UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Magistrsko delo

Podatkovno rudarjenje na primeru avtomobilskih zavarovanj

(Data mining in case of car insurance)

Ime in priimek: Sabrina Vatovec
Študijski program: Matematika s finančnim inženiringom, 2. stopnja
Mentor: doc. dr. Branko Kavšek
Somentor: izr. prof. dr. Mihael Perman
Delovni somentor: Marko Tuta

Koper, februar 2019

Ključna dokumentacijska informacija

Ime in PRIIMEK: Sabrina VATOVEC

Naslov magistrskega dela: Podatkovno rudarjenje na primeru avtomobilskih zavarovanj

Kraj: Koper

Leto: 2019

Število listov: 85 Število slik: 16 Število tabel: 5

Število referenc: 32

Mentor: doc. dr. Branko Kavšek

Somentor: izr. prof. dr. Mihael Perman

Delovni somentor: Marko Tuta

UDK: 004.42:368(043.2)

Ključne besede: zavarovalništvo, podatkovno rudarjenje, strojno učenje, klasifikacijski algoritmi, Naivni Bayes, odločitvena drevesa, metoda podpornih vektorjev

Math. Subj. Class. (2010): 68T01

Izvleček: V magistrskem delu predstavimo uporabo klasifikacijskih tehnik podatkovnega rudarjenja na primeru problema iz zavarovalništva – obnovitev polic avtomobilskih zavarovanj. Na začetku teoretičnega dela opišemo nekaj pojmov iz zavarovalništva, ki jih potrebujemo za razumevanje problema, ki ga obravnavamo v praktičnem delu. Nato v splošnem predstavimo podatkovno rudarjenje, metodologijo CRISP-DM ter proces predpriprave podatkov. V teoretičnem delu opišemo klasifikacijo ter izpeljavo nekaterih klasifikatorjev kot so Naivni Bayes, odločitvena drevesa ter metoda podpornih vektorjev. V zaključku teoretičnega dela opišemo, kako z različnimi merami natančnosti ocenjujemo rezultate klasifikatorjev. V praktičnem delu se posvetimo problemu obnovitev polic avtomobilskih zavarovanj. Cilj tega dela je zgraditi čim bolj natančen klasifikator, ki bi na podlagi podatkov vezanih na zavarovalno polico, napovedal, ali je bolj verjetno, da bo polica ob poteku zavarovalnega leta obnovljena ali ne. Za ta namen uporabimo klasifikacijske algoritme, s katerimi lahko zgradimo modele, ki police klasificirajo v obnovljene ali neobnovljene. Na podatkih ene od slovenskih zavarovalnic uporabimo klasifikacijske algoritme, ki smo jih opisali v teoretičnem delu. Na koncu primerjamo rezultate različnih klasifikatorjev in podrobneje opišemo odločitvena drevesa, ki so se v tem primeru izkazala za najbolj natančno klasifikacijsko orodje.

Key words documentation

Name and SURNAME: Sabrina VATOVEC

Title of the thesis: Data mining in case of car insurance

Place: Koper

Year: 2019

Number of pages: 85 Number of figures: 16 Number of tables: 5

Number of references: 32

Mentor: Assist. Prof. Branko Kavšek, PhD

Co-Mentor: Assoc. Prof. Mihael Perman, PhD

Working Co-Mentor: Marko Tuta

UDK: 004.42:368(043.2)

Keywords: insurance, data mining, machine learning, classification algorithms, Naive Bayes, decision trees, support vector machines

Math. Subj. Class. (2010): 68T01

Abstract: In this work we present the use of data mining classification techniques on a problem from an insurance company – the problem of renewal of car insurance policies. At the beginning of the theoretical part we describe some of the concepts from the insurance industry that are needed for understanding the problem at hand. Then, we present data mining in general, data mining tasks, the CRISP-DM methodology and the process of data preparation. The theoretical part of this work is mainly devoted to describing in detail classification and derivation of some classifiers such as the Naive Bayes, decision trees and support vector machines. In the conclusion of the theoretical part we describe how to evaluate the results of the classifiers using different measures for classification accuracy. In the practical part we focus on the problem of renewal of car insurance policies. The aim is to build the most accurate classifier, based on data related to the insurance policy, that would predict whether it is more likely that the policy will be renewed at the end of the insurance year or not. To build such classifiers we use classification algorithms described in the theoretical part. We apply classifiers on data from one of the the Slovenian insurance companies. Finally, we compare the results of different classifiers and give a detailed presentation of decision trees that in this case proved to be the most accurate classification tool.

Zahvala

Zahvaljujem se mentorju, doc. dr. Branku Kavšku, za vso pomoč, koristne nasvete ter strokovno vodenje tekom izdelave magistrskega dela.
Hkrati se zahvaljujem somentorju, izr. prof. dr. Mihaelu Permanu, za pomoč in konstruktivne nasvete pri pisanju magistrskega dela.
Za pomoč se zahvaljujem tudi Marku Tuti, ki je zagotovil podatke, s katerimi sem lahko delala v praktičnem delu te naloge.

Zahvaljujem se tudi prijateljem in sošolcem za pomoč pri študiju in za vse lepe trenutke med študijem.

Posebna zahvala gre Sandiju ter mojim staršem in sestri za vso moralno podporo in potrpežljivost v času študija.

Kazalo vsebine

Kazalo tabel

Kazalo slik

Seznam kratic

AO	avtomobilska odgovornost; zavarovanje avtomobilske odgovornosti
AK	avtomobilski kasko; zavarovanje avtomobilskega kaska
AO+	zavarovanje voznika za škodo zaradi telesnih poškodb
AUC	area under the ROC curve = ploščina pod ROC krivuljo
CART	Classification And Regression Trees = Klasifikacijska in regresijska drevesa
CRISP-DM	Cross Industry Standard for Data Mining = industrijski standard za podatkovno rudarjenje
*ID*3	Iterative Dichotomiser 3
KDD	knowledge discovery in data = pridobivanje znanja iz podatkov
KKT	Karush–Kuhn–Tucker
ROC	receiver operating characteristic = karakteristike delovanja sprejemnika
SMO	Sequential Minimal Optimization = minimalna sekvenčna optimizacija
SVM	Support Vector Machines = metoda podpornih vektorjev
WEKA	Waikato Environment for Knowledge Analysis = odprtokodna programska oprema za strojno učenje, ki je nastala na Univerzi Waikato

1 UVOD

V zadnjih desetletjih se je tehnologija zbiranja podatkov razvijala z izjemno hitrostjo. To je ustvarilo potrebo po novih, zmogljivejših orodjih za pretvorbo ogromnih količin podatkov v koristno znanje. Z razvojem metod, ki pripomorejo k zahtevnejši analizi podatkov, je nastalo novo raziskovalno področje, ki ga pogosto imenujemo podatkovno rudarjenje (ang. data mining). Podatkovno rudarjenje predstavlja širok pojem, ki ga lahko v splošnem interpretiramo kot pridobivanje novega znanja iz podatkov z namenom sprejemanja dobrih poslovnih odločitev. Pri podatkovnem rudarjenju gre za združevanje znanja z več področij in sicer iz strojnega učenja, statistike, urejanja podatkovnih baz ter tudi tehnik vizualizacije. Ena od pomembnejših in pogosteje uporabljenih nalog podatkovnega rudarjenja je klasifikacija, ki je v tej nalogi širše predstavljena in tudi uporabljena na področju zavarovalništva. Zavarovalnice so ene tistih ustanov, ki vsakodnevno beležijo ogromne količine različnih podatkov. Beležijo vse podatke o zavarovalnih pogodbah, nekatere podatke o strankah, ki sklepajo pogodbe ter ostale vrste podatkov, ki jih potrebujejo za dobro poslovanje. Eden od problemov, ki ga zavarovalnice pogosto rešujejo s pomočjo tehnik podatkovnega rudarjenja, je detekcija goljufivih strank. To delo pa opisuje, kako lahko podatkovno rudarjenje uporabimo na problemu obnovitve zavarovalnih polic. Za vsako zavarovalno polico je določen datum sklenitve, datum začetka zavarovalnega leta ter datum poteka zavarovalnega leta. Na datum poteka zavarovalnega leta ima stranka možnost pogodbo obnoviti. V kolikor stranka za zavarovalnico ne predstavlja pretirane grožnje, si zavarovalnica želi, da bi stranka pogodbo obnovila. V tem primeru se nam poraja vprašanje, v katerih primerih se stranke odločijo zavarovalno polico obnoviti ali ne-obnoviti oziroma kateri podatki o polici ali stranki vplivajo na odločitev o obnovitvi. Da bi odgovorili na to vprašanje, smo v magistrskem delu uporabili klasifikacijske algoritme na podatkih, sestavljenih iz zavarovalnih polic, za katere poznamo podatek o tem ali so bile ob datumu poteka obnovljene ali ne. Na podlagi teh podatkov smo želeli poiskati čim bolj natančen model, ki bi police, glede na njihove lastnosti, klasificiral v dva razreda in sicer na tiste, ki bodo obnovljene in na tiste, ki ne bodo. Podatke, ki smo jih uporabili, smo dobili od ene izmed slovenskih zavarovalnic in vsebujejo primere polic avtomobilskih zavarovanj.

Vsebina magistrskega dela je strukturirana na sledeči način. V drugem poglavju so na kratko predstavljeni nekateri pojmi iz področja zavarovalništva ter vrste avto-

mobilskih zavarovanj. V tretjem poglavju predstavimo pomen in naloge podatkovnega rudarjenja [8] ter metodologijo CRISP-DM [5], ki podrobneje opisuje proces izvajanja podatkovnega rudarjenja. Za dobro delovanje (klasifikacijskih) algoritmov je potrebna predpriprava podatkov, zato v tem delu opišemo tudi proces predpriprave podatkov, ki vključuje čiščenje podatkov in transformacijo. Glavno teoretično poglavje magistrskega dela je četrto poglavje z naslovom klasifikacija, kjer podrobneje opišemo nekatere klasifikatorje, ki smo jih uporabili na podatkih. Prvi tak klasifikator je Naivni Bayes [13, 17, 18], ki je statistični klasifikator, pri katerem je uporabljena Bayesova teorija z določenimi predpostavkami o neodvisnosti spremenljivk v podatkih. Za tem so predstavljena odločitvena drevesa, ki zaradi svoje enostavne strukture predstavljajo eno od pogostejših orodij, ki se uporablja za klasifikacijo. Pri odločitvenih drevesih nekoliko podrobneje opišemo algoritma ID3 [22] ter C45 [15] ter informacijsko teorijo [23], ki predstavlja temelj algoritmov odločitvenih dreves. Nazadnje predstavimo tudi metodo podpornih vektorjev [14, 21], pri kateri se podatke preslika v višjo dimenzijo ter se v novem prostoru išče najboljšo hiperravnino, ki klasificira primere v podatkovni množici. Za reševanje optimizacijskega problema pri metodi podpornih vektorjev potrebujemo tudi nekaj teorije iz konkveksne optimizacije [3], ki jo v tem delu tudi na kratko opišemo. Da bi lahko klasifikatorje primerjali med sabo, je potrebno vedeti, na kakšen način se ocenjuje njihovo natančnost. Mere, ki jih pri tem uporabljamo, so opisane v razdelku 4.4 [8]. V petem poglavju predstavimo, na kakšen način smo uporabili opisano teorijo na podatkih, ki so nam bili na voljo. Pogledamo si rezultate, ki so nam jih dali različni klasifikacijski algoritmi in te primerjamo med seboj ter podamo njihovo interpretacijo. Zadnje, šesto poglavje, je namenjeno podajanju zaključkov in predlogov za nadaljnje delo.

2 PREDSTAVITEV PROBLEMA

Za odkrivanje znanja v podatkih je potrebno dobro poznati področje, iz katerega podatki izhajajo. V tem magistrskem delu je to področje zavarovalništvo oziroma natančneje podpodročje avtomobilskih zavarovanj. V tem poglavju je predstavljenih nekaj osnovnih pojmov s področja zavarovalništva in avtomobilskih zavarovanj.

2.1 Zavarovalništvo, njegov pomen in osnovni pojmi

Praviloma se vsi ljudje v svojem življenju slej ko prej srečamo s pojmom zavarovanja. Obstaja nekaj definicij pojma "zavarovanje", ki povzemajo njegov pomen. Eno bolj znanih definicij, ki opisuje širši pojem zavarovalništva, je zapisal teoretik zavarovalne ekonomije dr. Boncelj, ki v svoji knjigi Zavarovalna ekonomika [2] navaja, da je zavarovanje ustvarjanje gospodarske varnosti z izravnavanjem gospodarskih nevarnosti. Temeljni gospodarski pomen zavarovanja je zaščititi premoženje in osebe pred raznimi riziki oziroma škodnimi dogodki. Zavarovalni riziki so negotovi bodoči dogodki oziroma nevarnosti, na katere zavarovana oseba s svojo voljo ne more vplivati. Zavarovalnica mora znati izmeriti velikost rizika, ki ga zavaruje in na podlagi tega določiti ustrezno zavarovalno premijo, ki jo mora stranka plačati, v kolikor želi rizik prenesti v kritje zavarovalnici. Zavarovalnica lahko oceni velikosti rizikov s pomočjo statističnih podatkov o preteklih škodnih dogodkih. Pomembno je, da zavarovalnica tveganja, katera zavaruje, razporedi na številne zavarovance, saj lahko le na tak način natančneje oceni pričakovano škodo.

Na zavarovalnem trgu se nahaja več subjektov, med katere sodijo ponudniki, uporabniki ter vmesni členi. Med ponudnike zavarovalnih poslov spadajo zavarovalnice in pozavarovalnice, med uporabnike sodijo fizične ter pravne osebe, vmesni člen pa so zavarovalni zastopniki ter zavarovalni posredniki. Za izvajanje zavarovalniških dejavnosti obstaja veliko zakonov, ki so za vse zavarovalnice skupni in se jih morajo te strogo držati. Eden od organov, ki skrbi za to, da se zavarovalnice teh držijo, je Agencija za zavarovalni nadzor (AZN). AZN izdaja tudi dovoljenja za opravljanje zavarovalnih poslov, preverja poročila zavarovalnic, nadzoruje poslovanja zavarovalnic itd. [10].

V nadaljevanju je opisanih nekaj zavarovalnih izrazov, ki jih potrebujemo za opis podatkov, na katerih so bile uporabljene tehnike podatkovnega rudarjenja. Podrob-

nejšo in popolnejšo razlago zavarovalnih pojmov lahko bralec najde v [29]. Oseba, ki z zavarovalnico sklene zavarovalno pogodbo, se imenuje **zavarovalec**. Poleg tega pa je za vsako zavarovalno pogodbo določen tudi **zavarovanec**, ki je oseba, katere zavarovalni interes je zavarovan. V večini pogodb sta to isti osebi, ni pa to nujno. Sestavni del zavarovalne pogodbe je **zavarovalna polica**, ki vsebuje vse bistvene podatke o zavarovalnem razmerju – predmet zavarovanja, nevarnosti, ki so za ta predmet zavarovane, čas trajanja zavarovanja, obliko zavarovalnega kritja, izračun zavarovalne premije ter tudi splošne in posebne pogoje zavarovalne pogodbe. S pogodbo se zavarovalec zaveže, da bo zavarovalnici plačal **premijo**, zavarovalnica pa se zaveže, da bo izplačala odškodnino ali zavarovalnino, v kolikor pride do zavarovalnega primera. Za vsak zavarovalni produkt imajo zavarovalnice način, s katerim izračunajo zavarovalno premijo in kaj pri tem upoštevajo. Pri nekaterih produktih (npr. produkti avtomobilskih zavarovanj) zavarovalnice za določanje zavarovalne premije upoštevajo tudi sistem bonus – malus, ki vpliva na višino zavarovalne premije, tako da upošteva škodne dogodke, ki jih je imel zavarovanec v predhodnih zavarovalnih letih. Tisti zavarovanec, ki v preteklih letih na polici ni imel škodnih dogodkov, ima pravico do plačila nižje zavarovalne premije od izhodiščne in temu pravimo bonus. V tem primeru zavarovancu pripada nižji premijski razred. Obratno, tisti zavarovanec s škodnimi primeri (enim ali več), pa mora plačati višjo zavarovalno premijo od izhodiščne in temu pravimo malus. V tem primeru zavarovancu pripada višji premijski razred. Začetni premijski razred predstavlja 14. razred. To je razred z 0% bonusa. Skupaj obstaja 20 premijskih razredov.

Poleg števila škodnih dogodkov je za zavarovalnico eden od pomembnih kazalnikov o obnašanju strank (zavarovalcev in zavarovancev) razmerje med likvidiranimi škodami in vplačanimi premijami, ki ga imenujemo tudi **škodni rezultat**. Zavarovalnice škodne rezultate računajo za posamična zavarovanja, produkte, zavarovalce ter zavarovance in to uporabljajo kot eno od meril za uspeh poslovanja zavarovalnice.

2.2 Avtomobilska zavarovanja

Znano je, da se cestni promet v zadnjih desetletjih znatno povečuje po vsem svetu. V Evropi, kot tudi v Sloveniji, večina ljudi za prevozno sredstvo uporablja osebni avtomobil, zato ni presenetljiv podatek, da so avtomobilska zavarovanja ena pomembnejših zavarovanj v Sloveniji oziroma v celotni Evropi. V poročilu Slovenskega zavarovalnega združenja za leto 2018 zasledimo podatek, da v skupini neživljenjskih zavarovanj z 28.5 odstotnim deležem prevladujejo avtomobilska zavarovanja in so tako najbolje prodajana neživljenjska zavarovanja [32].

V nadaljevanju so predstavljena avtomobilska zavarovanja, saj se podatki, na katerih smo uporabili metode podatkovnega rudarjenja, nanašajo na police iz zavarovanj osebnih avtomobilov. Obstaja več možnih delitev zavarovanj in sicer glede na kritje nevarnosti, predmet zavarovanja, način odločitve ter na gospodarsko panogo. Zakon o zavarovalništvu (Zzavar-1) v 7. členu navaja delitev zavarovanj glede na nevarnosti, ki jih krijejo. Zavarovanja deli na zavarovalne vrste. Ta del zakona navaja tudi zavarovalne podskupine, ki združujejo zavarovanja iz več zavarovalnih vrst. Ena od zavarovalnih podskupin je tudi zavarovanje vozil in pod to podskupino spadajo tudi avtomobilska zavarovanja. Zakon o zavarovalništvu pod to podskupino uvršča spodaj našteta zavarovanja [31].

- **Zavarovanje odgovornosti** pri uporabi vozil je zavarovanje, ki krije vse odgovornosti, ki izhajajo iz uporabe kopenskih vozil z lastnim pogonom. Med to vrsto zavarovanj spada zavarovanje avtomobilske odgovornosti ali krajše AO, ki je eno od pomembnejših avtomobilskih zavarovanj. Zakon o obveznih zavarovanjih v prometu [30] določa, da je zavarovanje AO obvezno za vsa vozila vključena v prometu. Iz naslova tega zavarovanja se krije škoda, ki jo oseba v prometni nesreči s svojim vozilom nenamerno povzroči tretjim osebam. Zavarovalnica v tem primeru oškodovancu povrne materialno škodo (npr. škodo na avtomobilu druge osebe, škodo na stavbi) in nematerialno škodo (npr. telesne poškodbe, okvare zdravja, smrt) [28].

- **Zavarovanje kopenskih vozil** (razen tirnih vozil) je zavarovanje, ki krije škode na kopenskih vozilih na lasten pogon ter kopenskih vozil brez lastnega pogona. Pri avtomobilskih zavarovanjih je to zavarovanje bolj znano po imenu avtomobilski kasko ali krajše AK. To krije škodo nastalo na lastnem vozilu zavarovanca, ki nastane zaradi presenetljivih dogodkov, na katere voznik s svojo voljo ne more vplivati. Gre lahko za škodo, ki nastane zaradi uničenja, poškodovanja vozila ali tudi izginotja in kraje [27, 28]. Vsaka zavarovalnica, ki nudi zavarovanje AK, ima v ponudbi določene različne kombinacije kritij pri kasko zavarovanju.

- **Nezgodno zavarovanje**, ki krije izplačila zaradi poškodbe, okvare zdravja ali smrti potnikov (sopotnikov in voznika), ki nastanejo ob nezgodi, prav tako spada med zavarovanje vozil [28]. Del nezgodnih zavarovanj, ki je povezan z zavarovanji vozil, vendar po zakonu o zavarovalništvu ne spada pod to podskupino, pa krije tudi razna denarna nadomestila ali povračila stroškov do katerih pride v primeru nezgode.

- **Zavarovanje prevoza blaga** zavaruje blago ter tudi izgubo blaga, ki se ga prevaža, vključno s prtljago.

Med zavarovanji, ki so zgoraj našteta, je obvezno le zavarovanje AO, ostala so sklenjena prostovoljno. Obstaja pa še nekaj dodatnih zavarovanj, ki se prostovoljno sklepajo skupaj s temi in nudijo kritja, ki jih prej omenjena zavarovanja ne krijejo.

- **Dodatek k zavarovanju AO** je zavarovanje voznika za škode, ki nastanejo zaradi telesnih poškodb ali tudi njegove smrti in ima kratico AO plus [29].
- Poleg ostalih zavarovanj, ki se nanašajo na motorna vozila, pa nekatere zavarovalnice nudijo tudi **avtomobilsko asistenco**, ki upravičencu nudi pomoč in kritje stroškov, ki nastanejo zaradi težav z vozilom na potovanju oziroma v ostalih primerih, ko je odsoten od doma [29].

Podatek v poročilu Slovenskega zavarovalnega združenja o motornih vozilih za leto 2018 pravi, da je v Sloveniji med motornimi vozili največ osebnih avtomobilov in sicer kar slabe tri četrtine [32]. Prav zavarovanja osebnih avtomobilov predstavljajo za zavarovalnice, ki nudijo takšen tip zavarovanj, eno izmed bolj zanimivih področij, zaradi visokega deleža premij, ki ga tvorijo. V tem magistrskem delu so metode strojnega učenja uporabljene na podatkih polic osebnih avtomobilov.

V nadaljevanju je opisano, kako je razvoj tehnologije pripeljal do nastanka podatkovnega rudarjenja ter kakšna je interpretacija izraza podatkovno rudarjenje. Na kratko so v naslednjem poglavju predstavljene tudi tehnike podatkovnega rudarjenja, tehnike predpriprave podatkov ter možni tipi podatkov, ki jih pri tem uporabljamo.

3 PODATKOVNO RUDARJENJE

To poglavje o podatkovnem rudarjenju je v večini povzeto iz [8], kjer lahko bralec najde tudi obširnejšo razlago. V današnjem času se možnosti beleženja in shranjevanja najrazličnejših podatkov čedalje bolj povečujejo. Zlasti v poslovnem svetu se v podatkovnih bazah preko transakcij vsakodnevno zbirajo ogromne količine podatkov. Družbi, v kateri trenutno živimo, pravimo tudi "informacijska družba". Dejstvo pa je, da lahko rečemo, da živimo bolj v "podatkovni" kot v "informacijski dobi".

Od 60ih let prejšnjega stoletja dalje se področje zbiranja podatkov in informacijske tehnologije sistematično razvija. Začelo se je z nadgradnjo osnovnih podatkovnih baz na sisteme podatkovnih baz. V 70ih letih so se začele pojavljati relacijske podatkovne baze ter orodja za modeliranje podatkov. Izboljšale so se tudi metode za dostop do podatkov preko poizvedbenih jezikov, uporabniških vmesnikov, upravljanja s transakcijami, itd. Po vzpostavitvi sistemov za upravljanje baz podatkov se je tehnologija podatkovnih baz razvijala v naprednejše sisteme baz podatkov in začela so nastajati tudi podatkovna skladišča. Z razvojem informacijske tehnologije in razpoložljivostjo velikih količin podatkov je nastala potreba po učinkovitejših orodjih, ki bi ogromne količine podatkov spremenili v uporabno znanje oziroma v uporabne informacije. Ta potreba je pripeljala do nastanka novejšega področja analize podatkov, ki ga imenujemo **podatkovno rudarjenje** (ang. data mining).

Pred resnejšim razvojem podatkovnega rudarjenja so se za reševanje nekaterih praktičnih problemov uporabljale tehnike analize podatkov, ki so vključevale razne statistične metode, regresijsko analizo, analizo časovnih vrst, ipd. Statistika je ena pomembnejših vej pri analiziranju podatkov, ampak včasih bi si želeli, da bi bilo v podatkih možno pojasniti obstoj določenih odvisnosti z "natančnejšimi" modeli in za to potrebujemo dodatna orodja. Pri statistiki analitik sam predlaga katere atribute bo primerjal in katero mero bo za to uporabil, vendar so v podatkih lahko tudi zanimivi vzorci, ki na prvi pogled niso poznani. Ko z uporabo statističnih metod v naboru podatkov ne najdemo koristnih informacij, pride na vrsto podatkovno rudarjenje. S prizadevanjem za boljši razvoj orodij za obdelavo podatkov so se raziskovalci s časom obrnili na področje strojnega učenja. Glavna naloga strojnega učenja je razvoj algoritmov za pridobivanje znanja iz "preteklih" podatkov.

3.1 Definicija podatkovnega rudarjenja

Izraz podatkovno rudarjenje nima univerzalne definicije, ampak ga lahko razumemo na več načinov. Različni viri podajajo rahlo različne definicije tega izraza. V zadnjem času se za podatkovno rudarjenje pogosto uporablja izraz "pridobivanje znanja iz podatkov" (ang. knowledge discovery in data). Zaradi angleškega prevoda tega izraza, se zanj uporablja kratica KDD. Obstajata dve najpogostejši interpretaciji podatkovnega rudarjenja. Ponekod s tem mislijo na sopomenko za celoten proces odkrivanja znanja iz podatkov (KDD kot proces), medtem ko ostali vidijo podatkovno rudarjenje kot enega izmed ključnih korakov v tem procesu – v tem primeru se podatkovno rudarjenje uporablja kot sopomenka za strojno učenje. V magistrskem delu bomo podatkovno rudarjenje razumeli v njegovem širšem pomenu, torej kot celoten proces odkrivanja zanimivih vzorcev in znanja iz velikih količin podatkov.

Področje podatkovnega rudarjenja je zelo široko in se lahko uporablja v različnih panogah. Pogosto uporabljeno je na področju prodaje in marketinga (preučevanje obnašanja strank ali analiziranje košaric produktov, ki jih stranke pogosto kupujejo), financ in bančništva (analiza strank z vidika njihove potrošnje, analiza obnašanja kreditojemalcev), zavarovalništva (prepoznavanje tveganih zavarovancev), medicine (prepoznavanje uspešnosti terapij za različne bolezni) in še na mnogih ostalih področjih.

Poglejmo si nekaj izrazov, ki jih potrebujemo pri delu s podatki, ko izvajamo podatkovno rudarjenje. Množica podatkov, s katero delamo, vsebuje podatkovne objekte. Te včasih imenujemo podatkovni vzorci, primeri ali predmeti. Tu jih bomo imenovali podatkovni **primeri**. **Atribut** (ang. attribute) je količina, ki predstavlja neko značilnost podatkovnega primera. Namesto izraza atribut so v uporabi tudi izrazi dimenzija, značilnost ali spremenljivka (ang. dimension, feature or variable) (slednjega zasledimo predvsem v statistični literaturi). Niz atributov, ki opisujejo določen podatkovni primer, imenujemo vektor atributov. Opazovano vrednost podatkovnega primera za določen atribut imenujemo tudi opazovanje (ang. observation). Celotno množico podatkov najpogosteje predstavimo v obliki matrike dimenzij $n \times m$, kjer n predstavlja število primerov v podatkih, m pa predstavlja število atributov. Podatke včasih lahko predstavimo tudi grafično in sicer s pomočjo točk, ki predstavljajo vektorje. Vsaka dimenzija vektorja predstavlja en atribut.

Atribute oziroma spremenljivke lahko delimo na različne načine glede na njihove lastnosti. Ena možna delitev je glede na množico možnih vrednosti, ki jih lahko zavzemajo. V tem primeru poznamo štiri različne vrste atributov in sicer **nominalne atribute**, **binarne atribute**, **ordinalne atribute** ter **numerične atribute**. Množico vrednosti nominalnega atributa predstavljata dve ali več kategorij, ki pa jih ni možno urediti po vrstnem redu. Navadno so vrednosti (kategorije) predstavljene z besedami,

vendar je možno te kategorije označiti tudi s številkami, ampak v tem primeru te številke ne predstavljajo merljivih količinskih vrednosti, temveč samo označujejo kategorijo. Binarni atribut je nominalni atribut, ki ima samo dve možni vrednosti. Za ordinalni atribut (tako kot za nominalni) velja, da njegove vrednosti predstavljajo različne kategorije, vendar lahko le-te uredimo po vrstnem redu; ne moremo pa opredeliti razlik med vrednostmi. Zgoraj opisane vrste atributov včasih imenujemo tudi **kategorični atributi**. Za razliko od zgoraj opisanih vrst atributov za numerične atribute velja, da predstavljajo dejansko merljivo količino. Vrednosti numeričnih atributov so lahko cela ali realna števila. Poznamo dve vrsti numeričnih atributov in sicer intervalne atribute ter razmernostne atribute. Intervalni atributi so tisti, ki imajo vrednosti razporejene hierarhično in lahko med njimi izmerimo razliko, vendar pa intervalni atributi nimajo določene absolutne ničle in zaradi tega ni mogoče določiti razmerij med vrednostmi. Pri razmernostnih atributih pa je določena absolutna ničla in lahko poleg razlike med vrednostmi izmerimo tudi razmerja med vrednostmi spremenljivke.

