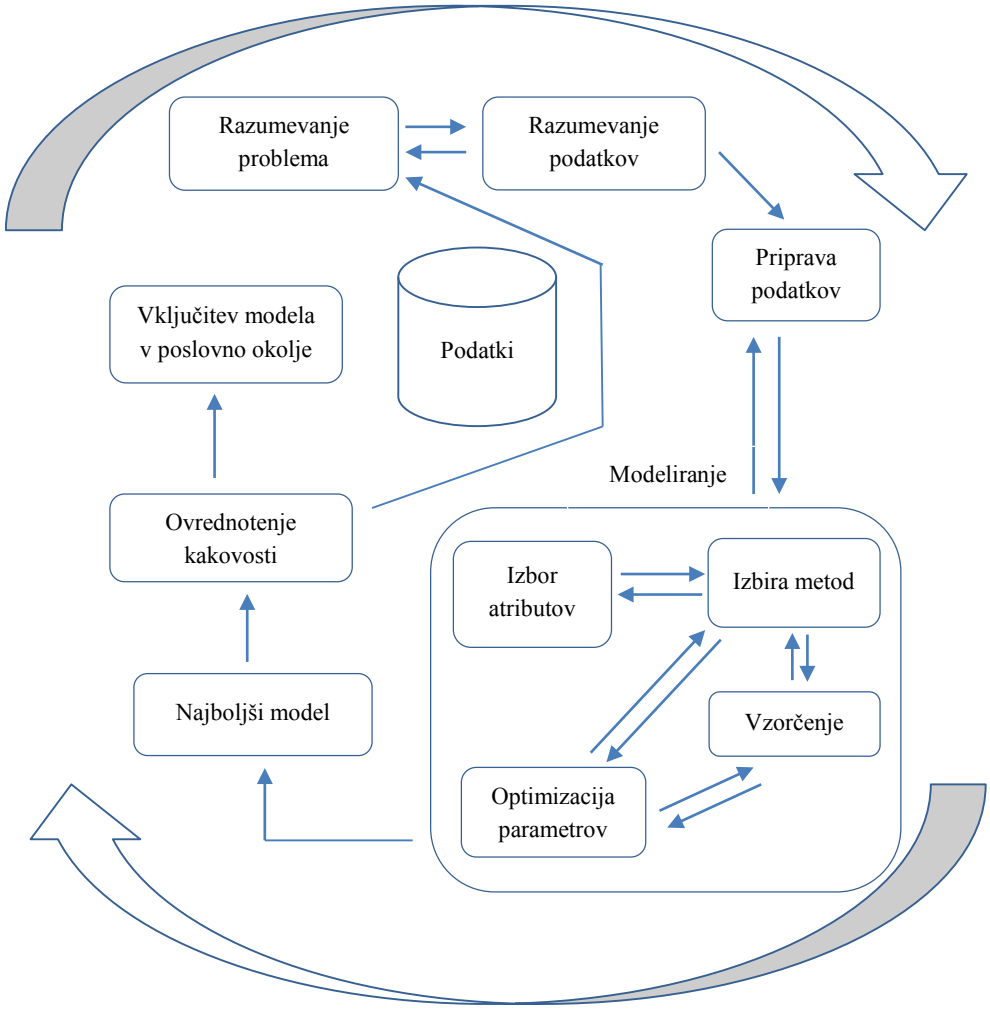
Tipični primeri metod za klasifikacijo so odločitvena drevesa, odločitvena pravila, Naivni Bayesov klasifikator, *k*-najbližjih sosedov, za regresijo pa modelna in regresijska drevesa, linearna regresija, logistična regresija, SVM, nevronske mreže in hibridni algoritmi.

Slika 1: Splošni pristop klasifikacije oz. regresije



Kot sem omenila že v uvodnem poglavju, sem pri reševanju problema sledila CRISP-DM pristopu. Konceptualno je pristop razmeroma svoboden in predvideva prilagajanje zaporedja in ponovitve faz glede na potrebe specifičnega podatkovnega rudarjenja. Pri svojem delu sem ta

Slika 2: CRISP-DM pristop prilagojen za problem tega magistrskega dela



Vir: Prilagojeno po I. Kononenko & M. Kukar, Machine Learning and Data Mining: Introduction to Principles and Algorithms, 2007, str. 3, slika 1.2.

proces prilagodila dejanskim potrebam (Slika 2). Vsako prilagoditev sem opisala v besedilu in s tem omogočila ponovljivost in ponovno uporabo spoznanj iz tega magistrskega dela.

2.2 Metode učenja zakonitosti v podatkih

V tem poglavju predstavljam nekaj metod učenja podatkovnega rudarjenja, ki so primerne za klasifikacijo in regresijo in so se pri mojem testiranju izkazale za uspešnejše. Preizkusila pa sem tudi večje število drugih metod, ki pa zaradi različnih razlogov niso dale dobrih rezultatov in jih zato v tem poglavju ne omenjam. Uporabila sem **netransparentne** (angl. *black box*) metode oz. algoritme, kjer poznamo vhodne in izhodne podatke ter osnovni princip delovanja, o samem notranjem delovanju algoritma pa ne vemo ničesar, in **transparentne** (angl. *white box*) metode oz. algoritme, ki so nasprotje prvih, saj njihov rezultat razkrije strukturo vzorca v podatkih. Med netransparentne metode uvrščamo npr. nevronske mreže, najbolj tipičen predstavnik transparentnih metod pa so odločitvena drevesa.

V tem delu uporabljam izraz metoda, ko predstavljam splošnejše koncepte podatkovnega rudarjenja (npr. metoda odločitvenih dreves), ter izraz algoritem, ko se sklicujem na točno določen algoritem oz. celo točno določeno implementacijo algoritma določene metode (npr. algoritem C4.5, ki je najbolj znan predstavnik metode odločitvenih dreves).

2.2.1 Naivni Bayes

Naivni Bayes je zelo preprost pristop k učenju zakonitosti v podatkih. V metodi se uporabijo vsi atributi, ki enako prispevajo h končni odločitvi in so enako pomembni in neodvisni (ime *naivni* izhaja ravno iz "naivne" predpostavke o neodvisnosti). To je seveda precej nerealna predpostavka, ki v splošnem ne drži, ampak v praksi se je izkazalo (Witten & Frank, 2005, str. 91), da Naivni Bayes da zelo dobre rezultate in velikokrat se zgodi, da celo preseže bolj zahtevne klasifikacijske metode. Zelo dobro deluje v kombinaciji z metodami za selekcijo atributov, ki temeljijo ravno na tem, da odstranijo odvečne oz. zelo korelirane (posledično odvisne) atribute, saj le-ti pokvarijo proces učenja. Če sta npr. dva atributa zelo podobna, se njun učinek multiplicira. Kjer torej obstajajo močne odvisnosti med atributi, Naivni Bayesov klasifikator odpove (Kononenko, 1997, str. 163).

Metoda temelji na Bayesovemu pravilu za pogojno verjetnost. Naj bo C naključna spremenljivka, ki označuje razred, X pa vektor naključnih spremenljivk, ki pomenijo atribute. Verjetnost, da določen testni primer x pripada razredu c, je pogojna verjetnost

$$p(\boldsymbol{X} = x|C = c) = p\left(\bigwedge_i X_i = x_i|C = c\right) = \prod_i p(X_i = x_i|C = c). \qquad (1)$$

Dogodek $\boldsymbol{X} = x$ se zaradi neodvisnosti atributov lahko zapiše kot $X_1 = x_1 \bigwedge X_2 = x_2 \bigwedge \ldots \bigwedge X_k = x_k$. Če je vektor x zvezen, potem namesto verjetnosti p, uporabimo gostoto porazdelitvene funkcije $f(\boldsymbol{X} = x|C = c)$. Največkrat je to normalna porazdelitev

$$f(\boldsymbol{X} = x|C = c) = g(x; \mu_c, \sigma_c) = \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}}, \qquad (2)$$

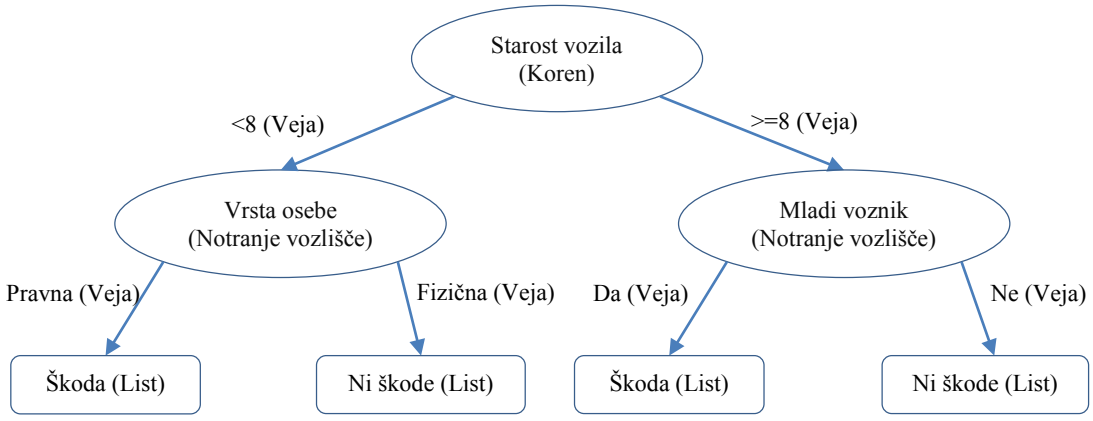
samega izračuna pa ne moti, če je porazdelitev mešana, lahko pa uporabimo celo procedure za t. i. oceno gostote jedra (ang. *kernel density estimation*), kjer se ne zahteva nobena posebna porazdelitev (John & Langley, 1995). Prav tako se lahko uporabi kombinacija zveznih in diskretnih spremenljivk.

V praktičnem delu sem uporabila Wekin klasifikacijski algoritem **Naive Bayes** (John & Langley, 1995).

2.2.2 Odločitvena drevesa

Odločitveno drevo (angl. *decision tree*) je sestavljeno iz notranjih vozlišč, ki ustrezajo atributom, iz vej, ki ustrezajo podmnožicam vrednosti atributov, in iz listov, ki ustrezajo razredom (Slika 3). Ena pot v drevesu od korena do lista ustreza enemu odločitvenemu pravilu (Kononenko, 1997, str. 147). Vozlišča v drevesu vključujejo testiranje določenega atributa. List drevesa pomeni klasifikacijo določenemu razredu, ki velja za vse primere, ki dosežejo list. Zgrajeno drevo se uporabi za klasifikacijo novih primerov, ki potujejo po drevesu glede na vrednosti atributov, ki se testirajo v vozliščih, vse do lista drevesa, ki pomeni razvrstitev v razred, ki je določen temu listu. List vsebuje informacijo o številu učnih primerov iz posameznih razredov.

Slika 3: Prikaz odločitvenega drevesa



Pri gradnji odločitvenega drevesa je ključnega pomena izbira najboljšega atributa. Za izbiro se najpogosteje uporabljajo mere nečistoče: informacijski prispevek, razmerje informacijskega prispevka (angl. *information gain*), Gini-indeks in ReliefF (Kononenko, 1997, str. 149).

Pogosto se zgradijo **binarna odločitvena drevesa**, kjer ima vsako vozlišče povezavi na največ dve drugi vozlišči, na svojega levega in desnega sina. Ta drevesa so manjša in imajo po navadi tudi boljšo klasifikacijsko točnost (Kononenko & Kukar, 2007, str. 229). Velikokrat se enostavna odločitvena drevesa bolje izkažejo kot pa kompleksnejša, zato se slednja poreže.

Poznamo dva splošna pristopa: rezanje vnaprej (angl. *prepruning*) in naknadno rezanje (angl. *postpruning*). Pri naknadnem rezanju se zgradi celotno drevo in šele nato poreže (odstrani se določene veje). Pri rezanju vnaprej pa se drevo poreže že med samim procesom graditve. Uporablja se tudi kombinacija obeh pristopov. Naknadno rezanje potrebuje več računanja kot rezanje vnaprej, vendar se je v praksi izkazalo boljše (Han & Kamber, 2001, str. 290).

Zelo znan primer algoritma za metodo odločitvenih dreves, ki se ga uporablja za klasifikacijo, je algoritem **C4.5**. (Quinlan, 1993). V praktičnem delu sem uporabila implementacijo **J48**, ki je odprto-kodna implementacija algoritma C4.5 v Javi oz. Weki. C4.5 zgradi odločitveno drevo iz učnih podatkov na konceptu informacijske entropije, kar pomeni, da je kriterij za razdelitev v podmnožice informacijski prispevek atributa (Quinlan, 1993, str. 17 - 24).

2.2.3 Prekrivni modeli na osnovi pravil – klasifikacijska pravila

Klasifikacijska pravila so oblike "če – potem" (angl. *if – then*). Prvi del "če" je pogoj in predstavlja množico testov za podatke (podobno kot testi v vozliščih odločitvenih dreves), drugi del "potem" pa vrne oznako razreda ali pa verjetnost porazdelitve določenega razreda za primere, ki ustrezajo določenemu pravilu. Pravila lahko izpeljemo ali direktno iz učnih podatkov ali pa indirektno tako, da zgradimo odločitveno drevo in ga pretvorimo v pravila, lahko pa se novo znanje naučimo z nevronskimi mrežami, iz le-teh pa naredimo izvleček pravil (Lanzi, 2007). Izpeljava iz odločitvenih dreves velikokrat ni trivialna. Drevesa se gradi tako, da se na vsakem koraku izbere en atribut in se na osnovi le-tega deli na poddrevesa. Tako lahko dobimo zelo veliko drevo, medtem ko je ekvivalenten problem rešljiv z odločitvenimi pravili, ki so precej kompaktnejša (Witten & Frank, 2005, str. 67).

Po direktni poti dobimo odločitveno pravilo tako, da pregledamo vsak razred posebej in poskušamo zajeti vse primerke v razredu, hkrati pa izločiti vse primerke, ki ne gredo v ta razred. Ta postopek se imenuje prekrivni, ker se na vsakem koraku identificira pravilo, ki prekrije določene primerke (Witten & Frank, 2005, str. 105). Prekrivni modeli na osnovi pravil delujejo tako, da se v vsakem koraku gradnje pravil ustvari pravilo z največjo natančnostjo. To je ravno v nasprotju s principom odločitvenih dreves, kjer se drevo gradi tako, da se čimbolj ločijo razredi. V obeh primerih je potrebno najti atribut, ki naredi najboljšo razdelitev, le kriterijska funkcija je različna. Npr. pri drevesu C4.5 je to razmerje informacijskega prispevka, pri pravilih pa verjetnost želene klasifikacije.

V primerjavi z odločitvenimi drevesi so odločitvena pravila predvsem lažja za razumevanje, po drugi strani pa jih zelo zmotijo napake v podatkih, ki pa so v podatkih iz realnega sveta zelo pogoste. Težave se pojavijo tudi s prenasičenostjo, ki pa se jo reši z različnimi pristopi rezanja pravil: naknadno rezanje, rezanje vnaprej, rezanje z zmanjšanjem napake (angl. *reduced error pruning*) (Holzapfel & Schmidt, 2003). Pri praktičnem delu sem uporabila algoritem **RIPPER** (angl. *repeated incremental pruning to produce error reduction*) oz. JRip, ki je njegova implementacija v Weki (Cohen, 1995). RIPPER uporablja postopno rezanje pravila takoj, ko je zgrajeno (angl. *incremental reduced-error pruning*) (Witten & Frank, 2005, str. 203).

2.2.4 *K*-najbližjih sosedov

Metode najbližjih sosedov uvrščamo med metode **lenega učenja** (angl. *lazy learning*). Pri teh metodah skorajda ni učenja, tu se le shranijo učni primeri ali pa njihove podmnožice. Ko se klasificira nov primer, se poišče podmnožica najbolj podobnih učnih primerov, ki se uporabijo za napoved razreda novega primera (Kononenko, 1997, str. 173).

Pri najpreprostejši varianti metode ***k*-najbližjih sosedov** (angl. *k-nearest neighbours*) ali *k*-NN se shranijo vsi učni primeri, izbrana metrika razdalje pa se uporabi za določitev, katerih *k* primerov učne množice je najbližje neznanemu testnemu primeru. Ko so najbližji učni primeri najdeni, se pri klasifikaciji napove večinski razred c_x, tj. razred, ki mu pripada največ izmed *k* najbližjih sosedov

$$c_x = \underset{c \in \{C_1,\ldots,C_{n_0}\}}{argmax} \sum_{i=1}^{k} \delta(c, c_i), \tag{3}$$

kjer je $\{C_1, \ldots, C_{n_0}\}$ množica vseh možnih razredov in

$$\delta(a, b) = \begin{cases} 1, & a = b \\ 0, & a \neq b \end{cases}. \tag{4}$$

Pri regresiji pa se napove povprečna vrednost razreda vseh *k* najbližjih sosedov

$$c_x = \frac{1}{k} \sum_{i=1}^{k} c_i, \tag{5}$$

kjer je *k* parameter, ki se ga običajno nastavi na neko liho število (npr. 1, 5, 7, 15). Če so učni primeri brez napak, potem po navadi najbolje dela algoritem 1-NN. Če pa so v učnih primerih tudi napake, potem je bolje izbrati večji *k*, saj se tako povpreči napovedi več bližnjih primerov, hkrati pa tudi zmanjša verjetnost, da je vseh *k* učnih primerov napačnih. V vsakem primeru pa je smiselno, da se za vsak učni problem posebej določi optimalni parameter *k* (Kononenko, 1997, str. 174). Velikokrat se pri *k*-NN uporabi tudi uteži prispevkov sosedov, tako da sosedi, ki so bližje, prispevajo več kot pa tisti bolj oddaljeni (Kononenko, 1997, str. 176).

V praktičnem delu sem uporabila Wekin klasifikacijski algoritem **IBk**, ki je implementacija *k*-NN z nekaj izboljšavami. IBk ne shranjuje vseh učnih primerov, ampak zgolj napačno klasificirane primere, kar bistveno zmanjša zahteve po shranjevanju. Prednost je pa tudi ta, da dela boljše s podatki, ki vsebujejo napake (Aha, Kibler & Albert, 1991).

2.2.5 Linearni modeli

Najbolj tipičen primer linearnih modelov je **linearna regresija**, ki se uporablja, če so vsi atributi in razred numeričnega tipa. Pri linearni regresiji iščemo tako funkcijo, ki je linearna

kombinacija vseh atributov oz. neke podmnožice atributov. Razred izrazimo kot linearno kombinacijo atributov z določenimi utežmi

$$x = \beta_0 a_0 + \beta_1 a_1 + \beta_2 a_2 + \cdots + \beta_k a_k = \sum_{j=0}^{k} \beta_j a_j = \boldsymbol{\beta}^T \boldsymbol{a}, \qquad (6)$$

kjer je x razred, a_0, a_1, …, a_k so vrednosti atributov, β_0, β_1, …, β_k so uteži (tu predpostavimo, da imamo atribut a_0 z vrednostjo 1). Uteži se izračunajo iz učne množice podatkov. Naloga linearne regresije je dobiti take koeficiente β_j, da minimizirajo vsoto kvadratov razlike med napovedano in dejansko vrednostjo razreda po vseh učnih primerih. Če imamo n učnih primerov, tako dobimo razliko vsote kvadratov

$$\sum_{i=1}^{n} \left(\boldsymbol{x}^{(i)} - \sum_{j=0}^{k} \beta_j a_j^{(i)} \right)^2, \qquad (7)$$

kjer je izraz v oklepaju razlika med dejanskim i-tim razredom primera i ter njegovim napovedanim razredom. Minimizirati je potrebno razliko v enačbi (7).

Linearna regresija je uspešna in enostavna metoda za numerično napoved, ki se že desetletja uporablja tudi v statistiki. Problem se pojavi, če podatki kažejo nelinearno odvisnost. V tem primeru se izračunana najbolje prilegajoča se premica ne bo najbolje ujemala. Ne glede na to pa so linearni modeli dobra osnova za kompleksnejše učne metode.

Logistična regresija se uporablja pri klasifikaciji, kjer je razred, ki ga napovedujemo, binarnega tipa. Metoda vrne razmerje med razredom in eno ali več neodvisnimi spremenljivkami, tako da uporabi verjetnostne ocene o predvidenih vrednostih odvisne spremenljivke (Putler & Krider, 2012, str. 117). Logistična regresija generira koeficiente β za napoved logit transformacije verjetnosti določenega dogodka (Gullickson, 2005)

$$logit(p) = \beta_0 a_0 + \beta_1 a_1 + \beta_2 a_2 + \cdots + \beta_k a_k, \qquad (8)$$

kjer je p verjetnost dogodka in

$$logit(p) = log \frac{p}{1-p}. \qquad (9)$$

Enačbi (8) in (9) preoblikujemo in dobimo

$$\frac{p_i}{1-p_i} = e^{\beta_0} e^{\beta_1 x_{i1}} e^{\beta_2 x_{i2}} \ldots e^{\beta_n x_{in}}, \qquad (10)$$

kjer izraz na levi strani enačbe (10) pomeni število uspešnih izvedb dogodka za vsako neuspelo izvedbo, ki se jo v povprečju pričakuje.

Metoda je poseben primer **generaliziranih linearnih modelov** (angl. *generalized linear models*, GLM), ki so poseben primer linearne regresije (Kononenko & Kukar, 2007, str. 267).

Pri praktičnem delu sem uporabila Wekin algoritem za logistično regresijo **Logistic** (le Cessie & van Houwelingen, 1992).

2.2.6 Modelna drevesa

Modelna drevesa so zelo podobna odločitvenim drevesom, le da imajo v listih nek linearni regresijski model. Ta drevesa so analogna odsekoma linearnim funkcijam. **Regresijsko drevo** je le poseben primer modelnih dreves, saj ima v listih vrednost razreda, ki predstavlja povprečno vrednost vseh primerov, ki dosežejo list (Quinlan, 1992).

Zelo znana metoda modelnega drevesa je **M5**, ki jo je razvil J. R. Quinlan leta 1992. M5 je znan po učinkovitem učenju in reševanju nalog, kjer je zelo veliko atributov, tudi do 100 in več. Tudi manjkajoče vrednosti ga ne motijo. Prednost M5 je pa tudi v tem, da so ta drevesa v splošnem precej manjša kot pa regresijska drevesa, poleg tega so v praksi dosegla večjo natančnost.

Modelna drevesa se konstruira podobno kot odločitvena. V notranjem vozlišču v drevesu se testira določeni atribut. Pri M5 je kriterij za delitev standardna deviacija vrednosti razreda vseh primerov iz učne množice. Izračuna se zmanjšanje standardne deviacije ali SDR (angl. *standard deviation reduction*), za delitev pa se izbere tisti atribut, ki maksimira SDR. Delitev se zaključi, ko se vrednosti primerov določenega razreda, ki dosežejo vozlišče, zelo malo razlikujejo, kar pomeni, da je njihova standardna deviacija le majhen delež (npr. manj kot 5 %) standardne deviacije originalne množice podatkov. Delitev se zaključi tudi, če ostane le nekaj primerov. Eksperimenti kažejo, da dobljeni rezultati niso občutljivi na natančno izbiro teh pragov (Witten & Frank, 2005, str. 245). Pri modelnem drevesu je v vsakem notranjem vozlišču in v listih nek linearni model oblike

$$w_0 + w_1 a_1 + w_2 a_2 + \cdots + w_k a_k, \qquad (11)$$

kjer so a_1, …, a_k vrednosti atributa, uteži w_1, …, w_k pa se izračunajo s standardno regresijo (glej podpoglavje 2.2.5). V regresiji se uporabijo samo atributi, ki se testirajo v poddrevesu tega vozlišča, ostali atributi so se upoštevali pri testih, ki vodijo na to vozlišče (Witten & Frank, 2005, str. 245).

V procesu podatkovnega rudarjenja sem uporabila algoritem **M5P**, ki je rekonstrukcija M5 algoritma v Weki (Quinlan, 1992).

2.2.7 Nevronske mreže

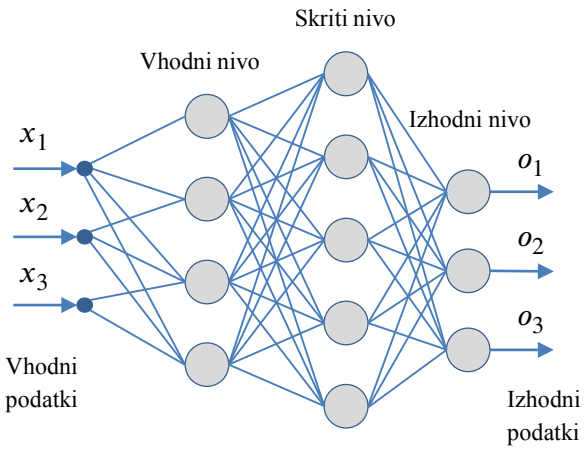
Nevronske mreže so sestavljene iz številnih neodvisnih in enostavnih procesorjev, ki jih imenujemo nevroni. V vsakem nevronu se shrani aktivacijska vrednost. Nevroni med sabo komunicirajo preko uteženih sinaptičnih povezav. Nevronske mreže se razlikujejo glede na to,

kako so nevroni povezani, glede na način, kako nevroni procesirajo vhodne podatke, in glede na metodo, ki jo uporabljajo. Zato je potrebno tudi izbrati ustrezno strukturo. Poleg tega je potrebno prevesti problem v tako obliko, da se ga da rešiti z nevronskimi mrežami. To pomeni, da je potrebno vse nenumerične atribute pretvoriti v numerične. V primeru klasifikacijskega problema je potrebno razrede spremeniti v binarni vektor, tako da je lahko vsak izhodni nevron predstavljen z enim razredom. Prav tako je potrebno spremeniti kategorične atribute. Šele potem se lahko konstruira mreža, ki ima en vhodni nevron za vsak atribut in en izhodni nevron za vsak razred (Klösgen & Żytkow, 2002, str. 305).

Prednost nevronskih mrež je ta, da omogočajo modeliranje kompleksnejših, nelinearnih problemov, največja slabost pa težavnost razlage njihovih odločitev, saj je končni rezultat skorajda nemogoče razložiti v povezavi z vhodnimi podatki. Nevronske mreže namreč uvrščamo med tako imenovane netransparentne algoritme.

Zelo znana metoda nevronskih mrež je **večnivojski perceptron** (angl. *multilayer perceptron*, v nadaljevanju **MLP**) ki je sestavljen iz enega vhodnega nivoja, enega ali več skritih nivojev in enega izhodnega nivoja nevronov (Slika 4). Enote vhodnega nivoja ne izvajajo nobenih izračunov, ampak samo posredujejo vrednosti svojih vhodnih podatkov. Ostali nevroni v mreži so navadne procesorske enote, ki kombinirajo več vhodov k enemu izhodu. MLP je razširitev enostavnega perceptrona, ki ga je predstavil Rosenblatt leta 1957 in ki je sestavljen iz samo enega vhodnega nivoja in enega izhodnega nivoja nevronov in nima nobenega skritega nivoja. Razširitev na skrite nivoje, ki sta jo predstavila Minsky in Papert leta 1969, je rešila tako imenovane linearno neločljive probleme (angl. *linearly nonseparable problems*) (Klösgen & Żytkow, 2002, str. 306).

Slika 4: Večnivojski perceptron



Vir: I. H. Witten & E. Frank, Data Mining: Practical Machine Learning Tools and Techniques (2nd ed.), 2005, str. 411, slika 10.20.

Pri praktičnem delu sem uporabila Wekin algoritem **MLP** (Witten & Frank, 2005).

V nadaljevanju predstavljam še **Bayesove nevronske mreže**. Osnovna ideja je uporabiti Naivni Bayesov klasifikator kot aktivacijsko funkcijo znotraj vsakega nevrona (Kononenko, 1997, str. 212). Po drugi strani pa je lahko ime Bayesova mreža zavajajoče, saj modeli Bayesovih mrež pogostokrat ne uporabljajo Bayesove statistike, čeprav le-ta zagotavlja učinkovit pristop proti prezasičenosti podatkov (Ben-Gal, 2007).

Bayesova mreža je mreža vozlišč, kjer vsako vozlišče predstavlja en atribut oz. spremenljivko. Vozlišča so med seboj povezana z usmerjenimi povezavami tako, da ni ciklov. Znotraj vsakega vozlišča se nahaja tabela pogojnih verjetnosti, ki se uporabi za napoved verjetnosti razreda za vsak primer. Povezave predstavljajo direktno odvisnost med spremenljivkami in so predstavljene kot puščice med vozlišči. Tako puščica od X_i k X_j pomeni, da je vrednost spremenljivke $X_j$odvisna od vrednosti spremenljivke X_i oz. X_i vpliva na X_j. Vozlišče X_i imenujemo tudi starš (oče) vozlišča X_j oz. X_j je otrok X_i; pogostokrat se uporabljata tudi izraza potomci oz. predniki (Ben-Gal, 2007).

V praktičnem delu sem uporabila Wekin klasifikacijski algoritem **Bayes Net** (Bouckaert, 2008).

2.2.8 Metoda podpornih vektorjev

Metoda podpornih vektorjev (angl. *Support Vector Machine*, v nadaljevanju SVM) je ena izmed uspešnejših metod strojnega učenja, primerna tako za regresijo kot za klasifikacijo (Kononenko & Kukar, str. 267, 268). Atributi se za napoved ciljne spremenljivke uporabijo v linearni kombinaciji. Metoda je primerna za velike množice podatkov z velikim številom atributov. Doseže lahko zelo visoko natančnost, čeprav je interpretacija naučenega znanja velikokrat težka.

Osnovna ideja je ta, kako v prostor atributov postaviti hiperravnino, da bo optimalno ločevala razrede med seboj. Ta ravnina je enako oddaljena do najbližjih primerov vseh razredov, učni primeri, ki so najbližji optimalni hiperravnini, pa so podporni vektorji. Razdalja med hiperravnino in podpornimi vektorji je stopnja, optimalno ravnino pa se določi tako, da se maksimira to stopnjo. Rešitev je klasifikacijsko pravilo za optimalno hiperravnino.

Velikokrat pa se razredov ne da ločiti z linearno hiperravnino, zato se uporabi ne-linearno transformacijo prostora atributov v kompleksnejšega, da lahko razrede ločimo tudi v več dimenzionalnem prostoru atributov. Prednost SVM je v tem, da se transformacije izrecno ne izvedejo, dovolj je že, da se izračuna skalarni produkt podpornih vektorjev z novim primerov v transformiranem prostoru atributov. Pri tem se uporabi jedrna funkcija (angl. *kernel function*). Za jedrno funkcijo se lahko uporabi več različnih funkcij, kar posledično spremeni transformacijo in pomeni tudi svojo SVM metodo.

Podrobnejše izpeljave enačb ne navajam, saj to presega okvir tega dela, le-te najdemo v ustrezni literaturi (Kononeko & Kukar, 2007, str. 267-273).

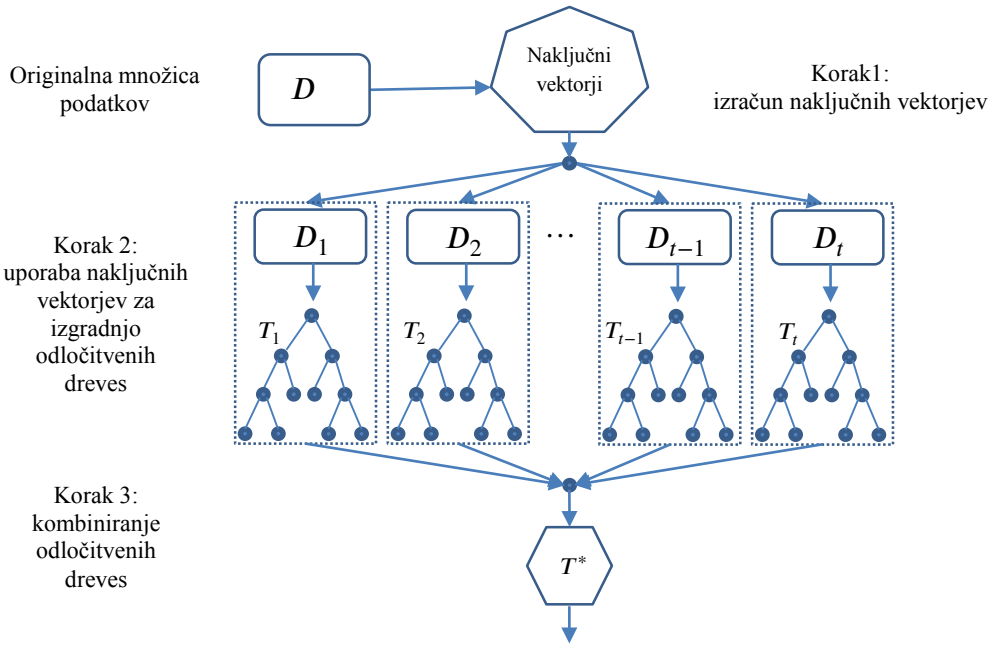
V praktičnem delu sem uporabila algoritem **SMO** (angl. *Sequential Minimal Optimization*) z jedrno funkcijo RBF Kernel. SMO je implementacija Plattovega algoritma za učenje klasifikatorja podpornih vektorjev (Platt, 1998).

2.2.9 Metode klasifikacijskih kombinacij

Metode klasifikacijskih kombinacij (angl. *classifier combination methods*) delujejo tako, da iz testne množice konstruirajo množico osnovnih klasifikatorjev in izvedejo klasifikacijo tako, da glasujejo o napovedi, ki jo da vsak osnovni klasifikator. Te metode dajo v praksi pogosto boljše rezultate kot pa posamezni klasifikatorji (Tan et al., 2014, str. 276). Najbolj znane klasifikacijske kombinacijske metode so "**bagging**", "**boosting**" in **Naključni gozdovi**. Slednje sem uporabila v praktičnem delu.

Naključni gozdovi (angl. *Random Forest*) so zasnovani kot klasifikator odločitvenih dreves. Kombinirajo se napovedi več odločitvenih dreves, kjer je vsako drevo generirano glede na vrednosti neodvisne množice naključnih vektorjev. Ti vektorji so generirani iz fiksne verjetnostne porazdelitve (Slika 5), naključnost pa pomaga zmanjšati korelacije med drevesi. Naključni vektor se vključi v izgradnjo drevesa na več načinov, ki pa jih tu ne bom podrobneje navajala. Vsako odločitveno drevo se zgradi iz celotne učne množice in se uporabi pri klasifikaciji novih primerov z enotnim principom glasovanja. Vsako odločitveno drevo glasuje za razred novega primera, vsi glasovi skupaj pa tvorijo razred verjetnostne porazdelitve (Kononenko & Kukar, 2007, str. 95).

Slika 5: Naključni gozdovi



Vir: P.-N. Tan et al., Introduction to Data Mining (1st ed.), 2014, str. 292, slika 5.40.

Pri praktičnem delu sem uporabila Wekin algoritem **Random Forest** (Breiman, 2001).

2.3 Izbor atributov

Pomemben del pri podatkovnem rudarjenju predstavlja **izbor atributov** (angl. *attibute selection, feature selection*), ki zajema izbor pomembnih atributov ter odstranitev odvečnih oz. nepomembnih atributov. Dimenzije podatkov se s tem zmanjšajo, to pa omogoča hitrejše in učinkovitejše delovanje algoritmov. S tem se tudi model poenostavi, postane bolj transparenten, poleg tega ga je lažje interpretirati. Poznavanje pomembnih atributov je že v osnovi koristno za uspešno uporabo modela.

V praksi se pogosto zgodi, da imamo preveč atributov. Preveliko število atributov lahko precej poslabša model. Težava je v tem, ker je na prvi pogled težko ugotoviti, kateri atributi so pomembni. Večina algoritmov je izdelanih tako, da se naučijo, kateri so najprimernejši atributi. Toda v praksi velja, da preveč nepomembnih atributov "zmede" algoritem. Zaradi negativnega učinka, ki ga imajo nepomembni atributi na algoritem, je smiselno izvesti fazo, v kateri prečistimo množico atributov tako, da v njej zadržimo le najpomembnejše atribute.

Iskanja pomembnih atributov se lotimo na več načinov. Najenostavnejši način je ročna izbira. V tem primeru moramo zelo dobro poznati problem, ki ga rešujemo, in pomen posameznih atributov. Lahko pa uporabimo avtomatične metode (Witten & Frank, 2005, str. 289). Večina teh metod temelji na nekem požrešnem iskanju po množici vseh atributov tako, da dodaja ali odstranjuje atribut za atributom. Najenostavnejša požrešna metoda je metoda **dodajanja atributov** (angl. *forward selection*). Metoda začne s prazno množico atributov, potem pa v vsakem koraku požrešno doda po en atribut. To se ponavlja, dokler natančnost modelov naučenih na tej povečujoči se množici atributov raste. Ko se natančnost preneha povečevati, se lahko ustavimo, saj novi atributi ne prinesejo nič dodatnega znanja.

Druga požrešna metoda je metoda **odstranjevanja atributov** (angl. *backward elimination*), ki začne z vsemi atributi v množici atributov, potem pa na vsakem koraku požrešno odvzame po en atribut, dokler ni množica atributov dovolj majhna, oziroma natančnost modelov na tej zmanjšujoči se množici ne začne prekomerno padati.