Obstajajo pa tudi druge možne delitve atributov. Pri nekaterih algoritmih strojnega učenja se atribute velikokrat deli na **diskretne** in **zvezne** atribute. Za diskretni atribut velja, da ima končno ali številno neskončno množico vrednosti in je lahko množica njegovih vrednosti predstavljena s števili ali z znaki. Nasprotno od diskretnih so zvezni atributi, za katere velja, da je množica možnih vrednosti za ta atribut neskončna.

Orodja, ki se uporabljajo za delo s podatki, na svoj način določajo tipe podatkov oziroma tipe atributov. Orodje Weka, s katerim smo v nalogi izvajali klasifikacijske algoritme, atribute razvršča v nominalne in numerične. V Weki pod nominalne atribute spadajo atributi, ki so zgoraj opisani kot kategorični atributi (nominalni, binarni in ordinalni atributi). Navadno vsa orodja, ki jih uporabljamo za obdelavo podatkov, avtomatično določijo tipe atributov glede na vhodne podatke. Vendar je včasih za določen atribut potrebno logično premisliti, katero vrsto podatka želimo, da predstavlja in mu morda spremeniti vrsto atributa. Na primer, včasih je nek atribut označen kot numeričen, ampak ga želimo predstaviti kot kategoričnega. To se pogosto dogaja v primerih letnic datumov. Poleg tega pa velja glede tipov atributov omeniti še to, da algoritmi strojnega učenja velikokrat atribute delijo na zvezne in diskretne, vendar zvezne atribute kar enačijo z numeričnimi, čeprav vemo, da numerični atributi niso nujno zvezni.

3.1.1 Naloge podatkovnega rudarjenja

Podatkovno rudarjenje lahko nastopa v več funkcijah. Na splošno se lahko naloge podatkovnega rudarjenja razdeli v dve kategoriji in sicer na opisne in napovedne. Cilj opisnih nalog podatkovnega rudarjenja je iskanje splošne lastnosti danih podatkov, pri napovedovanju pa gre za učenje iz preteklih podatkov, s ciljem, da se napove določeno spremenljivko na novih podatkih. Največkrat uporabljene tehnike podatkovnega rudarjenja so naslednje:

1. **Opis atributa "razred" in ostalih spremenljivk** (ang. class/concept description): Množice podatkov velikokrat vsebujejo atribute, ki primere v podatkih ločijo v razne kategorije. Včasih je smiselno narediti pregled podatkov glede na vrednosti teh atributov in tako bolje spoznati podatke znotraj teh kategorij. Nekatere množice podatkov vsebujejo tudi atribut "razred" (ang. class attribute), ki predstavlja kategorično spremenljivko in ga včasih imenujemo tudi ciljna spremenljivka. Koristno je, če se množico podatkov opiše glede na vrednosti atributa "razred".

2. **Analiza povezanosti v podatkih** (ang. association analysis): Ena od pomembnejših nalog te tehnike je odkrivanje asociativnih pravil. To so pravila, ki prikazujejo vrednosti atributov, ki se v podatkih pogosto pojavijo skupaj in so oblike $X \rightarrow Y$, kjer sta X in Y dva različna atributa v podatkovni množici. Asociativno pravilo $X \rightarrow Y$ se interpretira na način, da če podatkovni primer zadostuje nekemu pogoju atributa X, bo zelo verjetno zadostoval nekemu pogoju atributa Y.

3. **Klasifikacija** (ang. classification): Nekateri podatki vsebujejo kategorični atribut, ki ga imenujemo "razred" (ang. class attribute). Če v podatkih obstaja podatek o "razredu", tem pravimo klasificirani podatki (ang. class labeled data). Pri klasifikaciji gre za to, da na podlagi množice klasificiranih podatkov nastane model, ki ga imenujemo klasifikator in s katerim lahko napovemo vrednost atributa "razred" glede na vrednosti ostalih atributov v podatkovni množici.

4. **Regresija** (ang. regression): Pri klasifikaciji napovedujemo kategorični atribut "razred", regresija pa se uporablja za napovedovanje vrednosti numeričnih atributov. Pri klasifikaciji in regresiji pravimo, da so primeri v podatkovni množici, ki jo uporabimo za izgradnjo modela, označeni z atributom, ki ga napovedujemo.

5. **Razvrščanje v skupine** (ang. clustering): Medtem ko gre pri klasifikaciji in regresiji za analizo označenih podatkovnih primerov, gre pri združevanju za gru-

piranje neoznačenih podatkovnih primerov v skupine, ki so si med sabo čim bolj različne, znotraj ene skupine pa so si primeri čim bolj podobni.

6. **Analiza osamelcev** (ang. outlier analysis): Podatki lahko vsebujejo tudi takšne primere, ki močno odstopajo od ostalih primerov. Takim podatkom pravimo osamelci in z analizo lahko takšne izjeme poiščemo. V nekaterih primerih (npr. odkrivanje goljufij) so primeri osamelcev zanimivejši kot tisti podatki, ki niso izstopajoči.

Podatkovno rudarjenje uporablja številne tehnike iz drugih področij kot so statistika, strojno učenje, podatkovna skladišča, vizualizacija podatkov, ipd. **Strojno učenje** (ang. machine learning) je eno tistih področij, ki je bilo v zadnjem času ključnega pomena pri razvoju podatkovnega rudarjenja. Spada v računalniško področje umetne inteligence, ki računalniškim sistemom omogoča sposobnost "učenja" na podlagi podatkov. Glavni cilj strojnega učenja je ta, da se računalniški program samodejno nauči prepoznati kompleksne vzorce in na podlagi teh sprejema inteligentne odločitve. Obstajata dve osnovni obliki strojnega učenja.

- **Nadzorovano učenje** (ang. supervised learning) za izgradnjo modelov uporablja označene podatkovne primere (primere, ki imajo podatek o atributu, ki ga model napoveduje). Primera nadzorovanega učenja sta klasifikacija in regresija. Pri klasifikaciji so podatki označeni s kategoričnim atributom "razred", pri regresiji pa ta "razred" predstavlja numerični (zvezni) atribut. Primer naloge nadzorovanega učenja bi bil recimo ta, da bi računalnik prepoznal ročno napisano poštno kodo glede na množico učnih primerov, ki bi vsebovala slike ročno napisanih poštnih kod z njihovimi ustreznimi računalniško prepisanimi kodami.
- **Nenadzorovano učenje** (ang. unsupervised learning) pa za razliko od nadzorovanega učenja za izgradnjo modelov uporablja podatkovne primere, ki nimajo določenega "razreda" oziroma niso označeni. Primer nenadzorovanega učenja je recimo razvrščanje v skupine (ang. clustering).

3.2 Metodologija podatkovnega rudarjenja

Običajno se za izvajanje analize s podatkovnim rudarjenjem upošteva nek standardni postopek. Splošni postopki za izvajanje analiz podatkovnega rudarjenja so se začeli pojavljati ob koncu 90ih let prejšnjega stoletja. V tem magistrskem delu za podatkovno rudarjenje na primeru avtomobilskih zavarovanj uporabljamo metodologijo **CRISP–DM**, zasnovano leta 1996, ki je danes najpogosteje uporabljana metodologija (standard) za izvajanje podatkovnega rudarjenja. CRISP–DM je okrajšava za

Cross-Industry Standard Process for Data Mining in je standardizirana metodologija, ki podatkovno rudarjenje opredeljuje kot ciklični proces [5]. Podatkovno rudarjenje je tako opisano kot cikel, ki je sestavljen iz šestih faz, njihovih pripadajočih nalog ter povezav med temi nalogami (kot je to prikazano na sliki 1). Najpogosteje si faze sledijo po vrsti v cikličnem zaporedju, je pa možno tudi vračanje v predhodne faze, če se v kasnejših fazah pokaže, da je potrebno določene naloge iz prejšnjih faz opraviti drugače ali opraviti dodatne naloge. Metodologija je osredotočena na podatke ter njihovo analizo v poslovnem okolju. V nadaljevanju so podani kratki opisi glavnih faz CRISP-DM procesa.

1. **Razumevanje problema**: Prva faza, ki je potrebna za izvedbo celotnega procesa je seveda analiza (poslovnega) problema. Pomembno je dobro razumevanje (poslovnega) ozadja problema, zato da se lahko določijo (poslovni) cilji projekta. V tej fazi se v grobem določi, katere naloge bo potrebno izvesti, da bi uresničili zastavljene cilje. V kolikor je potrebno, se v tem delu določi tudi kolikšna so sredstva, ki so na voljo za izvedbo procesa podatkovnega rudarjenja.

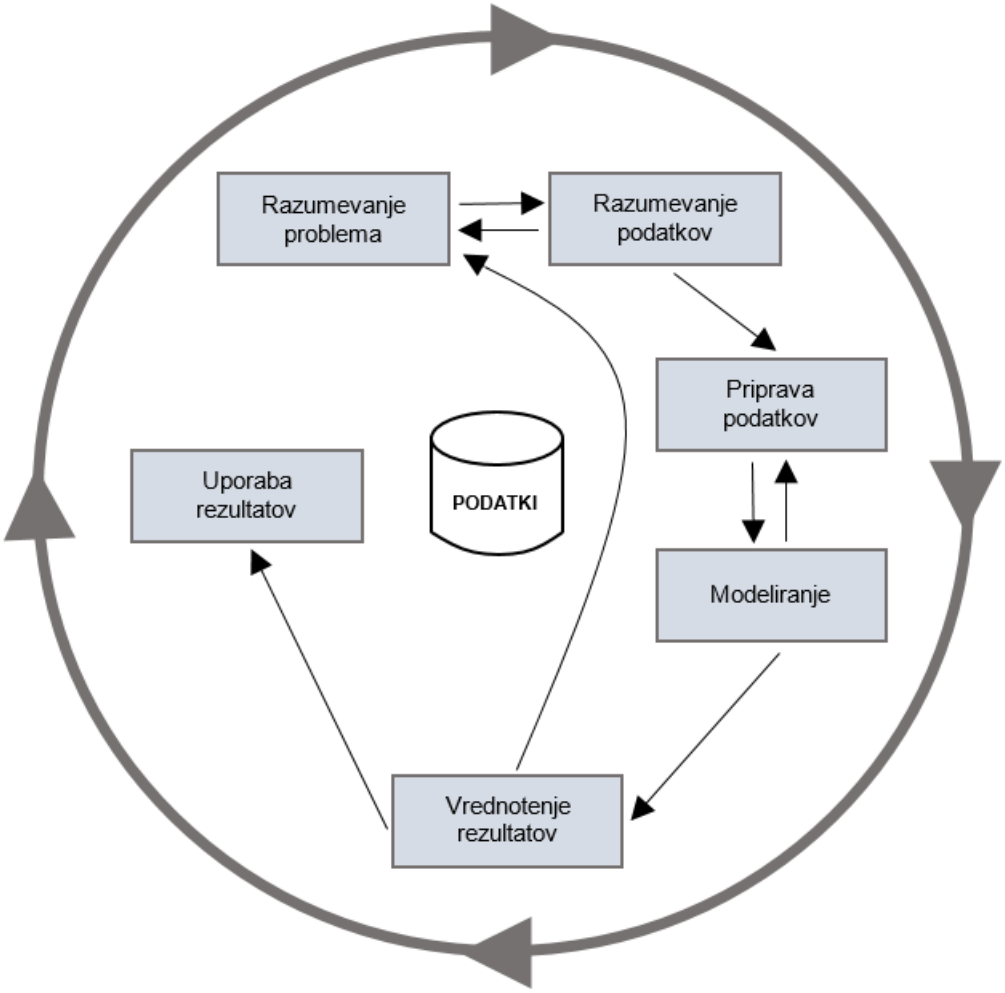
2. **Razumevanje podatkov**: Ko problem razumemo in imamo narejen načrt projekta, je pomembno, da vemo, kakšne so zahteve pri podatkih in kateri podatki so relevantni za analizo. Ta faza vsebuje zbiranje podatkov, opisovanje podatkov, raziskovanje podatkov ter preverjanje njihove kakovosti. V tej fazi se lahko izvede tudi vizualizacija podatkov.

3. **Priprava podatkov**: Ko so podatki zbrani in jih razumemo, je potrebno izbrati tiste atribute v podatkih, ki so za analizo najbolj relevantni. Podatke je (pogosto) potrebno prečistiti, transformirati v želeno obliko, obravnavati morebitne manjkajoče vrednosti ter jih ustrezno pripraviti za fazo modeliranja.

4. **Modeliranje**: V tej fazi se uporablja razne tehnike in algoritme za izgradnjo modelov. Uporabljajo se najrazličnejše statistične metode in metode strojnega učenja. Za en problem se včasih uporabi več tehnik, pri čemer ima vsaka od teh lahko različne zahteve za obliko vhodnih podatkov, zato se je včasih potrebno vrniti na fazo priprave podatkov in tu podatke prilagoditi določeni tehniki.

5. **Vrednotenje rezultatov**: Ko so enkrat modeli zgrajeni je potrebno oceniti natančnosti modelov iz vidika napovedne točnosti in/ali ustreznosti rešitve glede na dani problem. Poleg same natančnosti modelov pa je potrebno v tej fazi modele oceniti glede na doseganje poslovnih ciljev, ki so bili določeni v fazi razumevanja problema. Potrebno je ugotoviti ali je novo znanje uporabno in na kakšen način se ga lahko uporabi v poslovnem okolju. Če model vsebuje pomanjkljivosti, je potrebno le-te opisati in ugotoviti razloge zanje.

6. **Uporaba rezultatov**: Ko je model zgrajen in ovrednoten, to še ne pomeni zaključka projekta. Tudi če model deluje dobro in smo z njim pridobili želeno znanje, je potrebno to znanje še organizirati, dokumentirati in predstaviti v obliki, ki jo bo razumel končni uporabnik.



Slika 1: CRISP-DM proces

3.3 Tehnike priprave podatkov

Preden na podatkih uporabimo nek algoritem, je pomembna predpriprava podatkov. Podatki, ki jih bomo podali kot vhodne vrednosti v nek algoritem, morajo biti "kakovostni" in ustrezno pripravljeni. Zato se v fazi predpriprave podatkov poslužujemo postopkov čiščenja, redukcije, integracije in transformacije podatkov. V nadaljevanju so opisane tiste tehnike predobdelave podatkov, ki smo jih uporabili na naših podatkih o avtomobilskih zavarovanjih.

3.3.1 Čiščenje podatkov

Čiščenje podatkov pomeni reševanje problema manjkajočih vrednosti v podatkih, identifikacija in odstranjevanje ali izravnavanje močno odstopajočih podatkov (t.i. osamelcev) in odpravljanje drugih neskladnosti v podatkih.

Manjkajoči podatki

Podatke, ki vsebujejo manjkajoče vrednosti, lahko obravnavamo na več načinov. Eden od načinov je ta, da v matriki podatkov odstranimo vrstice, ki vsebujejo manjkajoči element pri nekem atributu. V tem primeru ignoriramo primer, ki smo ga izbrisali, vendar ta metoda ni najbolj učinkovita, saj lahko v primeru velikega števila manjkajočih vrednosti odstrani tudi primere, ki sicer vsebujejo koristne informacije. Problem manjkajočih podatkov pa lahko rešimo tudi tako, da manjkajoče vrednosti dopolnimo s smiselnimi podatki. Včasih se podatke dopolni ročno, vendar je to preveč zamudno, če gre za velike količine podatkov. Druga možnost je ta, da se jih dopolni vse z enako konstanto ali morebiti z oznako za manjkajočo vrednost (npr. "?"), vendar je težava v tem, da lahko potem algoritem zmotno razume, da se ti primeri povezujejo med sabo in tvorijo nekakšen zanimiv vzorec v podatkih. Ena od možnosti pa je tudi ta, da podatke dopolnimo z neko središčno mero za določen atribut, npr. s povprečjem ali mediano. To možnost lahko uporabimo tudi na ta način, da podatek dopolnimo tako, da pogledamo središčno mero atributa na podatkovnih primerih, ki so označeni z enako vrednostjo "razreda", s katero je označen manjkajoči podatek. Dodatna možnost za reševanje problema manjkajočih vrednosti je tudi ta, da manjkajoči podatek dopolnimo z vrednostjo, ki je najbolj verjetna. Le-to lahko pridobimo na primer tako, da izvedemo regresijo ali katero drugo metodo za napovedovanje.

Šum v podatkih

Podatki pogosto vsebujejo t.i. **šum** (ang. noise), ki kvari kakovost množice podatkov. Takšen šum so lahko naključne napake ali pa odstopanja v izmerjeni spremenljivki in

tem odstopajočim primerom pravimo tudi **osamelci** (ang. outliers). Pri odkrivanju osamelcev je koristno, če si podatke predstavimo grafično. Pri numeričnih podatkih si narišemo škatlo z brki (ang. box-plot) in iz te razberemo, katere vrednosti so osamelci. Pri glajenju (ang. smoothing) numeričnih podatkov zaradi šuma si lahko pomagamo tudi tako, da jih sortiramo in potem razdelimo v manjše skupine, tako da so v isti skupini sosedni primeri (npr. vrednosti 4, 8, 15, 21, 21, 24, 25, 28, 34 razdelimo v tri skupine, tako da so v prvi skupini vrednosti 4, 8, 15, v drugi 21, 21, 24 in v tretji 25, 28, 34). Podatke lahko izravnamo tako, da jih zamenjamo s povprečjem skupine, v kateri se nahajajo. Lahko jih zamenjamo tudi z mediano ali s katero drugo mero. Tudi pri šumnih podatkih se lahko le-tem izognemo tako, da "problematične" primere izločimo. Brisanje takšnih podatkov je neškodljivo takrat, ko vemo, da so ti podatki napake ali pa v primeru, ko zaradi takšnih podatkov ni bistvenih sprememb v rezultatih.

3.3.2 Transformacija

Pri transformaciji podatkov želimo le-te pretvoriti tako, da bo analiza s podatkovnim rudarjenjem učinkovitejša. Pod transformacijo podatkov na primer spada glajenje podatkov, ki smo ga opisali v prejšnjem razdelku. Pogosta tehnika transformacije podatkov je agregacija atributov, pri čemer gre za to, da na atributih uporabimo zbirne operacije, zato da jih združimo. Primer agregacije atributov je, če atribute o mesečni prodaji seštejemo na letno raven in iz teh ustvarimo nov atribut. Pogost primer transformacije je tudi normalizacija, ki numeričnim podatkom zalogo vrednosti priredi na nek, vnaprej določen, interval $[A, B]$ (najpogosteje je to kar interval $[0, 1]$). Normalizacija je ena od uporabnejših metod pri transformaciji, zato si podrobneje poglejmo njene osnovne tipe.

- **Min-max normalizacija**: Recimo, da imamo atribut, ki ga označimo z Z. Zalogo vrednosti tega atributa želimo preslikati na interval $[A, B]$, pri čemer je $\min_Z$ najmanjša vrednost (minimum) atributa Z, $\max_Z$ pa predstavlja največjo vrednost (maksimum). V tem primeru se vrednost z atributa Z preslika na $z^* = \frac{z-\min_Z}{\max_Z - \min_Z}(B - A) + A$.
- **Z-score normalizacija** (včasih imenovana tudi **standardizacija**): Pri tej vrsti normalizacije gre za normalizacijo na standardno normalno porazdelitev. Če gre za atribut Z, potrebujemo njegovo povprečje μ in standardni odklon σ. Vrednost z atributa Z se preslika na $z^* = \frac{z-\mu}{\sigma}$. Povprečna vrednost normaliziranega atributa je v tem primeru enaka 0, standardni odklon pa 1.

Poleg normalizacije in agregacije je za transformacijo podatkov pogosto v uporabi diskretizacija. Diskretizacija pomeni, da se vrednosti numeričnega atributa zamenjajo

z numeričnimi intervali ali kategoričnimi opisnimi oznakami. Lahko pa gre tudi za združevanje več intervalov v enega. Recimo, intervala $[0, 20]$ ter $[20, 40]$ se lahko "združita" v interval $[0, 40]$. Ta način, kjer vrednosti združujemo, lahko uporabimo tudi na atributih, ki niso numerični. Nekatere kategorične atribute lahko prav tako "agregiramo" tako, da jim zmanjšamo število možnih vrednosti (npr. mesta zamenjamo z državo). Tej metodi pravimo tudi posploševanje ravni.

Obstaja še veliko drugih tehnik predpriprave podatkov, ki jih pa v nalogi nismo uporabili. Ena izmed teh je recimo redukcija podatkovne množice. Tehnike redukcije ali tehnike za zmanjševanja podatkov uporabimo zato, da dobimo zmanjšano zbirko podatkovne množice, ki pa še vedno zajema originalnost podatkov. Podatkovno rudarjenje na zmanjšani množici mora biti učinkovitejše, vendar mora prikazati enake ali podobne rezultate kot na originalni množici podatkov. Ena od tehnik redukcije podatkov je ta, da zmanjšamo število atributov, tako da odstranimo tiste, ki so za določeno analizo irelevantni ali odvečni.

4 KLASIFIKACIJA

Klasifikacija je oblika analize podatkov, pri kateri se gradi modele, imenovane **klasifikatorji** (ang. classifiers), ki napovedujejo kategorične vrednosti **atributa "razred"** (ang. class attribute). Tipičen primer klasifikacije je model iz bančništva, ki prošnje za posojilo kategorizira v varne in tvegane glede na podatke o komitentu. Klasifikacijo se uporablja tudi v ostalih panogah, recimo za odkrivanje goljufij, v marketingu, pri medicinski diagnozi, v proizvodnjah, itd. Klasifikacijske tehnike so večinoma razvite iz področja statistike in strojnega učenja.

Atribut "razred", ki ga pri klasifikaciji napovedujemo, predstavlja kategorično spremenljivko. Klasifikacija je sestavljena iz dveh korakov. Prvi korak imenujemo **učenje** (ang. learning step), drugi korak pa je **klasifikacija** (ang. classification step). V prvem delu se na podlagi klasificiranih podatkov, torej podatkov, ki vsebujejo oznako o "razredu", zgradi model - **klasifikator**. Množico podatkov, ki jo uporabimo za učenje, imenujemo **učna množica** (ang. training set). Primerom, ki so v učni množici, pravimo **učni primeri** (ang. training examples). Vsak učni podatkovni primer je torej označen z vrednostjo atributa "razred" oziroma je klasificiran. V drugem koraku, pri klasifikaciji, klasifikator uporabimo na **testnih primerih** (ang. test examples), s katerimi izračunamo klasifikacijsko točnost. Množici takih primerov pravimo **testna množica** (ang. test set). V testni množici so primeri prav tako klasificirani [8].

Kot smo že omenili, lahko podatkovno množico predstavimo z matriko [12]. Naslednje oznake bomo potrebovali za opisovanje klasifikacijskih algoritmov. Recimo, da imamo m atributov (brez atributa "razred"), ki opisujejo lastnosti primerov in n primerov v podatkovni množici. Podatkovne primere označimo z $x_1, \ldots, x_n$, prvih m atributov pa označimo z $X_1, \ldots, X_m$. V nadaljevanju tega besedila indeks i označuje podatkovni primer, kjer $i \in \{1, \ldots, n\}$, indeks j pa atribut, kjer $j \in \{1, \ldots, m\}$. Naj x_{ij} predstavlja vrednost j-tega atributa za i-ti podatkovni primer. Podatke (brez atributa "razred") lahko torej predstavimo z matriko:

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} & \ldots & x_{1m} \\ x_{21} & x_{22} & x_{23} & \ldots & x_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \ldots & x_{nm} \end{pmatrix}.$$

Vsaka vrstica matrike predstavlja en podatkovni primer. Podatkovni primer x_i je torej predstavljen z m-razsežnim vektorjem atributov, ki ga zapišemo v obliki stolpca. Podatkovni primer x_i je torej enak:

$$x_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{im} \end{pmatrix}.$$

Poleg teh podatkov pa pri klasifikaciji potrebujemo še atribut "razred", ki ga označimo z Y. Vrednost atributa Y za podatkovni primer x_i označimo z y_i, kjer je $i = 1, \ldots, n$. V tem primeru lahko zapišemo novo matriko podatkov, ki je enaka:

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} & \ldots & x_{1m} & y_1 \\ x_{21} & x_{22} & x_{23} & \ldots & x_{2m} & y_2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \ldots & x_{nm} & y_n \end{pmatrix}.$$

Klasifikator si lahko predstavljamo tudi v obliki določene funkcije oziroma preslikave. Recimo, da bi obstajala funkcija f, ki bi imela za vsak podatkovni primer vrednost $f(x_i) = y_i$, torej, da bi vsakemu podatkovnemu primeru pripisala vrednost atributa "razred", ki jo ima ta primer. Naloga klasifikacije je poiskati čim boljšo oceno funkcije f, ki vsakemu primeru x_i pripiše eno vrednost iz zaloge vrednosti atributa Y. Želimo si torej poiskati takšno funkcijo, ki poišče povezavo med vrednostmi atributov v podatkovni množici in atributom Y in je čim bližje f oziroma je v idealnem primeru enaka f. Takšni funkciji pravimo klasifikator.

V nadaljevanju so predstavljeni in podrobneje opisani nekateri klasifikacijski algoritmi, ki so v praksi pogosto uporabljeni in ki smo jih tudi mi uporabili na podatkovni množici iz zavarovalnih polic. Opisani so klasifikator Naivni Bayes, odločitvena drevesa, klasifikacijska pravila ter metoda podpornih vektorjev. Obstaja še veliko ostalih klasifikatorjev, ki pa jih za našo podatkovno množico z zavarovalnimi policami nismo uporabili in jih zato tudi nismo opisovali.

4.1 Naivni Bayesov klasifikator

Naivni Bayes je statistični klasifikator. Cilj tega klasifikatorja je ta, da oceni verjetnost, da nekemu podatkovnemu primeru pripada določena vrednost atributa "razred". Za izpeljavo tega klasifikatorja potrebujemo Bayesovo formulo.

4.1.1 Bayesova formula

Spomnimo se najprej Bayesove formule iz verjetnosti. Poglejmo si najprej izpeljavo osnovne različice formule, ki spada v elementarno verjetnost. Naj bo (Ω, Σ, P) verjetnostni prostor. Naj bosta A in B dogodka. Predpostavimo še, da je $P(A), P(B) > 0$. $P(A)$ in $P(B)$ imenujemo priorni verjetnosti (ang. prior probability), $P(A|B)$ pa pogojna verjetnost dogodka A pri pogoju B, ki ji pravimo tudi posteriorna verjetnost dogodka A pogojno na B. Iz definicije pogojne verjetnosti za takšna dva dogodka vemo, da je:

$$P(A \cap B) = P(A)P(B|A)$$

oziroma

$$P(A \cap B) = P(B)P(A|B).$$

Iz tega sledi, da je:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}.$$

To lahko posplošimo tudi na več dogodkov. Recimo, da imamo zdaj verjetnosti prostor (Ω, Σ, P) in dogodke $B_1, \ldots, B_m$, ki tvorijo particijo prostora Ω, tako da $B_i \cap B_j = \emptyset$, za $i \neq j$ ter $B_1 \cup \ldots \cup B_m = \Omega$. Predpostavimo tudi, da je $P(B_k) > 0$, za $k \in \{1, \ldots, m\}$. Naj bo A dogodek ter $P(A) > 0$. Potem je posteriorna verjetnost dogodka B_k pri pogoju A enaka

$$P(B_k|A) = \frac{P(B_k \cap A)}{P(A)} = \frac{P(B_k)P(A|B_k)}{\sum_{i=1}^{m} P(B_i)P(A|B_i)}, \tag{4.1}$$

kjer smo upoštevali, da je

$$P(A) = P(A \cap B_1) + \ldots + P(A \cap B_m) = \sum_{i=1}^{m} P(B_i)P(A|B_i).$$

Formulo 4.1 imenujemo **Bayesova formula** [25].