Požrešno dodajanje oz. odstranjevanje atributov pa je lahko precej tvegano, saj se lahko kasneje izkaže, da lahko določen atribut, ki smo ga odvzeli (dodali) pomembno vpliva na rezultat. Za "varnejšo" proceduro se je izkazala t. i. **dvosmerna požrešna procedura**, ki na vsakem koraku doda oz. odvzame določen atribut. Pri tem lahko začne z vsemi atributi v množici, s prazno množico atributov ali pa z naključno množico atributov (Caruana & Freitag, 1994).

Poleg požrešnih metod lahko uporabimo še naslednje metode: "Best-first search" in genetski algoritmi (angl. *genetic algorithms*) (Guyon & Elisseeff, 2003), učenje po metodi najboljšega soseda in učenje z odločitvenimi drevesi (Hall & Holmes, 2003), metoda RELIEF in metoda FOCUS (Caruana & Freitag, 1994), izbor atributov na osnovi korelacije (Hall, 1999) …

Metode za izbor atributov se včasih delijo tudi na metode filtrov (angl. *filters*) in ovojev (angl. *wrappers*) (Pal & Mitra, 2004, str. 62-64). Filtri ocenijo vrednost atributov z uporabo določene funkcije, ki temelji na splošnih značilnostih podatkov. Množica atributov se filtrira tako, da dobimo najbolj obetavno podmnožico, še preden se učenje sploh začne in neodvisno od izbranega algoritma. Pri ovojih pa je učni algoritem "zavit" v proceduro izbire atributov, kar pomeni, da se vrednost atributov oceni s pomočjo učnega algoritma, ki se na koncu uporabi za podatke. Ovoji dajo po navadi boljše rezultate kot filtri, saj se izbira atributov optimizira glede na učni algoritem. Po drugi strani pa so ovoji lahko prostorsko in časovno preveč zahtevni za velike nabore podatkov, ki imajo veliko število atributov. Poleg tega so ovoji vezani na določen algoritem, kar posledično pomeni, da so manj splošni kot filtri in jih je ob zamenjavi učnega algoritma potrebno znova pognati (Hall, 1999, str.3).

Guyon in Elisseeff (2003, str. 1163-1165) podajata nekaj zanimivih rezultatov v povezavi z izbiro atributov:

- Z dodajanjem atributov, za katere sumimo, da so odvečni, lahko dosežemo odstranitev šumov v podatkih in posledično boljše ločevanje razredov. Izkazalo se je, da neodvisni in enako porazdeljeni atributi v resnici niso odvečni.
- Po drugi strani pa so popolnoma korelirani atributi resnično odvečni in z dodajanjem le-teh ne dobimo nobene dodatne informacije.
- Atribut, ki je samostojno popolnoma neuporaben, lahko skupaj z drugimi atributi prinese novo informacijo.

Hall in Holmes (2003) sta pokazala, da izbor atributov v splošnem izboljša uspešnost učnih algoritmov. Hkrati pa sta pokazala tudi, da ni najboljšega pristopa, ki bi ustrezal vsem situacijam. Pri procesu izbora atributov je pomembno razumevanje posameznih tehnik in pristopov, poznavanje prednosti in slabosti učnih algoritmov ter dobro poznavanje podatkov. Pri izboru pristopa za izbor atributov je zato potrebno upoštevati vse omenjene dejavnike.

V tem delu uporabljam poenostavljeno dvosmerno metodo, ki temelji na dveh stopnjah. V prvi stopnji najprej odstranjujem atribute, kar zaradi učinkovitosti ne delam za vsak atribut posebej, ampak odstranim več atributov naenkrat. V drugi stopnji pa atribute dodajam ter s tem popravim morebitne napake preveč odstranjenih atributov iz prve stopnje (to se lahko zgodi, ker z odstranitvijo več atributov naenkrat odstranim tudi morebitno informacijo, ki izhaja iz soodvisnosti med atributi). Tak pristop sem uporabila zaradi prevelike časovne zahtevnosti bolj varnih postopkov, hkrati pa sem se z njim delno zavarovala pred napačnim izborom atributov.

2.4 Neuravnoteženi podatki

Neuravnotežena množica podatkov (angl. *imbalanced dataset*) je množica, v kateri je število primerov, ki pripadajo določenim razredom, bistveno večje od števila primerov, ki pripadajo drugim razredom. Razred, ki ima zelo majhno število učnih primerov, imenujemo manjšinski razred, razred, ki ima večino učnih primerov, imenujemo večinski razred.

Takšne množice podatkov so v resničnem življenju precej pogoste. Razmerje med manjšinskim in večinskim razredom je v teh množicah npr. 1 proti 100, 1 proti 1000 oz. celo še večje (v mojem primeru je število pogodb z realiziranimi škodami glede na število pogodb brez škod enako 2,7 proti 100). V takih primerih nas manjšinski razred bolj zanima kot pa večinski, kar pomeni, da potrebujemo precej visoko stopnjo natančnosti (t. j. delež pravilno napovedanih oz. klasificiranih vrednosti) za manjšinski razred. Večina standardnih algoritmov pa ne upošteva porazdelitve take množice podatkov (Han, Wang & Mao, 2005) in ima posledično visoko stopnjo natančnosti za večinski razred, medtem ko je stopnja natančnosti za manjšinski razred precej nizka. Te metode težijo k temu, da minimizirajo stopnjo napake (t. j. delež nepravilno napovedanih oz. klasificiranih vrednosti), h kateri manjšinski razred prispeva zelo malo (Ganganwar, 2012). Posledično tak algoritem doseže visoko stopnjo natančnosti že, če vsak nov primer uvrsti v večinski razred (Provost, 2000).

Na neuravnoteženih množicah je težko rudariti s podatki. Weiss (2004) navaja najpogostejše težave ter rešitve za te probleme. Ena izmed rešitev je **vzorčenje**, ki je tudi eden izmed najbolj znanih in uveljavljenih pristopov za take probleme. To je proces generiranja množice podatkov, ki vsebuje manj elementov kot originalna množica podatkov (Berry & Linoff, 2000, str. 196). Če želimo izdelati dober model, mora biti podmnožica/vzorec reprezentativen, torej mora biti čim bolj podoben originalni množici podatkov, kar pomeni, da mora vzorec vsebovati vse vrednosti iz originalne množice ter da je porazdelitev teh vrednosti približno enaka kot v originalni množici.

Najpogosteje med metode vzorčenja štejemo tiste metode, pri katerih popravimo porazdelitev učnih primerov tako, da je manjšinski razred dobro zastopan v učni množici. To lahko izvedemo na tri načine (Tan et al, 2014, str.305):

- zmanjša se večinski razred - to imenujemo **podvzorčenje** (angl. *undersampling*),
- poveča se manjšinski razred – to imenujemo **nadvzorčenje** (angl. *oversampling*),
- uporabi se **hibrid** obeh zgornjih pristopov.

Naj bodo primeri iz manjšinskega razreda označeni kot pozitivni, primeri iz večinskega pa negativni (kot je to tudi v našem primeru podatkov).

Pri podvzorčenju se formira učna množica tako, da se v vzorec vzame vse pozitivne primere in enako število negativnih primerov. Pri tem se nekaj informacije izgubi, saj se iz učne množice izpustijo nekateri negativni primeri in tak model ni optimalen. To lahko popravimo tako, da se podvzorčenje nekajkrat ponovi, lahko pa se izvede t. i. fokusirano podvzorčenje, ki upošteva informacijo o negativnih primerih, ki jih lahko eliminiramo.

Nadvzorčenje replicira pozitivne primere, dokler nima učna množica enako število pozitivnih in negativnih primerov. Problem lahko predstavljajo primeri, ki vsebujejo napake, saj jih z nadvzorčenjem lahko prevečkrat repliciramo. V splošnem pa nadvzorčenje ne prinese nobene nove informacije v učno množico. S povečanjem manjšinskega razreda se le prepreči rezanje

določenih delov modela, ki opisuje območja, ki vsebujejo zelo malo učnih primerov. Dodatni pozitivni primeri tudi povečajo čas za izgradnjo modela.

Hibridni pristop je naprednejša metoda vzorčenja, saj se tu uporabi kombinacija obeh pristopov, zmanjšanje večinskega razreda in povečanje manjšinskega razreda, da dobimo željeno porazdelitev razreda. Te metode so se izkazale za uspešnejše. Primer take metode je npr. uporaba fokusiranega podvzorčenja, nadvzorčenje pa se izvede npr. z generiranjem novih pozitivnih primerov v bližini že obstoječih pozitivnih primerov. Drugi precej znani primer je metoda SMOTE (angl. *Synthetic Minority Over-sampling Technique*), kjer se uporabi nadvzorčenje tako, da se manjšinski razred poveča s "sintetičnimi" primeri (Chawla, Bowyer, Hall & Kegelmeyer, 2002). Ta metoda se ni izkazala za preveč uspešno na mojih podatkih, zato je podrobneje ne bom predstavljala.

Ne glede na izbiro metode pa je pristop vzorčenja še vedno precej dovzeten za prenasičenje (angl. *overfitting*). V praksi se pojavi vprašanje, koliko zmanjšati večinski oz. povečati manjšinski razred. Odgovor na to vprašanje je odvisen od podatkov in uporabljene metode podatkovnega rudarjenja, kar pomeni, da se ga določi empirično (Chawla, Japkowicz & Kołcz, 2004). Tudi v mojem primeru se je izkazalo, da sem dobila odgovore, ki so bili vezani na določeno uporabljeno metodo. Nekatere metode so primerne za uravnotežene podatke, druge za neuravnotežene, zato sem principe vzorčenja uporabila določeni metodi primerno.

2.5 Ocenjevanje uspešnosti klasifikacijskih metod

V tem poglavju bom predstavila najbolj znane pristope za ocenjevanje uspešnosti določene klasifikacijske metode. Ti klasični pristopi so zgolj pogojno primerni za zelo neuravnotežene podatke, kakršni so tudi moji, zato bom v drugem podpoglavju navedla primernejše metode za ovrednotenje takih modelov. Vse mere, tako klasične kot prilagojene, temeljijo na dveh množicah primerov, učni množici, na kateri naučimo model, in testni množici, na kateri izračunamo mere uspešnosti modela. Zato najprej predstavljam način izbire teh dveh množic.

2.5.1 Izbira učne in testne množice

Pravilna evalvacija klasifikacijskih metod vedno temelji na uporabi ločenih oz. disjunktnih množic podatkov (učnih in testnih primerov), saj je stopnja natančnosti, ki se izračuna iz množice testnih primerov, ki v času učenja niso vidni algoritmu, bolj nepristranska in zato tudi bolj realna. Torej se učni primer, ki se pojavi v množici učnih primerov, nikoli ne sme pojaviti tudi v množici testnih primerov, saj ga je algoritem že videl in se ga je naučil.

Največji problem pri delitvi na učno in testno množico je najti pravo razmerje za delitev: več primerov vzamemo za učno množico, manj jih ostane za testno množico in posledično moramo časovno kompleksno učenje velikokrat ponoviti, v kolikor želimo statistično signifikantne rezultate - idealno je imeti vsak primer vsaj enkrat v testni množici. Če pa vzamemo več

primerov za testno množico, je izpeljani model manj uporaben in posledično naredi več napak, kot bi jih, če bi mu pokazali več primerov. Tako rezultati ponovno niso statistično pravilni.

Kljub veliko pristopom k opisani dilemi delitve množic pa je na področju rudarjenja podatkov najbolj splošna in sprejeta metoda **prečnega preverjanja** (angl. *Cross validation*), ki temelji na predpostavki, da je vsak primer v testni množici uporabljen samo enkrat (Tan et al., 2014, str. 186). *K*-kratno prečno preverjanje razdeli podatke v *k* enako velikih množic. Vsakokrat, ko se metoda izvede, se ena množica uporabi za testiranje, vse ostale skupaj pa za učenje. Proces se ponovi *k*-krat, tako da se vsaka množica uporabi za testiranje natanko enkrat. Povprečje stopenj napake vseh iteracij se uporabi kot končna stopnja napake. Velikokrat se v praksi pri rudarjenju podatkov uporablja 10-kratno, obstajata pa dva skrajna primera:

- Če je *k* enak 2 (kar je najmanjša vrednost), dobimo "**holdout**" metodo, ki je najenostavnejša in razdeli množico primerov na pol. Najprej je učna množica prva polovica podatkov in je testna množica druga polovica, nato pa se vlogi zamenjata. Prednost te metode je hitrost (model se uči samo dvakrat in na polovici podatkov), slabost pa je slaba reprezentativnost evalvacije, saj učni algoritem vidi le pol podatkov.
- Če je *k* enak *N* (kar je največja vrednost), kjer je *N* število vseh primerov podatkov, je prečno preverjanje, ki ga imenujejo "**izpusti enega**" (angl. *leave-one-out*), časovno lahko zelo zahtevno, če je *N* velik. V tem primeru je testna množica le en primer, ostali so v učni množici. Prednost te metode pa je zelo dobra napoved končne natančnosti modela na celotni množici podatkov.

Dodatni zaplet, katerega se je potrebno zavedati, je tudi način vzorčenja podatkov. Poenostavljeno naključno vzorčenje množic je sicer dobro za zelo velike nabore podatkov, pri manjših množicah primerov pa je potrebno dodatno zagotoviti enako zastopanost primerov različnih razredov v učni in testni množici. Ta zaplet nam reši metoda razslojenega vzorčenja (angl. *stratified sampling*), ki zagotovi, da je zastopanost razredov pravilna v obeh množicah.

V tem delu sem uporabljala izključno 10-kratno prečno preverjanje v kombinaciji z razslojenim vzorčenjem.

2.5.2 Klasična primerjava uspešnosti

Uspešnost modela določimo na osnovi števila pravilnih klasifikacij oz. napovedi (testnih primerov, ki jih je model napovedal pravilno) in nepravilnih klasifikacij oz. napovedi. Ta števila predstavimo s **matriko zamenjav** (angl. *confusion matrix*). V nadaljevanju predstavljam matriko v primeru dveh razredov (Slika 6), kjer predstavlja

- TP število dejanskih pozitivnih primerov, ki so klasificiranih kot pozitivni,
- TN število dejanskih negativnih primerov, ki so klasificirani kot negativni,
- FN število dejanskih pozitivnih primerov, ki so klasificirani kot negativni,
- FP število dejanskih negativni primerov, ki so klasificirani kot pozitivni.

Slika 6: Matrika zamenjav

Matrika zamenjav		Napovedani razredi (angl. *Predicted Classes*)	
		Pozitivni	Negativni
Dejanski razredi (angl. *Actual Classes*)	Pozitivni	TP	FN
	Negativni	FP	TN

Vir: P.-N. Tan et al., Introduction to Data Mining (1st ed.), 2014, str. 149, tabela 4.2.

Od tod za model T definiramo **stopnjo napake** (angl. *error rate*)

$$stopnja\ napake\ (T) = \frac{število\ napačnih\ klasifikacij}{število\ vseh\ klasifikacij} = \frac{FP + FN}{TP + FN + FP + TN} \qquad (12)$$

in **stopnjo natančnosti** (angl. *accuracy*)

$$stopnja\ natančnosti\ (T) = 1 - stopnja\ napake\ (T). \qquad (13)$$

Glede na definirane mere si želimo model, ki bo imel čim višjo stopnjo natančnosti oz. čim nižjo stopnjo napake.

2.5.3 Metode evalvacije, primerne za neuravnotežene podatke

Klasične mere evalvacije modelov niso primerne za neuravnotežene podatke. Te mere so naravnane tako, da so neodvisne od razreda oz. je zanje vsak razred enako pomemben. To pa v primeru neuravnoteženih podatkov ni dobro, saj nas tu bolj zanima manjšinski razred. Za neuravnotežene podatke Weng in Poon (2006) priporočata naslednje mere: natančnost (angl. *precision*), priklic (angl. *recall*), F-mera ter krivulja ROC, stroškovna krivulja in preciznost-priklic krivulja (angl. *precision-recall curve*). V nadaljevanju nekatere od teh predstavljam bolj podrobno.

Oznake TP, FN, FP in TN so enake kot pri matriki zamenjav (Slika 6). Definicije zgoraj omenjenih mer so (Tan et al, 2014, str. 296, 300):

- **Priklic** (angl. *recall*) meri delež pravilno napovedanih pozitivnih primerov

$$r = \frac{TP}{TP + FN}. \qquad (14)$$

  Klasifikatorji z velikim priklicem imajo zelo malo pozitivnih primerov, ki so klasificirani kot negativni.
- **Natančnost** (angl. *precision*) pa je delež pozitivnih primerov v množici primerov, ki so klasificirani kot pozitivni

$$p = \frac{TP}{TP + FP}. \qquad (15)$$

Večja kot je natančnost, manjše je število nepravilno pozitivno napovedanih primerov.

- Iz zgoraj definiranega je razvidno, da si želimo klasifikator, ki bo maksimiral natančnost in priklic. Zato definiramo še **mero *F***

$$F = \frac{2rp}{r + p} = \frac{2\,TP}{2\,TP + FP + FN} = \frac{2}{\frac{1}{r} + \frac{1}{p}}, \qquad (16)$$

ki je harmonična sredina (angl. harmonic mean) priklica in natančnosti. Harmonična sredina dveh števil stremi k manjšemu od teh dveh števil. Zato velika harmonična sredina pomeni, da sta obe meri, priklic in natančnost, precej veliki.

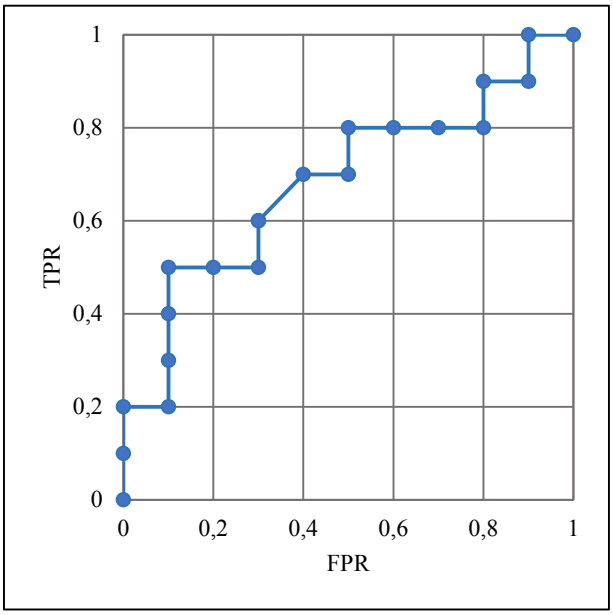
- **ROC krivulja** (angl. *Receiver Operating Characteristic curve*) je standardna evalvacijska metoda za neuravnotežen problem, vendar je z njo težko primerjati različne klasifikatorjev pri različnih stroških napačnega razvrščanja in različnih porazdelitvah razreda. Poleg tega je na istem grafu težko razločiti veliko število krivulj. Zato se velikokrat poleg ROC krivulje uporablja še njen agregat, **ploščina pod krivuljo ROC** ali **AUC** (angl. *area under the ROC curve*), ki meri, kateri model je boljši v povprečju. Če je model popoln, potem je AUC enaka 1, v najslabšem primeru pa model deluje tako, da le naključno razvršča in je vrednost AUC enaka 0,5 – krivulja ROC ima obliko diagonale, ki povezuje točki (0,0) in (1,1). Če primerjamo dva modela, je po tej meri boljši tisti, ki ima večjo AUC.

Pri praktičnem delu je bila krivulja ROC glavna metrika za primerjavo dobljenih modelov. ROC krivuljo rišemo na grafu, kjer so vrednosti točk med 0 in 1. To je grafični prikaz razmerja med TPR (angl. *true positive rate*) in FPR (angl. *false positive rate*). TPR se izračuna kot delež pravilno napovedanih pozitivnih primerov, FPR pa pomeni delež negativnih primerov, ki so napovedani kot pozitivni. TPR je prikazana na *y* osi, FPR pa na *x* osi.

Primer konstrukcije ROC krivulje prikazujem na primeru (Slika 7). Napovedi najprej uredimo padajoče glede na verjetnost, da bo primer klasificiran kot pozitiven (na primeru oznaka p). Na vrhu so primeri, ki imajo največjo verjetnost, da bodo klasificirani kot pozitivni. Če je dejanski razred primera pozitiven (TPR), potem gremo z ROC krivuljo en korak navzgor, sicer gremo en korak desno (FPR). Začnemo v točki (0,0). V tabeli (Slika 7) imamo najprej dva primera s pozitivnim razredom, zato gremo na grafu dva koraka navzgor do točke 0,2 na *y* osi, nato imamo en primer z negativnim razredom in gremo za en korak v desno, tako da pridemo v točko (0,1; 0,2). Nato gremo spet tri korake navzgor, ker imamo tri pozitivne vrednosti … Posebej se obravnava primere, ki imajo enako napovedano verjetnost pozitivnega razreda, npr. primera 10 in 11 iz slike. Tu bi prav tako šli levo v primeru negativnega razreda oz. navzgor v primeru pozitivnega razreda, vendar pri enaki verjetnosti več primerov dejanskih korakov ne narišemo, ampak le povežemo začetno in končno točko teh navideznih korakov (poševna črta na grafu).

Slika 7: Primer ROC krivulje

Primer	Dejanski razred	Napovedana verjetnost p	Smer risanja krivulje	
1	p	0,900	⇧	Vrstni red obravnave primerov ⇩
2	p	0,800	⇧	
3	n	0,700	⇨	
4	p	0,600	⇧	
5	p	0,550	⇧	
6	p	0,540	⇧	
7	n	0,530	⇨	
8	n	0,520	⇨	
9	p	0,510	⇧	
10	n	0,505	⇗	
11	p	0,505	⇗	
12	n	0,390	⇨	
13	p	0,380	⇨	
14	n	0,370	⇨	
15	n	0,360	⇨	
16	n	0,350	⇨	
17	p	0,340	⇨	
18	n	0,330	⇨	
19	p	0,300	⇨	
20	n	0,100	⇨	



V primeru idealnega modela, ki primere razvrsti popolno, so vsi primeri s pozitivnim razredom na začetku, tako da gre krivulja navpično do točke 1 na *y* osi, potem pa gre vodoravno do točke (1,1). ROC krivulja dobrega klasifikacijskega modela je bližja zgornjemu levemu kotu grafa, najslabši model pa ima krivuljo enako diagonali, ki povezuje točki (0,0) in (1,1), kar pomeni, da napoveduje naključno.

V svojem delu sem za evalvacijo uporabljala večinoma AUC mero. Le to sem uporabila povsod, kjer sem potrebovala primerjavo velikega števila rezultatov (npr. sočasna primerjava velikega števila modelov). AUC je za take primere priročna, ker je ocena strjena v eno samo številko. Primerjava več modelov je tako lahko narejena z enostavnim urejanjem, ali pa tudi z izrisom seznama AUC števil na graf (kjer je ena AUC meritev predstavljena z eno točko). Povsod, kjer sem potrebovala bolj natančno primerjavo manjšega števila modelov, pa sem uporabila ROC krivuljo. Tu je edina možna primerjave izris na grafu (kjer je ena meritev uspešnosti predstavljena z eno krivuljo). Zaradi tega lahko hkrati primerjamo zgolj manjše število modelov (npr. do 10), vendar pa je prednost to, da imamo možnost natančnega vpogleda, kako dobro vsak model razločuje v različnih delih prostora primerov (npr. kako se odreže med bolj tveganimi zavarovanji in kako med bolj varnimi).

2.6 Orodja podatkovnega rudarjenja

V magistrskem delu sem za reševanje opisane problematike uporabila dve orodji, primerni za podatkovno rudarjenje: orodje za informacijsko rudarjenje iz Univerze v Konstanzu (angl. *the*

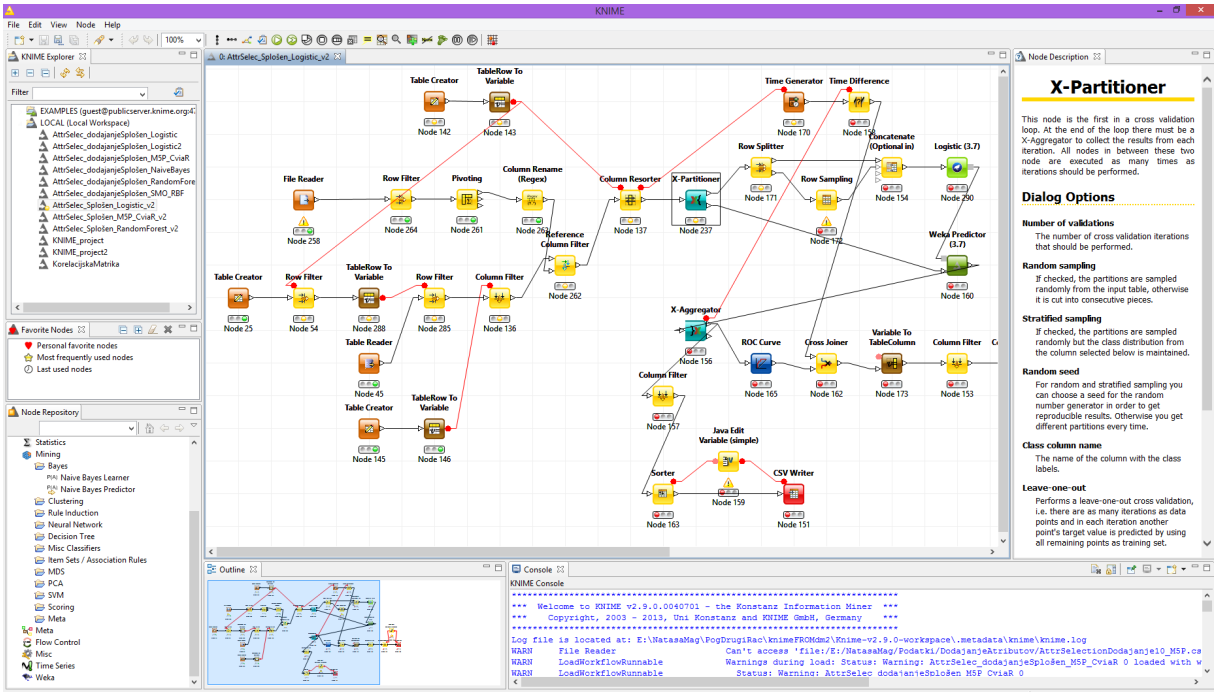
Konstanz Information Miner, v nadaljevanju KNIME) in Waikato okolje za analizo znanja (angl. *Waikato Environment for Knowledge Analysis*, v nadaljevanju Weka). Obe orodji sta odprtokodni in tudi precej razširjeni tako v akademskih krogih kot tudi v poslovni praksi. V tem poglavju bom obe orodji na kratko predstavila ter podala njune prednosti in slabosti pri praktični uporabi.

2.6.1 KNIME

KNIME je odprtokodno orodje, ki se ga uporablja za podatkovno rudarjenje in izvajanje zapletenih statističnih analiz pri analizi trendov in napovedovanju. Razvijajo ga na Univerzi Konstanz v Nemčiji, kjer ga uporabljajo tudi za poučevanje in raziskovanje. Že osnovna verzija vsebuje več sto vozlišč za dostop do podatkov, pred-obdelavo in čiščenje podatkov, modeliranje, analizo in podatkovno rudarjenje ter različne interaktivne poglede (angl. *views*), kot je npr. *scatter plot*. Dodatno se lahko vključi tudi več dodatnih knjižnic, med katerimi je bila zame zanimiva predvsem knjižnica Weka, ki vsebuje okrog 100 algoritmov iz znanega istoimenskega orodja za podatkovno rudarjenje. Iz zavarovalniškega vidika je zanimiva tudi knjižnica R-script, ki ponuja dostop do velike knjižnice statističnih metod, med katerimi je tudi veliko uporabnih v aktuarstvu oz. zavarovalništvu.

V nadaljevanju (Slika 8) prikazujem delovno okolje KNIME, kjer osrednji del zavzema urejevalnik delotokov, v katerem sestavljamo delotoke, jih urejamo … Na levi strani se nahajajo raziskovalec, kjer so prikazani vsi projekti, razdelek najljubših vozlišč ter shramba vseh vozlišč, na desni strani pa se nam ob izbiri določenega vozlišča odpre opis vozlišča. To je le osnovni prikaz, več informacij je dostopnih na internetni strani (KNIME, 2013). Za potrebe tega dela sem uporabljala več verzij orodja KNIME, ki so izšle med leti 2012 in 2014. Končni rezultati

Slika 8: Delovno okolje KNIME



predstavljeni spodaj so bili ponovno preračunani z KNIME verzijo 2.9.0, ki je izšla 6.12.2013 in je tako iz stališča tega dela referenčna verzija.

2.6.2 Weka

Weka je danes ena najbolj standardnih in široko uporabljenih odprtokodnih programskih oprem za strojno učenje. Nastala je na univerzi Waikato na Novi Zelandiji. Weka vsebuje zbirko orodij za vizualizacijo in algoritme za analizo podatkov in modeliranje skupaj z grafičnimi vmesniki. Zelo je priljubljena tako v akademskih krogih kot pri raziskovanju v industriji, poleg tega pa se uporablja tudi za učne namene.

Weka je brezplačno dostopna pod zelo dostopno licenco GNU General Public License. V celoti se izvaja v programskem jeziku Java in tako deluje na skoraj vsaki sodobni računalniški platformi. Tudi uporaba je precej enostavna, predvsem zaradi grafičnih uporabniških vmesnikov (Witten & Frank, 2005, str. v; Weka, 2013).

Weka podpira več standardnih nalog podatkovnega rudarjenja, predvsem za pred-obdelavo podatkov, razvrščanje v skupine, klasifikacijo, regresijo, vizualizacijo in izbor atributov. Vse tehnike temeljijo na predpostavki, da imajo na voljo nabor podatkov, kjer je vsaka točka podatkov opisana z določenim številom atributov.

Uporabniškega vmesnika Weka ne bom predstavljala, ker ga nisem uporabljala. Pri praktičnem delu sem uporabila uporabniški vmesnik iz orodja KNIME, v njem pa sem vključila knjižnico Weka, od koder sem črpala standardne algoritme. Referenčna verzija knjižnice Weka uporabljena v mojem delu je bila 2.9.0.0040628 in je vezana na verzijo 3.7 orodja Weka.

2.6.3 Kombiniran pristop

Tako orodje KNIME kot orodje Weka imata določene prednosti, ki mi omogočajo, da s kombinacijo obeh dobim močnejše in bolj prilagodljivo orodje, kot če bi uporabljala zgolj enega izmed njih. Kot sem omenila že v uvodu, sta obe orodji odprto-kodni in precej razširjeni. Pri svojem procesu rudarjenja podatkov sem uporabila delovno okolje v obliki delotokov oz. vizualnega programiranja, saj mi poleg preglednosti nudi veliko fleksibilnosti pri uporabi in kombiniranju algoritmov za pred-obdelavo in rudarjenje podatkov. Za kombinacijo obeh orodij sem se odločila iz naslednjih razlogov:

- KNIME ima uporabniku zelo prijazen uporabniški vmesnik. Poleg tega je tudi močno orodje za pred-obdelavo podatkov, kar vključuje spremembo tipa določenih podatkov, ima tudi široko bazo uporabnikov in posledično veliko dokumentacije na spletu. Prednost orodja KNIME je tudi njegova stabilnost in zmožnost obdelave zelo velikih količin podatkov, zato ga pogosto uporabljajo v poslovnih okoljih za reševanje dejanskih problemov.
- Weka je trenutno najbolj standardno in splošno sprejeto orodje v akademskih krogih. V tem okviru ima orodje Weka (podobno kot KNIME v industriji) najštevilčnejšo bazo

uporabnikov, kar posledično zopet pomeni dobro podporo in dokumentacijo na spletu. Nadalje ima Weka daleč najširši nabor različnih algoritmov podatkovnega rudarjenja, ki so v primerjavi z orodjem KNIME tudi bolj standardizirani oz. implementirani po originalnih delih iz svojih področji (referenčne implementacije).

Za glavni del rudarjenja sem uporabila hibrid med orodji KNIME in Weka. Za delotočno delovno okolje sem uporabila KNIME, v njem pa sem vključila knjižnico Weka, od koder sem črpala standardne algoritme podatkovnega rudarjenja. Tu se pokaže še ena prednost orodja Weka: vsi učni algoritmi imajo enako vozlišče za napovedni del (angl. *predictor*), medtem ko potrebujejo učni algoritmi KNIME za vsako metodo drugačen napovedni del. Orodje Weka ima tudi svoje delotočno delovno okolje, vendar v primerjavi z orodjem KNIME ni tako dodelano in prilagodljivo in ne vsebuje tako močnih postopkov za pripravo oz. pred-obdelavo podatkov.

3 RAZUMEVANJE IN PRIPRAVA PODATKOV

V tem poglavju bom predstavila dve stopnji CRISP-DM procesnega modela, razumevanje podatkov in priprava podatkov. Obe stopnji sta precej povezani, zato ne bom strogo ločevala, kateri del pripada eni in kateri drugi. Vsekakor pa sta to stopnji, ki sta časovno najbolj zahtevni. Po navedbah strokovnjakov naj bi stopnji raziskovanja in razumevanja podatkov ter čiščenja podatkov predstavljali tudi do 80 % napora in s tem določali 80 % vrednosti končnih rezultatov podatkovnega rudarjenja (Dasu & Johnson, 2003).

Kudyba in Hoptroff (2001, str. 44) poudarjata, da nobena metoda podatkovnega rudarjenja ne bo dala dobrih rezultatov, če so podatki zbrani slabo. Zato morajo zbrani podatki vsebovati neko informacijo, morajo biti relevantni in morajo biti v takem formatu, da se jih lahko učinkovito uporabi v procesu podatkovnega rudarjenja. Vse tri zahteve sem poskušala doseči tudi pri svojem naboru podatkov in jih predstavljam v sledečih podpoglavjih.

3.1 Opis podatkov

Podatke sem pridobila iz baze podatkov kompozitne zavarovalnice, ki se ukvarja tudi s trženjem zavarovanj vozil. Tu navajam osnovne lastnosti izbora podatkov (pogodb) iz baze:

- Iz baze sem izbrala samo pogodbe zavarovanj vozil, kjer je sklenjeno zavarovanje AO.
- Od teh sem izbrala pogodbe, ki so bile zaključene normalno, ali pa so bile prekinjene z razlogom škode ali prodaje vozila, predno se je izteklo zavarovanje. Ostale prekinjene pogodbe sem ignorirala zaradi majhnega deleža in dodatnega šuma v podatkih.
- Izvzela sem generalne pogodbe, to so pogodbe s pravnimi osebami, na katerih je zavarovanih več motornih vozil in so praviloma tudi večletne. Take pogodbe bi lahko pokvarile rezultat, saj je pri teh težko oz. nemogoče oceniti, koliko je določen voznik tvegan oz. ali bo določen voznik imel nesrečo, ker praviloma vozila na teh pogodbah niso vezana na enega ali dva voznika.
- Izključila sem vse anekse, to je dodatke k pogodbam.

- Časovno sem se omejila na pogodbe, ki so bile sklenjene med 1.6.2004 in 31.12.2012. Prvi datum sem izbrala zato, ker je delež pogodb pred tem datumom majhen, poleg tega pa so bile starejše pogodbe zelo skope z številom vpisanih lastnosti. Drugi datum pa sem izbrala zato, ker je v analizo smiselno zajeti le podatke zaključenih mesecev, saj se ti podatki ne spreminjajo. Zadnjega tekočega leta vključno z dodatnim mesecem nisem zajela, ker se škode praviloma prijavijo enakomerno med letom oz. velikokrat tudi po preteku pogodbe. Za pogodbe zadnjega leta tako trdim, da nimam podatka o vseh škodah in jih je zato nesmiselno obravnavati.
- Podatke sem pridobila iz baze na dan 30.1.2014 in zajela 838.073 pogodb.