4.1.2 Predpostavke Naivnega Bayesa

Bayesovo teorijo lahko uporabimo tudi na podatkovnih primerih, kjer želimo s pomočjo Bayesove formule izračunati verjetnost, da podatkovnemu primeru pripada določena vrednost atributa "razred", če imamo podane ostale vrednosti atributov tega primera. Za izpeljavo Naivnega Bayesovega klasifikatorja smo v tej nalogi uporabili [17, 18]. Spomnimo se, da lahko podatkovno množico opišemo z matematičnimi znaki. Za nadaljevanje najprej dodajmo še nekaj oznak, ki jih potrebujemo pri izpeljavi Naivnega Bayesovega klasifikatorja.

Naj X predstavlja vektor atributov $X_1, \ldots, X_m$. Z Y označimo atribut "razred", ki ima K različnih vrednosti $C_1, \ldots, C_K$. S k označimo indeks vrednosti atributa "razred", kjer $k \in \{1, \ldots, K\}$. Atribute $X_1, \ldots, X_m, Y$ v Naivnem Bayesu obravnavamo kot slučajne spremenljivke. Iz množice podatkov vzamemo en podatkovni primer in ga označimo z x ter ga opišemo z vrednostmi atributov $X_1, \ldots, X_m$, tako da je $x = (x^1, x^2, \ldots, x^m)$. Vrednost atributa X_j za primer x je torej x^j. Dogodek $\{X = x\}$ si razlagamo kot $X_1 = x^1 \wedge X_2 = x^2 \ldots \wedge X_m = x^m$, kjer $\wedge$ predstavlja logični operator in.

Glavni cilj Naivnega Bayesa je na podlagi učne množice zgraditi klasifikator, ki za podatkovni primer x izračuna, za katero vrednost C_k ($k \in \{1, \ldots, K\}$) je ocena verjetnosti

$$P(Y = C_k | X = x) \tag{4.2}$$

maksimalna. Klasifikator napove, da primeru x pripada tista vrednost "razreda", pri kateri je verjetnost 4.2 največja. Eden od načinov, kako oceniti verjetnost $P(Y = C_k|X = x)$, je ta, da se uporabi Bayesovo formulo, nato pa se na podlagi učne množice oceni verjetnosti $P(Y = C_k)$ in $P(X = x|Y = C_k)$. Z uporabo Bayesove formule zapišemo:

$$P(Y = C_k|X = x) = \frac{P(Y = C_k)P(X = x|Y = C_k)}{P(X = x)} = \frac{P(Y = C_k)P(X = x|Y = C_k)}{\sum_{j=1}^{K} P(Y = C_j)P(X = x|Y = C_j)}. \tag{4.3}$$

Učenje izvajamo na podlagi učnih primerov in s pomočjo teh naredimo klasifikator, ki klasificira neoznačene primere. Pravilo za klasifikacijo, ki ga Naivni Bayes uporablja za napovedovanje vrednosti atributa "razred", lahko zapišemo kot:

$$\underset{C_k}{\operatorname{argmax}}\, P(Y = C_k|X = x) = \underset{C_k}{\operatorname{argmax}}\, \frac{P(Y = C_k)P(X = x|Y = C_k)}{\sum_{j=1}^{K} P(Y = C_j)P(X = x|Y = C_j)}. \tag{4.4}$$

Ker je imenovalec v formuli konstantno enak za vsak k, ga lahko izpustimo in klasifikacijsko pravilo zapišemo kot:

$$\underset{C_k}{\operatorname{argmax}}\, P(Y = C_k)P(X = x|Y = C_k). \tag{4.5}$$

V nadaljevanju Naivni Bayes predpostavi, da so vrednosti atributov $X_1, \ldots, X_m$ pogojno neodvisne pri danemu pogoju za Y. To velja za naivno predpostavko in od tu izhaja ime Naivni Bayes. Zaradi te predpostavke lahko verjetnost $P(X = x|Y = C_k)$ zapišemo kot:

$$P(X = x|Y = C_k) = \prod_{j}^{m} P(X_j = x^j|Y = C_k). \tag{4.6}$$

Opazimo, da imamo v enačbi 4.6 "težavo" v zapisu, v primeru, ko je X_j zvezna slučajna spremenljivka. Vemo namreč, da je za poljubno zvezno slučajno spremenljivko Z verjetnost $P(Z = z) = 0$. Za verjetnost $P(X_j = x^j|Y = C_k)$ bomo v primeru, ko je X_j

zvezna spremenljivka, uporabili približek z gostoto verjetnosti, vendar bo način ocenjevanja z gostoto opisan v naslednjem razdelku pri ocenjevanju te verjetnosti in bo zaenkrat ta "težava" v zapisu spregledana. Upoštevamo 4.6 in klasifikacijski problem 4.5 zapišemo kot

$$\underset{C_k}{\operatorname{argmax}} P(Y = C_k)P(X = x|Y = C_k) = \underset{C_k}{\operatorname{argmax}} P(Y = C_k) \prod_{j}^{m} P(X_j = x^j|Y = C_k). \tag{4.7}$$

Za oceno verjetnosti $P(Y = C_k|X = x)$ je potrebno na podlagi učne množice oceniti verjetnosti $P(Y = C_k)$ in $P(X_j = x^j|Y = C_k)$, v primeru ko je X_j diskreten ali zvezni atribut.

4.1.3 Ocenjevanje parametrov z metodo največjega verjetja

Verjetnosti lahko ocenjujemo z metodo največjega verjetja (ang. maximum likelihood estimation). Najprej si poglejmo na kakšen način lahko ocenimo verjetnost $P(Y = C_k)$, pri čemer gledamo na atribut Y kot na diskretno slučajno spremenljivko s K možnimi vrednostmi. Da ocenimo to verjetnost privzamemo porazdelitev spremenljivke Y. V različnih literaturah so navedene različne porazdelitve, s katerimi se oceni to verjetnost. Če je Y kategorična binarna spremenljivka, je smiselno privzeti, da ima Y Bernoullijevo porazdelitev. V splošnejšem primeru ima Y lahko več kot dve možni vrednosti, zato je v tem primeru smiselno privzeti, da ima Y posplošeno Bernoullijevo porazdelitev, ki ji pravimo kategorična porazdelitev [20].

Kategorična porazdelitev opisuje rezultate slučajne spremenljivke, ki lahko zavzema C kategorij ($C > 0, C \in \mathbb{Z}$), pri čemer je za vsako kategorijo posebej določena verjetnost, da spremenljivka zajame to kategorijo. Je poseben primer multinomske porazdelitve. Pri multinomski porazdelitvi opazujemo izide n-tih poskusov, pri čemer je izid vsakega poskusa kategorična spremenljivka. Pri kategorični spremenljivki pa tako kot pri Bernoullijevi opazujemo rezultat enega poskusa.

Ocenjevanje parametrov v posplošeni Bernoullijevi spremenljivki

Poglejmo, kako v primeru kategorične spremenljivke izpeljemo ocene parametrov z metodo največjega verjetja. Recimo, da imamo množico opazovanih vrednosti $z_1, \ldots, z_n$, ki so nastale kot neodvisne enako porazdeljene slučajne spremenljivke $Z_1, \ldots, Z_n$. Spremenljivka Z_i ima kategorično (posplošeno Bernoullijevo) porazdelitev, ki ima K možnih vrednosti $1, \ldots, K$ in velja, da je $P(Z_i = k) = p_k$, kjer je $i = 1, \ldots, n$ ter $\sum_{k=1}^{K} p_k = 1$. Funkcija verjetnosti za spremenljivko Z_i je $P(Z_i = z_i) = p_1^{\mathbb{1}_{\{z_i=1\}}} p_2^{\mathbb{1}_{\{z_i=2\}}} \cdots p_K^{\mathbb{1}_{\{z_i=K\}}}$. Funkcija verjetja je v tem primeru enaka:

$$L(p_1, p_2, \ldots, p_k; z_1, z_2, \ldots, z_n) = \prod_{i=1}^{n} \left(\prod_{k=1}^{K} p_k^{\mathbb{1}_{\{z_i=k\}}} \right) = \prod_{k=1}^{K} p_k^{\sum_{i=1}^{n} \mathbb{1}_{\{z_i=k\}}}.$$

Označimo vsoto $\sum_{i=1}^{n} \mathbb{1}_{\{z_i=k\}} = m_k$. Hitro opazimo, da je $\sum_{k=1}^{K} m_k = n$. Sledi, da je logaritemska funkcija verjetja v tem primeru enaka:

$$l(p_1, p_2, \ldots, p_K; z_1, z_2, \ldots, z_n) = \log L(p_1, p_2, \ldots, p_K; z_1, z_2, \ldots, z_n) = \sum_{k=1}^{K} m_k \log(p_k).$$

Funkcijo l moramo maksimizirati kot funkcijo parametrov $p_1, p_2, \ldots, p_K$. Maksimizacijski problem lahko rešimo z Lagrangeom in sicer tako, da upoštevamo pogoj, da je $\sum_{k=1}^{K} p_k = 1$. Lagrangeovo funkcijo zapišemo kot:

$$F(p_1, p_2, \ldots, p_K, \lambda) = \sum_{k=1}^{K} m_k \log(p_k) + \lambda(\sum_{k=1}^{K} p_k - 1).$$

Odvajamo po p_k in odvod enačimo z 0. Dobimo enačbo $\hat{p_k} = -\frac{m_k}{\lambda}$. Če upoštevamo pogoj, da je $\sum_{k=1}^{K} p_k = 1$, dobimo, da je $\lambda = -n$. Torej je $\hat{p_k} = \widehat{P(Z_i = k)} = \frac{m_k}{n}$ [6].

Vrnimo se na klasifikacijo z Naivnim Bayesom. Kot že omenjeno, lahko pri Naivnem Bayesovem klasifikatorju predpostavimo, da je atribut "razred" kategorična spremenljivka s K možnimi vrednostmi $C_1, \ldots, C_K$. V prejšnjem odstavku smo z metodo največjega verjetja ocenili parametre v kategorični porazdelitvi. To izpeljavo zdaj uporabimo za ocenjevanje verjetnosti $P(Y = C_k)$, pri čemer je Y atribut "razred" oziroma slučajna spremenljivka, ki ima kategorično porazdelitev z vrednostmi $C_1, \ldots, C_K$. Upoštevamo še, da je učna podatkovna množica označena z D in je v njej $|D|$ učnih primerov. Z y_i označimo vrednost atributa "razred" za i-ti podatkovni primer. Ocena verjetnosti $P(Y = C_k)$ je v tem primeru enaka:

$$\widehat{P(Y = C_k)} = \frac{\sum_{i=1}^{|D|} \mathbb{1}_{\{y_i=C_k\}}}{|D|} = \frac{\text{število primerov, ki imajo vrednost atributa "razred" enako } C_k}{|D|}. \tag{4.8}$$

Izpeljavo ocen parametrov kategorične porazdelitve z metodo največjega verjetja pa lahko uporabimo tudi v primeru ocenjevanja verjetnosti $P(X_j = x^j | Y = C_k)$, takrat ko je X_j diskreten atribut. Recimo, da je X_j diskreten atribut in ima t možnih vrednosti, pri čemer je ena od teh vrednosti enaka x^j. V tem primeru predpostavimo, da je pogojno na $\{Y = C_k\}$ spremenljivka X_j porazdeljena posplošeno Bernoullijevo oziroma ima kategorično porazdelitev [20]. V tem primeru analogno s 4.8 velja, da je ocena za $P(X_j = x^j | Y = C_k)$ po metodi največjega verjetja enaka deležu:

$$\frac{\text{število primerov, ki imajo vrednost atributa } X_j \text{ enako } x^j \text{ in vrednost atributa "razred" enako } C_k}{\text{število primerov, ki imajo vrednost atributa "razred" enako } C_k}. \tag{4.9}$$

Lahko pa se zgodi, da v podatkih ni nobenega takega primera, ki bi zadostoval pogojem v števcu ulomka 4.9, oziroma je za nek $k \in \{1, \ldots, K\}$ ocena verjetnosti $P(X_j = x^j | Y = C_k)$ enaka nič. Iz tega sledi, da je ocena verjetnosti $P(Y = C_k | X = x)$

enaka nič, čeprav je ta verjetnost morda v resnici visoka. Ker sklepamo, da je učna množica dovolj velika, lahko ta problem rešimo na ta način, da v tem primeru za vsako vrednost atributa X_j dodamo v množico en izmišljen podatkovni primer in 4.9 se potem spremeni v:

$$\frac{\text{število primerov, ki imajo vrednost atributa } X_j \text{ enako } x^j \text{ in vrednost atributa "razred" enako } C_k + 1}{\text{število primerov, ki imajo vrednost atributa "razred" enako } C_k + t}, \tag{4.10}$$

pri čemer je t velikost zaloge vrednosti diskretne spremenljivke X_j. Temu rečemo Laplaceova korekcija ali Laplaceova ocena.

Oceniti je potrebno še verjetnost $P(X_j = x^j|Y = C_k)$ in sicer v primeru, ko je X_j zvezen atribut oziroma zvezna slučajna spremenljivka. Predpostavimo zdaj, da je X_j zvezna slučajna spremenljivka. V tem primeru lahko predpostavimo, da je pogojno na $\{Y = C_k\}$ spremenljivka X_j porazdeljena normalno s parametri μ_{jk} ter σ_{jk}. Zaradi te predpostavke bomo za izpeljavo Naivnega Bayesovega klasifikatorja predpostavili, da velja naslednje:

$$P(X_j = x^j|Y = C_k) = f(x^j; \mu_{jk}, \sigma_{jk}), \tag{4.11}$$

pri čemer je $f(x^j; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x^j-\mu)^2}{2\sigma^2}}$, kar je gostota slučajne spremenljivke porazdeljene $N(\mu, \sigma)$. Seveda pa ta zapis ni korekten zapis, saj vemo, da je za zvezne spremenljivke verjetnost v točno določeni točki enaka nič. V primeru zveznih spremenljivk vemo, da je potrebno namesto verjetnosti v eni točki $P(X_j = x^j)$ gledati interval $P(x^j < X_j < x^j + \Delta)$ oziroma $P(X_j = x^j) \approx f(x^j; \mu, \sigma) \times \Delta$, za neko majhno konstanto Δ. Vendar ker smo v ulomku v 4.3 opustili imenovalec, je potrebno izvesti normalizacijo, tako da je vsota verjetnosti $P(Y = C_k|X = x)$ po vseh C_k enaka 1. V tem procesu se lahko znebimo konstante Δ in lahko uporabimo zapis v 4.11 [13].

Ko je X_j zvezen atribut, moramo za oceno verjetnosti $P(X_j = x^j|Y = C_k)$ oceniti parametre μ_{jk} ter σ_{jk}. Poglejmo si najprej splošen primer ocenjevanja parametrov z metodo največjega verjetja pri normalni porazdelitvi.

Ocenjevanje parametrov v normalni porazdelitvi

Recimo, da imamo množico opazovanih vrednosti $z_1, z_2, \ldots, z_n$, ki so nastale kot enako porazdeljene neodvisne slučajne spremenljivke $Z_1, Z_2, \ldots, Z_n$, pri čemer je Z_i porazdeljena normalno $N(\mu, \sigma)$. V tem primeru je funkcija verjetja enaka:

$$L(\mu, \sigma; z_1, \ldots, z_n) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(z_i-\mu)^2}{2\sigma^2}}.$$

Logaritemska funkcija verjetja je v tem primeru:

$$l(\mu, \sigma; z_1, \ldots, z_n) = -\frac{n}{2} \log 2\pi - n \log \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (z_i - \mu)^2.$$

Ko odvajamo logaritemsko funkcijo verjetja po μ in σ ter odvoda enačimo z 0, dobimo, da je $\hat{\mu} = \frac{\sum_{i=1}^{n} z_i}{n} = \bar{z}$ in $\hat{\sigma^2} = \frac{1}{n}\sum_{i=1}^{n}(z_i - \bar{z})^2$. V tem primeru pa cenilka za σ^2 ni nepristranska, zato se včasih uporabi cenilko $\hat{\sigma^2} = \frac{1}{n-1}\sum_{i=1}^{n}(z_i - \bar{z})^2$ [6].

Izpeljavo cenilk parametrov v normalni porazdelitvi lahko uporabimo za ocenjevanje verjetnosti v našem klasifikacijskem problemu, v primeru, ko je X_j zvezen atribut. Kot smo že omenili, privzamemo, da je $P(X_j = x^j|Y = C_k) = f(x^j; \mu_{jk}, \sigma_{jk})$. Učna podatkovna množica je enaka $D = \{x_1, \ldots, x_{|D|}\}$ in x_{ij} je vrednost i-tega podatkovnega primera za zvezni podatkovni atribut X_j. Po izpeljavi cenilk parametrov normalne porazdelitve je ocena za μ_{jk} enaka:

$$\widehat{\mu_{jk}} = \frac{\sum_{i=1}^{|D|} x_{ij}\mathbb{1}_{\{y_i=C_k\}}}{\sum_{i=1}^{|D|}\mathbb{1}_{\{y_i=C_k\}}}. \tag{4.12}$$

Nepristranska ocena za σ_{jk}^2 je enaka:

$$\widehat{\sigma_{jk}^2} = \frac{\sum_{i=1}^{|D|}(x_{ij} - \widehat{\mu_{jk}})^2\mathbb{1}_{\{y_i=C_k\}}}{\sum_{i=1}^{|D|}\mathbb{1}_{\{y_i=C_k\}} - 1}. \tag{4.13}$$

Ko imamo vse potrebne ocene, lahko s klasifikacijskim pravilom

$$\underset{C_k}{\operatorname{argmax}}\, P(Y = C_k)\prod_j P(X_j = x^j|Y = C_k)$$

klasificiramo primere v testni množici oziroma primere, ki nimajo določene vrednosti atributa "razred".

Pri Naivnem Bayesu uporabimo predpostavko o pogojni neodvisnosti atributov, vendar vemo, da so v realnih situacijah podatki med sabo navadno korelirani in je predpostavka o pogojni neodvisnosti kršena. Vendar pa s tem, ko upoštevamo pogojno neodvisnost, zmanjšamo število parametrov, ki jih je potrebno oceniti, da dobimo klasifikator. Zaradi tega je lahko Naivni Bayesov klasifikator boljši kot klasifikatorji, ki upoštevajo koreliranost med podatki in morajo zato oceniti večje število parametrov v učni fazi klasifikacije [11].

4.2 Odločitvena drevesa

Odločitvena drevesa so ena od pogostejših orodij, ki se jih uporablja za klasifikacijo in ostale naloge podatkovnega rudarjenja, saj je njihova hierarhična struktura preprosta za razumevanje in odločanje uporabnikov. Odločitveno drevo je v formalnem smislu lahko predstavljeno kot graf, sestavljen iz vozlišč in povezav, ki spominja na strukturo drevesa. Predstavljajmo si torej, da imamo graf $G = (V, E)$, ki je sestavljen iz neprazne, končne množice vozlišč V ter množice povezav E in ima naslednje lastnosti:

(i) povezave predstavljajo urejeni pari vozlišč (v, w), za $v, w \in V$ - graf je usmerjen,

(ii) graf je acikličen,

(iii) obstaja samo eno (vrhnje) vozlišče, ki nima vhodnih povezav in ga imenujemo tudi **začetno ali korensko vozlišče** (ang. root node),

(iv) vsako vozlišče, razen začetnega, ima natanko eno vhodno povezavo,

(v) obstaja enolična pot (zaporedje povezav oblike $(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)$) od začetnega vozlišča do ostalih vozlišč v grafu,

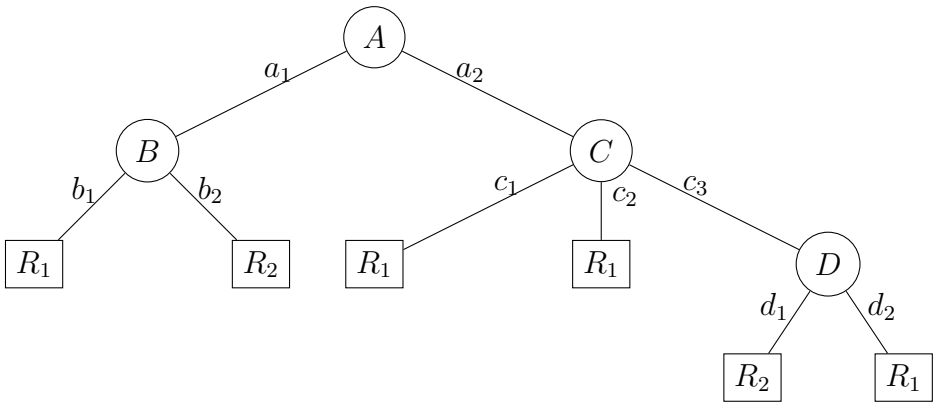
(vi) če obstaja pot iz vozlišča v do vozlišča w, $v \neq w$, potem rečemo, da je vozlišče v pravi prednik vozlišča w (ang. proper ancestor) in vozlišče w pravi potomec vozlišča v (ang. proper descendant),

(vii) vozlišče, ki nima potomcev, se imenuje **končno vozlišče** ali **list** (ang. leaf node). Vsa ostala vozlišča, razen začetnega, se imenujejo **notranja vozlišča** (ang. internal nodes).

V primeru klasifikacije podatkovnih primerov odločitveno drevo deluje tako, da korensko vozlišče ter notranja vozlišča prikazujejo teste oziroma preizkuse za atribute. Povezave, ki gredo iz nekega vozlišča, pa prikazujejo izide preizkusa v tem vozlišču. Vsako končno vozlišče ali list vsebuje vrednost atributa "razred". Korensko vozlišče predstavlja prvi test v odločitvenem drevesu. Če imamo zgrajeno odločitveno drevo na podlagi klasificiranih podatkov in podan podatkovni primer x, ki je predstavljen z vektorjem atributov, potem vrednosti atributov za primer x testiramo na zgrajenem odločitvenem drevesu. Pot sledi iz korenskega v končno vozlišče, ki vsebuje napoved "razreda" za x [1].

Na sliki 2 je enostaven primer odločitvenega drevesa, kjer ima atribut "razred" dve vrednosti in sicer R_1 ter R_2. V drevesu testiramo vrednosti štirih atributov, ki so označeni z A, B, C in D. Korensko vozlišče je označeno z atributom A, ki ima vrednosti a_1 in a_2, notranja vozlišča pa so označena z atributi B, C in D. Končna vozlišča vsebujejo vrednosti atributa "razred".

V zadnjih treh desetletjih je bilo opisanih veliko načinov kako zgraditi čim bolj zanesljivo odločitveno drevo v čim krajšem času, vendar pa se v literaturi najpogosteje navaja rekurzivni postopek nastajanja odločitvenega drevesa od vrha do dna (ang. top-down induction).



Slika 2: Enostavno odločitveno drevo

4.2.1 Generiranje odločitvenih dreves (algoritma ID3 in C4.5)

Eden od pomembnejših raziskovalcev na področju strojnega učenja Ross J. Quinlan je v sedemdesetih in osemdesetih letih prejšnjega stoletja razvil algoritem za odločitvena drevesa, znan pod imenom ID3 (Iterative Dichotomiser 3). Kasneje je predstavil še algoritem C4.5, ki je naslednik ID3 algoritma. C4.5 je postal eden od pogosteje uporabljenih algoritmov in merilo, s katerim se pogosto primerjajo novejši algoritmi za nadzorovano učenje. Nato je v letu 1984 skupina statistikov objavila knjigo Klasifikacijska in regresijska drevesa, v kateri so opisana binarna odločitvena drevesa, znana pod kratico CART, ki je okrajšava za *Classification And Regression Trees*. Binarno odločitveno drevo je tisto, pri katerem gredo iz vsakega vozlišča (razen iz končnih vozlišč) dve povezavi. Odločitvena drevesa CART so nastala neodvisno od ID3 algoritma, vendar za učenje iz podatkovne množice uporabljajo podoben pristop kot ID3.

V nadaljevanju bo besedilo osredotočeno na algoritma ID3 in C4.5. Najprej opišemo in navedemo splošen princip delovanja osnovnega algoritma za izgradnjo odločitvenega drevesa (ID3) [8]. Nato opišemo razlike med algoritmoma ID3 in C4.5 in navedemo na kakšen način C4.5 nadgradi osnovni algoritem [7, 15].

ID3 in C4.5 uporabljata rekurzivni pristop gradnje odločitvenega drevesa od vrha do dna z načinom deli ali vladaj (ang. top-down recursive divide and conquer manner). Temu pravimo tako, ker imamo na začetku algoritma celotno množico učnih primerov in tem pripadajoče vrednosti atributa "razred". Množica primerov se znotraj algoritma rekurzivno deli na manjše podmnožice in na ta način se gradi odločitveno drevo.

Vhod algoritmov za generiranje odločitvenih dreves (ID3 in C4.5) vsebuje tri parametre. Prvi parameter je množica učnih primerov, ki jo označimo z $D = \{x_1, \ldots, x_{|D|}\}$. Vsak od teh primerov je predstavljen z vrednostmi atributov $X_1, X_2, \ldots, X_m$ in z atributom "razred", ki je tudi v tem primeru označen z Y. Atribut Y je diskreten in ima vrednosti C_k za $k = 1, \ldots, K$. Atributi $X_1, \ldots, X_m$ so lahko zvezni ali diskretni glede na to, kakšne so njihove vrednosti. Kasneje bo opisano, kakšen način delitve podatkovne

množice velja pri enih in drugih atributih. Algoritem ID3 še ni bil pripravljen za delo z zveznimi atributi, je pa to možno z novejšim algoritmom C4.5. Drugi parameter vhoda je seznam atributov, ki opisujejo podatkovne primere. Tretji parameter je **metoda izbire atributa** (and. attribute selection method), ki predstavlja proces, ki določi **delitveni kriterij** (ang. splitting criterion) na podatkovni množici. Delitveni kriterij pove, glede na kateri atribut je potrebno razdeliti množico učnih primerov, tako da dobimo "najboljšo" particijo te množice. Takšnemu atributu pravimo **delitveni atribut** (ang. splitting attribute). Delitveni atribut je določen tako, da učno množico razdeli v čim bolj "čiste" (ang. pure) podatkovne podmnožice. V idealnem primeru je "čista" podmnožica tista, ki vsebuje samo primere z enako vrednostjo atributa "razred". Delitveni kriterij določi tudi, katere povezave naj nastanejo pri deljenju učne množice, glede na delitveni atribut. V primeru zveznih atributov mora delitveni kriterij, poleg delitvenega atributa, navesti tudi **delitveno točko** (ang. splitting point). Če je metoda izbire atributa uporabljena na množici podatkov D in na seznamu atributov X, to označimo kot MetodaIzbireAtributa(D, X). Metoda za izbiro delitvenega atributa uporablja neko mero, s katero poišče najboljši delitveni atribut. V nadaljevanju sta opisani dve meri, ki sta poznani pod izrazoma **informacijski prispevek** (ang. information gain) in **razmerje informacijskega prispevka** (ang. information gain ratio ali krajše gain ratio). Najprej bo opisana osnovna verzija algoritma za generiranje odločitvenih dreves oziroma algoritem ID3. Ko je algoritem uporabljen na množici podatkov D in seznamu atributov X to označimo z ID3(D, X). Ko imamo vse potrebne parametre, ki jih algoritem uporablja, lahko začnemo graditi odločitveno drevo. Začne se tako, da ustvarimo prazno vozlišče, ki ga poimenujemo M. Algoritem deluje rekurzivno, zato ima na začetku dva zaustavitvena kriterija.

1. Prvi zaustavitveni kriterij pravi, da če imajo vsi primeri v učni množici D enako vrednost atributa "razred", potem naj vozlišče M postane list in ta naj bo označen z vrednostjo "razreda", ki jo imajo vsi primeri v D.

2. Drugi zaustavitveni kriterij pa pravi, da če je množica atributov prazna, potem naj vozlišče M postane list, ki je označen z večinsko vrednostjo atributa "razred" v množici primerov D.

V kolikor ni izpolnjen noben od teh pogojev, mora algoritem uporabiti metodo izbire atributa, ki določi delitveni kriterij na dani učni množici. Kot smo že omenili, metoda izbire atributa (na učni množici D) delitveni kriterij določi tako, da so podmnožice podatkovnih primerov, ki nastanejo, čim bolj "čiste" glede na vrednosti atributa "razred". Želimo, da delitveni kriterij deluje tako, da bi v idealnem primeru naredil "popolno" particijo učne množice, kar pomeni, da bi vsi podatkovni primeri znotraj podmnožic,

ki bi nastale, pripadali enaki vrednosti atributa "razred". Ko metoda izbire atributa vrne delitveni atribut, vozlišču z oznako M spremenimo oznako v naziv delitvenega atributa. Iz tega vozlišča naredimo povezave v odločitvenem drevesu, ki predstavljajo izide testa pri tem delitvenem atributu. V nadaljevanju (Algoritem 1) je navedena osnovna psevdokoda za algoritem ID3.