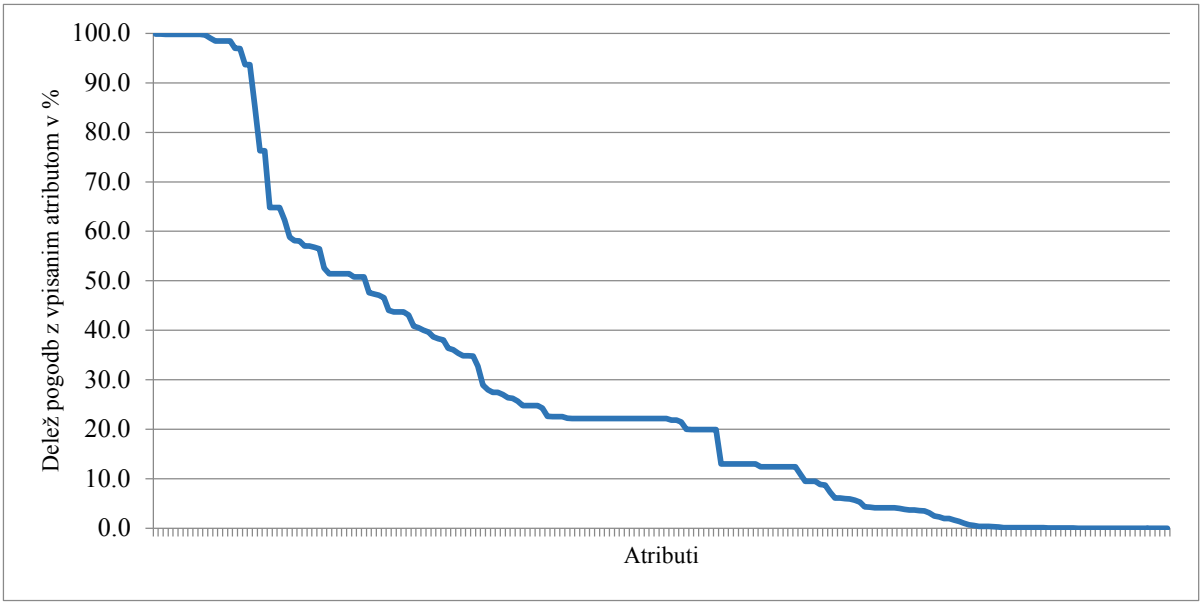
V bazi je več kot 200 atributov, vendar bi bilo nesmiselno vključiti vse atribute v rudarjenje podatkov. Zato sem naredila analizo, kjer sem določila število oz. delež pojavitev določenega atributa na izbranih pogodbah. Rezultat te analize je tabela (Slika 9), ki vsebuje ime atributa, ki sem ga generirala iz id-ja in imena atributa, zalogo vrednosti, ki pove, koliko je različnih vrednosti atributa na pogodbah ter tip atributa, ki opisuje tip vrednosti atributa in je lahko intervalni (zaloga vrednosti so cela ali realna števila), binarni (zaloga vrednosti sta dve vrednosti, po navadi da in ne) ali pa nominalni (zaloga vrednosti je šifrant ali besedilo nekega opisa). Tabela je zelo velika, zato na sliki prikazujem le del tabele.

Slika 9: Primer izpisa tabele atributov

Ime	Opis	Delež pogodb z vpisanim atributom v %	Zaloga vrednosti	Tip
A8_LETNIK	Letnik	99,87	270	intervalni
A23_STAROST	Starost	99,87	290	intervalni
A3_ZNAMKA	Znamka vozila	99,81	6.261	nominalni
A4_TIP_VOZILA	Tip vozila	99,81	73.639	nominalni
A6_MOC_MOTORJA_KW	Moč motorja (kW)	99,81	1.208	intervalni
A7_PROSTORNINA_MOTORJA	Prostornina motorja	99,81	4.026	intervalni
A5_ST_SASIJE	Številka šasije	99,81	313.509	nominalni
A15_EUROTAX_KODA	E-Koda	99,81	41.106	intervalni
A63_PRAVNA	Vozilo v lasti pravne osebe	99,81	2	binarni
A2_REG_ST	Registrska številka	99,81	286.630	nominalni
A12_ST_POTN_MEST	Št. potn. mest	99,68	107	intervalni
A193_REG_ST_INT	Registriska številka - interno	99,06	259.376	nominalni
A9_MASA_PRAZNO	Masa praznega vozila (kg)	98,47	3.922	intervalni
A10_MASA_DOVOLJENA	Dovoljena skupna masa	98,47	2.763	intervalni
A30_ALARM	Vozilo ima alarm	98,47	2	binarni
A11_NOSILNOST	Nosilnost (kg)	98,47	4.700	intervalni
A37_NABAV_VRED	Nabavna vrednost	96,98	63.354	intervalni
A16_NABAV_VRED_EUROTAX	Nabavna vred. (EUROTAX)	96,92	52.344	intervalni
A38_POVEC_OSEBE	Poveč. ZV - osebe	93,67	4	binarni
A39_POVEC_STVARI	Poveč. ZV - stvari	93,67	4	binarni
A13_IDD_SIFRA_VRSTE_VOZILA	IDD Šifra vrste vozila	85,21	11	intervalni
A32_RAZRED_MOCI_AO_AOPLUS	Razred moči AO, AO+	76,27	8	nominalni
A33_RAZRED_ST_SEDEZEV	Razred št. sedežev	76,27	2	nominalni
A93_NADSTD_XENON	Ksenonski žarometi, žarometi, ki svetijo v ovinek	64,82	2	binarni
A94_NADSTD_TIPALO	Tipalo za dež na vetrobranskem steklu	64,82	2	binarni

Na podlagi te analize sem ugotovila, da je v nadaljnjo analizo smiselno zajeti 114 atributov (meja je določena empirično, pri 114. atributu je opazen izrazit skok iz približno 20 % na 13 % pogodb), ostali atributi so izpuščeni (Slika 10). Na *x* osi so predstavljeni posamezni atributi (zaradi boljše predstavitve na *x* os nisem pisala imena atributov), na *y* osi pa delež pogodb, na katerih je vpisan določen atribut. Pri tem je potrebno poudariti, da vrednosti v zalogi vrednosti niso nujno smiselne, podan delež zgolj pove, koliko jih je dejansko vpisanih na pogodbe (v bazi različnih od NULL). Šele v podrobnejši analizi sem zalogo vrednosti preverila in popravila na smiselne vrednosti (npr. letnik avtomobila ne more biti več kot 2012), tako da so dejanski deleži smiselnih vrednosti manjši ali enaki, kot sem jih dobila v tej prvotni analizi. Pri tej analizi pa sem tudi ugotovila, kateri atributi so bodisi obvezni oz. se sami izpolnijo iz šifranta bodisi je nanje vezan določen popust in so zato tudi stranke in zastopniki bolj motivirani za vnos atributa.

Slika 10: Delež pogodb z vpisanim atributom



3.2 Zajem in čiščenje podatkov ter izpeljava novih atributov

Zajem podatkov iz baze je bil časovno zelo dolgotrajen in zahteven proces. Potekal je v več delih, ki pa so poleg samega izpisa primerov vključevali tudi čiščenje podatkov in izpeljavo novih atributov. Kot sem omenila že v poglavju 3.1, je bilo pri analizi atributov potrebno izvesti dodatno, podrobnejšo analizo. Za vsak atribut posebej je bilo potrebno določiti tip in zalogo vrednosti ter izvesti določene popravke te zaloge vrednosti. Postopek podrobnejše analize prikazujem v nadaljevanju.

Posamezni del analize sem izvedla s pomočjo poizvedb (angl. *query*) direktno v bazi podatkov (baza podatkov je na strežniku Microsoft SQL Server 2008). Kjerkoli sem potrebovala vizualizacijo, sem rezultate poizvedb prikazala v programu Microsoft Excel 2010.

Poizvedbe so vsebinsko precej raznolike, saj v določenih izpeljem nove atribute, nekatere so zgolj pomožne in služijo samo boljšemu pregledu, tretje pa vsebujejo čiščenje podatkov in

spremembe tipov atributov. Sledijo si v logičnem zaporedju: najprej sem izbrala pogodbe, nato sem izbrala osnovne atribute, izpeljala nove atribute, jih združila v eno tabelo, potem pa izvedla še precej čiščenja zbranih podatkov. Namen je bil zbrati podatke v čimbolj primerno obliko za proces podatkovnega rudarjenja. Postopek je bilo potrebno nekajkrat ponoviti, saj je končna tabela, ki sem jo dobila iz omenjenih poizvedb, v začetnih fazah zelo slabo ustrezala zahtevam, ki jih predpisuje orodje KNIME, v katerem sem izvajala proces podatkovnega rudarjenja. V končni fazi sem dobila naslednje poizvedbe (vsaka točka predstavlja eno poizvedbo), vse skupaj pa znese preko tri tisoč vrstic SQL kode:

1. Izbrala sem vse relevantne pogodbe (le-te sem opisala v prejšnjem podpoglavju) ter generirala id-je.
2. Za dobljene pogodbe sem iz različnih povezanih tabel pridobila vse željene osnovne atribute. Izbrala sem 114 atributov, ki so bili pogosto izpolnjeni na pogodbah. Izbrani atributi se nanašajo predvsem na zavarovanje samo in opisujejo lastnosti vozila (npr. letnik vozila, znamka vozila, prostornina motorja …), vozniške izkušnje (npr. leto izdelave izpita B kategorije), višino zavarovalnih vsot ipd.
3. Izpeljala sem nove atribute, ki predstavljajo višino premij, škod, škodnih rezervacij ter število škodnih spisov na različnih zavarovanjih vozil (npr. premija AO, škoda AK …). Poleg teh pa sem dobila še atribute, ki se nanašajo na osebe na pogodbi ter njihove določene lastnosti (npr. spol, datum rojstva …).
4. Pridobila sem nove atribute, ki se nanašajo na trajanje pogodbe in na lastnosti, ki opisujejo, na kakšen način je bila določena pogodba prekinjena.
5. Do sedaj izpeljane osnovne atribute sem prestavila v stolpce zaradi kasnejše lažje obravnave, saj so vsi izpeljani atributi podani v stolpcih, pa tudi proces podatkovnega rudarjenja zahteva tako organizacijo.
6. Izdelala sem boljši pregled nad podatki, saj sem izpeljanim atributom dala bolj smiselna imena oz. takšna imena, da ustrezajo enakemu načinu poimenovanja, kot jih imajo osnovni atributi. Izpeljane atribute sem preimenovala v naslednjo obliko, npr. B27_premija_AO, kjer oznaka B pomeni, da gre za del atributov, izpeljanih v poizvedbi pod 3. točko (če ima atribut oznako C, pomeni, da gre za del atributov, izpeljanih v poizvedbi pod 4. točko, medtem ko so osnovni atributi iz tabele atributov iz 2. točke označeni s črko A, npr. A8_letnik) številka predstavlja zaporedno številko atributa, zadnji del pa smiselno poimenovanje atributa.
7. Atribute sem nato združila v eno tabelo.
8. Sledili so razni popravki. Za vsak atribut posebej sem določila tip in smiselno zalogo vrednosti. V bazi so vse vrednosti atributov podane v obliki besedila (tipa varchar) in zato je bila potrebna konverzija v številske tipe (int, numeric, bit). Najprej sem popravila numerične atribute in določene spremenila v realna oz. cela števila (odvisno od posameznega atributa). Boolove atribute (npr. spol zavarovalca, ki je imel oznako M ali Ž) sem spremenila v tip bit, kar pomeni, da sem izvedla preslikavo v vrednosti 0 in 1. Pri kategoričnih atributih sem določila dovoljene kategorije ter izvedla preslikavo v te dovoljene kategorije, manjkajoče vrednosti pa spremenila v NULL. V tej poizvedbi sem

izvedla tudi nekaj čiščenja podatkov, da sem v končni fazi dobila smiselne vrednosti. Primere čiščenja navajam v nadaljevanju.

9. Na intervalnih atributih sem popravila intervale. Določila sem spodnjo in zgornjo mejo intervala in pri nekaterih atributih izvedla tudi preslikavo intervalov v dovoljene intervale. Razne osamelce sem nadomestila z vrednosti NULL, saj le-ti lahko zelo pokvarijo rezultat podatkovnega rudarjenja. Pri letih sem različne predstavitve let poenotila (npr. leto 99 sem nadomestila z zapisom 1999).
10. Pri atributih z oznako B sem izpeljala nove atribute tipa bit, kar pomeni, da sem jim določila vrednosti 0 ali 1. Primer: iz atributa B69_skoda_AK sem izpeljala atribut B69_ali_skoda_AK, ki je dobil vrednost 0, če je bila vrednost atributa B69_skoda_AK enaka 0 ali NULL (kar pomeni, da ni bilo škode na zavarovanju AK po tej pogodbi), oz. je dobil vrednost 1, če je bila izplačana škoda. Tako sem izdelala še nekaj dodatnih atributov, za katere sem kasneje ugotovila, da določeni od njih bistveno prispevajo h končnemu rezultatu moje analize.
11. Kasneje sem ugotovila, da določeni algoritmi v orodju KNIME zahtevajo, da so Boolovi atributi kategoričnega tipa (in ne tipa bit), zato sem tem atributom spremenila tip.
12. Naslednja ugotovitev je bila, da tudi zgodovina pogodb vpliva na rezultat. Zato sem dodala še nekaj izpeljanih atributov, ki so povezani z zgodovino pogodbe (število pogodb pred trenutno aktivno pogodbo, število škod na prejšnjih pogodbah, povprečno trajanje pogodb …). Tudi tem atributom sem dala oznako C.
13. V tej poizvedbi sem izvedla dodatna čiščenja in sicer je bilo potrebno natančno definirati registrsko območje in znamko vozil. Te poizvedbe nisem izpisala že v prvi fazi, ampak šele kasneje, saj je bil rezultat podatkovnega rudarjenja zelo slab, nanašal pa se je ravno na omenjena atributa.
14. Sledil je popravek vseh atributov, ki so označevali leta (npr. leto rojstva zavarovalca, leto začetka izdelave modela avtomobila …). Te sem spremenila v starosti (npr. starost zavarovalca, starost modela avtomobila …), kar se je izkazalo za dober popravek v smislu kakovosti rezultatov rudarjenja podatkov.
15. Tu sem dodala še časovno omejitev glede izbire pogodb, saj sem najprej v proces podatkovnega rudarjenja zajela vse pogodbe, v kasnejših fazah rudarjenja pa se je izkazalo, da časovna omejitev poda boljše rezultate, saj so zelo stare pogodbe skope z atributi, nove pa še niso zaključene in zato ne vemo, ali se bo škoda zgodila. Za najbolj optimalno se je izkazalo obdobje od 1.6.2004 do 31.12.2012. Podrobnejše razloge sem navedla že v razdelku 3.1.
16. Zadnja poizvedba je bila pomožna, kjer sem izpisala končno verzijo nabora podatkov.

Spodaj predstavljam še nekaj najbolj tipičnih sistemskih napak zalog vrednosti, ki se verjetno pojavljajo tudi v drugih podobnih nalogah rudarjenja podatkov iz podatkovnih baz. Podobnih napak je še veliko:

- leta spremenimo, da so zapisana na 4 mesta; glede na ostale atribute pogodb se sklepa, da npr. zapisi 1, 2 … pomenijo leta 2001, 2002 … zapisi 98, 99 pa leta 1998, 1999 ...,

- prostornina motorja: pri precejšnjem številu pogodb (okrog 10 % pogodb) je bil vnesen podatek v napačnih enotah in sicer v litrih, medtem ko bi moral biti v ccm, zato se ta del pomnoži s 1000, da dobimo prave enote,
- 0 je zamenjana z O ali obratno pri kategoričnih atributih (npr. Opel je zapisan kot 0pel),
- pri letih je velikokrat vnesen še kakšen dodaten znak, npr. „, e, n. …

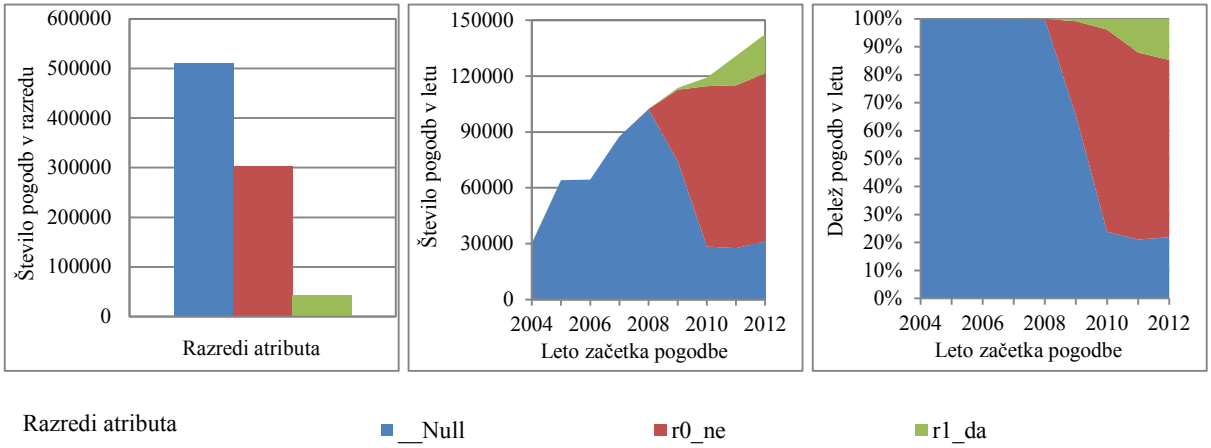
Posebej sem pripravila še statistike za podroben prikaz zaloge vrednosti atributa ter njene dinamike po letih začetka pogodbe. Še posebej problematični so bili numerični atributi, saj je bilo pri teh potrebno za prikaz smiselno določiti tudi razrede. Prikaz statistik se je izkazal za nujen pristop pri čiščenju podatkov. Na grafičnih predstavitvah se je zelo lepo videlo, kateri podatki vsebujejo napake, kje so odstopanja, osamelci, od kdaj se uporablja določen atribut … Statistike sem izvedla v več poskusih, dokler nisem prišla do končnega prikaza, kot ga prikazujem v nadaljevanju. Za vsak posamezen atribut je bilo potrebno narediti podrobno analizo zaloge vrednosti. Rezultat je prečiščena baza podatkov oz. nabor atributov s smiselno zalogo vrednosti. Ta rezultat je zelo pomemben in uporaben, saj smiselni in prečiščeni podatki niso pomembni samo pri podatkovnem rudarjenju, ampak pri vseh ostalih analizah podatkov iz teh baz (npr. klasične aktuarske analize, analize oddelka trženja …).

V nadaljevanju prikazujem na nekaj primerih še zgled razumevanja in čiščenja podatkov na podlagi vizualizacije statistik atributov. Tako analizo sem naredila za vse atribute, ki sem jih uporabila v rudarjenju. Za vsak atribut prikazujem tri grafe. Prvi graf predstavlja kumulativno porazdelitev vrednosti atributa za vsa leta skupaj. Na drugem grafu je prikazano absolutno število vrednosti atributa po letih; tu je tudi viden trend rasti pogodb, širina pasu pa pomeni absolutno število pogodb z določeno vrednostjo atributa. Tretji graf predstavlja relativno število pogodb z danim atributom za dano leto glede na vse pogodbe tistega leta.

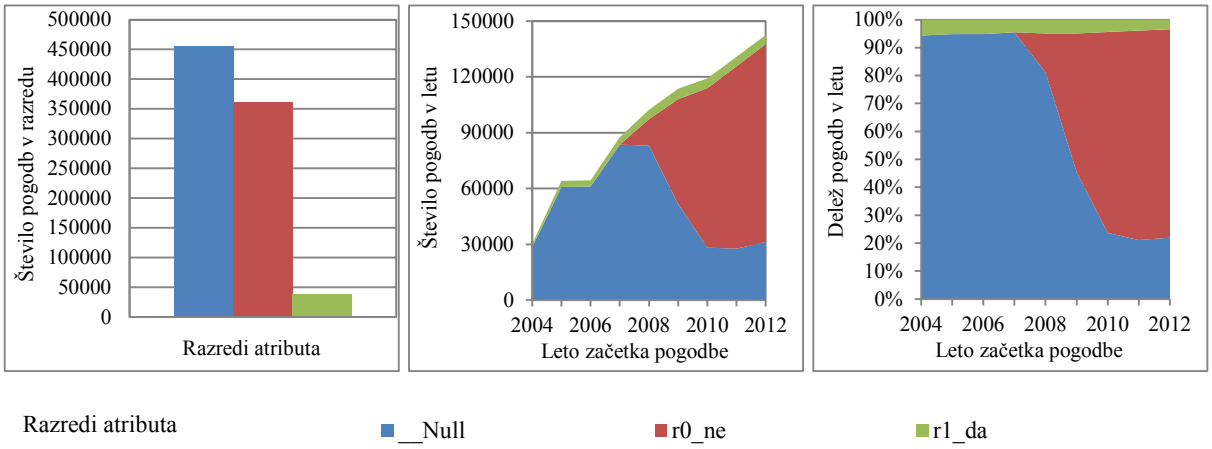
Atribut A125_POP_MLADA_DRUŽINA (Slika 11) prikazuje popust, ki ga prejmejo zavarovalci, ki imajo mlade družine (to so družine z otrokom mlajšim od določene starosti). Atribut je binarni, kar pomeni, da zaloga vrednosti vsebuje samo vrednosti da in ne (in seveda NULL). Iz drugega in tretjega grafa je razvidno, da se ta popust daje od leta 2008 dalje. Od tega leta dalje se tudi število pogodb z vrednostjo NULL zmanjšuje.

Atribut A98_JE_MLADI (Slika 12) pomeni doplačilo za voznika z manj kot tremi leti vozniških izkušenj na zavarovanju AO. Tudi ta atribut je binaren. Vidimo, da je delež pogodb z mladim voznikom v upadanju (čeprav je padec precej majhen). Od leta 2007 dalje se na vsako pogodbo zabeleži, ali gre za mladega voznika ali ne, saj od takrat dalje opazimo, da se delež pogodb z vrednostjo ne povečuje (pred letom 2007 se je očitno na pogodbo vpisala le vrednost da, ali pa se ni zapisalo nič, kar označuje vrednost NULL).

Slika 11: Statistike za atribut A125_POP_MLADA_DRUŽINA



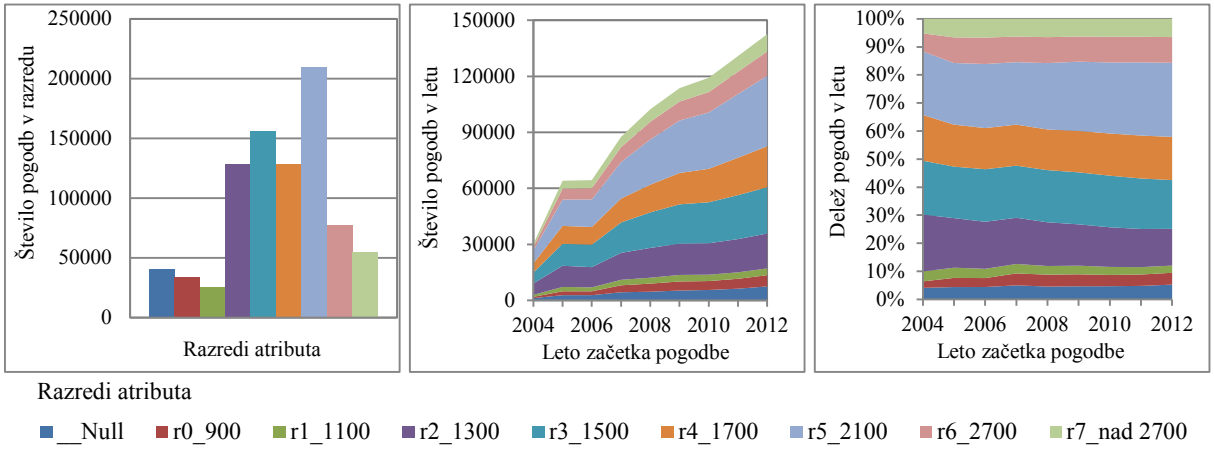
Slika 12: Statistike za atribut A98_JE_MLADI



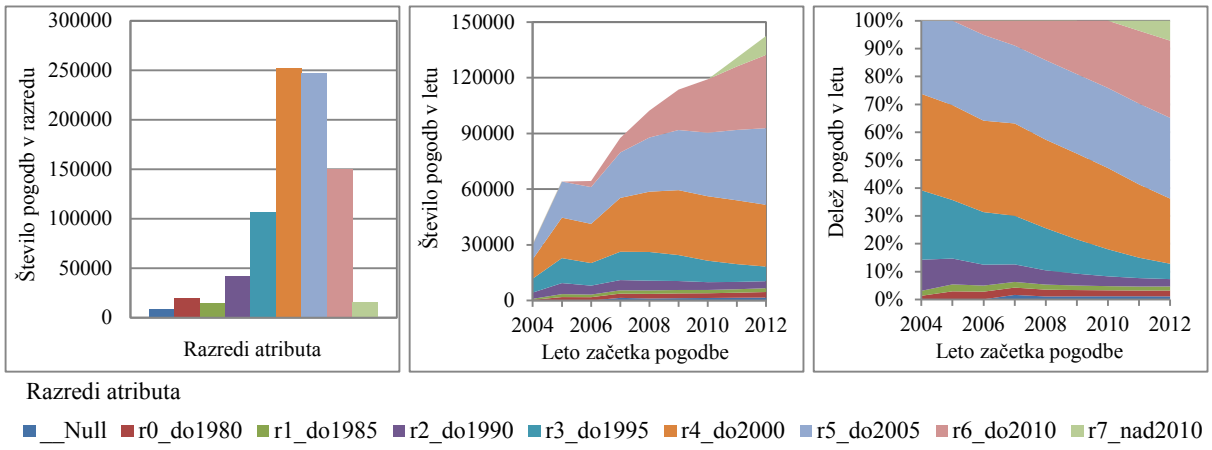
Atribut A7_PROSTORNINA_MOTORJA (Slika 13) je numeričen in zato je bilo tu potrebno narediti razrede zaloge vrednosti (glej legendo pod grafi): npr. r5_2100 pomeni razred vseh tistih vozil, ki imajo prostornino motorja med 1700 ccm in 2100 ccm – ta razred je tudi največji. Iz tretjega grafa je razvidno, da je moč motorja približno enako razporejena po letih začetka pogodb, majhno naraščanje je opazno v razredu r5_2100, pa tudi zgornja dva razreda, r6_2700 in r7_nad2700, sta se povečala glede na leta 2004 in 2005; r2_1300 pa je v upadanju.

Atribut A8_LETNIK (Slika 14) prikazuje letnik vozila. Tudi ta atribut je numeričen (v zalogi vrednosti ima preko 300 različnih zapisov). Za preglednejšo sliko sem tudi tu zalogo vrednosti razdelila na razrede (glej legendo pod grafi; npr. razred z oznako r1_do1985 pomeni razred tistih vozil, ki imajo letnik večji od 1980 in manjši ali enak 1985). Iz prvega grafa vidimo, da je zavarovanih največ vozil, ki imajo letnik večji med leti 1995 in 2000, najmanj pa vozil, ki imajo letnik med 1980 in 1985. Na tretjem grafu je viden padec pogodb po letnikih avtomobilov, kar je tudi smiselno, saj se vsako leto zavarujejo tudi novi avtomobili (npr. razred r7_nad2010 se pojavi šele od leta 2010 dalje in raste, saj vsi novi avtomobili padejo v ta razred).

Slika 13: Statistike za atribut A7_PROSTORNINA_MOTORJA



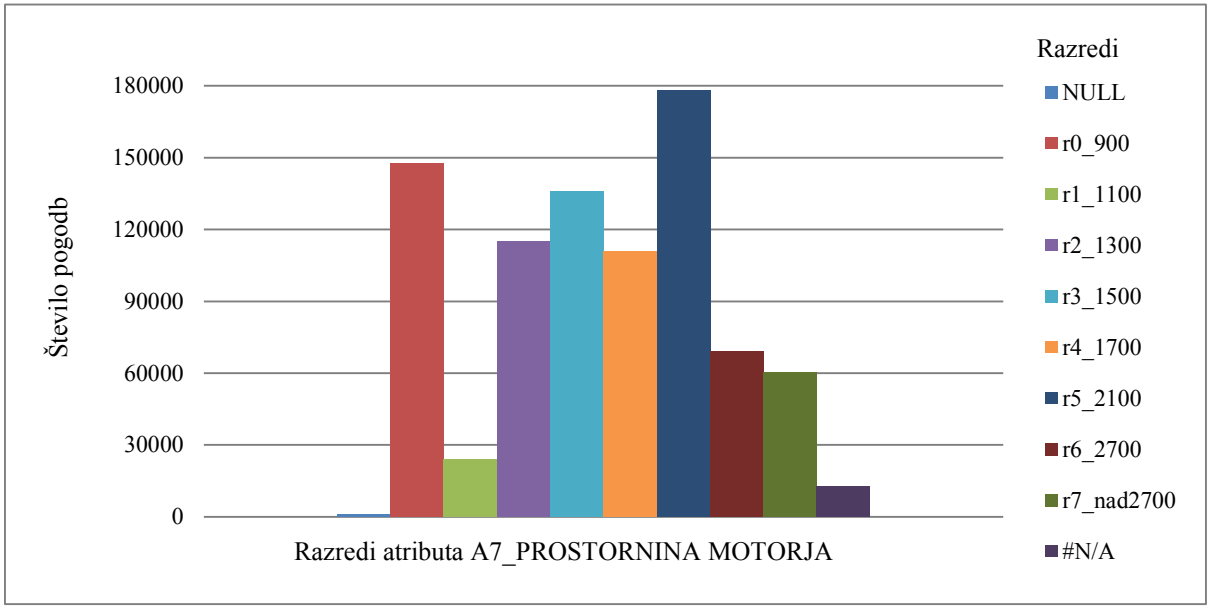
Slika 14: Statistike za atribut A8_LETNIK



Te primere zaključujem z razmislekom o pomembnosti pogleda v podatke (npr. na način kot je predstavljen zgoraj pa tudi z ročnim pregledom naključno izbranih primerov pogodb). Taka pred-analiza in čiščenje sta časovno zahtevna, sta pa izjemnega pomena, s čimer se strinjajo praktično vsi avtorji na področju rudarjenja podatkov. Na zgornjih primerih sem sicer pokazala že prečiščene podatke, zato posebnih anomalij ni opaziti (razen morda velikih deležev nevpisanih vrednosti NULL). V nadaljevanju pa prikazujem dva primera atributov na originalnih podatkih, A7_PROSTORNINA_MOTORJA in A8_LETNIK (za oba atributa sem že zgoraj prikazala prečiščene porazdelitve vrednosti Slika 13 in Slika 14).

Za atribut A7_PROSTORNINA_MOTORJA sem tudi na originalnih podatkih naredila razrede zaloge vrednosti. Iz grafa je razvidno (Slika 15), da je razred r0_900 izredno velik (v tem razredu so vsa vozila, ki imajo prostornino motorja velikosti do 900 ccm). Podrobnejši pregled podatkov je razkril, da je v tem razredu zelo veliko vozil, ki imajo prostornino motorja celo manjšo od 10 ccm, kar pa je očitno napaka, saj avtomobili ne morejo imeti tako majhne prostornine motorja. Izkazalo se je, da je bil na določenem deležu pogodb podatek o prostornini motorja vnesen v napačnih merskih enotah (v litrih namesto v ccm), zato sem te vrednosti

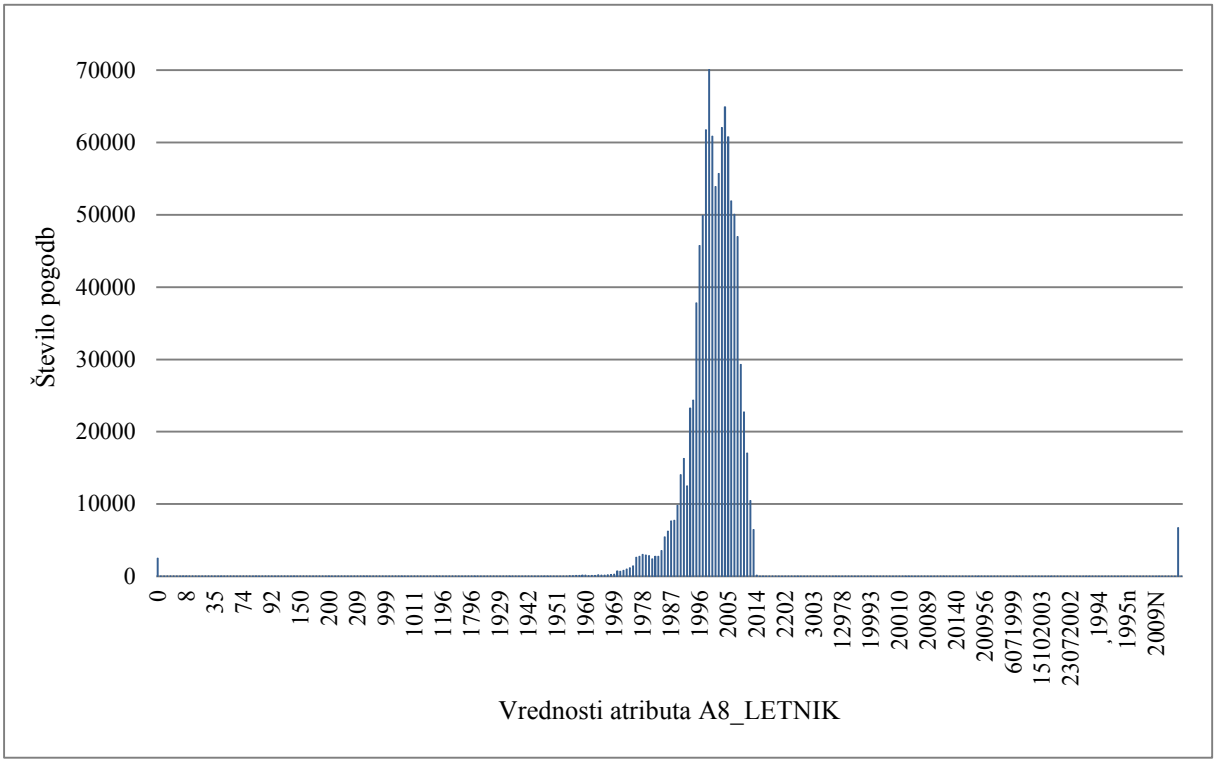
Slika 15: Razredi atributa A7_PROSTORNINA_MOTORJA na originalnih podatkih



spremenila v prave merske enote in dobila bolj smiselne podatke (Slika 13, prvi graf). V originalnih podatkih je precej velik tudi razred #N/A, ki vsebuje vse pogodbe, na katerih podatek o prostornini motorja ni bilo neko število. Tudi te podatke sem popravila s smiselnimi vrednosti, kjer je bilo to mogoče, sicer pa sem tako pogodbo uvrstila v razred NULL (Slika 13).

Za atribut A8_LETNIK pa originalnih podatkov nisem združila v razrede, ker sem hotela pokazati, kakšne vrednosti so se pojavile v zalogi vrednosti (Slika 16). Na *x* osi so prikazane

Slika 16: Porazdelitev vrednosti za atribut A8_LETNIK na originalnih podatkih



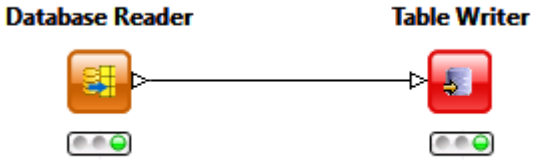
vrednosti atributa. Napake so očitne, saj npr. letnik vozila ne more biti 199 ali pa 20009. Na večini pogodb je sicer vnesen smiseln podatek, na nezanemarljivo majhnem deležu pogodb pa je bilo potrebno vrednost smiselno popraviti. Prečiščene podatke sem uredila v razrede in so razvidni iz zgoraj predstavljenega grafa (Slika 14).

Opisana analiza in razumevanje sta bila izhodišče za čiščenje in izboljšanje kakovosti podatkov.

3.3 Priprava podatkov za izgradnjo modelov

Uvoz podatkov sem izvedla s pomožnim delotokom v orodju KNIME, kjer sem podatke uvozila v KNIME direktno iz baze, torej iz tabele, ki sem jo dobila s poizvedbo pod zaporedno številko 16 (glej podpoglavje 3.2). Delotok je na prvi pogled precej enostaven (Slika 17), vendar pa vozlišče **Database Reader** zahteva precej različnih nastavitev: z vozliščem se vzpostavi in odpre povezavo z bazo podatkov (v mojem primeru z SQL bazo). Tu uporabnik vpiše SQL stavke, ki se tu tudi izvedejo (npr. *SELECT * FROM table*). Z vozliščem **Table Writer** zapišem podatkovno tabelo, ki jo pridobim iz SQL baze, v KNIME-ov matični podatkovni dokument s končnico *.table*. Ta dokument se kasneje prebere z vozliščem **Table Read**, ki sem ga uporabila v stopnji modeliranja (glej podpoglavje 4.1.1.1).

Slika 17: Delotok za uvažanje podatkov iz SQL baze podatkov v KNIME



4 MODELIRANJE

V tem poglavju bom predstavila cikel modeliranja iz CRISP-DM predstavitve (Slika 2). Izdelava modelov je bila v mojem primeru logično razdeljena na štiri sklope: izbira metod, vzorčenje, optimizacija parametrov in izbira atributov. Te štiri sklope sem zaradi proceduralne povezanosti izvedla v dveh korakih:

- V prvem koraku sem izbirala metodo, v kar sem vključila tudi izbor najoptimalnejših parametrov kot tudi izbor najboljšega vzorčenja pri dani metodi in danih parametrih.
- V drugem koraku pa sem z izbranimi metodami naredila izbor najboljših atributov.

Ta dva koraka sem v ciklu večkrat ponovila, tako da sem po izboru najboljših atributov zopet iskala najoptimalnejše metode in tako naprej. Zaradi izredno zahtevnega opisa tega cikličnega postopka ne bom opisovala na način, kot je bil dejansko izveden, ampak bom opis poenostavila in v sledečih podpoglavjih predstavila vsak korak neodvisno od dejanskega sosledja dogodkov, torej tako kot da je bil vsak korak izveden zgolj enkrat.