Algoritem 1: Osnovni algoritem za generiranje odločitvenega odrevesa (ID3)

Vhod: Množica učnih primerov D, Seznam atributov, Metoda izbire atributa.
Izhod: Odločitveno drevo M.

```
Ustvari vozlišče M
če primeri v D imajo vsi enako vrednost atributa "razred" (= C_k) potem
    Vrni M kot končno vozlišče označeno s C_k;
če seznam atributov je prazen potem
    Vrni M kot končno vozlišče, označeno z večinsko vrednostjo "razreda" v D;
Uporabi MetodaIzbireAtributa(D, Seznam atributov) in poišči najboljši
 delitveni kriterij;
Označi vozlišče M z delitvenim kriterijem;
če delitveni atribut je diskretna spremenljivka in drevo ni binarno potem
    Seznam atributov ← Seznam atributov - delitveni atribut;
za vsak izid i delitvenega kriterija
    Ustvari novo povezavo (poddrevo) iz vozlišča M in naj bo D_i podmnožica D,
     ki ustreza i-temu izidu delitvenega kriterija
    če množica D_i je prazna potem
        Pod novonastalo povezavo za i-ti izid naredi končno vozlišče in ga označi
         z večinsko vrednostjo atributa "razred" v množici D;
    sicer
        Pod novonastalo povezavo za i-ti izid uporabi ID3(D_i, Seznam atributov);
vrni M;
```

To je osnovna verzija algoritma za generiranje odločitvenih dreves. Algoritem C4.5 predstavlja nadgradnjo osnovnega algoritma. Prva razlika med osnovnim algoritmom ID3 in algoritmom C4.5 je ta, da je algoritem ID3 znal klasificirati samo primere z diskretnimi atributi, algoritem C4.5 pa zna klasificirati primere z diskretnimi in zveznimi atributi. Spodaj je opisano, na kakšen način algoritem gradi odločitveno drevo, glede na to, kakšne vhodne atribute ima. Recimo, da je na učni množici D algoritem za delitveni atribut izbral X_j za nek $j \in \{1, \ldots, m\}$.

Poglejmo kakšne povezave nastanejo iz vozlišča M, ki je bil označen z delitvenim atributom X_j, če je atribut zvezen ali diskreten.

1. Če je atribut X_j diskreten, so izidi testa v vozlišču M vse možne vrednosti atributa X_j. Recimo še, da je t število različnih vrednosti atributa X_j. Za vsako vrednost tega atributa je ustvarjena nova povezava iz vozlišča M in je označena s to vrednostjo. Učna množica D se v tem primeru razdeli na podmnožice $D_1, \ldots, D_t$, tako da za vsako vrednost atributa X_j nastane ena podmnožica množice D. Ker imajo v podmnožicah vsi podatkovni primeri enako vrednost za atribut X_j, je X_j po delitvi glede na njegove vrednosti izbrisan iz seznama atributov.

2. Če je X_j zvezni atribut, metoda izbire atributa kot del delitvenega kriterija vrne tudi delitveno točko in v tem primeru ima test v vozlišču M dva možna izida oziroma iz tega vozlišča nastaneta dve povezavi. Prva povezava predstavlja pogoj $X_j \leq$ delitvena točka, druga pa pogoj $X_j >$ delitvena točka. Učna množica D se posledično razdeli v dve podmnožici, D_1 in D_2.

4.2.2 Teorija informacij

Mere za izbor delitvenega atributa, ki so uporabljene v algoritmu za odločitvena drevesa slonijo na teoriji informacij [23], ki je v nadaljevanju na kratko opisana.

Glavni mejnik na področju teorije informacij je postavil Claude Elwood Shannon, ki je deloval na področju informacijske teorije in je leta 1948 izdal članek z naslovom Matematična teorija komunikacij (ang. A Mathematical Theory of Communication). Problem, s katerim se je v članku ukvarjal, je bilo ugotavljanje izgube podatkov pri prenosu določenega sporočila od ene točke do druge. V svojem delu Shanon ni upošteval pomena teh sporočil, vendar je na to gledal zgolj iz tehničnega vidika. Definiral je pomembno količino, ki jo imenujemo Shanonova entropija in jo bomo uporabili tudi kot merilo informacij pri odločitvenem drevesu. Poglejmo si na kakšen način je to definiral. Recimo, da imamo diskretno slučajno spremenljivko X s K možnimi vrednostmi $\{1, \ldots, K\}$. S p_k označimo verjetnost $P(X = k)$, $k = 1, \ldots, K$. Seveda velja, da je $p_k \geq 0$ in $\sum_{k=1}^{K} p_k = 1$. Te verjetnosti so nam znane, vendar je to vse, kar vemo o tem, kateri dogodek se bo zgodil pri opazovanju slučajne spremenljivke. Shannona je zanimalo ali obstaja mera informacije, ki bi opisala, kolikšna je negotovost glede izida v takšnem primeru. Predstavil je mero $H(p_1, p_2, \ldots, p_K)$, ki v opisanem primeru zadošča naslednjim lastnostim.

1. H je zvezna funkcija v odvisnosti od p_k.

2. Če so vsi izidi enako verjetni, oziroma če je $p_k = \frac{1}{K}$ za $k = 1, \ldots, K$, potem je $H(\frac{1}{K}, \ldots, \frac{1}{K})$ monotono naraščajoča funkcija K-ja.

3. Primer, ki ponazarja tretjo lastnost, je naslednji: recimo, da imamo slučajno spremenljivko Y z vrednostmi v $\{0, 1, 2\}$ in $P(Y = 0) = \frac{1}{2}$, $P(Y = 1) = \frac{1}{4}$, $P(Y = 2) = \frac{1}{4}$. Predstavljajmo si, da to slučajno spremenljivko predstavimo z metom kovanca, tako da najprej vržemo kovanec in določimo, ali je $Y = 0$ ali ni. Če $Y \neq 0$, potem vržemo kovanec še enkrat, da določimo, ali je $Y = 1$ ali $Y = 2$. V tem primeru za H zahtevamo, da je $H(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}) = H(\frac{1}{2}, \frac{1}{2}) + \frac{1}{2}H(\frac{1}{2}, \frac{1}{2})$. Desna stran enačbe ponazarja met kovanca v dveh korakih. Najprej mečemo nepristranski kovanec, da določimo, ali je $Y = 0$ ali $Y \neq 0$. Če $Y \neq 0$, ponovno z metom nepristranskega kovanca določimo ali je $Y = 1$ ali $Y = 2$, vendar to samo v primeru, ko je $Y \neq 0$, to pa se zgodi z verjetnostjo $\frac{1}{2}$. Primer je povzet po [16].

   V splošnem torej za H velja, da če nek izid predstavimo kot dva zaporedna izida, potem mora biti originalen H enak uteženi vsoti po vseh posameznih H-jih.

Shannon je v [23] pokazal, da je edina funkcija, ki ustreza zgornjim predpostavkam, oblike

$$H(X) = -C \sum_{i=1}^{K} p_i \log(p_i), \tag{4.14}$$

kjer je C pozitivna konstanta. Funkcija $H(X)$ je enolično določena do konstante natančno, kjer je C določena z izbiro merske enote. Izbira logaritemske osnove določa enoto za mero informacij. Če določimo, da je logaritemska osnova enaka 2, je v tem primeru merska enota za količino informacije imenovana bit. V nadaljevanju bodo vsi logaritmi imeli osnovo 2. Količino

$$H(X) = -\sum_{i=1}^{K} p_i \log(p_i) \tag{4.15}$$

imenujemo Shannonova entropija in jo lahko označimo tudi kot $H(p_1, \ldots, p_K)$. Funkcije oblike 4.15 igrajo pomembno vlogo v teoriji informacij in jih lahko uporabimo tudi kot mero informacij pri odločitvenih drevesih.

4.2.3 Teorija informacij in algoritmi za odločitvena drevesa

Vrnimo se k algoritmom za odločitvena drevesa in sicer k ID3 ter C4.5, katerih pomemben del je metoda izbire atributa. Za ta del uporabljamo osnove teorije informacij. Algoritem ID3 za mero izbire delitvenega atributa uporablja količino **informacijski prispevek** (ang. information gain), algoritem C4.5 pa to mero še nekoliko nadgradi in uporablja **razmerje informacijskega prispevka** (ang. information gain ratio). Izračun teh dveh mer sloni na Shannonovi teoriji informacij.

Poglejmo, na kakšen način ID3 uporablja mero informacijski prispevek za določanje delitvenega atributa. Recimo, da vozlišče M vsebuje vse primere iz množice učnih podatkov D. Atribut, ki ima največji informacijski prispevek pri tem vozlišču, je izbran kot delitveni atribut za to vozlišče. Atribut je izbran tako, da minimizira količino informacij, ki je potrebna za klasifikacijo podatkovnih primerov po delitvi le-teh v podmnožice glede na izbrani atribut. V povzetku Shannonove teorije informacij smo pokazali, koliko je pričakovana vrednost informacije diskretne porazdelitve. To mero informacije oziroma entropijo lahko uporabimo v algoritmu za odločitvena drevesa, tako da za diskretno slučajno spremenljivko vzamemo atribut "razred" (Y) z vrednostmi $C_1, \ldots, C_K$. Verjetnost $P(Y = C_k)$ označimo s p_k, kjer je $k = 1, \ldots, K$. Pričakovano vrednost informacije, ki je potrebna, da klasificiramo učni podatkovni primer v množici D, dobimo po formuli 4.15 in jo označimo kot Info(D). Velja:

$$\text{Info}(D) = H(Y) = H(p_1, \ldots, p_K) = -\sum_{k=1}^{K} p_k \log_2 p_k. \tag{4.16}$$

Za verjetnosti p_k se spomnimo ocene iz Naivnega Bayesa, pri kateri smo uporabili metodo največjega verjetja. Enako oceno lahko uporabimo v tem primeru in tako upoštevamo, da je:

$$\widehat{P(Y = C_k)} = \frac{\sum_{i=1}^{|D|} \mathbb{1}_{\{y_i = C_k\}}}{|D|} = \frac{\text{število primerov, ki imajo vrednost atributa "razred" enako } C_k}{|D|}.$$

Zanima nas, kateri atribut minimizira pričakovano količino informacije, ki je potrebna za klasifikacijo podatkovnih primerov v podmnožicah množice D, ki nastanejo z deljenjem glede na vrednosti tega atributa.

Recimo, da imamo atribut X_j (za nek $j \in \{1, \ldots, m\}$), ki je diskreten in ima t možnih vrednosti. Pričakovana količina informacije, ki jo potrebujemo za klasifikacijo nekega primera, po tem ko množico D razdelimo na podmnožice $D_1, \ldots, D_t$ po atributu X_j, je enaka

$$\text{Info}_{X_j}(D) = \sum_{i=1}^{t} \frac{|D_i|}{|D|} \times \text{Info}(D_i), \tag{4.17}$$

pri čemer $|D|$ predstavlja število primerov v D ter $|D_i|$ število primerov v D_i, kjer $i = 1, \ldots, t$. Info(D_i) je količina, ki jo izračunamo po formuli 4.16, le da v tem primeru upoštevamo le primere v podmnožici D_i. Razlika med količino informacije, ki jo potrebujemo za klasifikacijo pred delitvijo z atributom X_j in količino informacije, ki jo potrebujemo, ko D razdelimo, se imenuje **informacijski prispevek** (ang. information gain) in je enaka:

$$\text{Gain}(X_j) = \text{Info}(D) - \text{Info}_{X_j}(D). \tag{4.18}$$

Iščemo takšen atribut X_j med atributi $X_1, \ldots, X_m$, ki minimizira 4.17, oziroma, ki maksimizira 4.18. Iščemo torej:

$$\underset{X_j}{\operatorname{argmax}} \operatorname{Gain}(X_j) = \underset{X_j}{\operatorname{argmax}} (\operatorname{Info}(D) - \operatorname{Info}_{X_j}(D)).$$

Do zdaj smo opisovali le izračun informacijskega prispevka v primeru, ko je bil atribut diskreten. Algoritem C4.5 se od ID3 razlikuje tudi po tem, da zna delati tudi z zveznimi atributi. Recimo, da je X_j (za nek $j \in (1, \ldots, m)$) zvezen atribut. V tem primeru poteka izračun informacijskega prispevka $\operatorname{Gain}(X_j)$ na ta način, da je potrebno določiti, katera vrednost tega atributa je najboljša delitvena točka. To se lahko določi tako, da se vrednosti atributa X_j v podatkih najprej razvrsti od najmanjše do največje. Delitvene točke so navadno oblike $\frac{x^j + x^{j+1}}{2}$, kjer sta x^j in x^{j+1} zaporedni vrednosti atributa X_j. Če ima X_j v podatkih t različnih numeričnih vrednosti, potem je $(t-1)$ možnih delitvenih točk. Za vsako možno delitveno točko je potrebno izračunati $\operatorname{Info}_{X_j}(D)$. To naredimo na ta način, da množico D razdelimo na podmnožici D_1 in D_2, pri čemer D_1 predstavlja tiste primere v D, za katere je vrednost atributa $X_j \leq$ delitvena točka, D_2 pa predstavlja tiste primere, za katere je $X_j >$ delitvena točka. Za vsako možno delitveno točko izračunamo $\operatorname{Info}_{X_j}(D)$ in izberemo, da je najboljša delitvena točka tista, ki da minimalen $\operatorname{Info}_{X_j}(D)$.

Kot smo že omenili, algoritem ID3 za metodo izbire delitvenega atributa uporablja informacijski prispevek. Vendar pri uporabi te količine lahko opazimo pomanjkljivost v primerih, ko imamo atribute oziroma spremenljivke z veliko možnimi izidi. Pomanjkljivost pa je ta, da ta mera raje izbere tiste atribute, ki imajo veliko število izidov. Za primer poglejmo atribut, ki predstavlja zaporedno številko podatkovnega primera. Če bi naredili particijo učne množice glede na ta atribut, bi dobili veliko podmnožic (toliko kot je primerov v množici podatkov, s katero delamo) in v vsaki podmnožici bi bil zgolj en primer, ki pripada določeni zaporedni številki. Vsaka podmnožica, ki bi nastala z delitvijo glede na takšen atribut, bi bila v tem primeru popolnoma "čista", kar pomeni, da ne bi potrebovali nobene informacije več za klasifikacijo primerov v danih podmnožicah. To bi privedlo do tega, da bi takšen atribut imel najvišji informacijski prispevek. Jasno pa je, da je izbira takšnega delitvenega atributa nekoristna za klasifikacijo. Zaradi tega problema algoritem C4.5 uporablja popravljeno različico informacijskega prispevka 4.18, ki jo imenujemo **razmerje informacijskega prispevka** (ang. gain ratio). Razmerje informacijskega prispevka uporabi dodatno količino, definirano kot:

$$\operatorname{SplitInfo}_{X_j}(D) = -\sum_{i=1}^{t} \frac{|D_i|}{|D|} \times \log\left(\frac{|D_i|}{|D|}\right). \tag{4.19}$$

Ta vrednost je definirana na analogen način kot 4.16. Pove nam količino informacije, generirano s tem, ko učno množico D razdelimo v t podmnožic glede na t izidov,

ki nastanejo s testiranjem po atributu X_j. Da bi dobili smiselno količino, s katero bi določili, kateri atribut je najboljši delitveni atribut, 4.18 delimo s 4.19 in dobimo razmerje informacijskega prispevka, ki je enako:

$$\text{GainRatio}(X_j) = \frac{\text{Gain}(X_j)}{\text{SplitInfo}_{X_j}(D)}.$$

Med vsemi atributi iščemo tistega, ki doseže maksimalno vrednost oziroma iščemo:

$$\underset{X_j}{\text{argmax}}\, \text{GainRatio}(X_j) = \underset{X_j}{\text{argmax}}\, \frac{\text{Gain}(X_j)}{\text{SplitInfo}_{X_j}(D)}.$$

4.2.4 Manjkajoče vrednosti pri odločitvenih drevesih

Vemo, da so manjkajoče vrednosti pogost pojav v podatkovnih množicah in lahko vplivajo na rezultat klasifikatorja. Algoritem ID3 zna delati samo z množicami podatkov, ki ne vsebujejo manjkajočih vrednosti. Algoritem C4.5 pa zna upoštevati tudi manjkajoče vrednosti in to naredi na ta način, da nekoliko popravi formule za $\text{Gain}(X_j)$ ter $\text{SplitInfo}_{X_j}(D)$. Recimo, da imamo množico podatkov D in recimo, da X_j predstavlja potencialen test za to množico oziroma atribut, po katerem se ta množica razdeli glede na vrednosti tega atributa. D_0 naj predstavlja množico tistih primerov, ki imajo pri atributu X_j neznano vrednost. $D - D_0$ predstavlja množico primerov D, brez množice D_0. $\text{Gain}(D - D_0, X_j)$ je informacijski prispevek za atribut X_j, ki je izračunan na množici podatkov $D - D_0$. Tega pomnožimo z deležem $\frac{|D-D_0|}{|D|}$ in dobimo formulo za $\text{Gain}(X_j)$, ki je enaka:

$$\text{Gain}(X_j) = \frac{|D - D_0|}{|D|}\text{Gain}(D - D_0, X_j).$$

Formula za $\text{SplitInfo}_{X_j}(D)$ pa se v tem primeru spremeni v:

$$\text{SplitInfo}_{X_j}(D) = -\frac{|D_0|}{|D|}\log\left(\frac{|D_0|}{|D|}\right) - \sum_{i=1}^{t}\frac{|D_i|}{|D|} \times \log\left(\frac{|D_i|}{|D|}\right).$$

Obe formuli sta spremenjeni na ta način, da če ima nek atribut velik delež primerov z manjkajočimi vrednostmi, bo težje izbran kot delitveni atribut. Ko je atribut X_j z manjkajočimi vrednostmi izbran za delitveni atribut na množici D, za množico primerov z manjkajočimi vrednostmi ne naredimo posebne veje v odločitvnem drevesu, ampak se te primere vključi med ostale. Recimo, da dobimo z delitvijo množice D po atributu X_j podmnožice $D_1, \ldots, D_t$. V tem primeru je k vsaki od teh podmnožic D_i dodan delež primerov iz D_0 in sicer z utežjo $\frac{|D_i|}{|D-D_0|}$.

4.2.5 Rezanje dreves

Ko algoritem gradi odločitveno drevo, se lahko zgodi, da se preveč prilagodi dani učni množici (ang. overfitting of the data). Takšna odločitvena drevesa so lahko neučinkovita pri klasifikaciji primerov iz testne množice, saj so v njih vsebovani tudi šum in napake v podatkih iz učne množice. Ta problem rešujemo z metodami rezanja odločitvenih dreves. Porezana drevesa so ponavadi manjša in lažja za razumevanje ter imajo navadno pri klasifikaciji testnih podatkov boljši rezultat kot neporezana drevesa. Obstajata dva načina za rezanje odločitvenih dreves. En način je rezanje dreves v učni fazi, ko se drevo gradi, na ta način, da se glede na nek kriterij odločimo, da množice podatkov ne delimo dalje in množica podatkov postane končno vozlišče. Na primer, lahko bi vzeli kriterij velikosti podmnožic v particiji učne množice. Ko bi podmnožice vsebovale manj podatkovnih primerov, kot je željeno, bi zaključili z delitvijo in bi vozlišče, v katerem se nahajajo podatki, označili z najpogostejšo vrednostjo atributa "razred". Lahko pa se uporabi tudi kakšen drugi kriterij, ki pove, v katerem primeru se množice podatkov deli dalje ali ne. Ta način rezanja odločitvenih dreves imenujemo **predhodno rezanje** (ang. pre-pruning).

Drugi, pogostejši način rezanja, se imenuje **naknadno rezanje** (ang. post-pruning). Pri tem načinu algoritem drevo ustvari do konca in se šele nato vrne k rezanju. Drevo se v tem primeru reže na ta način, da se odstranjuje in/ali zamenjuje poddrevesa. ID3 algoritem se še ni ukvarjal z rezanjem dreves, algoritem C4.5 pa uporablja **metodo pesimističnega rezanja** (ang. pessimistic pruning), ki drevo reže po tem, ko je že do konca zgrajeno. Pri tej metodi se ocenjuje stopnjo napake (ang. error rate estimate) na učni množici in se na podlagi le-te odloča ali se neko poddrevo odstrani (zamenja z listom ali manjšim poddrevesom) ali ne.

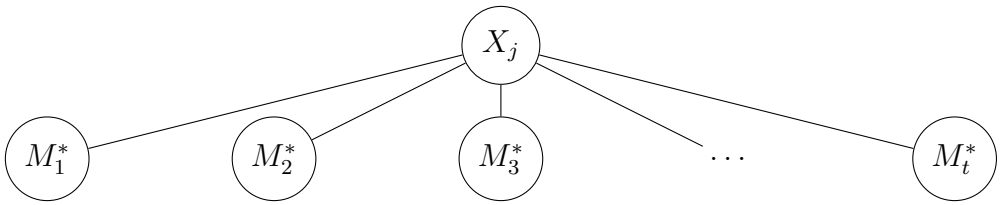
Recimo, da je M klasifikator, ki ga dobimo z algoritmom C4.5 na učni množici D. Recimo, da klasifikator M narobe klasificira F primerov iz učne množice. Stopnja napake klasifikatorja M, izračunana na učni množici, je torej ocenjena na $\frac{F}{|D|}$ (to bomo videli tudi v poglavju klasifikacijska točnost). Točna vrednost napake klasifikatorja se izračuna na podatkih iz celotne populacije iz katere smo vzeli vzorec D. Točna vrednost napake je navadno precej višja od napake izračunane na učni množici D, ki je velikokrat tudi blizu 0, ko klasificiramo z neporezanim drevesom. Ker navadno nimamo podatkov o celotni populaciji, se stopnjo napake računa s podatki, ki niso bili uporabljeni pri učenju. To je najboljši način za izračun ocene, če imamo takšne podatke na voljo. Velikokrat pa se zgodi, da je množica podatkov, s katero delamo, majhna in je potrebno vse podatke, ki so nam na voljo, uporabiti že pri učenju in potem napako računati na istih podatkih. Pri rezanju dreves z metodo pesimističnega rezanja se stopnjo napake računa na učni množici, vendar algoritem C4.5 poskrbi, da to oceno

nekoliko "popravi" in se tako izogne njeni pristranskosti. Najprej si pa poglejmo nekaj statistične teorije, ki nam je pri tem v pomoč.

Recimo, da imamo množico opazovanih vrednosti $z_1, \ldots, z_n$, ki so nastale kot neodvisne enako porazdeljene slučajne spremenljivke $Z_1, \ldots, Z_n$, kjer so vse $Z_i \sim Ber(p)$. Označimo $N = \sum_{i=1}^{n} z_i$. Pri Naivnem Bayesu smo pokazali kako lahko parameter p ocenimo z metodo največjega verjetja. Cenilka za p je v tem primeru enaka $\hat{p} = \frac{N}{n}$. Iz statistike pa vemo, da lahko za parameter p določimo tudi interval zaupanja, pri neki stopnji značilnosti α. Interval zaupanja za p je enak $p = \hat{p} \pm \Phi^{-1}(1 - \frac{\alpha}{2})\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$ [26]. Zgornjo mejo intervala zaupanja za p označimo kot $U_\alpha(N, n)$.

To lahko zdaj uporabimo pri ocenjevanju stopnje napake klasifikatorja M na učni množici podatkov D. Na to lahko pogledamo tudi iz vidika verjetnosti. S p označimo verjetnost, da klasifikator zgreši pri napovedi "razreda" za nek podatkovni primer v podatkovni množici. Kot je bilo že omenjeno, lahko za klasifikator M, ki zgreši v F od $|D|$ primerov, verjetnost p ocenimo z $\frac{F}{|D|}$. Vemo pa, da je ta način ocenjevanja napak klasifikatorja pristranski, saj je M naučen s pomočjo učne množice D, zato za boljšo oceno stopnje napake klasifikatorja M na učni množici uporabimo zgornjo mejo intervala zaupanja za p. V Weki lahko nastavimo različne vrednosti za α, privzeta vrednost pa je 0.25. S pomočjo tega potem drevo režemo na naslednji način:

Recimo, da imamo odločitveno drevo M, ki je nastalo po algoritmu C4.5 na podlagi učne podatkovne množice D. Rezanje odločitvenega drevesa poteka od spodaj navzgor. M naj bo odločitveno drevo z več kot enim vozliščem in naj bo oblike kot odločitveno drevo na sliki 3.



Slika 3: Postopek rezanja odločitvenih dreves

Naj bo X_j za nek $j \in \{1, \ldots, m\}$ atribut, ki razdeli množico podatkov D na t podmnožic, tako da nastanejo poddrevesa $M_1^*, M_2^*, \ldots, M_t^*$. Recimo, da so poddrevesa $M_1^*, M_2^*, \ldots, M_t^*$ že porezana. Poddrevo M_i^* (za nek $i \in \{1, \ldots, t\}$) naj bo drevo, ki mu pripada največ podatkovnih primerov, glede na delitev po atributu X_j. Naj bo L vozlišče označeno z najpogostejšo vrednostjo atributa "razred" v množici D. Zdaj lahko pogledamo kaj se zgodi, če podatkovne primere v D klasificiramo glede na tri različne možnosti. Podatkovne primere v D lahko klasificiramo s celotnim drevesom M in število zgrešenih klasifikacij označimo z E_M, lahko jih klasificiramo tudi glede na poddrevo M_i^* in dobimo $E_{M_i^*}$ napak ali pa jih klasificiramo kar z najpogostejšo

vrednostjo atributa "razred", ki je označena na vozliču L in dobimo E_L napak. Kot smo v prejšnjem razdelku zapisali, lahko ocene za stopnje napak v vseh treh primerih nadomestimo z ocenami $U_\alpha(E_M, |D|)$, $U_\alpha(E_{M_i^*}, |D|)$ ter $U_\alpha(E_L, |D|)$. Ocene med sabo primerjamo in glede na velikosti teh določimo, kako porežemo drevo M. Seveda želimo, da je ocena napake čim manjša, zato v primeru, ko je $U_\alpha(E_M, |D|)$ najmanjša od teh vrednosti, drevo M ne režemo, ko je $U_\alpha(E_{M_i^*}, |D|)$ najmanjša, drevo M zamenjamo s poddrevesom M_i^* in ko je $U_\alpha(E_L, |D|)$ najmanjša, drevo M zamenjamo s končnim vozliščem L.

Opazimo lahko, da ima algoritem C4.5 kar nekaj pomembnih izboljšav algoritma ID3. Algoritem C4.5 je implementiran v orodju Weka, ki predstavlja odprtokodno programsko opremo za strojno učenje. Ker je koda za algoritem C4.5 zapisana v Javi, se v Weki imenuje J48. Avtorji orodja Weka so algoritem C4.5 opisali kot mejnik algoritmov za generiranje odločitvenih dreves.

4.2.6 Klasifikacijska pravila in algoritem PART

Za klasifikacijo so pogosto uporabljena tudi klasifikacijska pravila ČE-POTEM (ang. IF-THEN rules), ki so oblike

ČE pogoj **POTEM** zaključek.

"ČE" delu pravila na levi strani pravimo **predpogoj** (ang. precondition), desnemu delu pravila pa **posledica pravila** (and. rule consequent). Na levi strani pravila so navedeni pogoji za enega ali več atributov, desna stran pravila pa vsebuje napoved za vrednost atributa "razred", glede na pogoje na levi. Če nek podatkovni primer ustreza pogojem v pravilu, pravimo, da pravilo pokriva (ang. covers) ta primer. Pravilo lahko ocenimo s **pokrivnostjo pravila** (ang. rule coverage), ki pove, kolikšen delež primerov v podatkovni množici je pokrit s tem pravilom ter z **natančnostjo pravila** (ang. rule accuracy), ki pove, kolikšen delež pokritih primerov pravila je pravilno klasificiranih. Ko klasificiramo nek primer, se lahko zgodi, da več pravil pokriva nek primer. S tem namenom so vsa pravila navadno urejena glede na neko mero (npr. natančnost, pokrivnost). Primer je potem klasificiran s tistim pravilom, ki je najvišje rangiran (in ki pokriva primer). Obstaja veliko algoritmov, s katerimi lahko iz množice klasificiranih podatkov zgradimo pravila oblike ČE-POTEM za klasifikacijo primerov. V nadaljevanju je na kratko opisano, kako lahko takšna pravila nastanejo iz odločitvenih dreves.