Rezultat (izhod) vsakega koraka je bil vhod v naslednji korak. Pri prvem koraku sem tako iskala najboljše metode (glede na AUC kriterij), hkrati pa me je zanimala tudi časovna kompleksnost, saj sem oboje potrebovala za drugi korak, ki je bil časovno zelo zahteven (število atributov za testiranje je bilo veliko). Izhod iz drugega koraka pa je bil nabor atributov, ki je bil ponovno uporabljen v prvem koraku.

V dveh sledečih podpoglavjih predstavljam zgoraj opisana koraka modeliranja problema.

4.1 Izbira metod, vzorčenje in optimizacija parametrov

V tem podpoglavju bom predstavila postopek izbire klasifikacijskih in regresijskih metod, njihovih parametrov in načinov vzorčenja ter opis delotokov, ki sem jih sestavila za te potrebe.

4.1.1 Predstavitev delotoka

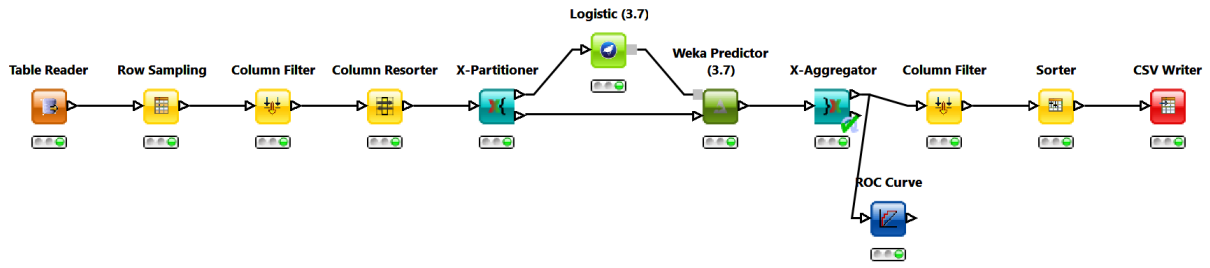
Delotok sem gradila v več korakih. Najprej sem sestavila delotok, v katerem sem na nekem vzorcu podatkov pognala določeno metodo oz. algoritem, ki sem ga s pomočjo prečnega preverjanja ovrednotila ter kot rezultat dobila krivuljo ROC in podatek AUC. V naslednjem koraku sem dodala merjenje časa izvajanja določenega algoritma. V zadnjem koraku pa sem delotok dopolnila še z dodatno tabelo, ki je vsebovala različne parametre (za vsak algoritem posebej) in različne vzorce nabora podatkov. Tako sem določen algoritem sistematično poganjala ob različnih parametrih na izbranih reprezentativnih vzorcih nabora podatkov, ki so vsebovali še nadvzorčenje in podvzorčenje podatkov. Delotok je bil enak za vse algoritme, spreminjala sem le vozlišče, ki vsebuje algoritem.

Imena vozlišč so v angleškem jeziku in jih je včasih težko prevesti. Zato v nadaljevanju uporabljam originalna imena vozlišč, za lažje razumevanje pa sem določena tudi opisala.

4.1.1.1 Enostavni klasifikacijski delotok

V prvem koraku sem zgradila enostavni delotok (Slika 18), v katerem sem prebrala podatke in na nekem vzorcu podatkov pognala določen algoritem, ki sem ga s pomočjo prečnega preverjanja ovrednotila ter na podlagi teh rezultatov dobila krivuljo ROC in podatek AUC (glej podpoglavje 2.5.3). Kot sem že omenila, sem za vsak algoritem sestavila enak delotok, razlika je bila le v vozlišču, ki vsebuje algoritem (Slika 18 vsebuje algoritem Logistic).

Slika 18: Osnovni delotok



Jedro delotoka predstavlja zanka za prečno preverjanje (glej podpoglavje 2.5.1), ki je sestavljena iz dveh vozlišč: X-Partitioner in X-Aggregator. Slednji na koncu vsake iteracije zbere rezultate. Vsa vozlišča, ki se nahajajo med tema dvema vozliščema, se izvedejo tolikokrat, kolikor iteracij določimo v prvem vozlišču. V mojem primeru sem izbrala 10 iteracij. Izhodna rezultata prvega vozlišča sta dve tabeli: tabela učnih primerov in tabela testnih primerov. V mojem primeru služi tabela učnih primerov kot vhod vozlišča z določenim algoritmom. Na teh podatkih se algoritem nauči ter kot rezultat vrne naučeni model. Tabela testnih primerov pa služi kot vhod vozlišča Weka Predictor, ki na vhodu dobi tudi naučen model. Testne primere nato oceni na naučenem modelu ter kot rezultat vrne klasificirane testne podatke oz. oceno za ciljno spremenljivko testnih podatkov.

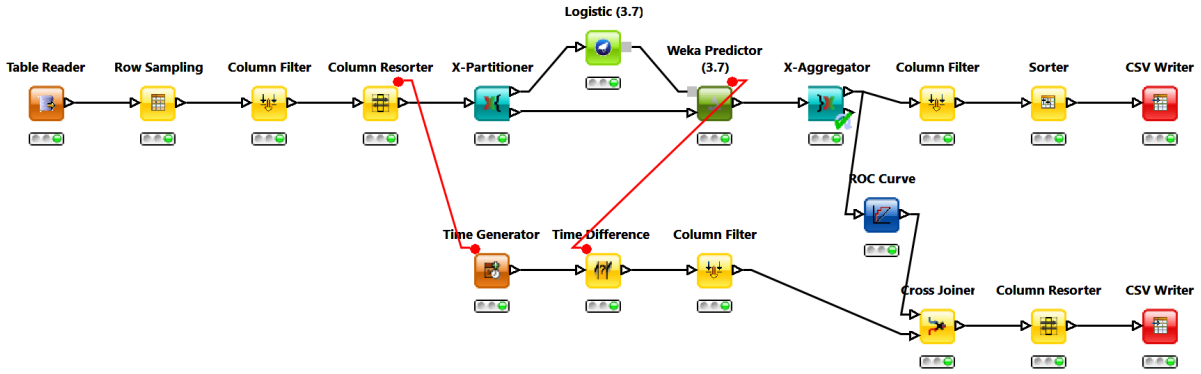
Rezultat sem ovrednotila s pomočjo ROC krivulje. Vozlišče ROC Curve kot rezultat nariše ROC krivuljo za klasifikacijski oz. regresijski problem, vrne pa tudi podatek AUC. Rezultati se shranijo tudi v tekstovno datoteko s končnico *.csv*, ker taka oblika ustreza programu Microsoft Excel 2010, v katerem sem zaradi enostavnejše primerjave skonstruirala ROC krivulje.

Ostala vozlišča so le pomožna in služijo za izbiranje, sortiranje, urejanje ipd., zato jih podrobneje ne predstavljam.

4.1.1.2 Meritev časa izvajanja

V drugem koraku sem delotok dopolnila z merjenjem časa izvajanja določene metode. Zanimalo me je, v kolikšnem času določena metoda uspe izvesti postopek učenja videnih primerov in testiranja novih primerov. Ta podatek sem potrebovala v kasnejši fazi podatkovnega rudarjenja, v fazi izbora pomembnih atributov (podroben postopek je predstavljen v podpoglavju 4.2). V delotok sem dodala še vozlišča za merjenje časa (Time Generator in Time Difference, Slika 19). Čas sem začela meriti, pred začetkom izvajanja zanke prečnega preverjanja, z merjenjem pa končala, ko se je zanka zaključila. Tudi tu sem rezultate zapisala v dokument *.csv*.

Slika 19: Delotok dopolnjen z merjenjem časa



4.1.1.3 Optimizacija parametrov in vzorčenje

V zadnjem koraku sem delotok dopolnila še z optimizacijo parametrov in vzorčenjem.

Optimizacije parametrov in vzorčenja sem se lotila sistematično. V delotok sem dodala tabelo s štirimi stolpci, ki so predstavljali posamezne parametre vzorčenja (prvi trije stolpci) ter parametre algoritma (četrti stolpec). Parametri vzorčenja so bili za vse algoritme enaki in so predstavljali velikost vzorca, podvzorčenje in nadvzorčenje podatkov, zadnji stolpec pa je vseboval različne parametre, ki so bili za vsak algoritem drugačni. Za vsak algoritem sem naredila kartezični produkt po vseh vrednostih teh parametrov in tako za vsak algoritem dobila različno tabelo parametrov.

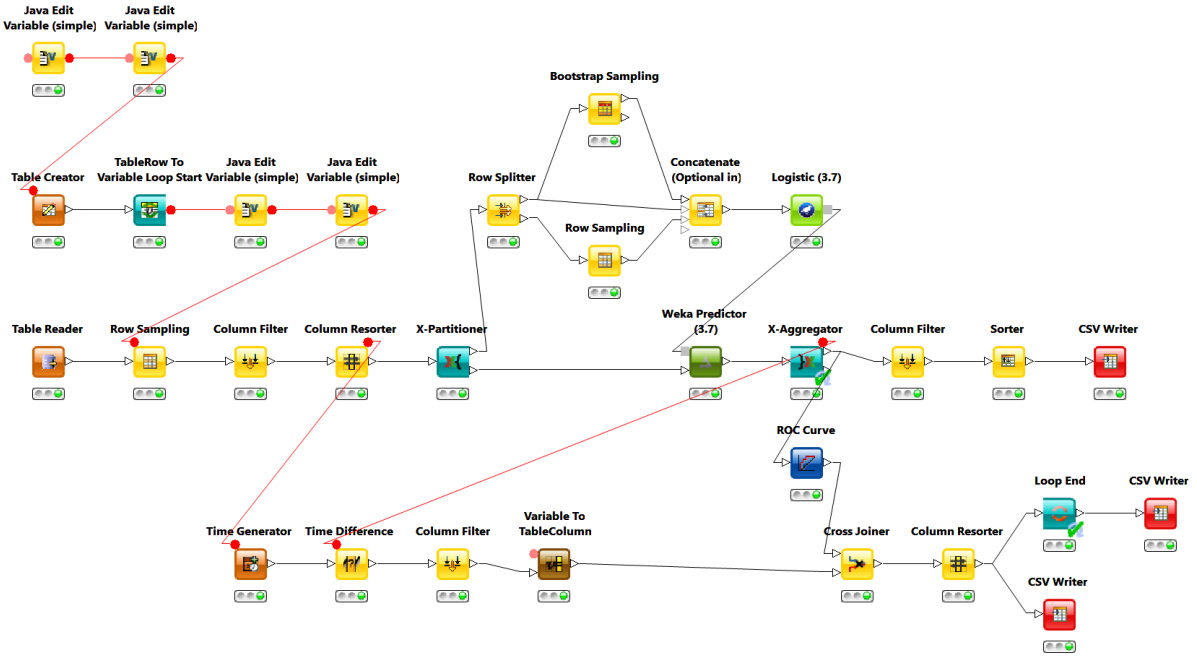
Tabela parametrov je predstavljala jedro nove zunanje zanke, ki je v vsakem ciklu uporabila eno vrstico iz tabele parametrov (vozlišče Table Row To Variable Loop Start pomeni začetek zanke, Loop End označuje konec zanke). Zanka se je ponovila tolikokrat, kolikor je bilo vrstic v tabeli. Kot je razvidno iz predstavitve (Slika 20), je bila zanka postavljena tako, da se je v vsaki ponovitvi zanke izvedel celoten proces rudarjenja podatkov, ki sem ga opisala v razdelkih 4.1.1.1 in 4.1.1.2, le-tega pa sem dopolnila še z nekaj modifikacijami za upoštevanje parametrov, ki jih navajam v nadaljevanju.

Največja dopolnitev delotoka iz razdelka 4.1.1.2 je podvzorčenje in nadvzorčenje podatkov, ki je bilo realizirano na naslednji način: z vozliščem Row Splitter, s katerim se filtrira vrstice glede na določen kriterij, sem v prvi izhodni tabeli dobila vse vrstice, ki ustrezajo filtru, v drugi pa vse vrstice, ki filtru ne ustrezajo. V mojem primeru sem filter nastavila na razred, torej ali je vrednost v stolpcu ali_skoda_AO enaka da. Prva izhodna tabela je tako vsebovala vse vrstice, ki imajo vrednost da v omenjenem stolpcu (pogodbe s škodami), druga izhodna tabela pa vrstice z vrednostjo ne (pogodbe brez škod):

- Iz tabele, ki vsebuje pogodbe brez škod, sem z vozliščem Row Sampling izbrala določen delež teh primerov. Ta del predstavlja podvzorčenje. Delež primerov je bil določen z vrednostjo celice za podvzorčenje iz zgoraj opisane tabele parametrov. Na ta način sem zmanjšala večinski razred.
- Tabelo, ki je vsebovala samo pogodbe z izplačanimi škodami, pa sem uporabila dvakrat. V prvem primeru sem vzela celotno tabelo, brez kakršnihkoli sprememb. V drugem primeru sem iz tabele z metodo izbiranja z vračanjem, izbrala določen delež primerov. Tudi delež sem dobila iz tabele parametrov iz celice za nadvzorčenje. Na ta način sem povečala manjšinski razred.

Z vozliščem Concatenate (Optional in) sem zlepila skupaj tri tabele, ki sem jih opisala zgoraj: tabelo vseh originalnih pogodb z izplačanimi škodami, tabelo vseh dodatnih pogodb z izplačanimi škodami, ki sem jih dobila iz nadvzorčenja, ter tabelo pogodb brez škod, ki sem jo dobila iz podvzorčenja. Ta nova tabela je predstavljala učno množico podatkov, ki deloma rešuje problem neuravnoteženosti podatkov (glej podpoglavje 2.4).

Slika 20: Delotok z vzorčenjem in optimizacijo parametrov



Ostale dopolnitve se nanašajo še na ostala dva parametra iz tabele parametrov. Prvi parameter oz. prvi stolpec v tabeli je vseboval podatke o velikosti vzorca. Ta podatek se je uporabil v vozlišču Row Sampling, v katerem se je izbral vzorec podatkov za rudarjenje. Podatki iz četrtega stolpca tabele parametrov pa so bili parametri, ki so bili specifični za posamezen algoritem. Ti podatki so se uporabili (v vsaki ponovitvi zanke en parameter) v vozlišču, ki vsebuje algoritem.

Ostala vozlišča so bila pomožna. V glavnem sem jih uporabila za definicije imen različnih datotek in dokumentov. Na tak način sem celoten proces rudarjenja podatkov precej avtomatizirala in se izognila tudi morebitnih napakam, saj sem spreminjala le imena algoritmov in njihove parametre. Ostali podatki so bili enaki za vse algoritme. Poleg tega sem si z izpisovanjem datotek izdelala popolno dokumentacijo o vsakem izvajanju delotoka, kar je bilo neprecenljivo zaradi velikega števila ponovitev izvajanja (tudi zaradi vmesnega čiščenja podatkov, evalvacije različnih algoritmov in najboljših parametrov, iskanja najboljšega nabora vzorčenja …).

4.1.2 Metodologija

Reševanje problema tega magistrskega dela sem začela z desetimi različnimi algoritmi (opis posameznega algoritma najdemo v podpoglavju 2.2). Dejanskih algoritmov, ki sem jih obravnavala v pred-analizi je bilo še mnogo več, vendar jih zaradi različnih vzrokov (po navadi so imeli omejitve, katere tipe atributov sprejemajo, npr. samo racionalne) nisem nadalje obravnavala. Deset končnih algoritmov, ki so prestali pred-analizo so:

1. Bayesove mreže
2. IBk
3. J48
4. JRip
5. Logistic
6. M5P
7. MLP
8. Naivni Bayes
9. Naključni gozdovi
10. SMO

Zaradi lažjega razumevanja in poenostavitve samega procesa rudarjenja podatkov sem poskušala metodologijo dela čim bolj sistematizirati ter se izogniti morebitnim napakam, zato sem uporabljala enake osnove za vse algoritme, čeprav so algoritmi med seboj precej različni in raznoliki. S takim sistemskim pristopom sem tako na enostaven način dobila dober vpogled v delovanje algoritma oz. njegove potrebe in zahteve.

V prvem koraku sem algoritme pognala na celotnem naboru podatkov, vendar se je izkazalo, da so določeni algoritmi prezahtevni in niso uspeli obdelati vseh podatkov ter posledično v doglednem času niso vrnili rezultata. Nekaterim celo ni uspelo zaključiti nisi enega samega procesa učenja znotraj 10-kratnega prečnega preverjanja.

Zato sem v naslednjem koraku vzela vzorec podatkov ter algoritme pognala na tej podmnožici. Velikosti vzorcev sem izbrala v deležih glede na prvotno množico nabora podatkov: 0,1 %, 0,2 %, 0,5 %, 1 %, 2 %, 5 %, 10 %, 20 %, 50 %, 100 %. Izbirala sem reprezentativne vzorce. Primer: če je bil vzorec velikosti 20 % glede na prvotno množico podatkov, je to pomenilo, da sem vzorec dobila tako, da sem iz prvotne množice podatkov izločila 80 % pogodb s škodami kot tudi 80 % tistih brez škod, vendar pa je njuno razmerje ostalo enako, torej 2,7 % pogodb s škodami glede na pogodbe brez škod. Deleže vzorcev sem določila empirično in ob nekajkratnih ponovitvah sem ugotovila, da deleži 0,1 %, 0,2 %, 0,5 % pomenijo premajhne vzorce podatkov in da so morebitni dobri rezultati (precej velik podatek AUC) le naključje zaradi slučajnega dobrega vzorca. Zato sem v končni izbor vzela le deleže večje od 1 %.

Zaradi velike časovne kompleksnosti določenih algoritmov pa sem začela meriti čas izvajanja določenega algoritma. S tem sem pridobila informacijo, kateri algoritmi so časovno zelo zahtevni in posledično tudi manj uporabni. Ta podatek je bil ključen v naslednji stopnji modeliranja, tj. pri izboru najpomembnejših atributov. Pri tem izboru je bilo potrebno določen algoritem velikokrat ponoviti, zato je bilo potrebno imeti na razpolago tisto kombinacijo parametrov vzorčenja določenega algoritma, ki je dovolj hitra, a pri tem vseeno rezultira v čim boljši rezultat.

Nadalje je bilo potrebno uporabiti nadvzorčenje oz. podvzorčenje, ker se, kot sem že omenila, omenjena problematika uvršča v probleme neuravnoteženih podatkov. V naboru podatkov, ki sem ga uporabila, je manjšinski razred vseboval pogodbe z izplačanimi škodami, večinski razred pa pogodbe brez škod. Razmerje med manjšinskim in večinskim razredom je bilo 2,7 proti 100.

Uporabila sem različne kombinacije nadvzorčenja in podvzorčenja podatkov. V mojem primeru je to pomenilo, da sem povečala razred pogodb s škodami ali pa zmanjšala razred pogodb brez škod. Tudi postopke vzorčenja sem uporabila za vse algoritme enako, čeprav se algoritmi med seboj razlikujejo tudi v tem, ali potrebujejo uravnotežene ali neuravnotežene podatke za uspešno izvedbo. Tako sem v učni množici primerov

- za povečanje manjšinskega razreda (nadvzorčenje) določila tri vrednosti, 0, 100 in 300, kar pomeni naslednje:
    - 0 pomeni, da je razred pogodb s škodami ostal nespremenjen, torej 2,7 % pogodb s škodami glede na pogodbe brez škod,
    - 100 pomeni, da sem razred pogodb s škodami povečala za 100 % - torej na dvakratno vrednost: vzela sem vse primere pogodb s škodami, novih 100 % pogodb s škodami pa sem generirala tako, da sem pogodbe s škodami izbrala z metodo izbiranja z vračanjem (angl. *bootstrap*), razmerje je bilo 5,4 % pogodb s škodami glede na pogodbe brez škod.
    - 300 pomeni, da sem razred pogodb s škodami povečala za 300 % - torej na štirikratno vrednost, tudi tu sem uporabila metodo izbiranja z vračanjem, razmerje je bilo 10,8 % pogodb s škodami glede na pogodbe brez škod;
- za zmanjšanje večinskega razreda (podvzorčenje) sem vzela določene deleže primerov (1,4 %, 2,7 %, 5 %, 10 %, 20 %, 50 % in 100 %) pogodb brez škod iz prvotne tabele. Tako npr. delež 2,7 % pomeni, da iz večinskega razreda iz prvotne tabele naključno izberem samo 2,7 % vseh pogodb brez škod – v tem primeru dobim razmerje pogodb s škodami in pogodb brez škod približno 1:1.

Iz naborov za nadvzorčenje, podvzorčenje ter iz nabora različno velikih vzorcev sem naredila kartezični produkt ter dobila 147 možnih kombinacij (produkt 3 izbir za nadvzorčenje, 7 izbir za podvzorčenje, 7 izbir za velikost vzorca). Te kombinacije sem kombinirala še s parametri, ki pa so se od algoritma do algoritma razlikovali. Na vseh možnih kombinacijah sem nato pognala izbranih 10 algoritmov. Primer možne kombinacije parametrov: če sem pri nadvzorčenju izbrala vrednost 0, pri podvzorčenju vrednost 2,7 % in velikosti vzorca 10 %, sem dobila uravnoteženo množico podatkov s približno 80.000 pogodbami (10 % od 800.000 vseh pogodb), saj je 2,7 % ravno delež pogodb s škodami glede na pogodbe brez škod in v tem primeru sem med ne-škodnimi pogodbami vzela ravno 2,7 % vseh pogodb.

Število parametrov pa je bilo za vsak algoritmov različno, pri vsakem algoritmu so bili nekateri parametri že privzeti. Vendar pa privzeti parametri niso nujno tudi najboljši. Za vsak posamezen primer je smiselno parametre pregledati oz. preizkusiti različne možnosti. Tako je bil algoritem SMO primer, kjer je bila privzeta vrednost za parameter jedro PolyKernel popolnoma neustrezna za moj nabor podatkov in je pri vseh kombinacijah vračala vrednosti AUC okrog 50%. Ko sem nastavitev za jedro zamenjala z RBFKernel, pa sem dobila precej boljše rezultate za AUC, nekatere kombinacije so se uvrstile celo med najboljše rezultate (na Pareto fronto, kot je opisano v nadaljevanju).

Algoritem Naključnih gozdov pa je primer, ko sem pri različnih nastavitvah parametra *število dreves* dobila dobre rezultate. V tem primeru sem na grafu prikazala združene rezultate za različne nastavitve tega parametra (40, 50 ali 100 dreves), saj so se kombinacije algoritma ob različnih nastavitvah tega parametra uvrstile med najboljše – in ne zgolj pri eni nastavitvi.

4.1.3 Rezultati

Pri reševanju omenjene problematike sem uporabila opisanih deset algoritmov za vsakega izmed njih pa sem v končnem koraku empirično določila najboljšo kombinacijo parametrov (velikost vzorca, podvzorčenje, nadvzorčenje, parametri algoritma; glej razdelek 4.1.1.3).

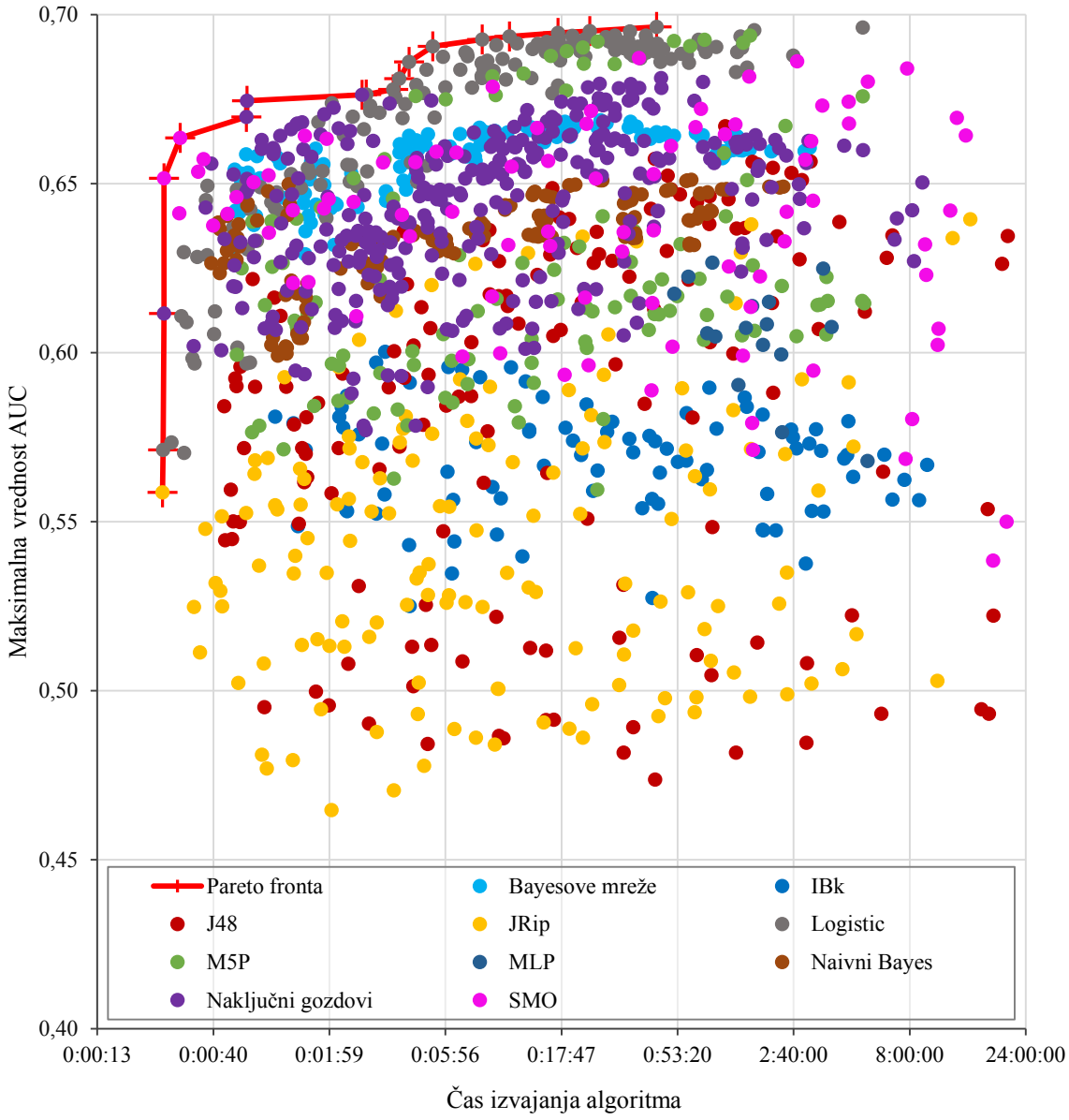
Posamezne kombinacije parametrov in algoritmov sem poganjala v časovno naraščajočem vrstnem redu, kar pomeni, da sem najprej pognala tiste kombinacije parametrov, ki so se na testiranju izkazale za hitre, in šele nato počasnejše kombinacije. Izkazalo se je, da so časovno zelo zahtevne kombinacije (čas se je tu meril v dnevih) lahko dale tudi slabše rezultate od hitrejših kombinacij, pri nekaterih kombinacijah pa se proces sploh ni zaključil, zato take kombinacije dejansko niso bile uporabne. Izpadle pa so tudi tiste kombinacije, pri katerih sem izbrala parametre algoritmov, ki niso ustrezali mojemu naboru podatkov. Take kombinacije se sploh niso izvedle ali pa so bili rezultati zelo slabi (AUC je bila okoli 50 % oz. naključni model, kar pomeni, da izbrani parametri algoritma niso ustrezni).

Meri za uspešnost algoritma v tej fazi sta bili zame vrednost AUC in podatek, v kolikšnem času se določen algoritem izvede. Na podlagi teh dveh mer sem iskala algoritme, ki mi v vnaprej definiranemu času dajo najboljšo rešitev, torej največjo vrednost AUC. Problem iskanja take optimalne rešitve se uvršča med probleme več-objektivne ali Pareto optimizacije (angl. *multi-objective optimization, Pareto optimization*). V tem primeru iščem optimizacijo po dveh parametrih, po najkrajšem času izvajanja določenega algoritma in največji vrednosti AUC. Rezultat je Pareto krivulja ali Pareto fronta (angl. *Pareto curve, Pareto front*), ki v mojem primeru vsebuje vse tiste algoritme, ki so najboljši (oz. se jih splača podrobneje ogledati in morebiti še izboljšati rezultat). Najboljše kombinacije parametrov in algoritma so tiste, ki dajo v danem času najboljši rezultat, kar pomeni, da ne obstaja noben hitrejši algoritem, ki bi dal enak ali boljši AUC (Slika 21). S pomočjo Pareto fronte sem lažje določila, katera kombinacija določenega algoritma je najbolj uporabna za naslednjo stopnjo modeliranja, torej za izbor najpomembnejših atributov.

Na grafu (Slika 21) so prikazane vrednosti AUC v odvisnosti od časa izvajanja metode. Vsak algoritem je označen s svojo barvo, vsaka točka pomeni eno kombinacijo določenega algoritma. Opazimo, da nimajo vsi algoritmi enako število točk. Razlog je ta, da že na začetku niso vsi algoritmi štartali z enakim številom kombinacij (razlog tiči v različnem številu parametrov za določen algoritem, čeprav je bilo število vzorcev ter število za podvzorčenje in nadvzorčenje enako), ostali razlogi pa so v tem, da se določene kombinacije niso izvedle do konca, ali pa so bile nastavitve parametrov neustrezne in je bil rezultat slab za vse vzorce in za vse nabore podvzorčenja in nadvzorčenja. Take kombinacije sem izločila iz rezultatov, ker bi le-te

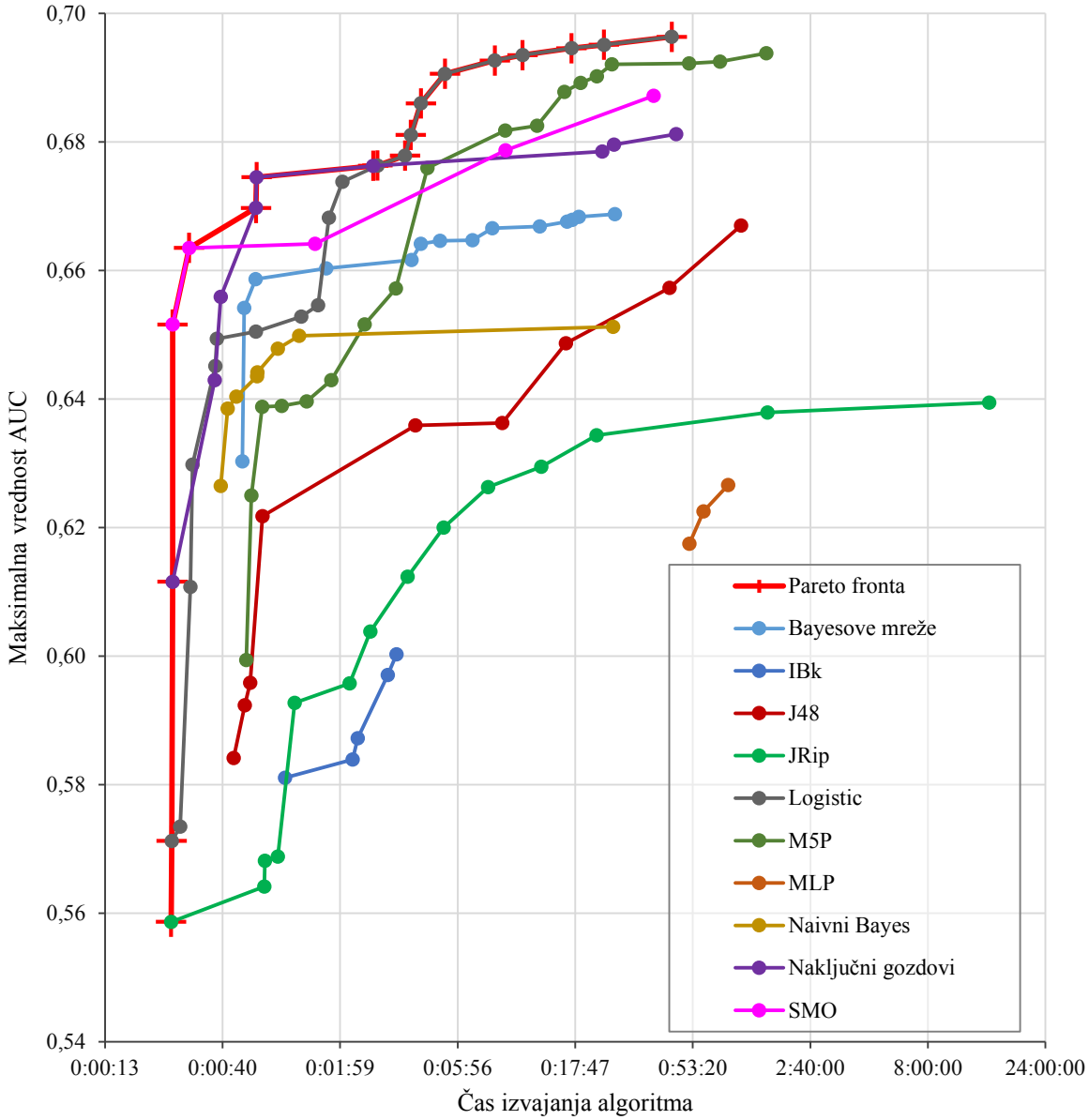
pokvarile sliko. Iz grafa je razvidno tudi, da je bil najbolj uspešen algoritem Logistic, saj deset njegovih kombinacij leži na Pareto fronti (sive točke na rdeči liniji).

Slika 21: Pareto fronta



Za bolj nazorno primerjavo med algoritmi, sem določila tudi Pareto fronte za vsak algoritem posebej (Slika 22). Kandidati za nadaljnjo obravnavo so bili tisti algoritmi, katerih Pareto fronte so bile del skupne Pareto fronte, ali pa so bili blizu le-te. Zelo uspešni so bili algoritmi Logistic, Naključni gozdovi ter SMO, katerih točke ležijo na Pareto fronti. Čeprav točke ne ležijo na Pareto fronti, sem v nadaljnjo analizo uvrstila še algoritem modelnega drevesa M5P ter Naivnega Bayesa. Slednji je služil za primerjavo, kot osnova za merilo kompleksnosti problema in soodvisnosti atributov.

Slika 22: Pareto fronte za posamezne algoritme



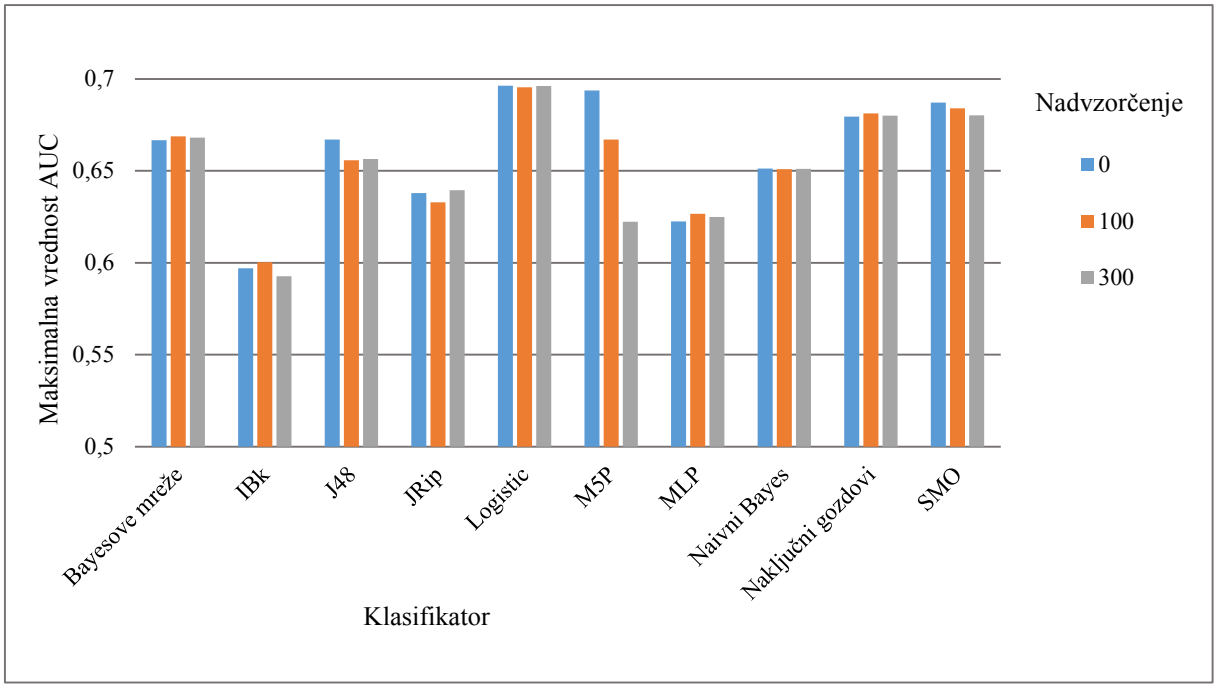
Analiza rezultatov je pokazala tudi, kako določene vrednosti parametrov vplivajo na določen algoritem.