Ko je odločitveno drevo zgrajeno, lahko le-tega pretvorimo v odločitvena pravila. Za vsako pot od korenskega vozlišča do končnega vozlišča v odločitvenem drevesu lahko ustvarimo eno pravilo, tako da vsak delitveni kriterij na poti predstavlja logično

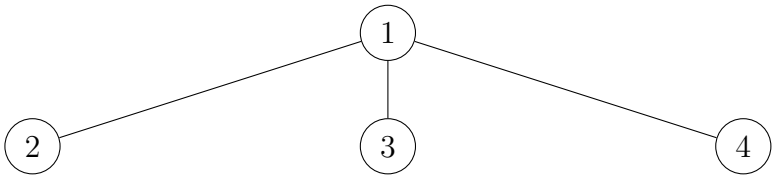
povezavo IN v "ČE" pogoju, vrednost "razreda" v končnem vozlišču pa predstavlja posledico pravila. V tem primeru se ne more zgoditi, da bi en podatkovni primer pripadal dvema ali več različnim pravilom. Vendar na ta način, ko pravila generiramo direktno iz odločitvenega drevesa, nič ne pridobimo. Algoritem C4.5 zato pravila generira iz neporezanega drevesa in šele potem pravila reže s pesimistično metodo, ki je podobna tisti pri rezanju odločitvenih dreves [8].

Takšen postopek pridobivanja pravil, kot ga uporablja algoritem C4.5, je precej zamuden, zato je v nadaljevanju opisan postopek, ki za generiranje pravil uporablja "delno" zgrajena odločitvena drevesa (ang. partial decision trees) skupaj s postopkom deli in vladaj. V Weki je implementiran algoritem za generiranje klasifikacijskih pravil z uporabo "delnih" odločitvenih dreves, ki se imenuje **PART** [9]. Algoritem PART deluje tako, da iz "delnega" odločitvenega drevesa naredi eno pravilo in izbriše vse primere, ki so pokriti s tem pravilom in tako nadaljuje, dokler ni nobenega primera več na voljo. Da generira pravilo, uporabi že porezano "delno" odločitveno drevo in pogleda tisti list v drevesu, ki pokriva največ primerov. Pri tem algoritmu množice generiranih klasifikacijskih pravil ni potrebno naknadno rezati in/ali izboljševati, ampak sklepamo, da so generirana pravila že dovolj dobra, ker izhajajo iz porezanega "delnega" odločitvenega drevesa in iz tega vidika je PART hitrejši in manj kompleksen.

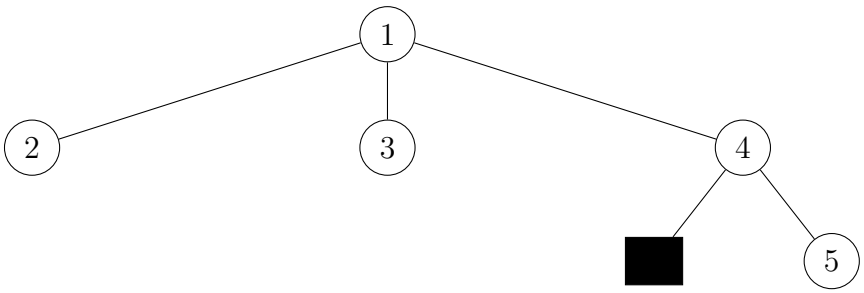
Pri algoritmu PART je bistvenega pomena, kako generirati "delno" odločitveno drevo. Prvi korak je enak kot pri "navadnih" odločitvenih drevesih, torej algoritem na začetni množici podatkov določi delitveni kriterij, na enak način kot to stori C4.5. V nadaljevanju se delitveni kriteriji v vozliščih prav tako določajo na enak način kot v C4.5, le s to razliko, da se najprej razdeli podmnožico, ki ima najmanjšo povprečno informacijsko entropijo. To pomeni, da po tem, ko je začetna podatkovna množica razdeljena v podmnožice glede na delitveni atribut, algoritem izbere tisto podatkovno podmnožico, ki ima najmanjšo povprečno entropijo in na tej množici določi delitveni kriterij na tak način kot to stori algoritem C4.5. To se rekurzivno nadaljuje, dokler podmnožica, ki nastane, ni list. Potem algoritem nadaljuje delitev podmnožic, tako da gre "nazaj" po vozliščih v drevesu (ang. backtracking). Ampak, takoj ko obstaja vozlišče, za katerega so vsi potomci listi, nastopi postopek rezanja. Rezanje poteka na podoben način kot pri algoritmu C4.5. Ko pridemo do vozlišča, katerega vsi potomci so listi, preverimo, ali je stopnja napake poddrevesa (od tega notranjega vozlišča) večja od stopnje napake lista, ki je označen z večinsko vrednostjo "razreda" za primere, ki so v tem vozlišču. Če je stopnja napake slednjega manjša, potem list zamenja poddrevo. Če je narejena menjava, potem algoritem rekurzivno deluje naprej, tako da pogleda katero od so-vozlišč (na isti stopnji) ima najmanjšo povprečno entropijo in deli naprej. Če se na nekem vozlišču, ki ima vse potomce liste, ta menjava ne izvede, to pomeni konec generiranja odločitvenega drevesa in nekatera vozlišča lahko ostanejo nedokončana.

Ko je takšno "delno" odločitveno drevo zgrajeno iz njega nastane eno klasifikacijsko pravilo in sicer PART navadno uporabi tisti list v odločitvenem drevesu, ki pokriva največje število podatkovnih primerov. Primeri, ki so s tem pravilom pokriti, se izbrišejo iz podatkovne množice in na novi podatkovni množici se ponovno generira "delno" odločitveno drevo, iz katerega nastane novo pravilo. To se ponavlja, dokler ne zmanjka primerov v podatkih. Glavna prednost tega algoritma je ta, da je preprost za uporabo in terja krajši čas za generiranje množice pravil.

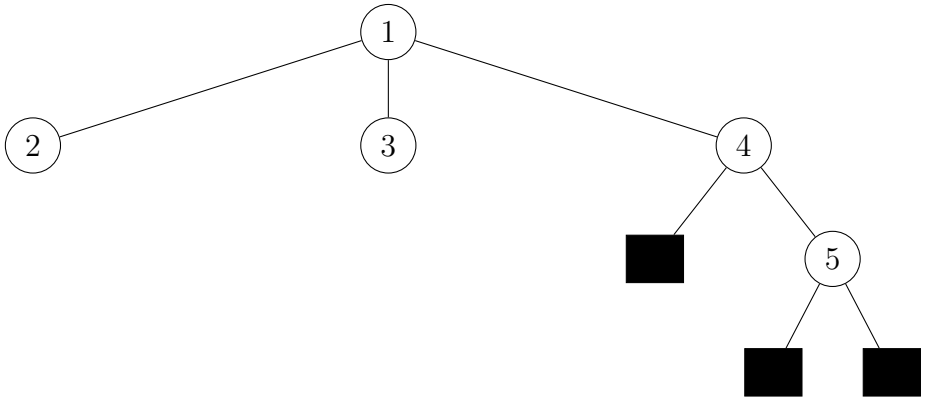
Na slikah 4 - 8 so na primeru prikazani koraki generiranja delnega odločitvenega drevesa. Na teh slikah črno obarvana kvadratna vozlišča predstavljajo končna vozlišča oziroma liste. Na začetku algoritem PART, na enak način kot algoritem C4.5, določi delitveni atribut v vozlišču 1. Ta razdeli začetno podatkovno množico in nastanejo tri nova vozlišča (slika 4) označena kot $2, 3, 4$. Med temi algoritem izbere tisto vozlišče oziroma tisto podmnožico, ki ima najmanjšo povprečno entropijo in na tej podmnožici nadaljuje z delitvijo podatkov. V našem primeru je to vozlišče 4. Na takšen način, kot smo ga že prej opisali, se nadaljuje generiranje drevesa, dokler ne pride do situacije, prikazane na sliki 6, kjer sta oba potomca vozlišča 5 lista. Stopnja napake poddrevesa v vozlišču 5 je večja, kot je stopnja napake, če podatke v tem vozlišču klasificiramo z večinsko vrednostjo atributa "razred", zato se poddrevo v vozlišču 5 zamenja z listom, označenim z večinsko vrednostjo atributa "razred" v vozlišču 5. Enak način rezanja se nadaljuje v vozlišču 4 (slika 7). Po temu koraku gre algoritem po drevesu "nazaj" in določi, da ima med vozlišči 2 in 3, podmnožica v vozlišču 3 manjšo povprečno entropijo. V vozlišču 3 se nato ponovno nadaljuje generiranje drevesa kot v algoritmu C4.5 in ponovno pridemo do situacije, kjer sta oba potomca lista. V tem primeru se zamenjava poddrevesa v vozlišču 3 ne izvede in to pomeni konec generiranja odločitvenega drevesa. Zadnji korak je prikazan na sliki 8.



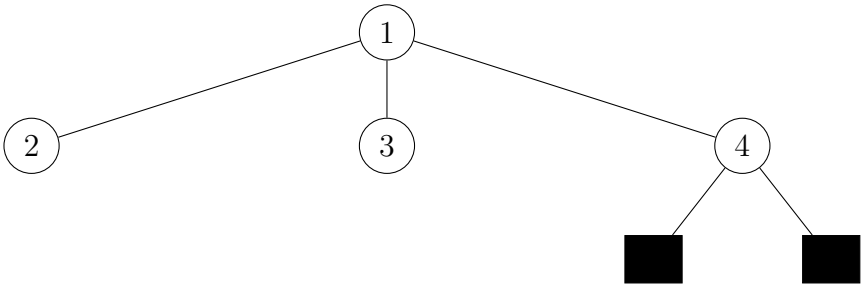
Slika 4: Generiranje "delnega" odločitvenega drevesa - 1. faza



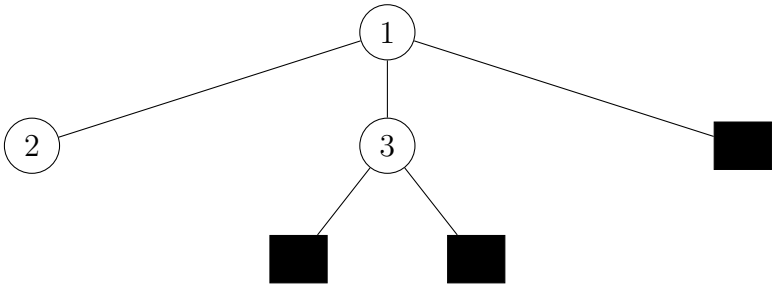
Slika 5: Generiranje "delnega" odločitvenega drevesa - 2. faza



Slika 6: Generiranje "delnega" odločitvenega drevesa - 3. faza



Slika 7: Generiranje "delnega" odločitvenega drevesa - 4. faza



Slika 8: Generiranje "delnega" odločitvenega drevesa - 5. faza

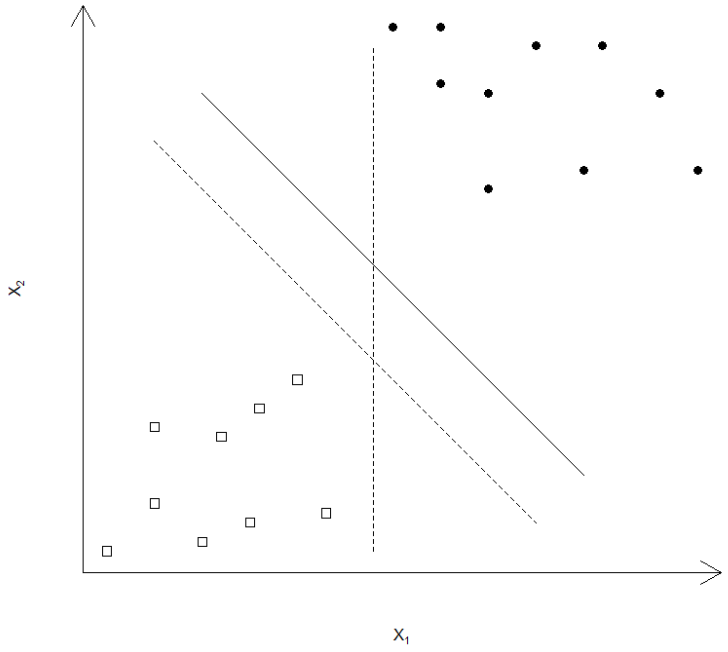
4.3 Metoda podpornih vektorjev

Metoda podpornih vektorjev (ang. Support Vector Machines) je ena od pogosteje uporabljenih metod za nadzorovano učenje. Delovanje algoritma podpornih vektorjev je

bilo prvič uradno opisano leta 1992 v delu Vladimirja Vapnika in njegovih kolegov. Metodo podpornih vektorjev krajše označimo s kratico SVM. Algoritem za SVM podatke z nelinearno preslikavo preslika v višjo dimenzijo in v tej dimenziji išče najboljšo hiperravnino, ki podatke loči glede na vrednosti atributa "razred".

4.3.1 Intuitivna ideja metode podpornih vektorjev

V nadaljevanju je najprej opisana ideja metode podpornih vektorjev na enostavnem klasifikacijskem problemu z binarnim atributom "razred". Množica učnih podatkovnih primerov je enaka $D = \{x_1, \ldots, x_{|D|}\}$. Vsakemu primeru x_i pripada vrednost atributa "razred", ki ga označimo z Y in za katerega predpostavimo, da je $y_i \in \{-1, +1\}$, $i = 1, \ldots, |D|$. Na začetku predpostavimo tudi to, da so podatki linearno ločljivi (ang. lineary separable). Pravimo, da so podatkovni primeri linearno ločljivi, če v prostoru obstaja hiperravnina [1], ki primere loči, glede na vrednost atributa Y.



Slika 9: Linearno ločljivi podatki

Na sliki 9 so prikazani podatkovni primeri, ki so določeni z vrednostmi dveh atributov (X_1, X_2). Na grafu krogi označujejo primere z $y_i = +1$, kvadratki pa označujejo primere $y_i = -1$. Iz grafa je hitro razvidno, da so v tem dvodimenzionalnem primeru

[1] Hiperravnina v n-razsežnem prostoru F je ravninska podmnožica z razsežnostjo $n - 1$. V dvorazsežnem prostoru je hiperravnina premica, v višjih razsežnostih pa uporabljamo izraz hiperravnina.

podatki, glede na vrednosti atributa Y, ločljivi s premico, kar pomeni, da so linearno ločljivi. Hitro opazimo, da v tem primeru obstaja veliko število premic, ki prav tako ločujejo podatke glede na pripadajoče vrednosti atributa "razred". Cilj SVM algoritma je poiskati najboljšo premico za ločitev podatkov oziroma tisto, ki bi imela najmanjšo stopnjo napake pri klasifikaciji podatkov. V dvodimenzionalnem primeru govorimo o premici, v višjih dimenzijah pa je potrebno poiskati hiperravnino, ki ločuje podatke. Ne glede na dimenzijo bomo uporabljali izraz hiperravnina (tudi če bomo imeli v mislih dvodimenzionalen prostor). Takšno hiperravnino, ki podatke loči glede na vrednosti atributa "razred", imenujemo tudi **ločitvena hiperravnina** (ang. separating hyperplane) ali meja odločanja (ang. decision boundary). Cilj metode podpornih vektorjev je poiskati ločitveno hiperravnino, ki bi dajala čim bolj zanesljive in točne klasifikacijske napovedi na podatkih [8].

Pri izpeljavi optimizacijskega problema za metodo podpornih vektorjev potrebujemo še nekaj oznak. Recimo, da je v podatkovni množici m atributov, ki opisujejo podatke. Vektor atributov lahko spet označimo z X. Pri učni fazi SVM algoritem poišče vektor parametrov

$$W = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{pmatrix}$$

ter konstanto b, ki določata odločitveno funkcijo (ang. decision function), ki jo označimo kot

$$d(X; w, b) = W^T X + b.$$

Ločitvena hiperravnina je definirana kot $W^T X + b = 0$. Klasifikator za podatkovni primer x je v primeru metode podpornih vektorjev funkcija $h_{W,b}(x) = g(W^T x + b)$, pri čemer je $g(z) = 1$, če je $z \geq 0$ in $g(z) = -1$ sicer [14].

4.3.2 Funkcionalni in geometrični rob

Obstaja veliko različnih opisov problema SVM, v tem magistrskem delu pa sledimo razlagi v [21] ter [14]. Za intuitivno idejo SMV metode je potrebna razlaga nekaterih pojmov. Za i-ti podatkovni primer je funkcionalni rob (ang. functional margin) definiran kot:

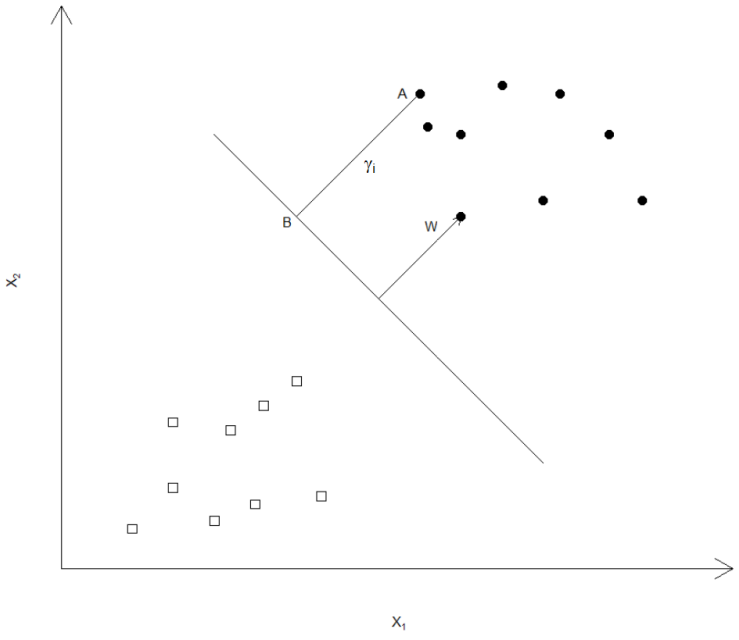
$$\hat{\gamma}_i = y_i \left(W^T x_i + b \right). \tag{4.20}$$

Želimo, da bi bil klasifikator kar se da točen in zanesljiv. Torej, če imamo primer x_i z $y_i = 1$, vidimo, da klasifikator pravilno napove vrednost atributa "razred", ko je

$W^T x_i + b \geq 0$. Klasifikacija je torej v tem primeru tem bolj zanesljiva, čim večje pozitivno število je $W^T x_i + b$. Nasprotno, če imamo primer x_i, kjer je $y_i = -1$, klasifikator pravilno napove vrednost atributa razred, ko je $W^T x_i + b < 0$. V tem primeru je klasifikator tem bolj zanesljiv, čim večje negativno število je $W^T x_i + b$. Torej, v splošnem mora za dober klasifikator veljati $y_i(W^T X + b) \gg 0$. V tem primeru je klasifikacija pravilna in zanesljiva. Minimalni funkcionalni rob med vsemi podatkovnimi primeri v učni množici D označimo kot:

$$\hat{\gamma} = \min_i \hat{\gamma}_i. \tag{4.21}$$

Hitro opazimo, da funkcionalni rob ni v vsakem primeru najboljša mera zanesljivosti. Glede na obliko funkcije g lahko W in b vedno množimo s pozitivno konstanto, recimo namesto W pišemo $2W$ ter namesto b pišemo $2b$ in v tem primeru velja $g(W^T x_i + b) = g(2W^T x_i + 2b)$. Na ta način se napoved klasifikatorja ne spremeni, ker je ta odvisna le od predznaka, ne pa tudi od absolutne vrednosti $W^T x_i + b$. Na ta način lahko postane funckionalni rob poljubno velik, ne da bi spremenili karkoli smiselnega. Želimo, da bi imeli mero, ki bi upoštevala to pomanjkljivost funkcionalnega roba. V nadaljevanju definiramo še geometrični rob (ang. geometric margin) za i-ti podatkovni primer.



Slika 10: Geometrični rob

Na sliki 10 je narisana odločitvena meja oziroma hiperravnina, ki ločuje podatke. Hitro lahko opazimo, da je ločitvena hiperravnina ortogonalna na vektor W. Točka A predstavlja i-ti podatkovni primer, za katerega je $y_i = 1$. Razdalja od točke A do ločitvene hiperravnine, ki je prikazana na sliki, je podana z dolžino daljice AB. Ta razdalja predstavlja geometrični rob za i-ti podatkovni primer. Geometrični rob za i-ti

podatkovni primer označimo z γ_i. Tega lahko na preprost način izpeljemo. Enotski vektor, ki kaže v enako smer kot W, je enak $\frac{W}{\|W\|}$, pri čemer je $\|W\|$ Evklidska norma vektorja W. Točka, ki je označena z B, je torej podana z $x_i - \gamma_i \frac{W}{\|W\|}$. Točka B leži na ločitveni hiperravnini. Sledi torej, da je $W^T \left(x_i - \gamma_i \frac{W}{\|W\|} \right) + b = 0$. Če iz te enačbe izrazimo γ_i dobimo:

$$\gamma_i = \left(\frac{W}{\|W\|} \right)^T x_i + \frac{b}{\|W\|}.$$

Ta izpeljava deluje na primeru za točko A, za katero je $y_i = 1$. Če želimo, da ta formula za geometrični rob velja za vse podatkovne primere (tudi za primere, kjer je $y_i = -1$), potem je potrebno geometrični rob za i-ti primer definirati kot:

$$\gamma_i = y_i \left(\left(\frac{W}{||W||} \right)^T x_i + \frac{b}{||W||} \right). \tag{4.22}$$

Vidimo, da je $\gamma_i = \frac{\hat{\gamma}_i}{||W||}$. Geometrični rob za i-ti primer je za razliko od funkcionalnega roba invarianten za množenje s konstanto. Za celotno učno množico podatkov je geometrični rob definiran kot:

$$\gamma = \min_i \gamma_i. \tag{4.23}$$

Cilj SVM klasifikatorja je poiskati najboljšo odločitveno mejo oziroma hiperravnino, ki loči podatke glede na vrednost atributa "razred". Smiselno je privzeti, da je najboljša ločitvena hiperravnina tista, ki maksimizira (geometrični) rob.

Kot smo že omenili, v tem primeru predpostavljamo, da so podatki linearno ločljivi. Vprašanje je, kako poiskati hiperravnino, ki maksimizira geometrični rob. Optimizacijski problem, ki ga določa SVM, je potrebno na tej točki še nekoliko poenostaviti z določenimi privzetki. Spomnimo se, da lahko W in b množimo s poljubno pozitivno konstanto, ne da bi spremenili logiko funkcionalnega roba. Predpostavimo, da mora za obe množici primerov (za primere z $y_i = 1$ in primere z $y_i = -1$) veljati, da imajo točke, ki so najbližje odločitveni meji, funkcionalni rob enak 1. Iz tega sledi, da mora veljati:

$$W^T x_i + b \geq 1, \text{ za } y_i = +1$$

$$W^T x_i + b \leq -1, \text{ za } y_i = -1.$$

Ti dve množici pogojev lahko združimo v:

$$y_i(W^T x_i + b) \geq 1, i = 1, \ldots, |D|.$$

Po tem privzetku velja, da je razdalja najbližjega podatkovnega primera (za množico podatkovnih primerov z $y_i = 1$ in množico primerov z $y_i = -1$) do ločitvene hiperravnine enaka $\frac{1}{\|W\|}$. Razdaljo med množico podatkovnih primerov z $y_i = 1$ in množico

podatkovnih primerov z $y_i = -1$ imenujenujemo rob (ang. margin). Za iskanje najboljše ločitvene hiperravnine želimo rob maksimizirati. V našem primeru to pomeni, da želimo maksimizirati razdaljo med podatkovnimi primeri z $y_i = 1$ in $y_i = -1$, ki je v tem primeru enaka $\frac{2}{\|W\|}$. To je ekvivalentno minimizaciji $\|W\|$ oziroma $\|W\|^2$ [4]. Posledično dobimo naslednji optimizacijski problem:

$$\begin{aligned} \min_{W,b} \quad & \frac{1}{2}\|W\|^2 \\ \text{p.p.} \quad & y_i(W^T x_i + b) \geq 1,\ i = 1, \ldots, |D|. \end{aligned} \tag{4.24}$$

4.3.3 Reševanje konveksnega optimizacijskega problema

Optimizacijski problem, ki smo ga izpeljali, je konveksni kvadratični optimizacijski problem. V nadaljevanju na kratko opišemo, na kakšen način poteka reševanje tega optimizacijskega problema. Pri reševanju takšnega problema je v pomoč teorija konveksne optimizacije [3]. Dejstva, ki so v tem delu uporabljena, so v delu [3] tudi dokazana. Konveksni optimizacijski problem je problem oblike

$$\begin{aligned} \min_{x} \quad & f_0(x) \\ \text{p.p.} \quad & f_i(x) \leq 0,\ i = 1, \ldots, m \\ & h_i(x) = 0,\ i = 1, \ldots, p, \\ & x \in X, \end{aligned} \tag{4.25}$$

pri čemer so $f_0, f_1, \ldots, f_m$ konveksne funkcije in $h_1, \ldots, h_p$ afine funkcije definirane na nekem podprostoru X v $\mathbb{R}^n$. Funkcijo f_0 imenujemo tudi ciljna funkcija (ang. objective function). Konveksni optimizacijski problem je kvadratičen (ang. quadratic program), če je ciljna funkcija kvadratna in so funkcije v pogoju afine.

Optimizacijskemu problemu 4.25 pripada posplošena Lagrangeova funkcija

$$L(x, \alpha, \beta) = f_0(x) + \sum_{i=1}^{m} \alpha_i f_i(x) + \sum_{i=1}^{p} \beta_i h_i(x),$$

kjer konstante $\alpha_i, \ldots, \alpha_m, \beta_1, \ldots, \beta_p$ imenujemo Lagrangeovi multiplikatorji. Obstaja več poti po katerih z Lagrangeovo funkcijo pridemo do rešitve optimizacijskega problema. En način je ta, da najprej maksimiziramo Lagrangeovo funkcijo, glede na Lagrangeove multiplikatorje $\alpha = (\alpha_1, \ldots, \alpha_m)$ ter $\beta = (\beta_1, \ldots, \beta_p)$. Dobimo:

$$\theta_P(x) = \max_{\alpha,\beta:\alpha_i \geq 0} L(x, \alpha, \beta).$$

Rezultat te maksimizacije potem minimiziramo po x, torej:

$$\min_{x} \max_{\alpha,\beta:\alpha_i \geq 0} L(x, \alpha, \beta) = \min_{x} \theta_P(x). \tag{4.26}$$

Temu minimizacijskemu problemu pravimo primarni problem (ang. primal problem). Z x^* označimo optimalni x, pri katerem je dosežen minimum. Označimo še $p^* = \theta_P(x^*)$. Za primarni problem lahko definiramo tudi dualni problem. Dualni problem na nek način zamenja vrstni red minimizacije in maksimizacije. Naj bo:

$$\theta_D(\alpha, \beta) = \min_x L(x, \alpha, \beta).$$

Dualni problem, ki pripada primarnemu problemu 4.26, je enak:

$$\max_{\alpha,\beta:\alpha_i \geq 0} \min_x L(x, \alpha, \beta). = \max_x \theta_D(\alpha, \beta)$$

Z (α^*, β^*) označimo parametre (α, β), pri katerih je dosežen maksimum in označimo še $d^* = \theta_D(\alpha^*, \beta^*)$. Za vsak tak optimizacijski problem lahko pokažemo, da velja $d^* \leq p^*$ in temu pravimo, da velja pogoj šibke dualnosti. Pod določenimi pogoji pa velja tudi $d^* = p^*$ in v tem primeru pravimo, da velja pogoj krepke dualnosti. V tem primeru lahko namesto primarnega problema rešujemo dualni problem. V konveksnem optimizacijskem problemu 4.25 je $d^* = p^*$, če je izpolnjen Slaterjev pogoj (ang. Slater's condition), ki pravi, da obstaja takšen x, za katerega vse pogojne neenakosti v optimizacijskem problemu strogo veljajo, tj. $f_i(x) < 0, i = 1, \ldots, m$.

Za iskanje rešitve konveksnega optmizacijskega problema pa si lahko pomagamo s Karush-Kuhn-Tuckerjevimi pogoji ali krajše KKT pogoji. Recimo, da imamo konveksni optimizacijski problem 4.25. Naj bodo x^* ter (α^*, β^*) optimalne rešitve primarnega in dualnega problema ter naj bo $d^* = p^*$. V tem primeru veljajo naslednji (KKT) pogoji:

$$\begin{aligned}
&\nabla_x L(x^*, \alpha^*, \beta^*) = 0 \\
&f_i(x^*) \leq 0,\ i = 1, \ldots, m \\
&h_i(x) = 0,\ i = 1, \ldots, p \\
&\alpha_i \geq 0,\ i = 1, \ldots, m \\
&\alpha_i^* f_i(x^*) = 0,\ i = 1, \ldots, m
\end{aligned} \tag{4.27}$$

Za konveksni optimizacijski primer pa v obratno smer velja še več. V tem primeru velja, da če katerikoli x^* ter (α^*, β^*) zadoščajo KKT pogojem 4.27, je to zadosten pogoj za optimalnost oziroma so (x^*, α^*, β^*) optimalne rešitve primarnega in dualnega problema in je $d^* = p^*$.

Reševanje optimizacijskega problema SVM

Problem 4.24, ki smo ga izpeljali za metodo podpornih vektorjev, je konveksni kvadratični optimizacijski primer. Ciljna funkcija je kvadratična in konveksna in funkcije v pogoju so prav tako konveksne. Teorijo konveksne optimizacije lahko uporabimo za reševanje problema 4.24 [14, 21].

V optimizacijskem problemu 4.24 imamo v pogojih samo neenakosti. Pogoje lahko zapišemo kot:

$$-y_i(W^T x_i + b) + 1 \leq 0, i = 1, \ldots, |D|.$$

Lagrangeova funkcija, ki pripada optimizacijskemu primeru v 4.24, je enaka:

$$L(W, b, \alpha) = \frac{1}{2}\|W\|^2 - \sum_{i=1}^{|D|} \alpha_i [y_i(W^T x_i + b) - 1].$$

Optimizacijski problem 4.24 želimo zapisati le v odvisnosti od skalarnih produktov $\langle x_i, x_j \rangle$, saj to kasneje omogoči učinkovito delovanje algoritma v višjih dimenzijah. Iz tega razloga rešitve problema 4.24 iščemo v dualnem prostoru. Pri tem upoštevamo KKT pogoje, za katere vemo, da so v primeru konveksnega optimizacijskega problema zadostni pogoji za optimalnost rešitev. KKT pogoji za naš optimizacijski problem so enaki:

$$\begin{aligned}
&\frac{\partial}{\partial W} L(W, b, \alpha) = 0 \\
&\frac{\partial}{\partial b} L(W, b, \alpha) = 0 \\
&\alpha_i \geq 0, \ i = 1, \ldots, |D| \\
&y_i(W^T x_i + b) - 1 \geq 0, \ i = 1, \ldots, |D| \\
&\alpha_i(y_i(W^T x_i + b) - 1) = 0, \ i = 1, \ldots, |D|.
\end{aligned} \tag{4.28}$$

Poiščimo torej dualni problem. Najprej minimiziramo $L(W, b, \alpha)$ glede na W in b in tako dobimo θ_D. Računamo odvode L po W in b in dobimo:

$$\frac{\partial L(W, b, \alpha)}{\partial W} = W - \sum_{i=1}^{|D|} \alpha_i y_i x_i = 0.$$

Iz tega sledi:

$$W = \sum_{i=1}^{|D|} \alpha_i y_i x_i.$$

Z odvajanjem po b dobimo še:

$$\frac{\partial L(W, b, \alpha)}{\partial b} = -\sum_{i=1}^{|D|} \alpha_i y_i = 0,$$

torej mora veljati:

$$\sum_{i=1}^{|D|} \alpha_i y_i = 0.$$

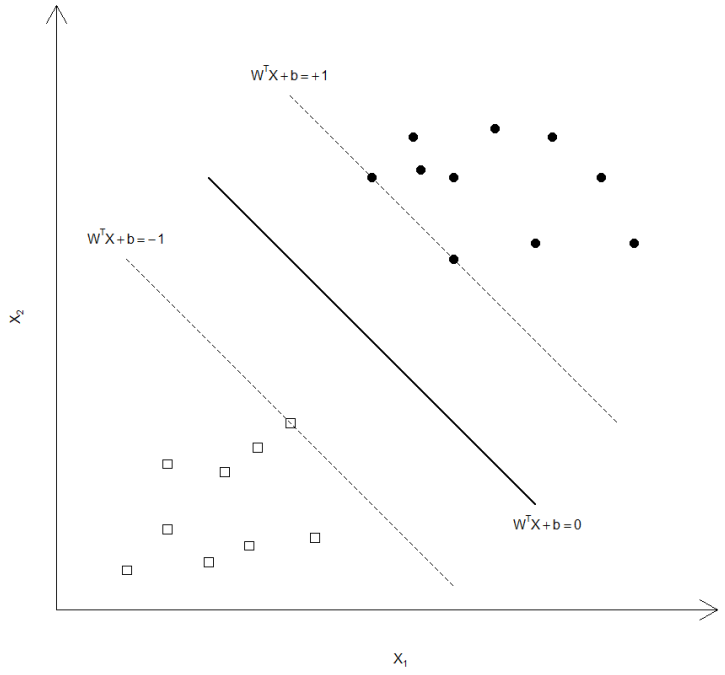
Zdaj lahko enačbe, ki smo jih dobili iz KKT pogojev, uporabimo in Lagrangeovo funkcijo prepišemo v:

$$L(W, b, \alpha) = \sum_{i=1}^{|D|} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{|D|} y_i y_j \alpha_i \alpha_j x_i^T x_j.$$

Dualni problem zapišemo kot:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{|D|} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{|D|} y_i y_j \alpha_i \alpha_j x_i^T x_j \\ \text{p.p.} \quad & \alpha_i \geq 0, \ i = 1, \ldots, |D| \\ & \sum_{i=1}^{|D|} \alpha_i y_i = 0. \end{aligned} \tag{4.29}$$

Lagrangeova funkcija je tako izražena z učnimi podatki in je odvisna samo od njihovih skalarnih produktov $\langle x_i, x_j \rangle$. Z minimiziranjem Lagrangeove funkcije po W dobimo obliko rešitve za W. Optimalna rešitev za W je odvisna od optimalne rešitve α. Da dobimo rešitev za b je potrebno upoštevati zadnji KKT pogoj v 4.28, ki ga imenujemo tudi pogoj dualne komplementarnosti (ang. dual complementary condition). Podatkovnim primerom, za katere v rešitvi velja enakost $y_i(W^T x_i + b)$, pravimo **podporni vektorji** (ang. support vectors).



Slika 11: Na sliki je z odebeljeno črto prikazana ločitvena hiperravnina $W^T X + b = 0$. Podporni vektorji so primeri, ki ležijo na hiperravninama $W^T X + b = -1$ in $W^T X + b = -1$.

Zaradi pogoja dualne komplementarnosti velja, da če je $\alpha_i \neq 0$, sledi, da je $y_i(W^T x_i + b) = 1$ oziroma je α_i podporni vektor in takšen primer leži najbližje ločitveni hiperravnini. Za izračun optimalnega b si pomagamo s primeri, za katere je $\alpha_i > 0$. Če vzamemo i-ti podatkovni primer, za katerega je $\alpha_i > 0$, iz zadnjega KKT pogoja velja, da je:

$$b = \frac{1}{y_i} - W^T x_i.$$

Ker je $y_i \in \{-1, 1\}$, je:

$$b = y_i - W^T x_i.$$

Z b_i označimo b izračunan iz enačbe za i-ti podatkovni primer. Numerično ustrezneje je, če za končno vrednost b vzamemo povprečno vrednost vseh b_i, ki jih dobimo iz primerov, za katere je $\alpha_i > 0$.

Recimo, da smo rešili optimizacijski problem 4.24 in optimalne vrednosti za vse parametre označimo z (W^*, b^*, α^*). Optimizacijski problem smo reševali na primerih iz učne množice. Recimo, da imamo zdaj nov primer x, ki ni v učni množici in mu želimo napovedati vrednost atributa "razred". V tem primeru izračunamo predznak izraza $(W^*)^T x + b^*$ in ta nam pove kako klasificiramo podatkovni primer x. To lahko zapišemo tudi malo drugače in sicer kot:

$$(W^*)^T x + b^* = \left(\sum_{i=1}^{|D|} \alpha_i^* y_i x_i \right)^T x + b^* = \sum_{i=1}^{|D|} \alpha_i^* y_i \langle x_i, x \rangle + b^*.$$

Za podatkovne primere, ki niso podporni vektorji, velja da je $\alpha_i = 0$, zato je veliko členov v vsoti enakih 0, kar poenostavi izračun.

Do zdaj smo pri izpeljavi optimizacijskega problema za metodo podpornih vektorjev predpostavljali, da so vhodni učni podatki linearno ločljivi. V realnosti se pogosto zgodi, da podatki niso linearno ločljivi ali pa je te velikokrat nesmiselno ločevati s hiperravnino. V nekaterih primerih je enostavneje, če podatke ločimo z nelinearno hiperploskvijo. Ta problem rešujemo tako, da prostor vhodnih podatkovnih primerov najprej preslikamo v nov prostor, ki je ponavadi višjih razsežnosti. Ko so podatkovni primeri preslikani v vektorje višjih razsežnosti, v novem prostoru poiščemo linearno odločitveno mejo. V večini primerov je to mogoče poiskati, obstajajo pa tudi primeri, ko to ni mogoče. Za transformacijo podatkov v višje dimnezije uporabimo teorijo jeder (ang. kernels).

4.3.4 Jedra (ang. kernels)

Kot že omenjeno lahko podatkovne primere preslikamo v nov prostor (višjih razsežnosti) in na novem prostoru poiščemo ločitveno hiperravnino. Če za transformacijo vhodnih

podatkov v nov prostor uporabimo nelinearno preslikavo, to pomeni, da originalne vhodne podatke potem ločimo z nelinearno hiperploskvijo.

Naj bo Θ preslikava, ki slika iz $\mathbb{R}^m$ v $\mathbb{R}^f$, kjer je $f \geq m$. S preslikavo vhodnih podatkov v nov prostor upamo, da bodo v novem prostoru podatkovni primeri linearno ločljivi. Recimo, da imamo dva vhodna učna primera, ki ju označimo z x in z. Pokazali smo, da je optimizacijski problem v primeru linearno ločljivih podatkov v celoti izražen s skalarnimi produkti $\langle x, z \rangle$. Ko podatke transformiramo v nov prostor s preslikavo Θ, se ta skalarni produkt torej spremeni v $\langle \Theta(x), \Theta(z) \rangle$. V optimizacijskemu problemu 4.29 lahko torej skalarni produkt vhodnih podatkovnih vektorjev zamenjamo s skalarnim produktom preslikanih vhodnih vektorjev.

Primer 4.1. Recimo, da imamo podatkovni primer x, ki je opisan s tremi atributi oziroma je predstavljen kot tridimenzionalen vektor. Recimo, da je $x = (x^1, x^2, x^3)$. Preslikava θ je definirana kot:

$$\Theta(x) = \begin{pmatrix} x^1x^1 \\ x^1x^2 \\ x^1x^3 \\ x^2x^1 \\ x^2x^2 \\ x^2x^3 \\ x^3x^1 \\ x^3x^2 \\ x^3x^3 \end{pmatrix}. \tag{4.30}$$

Tridimenzionalni vektor smo s preslikavo transformirali v devetdimenzionalni vektor in v devetdimenzionalnem prostoru upamo, da so transformirani vhodni podatki linearno ločljivi. V novem prostoru želimo poiskati ločitveno hiperravnino na takšen način, kot smo ga opisali v primeru linearno ločljivih podatkov. Ko to poiščemo, se vrnemo v prostor originalnih vhodnih podatkov in v tem prostoru je odločitvena meja polinom drugega reda.

Ob preslikavi podatkov v nov prostor imamo nekaj težav. Prva težava je ta, da težko izberemo, katero preslikavo Θ uporabiti na vhodnih podatkih. Druga težava pa je ta, da je računanje $\Theta(x)$ oziroma računanje v novih (višjih) razsežnostih, lahko precej časovno zahtevno. Zato je na tem mestu potrebno uporabiti nov trik, ki ga imenujemo **trik z jedri** (ang. kernel trick). Za preslikavo Θ definiramo pripadajoče **jedro** (ang. kernel) kot $K(x, z) = \Theta(x)^T\Theta(z)$. Povsod, kjer smo zamenjali skalarni produkt vhodnih podatkov $\langle x, z \rangle$ s skalarnim produktom $\langle \Theta(x), \Theta(z) \rangle$, zdaj pišemo $K(x, z)$. Prvi način kako izračunati $K(x, z)$ je ta, da izračunamo produkt $\Theta(x)^T\Theta(z)$,

vendar vemo, da je lahko ta izračun časovno precej zahteven. Zanimivo je, da obstajajo funkcije, s katerimi lahko na nezahteven način izračunamo $K(x, z)$, ne da bi bilo potrebno eksplicitno računati $\Theta(x)$.

Za preslikavo Θ, ki je navedena v primeru 4.1, velja, da je $K(x, z) = (x^T z)^2$. Lahko se prepričamo, da je v tem primeru $K(x, z) = \Theta(x)^T\Theta(z)$. Razlika pa je ta, da v primeru računanja $\Theta(x)^T\Theta(z)$ računamo z vektorji dimenzije devet, v drugem primeru, ko pa posebej definiramo $K(x, z)$ pa z vektorji dimenzije tri.

Vprašanje je, kdaj je neka funkcija K veljavno jedro (ang. valid kernel) oziroma kdaj za funkcijo K obstaja preslikava Θ, da velja $K(x, z) = \Theta(x)^T\Theta(z)$ za vse primere x, z, ki so v podatkovni množici. Predpostavimo, da je K neko veljavno jedro za preslikavo Θ, ki vektor x preslika v m dimenzionalen vektor. Poglejmo si primer, ko imamo množico vektorjev $\{x_1, \ldots, x_n\}$ (to niso nujno podatkovni vektorji iz učne množice). Z G označimo matriko velikosti $n \times n$, katere (i, j)-ti element je enak $G_{ij} = K(x_i, x_j)$. To matriko imenujemo matrika jedra (ang. kernel matrix). Če je K veljavno jedro, potem velja:

$$G_{ij} = K(x_i, x_j) = \Theta(x_i)^T\Theta(x_j) = \Theta(x_j)^T\Theta(x_i) = K(x_j, x_i) = G_{ji}$$

oziroma to pomeni, da je G simetrična matrika. Preverimo pa lahko še to, da je za poljuben n dimenzionalen neničeln vektor z, vrednost $z^T Gz$ vedno nenegativna. To pomeni, da je matrika G pozitivno semidefinitna. Pokažimo, da je v primeru veljavnega jedra K matrika G pozitivno semidefinitna.

$$\begin{aligned} z^T Gz &= \sum_{i=1}^{n}\sum_{j=1}^{n} z_i G_{ij} z_j = \sum_{i=1}^{n}\sum_{j=1}^{n} z_i \Theta(x_i)^T\Theta(x_j) z_j = \sum_{i=1}^{n}\sum_{j=1}^{n} z_i \sum_{k=1}^{m} \Theta_k(x_i)\Theta_k(x_j) z_j \\ &= \sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{n} z_i \Theta_k(x_i)\Theta_k(x_j) z_j = \sum_{k=1}^{m}\left(\sum_{i=1}^{n} z_i \Theta_k(x_i)\right)^2 \geq 0 \end{aligned}$$

Pokazali smo, da je v primeru, ko je K veljavno jedro, pripadajoča matrika G simetrična pozitivno semidefinitna. Pokazati se da, da je to zadosten pogoj za veljavnost jedra. Temu pravimo tudi Mercerjev pogoj (ang. Mercer condition), katerega dokaz lahko najdemo v [24]. Pogosto uporabljene funkcije za jedra so naslednje:

- $K(x_i, x_j) = (x_i^T x_j)$: linearno jedro,
- $K(x_i, x_j) = [(x_i^T x_j) + 1]^d$: polinomsko jedro.

4.3.5 Problem neločljivih podatkov in posplošitev SVM

V prejšnjem razdelku smo opisali kako SVM algoritem vhodne podatke preslika v prostor višjih dimenzij in na tem prostoru išče najboljšo ločitveno hiperravnino. Transformacija podatkov v višje razsežnosti v splošnem poveča možnost, da postanejo podatki

linearno ločlljivi, vendar ni nujno, da to velja v vsakem primeru. Poleg tega so v podatkih včasih tudi "zašumljeni" podatkovni primeri, ki lahko močno vplivajo na to kakšna bo ločitvena hiperravnina, česar pa ne želimo. Optimizacijski problem, ki smo ga izpeljali za linearno ločljive podatke, je potrebno posplošiti tako, da je manj občutljiv na močno izstopajoče podatke in da deluje tudi v primeru linearno neločljivih podatkov. V optimizacijskem problemu 4.24 smo s pogoji ustvarili pas okrog ločitvene hiperravnine, tako da smo določili, da znotraj pasu ne sme ležati noben podatkovni primer. Posplošen algoritem deluje na ta način, da nekoliko sprosti pogoj v optimizacijskem problemu 4.24, ki pravi, da mora za vse primere v podatkih veljati:

$$y_i(W^T x_i + b) \geq 1.$$

V posplošeni verziji algoritma dovolimo, da za kakšen primer velja tudi, da je:

$$y_i(W^T x_i + b) < 1.$$

Pogoj v 4.24 se spremeni v:

$$y_i(W^T x_i + b) \geq 1 - \epsilon_i,\ i = 1, \ldots, |D|$$
$$\epsilon_i \geq 0,\ i = 1, \ldots, |D|.$$

Optimizacijski problem je zdaj oblike:

$$\begin{aligned} \min_{W,b,\epsilon} \quad & \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{|D|}\epsilon_i \\ \text{p.p.} \quad & y_i(W^T x_i + b) \geq 1 - \epsilon_i,\ i = 1, \ldots, |D| \\ & \epsilon_i \geq 0,\ i = 1, \ldots, |D|. \end{aligned} \tag{4.31}$$

V optimizacijskemu problemu ϵ predstavlja vektor s koordinatami $\epsilon_1, \ldots, \epsilon_{|D|}$. C predstavlja nenegativen parameter, ki utežuje med dvema ciljema in sicer med tem, da je objektna funkcija kar se da majhna in med tem, da ima večina primerov funkcionalni rob vsaj 1. Če za i-ti primer velja, da je $\epsilon_i > 0$, oziroma če ima funkcionalni rob manjši kot 1, to "plačamo" na ta način, da se objektna funkcija poveča za $C\epsilon_i$.

Tudi v tem primeru rešujemo konveksni optimizacijski problem. Tako kot za 4.24 poiščemo dualni problem, ki mu pripada. Zapišimo najprej Lagrangeovo funkcijo, ki je v tem primeru enaka:

$$L(W, b, \epsilon, \alpha, r) = \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{|D|}\epsilon_i - \sum_{i=1}^{|D|}\alpha_i[y_i(W^T x_i + b) - 1 + \epsilon_i] - \sum_{i=1}^{|D|}\beta_i\epsilon_i.$$

KKT pogoji v tem primeru so enaki:

$$\begin{aligned}
&\frac{\partial}{\partial W}L(W,b,\epsilon,\alpha,\beta)=0\\
&\frac{\partial}{\partial b}L(W,b,\epsilon,\alpha,\beta)=0\\
&\frac{\partial}{\partial \epsilon}L(W,b,\epsilon,\alpha,\beta)=0\\
&\alpha_i \geq 0,\ i=1,\ldots,|D|\\
&\beta_i \geq 0,\ i=1,\ldots,|D|\\
&y_i(W^T x_i+b)-1+\epsilon_i \geq 0,\ i=1,\ldots,|D|\\
&\epsilon_i \geq 0,\ i=1,...,|D|\\
&\alpha_i[y_i(W^T x_i+b)-1+\epsilon_i]=0,\ i=1,\ldots,|D|\\
&\beta_i\epsilon_i=0,\ i=1,\ldots,|D|.
\end{aligned} \tag{4.32}$$

Zadnjim dvem pogojem pravimo pogoja dualne komplementarnosti.
Najprej minimiziramo Lagrangeovo funkcijo $L(W,b,\epsilon,\alpha,\beta)$ po W,b ter ϵ tako, da jo odvajamo po W,b ter ϵ in odvode enačimo z nič. Iz tega dobimo naslednje pogoje:

$$\begin{aligned}
W&=\sum_{i=1}^{|D|}\alpha_i y_i x_i,\\
\sum_{i=1}^{|D|}\alpha_i y_i&=0,\\
\alpha_i+\beta_i&=C.
\end{aligned}$$

Z upoštevanjem teh pogojev lahko Lagrangeovo funkcijo prepišemo v:

$$L(W,b,\epsilon,\alpha,\beta)=\sum_{i=1}^{|D|}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{|D|}y_i y_j\alpha_i\alpha_j\langle x_i,x_j\rangle.$$

Dualni problem, ki ga dobimo, je enak tistemu, ki smo ga zapisali za linearno ločljive podatke z eno majhno izjemo. Vemo namreč, da je $\beta_i=C-\alpha_i$ in ker mora veljati, da je $\beta_i\geq 0$, mora biti $\alpha_i\leq C$. Dualni problem, ki ga dobimo, je torej enak

$$\begin{aligned}
\max_{\alpha}\quad &\sum_{i=1}^{|D|}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{|D|}y_i y_j\alpha_i\alpha_j x_i^T x_j\\
\text{p.p.}\quad &0\leq\alpha_i\leq C,\ i=1,\ldots,|D|\\
&\sum_{i=1}^{|D|}\alpha_i y_i=0.
\end{aligned}$$

Poglejmo si, kaj lahko povemo o podatkovnem primeru x_i, glede na vrednost α_i, ki jo dobimo pri reševanju optimizacijskega problema. Naslednji trije pogoji sledijo iz KKT pogojev v 4.32.

- Če je $\alpha_i = 0$ in če je C pozitiven parameter, iz zadnjih dveh KKT pogojev velja, da je $\epsilon_i = 0$ ter $y_i(W^T x_i + b) \geq 1$. V tem primeru je i-ti podatkovni primer pravilno klasificiran.

- Če je $\alpha_i = C$, potem (ker je $\epsilon_i \geq 0$) zaradi predzadnjega KKT pogoja v 4.24 velja, da je $y_i(W^T x_i + b) = 1 - \epsilon_i$ oziroma $y_i(W^T x_i + b) \leq 1$. Če je $0 \leq \epsilon_i < 1$, velja, da je i-ti podatkovni primer pravilno klasificiran, če pa je $\epsilon_i \geq 1$, je i-ti podatkovni primer napačno klasificiran. Primerom x_i, za katere je $\alpha_i = C$, pravimo omejeni podporni vektorji.

- Če je $0 < \alpha_i < C$, iz zadnjih dveh KKT pogojev v 4.32 velja, da je $y_i(W^T x_i + b) = 1$ in je v tem primeru i-ti podatkovni primer pravilno klasificiran. Primerom x_i za katere je $0 < \alpha_i < C$ pravimo neomejeni ali prosti podporni vektorji.

Za izračun optimalnega b tudi v tem primeru upoštevamo tiste podatkovne primere za katere je $\alpha_i > 0$ in iz pogojev dualne komplementarnosti izrazimo b. Eden pogosteje uporabljenih algoritmov za metodo podpornih vektorjev je SMO, ki je kratica za *sequential minimal optimization*. Algoritem je zapisal John Platt leta 1998 in vsebuje reševanje dualnega problema, ki smo ga opisali v tem poglavju. Algoritem SMO je implementiran tudi v orodju Weka.

4.4 Ocenjevanje klasifikacijskih modelov

V prvem koraku klasifikacije zgradimo klasifikator na podlagi učne množice podatkov. V drugem koraku je potrebno oceniti natančnost klasifikatorja na testni množici. Če bi klasifikacijsko natančnost preverjali na učni množici, torej na množici, s katero zgradimo klasifikator, bi bili rezultati pristranski in nekoliko preveč optimistični. To je logično, saj se klasifikator "uči" iz učne množice in je torej prilagojen tej množici. Veliko bolje je, da se natančnost klasifikatorja meri na množici podatkov, ki ni bila uporabljena v učni fazi klasifikacije. Teorija ocenjevanja klasifikacijskih modelov je navedena v osmem poglavju knjiige [8].

4.4.1 Mere klasifikacijske natančnosti

Obstaja več načinov ocenjevanja natančnosti modelov. Najbolj osnovna mera, ki je za to uporabljena, se imenuje **klasifikacijska natančnost** (ang. accuracy). Recimo, da

imamo klasifikacijski problem, kjer ima atribut "razred" K kategorij. Testno množico označimo s T in število primerov v le-tej s $|T|$. Primere v testni množici označimo z x_i, kjer je $i = 1, \ldots, |T|$. Za podatkovni primer x_i z y_i označimo dejansko vrednost atributa "razred", z $\hat{y}_i$ pa označimo vrednost atributa "razred", ki jo je napovedal klasifikator, katerega natančnost preverjamo. Klasifikacijska natančnost klasifikatorja M je v tem primeru enaka:

$$Acc(M) = \frac{1}{|T|} \sum_{i=1}^{|T|} \mathbb{1}_{\{y_i = \hat{y}_i\}}. \tag{4.33}$$

Obratno od natančnosti lahko preverimo tudi **stopnjo napake** (ang. error rate), ki je enaka $Err(M) = 1 - Acc(M)$. Čeprav je natančnost zelo široko uporabljena mera, ko govorimo o točnosti modelov, pa se v veliko primerih izkaže kot slaba mera, ki da premalo informacije o modelu. To velja predvsem za podatkovne množice, v katerih je prisoten **problem neuravnoteženosti "razreda"** (ang. class imbalance problem). Poglejmo si primer, kjer imamo množico podatkov o ženskah in rakavih obolenjih. Atribut "razred" je poimenovan z "rak" in ima vrednost "DA", če je bil pri ženski ugotovljen rak in "NE" v nasprotnem primeru. V učni množici je 97% primerov, ki imajo pri "rak" vrednost "NE" in 3% primerov, ki imajo vrednost "DA". Recimo, da imamo model, ki vedno napove, da bo vrednost atributa "rak" enaka "NE". Na testni množici, ki ima enako porazdelitev pri "DA" in "NE" kot učna množica podatkov, bi v tem primeru dobili klasifikacijsko natančnost 97%, ki velja za precej visoko natančnost modela, ki pa je v resnici zelo slab. Če bi takšen model uporabljali zdravniki, bi pravilno ugotovili samo primere, ko ženska nima raka, za vse tiste primere v katerih bi bilo pa nujno ugotoviti, da je rak prisoten, bi pa zgrešili.

V tem delu je potrebno uvesti dodatne mere za ugotavljanje točnosti modelov, ki dajo več informacij o klasifikatorju. Recimo, da imamo množico podatkov z binarnim atributom "razred". Z izrazom **pozitivni primeri** (ang. positive example) označimo tiste primere, ki pripadajo vrednosti "razreda", ki nas najbolj zanima. V primeru žensk in rakavih obolenj bi bili to primeri, ki imajo pri "rak" vrednost "DA". Ostale primere v podatkovni množici imenujemo **negativni primeri** (ang. negative example). Ko izvedemo klasifikacijo in testiramo model na testni množici, pravilno klasificirane pozitivne primere imenujemo **resnično pozitivni primeri** (ang. true positives) in število takšnih primerov označimo s TP, pravilno klasificirane negativne primere pa imenujemo **resnično negativni primeri** (ang. true negatives) in število teh označimo s TN. Klasifikator pa lahko nekatere negativne primere označi kot pozitivne in takšim pravimo **lažni pozitivni primeri** (ang. false positives) in število teh označimo s FP. Prav tako lahko pozitivne primere označi kot negativne in takšnim pravimo **lažni negativni primeri** (ang. false negatives) in število teh označimo s FN.

Večina orodij za strojno učenje rezultate modelov na testni množici med drugim prikaže tudi s **kontingenčno tabelo** (ang. contingency table). Kontingenčna tabela za klasifikacijski problem , kjer ima atribut "razred" K vrednosti ($K \geq 2$) je matrika velikosti vsaj $K \times K$. V prvih K vrsticah in K stolpcih (i, j)-ti element matrike predstavlja število primerov i-te vrednosti atributa "razred", ki jih je klasifikator razvrstil v j-to vrednost atributa "razred", za $i, j = 1, \ldots, K$.

Poglejmo si kakšna je oblika tabele za klasifikacijski model z binarno spremenljivko "razred", z vrednostima "DA" in "NE", kjer so pozitivni primeri označeni z "DA". Prikazana je v tabeli 1.

	Napoved "DA"	Napoved "NE"	Vsota
Dejanski "DA"	TP	FN	$P = TP + FN$
Dejanski "NE"	FP	TN	$N = FP + TN$
Vsota	$P' = TP + FP$	$N' = TN + FN$	$P + N$

Tabela 1: Kontingenčna tabela

Oznaka P v tabeli pomeni število pozitivnih primerov, P' pa število tistih primerov, ki jih je klasifikator označil za pozitivne. Z N je označeno število negativnih primerov, z N' pa število tistih primerov, ki jih je klasifikator označil za negativne. V množici, na kateri testiramo klasifikator, je $P + N$ primerov, kar je enako kot $P' + N'$ oziroma $TP + TN + FP + FN$. Če je klasifikator dober, je večina primerov na diagonali kontingenčne tabele, ostali elementi pa želimo, da so ničelni ali blizu nič. Z oznakami, ki smo jih navedli, lahko mero natančnosti klasifikatorja M, ki smo jo prej označili z $Acc(M)$, zapišemo kot $Acc(M) = \frac{TP+TN}{P+N}$. Stopnjo napake lahko zapišemo kot $Err(M) = \frac{FP+FN}{P+N}$. Kot smo že omenili, želimo poleg mere natančnosti imeti še kakšno drugo mero, ki bi bila bolj primerna za množice z neuravnoteženim "razredom". S tem namenom definiramo **mero občutljivosti** (ang. sensitivity), ki je včasih poimenovana kot **stopnja resnično pozitivnih primerov** (ang. true positive rate) in je enaka $TP_{rate} = \frac{TP}{P}$. Vidimo, da ta prikazuje delež tistih primerov, ki so resnično pozitivni med vsemi pozitivnimi primeri. Poleg tega lahko definiramo tudi stopnjo **specifičnosti** (ang. specificity), ki je včasih poimenovana tudi kot **stopnja resnično negativnih primerov** (ang. true negative rate) in je definirana kot $TN_{rate} = \frac{TN}{N}$.