Najprej predstavljam vpliv nadvzorčenja na rezultate. Na grafu (Slika 23) je na *x* osi prikazanih vseh deset algoritmov, *y* os pa prikazuje vrednosti AUC. Pri vsakem algoritmu so prikazani trije stolpci, vsak stolpec pa prikazuje največjo vrednost AUC, ki jo je določen algoritem dosegel pri določenem nadvzorčenju. Opazimo naslednje:

- Na algoritme Bayesove mreže, Logistic, Naivni Bayes ter Naključni gozdovi nadvzorčenje nima vpliva oz. ima zanemarljivo majhen vpliv.

- Pri IBk in MLP se nekoliko izboljša rezultat, če razred pogodb s škodami povečam na dvakratno vrednost (vzamem vrednost 100), vendar je izboljšava minimalna (iz 59,7 % na 60 % pri IBk, oz. iz 62,25 % na 62,66 % pri MLP), kar je lahko tudi zgolj naključje.
- Pri drevesih J48 in M5P ter pri SMO pa se rezultat celo poslabša. Največje poslabšanje je pri modelnem drevesu M5P, ko rezultat pade iz 69,38 % na 66,7 % pri enkratnem povečanju pogodb s škodami oz. na 62,2 % pri trikratnem povečanju pogodb s škodami.

Slika 23: Vpliv nadvzorčenja podatkov na rezultat algoritmov

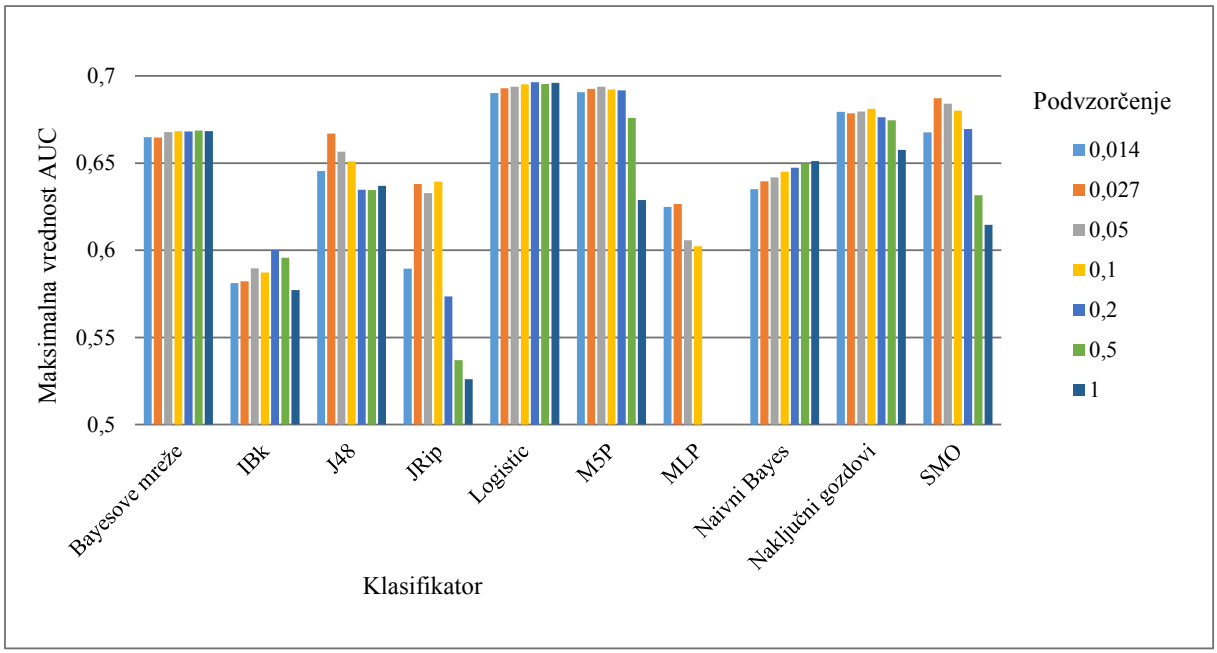


Podvzorčenje podatkov ima večji vpliv na rezultat (Slika 24):

- Pri Naivnem Bayesu rezultat počasi raste, medtem ko večamo večinski razred, čeprav to povečanje ni zelo izrazito (rezultat zraste iz 63,5 % pri podvzorčenju 1,4 % na 65 % pri podvzorčenju 100 %).
- Pri algoritmih IBk, J48, JRip in SMO rezultati precej nihajo. Pri odločitvenem drevesu J48 je rezultat najboljši, če vzamem vrednost za podvzorčenje 2,7 %, kar pomeni, da dobim uravnoteženo učno množico podatkov, kar pravzaprav ni presenetljivo, saj za le-tega vemo, da potrebuje uravnotežene podatke za uspešno delovanje. Podobno se obnašata tudi SMO in JRip.
- Po drugi strani pa neuravnoteženi podatki modelnega drevesa M5P ne motijo, oz. zmoti ga prevelik večinski razred (pri vrednosti 100 %, kar pomeni, da vzame vse pogodbe brez škod).
- Pri algoritmih, ki so se izkazali za precej uspešne na mojih podatkih, Bayesove mreže, Logistic ter Naključni gozdovi, pa ima podobno kot nadvzorčenje tudi podvzorčenje zanemarljiv vpliv, kar pomeni, da so ti algoritmi primerni za neuravnotežene podatke.

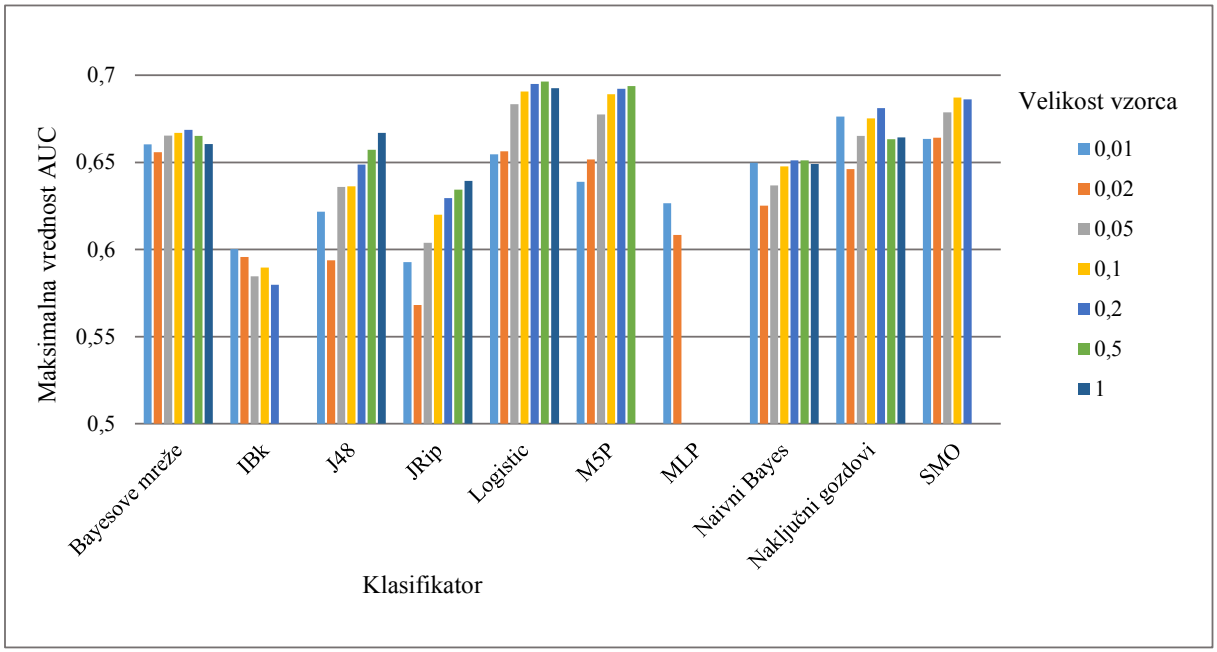
- Pri algoritmu MLP pa vidimo, da je bil algoritem zelo počasen in mu za določene večje vrednosti podvzorčenja sploh ni uspelo obdelati nobene kombinacije.

Slika 24: Vpliv podvzorčenja podatkov na rezultat algoritmov



Tudi velikost vzorca ima razmeroma velik vpliv na rezultat (Slika 25):

Slika 25: Vpliv velikosti vzorca podatkov na rezultat algoritmov



- To je očitno predvsem pri drevesih J48 in M5P ter pri algoritmih JRip in SMO. Tudi pri algoritmu Logistic se rezultat izboljšuje, čeprav je pri 20 % vseh podatkov enako dober kot pri 50 % podatkov ali pri vseh podatkih.
- Za Bayesove mreže in Naivnega Bayesa izrazitega padca ali povečanja rezultata ne opazimo, podobno je tudi pri Naključnih gozdovih, le da so tu nihanja malce večja.
- Pri IBk se rezultat poslabšuje, vendar je poslabšanje minimalno.
- Za MLP pa je težko oceniti, ali bi se rezultat poslabšal ali povečal, saj sem imela premalo rezultatov, ker algoritmu, kot že rečeno, ni uspelo obdelati veliko kombinacij.

S pomočjo zgornje analize sem tudi v naslednji fazi izbora najboljših atributov izbrala ustrezne kombinacije. Za nadaljnje potrebe sem na podlagi zgornje analize izbrala naslednje algoritme oz. kombinacije njihovih parametrov (v oklepaju navajam parametre, ki sem jih izbrala za določen algoritem: velikost vzorca, podvzorčenje, nestandardni parametri algoritma; nadvzorčenje sem iz nadaljnje analize izpustila, saj ima na končen rezultat zanemarljiv vpliv):

- Logistic (20 %, 20 %),
- modelno drevo M5P (20 %, 5 %),
- SMO (5 %, 2,7 %, RBF Kernel),
- Naključni gozdovi (20 %, 2,7 %, število dreves 40),
- Naivni Bayes (20 %, 100 %), ki pa sem ga uporabila kot osnovo za primerjanje.

Na teh 5 algoritmih sem v naslednji fazi naredila tudi izbor atributov (angl. *attribut selection*).

4.2 Izbira najprimernejših atributov

Izbira najpomembnejših atributov je zelo pomemben del podatkovnega rudarjenja. Tudi v mojem primeru se je izkazalo, da je možno z ustreznim naborom atributov izboljšati učinkovitost učenja ter povečati kakovost rezultatov. Pri določenih zahtevnejših metodah je bila izbira atributov celo predpogoj, saj brez tega metode niso zmogle obdelati količine podatkov, ki bi zadostila mojim potrebam. Pri reševanju problema sem uporabila pristope, ki sem jih navedla v podpoglavju 2.3.

Podpoglavje je sestavljeno podobno kot 4.1, kjer sem najprej predstavila delotoke, nato metodologijo dela in na koncu še rezultate. Razlika pa je v tem, da je bil postopek izbire atributov precej bolj obširen in zapleten kot pa sama izbira ustreznih metod oz. algoritmov, saj sem preizkusila več različnih pristopov, zato je bilo potrebno sestaviti tudi več delotokov. Vsled temu v prvem razdelku predstavljam delotoke za vsak pristop posebej, prav tako bom v drugem razdelku opisala vse pristope, ki sem jih uporabila, tretji razdelek pa vsebuje rezultate.

4.2.1 Predstavitev delotokov

Postopek izbire atributov je kompleksen in posledično so bili tudi delotoki, ki sem jih sestavila za ta namen, zapleteni. V podpoglavju 4.1.1 sem delotoke opisala na način, da sem opisala

glavna vozlišča, iz teh opisov pa je bilo razvidno tudi samo delovanje določenega delotoka. V tem podpoglavju pa se bom raje osredotočila na opis vsebine delotoka, saj bi bil sicer opis preveč zapleten in tudi nerazumljiv. Večino vozlišč, ki sem jih uporabila v teh delotokih, sem že predstavila v podpoglavju 4.1.1, dodala pa sem še nekaj novih, vendar imajo vsa nova le pomožno vlogo, bistvo delotoka ostaja enako.

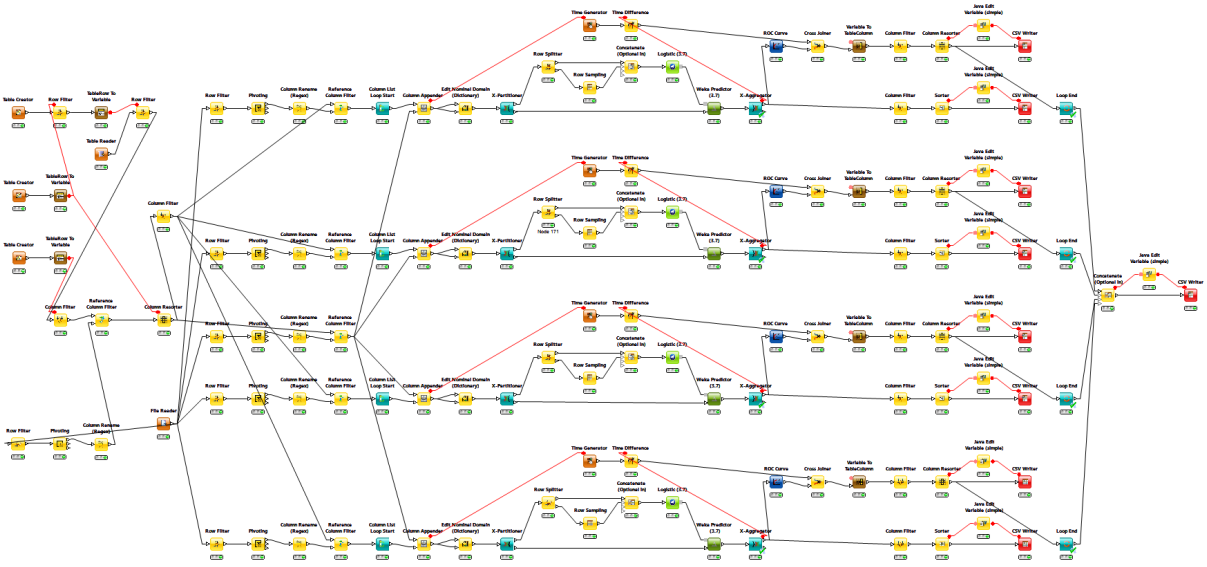
4.2.1.1 Odstranjevanje atributov

Za pohitritev postopka sem delotok razdelila na štiri enake dele, kar pravzaprav pomeni, da sem množico atributov, ki sem jih odstranjevala, razdelila na štiri podmnožice. Tehnično mi je ta razdelitev omogočila štiri sočasna izvajanja odstranjevanj in posledično štiri krat hitrejšo izvedbo eksperimenta. V procesu odstranjevanja atributov sem v prvem koraku iz vsake podmnožice (v nadaljevanju jih imenujem grupe) odstranila po en atribut in na množici atributov z odstranjenim enim atributom izvedla postopek klasifikacije oz. regresije ter izračunala AUC. Na prvi predstavitvi (Slika 26) je prikazan celoten delotok, na drugi (Slika 27) pa je prikazan le eden od štirih enakih delov iz celotnega delotoka.

Najprej predstavljam celotni delotok. Na začetku s pomočjo vozlišč Table Creator definiram imena datotek, v katere se shranjujejo rezultati, ter poimenujem uporabljen algoritem. S pomočjo vozlišča File Reader preberem tekstovni dokument, v katerem so shranjeni atributi skupaj s podatki o grupah, ki se bodo uporabili v določenem koraku učenja. Ta tekstovni dokument si pripravim posebej ter za vsako metodo svojega, v njem pa beležim, katere atribute sem odstranila in v katero grupo sem jih uvrstila. Na skrajnem desnem koncu delotoka se po zaključenih zankah (štiri vozlišča Loop End) rezultati vseh štirih delov z vozliščem Concatenate (Optional in) združijo v eno tabelo.

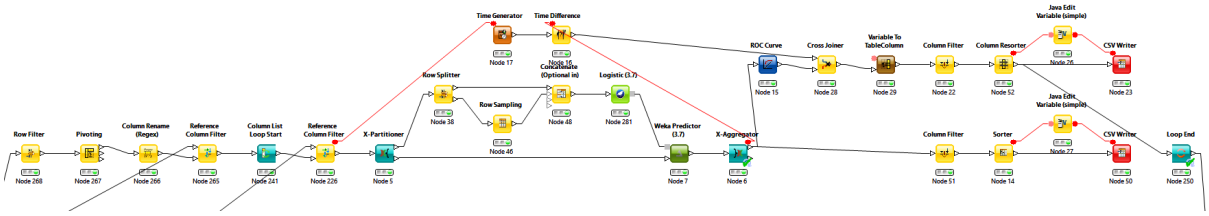
Osnova testiranja pa je že znani del prečnega preverjanja, ki sem ga predstavila v razdelku 4.1.1.3, okoli tega pa so pomožna vozlišča. To je predstavljeno tudi na diagramu (Slika 27),

Slika 26: Shematski prikaz celotnega delotoka za odstranjevanje atributov



kjer predstavljam enega od štirih delov celotnega delotoka. Pred zanko prečnega preverjanja določim nabor stolpcev oz. atributov, ki jih v določenem koraku učenja potrebujem: uporabim vozlišča Row Filter, Pivoting, Column Rename (Regex), Reference Column Filter ter zanko Column List Loop Start, katera v vsakem koraku iz vhodne tabele vzame en atribut, ki ga v vozlišču Reference Column Filter odstranim iz tabele, ki vstopi v zanko prečnega preverjanja. Na drugi strani zanke prečnega preverjanja pa so že znana vozlišča, ki skrbijo predvsem za to, da se rezultati pravilno zapišejo v določene datoteke.

Slika 27: Shematski prikaz osnove delotoka za odstranjevanje atributov

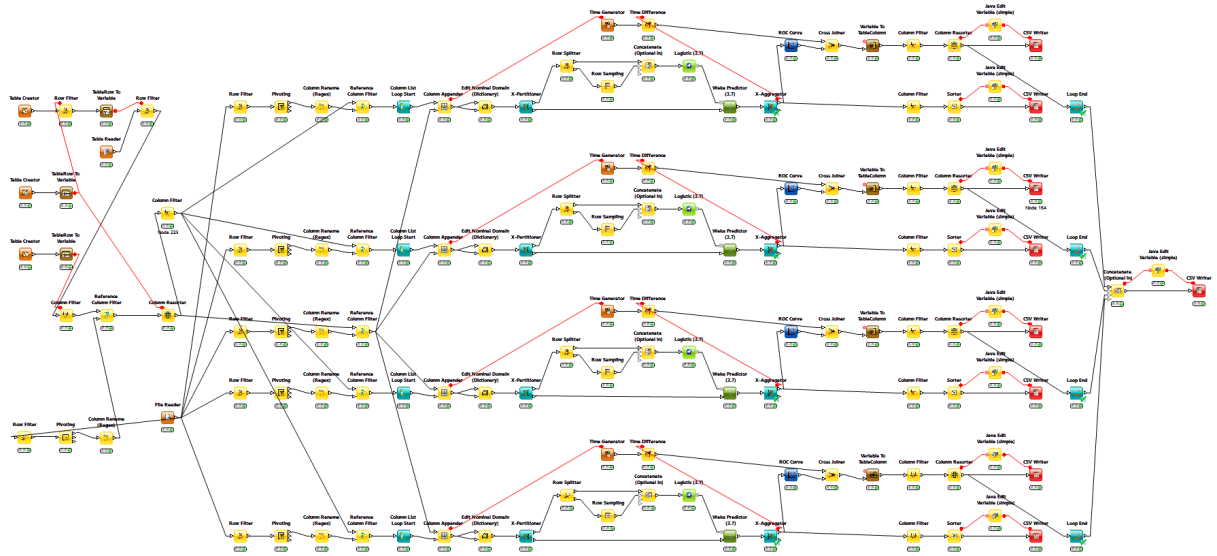


Osnova testiranja je že znana zanka prečnega preverjanja, ki vsebuje tudi vozlišče z določenih algoritmom, le da se od delotoka iz razdelka 4.1.1 razlikuje v tem, da v tem delotoku nisem uporabila nadvzorčenja, saj sem na podlagi analize in rezultatov iz podpoglavja 4.1.3 ugotovila, da nadvzorčenje na izbranih algoritmih ne vpliva na rezultat. Podvzorčenje in ostala vozlišča v zanki so ostala enaka.

4.2.1.2 Dodajanje atributov

Delotok za dodajanje atributov (Slika 28) je zelo podoben delotoku za odstranjevanje atributov, ki sem ga opisala v prejšnjem razdelku 4.2.1.1 (Slika 26), zato bom navedla le razlike.

Slika 28: Shematski prikaz celotnega delotoka za dodajanje atributov



Tudi v tem delotoku uporabim podoben tekstovni dokument, kot sem ga imela pri odstranjevanju atributov (vozlišče File Reader), kjer v vsakem koraku dodajanja atributov shranjujem nabor atributov skupaj s podatki o grupah.

Glavna razlika pa je v izboru atributov, ki se jih uporabi pri učenju. Ta del je predstavljen z vozlišči Row Filter, Pivoting, Column Rename (Regex), Reference Column Filter, kjer s pomočjo podatkov o grupah iz tekstovnega dokumenta določim, kateri atributi se uporabijo v določenem koraku učenja. S Column List Loop Start v vsakem koraku zanke izberem en atribut, ki ga v vozlišču Column Appender dodam že obstoječi množici atributov. Ta nova množica potem vstopi v zanko prečnega preverjanja.

4.2.2 Metode dela

V nadaljevanju predstavljam pristope, ki sem jih uporabila pri izboru atributov. Pristope predstavljam v kronološkem vrstnem redu.

4.2.2.1 Ročna izbira atributov

V prvem koraku sem se lotila ročne izbire. Iz množice atributov sem odstranila tiste atribute, ki so se že na prvi pogled zdeli neuporabni ali pa napačni (npr. podatka o tem, ali ima določen zavarovalec odprt škodni spis na zavarovanju AO ne smem uporabiti, saj mi ta podatek že pove, da je bila na določeni pogodbi izplačana škoda). Odstranila sem tudi vse atribute, ki ob sklenitvi zavarovanja niso znani, saj je tako zastavljen moj problem (npr. ob sklenitvi zavarovanja ne vem, ali bo pogodba prekinjena in zato atributa, ki vsebuje podatek o prekinitvi pogodbe ne smem upoštevati). Tako sem odstranila naslednje atribute:

- atributi, ki pomenijo šifro oz. kodo, so za vsak zapis različni in ne nosijo nobene informacije o določeni pogodbi, npr. pogodba id, koda opreme, številka izkaznice, številka šasije, zavarovalec id …;
- atribute, ki vsebujejo opise, npr. opomba, opis vozila …;
- atribute, ki vsebujejo datume, npr. datum začetka in datum konca pogodbe; te atribute sem že v fazi priprave podatkov spremenila v atribute, ki vsebujejo podatek o starosti, ki dejansko pomeni lastnost vozila ali osebe;
- atribute, ki vsebujejo podatek o višini premije za določeno zavarovanje, saj nam taki atributi posredno že nakažejo, ali je določen zavarovalec tvegan (npr. visoka premija AO že nakazuje, da je zavarovalec najbrž v višjem premijskem razredu, kar pomeni, da je v preteklosti imel škodo); take atribute sem v fazi priprave podatkov spremenila v binarne atribute, ki vsebujejo podatek, ali je določeno zavarovanje sklenjeno ali ne, npr. atribut ali_premija_AK ima zalogo vrednosti 0 in 1, kjer nam 1 pove, da je sklenjeno zavarovanje AK, 0 pa, da to zavarovanje ni sklenjeno – ta atribut sem izpeljala iz atributa premija_AK, ki vsebuje podatek o višini premije, tako da sem naredila preslikavo in vse primere, ki imajo premijo AK večjo od 0, preslikala v vrednost 1, če pa je bila vrednost enaka 0 ali NULL, sem le-te preslikala v vrednost 0;

- atribute, ki vsebujejo podatek o višini škode, o številu škodnih spisov ali o škodni rezervaciji za določeno zavarovanje, saj je to direkten podatek o tem, ali je bila škoda na nekem zavarovanju.

Tako sem dobila množico 119 atributov, na tej množici pa sem v naslednjih fazah preizkusila še druge metode za selekcijo atributov.

4.2.2.2 Izbira atributov s pomočjo znanja iz zgrajenih modelov

V drugem koraku sem poskusila pristop, kjer se atribute izbere s pomočjo učnega algoritma, ki se ga potem uporabi tudi za podatke. Atributov nisem izbirala, ampak sem na množici 119 atributov in vseh podatkih pognala dva različna transparentna algoritma, odločitveno drevo J48 in Logistic, kjer lahko rezultat v obliki modela naučenega znanja tudi prikažemo, kar v tem primeru pomeni, da lahko vidimo, katere atribute sta upoštevala pri učenju.

Odločitveno drevo J48 in Logistic sta vrnila precej nasprotujoči si rešitvi, kar je bilo mogoče tudi pričakovati, saj je ta pristop očitno naravnan na sam algoritem in zaradi tega posledično tudi ne tako splošno uporaben.

Vendar pa rezultati niso zadostili mojim pričakovanjem, saj so bili nekateri uporabljeni oz. odstranjeni atributi nasprotujoči razumskemu razmisleku o pomembnosti. Prav tako mi novi modeli zgrajeni iz tako pridobljene selekcije atributov niso nudili konsistentno dobrih rezultatov. Zato se v nadaljevanju s to metodo nisem ukvarjala, ampak sem raje uporabila naslednja pristopa.

4.2.2.3 Odstranjevanje atributov

V tretji fazi sem izvedla postopek odvzemanja atributov, ki ustreza enosmerni požrešni metodi, kjer na vsakem koraku en atribut odvzamem, ocenim rezultat, atribut vrnem v množico ter odvzamem naslednji atribut (glej poglavje 2.3). Ta metoda, v kolikor jo izvajamo s principom odstranitve enega atributa na enkrat, je bolj varna v smislu iskanja optimalne množice atributov, saj se z odstranitvijo atributa odstranijo tudi vse informacije, ki jih ta atribut morebiti nosi v kombinaciji z ostalimi atributi.

Začela sem z množico 119 atributov, ki sem jo dobila v koraku ročne izbire (glej razdelek 4.2.2.1). Poskus sem izvedla ločeno na vseh petih kombinacijah izbranih algoritmov. Za vse algoritme sem imela podatek AUC na celotni množici atributov (v nadaljevanju osnovna AUC). Z izborom atributov sem želela izboljšati ta rezultat, kar pomeni povečati AUC.

Postopek odstranjevanje atributov sem začela s testom odstranitve vsakega atributa posebej - iz množice 119 atributov sem odvzela en atribut ter na preostali množici atributov pognala učni algoritem. Vsakokrat sem izračunala AUC. V naslednjem koraku sem atribut, ki sem ga odstranila, vrnila v množico atributov ter odvzela naslednji atribut ter pognala učni algoritem

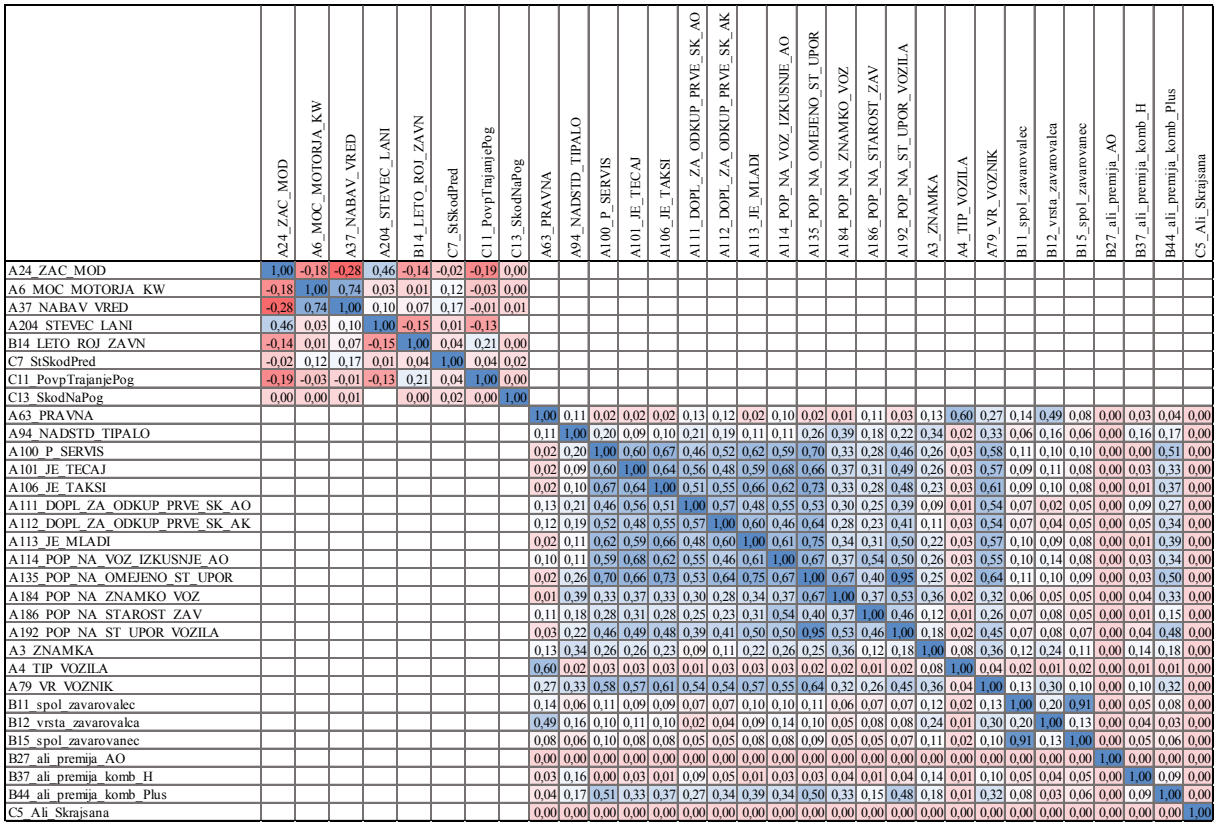
na novi množici. V vsaki ponovitvi zanke sem v prvem koraku pognala učni algoritem na množici 118 atributov.

Rezultat tega postopka je bila tabela vseh 119 atributov s pripadajočimi AUC vrednostmi. V idealnem primeru bi med temi atributi našla tistega z največjim AUC in ga odstranila iz množice atributov ter tako dobila množico 118 atributov, na katerih bi testno odstranjevanje ponovila. Težava tega pristopa je velika časovna zahtevnost, saj je potrebno za odstranitev enega atributa preizkusiti vse atribute v trenutnem naboru. Zaradi prezahtevnosti tega postopka sem po vsakem testiranju atributov odstranila več atributov naenkrat. Tako sem na primer pri algoritmu Logistic najprej odstranila 31 najslabših atributov ter enak postopek ponovila na 88 atributih. V naslednjih korakih sem odstranila še 24, 20, 20 in nato še 10 atributov, tako da sem dobila končno množico 14 atributov. Atribute sem izbirala za vsakega izmed petih algoritmov posebej, tako da sem na koncu dobila pet množic z izbranimi atributi, za vsak algoritem svoj nabor atributov. Število odstranjenih atributov v posameznem koraku je pri vseh algoritmih približno sledilo zgoraj podanim vrednostim za primer Logistic, vendar pa so bili zaključki različni od primera do primera (npr. algoritem M5P je končal s 5, Naključni gozdovi pa s 64 atributi).

Postopek določanja atributov za odstranitev je potekal tako, da sem najprej v tabeli rezultatov uredila vrednosti AUC naraščajoče ter tako dobila pregled, kateri atributi vplivajo na AUC rezultat najbolj pozitivno (večji AUC pomeni, da sem z odstranitvijo atributa algoritmu pomagala doseči večjo natančnost). Pri izbiri pa sem pazila tudi na morebitne odvisnosti oz. korelacije med atributi. Odvisni atributi namreč lahko pokvarijo rezultat, zato sem vse atribute, ki so bili očitno korelirani med seboj (npr. zavarovalna vsota za stvari in zavarovalna vsota za osebe), odstranila ročno, pri ostalih, kjer korelacija ni bila očitna, pa sem si pomagala s korelacijsko matriko (Slika 29). Če sta bila dva ali več atributov med seboj pozitivno oz. negativno korelirani (koeficient je blizu 1 oz. -1), sem odstranila samo tiste, z odstranitvijo katerih sem izgubila manj informacije (atributi, ki so imeli v tabeli rezultatov večji AUC). Atribute sem odstranjevala v takem vrstnem redu, da sem najprej odstranila korelirane atribute (le eden izmed koreliranih atributov je ostal za nadaljnje učenje), šele nato sem odstranila ostale slabe atribute. Če je bilo več takih atributov, ki so imeli enak vpliv na rezultat (kar pomeni približno enake vrednosti AUC) in so bili vsaj približno enako nekorelirani, sem po lastni presoji izbrala katere odstraniti.

Korelacijska matrika izračuna korelacijski koeficient med dvema atributoma. V vrsticah in stolpcih so navedeni atributi, celice v matriki so korelacijski koeficienti. Matrika je simetrična, po diagonali so enke, saj je vsak atribut koreliran sam s seboj. Korelacijski koeficient je mogoče izračunati med dvema numeričnima atributoma ali pa med dvema nominalnima atributoma. Za ostale pare tipov atributov se korelacijskega koeficienta ne da izračunati in to je v matriki predstavljeno s praznimi celicami. Take atribute sem prevzela za nekorelirane med seboj, čeprav seveda to ni nujno res. V tabeli (Slika 29) so v levem zgornjem kotu korelacijski koeficienti med numeričnimi atributi, ostali atributi (desni spodnji kot) so nominalni.