Spomnimo se primera z ženskami in rakavimi obolenji, kjer je porazdelitev pri atributu "rak" z vrednostmi "DA" in "NE" zelo neuravnotežena. Spomnimo se klasifikatorja, ki vedno napove, da ima "razred" vrednost "NE". V tem primeru je klasifikacijska natančnost tega klasifikatorja enaka 97%, vendar je mera občutljivosti enaka 0%, saj klasifikator zgreši vse pozitivne primere.

V klasifikaciji sta pogosto uporabljeni tudi **mera popolnosti** (ang. recall), ki je definirana enako kot stopnja resnično pozitivnih primerov ter **mera točnosti** (ang. precision), ki jo definiramo kot razmerje med resnično pozitivnimi primeri in med vsemi primeri, ki so bili označeni kot pozitivni.

Opazimo, da za mero točnosti in mero občutljivosti oziroma popolnosti velja, da če eno mero zvišamo, lahko pri tem drugo mero znižamo. Recimo, da v primeru rakavih obolenj za vse primere napovemo, da imajo vrednost atributa "rak" enako "DA". V tem primeru dobimo visoko mero občutljivosti, saj klasifikator pravilno klasificira vse pozitivne primere, vendar je v primeru takšnega klasifikatorja mera točnosti nizka, saj za veliko negativnih primerov klasifikator zmotno napove, da je vrednost za "rak" enaka "DA".

ROC krivulja

Za primerjavo klasifikatorjev med seboj pa je pogosto v uporabi **ROC krivulja**. Okrajšava ROC krivulja izhaja iz angleškega izraza *receiver operating characteristic curve*. V nadaljevanju je ROC krivulja opisana na primeru klasifikacije z binarnim atributom "razred". ROC krivulja je grafično orodje, ki prikazuje odvisnost stopnje resnično pozitivnih primerov (TP_{rate}) od stopnje lažno pozitivnih primerov (FP_{rate}). Stopnjo lažno pozitivnih primerov lahko izračunamo kot $1 - TN_{rate}$ oziroma kot delež lažno pozitivnih primerov med vsemi negativnimi primeri. Ploščina pod ROC krivuljo je pomemben pokazatelj natančnosti klasifikatorja in jo označimo z AUC, ki je okrajšava za izraz *Area under the ROC curve*. Ko preverjamo rezultate klasifikatorja, si želimo, da je AUC čim večja. Poglejmo si najprej, kako poteka risanje ROC krivulje v splošnem in nato še na konkretnem primeru.

Recimo, da imamo klasifikator M in $|T|$ testnih primerov. Za risanje ROC krivulje potrebujemo (za vsak primer v podatkovni množici) verjetnost, da ima primer vrednost "razreda", ki jo je določil klasifikator. Drugače povedano, za risanje ROC krivulje je potrebno, da klasifikator izračuna verjetnost, da je podatkovni primer pozitiven. Verjetnost, da je i-ti primer pozitiven označimo s p_i, $i = 1, \ldots, |T|$. V nadaljevanju testne primere razvrstimo glede na te verjetnosti. Primer z največjo verjetnostjo, je najvišje rangiran. Te verjetnosti uporabimo za risanje ROC krivulje na ta način, da na vsakem koraku določimo eno od teh verjetnosti za prag (ang. threshold) in glede na tega klasificiramo primere. Recimo, da je prag enak t. Potem i-ti primer klasificiramo na ta način, da če je $p_i \geq t$, ga označimo kot pozitiven primer, sicer pa kot negativen. Za prag t izračunamo TP_{rate} in FP_{rate}, ki določata koordinati točke ROC krivulje. Ta postopek ponovimo z različnimi pragovi (ang. thresholds) in sproti rišemo ROC krivuljo.

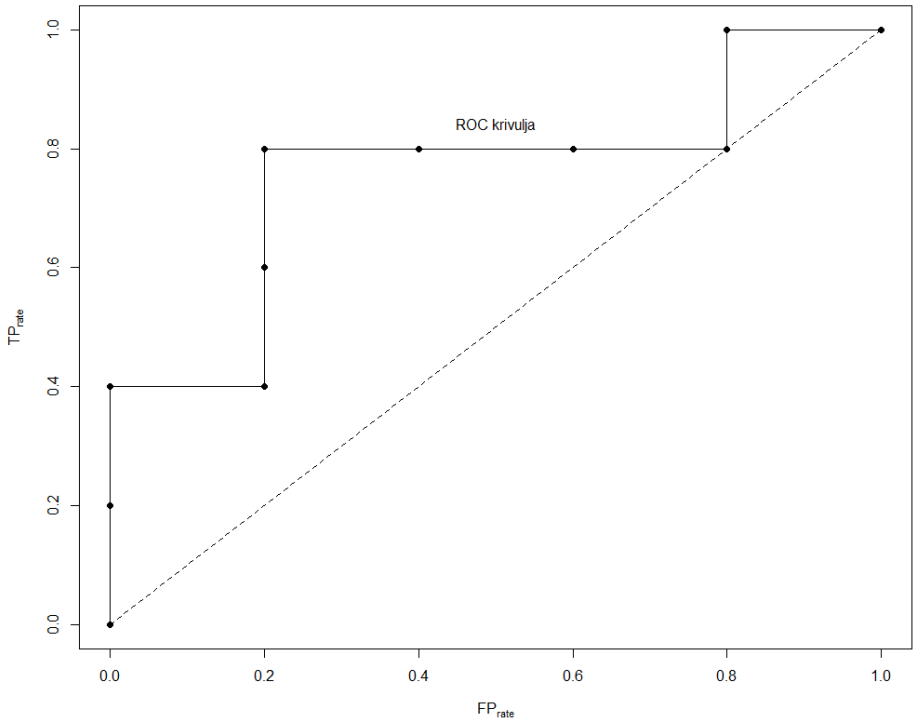
Pri ROC krivulji x-os predstavlja FP_{rate}, y-os pa predstavlja TP_{rate}. Risanje krivulje se začne v spodnjem levem kotu, kjer je $TP_{rate} = 0$ ter $FP_{rate} = 0$. Nato nadaljujemo tako, da za vsak prag izračunamo TP_{rate} in FP_{rate} in ti razmerji predstavljata koordinati točk ROC krivulje. V koordinatnem prostoru, kjer je narisana ROC krivulja je navadno narisana tudi diagonala, ki prikazuje, da je enako verjetno, da je primer TP ali FP. Bližje kot je ROC krivulja diagonali, slabši je model. V primeru dobrega klasifikatorja ROC krivulja najprej strmo narašča do nekega praga in potem, ko je vse manj TP primerov, naraščanje upada in krivulja postaja horizontalna.

Primer 4.2. V tabeli 2 je 10 testnih primerov. Drugi stolpec vsebuje podatek o tem ali je primer dejansko pozitiven (=P) ali negativen (=N). V tretjem stolpcu so podane verjetnosti p_i, ki jih je določil klasifikator. Vidimo, da je v tabeli pet pozitivnih primerov in pet negativnih primerov. Začetna točka risanja ROC krivulje

PRIMER (i)	DEJANSKI RAZRED (P/N)	p_i	TP	FP	TN	FN	TP_{rate}	FP_{rate}
1	P	0.9	1	0	5	4	0.2	0
2	P	0.8	2	0	5	3	0.4	0
3	N	0.7	2	1	4	3	0.4	0.2
4	P	0.6	3	1	4	2	0.6	0.2
5	P	0.55	4	1	4	1	0.8	0.2
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4	1	1	0.8	0.8
9	P	0.5	5	4	0	1	1.0	0.8
10	N	0.4	5	5	0	0	1.0	1.0

Tabela 2: Tabela za risanje ROC krivulje

je $(0, 0)$. Nadaljujemo s prvim primerom in za tega vidimo, da je klasifikator določil $p_1 = 0.9$. Zaradi tega je t na začetku enak 0.9 in klasifikator v tem primeru določi, da je pozitiven le prvi primer, ostali pa so negativni. Za $t = 0.9$ je $TP = 1$, saj je prvi primer res pozitiven in ga je tudi klasifikator označil kot pozitivnega, ostale primere pa kot negativne, zato je $FP = 0$. Za prvi primer je torej $TP_{rate} = \frac{1}{5} = 0.2$ in $FP_{rate} = 0$. To vzamemo za koordinati naslednje točke ROC krivulje in dobimo (0.2,0). Za drugi primer vzamemo $t = 0.8$. V tem primeru klasifikator določi, da sta prva dva primera označena kot pozitivna, ostali pa kot negativni. V tem primeru je $TP_{rate} = \frac{2}{5} = 0.4$ in $FP_{rate} = 0$. Na ta način (s premikanjem t-ja) nadaljujemo do zadnjega primera in dobimo ROC krivuljo, ki je narisana na sliki 12.



Slika 12: ROC krivulja

4.4.2 Metode vzorčenja

Kot smo že na začetku tega razdelka omenili, je zelo pomembno, katero množico podatkov uporabimo za učenje klasifikatorja in katero uporabimo za ocenjevanje natančnosti klasifikatorja. Če za oba koraka klasifikacije uporabljamo enako množico podatkov, so lahko rezultati klasifikatorja preveč optimistični. Zato je potrebno imeti poleg učne množice še testno množico podatkovnih primerov, ki niso vsebovani v učni množici.

Naključno vzorčenje

Najenostavnejši način kako pridobiti testno množico je ta, da preden začnemo s klasifikacijo, množico podatkov, ki nam je na voljo, naključno razdelimo na učno in testno množico. Navadno se za učne primere vzame $\frac{2}{3}$ primerov celotne množice, preostalo $\frac{1}{3}$ podatkov pa za testne primere. Težava te metode je, da se učna množica tako zmanjša in so rezultati na testni množici lahko zato slabši. Poleg tega pa je slabost tudi ta, da je rezultat na testni množici nekoliko odvisen od delitve podatkov na učno in testno množico. Ta način vzorčenja lahko izboljšamo tako, da ta postopek ponovimo k-krat. Za končno klasifikacijsko natančnost naredimo povprečje vseh k modelov.

K-kratno prečno preverjanje

Naslednja metoda, ki je pogosto uporabljena pri ocenjevanju natančnosti modela, je k-kratno prečno preverjanje (ang. k-fold cross validation). Pri tej metodi začetno množico vseh podatkov, ki so nam na voljo, razdelimo v k približno enako velikih množic $D_1, \ldots, D_k$, tako da je vsak podatkovni primer lahko samo v eni od teh množic. Pri k-kratnem prečnem preverjanju učenje in ocenjevanje natančnosti klasifikatorja izvedemo k-krat. V prvi iteraciji uporabimo D_1 za preverjanje natančnosti klasifikatorja, vse ostale množice skupaj pa uporabimo za učenje klasifikatorja. Ta postopek ponovimo še za vse ostale množice, tako, da je vsakič ena od množic testna množica. Za razliko od "navadnega" naključnega vzorčenja, kjer smo množico razdelili na dva dela, pri k-kratnem prečnem preverjanju vsak podatkovni primer $(k-1)$ krat uporabimo za učenje in enkrat za testiranje. Natančnost klasifikatorja se v tem primeru izračuna kot povprečje vseh k-iteracij.

Recimo, da s k-kratnim prečnim preverjanjem računamo klasifikacijsko natančnost klasifikatorja M. Za vsako ponovitev dobimo neko klasifikacijsko natančnost Acc_i, za $i = 1, \ldots, k$. Če označimo klasifikacijsko natančnost, ki jo dobimo s k-kratnim prečnim preverjanjem za klasifikator M s $CV_k(M)$, velja, da je $CV_k(M) = \frac{1}{k} \sum_{i=1}^{k} Acc_i$. Poseben primer k-kratnega prečnega preverjenja je metoda izpusti enega (ang. leave-one-out), kjer je k število vseh primerov v začetni podatkovni množici.

Obstaja še veliko drugih načinov vzorčenja podatkov, s katerimi ocenjujemo natančnost modela, v tej nalogi pa smo uporabljali naključno vzorčenje in 10-kratno prečno preverjanje.

5 KLASIFIKACIJA NA PRIMERU

Kot smo že v uvodu zapisali, smo klasifikacijske algoritme uporabili na primeru zavarovanj osebnih avtomobilov. Na začetku je bila glavna naloga določiti problem, na katerem bi izvedli klasifikacijo. Problem, ki ga obravnavamo v magistrskem delu, se nanaša na obnovitve zavarovanj osebnih avtomobilov. Zavarovalne police imajo določen datum sklenitve, datum začetka zavarovalnega leta in datum poteka zavarovalnega leta. Datum sklenitve predstavlja tisti datum, ko stranka z zavarovalnico sklene zavarovalno pogodbo in jo obe strani s podpisi potrdita. Ob sklenitvi zavarovanja se določi, od katerega datuma naprej velja zavarovanje in ta datum predstavlja datum začetka zavarovalnega leta. Po enem letu ali manj (odvisno od časa trajanja police) od datuma začetka zavarovalnega leta nastopi datum poteka zavarovalnega leta. Ob tem datumu ima stranka možnost polico obnoviti [1]. V kolikor stranka polico obnovi, se na polici zapiše nov datum začetka zavarovalnega leta in tako se nadaljuje dokler se police ne prekine. Obnovitev zavarovalne police pomeni na nek način podaljšanje zavarovanja, vendar se lahko pogoji zavarovanja na polici ob tem spremenijo (npr. kritja, premija, popusti, ipd.). V večini primerov si zavarovalnice želijo, da bi stranke zavarovalno polico obnovile. Tega si ne želijo le v primerih, ko stranka za zavarovalnico predstavlja grožnjo, oziroma ko nekateri kazalniki kažejo, da je to izredno slaba stranka (npr. stranka, ki ima izredno slab škodni rezultat). Namen praktičnega dela te naloge je bil s klasifikacijskimi algoritmi, ki smo jih v teoretičnem delu opisali, zgraditi klasifikatorje, ki na podlagi podatkov iz zavarovalne police napovejo, ali je bolj verjetno, da bo stranka polico obnovila ali ne.

5.1 Predstavitev in predpripava podatkov

Praktični del te naloge se je izvajal na dejanskih podatkih ene od slovenskih zavarovalnic. V začetnih fazah je bilo potrebno sodelovati s strokovnjaki iz področja zavarovalne tehnologije, ki problem dobro poznajo. Poleg tega vedo tudi, kateri podatki so primerni

[1] Če je čas trajanja zavarovalne police več kot eno leto (npr. permanentno), potem se ob poteku zavarovalnega leta polica avtomatično obnovi, razen v primeru, ko jo stranka sama pravočasno prekine.

za takšno nalogo in so na voljo za obdelavo. Podatki, ki smo jih prejeli v obdelavo, so bili pripravljeni v zavarovalnici in so bili "popravljeni" na ta način, da iz njih ni mogoče razbrati, za katero polico gre. Začetna množica podatkov, ki je bila pripravljena v zavarovalnici, je vsebovala 80000 primerov polic zavarovanj osebnih avtomobilov s podatkom o tem ali je bila polica ob datumu poteka zavarovalnega leta obnovljena ali ne. Podatke o policah se je pripravljalo v maju 2018 in vsi podatki v začetni tabeli so prikazovali stanje, ki je bilo zapisano v podatkovnih bazah zavarovalnice ob tistem času, ko se je podatke zbiralo. Zajelo se je samo tiste police, ki so imele datum poteka zavarovalnega leta v letu 2017 in ki izhajajo iz zavarovanj osebnih avtomobilov. Podatki, ki smo jih prejeli v obdelavo so nam bili predani v obliki Excelove tabele. Podatki v tabeli niso vsebovali vseh polic zavarovanj osebnih avtomobilov, ki so imele datum poteka v 2017, ampak le en delež, v katerem je bilo ohranjeno originalno razmerje obnovljenih in neobnovljenih polic.

Na kratko si poglejmo opis prvotne tabele. Začetna tabela je imela 80000 primerov polic ter 53 stolpcev oziroma atributov. Prvi stolpec je predstavljal zgolj zaporedno številko police. Temu so sledili podatki o tem, kje in kako je bila polica sklenjena in sicer **poslovna enota, prodajna pot** ter **prodajni kanal**. Za tem so bili podani razni datumi in sicer **datum sklenitve, datum začetka zavarovalnega leta, datum poteka zavarovalnega leta** ter podatek **trajanje police**. Zbrani so bili tudi podatki o zavarovalcu in sicer **pošta, spol, starost, vrsta osebe** (fizična ali pravna) ter **občina zavarovalca**. Enaki podatki so bili navedeni tudi za zavarovanca. Poleg teh podatkov je bil v tabeli za zavarovanca naveden tudi podatek o **dejavnosti, ki jo opravlja**. Pomemben podatek v tabeli je predstavljal tudi **škodni rezultat zavarovalca za vse produkte, za katere je zavarovan pri tej zavarovalnici** ter **škodni rezultat zavarovalca za produkte motornih vozil**. V prvotno tabelo so bili vključeni tudi razni podatki o prometu na polici in sicer **policirana premija brez davka ter fakturirana premija brez davka, likvidirane škode, število škodnih spisov, premijski razred AO** ter **premijski razred AK**. Tabela je vsebovala tudi atribute, ki opisujejo, ali je na polici vključeno določeno kritje ali ne in sicer za zavarovanja **AO, AK, AO +, AN (avtomobilske nezgode), ASS (avtomobilska asistenca)**. Poleg tega je bil v tabelo dodan tudi podatek o **številu kritij AK**. Naslednji sklop stolpcev je vseboval podatke o popustih na polici in sicer **število popustov na polici, znesek popustov na polici, podatek o tem ali ima polica izredni popust (krajše IP) ali ne, procent IP na polici** ter **znesek IP na polici** [2]. V tabeli so bili tudi podatki o avtomobilu, za katerega je bilo sklenjeno zavarovanje in sicer **starost vozila, leto izdelave vozila, datum prve registracije, prevoženi**

[2]Podatki o prometu, kritjih, škodah ter popustih na polici so se v začetni tabeli nanašali na obdobje od začetka zavarovalnega leta do poteka zavarovalnega leta, ki je bilo navedeno v tabeli.

kilometri, moč motorja vozila, nabavna vrednost vozila, vrednost dodatne opreme vozila, oblika karoserije, prostornina motorja, podatek o tem ali je vozilo starodobno ali ne, znamka vozila, vrsta goriva, masa vozila in **opis dodatne opreme**. Zadnji in najpomembnejši stolpec v Excelovi tabeli pa je vseboval **podatek o tem ali je bila zavarovalna polica obnovljena ali ne**. V tabeli 3 lahko vidimo, kakšne oblike je bila prvotna tabela podatkov.

POLICA	POSLOVNA ENOTA	TRAJANJE POLICE	ZAC. ZAV. LETA	POTEK ZAV. LETA	...	OBNOVLJENA
1	08 Poslovna enota Nova Gorica	1 leto	31.8.2016	31.8.2017	...	DA
2	09 Poslovna enota Novo mesto	1 leto	17.4.2016	17.4.2017	...	DA
3	03 Poslovna enota Ljubljana	1 leto	13.2.2016	13.2.2017	...	NE
4	03 Poslovna enota Ljubljana	1 leto	15.10.2016	15.10.2017	...	DA
5	04 Poslovna enota Celje	Do 1 leta	5.10.2017	9.10.2017	...	NE
6	04 Poslovna enota Celje	1 leto	28.8.2016	28.2.2017	...	DA

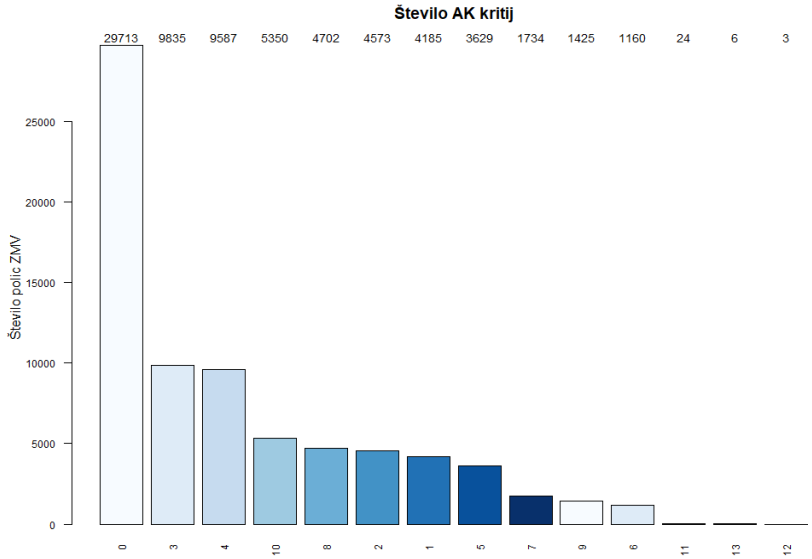
Tabela 3: Prikaz prvotne tabele s podatki

Na začetku smo podatke pregledali in glede na problem, ki smo ga obravnavali, smo en del podatkov izbrisali. V problemu z obnovitvijo polic nas je zanimalo predvsem obnašanje fizičnih oseb, zato smo se odločili, da na nek način pravne osebe izbrišemo in jih pri analizi ne upoštevamo. To smo naredili na ta način, da smo iz oseb zavarovalec in zavarovanec naredili eno osebo, ki jo imenujemo **stranka**. Stranko smo oblikovali na ta način, da če je bil zavarovanec na polici fizična oseba, smo za stranko upoštevali zavarovanca, ne glede na to kakšne vrste je zavarovalec. Če je bil na polici zavarovanec pravna oseba in zavarovalec fizična oseba, potem smo za stranko upoštevali zavarovalca. Če sta bila zavarovanec in zavarovalec na polici oba pravni osebi, smo takšne primere izbrisali. Ko smo izbrisali pravne osebe, nam je ostalo 77309 primerov.

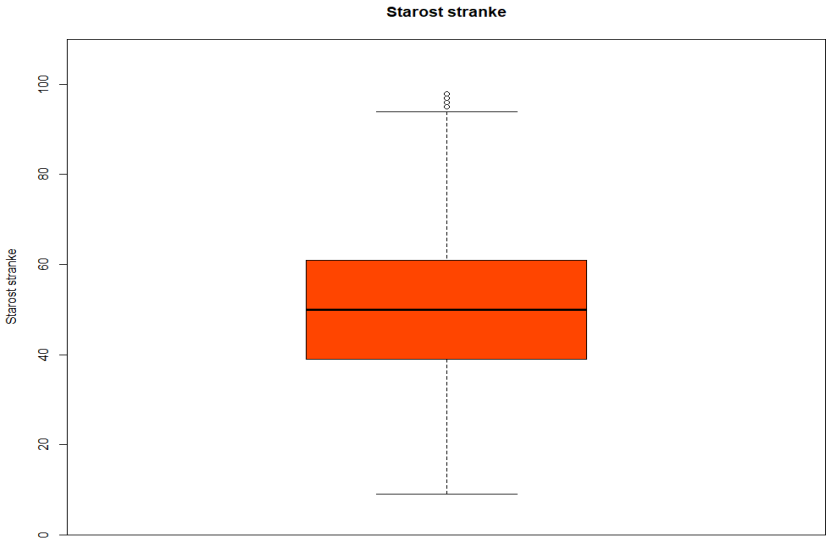
V podatkovni tabeli smo opazili tudi nekaj takšnih podatkov, za katere smo vedeli, da predstavljajo neke vrste napako in takšnih nismo želeli upoštevati. Na primer, v podatkovni množici je bilo okrog 725 polic, ki so imele za atribut "leto izdelave avtomobila" ali "leto prve registracije avtomobila" navedeno leto "2018" (kar pa ni logično, saj so v tabeli bile samo police z datumom poteka zavarovalnega leta v 2017). Takšne primere smo v celoti izbrisali. V podatkih je bilo tudi izredno malo takšnih primerov polic (manj kot 0.01%), ki niso imele vključenega AO kritja, zato smo takšne primere izbrisali in potem sklepali, da imamo v podatkih zgolj police, ki vključujejo AO kritje. Nekateri podatki v podatkovni množici pa so bili zastareli oziroma se v novih policah ne pojavljajo več. "organizator" kot vrsta "prodajnega kanala" se ne uporablja več, zato smo primere s takšnim prodajalnim kanalom izbrisali. Ponekod, kjer smo opazili, da gre za napako v podatkih, pa nismo izbrisali celotnega primera ampak zgolj tisto vrednost, ki je bila napačna.

V tabeli smo imeli kar nekaj atributov, ki so bili podani v obliki datumov. Iz vsakega takšnega atributa smo naredili tri nove stolpce za dan, mesec ter leto posameznega

datuma. Pri obdelavi podatkov smo vse atribute tudi grafično predstavili in si s tem pomagali pri urejanju podatkov. Za kategorične podatke smo narisali histograme, s katerimi smo prikazali porazdelitev kategorij. Prav tako smo si tudi za nekatere numerične podatke risali histograme, ker nas je zanimala porazdelitev po vrednostih (npr. leta pri datumih). Za ostale numerične podatke smo narisali škatle z brki (ang. boxplot), na katerih so jasno razvidni zašumljeni podatki oziroma osamelci. Na sliki 13 vidimo primer histograma za atribut "število AK kritij", na sliki 14 pa primer škatle z brki za atribut "starost stranke".



Slika 13: Primer histograma za atribut "število AK kritij"



Slika 14: Primer škatle z brki za atribut "starost stranke"

S pomočjo grafov smo se odločili združiti vrednosti nekaterih kategoričnih atributov, saj smo si želeli, da bi bile kategorije po številu primerov pri takšnih atributih čim bolj enakomerne. Za atribut "pošta stranke" smo vrednosti združili po okrajih, tako da smo dobili nov atribut "poštni okraj stranke". Naredili smo tudi nov atribut "regija stranke", ki je občine združil v dvanajst regij. Atribut "trajanje police" je imel pet možnih vrednosti ("do 1 leta", "1 leto", "5 let", "7 let", "do 10 let", "permanentno") in tega smo spremenili tako, da smo vrednost "do 1 leta" spremenili v kategorijo "manj kot 1 leto", ostale vrednosti pa smo združili v kategorijo "vsaj eno leto". Atribut "število škodnih spisov" smo spremenili tako, da smo tiste police, ki so imele število škodnih spisov enako 0, združili v kategorijo "brez škodnih spisov", ostale pa v kategorijo "vsaj en škodni spis". Atributa "premijski razred AO" in "premijski razred AK" imata v originalu oba 20 možnih vrednosti. Pri obeh atributih smo vrednosti razdelili v pet kategorij in sicer v kategorije "ni premijskega razreda", "premijski razred 1.-4.", "premijski razred 5.-9.", "premijski razred 10.-14." ter "premijski razred nad 14.". Vrednosti pri atributu "število AK kritij" smo združili v tri kategorije in sicer "nima AK kritij", "manj kot 5 AK kritij", "vsaj 5 AK kritij". Na podoben način smo združili tudi vrednosti atributa "število popustov" v kategorije "ni popustov", "manj kot 5 popustov", "vsaj 5 popustov". Pri atributih "znamka vozila" in "oblika karoserije" je bilo veliko ponovljenih vrednosti, ki so bile drugače zapisane (npr. "MERCEDES BENZ", "MERCEDES-BENZ") in takšne smo ročno pregledali ter jih združili. V tabeli je bil tudi podatek o "starosti vozila", pri katerem je bilo veliko manjkajočih vrednosti, zato smo iz atributa "datum prve registracije" naredili nov atribut za "starost vozila", tako da smo pogledali razliko med letom prve registracije in letom 2018.

Kot že omenjeno je za izvajanje vseh tehnik podatkovnega rudarjenja vedno potrebna dobra predpriprava podatkov, med katero spada tudi dopolnjevanje manjkajočih vrednosti, odkrivanje osamelcev, ipd. V podatkovni tabeli je bilo kar nekaj primerov zavarovalnih polic, ki so vsebovali manjkajoče vrednosti. Manjkajoče vrednosti smo v primeru kategoričnih atributov dopolnili z najpogostejšo vrednostjo, v primeru numeričnih atributov pa z mediano. V kolikor je nek atribut imel več kot 40 % manjkajočih vrednosti, smo se odločili, da tega ne upoštevamo pri podatkovni analizi.