Slika 29: Primer korelacijske matrike



	A24_ZAC_MOD	A6_MOC_MOTORJA_KW	A37_NABAV_VRED	A204_STEVEC_LANI	B14_LETO_ROJ_ZAVN	C7_StSkodPred	C11_PovpTrajanjePog	C13_SkodNaPog	A63_PRAVNA	A94_NADSTD_TIPALO	A100_P_SERVIS	A101_JE_TECAJ	A106_JE_TAKSI	A111_DOPL_ZA_ODKUP_PRVE_SK_AO	A112_DOPL_ZA_ODKUP_PRVE_SK_AK	A113_JE_MLADI	A114_POP_NA_VOZ_IZKUSNJE_AO	A135_POP_NA_OMEJENO_ST_UPOR	A184_POP_NA_ZNAMKO_VOZ	A186_POP_NA_STAROST_ZAV	A192_POP_NA_ST_UPOR_VOZILA	A3_ZNAMKA	A4_TIP_VOZILA	A79_VR_VOZNIK	B11_spol_zavarovalec	B12_vrsta_zavarovalca	B15_spol_zavarovanec	B27_ali_premija_AO	B37_ali_premija_komb_H	B44_ali_premija_komb_Plus	C5_Ali_Skrajsana
A24_ZAC_MOD	1,00	-0,18	-0,28	0,46	-0,14	-0,02	-0,19	0,00																							
A6_MOC_MOTORJA_KW	-0,18	1,00	0,74	0,03	0,01	0,12	-0,03	0,00																							
A37_NABAV_VRED	-0,28	0,74	1,00	0,10	0,07	0,17	-0,01	0,01																							
A204_STEVEC_LANI	0,46	0,03	0,10	1,00	-0,15	0,01	-0,13																								
B14_LETO_ROJ_ZAVN	-0,14	0,01	0,07	-0,15	1,00	0,04	0,21	0,00																							
C7_StSkodPred	-0,02	0,12	0,17	0,01	0,04	1,00	0,04	0,02																							
C11_PovpTrajanjePog	-0,19	-0,03	-0,01	-0,13	0,21	0,04	1,00	0,00																							
C13_SkodNaPog	0,00	0,00	0,01		0,00	0,02	0,00	1,00																							
A63_PRAVNA									1,00	0,11	0,02	0,02	0,02	0,13	0,12	0,02	0,10	0,02	0,01	0,11	0,03	0,13	0,60	0,27	0,14	0,49	0,08	0,00	0,03	0,04	0,00
A94_NADSTD_TIPALO									0,11	1,00	0,20	0,09	0,10	0,21	0,19	0,11	0,11	0,26	0,39	0,18	0,22	0,34	0,02	0,33	0,06	0,16	0,06	0,00	0,16	0,17	0,00
A100_P_SERVIS									0,02	0,20	1,00	0,60	0,67	0,46	0,52	0,62	0,59	0,70	0,33	0,28	0,46	0,26	0,03	0,58	0,11	0,10	0,10	0,00	0,00	0,51	0,00
A101_JE_TECAJ									0,02	0,09	0,60	1,00	0,64	0,56	0,48	0,59	0,68	0,66	0,37	0,31	0,49	0,26	0,03	0,57	0,09	0,11	0,08	0,00	0,03	0,33	0,00
A106_JE_TAKSI									0,02	0,10	0,67	0,64	1,00	0,51	0,55	0,66	0,62	0,73	0,33	0,28	0,48	0,23	0,03	0,61	0,09	0,10	0,08	0,00	0,01	0,37	0,00
A111_DOPL_ZA_ODKUP_PRVE_SK_AO									0,13	0,21	0,46	0,56	0,51	1,00	0,57	0,48	0,55	0,53	0,30	0,25	0,39	0,09	0,01	0,54	0,07	0,02	0,05	0,00	0,09	0,27	0,00
A112_DOPL_ZA_ODKUP_PRVE_SK_AK									0,12	0,19	0,52	0,48	0,55	0,57	1,00	0,60	0,46	0,64	0,28	0,23	0,41	0,11	0,03	0,54	0,07	0,04	0,05	0,00	0,05	0,34	0,00
A113_JE_MLADI									0,02	0,11	0,62	0,59	0,66	0,48	0,60	1,00	0,61	0,75	0,34	0,31	0,50	0,22	0,03	0,57	0,10	0,09	0,08	0,00	0,01	0,39	0,00
A114_POP_NA_VOZ_IZKUSNJE_AO									0,10	0,11	0,59	0,68	0,62	0,55	0,46	0,61	1,00	0,67	0,37	0,54	0,50	0,26	0,03	0,55	0,10	0,14	0,08	0,00	0,03	0,34	0,00
A135_POP_NA_OMEJENO_ST_UPOR									0,02	0,26	0,70	0,66	0,73	0,53	0,64	0,75	0,67	1,00	0,67	0,40	0,95	0,25	0,02	0,64	0,11	0,10	0,09	0,00	0,03	0,50	0,00
A184_POP_NA_ZNAMKO_VOZ									0,01	0,39	0,33	0,37	0,33	0,30	0,28	0,34	0,37	0,67	1,00	0,37	0,53	0,36	0,02	0,32	0,06	0,05	0,05	0,00	0,04	0,33	0,00
A186_POP_NA_STAROST_ZAV									0,11	0,18	0,28	0,31	0,28	0,25	0,23	0,31	0,54	0,40	0,37	1,00	0,46	0,12	0,01	0,26	0,07	0,08	0,05	0,00	0,01	0,15	0,00
A192_POP_NA_ST_UPOR_VOZILA									0,03	0,22	0,46	0,49	0,48	0,39	0,41	0,50	0,50	0,95	0,53	0,46	1,00	0,18	0,02	0,45	0,07	0,08	0,07	0,00	0,04	0,48	0,00
A3_ZNAMKA									0,13	0,34	0,26	0,26	0,23	0,09	0,11	0,22	0,26	0,25	0,36	0,12	0,18	1,00	0,08	0,36	0,12	0,24	0,11	0,00	0,14	0,18	0,00
A4_TIP_VOZILA									0,60	0,02	0,03	0,03	0,03	0,01	0,03	0,03	0,03	0,02	0,02	0,01	0,02	0,08	1,00	0,04	0,02	0,01	0,02	0,00	0,01	0,01	0,00
A79_VR_VOZNIK									0,27	0,33	0,58	0,57	0,61	0,54	0,54	0,57	0,55	0,64	0,32	0,26	0,45	0,36	0,04	1,00	0,13	0,30	0,10	0,00	0,10	0,32	0,00
B11_spol_zavarovalec									0,14	0,06	0,11	0,09	0,09	0,07	0,07	0,10	0,10	0,11	0,06	0,07	0,07	0,12	0,02	0,13	1,00	0,20	0,91	0,00	0,05	0,08	0,00
B12_vrsta_zavarovalca									0,49	0,16	0,10	0,11	0,10	0,02	0,04	0,09	0,14	0,10	0,05	0,08	0,08	0,24	0,01	0,30	0,20	1,00	0,13	0,00	0,04	0,03	0,00
B15_spol_zavarovanec									0,08	0,06	0,10	0,08	0,08	0,05	0,05	0,08	0,08	0,09	0,05	0,05	0,07	0,11	0,02	0,10	0,91	0,13	1,00	0,00	0,05	0,06	0,00
B27_ali_premija_AO									0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
B37_ali_premija_komb_H									0,03	0,16	0,00	0,03	0,01	0,09	0,05	0,01	0,03	0,03	0,04	0,01	0,04	0,14	0,01	0,10	0,05	0,04	0,05	0,00	1,00	0,09	0,00
B44_ali_premija_komb_Plus									0,04	0,17	0,51	0,33	0,37	0,27	0,34	0,39	0,34	0,50	0,33	0,15	0,48	0,18	0,01	0,32	0,08	0,03	0,06	0,00	0,09	1,00	0,00
C5_Ali_Skrajsana									0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	1,00

Atribute sem odstranjevala, dokler nisem dobila množice atributov, pri kateri se je AUC prekomerno zmanjšala (AUC je bila manjša od osnovne AUC), kar pomeni, da sem odstranila preveč atributov in sem izgubila del informacije.

Odstranjevanje več atributov na enkrat v primerjavi z odstranjevanjem vsakega posebej prinaša tveganje. Zgodi se lahko, da kljub skrbni izbiri množice za odstranitev ponesreči odstranimo informacijo na podlagi kombinacije več atributov v množici. Zato sem postopku odstranjevanja atributov dodala še fazo dodajanja atributov, ki mi je take nesrečno odstranjene atribute dodala nazaj v optimalni izbor.

4.2.2.4 Dodajanje atributov

V četrtem koraku sem poskusila z dodajanjem atributov popraviti napake iz odvzemanja. Tudi ta poskus sem izvedla na omenjenih pet algoritmih. Izhajala sem iz nabora atributov, ki sem ga dobila v razdelku 4.2.2.3. Za vsak algoritem je bil to osnovni nabor.

Osnovnemu naboru sem dodala en atribut iz množice preostalih atributov (tistih, ki niso bili v osnovnem naboru) in izračunala AUC. V naslednji ponovitvi zanke, sem ta atribut odstranila iz osnovnega nabora ter dodala naslednji atribut iz preostale množice atributov in izračunala AUC. Ko sem uporabila vse atribute iz množice preostalih atributov, sem iz dobljenih rezultatov izbrala najboljši atribut, torej tistega, pri katerem je bila vrednost AUC največja. Npr. pri algoritmu Logistic sem začela s 14 atributi, ki sem jih dobila v postopku odstranjevanja

atributov. V naslednjem koraku sem osnovnemu naboru dodala en atribut ter postopek ponovila na naboru 15 atributov. Atribute sem dodajala, dokler nisem prišla do točke, ko dodajanje novih atributov osnovnemu naboru ni povečalo AUC. Pri algoritmu Logistic sem tako dodala 13 atributov in dobila končen nabor 27 atributov.

Analogno sem naredila tudi za ostale štiri algoritme, razlika je le v končnem naboru atributov. Rezultate predstavljam v naslednjem podpoglavju 4.2.3.

4.2.3 Rezultati

V tem podpoglavju bom predstavila rezultate pristopov odstranjevanja in dodajanja atributov.

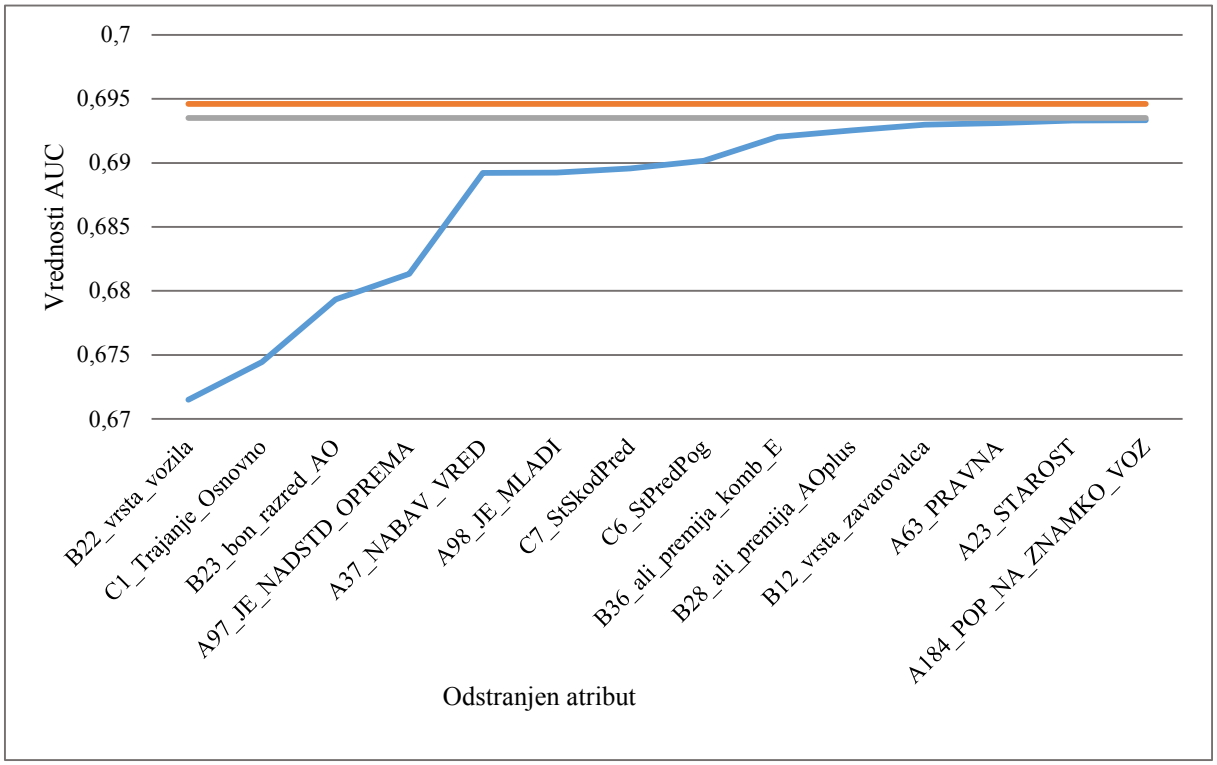
4.2.3.1 Rezultati odstranjevanja atributov

Rezultat postopka odstranjevanja atributov je nabor atributov, s katerimi dosežemo enako dober ali pa boljši rezultat AUC, kot če vzamemo vseh 119 atributov. Izkazalo se je, da se s tem postopkom ne samo izboljša natančnost AUC, ampak tudi preglednost nad množico atributov, ki dejansko vplivajo na rezultat. Za vsakega izmed petih algoritmov sem dobila svoj nabor atributov. V predstavitvi (Tabela 1) predstavljam končne nabore atributov. Opazimo, da je kar nekaj atributov takih, ki se pojavijo v več naborih, drugi spet samo v določenih. Tudi število atributov v naborih je različno. Tabelo bom podrobneje opisala v razdelku 4.2.3.2, tu pa bom navedla le, katere atribute so algoritmi izbrali za pomembne.

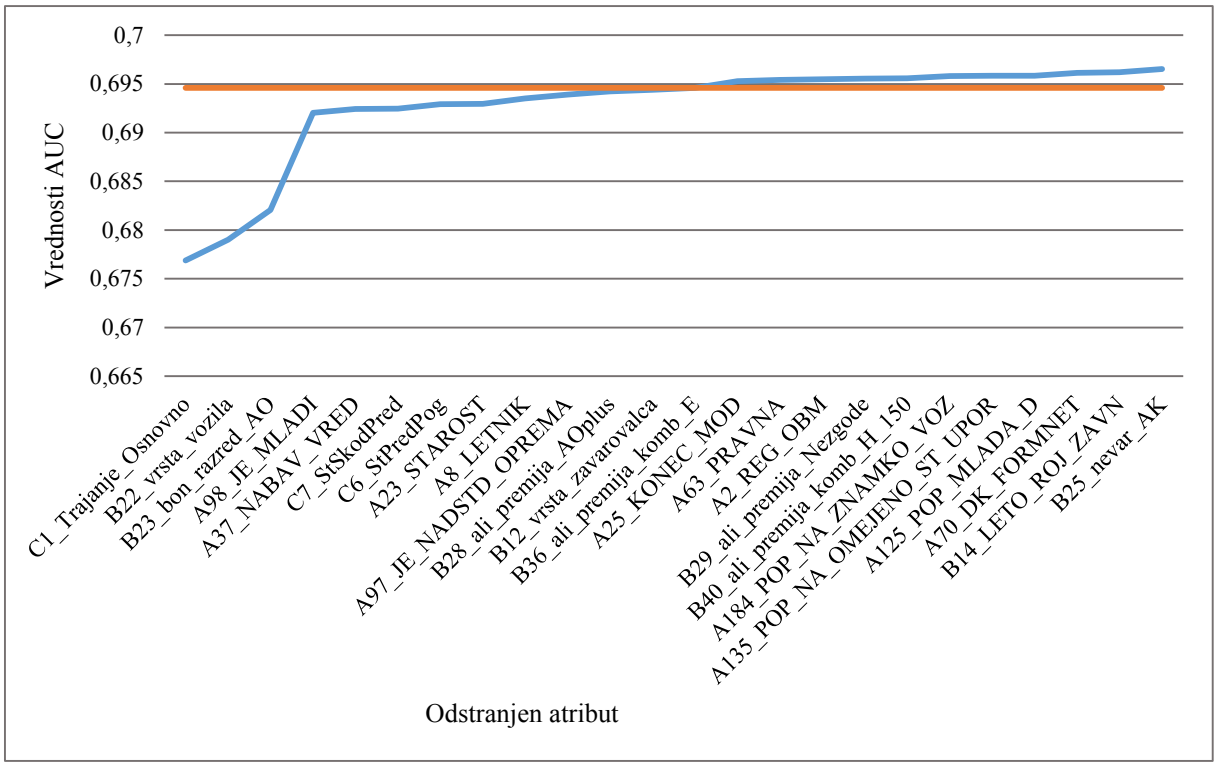
Za vseh pet algoritmov sta pomembna podatek o premijskem razredu AO (B23_bon_razred_AO), in pa podatek, ali je določen voznik mladi voznik na zavarovanju AO (A98_JE_MLADI). Rezultat ni presenetljiv, saj sta oba atributa tudi v trenutnem izračunu premije ključnega pomena. Za pomembne pa so bili izbrani tudi naslednji atributi: podatek o trajanju pogodbe (C1_Trajanje_Osnovno), podatek o tem, ali je na določeni pogodbi sklenjeno delno kasko zavarovanje kombinacije D (B35_ali_premija_komb_D), ki krije škodo, ki nastane na vozilu z neposrednim dotikom divjadi in domačih živali ter podatek o nadstandardni opremi (A97_JE_NADSTD_OPREMA). Ti podatki so bolj zanimivi, saj v sedanjem izračunu premije ne igrajo zelo pomembne vloge.

Na grafih (Slika 30, Slika 31) prikazujem primer vizualizacije, ki sem ga uporabljala pri odstranjevanju atributov. Na obeh vizualizacijah so na *x* osi prikazani atributi (v vsakem koraku sem odstranila en atribut), na *y* osi pa vrednost AUC po odstranitvi določenega atributa iz *x* osi (npr. če sem med 24 atributi odstranila atribut C1_Trajanje_Osnovno, je bila vrednost AUC enaka 0,676). Modra krivulja prikazuje vrednosti AUC, ko sem odstranila po enega izmed atributov, rdeča krivulja pa prikazuje vrednost AUC izračunano pri upoštevanju vseh 119 atributov (zato je konstantna ne glede na odstranjen atribut).

Slika 30: Vrednosti AUC pri vključenih 24-ih atributih



Slika 31: Vrednosti AUC pri vključenih 14-ih atributih



Prvi graf (Slika 30) prikazuje stanje, ko je bilo v izračun vključenih 24 atributov, drugi graf pa stanje, ko je bilo vključenih 14 atributov. Na drugem grafu (Slika 31) siva krivulja prikazuje vrednost AUC, če jo računamo na izbranih 14-ih atributih. V vmesnem koraku med prvim in

drugim grafom sem odstranila 10 atributov, kar na grafu pokaže, da sem odstranila preveč atributov (modra krivulja je v vseh točkah pod rdečo). Zato sem na tem koraku končala z odstranjevanjem in začela z dodajanjem atributov.

Enako analizo, kot sem jo opisala pri algoritmu Logistic, sem izvedla tudi za ostale štiri algoritme ter tako dobila končni nabor in število atributov. Pri Naključnih gozdovih sem končala z odstranjevanjem že pri 64 atributih, pri SMO pri 9 atributih, pri M5P pa sem postopek ponavljala, dokler nisem dobila 5 atributov. Naivni Bayes je bil malce poseben: očitno je bilo, da ga morebitne povezave in korelacije med atributi zelo motijo, saj je vrednost AUC presegla vrednost osnovne AUC že, ko sem odstranila prvih 31 atributov. Vrednost AUC je potem rastla, dokler nisem odstranila 105 atributov, kar pomeni, da sem z odstranjevanjem končala pri množici 14 atributov.

4.2.3.2 Rezultati dodajanja atributov

Rezultate dodajanja atributov prikazujem v predstavitvi (Tabela 1) skupaj z rezultati odstranjevanja atributov.

V tej tabeli vrstice predstavljajo atribute, ki sem jih dobila s postopkom odstranjevanja in dodajanja, stolpci pa algoritme, ki sem jih obravnavala. Pod imenom vsakega algoritma navajam še velikost končnega nabora atributov. Oranžne celice označujejo nabor atributov po končanem postopku odstranjevanja, modro obarvane celice pa označujejo tiste atribute, ki so bili dodani kasneje v fazi dodajanja atributov. Izjema v teh predstavitvi je algoritem Naključnih gozdov, ki je postopek zaključil z naborom 69 atributov. Ker bi bila končna tabela prevelika, sem pri tem algoritmu prikazala samo tiste atribute, ki so skupni tudi ostalim algoritmom. Pri ostalih algoritmih so bile velikosti končnih naborov naslednje: Logistic je vseboval 27 atributov, pri M5P sem zaključila pri 15 atributih, za SMO je bilo dovolj 19 atributov, za Naivnega Bayesa pa 21 atributov.

Analiza rezultatov pokaže, da je atribut, ki nosi podatek o nadstandardnem tipalu za dež (A94_NADSTD_TIPALO), ki sem ga ponesreči odstranila pri vseh algoritmih, pomemben, saj ga je v fazi dodajanja vključilo nazaj med pomembne atribute. Za pomembne so se izkazali tudi naslednji atributi: podatek o nadstandardni opremi (A97_JE_NADSTD_OPREMA), podatek o doplačilu za odkup prve škode na AK (A112_DOPL_ZA_ODKUP_PRVE_SK_AK), podatek o osnovnem trajanju pogodbe (C1_Trajanje_Osnovno), moč motorja vozila (A6_MOC_MOTORJA_KW), podatek o tem, ali je sklenjeno delno kasko zavarovanje kombinacije D (B35_ali_premija_komb_D) ter podatek o premijski skupini zavarovanja (B21_prem_skupina).

Na koncu za ilustracijo pomembnosti postopka prikazujem še gibanje vrednosti AUC med odstranjevanjem in dodajanjem atributov za posamezen algoritem. Na grafu (Slika 32) je za algoritem Logistic prikazano, kako se je spreminjala vrednost AUC v posameznih fazah. Na *x* osi prikazujem število atributov, ki so bili vključeni v izdelavo modela in izračun AUC (pri

Tabela 1: Končen nabor atributov

Atributi \ Algoritem	Naivni Bayes	Logistic	M5P	Naključni gozdovi	SMO
število atributov	21*	27*	15*	69*	19*
A98_JE_MLADI					
B23_bon_razred_AO					
A97_JE_NADSTD_OPREMA					
A6_MOC_MOTORJA_KW					
A94_NADSTD_TIPALO					
C1_Trajanje_Osnovno					
A112_DOPL_ZA_ODKUP_PRVE_SK_AK					
B35_ali_premija_komb_D					
B21_prem_skupina					
B28_ali_premija_AOplus					
A80_VR_POTNIK					
A63_PRAVNA					
B24_bon_razred_AK					
C7_StSkodPred					
C6_StPredPog					
C12_SkodNaLeto					
B22_vrsta_vozila					
B36_ali_premija_komb_E					
A2_REG_OBM					
A106_JE_TAKSI					
A24_ZAC_MOD					
A184_POP_NA_ZNAMKO_VOZ					
A37_NABAV_VRED					
A23_STAROST					
A8_LETNIK					
A16_NABAV_VRED_EUROTAX					
A204_STEVEC_LANI					
A210_VOZ_10					
A3_ZNAMKA					
B10_LETO_ROJ_ZAVL					
A113_JE_MLADI					
B12_vrsta_zavarovalca					
A125_POP_MLADA_D					
A41_ZV_OSEBE					
B40_ali_premija_komb_H_150					
B32_ali_premija_AK					
C2_Trajanje_DoDanes					
A30_ALARM					
A99_JE_HISA					
A114_POP_NA_VOZ_IZKUSNJE_AO					
A79_VR_VOZNIK					
B38_ali_premija_komb_H_0					
B43_ali_premija_krajaVandalizem_MINIKASKO					

Legenda:

atribut izbran v postopku odstranjevanja atributov

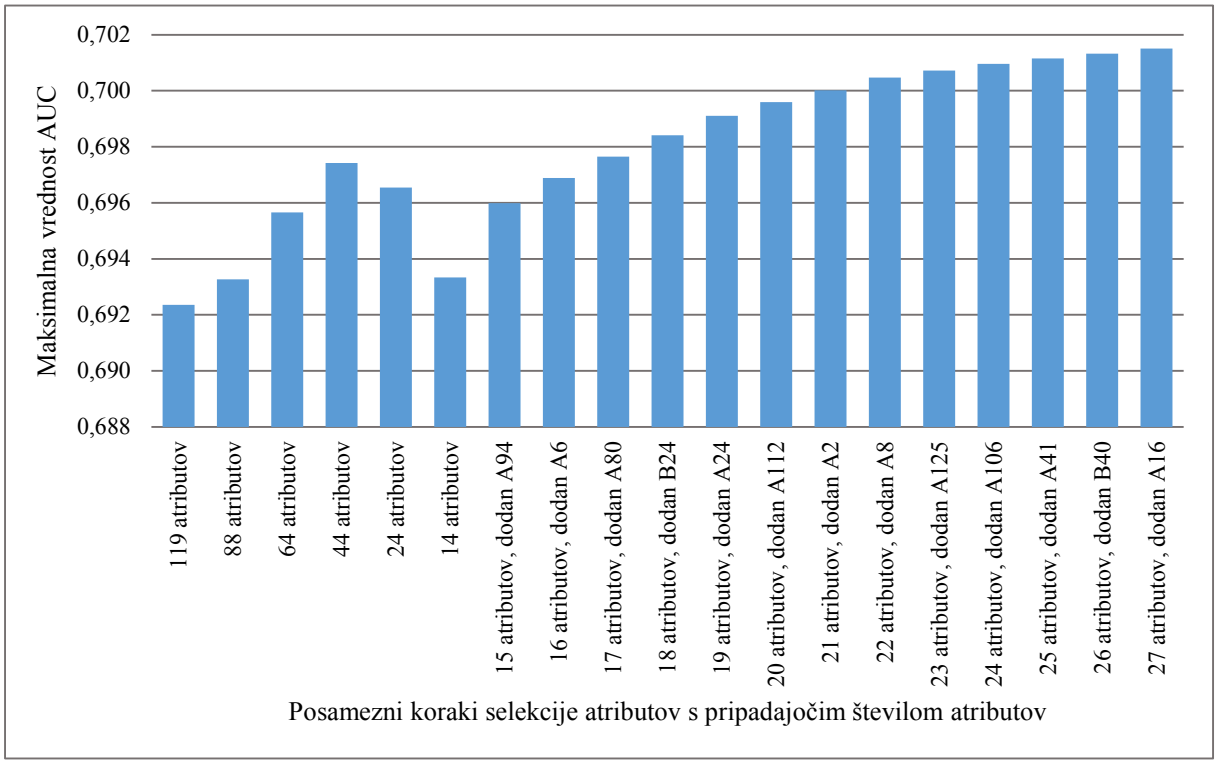
atribut izbran v postopku dodajanja atributov

* število atributov v končnem naboru

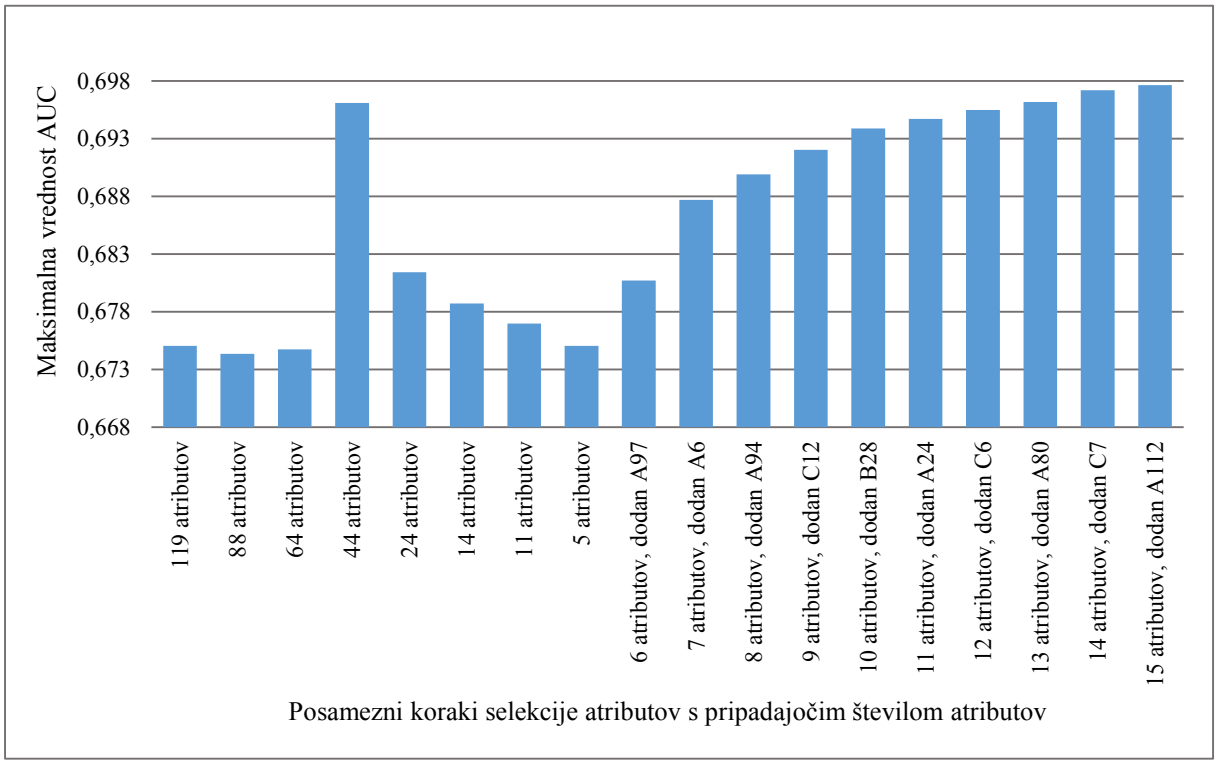
dodajanju atributov sem dodala še oznako dodanega atributa), na *y* osi pa je prikazana največja vrednost AUC, izračunana na določenem naboru atributov. Iz grafa je razvidno, kako je vrednost AUC do določene mere rastla, nato pa padala med odstranjevanjem atributov, z dodajanjem le-teh pa asimptotično rastla do neke vrednosti.

Analogne grafe prikazujem tudi za ostale algoritme (Slika 33, Slika 34, Slika 35, Slika 36). Vidimo, da je gibanje vrednosti AUC precej podobno kot pri algoritmu Logistic.

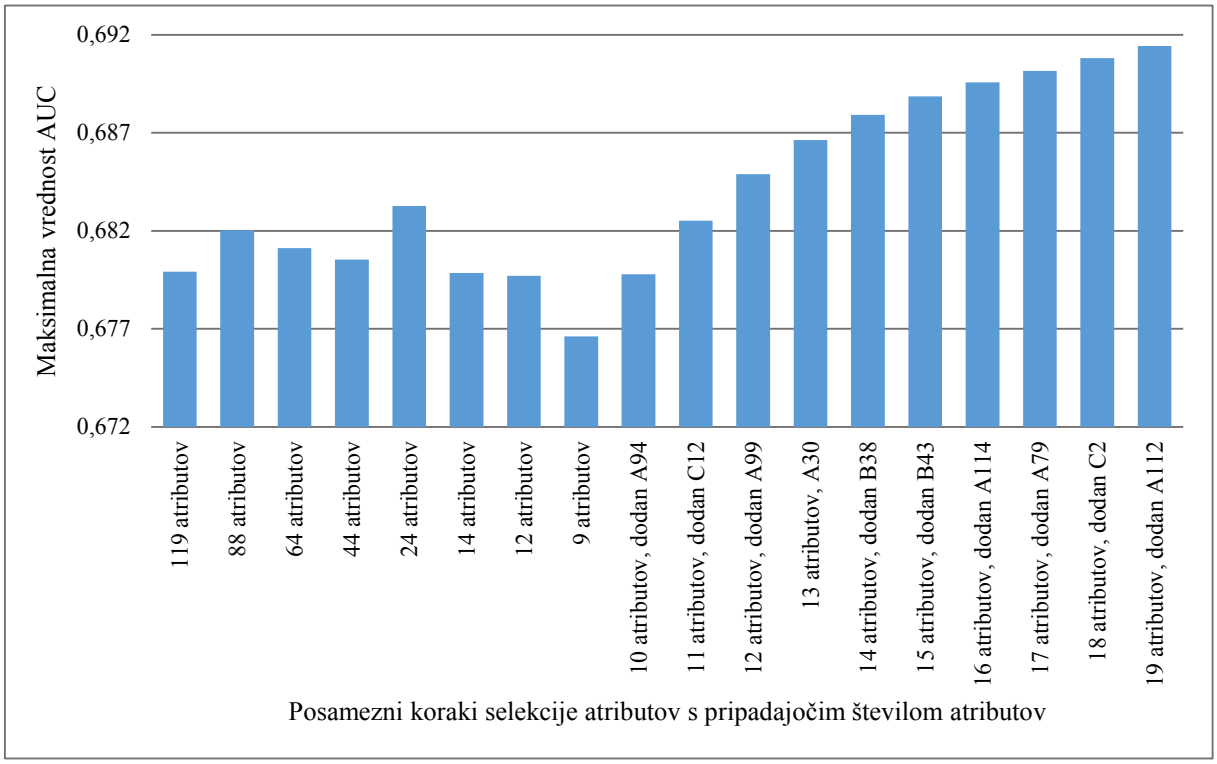
Slika 32: Gibanje AUC med selekcijo atributov pri algoritmu Logistic



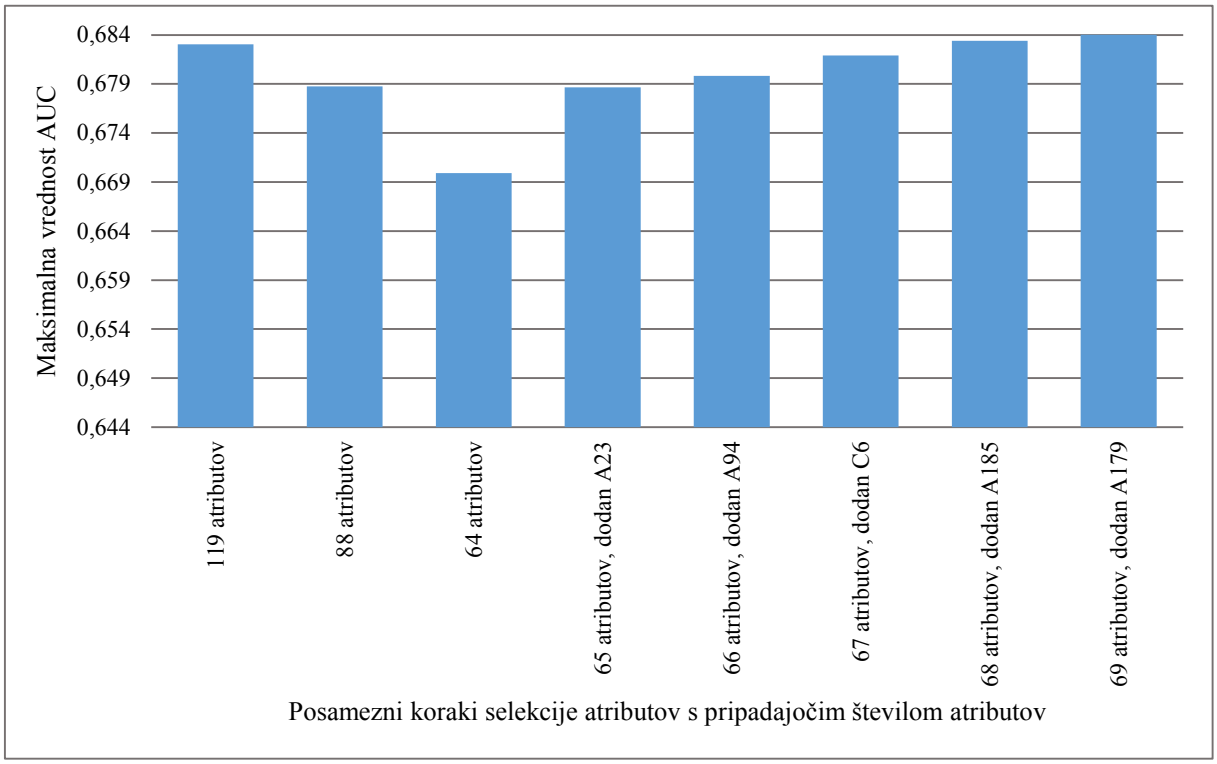
Slika 33: Gibanje AUC med selekcijo atributov pri algoritmu M5P



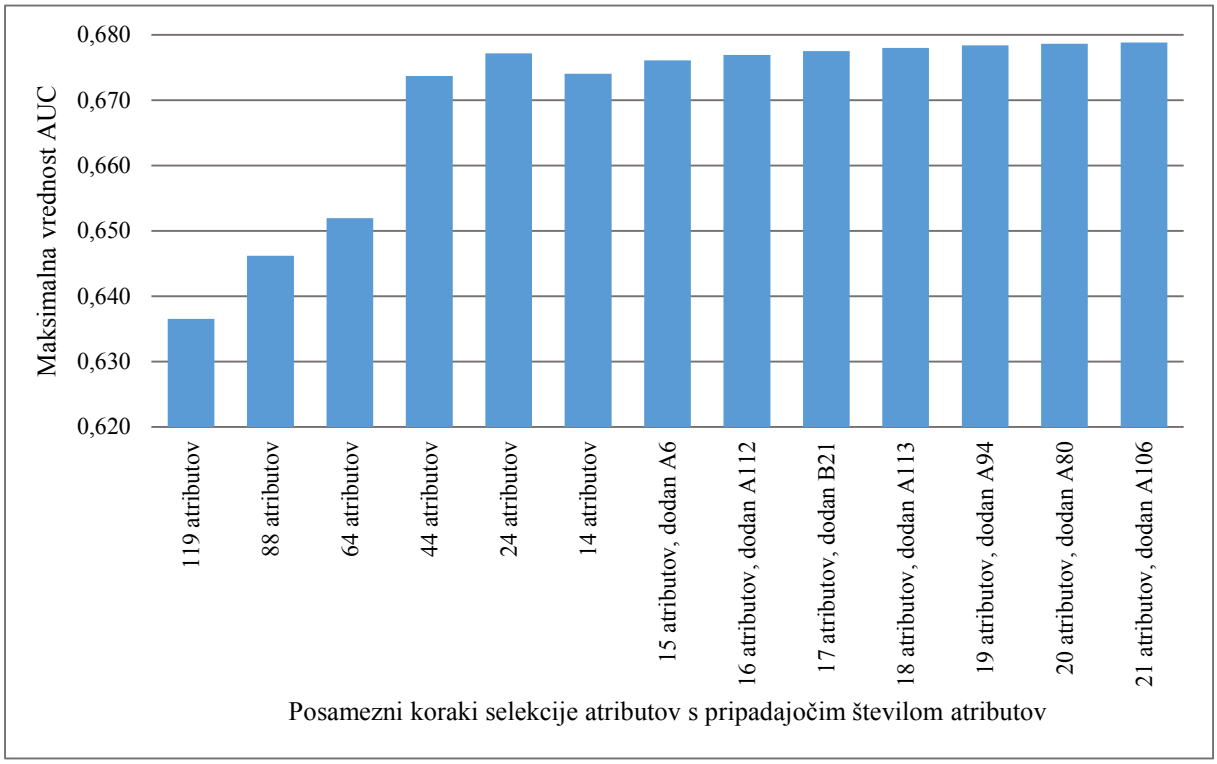
Slika 34: Gibanje AUC med selekcijo atributov pri algoritmu SMO



Slika 35: Gibanje AUC med selekcijo atributov pri algoritmu Naključni gozdovi



Slika 36: Gibanje AUC med selekcijo atributov pri algoritmu Naivni Bayes



Ob teh grafih je na mestu tudi krajša diskusija o nekaj ključnih zakonitostih podatkovnega rudarjenja. Za podatkovno rudarjenje namreč ni na voljo ultimativnih algoritmov, v katere bi lahko vstavili poljubno velik nabor atributov, le-ta pa bi nam brez naše pomoči izbral pomembne in izločil nesmiselne, ob tem pa dobil optimalno mero kakovosti (npr. AUC). Žal je med postopkom še vedno potrebna človeška interpretacija in večkratna ponovitev CRISP-DM cikla, ki pa je lahko zelo zamudno opravilo.

Nemoč algoritmov ločiti zrnje od plev (dobrih od slabih atributov) se vidi ravno iz prikazanih vizualizacij. Predstavitev za algoritem Logistic (Slika 32) nam npr. pokaže, da je bil optimalen AUC med odstranjevanjem dosežen pri 44 atributih. Torej, čeprav teoretično vseh 119 atributov zagotovo nosi več informacije kot samo 44, se je Logistic naučil boljšega modela pri 44 atributih. V želji še reducirati atribute sem sicer v prvi fazi s preveč agresivno odstranitvijo žrtvovala AUC - pri 14 atributih je bil AUC že razmeroma slab, kar se je zgodilo zaradi neoptimalne izbire atributov za odstranitev. Vendar vidimo, da je dodajanje atributov hitro popravilo manjše neoptimalnosti pri odstranjevanju. Tako sem pri končni množici 27 atributov dobila bistveno boljši rezultat kot kadarkoli med odstranjevanjem.

Kratek poduk te zgodbe, kot tudi celotnega magistrskega dela, je, da je potrebno pri metodah rudarjenja podatkov algoritmom pomagati z raznimi sredstvi, npr. s čiščenjem podatkov, selekcijo atributov, razumevanjem domene, kombiniranjem atributov v nove, bolj smiselne, pred-analizami zalog podatkov, izbiro parametrov, pametnim vzorčenjem, in podobnimi pristopi pokazanimi v tem delu. Metode rudarjenja podatkov in krovni CRISP-DM pristop nam

lahko služita zgolj kot vodilo, ki nakazuje različne aspekte analiz, ki jih moramo še obdelati. V nobenem primeru pa to ni avtonomni postopek, ki bo analize naredil namesto nas.

5 ANALIZA PRIMERNOSTI PODATKOVNEGA RUDARJENJA

Namen tega magistrskega dela je bil analizirati primernost uporabe metod podatkovnega rudarjenja pri problemu določanja tveganja zavarovalne pogodbe motornih vozil ter izdelati najprimernejši model na podlagi dejanskih podatkov iz baze kompozitne zavarovalnice, ki trži ta zavarovanja. V tem poglavju predstavljam rezultate v skladu z namenom dela. V prvem delu pokažem in primerjam dejanske modele, ki so rezultat podatkovnega rudarjenja. Ti rezultati se nanašajo na pet izbranih algoritmov, prikazujem pa njihovo ovrednotenje s krivuljo ROC ter vrednostmi AUC. Modele sem ocenila tudi na podlagi treh dopolnjujočih se kriterijev, ki se v podatkovnem rudarjenju pogosto upoštevajo: učinkovitosti, preglednosti in sposobnosti razlage. V drugem delu pokažem zelo pomembne stranske produkte podatkovnega rudarjenja ter podajam nekaj primerov uporabe modelov in ostalih rezultatov v zavarovalniškem poslovnem okolju, kar delno vključuje tudi zadnjo stopnjo CRISP-DM pristopa, tj. vključitev modela v poslovno okolje. Na koncu v diskusiji ob analizi napovedi tveganja zavarovalnih pogodb empirično pokažem, zakaj sem mnenja, da je podatkovno rudarjenje ob klasičnih aktuarskih postopkih zelo primerna komplementarna metoda za analizo podatkov.

5.1 Izbira ustreznega modela

V praksi je uspešnost uporabe modelov, ki jih zgradimo z metodami podatkovnega rudarjenja, odvisna od različnih kriterijev. Med bolj pogosto upoštevanimi so (Michalski, Bratko & Kubat, 1998, str. 394; Han & Kamber, 2001, str. 283) – pri vsakem kriteriju podajam tudi kratko razlago, kako sem ga upoštevala pri svojem delu:

- Natančnost (angl. *accuracy*): algoritem, na osnovi katerega je zgrajen nek model, mora iz učne množice podatkov pridobiti tiste informacije, ki mu omogočajo dobro napoved z visoko stopnjo natančnosti na novih primerih. V mojem primeru sem namesto klasične mere natančnosti uporabila ROC krivuljo oz. AUC statistiko, ki je tu primernejša.
- Hitrost (angl. *speed*) algoritma se nanaša predvsem na to, koliko časa in kakšni stroški nastanejo pri generiranju in uporabi modela. Hitrost sem direktno upoštevala, saj sem preko časa izvajanja izločila vse algoritme, ki so bili prepočasni.
- Razširljivost (angl. *scalability*) se nanaša na to, kako se učinkovito zgradi model v primeru velikih količin podatkov; v mojem primeru sem imela razmeroma veliko količino podatkov in tiste metode, ki niso zmogle obdelati take količine podatkov, so že v začetku izpadle iz analize.
- Zmogljivost oz. učinkovitost (angl. *performance*): je kriterij, ki ga določeni avtorji uporabljajo namesto natančnosti, včasih pa je definiran tudi kot kombinacija zgornjih treh kriterijev: natančnosti, hitrosti in tudi razširljivosti. Zmogljiv algoritem je tisti, ki se zna hitro naučiti na veliko podatkih. Zmogljivost sem ocenjevala s primerjavo algoritmov na

Pareto fronti (razdelek 4.1.3), saj je bil zame ta kriterij zelo pomemben za učinkovito odstranjevanje atributov.

- Obravnavanje manjkajočih vrednosti: različni algoritmi različno upoštevajo manjkajoče vrednosti; nekatere algoritme manjkajoče vrednosti ne motijo (npr. Naivni Bayes), spet drugi manjkajoče vrednosti nadomestijo s povprečno vrednostjo določenega atributa (npr. odločitveno drevo), nekateri jih obravnavajo kot novo vrednost atributa (npr. nekatera odločitvena pravila). Zmožnost algoritmov dobro obravnavati manjkajoče vrednosti je bila v mojem primeru zelo pomembna, saj je bilo manjkajočih vrednosti procentualno zelo veliko in tega ročno nisem popravljala, saj to ni bilo mogoče.
- Obravnavanje podatkov z napakami oz. šumom: napake v podatkih so zelo pogoste v bazah, kjer je pri vnosu podatkov v bazo prisoten človeški faktor, zato morajo algoritmi ustrezno obravnavati tudi ta vidik. V mojem primeru sem do določene stopnje izvedla čiščenje podatkov, vendar so kljub temu določene napake še ostale.
- Robustnost: včasih se kot kriterij upošteva tudi kombinacijo dveh zgoraj opisanih sposobnosti algoritma, da hkrati obravnava manjkajoče podatke kot tudi šumne.
- Preglednost oz. transparentnost modelov: novo znanje in rezultati, ki ga vrnejo modeli, morajo biti pregledni, tako da jih lahko uporabniki analizirajo in razumejo, poleg tega pa naj bi modeli vrnili neke nove zakonitosti oz. nov pogled na dani problem, ki je bil predhodno neznan.
- Sposobnost razlage (angl. *explanation ability*): modeli naj bi znali razložiti odločitev, ko se obravnava nov primer – to je v praksi prednost transparentnih algoritmov, netransparentne algoritme se uporabi le, če slednji bistveno prekašajo transparentne v učinkovitosti.
- Zmanjšanje števila atributov in primerov: zbiranje podatkov je lahko zelo dolgotrajen in zamuden postopek, zato so boljši tisti algoritmi, ki že z majhnim številom atributov dajo dobre rezultate.

V nadaljevanju bolj podrobno predstavljam vidike natančnosti, preglednosti in sposobnosti razlage. Natančnost je po navadi najpomembnejši kriterij, saj pravzaprav pove, kako dober je model, ostala dva kriterija pa sta zanimiva predvsem iz vidika, kako dobro bo model sprejet pri končnem uporabniku, za katerega je po navadi pomembno, da je model čim bolj razumljiv in pregleden.

5.1.1 Natančnost modelov

Mera za natančnost določenega algoritma je bila krivulja ROC ter površina pod njo, AUC. Na prikazu (Tabela 2) podajam vrednosti AUC za izbranih pet algoritmov pri končni izbiri atributov in najoptimalnejših parametrih. Iste podatke prikažem tudi v obliki ROC krivulj na vizualizaciji (Slika 37). Na prvi pogled izstopajo tri opažanja: 1. lepe in konsistentne oblike ROC krivulj z zelo dobro napovedno točnostjo na začetku in koncu krivulje, 2. neprepričljive razlike med različnimi algoritmi v ROC in AUC prostoru ter 3. navidezno slabi AUC rezultati. Na vsa tri opažanja bom odgovorila spodaj ob analizi začetka in konca ROC krivulj (najbolj in

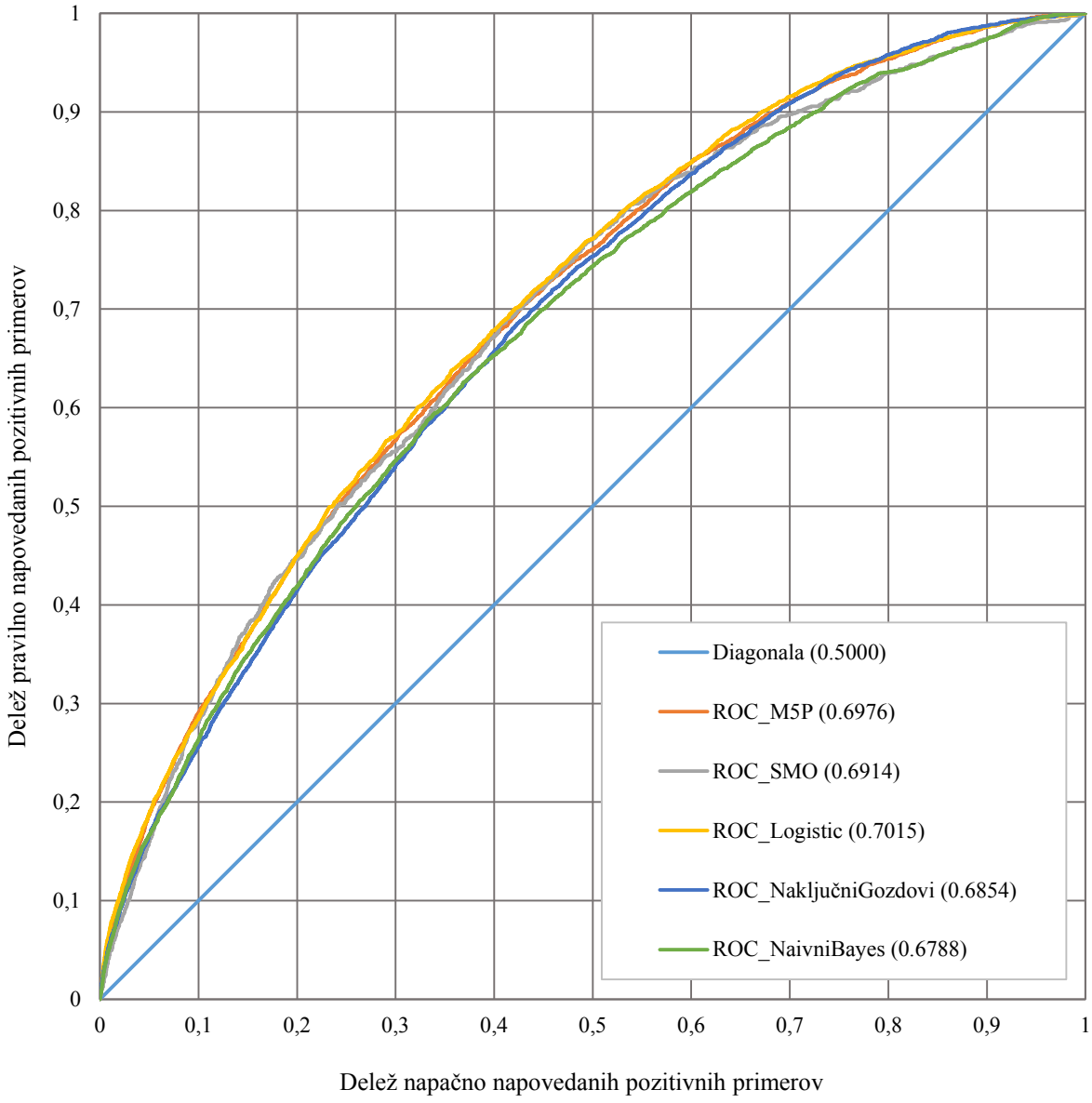
najmanj tveganih pogodb). Opazimo tudi, da se je najbolje izkazal algoritem Logistic. Sledi mu modelno drevo M5P, algoritem podpornih vektorjev SMO ter Naključni gozdovi.

Tabela 2: Primerjava vrednosti AUC

Algoritem	Naivni Bayes	Logistic	M5P	Naključni gozdovi	SMO
AUC (vsi atributi)	0,6348	0,6946	0,6730	0,6760	0,6787
AUC (najpomembnejši atributi)	0,6788	0,7015	0,6976	0,6854	0,6914

Kot sem že omenila, sem algoritem, ki je implementacija metode Naivnega Bayesa, uporabila za primerjavo, saj ta metoda velikokrat služi kot osnova za merilo kompleksnosti določenega problema in soodvisnosti atributov. Kljub temu, da se je problem, ki sem ga reševala, na začetku zdel precej kompleksen (v naboru podatkov je bilo prek 280 atributov), se je izkazalo, da je

Slika 37: Krivulje ROC izbranih algoritmov



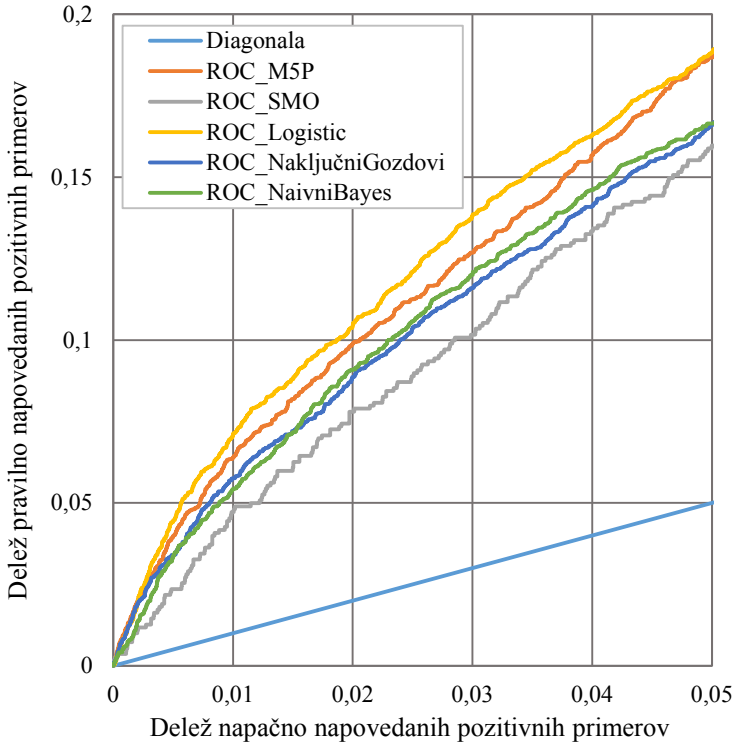
obvladljiv. Naivni Bayes je za učinkovito učenje potreboval le 21 atributov, njegova uspešnost pa je primerljiva z ostalimi štirimi precej bolj kompleksnimi metodami in tudi njegova oblika krivulje zelo dobro sledi ostalim štirim.

Vendar pa statistika AUC velikokrat ni najprimernejša mera za merjenje učinkovitosti. Krivulja ROC oz. AUC je sicer najbolj znana mera za primerjavo, ko so podatki neuravnoteženi, vendar z njo merimo le povprečno učinkovitost na celotnem naboru podatkov. Če je naš cilj bolj splošen, je smiselno opazovati AUC ali krivuljo ROC kot celoto. Če je pa naš cilj bolj specifičen, kot v mojem primeru najti najboljšo metodo za ugotavljanje najslabših pogodb, je smiselno podrobno preveriti učinkovitost metode na začetku krivulje.

Na vizualizaciji (Slika 38) je prikazan začetek krivulje ROC. Analiza natančnosti ROC deluje po principu, da pogodbe razvrsti od najbolj tveganih do najmanj tveganih. Na začetku krivulje se torej nahajajo najbolj tvegane pogodbe. Opazimo, da so algoritmi dobro izluščili znanje o najbolj tveganih pogodbah, saj se vseh pet krivulj na začetku hitro dvigne - veliko hitreje kot diagonala. Če pogledamo krivuljo algoritma M5P, vidimo, da gre krivulja skozi točko (0,02; 0,1), kar pomeni, da je v tej množici pogodb kar 10 % vseh pogodb s škodo in le 2 % vseh pogodb brez škode. Torej je gostota škod v tem delu grafa petkrat večja, kot bi bila, če bi imeli naključni model, tj. diagonala, ki gre skozi točko (0,02; 0,02). Model algoritma Logistic se je tu odrezal najbolje, najslabše pa škodo napoveduje model na osnovi algoritma SMO.

Če se od zgoraj omenjene točke pri x = 2 % po osi *x* premikamo k še manjšim številom (k napovedano še bolj tveganim pogodbam) vidimo, da je relativna razlika med naključnim in

Slika 38: Spodnji levi rob ROC

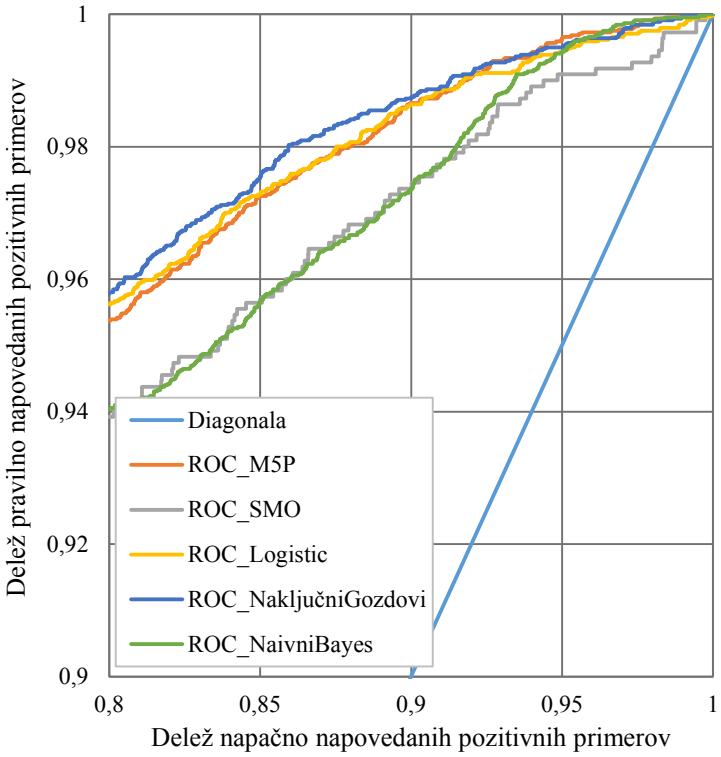


mojimi modeli še večja, kar pomeni da znajo moji modeli res dobro napovedati verjetnost škode za najbolj tvegane pogodbe. Npr. pri. točki x = 5 ‰ model Logistic napoveduje skoraj 10-krat bolje od naključja. Iz perspektive uporabe modela lahko interpretiram to dejstvo kot: v kolikor mi model napove za neko novo pogodbo, da se nahaja približno v zgornjih 5 ‰ tveganih pogodb, potem lahko rečem, da ima ta pogodba 10-krat večjo verjetnost škode, kot jo ima naključna pogodba. Ta interpretacija je tudi ključna za razumevanje primarnega načina uporabe modelov v praksi.

Podobno lahko sklepamo tudi za pogodbe, ki predstavljajo zelo majhno tveganje za zavarovalnico. V tem primeru moramo analizirati krivuljo, ko se približuje točki (1,1), kar je prikazano na naslednji vizualizaciji (Slika 39). V tem primeru je najboljši model Naključnih gozdov, saj se krivulja najbolj približa premici y = 1, vendar tudi modela na osnovi algoritmov Logistic in M5P ne zaostajata veliko. Podobno kot interpretacija zgoraj lahko rečem tu na podlagi točke (0.95; 0,995): če mi algoritem napove pogodbo s tveganjem med najvarnejšimi 5 % pogodb, potem vem, da ima taka pogodba približno 10-krat manjšo verjetnost škode od povprečne pogodbe.

Zanimiva točka opazovanja je tudi primerjava uspešnosti modelov z obeh strani hkrati. Opažam namreč, da se zmorejo modeli bistveno bolje naučiti, katere so lastnosti dobrih pogodb, kot pa katere so lastnosti slabih. To je vidno že iz zgornjih primerov: 10-kratno povečanje/zmanjšanje verjetnosti škode lahko trdimo na podlagi uvrstitve v 5 % najvarnejših pogodb a zgolj 5 ‰ najbolj tveganih pogodb (kar je veliko bolj omejujoče). Če pogledamo še malo širši pas, lahko trdimo, da je pogodba, ki je napovedana med 20 % najvarnejših pogodb, v povprečju kar

Slika 39: Zgornji desni rob ROC



4,4-krat bolj varna od naključne pogodbe – točka Logistic (0.8; 0.955). Po drugi strani pa je pogodba, ki je napovedana med 20 % najbolj tveganih pogodb, v povprečju "zgolj" 2,2-krat bolj tvegana od naključne pogodbe – točka Logistic (0,2; 0,44).

Iz zgornjega odstavka sklepam, da je precej lažje napovedovati varne pogodbe kot napovedovati tvegane in temu sledijo tudi napovedne točnosti modelov v različnih delih prostora pogodb. Iz vsebinske perspektive je razlog morda v tem, da so škode/nesreče zelo naključni dogodki, ki se jih ne da zelo dobro predvideti in včasih zadenejo tudi varne voznike/pogodbe. Na drugi strani pa odsotnost škode na pogodbi pomeni neko varnost, kar pa ni ravno naključje, ampak bolj lastnost pogodbe in se jo da očitno na podlagi atributov precej dobro predvideti.

Iz zgornje analize je razvidno, da model na osnovi algoritma Logistic malce prednjači pred ostalimi modeli – predvsem zaradi opazno boljših rezultatov na področju najbolj tveganih pogodb, ki so tudi fokus te magistrske naloge.

5.1.2 Preglednost modelov in sposobnost razlage

Vidika preglednosti modelov in sposobnosti razlage odločitev napovedi prikazujem skupaj, saj sta precej povezana.

Z vidika preglednosti so modeli na osnovi algoritmov Logistic, M5P in Naivnega Bayesa pregledni, SMO in Naključni gozdovi pa nepregledna. Vendar pa lahko tudi pregledni modeli kaj kmalu postanejo nepregledni, sploh če je v kalkulacije vključenih veliko število atributov. To se je zgodilo tudi v mojem primeru, saj sem začela s precej večjih številom atributov, kot jih je bilo dejansko potrebnih za uspešno izvedbo. S spoznavanjem baze podatkov, s postopkom čiščenja atributov ter z izborom atributov pa so se modeli poenostavili in postali dejansko bolj pregledni ter tudi učinkovitejši - iz predstavitve (Tabela 2) je razvidno povečanje vrednosti AUC za posamezen algoritem.

Logistic, Naivni Bayes in M5P so predstavljeni z določenimi linearnimi kombinaciji atributov (v nadaljevanju prilagam tudi slike modelov). Take predstavitve naj bi bile tudi za končnega uporabnika sprejemljive, saj se da iz njih videti, kateri atributi imajo večji vpliv na razred.

Po drugi strani pa je model na osnovi algoritma SMO tipičen predstavnik netransparentnih modelov, saj je predstavitev s podpornimi vektorji že na prvi pogled dokaj kompleksna in tudi podrobnejši pregled pokaže, da jo je precej težko oz. skoraj nemogoče analizirati.

Tudi model na osnovi algoritma Naključnih gozdov je primer netransparentnega modela. Kot že ime pove, je procedura v ozadju tega algoritma precej naključna: uporabi se določeno število odločitvenih dreves (v mojem primeru sem jih izbrala 40), ta pa izbirajo svoje atribute. To je bilo opazno že pri izboru pomembnih atributov, saj je ta algoritem za uspešno izvedbo potreboval precej večje število atributov v primerjavi z ostalimi štirimi. Tudi analiza končnega izbora atributov kaže na to naključnost: ostali štirje algoritmi so težili k temu, da so izbirali čim

bolj nekorelirane atribute, medtem ko je algoritem Naključnih gozdov izbral tudi take atribute, ki so imeli korelacijski koeficient blizu 1. To je posledica tega, da je eno odločitveno drevo, ki deluje v ozadju tega algoritma, izbralo enega izmed koreliranih atributov, drugo drevo pa drugega izmed koreliranih atributov.

Pri modelu na osnovi algoritma Logistic, pri katerem sem dobila najboljše rezultate, KNIME vrne koeficiente β (glej podpoglavje 2.2.5) ter hkrati redundančno tudi vrednosti e^{β} za pogodbe s škodo oz. pozitivni razred (Slika 40). Če vrednost atributa x_{i1} povečamo za eno enoto, potem se rezultat v enačbi (10) poveča za faktor e^{β_1} (glej razdelek 2.2.5). Negativne vrednosti β pomenijo negativno razmerje med verjetnostjo škode na pogodbi in atributom x; če so vrednosti β pozitivne, pomenijo pozitivno razmerje.

Rezultat modela na osnovi algoritma Logistic, ki sem ga dobila v programu KNIME, namreč nima tako lepe predstavitve, zato sem ga zgenerirala v programu Microsoft Excel 2010 (Slika 40). Prikazujem le del rezultata, saj je celotna tabela predolga za predstavitev. V prvem stolpcu

Slika 40: Predstavitev modela za napovedovanje škod na osnovi algoritma Logistic

Spremenljivke	Koeficienti za razred pogodb s škodami	Grafični prikaz koeficientov
A106_JE_TAKSI	-1,4489	
A112_DOPL_ZA_ODKUP_PRVE_SK_AK	-0,5074	
A125_POP_MLADA_D	0,1730	
A184_POP_NA_ZNAMKO_VOZ	0,3141	
B23_bon_razred_AO=1	-0,4680	
B23_bon_razred_AO=2	-0,3815	
B23_bon_razred_AO=3	-0,1134	
B23_bon_razred_AO=4	-0,1825	
B23_bon_razred_AO=5	0,0023	
B23_bon_razred_AO=6	-0,2183	
B23_bon_razred_AO=7	-0,1805	
B23_bon_razred_AO=8	-0,0192	
B23_bon_razred_AO=9	0,1675	
B23_bon_razred_AO=10	0,0378	
B23_bon_razred_AO=11	0,1223	
B23_bon_razred_AO=12	0,3174	
B23_bon_razred_AO=13	0,3449	
B23_bon_razred_AO=14	0,4331	
B23_bon_razred_AO=15	-0,2587	
B23_bon_razred_AO=16	-0,1089	
B23_bon_razred_AO=17	0,2760	
B23_bon_razred_AO=18	1,1050	
B23_bon_razred_AO=20	0,7137	
B22_vrsta_vozila=Motornokolo	-0,9008	
B22_vrsta_vozila=Traktorjivlailci	-0,7707	
B22_vrsta_vozila=Specialnomotornovozilo	0,0745	
B22_vrsta_vozila=Avtobus	0,4055	
B22_vrsta_vozila=Osebniavtomobil	0,4066	
B22_vrsta_vozila=Tovornovozilo	0,6688	

so navedene spremenljivke oz. atributi, v drugem pa koeficienti β za razred pogodb, na katerih so bile izplačane škode. V rezultatu, ki ga vrne KNIME, so na analogen način prikazani tudi koeficienti e^{β}. Dodala sem še tretji stolpec, ki vsebuje obarvano skalo, ki grafično ponazarja vpliv koeficientov β na končni rezultat; to nam daje boljši pregled, kateri koeficienti povečujejo verjetnost škode in kateri zmanjšujejo. Pozitivni koeficienti atributov povečajo napoved škode, negativni pa to napoved zmanjšujejo. Primer: iz rezultata je razvidno, da višji bonitetni razred AO povečuje verjetnost škode na AO (z nekaj izjemami), nižji razred pa verjetnost zmanjšuje.

Model Logistic nam torej vrne rezultate v taki obliki, da jih znamo razložiti in interpretirati, kar je z vidika končnega uporabnika zelo zaželena lastnost. Model je tudi pregleden, saj se da videti tudi to, kateri atributi imajo večji oz. manjši vpliv na razred.

Pri ostalih modelih prikazujem le izpise iz orodja KNIME. Model na osnovi M5P je porezano modelno drevo, kjer za vsak razred dobimo eno pravilo, ki je pravzaprav linearna kombinacija določenih atributov (Slika 41).

Slika 41: Izpis modela M5P

```
Classification via Regression

Classifier for class with index 0:

M5 pruned model tree:
(using smoothed linear models)
LM1 (11285/93.329%)

LM num: 1
B64_ali_skoda_AO =
        0.1018 * A112_DOPL_ZA_ODKUP_PRVE_SK_AK=da
        + 0.0051 * A24_ZAC_MOD
        + 0.001 * A6_MOC_MOTORJA_KW
        + 0.0643 * A80_VR_POTNIK= Uiteljivonjemedpoukom,
        Delavcinakamionuvkljunosspremljevalci, Potnikivtaksijihinrent-a-carvozilih,
        Potnikivavtobusihtrolejbusihvlakih, Sprevodnikiavtobusovtrolejbusovvlakov
        + 0.1206 * A94_NADSTD_TIPALO=da
        + 0.1323 * A97_JE_NADSTD_OPREMA=ne
        + 0.1547 * A98_JE_MLADI=da
        + 0.128 * B21_prem_skupina=6,4,1,5,2,3,10
        + 0.2025 * B21_prem_skupina=1,5,2,3,10
        - 0.0663 * B21_prem_skupina=5,2,3,10
        + 0.1411 * B21_prem_skupina=2,3,10
        + 0.0635 * B23_bon_razred_AO=4,2,6,7,11,8,9,15,10,14,3,5,12,13,16,17,20,19,18
        - 0.0344 * B23_bon_razred_AO=2,6,7,11,8,9,15,10,14,3,5,12,13,16,17,20,19,18
        + 0.0324 * B23_bon_razred_AO=6,7,11,8,9,15,10,14,3,5,12,13,16,17,20,19,18
        + 0.0921 * B23_bon_razred_AO=11,8,9,15,10,14,3,5,12,13,16,17,20,19,18
        - 0.0527 * B23_bon_razred_AO=8,9,15,10,14,3,5,12,13,16,17,20,19,18
        + 0.0331 * B23_bon_razred_AO=9,15,10,14,3,5,12,13,16,17,20,19,18
        + 0.0824 * B23_bon_razred_AO=14,3,5,12,13,16,17,20,19,18
        - 0.0889 * B23_bon_razred_AO=3,5,12,13,16,17,20,19,18
        + 0.0698 * B23_bon_razred_AO=12,13,16,17,20,19,18
        + 0.0513 * B28_ali_premija_AOplus=da
        + 0.0406 * B35_ali_premija_komb_D=ne
        + 0.3595 * C12_SkodNaLeto
        + 0.3315 * C1_Trajanje_Osnovno
        - 0.0013 * C6_StPredPog
        + 0.0145 * C7_StSkodPred
        - 0.7303
```

Pri Naivnem Bayesu se za vsak kategoričen atribut prešteje število pojavitev tega atributa v vsakem razredu, za numerične atribute pa se izračuna povprečna vrednost, standardna deviacija,

utežena vsota ter natančnost (Slika 42). V rezultat, ki je spet linearna kombinacija, vsi atributi prispevajo enako.

Slika 42: Izpis modela Naivnega Bayesa

```
Naive Bayes Classifier

                                                  Class
Attribute                                            da          ne
                                                  (0.03)      (0.97)
===================================================================
A106_JE_TAKSI
  da                                                11.0        55.0
  ne                                              1765.0     57518.0
  [total]                                         1776.0     57573.0

A112_DOPL_ZA_ODKUP_PRVE_SK_AK
  da                                               135.0      2428.0
  ne                                              1962.0     61975.0
  [total]                                         2097.0     64403.0

A113_JE_MLADI
  da                                               137.0      2329.0
  ne                                              1507.0     50626.0
  [total]                                         1644.0     52955.0

A204_STEVEC_LANI
  mean                                       110121.1304 108232.5053
  std. dev.                                   72341.3338  90908.1797
  weight sum                                         113        4366
  precision                                    1634.1021   1634.1021

A210_VOZ_10
  da                                               735.0     29097.0
  ne                                               458.0     11764.0
  [total]                                         1193.0     40861.0

A2_REG_OBM
  CD                                                 2.0        12.0
  CE                                               224.0      7193.0
  GO                                               190.0      6241.0
  KK                                               195.0      9566.0
  KP                                               181.0      5299.0
  KR                                               293.0     10970.0
  LJ                                              1320.0     42372.0
  MB                                               194.0      5373.0
  MS                                                44.0      1535.0
  NM                                              1008.0     49596.0
  PO                                                23.0       700.0
  SG                                                41.0       886.0
  [total]                                         3715.0    139743.0
```

Model na osnovi SMO prikazuje podporne vektorje. Že preglednost modela je slaba, interpretacija rezultata pa praktično nemogoča (Slika 43).

Za model na osnovi Naključnih dreves pa KNIME ne vrne nobene predstavitve, zato tudi tega modela tu ne prikazujem. Kot sem že omenila, je to primer netransparentnega modela in četudi bi imela podano predstavitev, bi bila razlaga rezultatov zaradi upoštevanja velikega števila dreves praktično nemogoča.

Slika 43: Izpis modela SMO

```
SMO

Kernel used:  RBF kernel: K(x,y) = e^-(0.01* <x-y,x-y>^2)

Classifier for classes: da, ne

BinarySMO

- 8* <11 1 0.021191 0 1 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1
0 1 1 0 0.552198 0.552198> * X]
 - 8* <10 1 0.042381 0 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
1 0 1 1 0 1 1> * X]
 - 8* <10 1 0.04054 0 1 0 0 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1
1 1 1 0.117952 0.997252 0.997252> * X]
 + 8* <10 1 0.019677 0 1 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
1 1 1 1 0 0.997252 0.997252> * X]
 + 8* <11 0 0.040363 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 1 1 0 0.994506 0.994506> * X]
 + 8* <10 1 0.04054 0 1 0 0 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1
1 1 1 0.317193 1 1> * X]
 - 8* <10 1 0.032795 0 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 1 0.042024 0.997252 0.997252> * X]
 - 8* <10 1 0.042381 0 1 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 0 0.997252 0.997252> * X]
 + 8* <10 1 0.042381 0 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
1 0 1 1 0 0.994506 0.994506> * X]
 + 8* <10 1 0.043895 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 1 1 1 0.042024 0.362638 0.362638> * X]
 - 8* <10 1 0.03885 0 1 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
1 1 1 0.042024 1 1> * X]
 + 8* <10 1 0.019677 0 1 0 0 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
1 1 1 1 0 0.997252 0.997252> * X]
 - 8* <10 1 0.037841 0 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1
1 1 1 1 0 0.997252 0.997252> * X]
 + 8* <10 0 0.038345 0 1 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 1 1 1 0.057299 0.994506 0.994506> * X]
 - 8* <10 1 0.025227 0 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
1 1 1 1 0.269176 0.994506 0.994506> * X]
 - 8* <10 0 0.032795 0 1 0 0 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 1 1 1 0.042024 0.997252 0.997252> * X]
 + 8* <10 1 0.032795 0 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 1 1 1 0 0.936813 0.936813> * X]
 + 8* <10 1 0.038345 0 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1
1 1 1 1 0 0.997252 0.997252> * X]
 + 8* <10 0 0.029263 0 1 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 1 1 0 0.997252 0.997252> * X]
 + 8* <00 0 0.023713 0 1 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 1 1 1 0.042024 0.994506 0.994506> * X]
 - 8* <10 1 0.140767 1 0 0 0 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
1 1 1 1 0.042024 0.994506 0.994506> * X]
 - 8* <10 1 0.042381 0 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
1 1 1 1 0.295671 1 1> * X]
 - 8* <10 1 0.028759 0 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 0.042024 0.997252 0.997252> * X]
```

5.2 Rezultati in uporaba podatkovnega rudarjenja

Problem, ki ga rešujem v tem magistrskem delu, je napovedovanje tveganja zavarovalne pogodbe AO. Modeli, ki sem jih zgradila s podatkovnim rudarjenjem, napovedujejo verjetnost izplačila škode na pogodbi AO. Eksperimenti so bili zastavljeni tako, da imajo algoritmi za učenje na voljo zgolj podatke, ki so bili prisotni ob sklenitvi pogodbe, napovedujejo pa izplačilo škode v celotnem trajanju pogodbe. Tak način simulacije ustreza realnemu scenariju uporabe modela, ko bi želeli oceniti tveganje nove pogodbe tik pred njeno sklenitvijo (in na podlagi rezultatov npr. dovoliti dodaten popust pri premiji ali pa ne).

Krivulja ROC, s katero merim učinkovitost določenega modela, predstavlja spodnjo mejo natančnosti modelov in predvidevam, da bi bila učinkovitost ocene tveganja v realnejših pogojih še bistveno boljše ocenjena. Razlog za to je dejstvo, da sem škodno dogajanje spremljala na pogodbi zgolj v njenem trajanju (kar ni več kot eno leto). Ker je v tako kratkem obdobju verjetnost škode majhna (četudi je pogodba morebiti zelo tvegana), so mi modeli vračali veliko napačno ocenjenih pogodb – pogodbe so bile ocenjene kot tvegane, vendar jih je ROC analiza ocenila kot napako, ker se na njih škoda dejansko ni zgodila. Zelo težko je oceniti, koliko je bilo sicer pravilno označenih tveganih pogodb, ki pa so imele v času trajanja pač srečo in se jim škoda ni zgodila. Domnevam, da je delež takih pogodb velik in da bi v bistveno bolj kompleksni večletni analizi spremljanja vozil ter zavarovalcev dobili lepšo sliko, kako dobri so modeli v resnici – glede na intuicijo, razvito med obravnavanjem teh podatkov, ocenjujem, da so modeli v resnici bistveno boljši kot kažejo trenutne statistike.

Analiza krivulj ROC tudi pokaže, da so algoritmi dobro izluščili znanje o najboljših in najslabših pogodbah, saj se krivulje na začetku hitro dvignejo, na koncu pa se tudi precej dobro približajo premici y = 1. Kot rečeno zgoraj, pogodbe, ki so razvrščene na začetku, resnično predstavljajo večje tveganje za zavarovalnico, vendar pa to ni zagotovilo, da bodo v naslednjem letu imele škodo. Skratka diskusija v tem in prejšnjem odstavku se pravzaprav dotika problema ne najboljšega ocenjevanja kakovosti modelov za tveganje preko ROC analize. Tveganje sem za potrebe analize kar enačila z izplačilom škode. To je seveda najboljše, kar sem ob danih podatkih lahko naredila (tveganja nimam na voljo), ni pa optimalno, saj škoda ne predstavlja direktno tveganja. Poleg tega so nesreče oz. izplačila škod redek pojav, kar še zaplete statistično enačenje s tveganjem.

Pri interpretaciji se je pomembno zavedati tudi, da napoved modela ne predstavlja direktno verjetnosti, da bo določena pogodba imela škodo, ampak le razvršča pogodbe tako, da na začetku razvrsti tiste pogodbe, ki predstavljajo večje tveganje za zavarovalnico, na koncu pa tiste, ki so manj tvegane. Poleg razvrstitve dobim tudi neko oceno o tveganju. Če bi tako razvrščene pogodbe prilagodila dejanski verjetnostni krivulji, bi dobila neko grobo oceno verjetnosti tudi za vsako pogodbo posebej. Trenutno pa lahko govorim le o razmerju med zavarovalnimi pogodbami, o tem, katera pogodba je bolj tvegana glede na ostale.

5.2.1 Stranski produkti podatkovnega rudarjenja

Kot sem v tem delu že večkrat poudarila, je proces rudarjenja podatkov zapleteno in zamudno opravilo, ki pa ima ravno zaradi tega poleg samega modela znanja tudi več stranskih rezultatov. Preden se proces podatkovnega rudarjenja sploh prične, je pomembno določiti cilje tega procesa. Vendar pa že vsaka faza tega procesa pridela nek rezultat, tako da na koncu dobimo nekakšno zaporedje stranskih rezultatov, ki skupaj tvorijo končni rezultat (Reinartz, 1999, str. 20). Pogosto se zgodi, da so na koncu dejansko bolj uporabljeni stranski produkti, glavni rezultat pa se včasih celo zavrže.

Stranski rezultati analize tega magistrskega dela so naslednji:

- Ureditev in čiščenje podatkovne baze zavarovalnice: to je bil zelo zahteven proces, saj so podatki vsebovali precej napak in anomalij, ki jih je bilo potrebno odpraviti oz. smiselno dopolniti. Veliko nepravilnosti sem odpravila že na začetku procesa, torej še preden sem podatke uvozila v KNIME, vendar pa se je v fazi grajenja modelov pokazalo še nekaj napak, ki sem jih spregledala, zato je bilo potrebno ta postopek čiščenja ponoviti. Rezultat je prečiščena in urejena baza podatkov, kar je zelo pomemben rezultat, saj so smiselni podatki jedro vsake analize v zavarovalništvu.
- Spoznavanje z domeno in podatki: to je bil krovni proces med izdelavo tega dela. Objektivno žal ne morem ovrednotiti pridobljenega znanja in novega vpogleda v podatke. Je pa dejstvo, da so se mi preko analiz odpirala nova spoznanja o podatkih, kot primer: za zelo pomembno se je izkazala korelacijska analiza, saj je medsebojna odvisnost oz. neodvisnost atributov ključno vplivala na končni rezultat.
- Sistematizacija obstoječih in odkrivanje novih atributov: za sistematizacijo atributov je poleg medsebojne odvisnosti atributov pomembna tudi odvisnost posameznega atributa od razreda. Prav tako lahko na atribute pogledamo iz perspektive razreda tveganja (subjektivni se nanašajo na osebo, objektivni pa na vozilo, glej podpoglavje 1.3). Na podlagi takih atributno osredotočenih analiz lahko s sodelovanjem različnih oddelkov v zavarovalnici dosežemo, da se ob sklepanju novih zavarovanj zbira nove atribute, za katere pričakujemo veliko korelacijo s tveganjem. Ko dobimo dovolj podatkov za novi atribut, lahko z dodatno analizo ovržemo ali potrdimo pomembnost takega atributa in ga v slednjem primeru tudi dodamo v izračun premije.
- Izbira vzorčenja: rezultati pri določanju primernega vzorčenja so že sami po sebi zanimiv rezultat, saj nam pokažejo zakonitosti naših podatkov. Vzorčenje nam pokaže, kakšno je minimalno število primerov (zavarovalnih pogodb), ki jih potrebujemo za spodobno analizo. Tako lahko v primeru novega produkta približno ocenimo, kdaj bomo sposobni (to pomeni, da bomo imeli dovolj primerov) izdelati analizo s pomočjo podatkovnega rudarjenja.
- Odkrivanje najrazličnejših zanimivih zakonitosti v domeni in podatkih, npr.:
    - Noben algoritem ni izbral spol zavarovalca za pomemben atribut.
    - Tudi starost zavarovalca ali leto opravljenega vozniškega izpita ne vplivata na tveganje voznika. Verjetno se vozniške izkušnje upoštevajo pri mladem vozniku ter pri bonitetnem razredu (ta dva atributa je izbralo vseh pet algoritmov), atributi, ki označujejo starost direktno pa niso nikoli izbrani.
    - Kar zadeva lastnosti vozila pa so iz izbora pomembnih atributov izpadli atributi, ki označujejo najrazličnejše mase ter nosilnosti vozil.
    - Dejansko število atributov, ki vplivajo na tveganje je majhno (okoli 20).

5.2.2 Uporaba modelov v praksi

Modeli oz. posledično tudi metode podatkovnega rudarjenja imajo zelo široko področje uporabe v najrazličnejših vejah zavarovalništva. Predno pa nakažem možne načine uporabe

modelov, navajam še načine, kako pravilno izbrati kriterij za končno izbiro modela (kateri izmed algoritmov je najboljši) glede na njegovo uporabo:

- Če uporaba modela ni vnaprej znana, je primerna splošna metrika za ocenitev kakovosti modela AUC. Modele se razvrsti glede na AUC: večji AUC pomeni bolj natančen model (v splošnem). Če pa hočemo pridobiti še večjo natančnost modela, je smiselno pregledati še vzorčenje, ki lahko bistveno izboljša rezultat. Pri tem lahko izberemo modele, ki potrebujejo več časa in posledično dajo tudi boljše rezultate (glej Pareto fronto v podpoglavju Rezultati 4.1.3).
- Če je osnovni namen zgrajenih modelov odkrivanje najboljših oz. najslabših pogodb, ki predstavljajo visoko tveganje za zavarovalnico, potem kakovost modela težko ocenimo samo z vrednostjo AUC. Tu je smiselno pregledati tudi krivuljo ROC (večji kot je lok krivulje, bolj natančen je model) oz. njen začetni del (npr. prvih 10%): če se krivulja hitro dvigne, torej ima naklon precej večji od 1, potem ta krivulja oz. model dobro napoveduje slabe zavarovalce. Analogno lahko iščemo varne oz. dobre zavarovalce, le da v tem primeru analiziramo končen del krivulje, ki pa mora imeti naklon blizu 0.

Modeli, ki sem jih zgradila, so namenjeni predvsem odkrivanju tveganih pogodb oz. posledično tudi zavarovalcev na zavarovanju AO, z določenimi prilagoditvami pa se jih lahko uporabi tudi za druge namene. Nekaj splošnih primerov uporabe sem navedla že v uvodu tega dela, sedaj pa navajam konkretnejše primere, za katere se dotični zgrajeni modeli lahko uporabijo:

- Najbolj direkten način uporabe je seveda prilagoditev premije: visoko tveganim zavarovalcem je potrebo zaračunati višjo premijo, saj obstaja večja verjetnost za nastanek škode. Pri tem je potrebno določiti atribute oz. dejavnike tveganja ter jih povezati s pričakovano škodo. Iz zgrajenih modelov je trivialno razbrati te dejavnike, saj so rezultat faze izbora najpomembnejših atributov.
- V povezavi z napovedovanjem škod je možno modele prilagoditi tako, da se z njimi lahko napoveduje ne samo pogostost škod ampak tudi velikost škod. Tak problem se rešuje z metodami regresije, ker je ciljna spremenljivka numerična in ne kategorična. Kljub temu pa se lahko uporabi enak pristop za odkrivanje dejavnikov tveganja ter morebitnih novih ne intuitivnih povezav med njimi.
- Če modele uporabljamo za napovedovanje velikosti škod, posledično lahko odkrijemo morebitne nove trende v škodah. S tem se izboljša proces napovedovanja škod, bolj natančno se lahko oceni škodna rezerva.
- Modeli se lahko uporabijo tudi v postopkih, ki jih zavarovalnica izvaja pred prevzemom tveganja oz. nevarnosti v zavarovanje ter posledično v procesu upravljanja s tveganji. Bistvo teh postopkov je, da zavarovalnica, še preden sprejme neko nevarnost v zavarovanje, naredi ustrezno segmentacijo nevarnosti (loči dobre in slabe nevarnosti) ter ponudi ustrezno premijo. To je ključnega pomena, če želi zavarovalnica povečati dobiček. Analiza atributov pa bi lahko posledično tudi pohitrila te postopke sprejemanja v zavarovanje, saj se že ob sklenitvi zavarovalne pogodbe lahko pregleda samo tiste

dejavnike tveganja, ki resnično vplivajo na rezultat, ostale pa se izpusti. Na eni strani imamo tako zadovoljno stranko, ki ji ni potrebno izpolnjevati nekih dolgih formularjev, po drugi strani pa je cena zavarovanja ustreznejša. Posledično lahko na ta način pridobimo nove manj tvegane stranke, saj te plačajo nižjo premijo, hkrati pa nismo privlačni za tvegane pogodbe oz. zavarovalce, saj je zanje premija višja.

- Modeli se lahko direktno uporabijo tudi za napoved škode oz. napoved tveganih zavarovalcev pri zavarovanju AK in posledično tudi pri ostalih avtomobilskih zavarovanjih, le ciljno spremenljivko oz. razred je potrebno zamenjati.
- Modeli se lahko uporabijo tudi v drugih vrstah zavarovanj, predvsem na premoženjskih zavarovanjih, ki imajo podobne lastnosti oz. atribute kot avtomobilska (predvsem ožja premoženjska in odgovornostna zavarovanja). Z določenimi modifikacijami pa bi modele lahko uporabili tudi na področju osebnih zavarovanj: kot izhodišče bi lahko uporabili podobne modele, kot sem jih zgradila (potrebna bi bila seveda zamenjava ustreznih atributov in razreda, vendar pa bi lahko uporabili že preverjene algoritme in njihove optimalne parametre), v naslednjih ciklih pa bi se zelo verjetno tudi algoritmi prilagodili.

Modeli, ki sem jih dobila s podatkovnim rudarjenjem, imajo precej širok nabor uporabe in so lahko v pomoč pri različnih odločitvenih problemih, pri določanju cene zavarovanja, pri grajenju modelov tveganja, pridobivanju novih strank … Bolj kot številski rezultat tega dela je pomembno to, da je možno metode podatkovnega rudarjenja uporabiti tudi na področju zavarovalništva kot nek nov pristop, ki pa dejansko prinese tudi novo uporabno znanje in komplementarnost obstoječim metodam na tem področju.

5.3 Primerjava napovedi tveganja z diskusijo

Predno bom podala zaključno misel tega dela, bom predstavila še zanimivo analitično primerjavo obstoječih metod v zavarovalništvu z metodami, ki sem jih obravnavala v tem delu. Ta primerjava mi nudi dodaten dokaz smiselnosti uporabe sicer zamudne, a vendar hitrejše analize podatkov z metodami rudarjenja podatkov v primerjavi z klasičnimi aktuarskimi metodami.

Na tem mestu bom naredila dve predpostavki, za kateri vem, da nista čisto pravilni, sta pa ob prilagoditvi zelo blizu resnici: 1. višina premije na pogodbi je zelo dobro korelirana s stopnjo tveganja po zavarovalni pogodbi in 2. v izračun višine premije so vključeni vsi znani parametri pridobljeni iz klasičnih aktuarskih metod analize tveganj.

Prva predpostavka ni čisto pravilna, saj višina premije poleg stopnje tveganja vključuje tudi pričakovano vrednost škode. Četudi je tveganje majhno in je pričakovana višina škode velika, se lahko oboje sešteje v visoko premijo; tak primer so zavarovanja tovornjakov. Za odstranitev tega problema bom v tej primerjavi vzela zgolj zelo homogeno skupino klasičnih avtomobilskih zavarovanj (ostale tipe vozil sem izključila). Tu naj bi bila pričakovana višina škode neodvisna, zato bi morala premija v prvi fazi odražat stopnjo tveganja pogodbe. Dodaten zaplet, ki ga ne

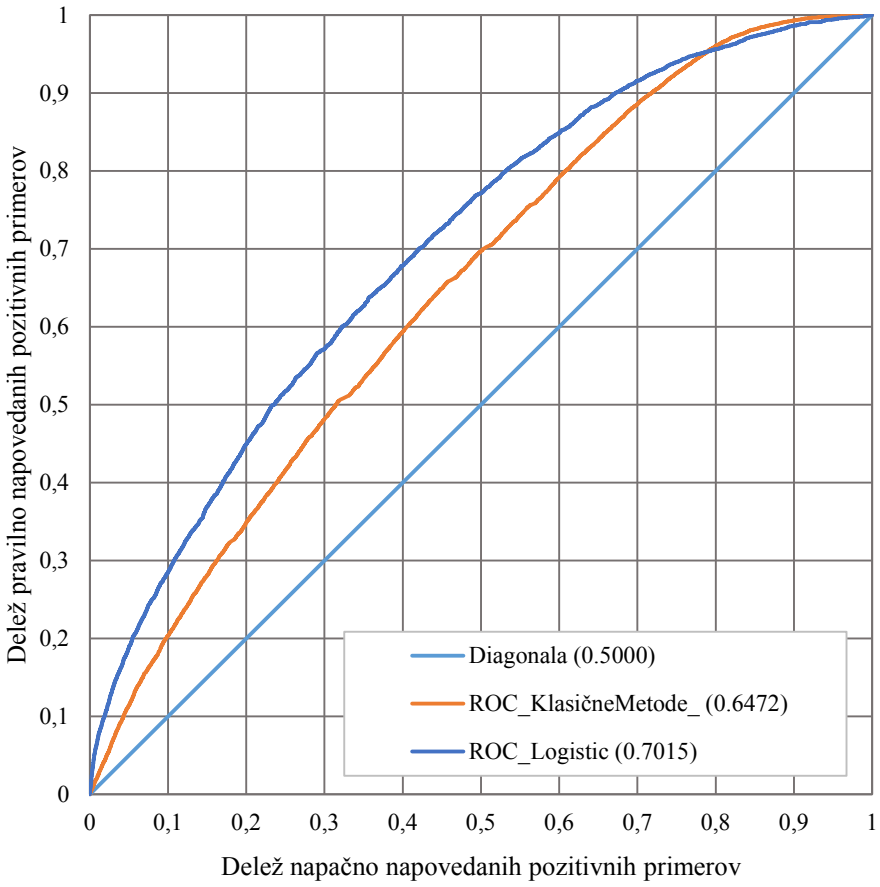
bom upoštevala, vendar se lahko pojavi, je tudi tržna naravnanost premij, torej izdelava cenikov proti zakonitostim tveganja, zgolj zato, da se optimizira trženje produkta.

Druga predpostavka prav tako ni popolnoma resnična, saj se ceniki skozi čas izboljšujejo, dodajajo se novi parametri in enačbe se spreminjajo. Kljub temu mislim, da bi morala višina premije v določenem časovnem trenutku precej dobro odražati informacije oz. znanje, ki ga strokovnjaki o tej domeni imajo.

Kljub nepopolnosti predpostavk menim, da je analiza smiselna, saj od premij oz. strokovnjakov, ki so jih sestavljali, ne zahteva, da dobro odražajo stopnjo tveganja. Pomembno je zgolj to, da je premija vsaj približno dobro korelirana s tveganjem. ROC analiza je v svoji osnovi zelo tolerantna do ne prevelikih napak pri oceni tveganja (v tem primeru višine premije). Zato rezultati kar dobro odražajo dejansko stanje, čeprav so predpostavke precej približne.

Na vizualizaciji (Slika 44) prikazujem primerjavo klasičnih aktuarskih metod z metodami rudarjenja podatkov pri uspešnosti učenja zakonitosti o stopnji tveganja zavarovalne pogodbe. Iz stališča rudarjenja podatkov sem uporabila metodo Logistic, ki je bila v tem delu odkrita kot najoptimalnejša za ta problem. Prav tako sem uporabila optimalni nabor atributov ter najboljšo množico parametrov. S stališča klasičnih metod sem uporabila za napovedovanje tveganja kar

Slika 44: Primerjava klasičnih aktuarskih metod z metodami rudarjenja podatkov



višino premije, saj ta po predpostavki dobro odraža znanje, ki so ga strokovnjaki s klasičnimi metodami pridobili iz podatkov.

Analiza (Slika 44) nam kaže zelo zanimivo sliko ob danih predpostavkah. Algoritem Logistic je v povprečju uspel iz podatkov izluščiti več informacij (AUC = 70,15 %), kot je to uspelo strokovnjakom s klasičnimi analizami (AUC = 64,72 %). Podrobna analiza pa mi pove še to, da je v začetku ROC krivulje (torej med zelo tveganimi pogodbami) model na osnovi algoritma Logistic pri učenju zakonitosti tveganja nekajkrat boljši od klasičnega aktuarskega modela. Vendar pa ta prednost s pomikanjem v desno proti bolj varnim pogodbam hitro kopni in je v osrednjem delu pogodb zgolj še zmerna (od okrog 10 % do 20 % bolj natančna napoved). Presenečenje pa je, da znajo klasične metode veliko bolje izluščiti pravila, ki veljajo za najbolj varne pogodbe in so v tem nekajkrat bolj uspešne kot algoritem Logistic.

Rezultat predstavljene analize v tem podpoglavju je, da so metode rudarjenja podatkov ter klasične analize komplementarni pristopi, ki bi ob hkratni uporabi zagotovo dali še bistveno boljše rezultate kot vsak posebej.

SKLEP

V magistrskem delu sem se sistematično poglobila v analizo primernosti uporabe metod podatkovnega rudarjenja pri problemu tveganja zavarovalne pogodbe s področja zavarovanj motornih vozil.

Na osnovi analiz ter rezultatov, ki sem jih dobila v procesu podatkovnega rudarjenja, ocenjujem, da so modeli, ki sem jih razvila, v splošnem primerni za uporabo v zavarovalništvu, vendar pa so od primera do primera bolj ali manj uspešni (različne metode, različni načini vzorčenja, izbor pomembnih atributov). Ponovno pa poudarjam, da so rezultati podatkovnega rudarjenja pogosto večplastni in lahko prinesejo celo več koristi preko stranskih rezultatov kot preko končnih modelov. Velja še več: modele, ki sem jih izdelala, se da z določenimi modifikacijami uporabiti tudi za druge v uvodu predstavljene zavarovalniške probleme.

Konkretna uporaba metod rudarjenja podatkov je pokazala, da so algoritmi dobro izluščili znanje o najboljših in najslabših pogodbah. Pri tem se je najbolj izkazal model algoritma Logistic (z vidika več kriterijev), čeprav tudi ostali štirje, ki sem jih vključila v nadaljnjo analizo, niso veliko zaostajali. Kljub temu, da pri moji zasnovi poskusov napovedi modelov ne predstavljajo direktno verjetnosti škode, pa lahko potrdim, da dobro predstavljajo oceno o tem, katere pogodbe predstavljajo večje tveganje za zavarovalnico. Nadalje bi lahko te razvrščene pogodbe prilagodila dejanski verjetnostni krivulji. Tako bi dobila grobo oceno verjetnosti škode za vsako pogodbo posebej (in to pred njeno sklenitvijo), kar bi še olajšalo razlago rezultatov modelov za potencialne končne uporabnike.

Ob tem sklepu sprejemanja rudarjenja podatkov kot zanimivega dodatka k obstoječim metodam pa moram povzeti še nekaj kritičnih opazk, do katerih sem prišla med izdelavo tega dela.

Podatkovno rudarjenje namreč ne vsebuje "vsemogočnih" algoritmov, ki bi popolnoma samostojno izluščili znanje iz podatkov brez naše pomoči. Med postopkom CRISP-DM cikla je še vedno potrebna človeška interpretacija in večkratna ponovitev, kar je lahko precej časovno zahtevno. Tako je potrebno metodam rudarjenja podatkov smiselno pomagati, na primer s čiščenjem podatkov, selekcijo atributov, razumevanjem domene, sestavljanjem novih boljših atributov, pred-analizami zalog vrednosti, izbiro parametrov, pametnim vzorčenjem in podobnimi pristopi, od katerih sem ključne predstavila tudi v tem delu. Dejstvo pa je tudi, da je velika večina teh pred-obdelav potrebna tudi pri uveljavljenih aktuarskih analizah, zato to ni nujno dodatno delo pri podatkovnem rudarjenju.

Rudarjenje podatkov je torej orodje oz. avtomatiziran proces, uporabnik, ki dobro pozna področje, pa ga mora uporabljati v skladu s svojim strokovnim znanjem. V nasprotnem primeru rezultati nimajo zagotovljene kakovosti. Z empirično analizo sem tudi pokazala, da so metode rudarjenja podatkov ter klasične analize komplementarna pristopa, ki bi ob hkratni uporabi morebiti dala še bistveno boljše rezultate kot vsak posebej.

Na osnovi pridobljenih rezultatov lahko potrdim, da so metode podatkovnega rudarjenja primerne za modeliranje tveganja pri zavarovanju motornih vozil, kar je bil tudi namen tega magistrskega dela.

LITERATURA IN VIRI

1. Aha, D. W., Kibler, D. & Albert, M. K. (1991). Instance-based learning algorithms. Najdeno 28. aprila 2014 na spletnem naslovu http://sci2s.ugr.es/pr/pdf/1991-Aha-ML.pdf
2. Anderson, D., Feldblum, S., Modlin, C., Schirmacher, D., Schirmacher, E. & Thandi, N. (2007). A Practitioner's Guide to Generalized Linear Models. Najdeno 20. februarja 2014 na spletnem naslovu https://www.casact.org/pubs/dpp/dpp04/04dpp1.pdf
3. Apte, C., Grossman, E., Pednault, E., Rosen, B., Tipu, F. & White, B. (1998). Insurance Risk Modeling Using Data Mining Technology. Najdeno 20. februarja 2014 na spletnem naslovu http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.71.1766&rep=rep1&type=pdf
4. Berry, M. J. A. & Linoff, G. S. (2000). *Mastering Data Mining: The Art and Science of Customer Relationship Management*. New York: John Wiley & Sons, Inc.
5. Ben-Gal, I. (2007). Bayesian Networks. Najdeno 28. aprila 2014 na spletnem naslovu http://www.eng.tau.ac.il/~bengal/BN.pdf
6. Bouckaert, R. R. (2008). Bayesian Network Classifiers in Weka for Version 3-5-7. Najdeno 15. marca 2014 na spletnem naslovu http://www.cs.waikato.ac.nz/~remco/weka.bn.pdf
7. Breiman, L. (2001). Random Forests. Najdeno 15. marca 2014 na spletnem naslovu http://oz.berkeley.edu/~breiman/randomforest2001.pdf
8. Caruana, R. & Freitag, D. (1994) Greedy Attribute Selection. Najdeno 20. februarja 2014 na spletnem naslovu http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.41.3576&rep=rep1&type=pdf

9. Chapados, N. (2010, 8. februar). Data Mining Algorithms for Actuarial Ratemaking. Najdeno 28. februarja 2014 na spletnem naslovu http://www.apstat.com/documents/apstat_datamining_insurance.pdf
10. Chattamvelli, R. (2008). *Data Mining Methods*. Oxford: Alpha Science International Ltd.
11. Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*. Najdeno 8. avgusta 2013 na spletnem naslovu http://www.jair.org/media/953/live-953-2037-jair.pdf
12. Chawla, N. V., Japkowicz, N. & Kołcz, A. (2004). Editorial: Special Issue on Learning from Imbalanced Data Sets. Najdeno 8. avgusta 2013 na spletnem naslovu https://www3.nd.edu/~dial/papers/ACM04.pdf
13. Christmann, A. (2004). An approach to model complex high-dimensional insurance data. *Allgemeines Statistisches Archiv*, 88. Najdeno 28. februarja 2014 na spletnem naslovu http://www.stoch.uni-bayreuth.de/de/CHRISTMANN/Christmann_files/ChristmannASTA.pdf
14. Cohen, W. W. (1995). Fast Effective Rule Induction. Najdeno 15. marca 2014 na spletnem naslovu http://www.cs.utsa.edu/~bylander/cs6243/cohen95ripper.pdf
15. Dasu, T. & Johnson, T. (2003). *Exploratory Data Mining and Data Cleaning*. Hoboken: John Wiley & Sons, Inc.
16. Deloitte Touche Tohmatsu. (2007). Advanced analytics and the art of underwriting: Transformation the insurance industry. Najdeno 11. marca 2014 na spletnem naslovu http://www.deloitte.com/assets/Dcom-Australia/Local%20Assets/Documents/Deloitte_FSI_AdvancedAnalytics.pdf
17. Devale, A. B. & Kulkarni, R. V. (2012). Applications of Data Mining Techniques in Life Insurance. *International Journal of Data Mining & Knowledge Management Process, 2*(4). Najdeno 31. januarja 2014 na spletnem naslovu http://airccse.org/journal/ijdkp/papers/2412ijdkp04.pdf
18. Du, H. (2010). *Data Mining Techniques and Applications: An Introduction*. Andover: Cengage Learning EMEA.
19. Easy.Data.Mining™. (b.l.). Data Mining Example: Insurances. Najdeno 31. januarja 2014 na spletnem naslovu http://www.easydatamining.com/en/data-mining/data-mining-in-practice/insurances/
20. Fric, L. & Prijanovič, D. (2010). *Premoženjska zavarovanja* (1. izd.). Ljubljana: Slovensko zavarovalno združenje, g. i. z.
21. Ganganwar, V. (2012). An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering, 2*(4). Najdeno 28. februarja 2014 na spletnem naslovu http://www.ijetae.com/files/Volume2Issue4/IJETAE_0412_07.pdf
22. Gullickson, A. (2005). Lecture Outline for Sociology G4075: Introduction to Social Data Analysis II. Najdeno 15. julija 2014 na spletnem naslovu http://pages.uoregon.edu/aarong/teaching/G4075_Outline/G4075_Outline.html
23. Guyon, I. & Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. Najdeno 15. junija 2013 na spletnem naslovu

http://machinelearning.wustl.edu/mlpapers/paper_files/GuyonE03.pdf
24. Hall, M. A. (1999). Correlation-based Feature Selection for Machine Learning. Najdeno 20. februarja 2014 na spletnem naslovu http://www.cs.waikato.ac.nz/~mhall/thesis.pdf
25. Hall, M. A. & Holmes, G. (2003). Benchmarking Attribute Selection Techniques for Discrete Class Data Mining. *IEEE Transactions on Knowledge and Data Engineering*, *15*(3). Najdeno 20. februarja 2014 na spletnem naslovu http://www.cs.waikato.ac.nz/~mhall/HallHolmesTKDE.pdf
26. Han, H., Wang, W.-Y. & Mao, B.-H. (2005). Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. Najdeno 15. marca 2014 na spletnem naslovu http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.308.9315&rep=rep1&type=pdf
27. Han, J. & Kamber, M. (2001). *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann.
28. Hand, D. J., Mannila, H. & Smyth, P. (2001). *Principles of Data Mining*. Cambridge: MIT Press.
29. Holzapfel, M. & Schmidt, M. (2003). Ripper: Fast Effective Rule Induction. Najdeno 28. aprila 2014 na spletnem naslovu http://read.pudn.com/downloads95/sourcecode/others/385491/Ripper.ppt
30. IBM Corporation. (2011). Using Data Mining to Detect Insurance Fraud. Najdeno 11. marca 2014 na spletnem naslovu http://public.dhe.ibm.com/common/ssi/ecm/en/imw14283usen/IMW14283USEN.PDF
31. John, G. H. & Langley, P. (1995). Estimating Continuous Distributions in Bayesian Classifiers. Najdeno 28. aprila 2014 na spletnem naslovu http://www.cs.iastate.edu/~jtian/cs573/Papers/John-UAI-95.pdf
32. Kahane, Y., Levin, N., Meiri, R. & Zahavi, J. (2005). Applying Data Mining Technology for Insurance Rate Making – An Example of Automobile Insurance. Najdemo 20. februarja 2014 na spletnem naslovu http://www.fbe.bahcesehir.edu.tr/UserFiles/File/RiskArticleFinal%20-%20Kahane%20et%20al%20Journal%20APRIA%20to%20Irini.pdf
33. Klösgen, W. & Żytkow, J. M. (2002). *Handbook of data mining and knowledge discovery*. Oxford: Oxford University Press, Inc.
34. *KNIME*. (2013). Najdeno 12. aprila 2013 na spletnem naslovu http://www.knime.org/
35. Komelj, J. (2012). *Splošno zavarovanje* (interno gradivo). Ljubljana: Ekonomska fakulteta.
36. Kononenko, I. (1997). *Strojno učenje* (1. izd.). Ljubljana: Založba FE in FRI.
37. Kononenko, I. & Kukar, M. (2007). *Machine Learning and Data Mining: Introduction to Principles and Algorithms.* Chichester: Horwood Publishing.
38. Kudyba, S. & Hoptroff, R. (2001). *Data Mining and Business Intelligence: A Guide to Productivity*. Hershey: Idea Group Publishing.
39. Lanzi, P. L. (2007, 11. april). Slideshare. Najdeno 28. aprila 2014 na spletnem naslovu http://www.slideshare.net/pierluca.lanzi/machine-learning-and-data-mining-12-classification-rules
40. Larose, D. T. (2005). *Discovering knowledge in data: an introduction to data mining*. Hoboken: John Wiley & Sons, Inc.

41. le Cessie, S. & van Houwelingen, J.C. (1992). Ridge Estimators in Logistic Regression. Najdeno 15. marca 2014 na spletnem naslovu http://sci2s.ugr.es/keel/pdf/algorithm/articulo/1992-JSTOR-Cessie-Logistic_Regression.pdf
42. Michalski, R. S., Bratko, I. & Kubat, M. (1998). *Machine Learning and Data Mining: Methods and Applications*. Chichester: John Wiley & Sons, Ltd.
43. Pal, S. K. & Mitra, P. (2004). *Pattern Recognition Algorithms for Data Mining: Scalability, Knowledge Discovery and Soft Granular Computing*. Boca Raton: CRC Press.
44. Platt, J. C. (1998). Fast Training of Support Vector Machines using Sequential Minimal Optimization. Najdeno 30. aprila 2014 na spletnem naslovu http://research.microsoft.com/en-us/um/people/jplatt/smo-book.pdf
45. Provost, F. (2000). Machine Learning from Imbalanced Data Sets 101. Najdeno 8. avgusta 2013 na spletnem naslovu https://archive.nyu.edu/bitstream/2451/27763/2/CPP-02-00.pdf
46. Putler, D. S. & Krider, R. E. (2012). *Customer and Business Analytics: Applied Data Mining for Business Decision Making Using R*. Boca Raton: CRC Press.
47. Quinlan, J. R. (1992). Learning with Continuous Classes. Najdeno 30. aprila 2014 na spletnem naslovu http://sci2s.ugr.es/keel/pdf/algorithm/congreso/1992-Quinlan-AI.pdf
48. Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Mateo: Morgan Kaufmann.
49. Reinartz, T. (1999). *Focusing Solutions for Data Mining: Analytical Studies and Experimental Results in Real-World Domains*. Berlin: Springer.
50. Ristin, G., Korbar, T. & Simoniti, S. (2008). *Zakon o obveznih zavarovanjih v prometu (ZOZP) s komentarjem*. Ljubljana: Slovensko zavarovalno združenje, GIZ.
51. SAS Institute Inc. (2002). Data Mining in the Insurance Industry: Solving Business Problems using SAS® Enterprise Miner™ Software. Najdeno 18. februarja 2014 na spletnem naslovu http://www.google.si/url?sa=t&rct=j&q=&esrc=s&source=web&cd=8&ved=0CHEQFjAH&url=http%3A%2F%2Fishare.edu.sina.com.cn%2Fdownload.php%3Ffileid%3D18315475&ei=-mYvU8CfN-GlyAOf-oDYCw&usg=AFQjCNG8jNsu4xp5Gng-Arp2-VvK82BMCw&sig2=vHAD0rXlfkZCkSxBBTUF6w
52. Scism, L. & Maremont, M. (2010, 19. november). Insurers Test Data Profiles to Identify Risky Clients. *The Wall Street Journal*. Najdeno 31. januarja 2014 na spletnem naslovu http://online.wsj.com/news/articles/SB10001424052748704648604575620750998072986
53. Slovensko zavarovalno združenje. (2013). *Statistični zavarovalniški bilten 2013*. Ljubljana: Slovensko zavarovalno združenje, GIZ.
54. Steinberg, D. (2008). Data Mining Opportunities in Health Insurance. Najdeno 18. februarja 2014 na spletnem naslovu http://www.ehcca.com/presentations/predmodel2/2_02.pdf
55. Sumathi, S. & Sivanandam, S. N. (2006). *Introduction to Data Mining and its Applications*. Berlin: Springer–Verlag.
56. Tan, P.-N., Steinbach, M. & Kumar, V. (2014). *Introduction to data mining* (1st ed.). Harlow: Pearson Education Limited.
57. Weiss, G. M. (2004). Mining with rarity: A unifying framework. *Sigkdd Explorations, 6*(1). Najdeno 16. decembra 2013 na spletnem naslovu

http://www.kdd.org/sites/default/files/issues/6-1-2004-06/weiss.pdf
58. *Weka*. (2013). Najdeno 12. aprila 2013 na spletnem naslovu http://www.cs.waikato.ac.nz/~ml/weka/
59. Weng, C. G. & Poon, J. (2006). A New Evaluation Measure for Imbalanced Datasets. Najdeno 16. decembra 2013 na spletnem naslovu http://crpit.com/confpapers/CRPITV87Weng.pdf
60. Witten, I. H. & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques* (2nd ed.). Amsterdam: Morgan Kaufmann Publishers.
61. Zakon o obveznih zavarovanjih v prometu. *Uradni list RS* št. 93/2007-UPB3.
62. Zakon o varnosti cestnega prometa. *Uradni list RS* št.56/2008-UPB5.
63. Zakon o zavarovalništvu. *Uradni list RS* št. 99/2010-UPB7.

PRILOGA

Priloga 1: Slovarček in kratice

AUC	area under the ROC curve - ploščina pod krivuljo ROC
Bayes Net	Bayesove nevronske mreže
Cross validation	prečno preverjanje
Data Mining	podatkovno rudarjenje
Decision tree	odločitveno drevo
J48	implementacija algoritma odločitvenih dreves C4.5 v Weki
JRip	implementacija algoritma odločitvenih pravil RIPPER v Weki
k-NN	*k*-nearest neighbours - *k*-najbližjih sosedov
Logistic	algoritem logistične regresije v Weki
M5P	algoritem modelnega drevesa M5 v Weki
MLP	multilayer perceptron - večnivojski perceptron
Oversampling	nadvzorčenje
Random Forest	Naključni gozdovi (klasifikacijska kombinacijska metoda)
ROC	Receiver Operating Characteristic curve - ROC krivulja
SMO	implementacija algoritma za učenje klasifikatorja podpornih vektorjev
SVM	Support Vector Machine - Metoda podpornih vektorjev
Undersampling	podvzorčenje
Workflow	delotok