Pri numeričnih atributih smo na grafih (škatle z brki) lahko takoj opazili osamelce. Takšne vrednosti, ki so bile prepoznane kot osamelci, smo najprej izbrisali. Na podatkovni množici smo potem ponovno uporabili klasifikacijske algoritme. Ugotovili smo, da se rezultati klasifikacijskih modelov po izbrisu osamelcev razlikujejo, zato smo se odločili da izbris vseh osamelcev ni najboljša metoda. Izbrisali smo le tiste vrednosti, za katere smo bili prepričani, da gre za napako v vrednosti.

Poleg ostalih metod predpriprave podatkov, smo se odločili, da nekatere numerične atribute diskretiziramo. Na ta način smo nekoliko omilili problem osamelcev oziroma

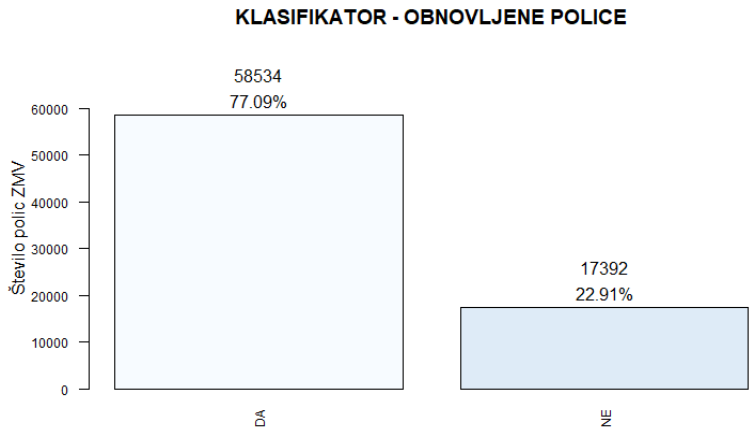
smo jih v obliki novih (diskretiziranih) atributov obravanavali kot posebno kategorijo. Diskretizirali smo vse atribute, ki so vsebovali kakšen podatek o "škodnem rezultatu". Te smo diskretizirali na naslednji način:

$$\text{Škodni rezultat - diskretno} = \begin{cases} \text{brez škod}, & \text{če škodni rezultat} = 0\% \\ \text{normalen škodni rezultat}, & \text{če } 0\% < \text{škodni rezultat} \leq 40\% \\ \text{povprečen škodni rezultat}, & \text{če } 40\% < \text{škodni rezultat} \leq 70\% \\ \text{kritičen škodni rezultat}, & \text{sicer.} \end{cases}$$

Na podoben način smo diskretizirali tudi atribut "delež izrednega popusta (IP)". Iz numeričnega atributa smo naredili nominalen atribut s tremi vrednostmi.

$$\text{Delež IP - diskretno} = \begin{cases} \text{ni IP}, & \text{če ni IP na polici} \\ \text{normalen IP}, & \text{če } 0\% < \text{IP} \leq 25\% \\ \text{kritičen IP}, & \text{sicer.} \end{cases}$$

V podatkovni množici je bil za nas najpomembnejši atribut "razred", ki pove, ali je bila polica obnovljena ali ne. Na grafu si poglejmo porazdelitev števila obnovljenih in neobnovljenih polic na končni urejeni množici podatkov, na kateri smo izvajali klasifikacijske algoritme.



Slika 15: Porazdelitev obnovljenih in neobnovljenih polic v podatkovni množici

Na grafu, ki je prikazan na sliki 15, vidimo, da je bilo v podatkovni množici približno 77% polic, ki so bile ob datumu poteka zavarovalnega leta obnovljene, preostalih 23% pa takšnih, ki niso bile obnovljene. Naš cilj je bil narediti čim boljši klasifikator, ki bi napovedal, ali bo neka zavarovalna polica obnovljena ali ne. Seveda so bile za nas

bolj zanimive tiste police v podatkovni množici, ki niso bile obnovljene. Želeli smo si zgraditi klasifikator, ki bi znal čim bolje prepoznati police, za katere je bolj verjetno, da ne bodo obnovljene. V našem primeru smo zato tistim primerom, za katere polica ni bila obnovljena, rekli pozitivni primeri.

Za izvajanje klasifikacijskih algoritmov na podatkih smo v nalogi uporabljali orodje Weka. Weka je odprtokodna programska oprema za strojno učenje, ki ima kodo napisano v javi. V Weki je implementiranih veliko algoritmov za razne naloge podatkovnega rudarjenja. V tem orodju lahko podatkovno množico tudi urejamo in grafično predstavimo. Za obdelovanje podatkovnih množic v Weki je te potrebno najprej spremeniti v format ARFF, ki ga Weka uporablja za branje in shranjevanje rezultatov.

V praktičnem delu smo se osredotočili na to, da na podatkih uporabimo tiste klasifikacijske algoritme, ki smo jih v četrtem poglavju opisali. V nadaljevanju so opisani rezultati, ki smo jih dobili z algoritmi za klasifikacijo z Naivnim Bayesom, odločitvenimi drevesi, klasifikacijskimi pravili ter z metodo podpornih vektorjev. Rezultate za posamezne algoritme smo med seboj primerjali.

5.2 Rezultati klasifikacijskih algoritmov

V končno urejeni množici podatkov je ostalo 75926 primerov in 46 atributov. V nadaljevanju so opisani rezultati, ki smo jih dobili z uporabo klasifikacijskih algoritmov na končno urejeni množici. Klasifikacijsko natančnost smo preverjali z naključnim vzorčenjem, kjer smo za učno množico vzeli 66% vseh podatkov, preostalih 34% podatkov pa smo uporabili za testno množico. Prav tako smo natančnost klasifikatorjev preverjali tudi z 10-kratnim prečnim preverjanjem. Opazovali smo več različnih mer natančnosti. V tabeli 4 so navedeni rezultati algoritmov, ki smo jih uporabljali.

KLASIFIKATOR	**METODA VZORČENJA**	**TOČNOST (%)**	**TP$_{rate}$**	**AUC**
Zero-R	Naključno vzorčenje	77.09	0.00	0.50
	10 - CV	77.09	0.00	0.50
Naivni Bayes	Nakjučno vzorčenje	74.81	0.39	0.70
	10 - CV	76.64	0.29	0.69
J48 (C4.5)	Naključno vzorčenje	79.44	0.19	0.68
	10 - CV	79.66	0.19	0.68
PART	Naključno vzorčenje	79.43	0.24	0.69
SMO	Naključno vzorčenje	77.11	0.004	0.23

Tabela 4: Rezultati algoritmov

Uporabili smo algoritme Zero-R, Naivni Bayes, C4.5 (v Weki se C4.5 imenuje J48), PART ter SMO. Zero-R je eno najbolj preprostih klasifikacijskih pravil, ki deluje tako,

da pogleda katera je večinska vrednost atributa "razred" na učnih podatkih in to uporabi kot napoved. V našem primeru ZERO-R klasifikator torej vedno napove, da bo polica obnovljena. Seveda pa takoj opazimo, da je takšen klasifikator slab, saj zgreši vse neobnovljene police in zato je v tem primeru $\mathrm{TP}_{rate} = 0$.

Pri uporabi Naivnega Bayesa na podatkih smo dobili klasifikator s klasifikacijsko natančnostjo 74.81%, ko smo preverjali z naključnim vzorčenjem. V tem primeru je klasifikator približno 39% pozitivnih primerov pravilno označil kot pozitivne, kar je seveda bolje kot pri Zero-R, ki nobenega pozitivnega primera ni klasificiral pravilno. Ploščina pod krivuljo ROC je bila v primeru Naivnega Bayesa kar visoka in sicer 0.70. V primeru preverjanja rezultatov klasifikatorja z 10-kratnim prečnim preverjanjem je bila klasifikacijska natančnost nekoliko višja in sicer 76.64%, kar pa je še vedno manj kot je delež večinske vrednosti atributa "razred" v naših podatkih.

Najvišjo klasifikacijsko natančnost je dosegel klasifikator odločitveno drevo, ki smo ga dobili z algoritmom J48 v Weki in sicer 79.44%, ko smo preverjali z naključnim vzorčenjem in 79.66%, ko smo preverjali z 10-kratnim prečnim preverjanjem. Opazimo, da je odločitveno drevo slabše prepoznalo pozitivne primere kot Naivni Bayes, saj je bil v tem primeru TP_{rate} precej nizek.

S klasifikacijskimi pravili, ki jih je generiral algoritem PART, smo dobili skoraj tako visoko klasifikacijsko natančnost kot pri odločitvenem drevesu in sicer 79.43% (pri naključnem vzorčenjenju). Pri algoritmu PART smo zahtevali, da algoritem generira klasifikacijska pravila, ki pokrivajo vsaj 50 podatkovnih primerov. Algoritem PART je tako generiral 41 pravil.

Z algoritmom SMO, ki je zgradil klasifikator iz podpornih vektorjev, smo dobili klasifikacijsko natančnost 77.11%, kar je približno toliko, kot je delež obnovljenih polic v podatkih. Pri SMO algoritmu smo upoštevali polinomsko jedro s stopnjo 1, kar pomeni, da je bil naš klasifikator linearne oblike. Parameter C v SMO algoritmu smo nastavili na 0.1. V primeru algoritmov PART in SMO smo klasifikacijsko natančnost preverjali le z naključnim vzorčenjem, ker je preverjanje natančnosti z 10-kratnim prečnim preverjanjem, zaradi velikosti podatkovne množice, vzelo veliko časa (tudi po več kot nekaj dneh izvajanja se algoritem ni ustavil), kar ni bilo učinkovito za izvajanje analize s tema algoritmoma.

Rezultati za odločitvena drevesa

Opazili smo, da je najvišjo klasifikacijsko natančnost imel klasifikator odločitveno drevo, ki ga dobimo z algoritmom J48. Pri odločitvenih drevesih vemo, da so bolj porezana oziroma manjša odločitvena drevesa navadno bolj točna in tudi bolj razumljiva. Odločitveno drevo, ki ga je generiral algoritem J48 v Weki, je bilo avtomatično porezano s pesimistično metodo obrezovanja. Odločitveno drevo ima 482 listov in iz

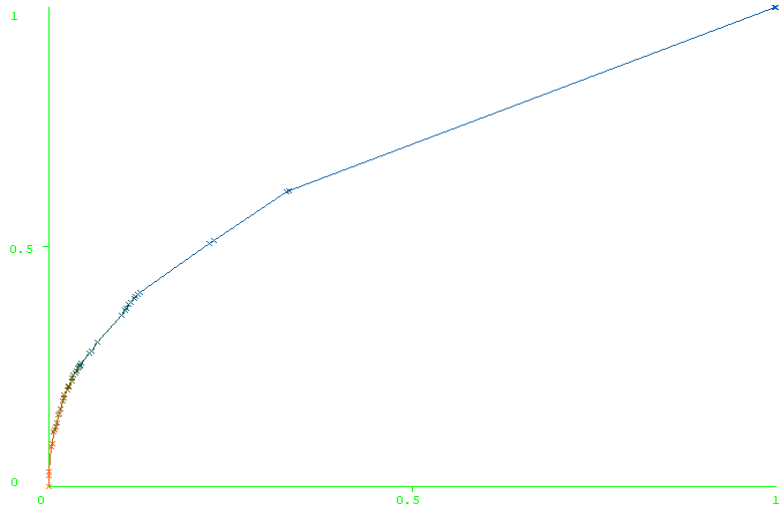
tega težko razberemo pravila za klasifikacijo primerov. Obstajajo tudi drugi načini, kako lahko drevo še bolj porežemo, tako da spremenimo nekatere parametre algoritma. Eden takšnih parametrov je minimalno število primerov, ki dosežejo list v odločitvenem drevesu. Parameter je v Weki avtomatično nastavljen na 2 primera. Ker imamo v naši podatkovni množici ogromno različnih podatkovnih primerov, to pomeni, da bomo najverjetneje dobili ogromno število listov, ki bodo lahko vsebovali le dva primera, kar pa ni najbolj uporabno. Zato smo se odločili, da bomo ta parameter nekoliko popravljali in spremljali rezultate, ki jih pri tem dobimo. Parameter minimalno število primerov v listu označimo z m. V tabeli 5 opazimo, da z višanjem minimalnega števila primerov v listih, zmanjšujemo število listov. Opazimo tudi, da se z večanjem parametra m do neke vrednosti zvišuje tudi klasifikacijska natančnost.

C4.5 - m	**Metoda vzorčenja**	**Točnost**	TP_{rate}	**AUC**	**Število listov**
2	Naključno vzorčenje	79.44	0.19	0.68	482
	10 - CV	79.66	0.19	0.68	48
50	Nakjučno vzorčenje	79.46	0.18	0.68	58
	10 - CV	79.65	0.21	0.69	58
100	Naključno vzorčenje	79.72	0.23	0.68	45
	10 - CV	79.68	0.21	0.68	45
250	Naključno vzorčenje	79.48	0.21	0.68	32
	10 - CV	79.52	0.21	0.68	32

Tabela 5: Rezultati C4.5 algoritma

Algoritem J48 smo na podatkih izvedli z različnimi vrednostmi parametra m. Za vrednosti, ki smo jih preizkusili, je imelo odločitveno drevo z $m = 100$ najvišjo klasifikacijsko natančnost (79.72 %). V tem primeru smo dobili odločitveno drevo s 45 listi, kar je razmeroma majhno, glede na velikost podatkovne množice. Ploščina pod krivuljo ROC je v tem primeru znašala 0.68.

V nadaljevanju je podrobneje opisano odločitveno drevo, ki ga dobimo z J48 pri $m = 100$. To odločitveno drevo ($m = 100$) v vozliščih testira 10 različnih atributov. Ti atributi so "fakturirana premija brez davka", "znesek popustov na polici", "likvidirane škode", "trajanje police", "starost vozila", "premijski razred AO", "prodajna pot", "škodni rezultat zavarovalca - motorna vozila", "škodni rezultat zavarovalca - vsi produkti" ter "starost stranke". Sklepamo, da na odločitev, ali bo stranka polico obnovila ali ne, najbolj vplivajo vrednosti naštetih atributov. Zavarovalnico bolj zanimajo tiste police, za katere odločitveno drevo napove, da ne bodo obnovljene. Poglejmo si nekatera pomembnejša pravila v odločitvenem drevesu, za katera je odločitveno drevo napovedalo, da polica ne bo obnovljena in za katera je vsaj 77% vseh primerov polic pravilno klasificiranih. Ta pravila lahko ločimo v dve skupini glede na vrednost atributa "fakturirana premija brez davka" (v nadaljevanju "fakturirana premija").



Slika 16: ROC krivulja za odločitveno drevo, ki je nastalo z algoritmom J48 ($m = 100$). Pri ROC krivulji x-os predstavlja FP_{rate}, y-os pa predstavlja TP_{rate}. AUC je v tem primeru enaka 0.68. Odtenki barv v krivulji se spreminjajo glede na vrednost praga.

1. Najnatančnejša pravila, za katera je "fakturirana premija"[3] $\leq$ 95.52:

   - **Če** [fakturirana premija $\leq$ 95.52] & [starost vozila $\leq$ 28 let] & [fakturirana premija $\leq$ 77.07] & [trajanje police = vsaj eno leto] & [znesek popustov na polici > -29.8] **potem** [obnovljena: NE].
     V podatkih je 433 takšnih primerov in približno 94% teh je pravilno klasificiranih.
   - **Če** [fakturirana premija $\leq$ 95.52] & [starost vozila $\leq$ 28 let] & [fakturirana premija $\leq$ 77.07] & [trajanje police = vsaj eno leto] & [znesek popustov na polici $\leq$ -29.8] & [premijski razred AO = 10.-14.] **potem** [obnovljena: NE].
     V podatkih je 137 polic, ki ustrezajo pogoju in približno 81% teh je pravilno klasificiranih.
   - **Če** [fakturirana premija $\leq$ 95.52] & [starost vozila $\leq$ 28 let] & [fakturirana premija $\leq$ 77.07] & [trajanje police = manj kot eno leto] & [premijski razred AO = 10.-14.] & [znesek popustov na polici > -18.62] **potem** [obnovljena: NE].
     V podatkih je 1228 takšnih primerov in približno 79% teh je pravilno klasificiranih.

   Za vsa tri zgodaj našteta pravila, za katera velja, da je "fakturirana premija" $\leq$ 95.51, vidimo, da je "starost vozila" $\leq$ "28 let" & "fakturirana premija" $\leq$ 77.07.

[3]Vrednosti atributov "fakturirana premija" ter "znesek popustov na polici" so podane v eurih

Prvi dve pravili imata še pogoj, da je "trajanje police" = "vsaj eno leto". Pri teh dveh pravilih pa je potem vse odvisno od vrednosti "zneska popustov na polici" in "premijskega razreda AO". V prvem pravilu velja, da če je "znesek popustov"> -29.8 (to pomeni, da se je začetni premiji odštelo manj kot 29.8), potem lahko že iz tega sklepamo, da oseba police najverjetneje ne bo obnovila. Morda je v tem primeru razlog za neobnovitev ta, da je znesek popustov na polici ob ostalih pogojih prenizek oziroma morda popusta sploh ni ali pa je potrebno k premiji še kaj doplačati. Če pa je "znesek popustov na polici" ≤-29.8 (to pomeni, da se je začetni premiji odštelo vsaj 29.8) in če je v tem primeru hkrati "premijski razred AO" = "10.-14.", je najverjetneje, da stranka police ne bo obnovila. Iz tega lahko morda sklepamo, da kljub visokemu znesku popusta (glede na plačano premijo), stranka meni, da spada v previsok premijski razred. Za tretje pravilo med zgoraj naštetimi pravili velja, da če je "trajanje police" = "manj kot eno leto" in poleg tega za to pravilo velja še, da je "premijski razred AO" = "10.-14." in "znesek popustov na polici" > - 18.62 (to pomeni, da se je začetni premiji odštelo manj kot 18.62) potem stranka police najverjetneje ne bo obnovila. V tem primeru je morda ključni razlog za neobnovitev police relativno nizek znesek popustov na polici.

2. Najnatančnejša pravila, za katera je "fakturirana premija" > 95.52:

   - **Če** [fakturirana premija > 95.52] & [znesek popustov na polici > 25.91] & [fakturirana premija ≤ 126.4] **potem** [obnovljena: NE].
     V podatkih je 191 takšnih primerov in približno 90% jih je pravilno klasificiranih.
   - **Če** [fakturirana premija > 95.52] & [znesek popustov na polici ≤ 25.91] & [premijski razred AO = 5.-9.] **potem** [obnovljena: NE].
     V podatkih je 7607 takšnih primerov in 77% teh je pravilno klasificiranih.

   V tej skupini pravil so tista pravila, za katera velja pogoj, da je "fakturirana premija" > 95.52. Če velja še, da je "znesek popustov na polici" > 25.91 (pozitiven znesek popusta lahko pomeni, da je bila polica stornirana ali pa, da je bilo potrebno kakšno doplačilo) ter "fakturirana premija" ≤ 126.4, lahko sklepamo, da polica najverjetneje ne bo obnovljena.

V nadaljevanju so našteta še nakatera pravila v istem odločitvenem drevesu, ki določajo obnovljene police.

1. Najnatančnejše pravilo (za obnovljene police), za katero je "fakturirana premija" ≤ 95.52:

   - **Če** [fakturirana premija ≤ 95.52] & [starost vozila > 28] **potem** [obnovljena: DA].
     V podatkih je 414 takšnih primerov in približno 77% teh je pravilno klasificiranih.

   Edino pravilo, ki smo ga navedli za obnovljene police, pri čemer je "fakturirana premija" ≤ 95.52, je zelo enostavno in pravi, da če je v tem primeru vozilo starejše od 28 let, bo polica najverjetneje obnovljena.

2. Najnatančnejše pravilo (za obnovljene police), za katero je "fakturirana premija" > 95.52:

   - **Če** [fakturirana premija > 95.52] & [znesek popustov na polici ≤ 25.91] & [premijski razred AO = 1.-4.] **potem** [obnovljena: DA].
     V podatkih je 45506 takšnih primerov in približno 85% teh je pravilno klasificiranih.

   Pravilo, ki smo ga navedli za obnovljene police, pri čemer je "fakturirana premija" > 95.52, ima zraven še dva pogoja in sicer mora veljati, da je "znesek popustov na polici" ≤ 25.91 ter "premijski razred AO"= "1.-4.". Ob teh pogojih lahko sklepamo, da bo polica najverjetneje obnovljena. To pravilo pokriva daleč največ primerov v odločitvenem drevesu. Zanimivo pa je to, da to pravilo lahko primerjamo z zadnjim navedenim pravilom za neobnovljene police in vidimo, da so pogoji pri obeh enaki z izjemo zadnjega pogoja, ki določa vrednost "premijskega razreda AO".

Pravila, ki jih je generiralo odločitveno drevo so zelo različna. Ko ta pravila dobimo, jih je potrebno na razumljiv način predstaviti zavarovalnim analitikom, ki se odločijo, ali bodo pravila lahko uporabili pri svojem poslovanju ali ne. Potrebno je določiti neko mejo oziroma stopnjo natančnosti pravila, pri kateri se uporabniki odločijo ali pravilo lahko upoštevajo ali ne.

6 ZAKLJUČEK

Zavarovalnice, kot tudi ostala velika podjetja, se pri svojem delu vsakodnevno srečujejo z vprašanji, na katera lahko odgovorijo z analiziranjem podatkov, ki jih pridobivajo pri poslovanju s strankami. Nekateri problemi, ki jih rešujejo zavarovalnice, od zavarovalnih tehnologov zahtevajo podrobnejše analize, pri katerih ne zadostuje le uporaba standardnih statističnih orodij. V tem primeru si lahko pomagajo z raznimi algoritmi strojnega učenja, ki so v zadnjem času precej pogosta metoda, ko pride do analiziranja podatkov. V magistrskem delu smo spoznali problem obnovitev zavarovalnih polic avtomobilskih zavarovanj in tega skušali rešiti z uporabo klasifikacijskih tehnik podatkovnega rudarjenja. Pri tem smo uporabljali CRISP-DM metodologijo s ciljem, da bi poiskali čim bolj natančen klasifikator, ki bi napovedal, ali je bolj verjetno, da bo zavarovalna polica ob poteku zavarovalnega leta obnovljena ali ne. Uporabili smo Naivni Bayesov klasifikator, odločitvena drevesa, klasifikacijska pravila generirana z algoritmom PART ter metodo podpornih vektorjev. Na podatkovni množici, ki smo jo uporabljali za analiziranje problema z obnovitvami zavarovalnih polic, je bilo približno 77% polic, ki so bile obnovljene in preostalih 23%, ki niso bile obnovljene. Izkazalo se je, da je najvišjo natančnost dosegel klasifikator odločitveno drevo in sicer okrog 80%. Klasifikacijska natančnost odločitvenih dreves na prvi pogled ni ravno najboljša, vendar pa je vseeno boljša kot natančnost izhodiščnega modela (Zero-R). Če želimo pravila, ki jih generira odločitveno drevo, uporabiti tudi pri poslovanju, je potrebno ta dobro pregledati in v sodelovanju z zavarovalnimi strokovnjaki oceniti, ali so ta dovolj natančna za uporabo pri poslovanju. Znanje, pridobljeno iz klasifikacijskih pravil, bi zavarovalnice lahko uporabile pri prodaji zavarovanj. Zavarovalnice bi morale biti bolj pozorne na tiste primere polic, za katere klasifikator napove, da najverjetneje ne bodo obnovljene. Pri takšnih primerih bi oseba, ki je odgovorna za prodajo police, morala ukrepati preden bi prišlo do poteka zavarovalne police in stranki (v kolikor je to možno) ponuditi ugodne pogoje za nadaljevanje zavarovanja.

Pri uporabi algoritmov strojnega učenja se je potrebno zavedati, da so poleg podatkov, ki jih imamo na razpolago, skoraj pri vsakem problemu prisotni tudi nekateri ostali dejavniki, ki vplivajo na problem, ki ga obravnavamo. Zaradi tega je včasih težko pričakovati, da lahko samo iz razpoložljivih podatkov uspemo dobiti zelo natančen in uporaben model. V primeru obnavljanja zavarovalnih polic pravzaprav napovedujemo

odločitev posamezne stranke, ali bo polico obnovila ali ne. Takšne vrste odločitev vsebuje tudi subjektivni človeški faktor, za katerega nimamo podatka. Poleg človeškega faktorja, bi bilo za analizo takšnega problema morda koristno, če bi imeli na razpolago tudi podatke o premijah ostalih konkurenčnih zavarovalnic, saj je velikokrat razlog za neobnovo zavarovanja ta, da želi stranka skleniti zavarovanje pri kateri drugi konkurenčni zavarovalnici, ki ji nudi boljše pogoje zavarovanja.

7 LITERATURA

[1] R. C. Barros, A. C. P. L. F. de Carvalho in A. A. Freitas, Decision-Tree Induction, v: *Automatic Design of Decision-Tree Induction Algorithms*, Springer International Publishing, 2015, 7–45. *(Citirano na strani 25.)*

[2] J. Boncelj, *Zavarovalna ekonomika*, Obzorja, Maribor, 1983. *(Citirano na strani 3.)*

[3] S. Boyd in L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, 2004. *(Citirano na straneh 2 in 44.)*

[4] C.J.C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, *Data Mining and Knowledge Discovery* **2** (1998) 121–167. *(Citirano na strani 44.)*

[5] P. Chapman, J. Clinton, T. Khabaza, T. Reinarts, C. Shearer in R. Wirth, *CRISP-DM 1.0: Step-by-step data mining guide*, SPSS, The CRISP-DM consortium, 2000. *(Citirano na straneh 2 in 12.)*

[6] B. Chen, *Maximum Likelihood Estimation*, `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.713.4802&rep=rep1&type=pdf`. (Datum ogleda: 10. 10. 2018.) *(Citirano na straneh 22 in 24.)*

[7] M. Erritali, H. Ezzikouri, B. Hssina in A. Merbouha, A comperative study of decision tree ID3 and C4.5, *(IJACSA) International Journal of Advanced Computer Science and Applications,* Special Issue on Advances in Vehicular Ad Hoc Networking and Applications (2014). *(Citirano na strani 26.)*

[8] E. Frank, M. A. Hall in I. H. Witten, *Data Mining: Concepts and Techniques, 3. izdaja*, Morgan Kaufmann Publishers, Massachusetts, 2012. *(Citirano na straneh 2, 7, 17, 26, 37, 41 in 53.)*

[9] E. Frank in I.H. Witten, Generating Accurate Rule Sets Without Global Optimization, v: *Proceedings of the Fifteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers, San Francisco, 1998, 144–151. *(Citirano na strani 37.)*

[10] T.P. Frece, *Osnove zavarovalništva*, Konzorcij višjih strokovnih šol za izvedbo projekta IMPLETUM, Ljubljana, 2011. *(Citirano na strani 3.)*

[11] D. J. Hand in K. Yu, Idiot's Bayes: Not so Stupid after All?, *International Statistical Review* **69**(3) (2001) 385-398. *(Citirano na strani 24.)*

[12] T. Hastie, G. James, R. Tibshirani in D. Witten, *An Introduction to Statistical Learning: with Applications in R*, Springer-Verlag, New York, 2013. *(Citirano na strani 17.)*

[13] G.H. John in P. Langley, Estimating Continuous Distributions in Bayesian Classifiers, v: *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann Publishers, San Mateo, 1995, 338–345. *(Citirano na straneh 2 in 23.)*

[14] V. Kecman, Support Vector Machines - An Introduction, v: *Support Vector Machines: Theory and Applications*, Studies in Fuzziness and Soft Computing, Volume 177, Springer, Berlin, Heidelberg, 2005, 1–47. *(Citirano na straneh 2, 41 in 45.)*

[15] R. Kohavi in R. Quinlan, Decision Tree Discovery, v: *Handbook of data mining and knowledge discovery*, Oxford University Press, New York, 2002, 267–276. *(Citirano na straneh 2 in 26.)*

[16] D. MacKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, New York, 2003. *(Citirano na strani 30.)*

[17] T.M. Mitchell, Generative and discriminative classifiers: Naive bayes and logistic regression, v *Machine Learning*, Draft of September 23, 2017. *(Citirano na straneh 2 in 19.)*

[18] T.M. Mitchell, *Machine Learning*, McGraw-Hill, New York, 1997. *(Citirano na straneh 2 in 19.)*

[19] J. M. Moguerza in A. Munoz, Suport Vector Machines with Applications, *Statistical Science* **21**(3) (2006) 322-336. *(Ni citirano.)*

[20] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, The MIT Press, Cambridge, Massachusetts, 2012. *(Citirano na straneh 21 in 22.)*

[21] A. Ng, *CS229 Lecture notes*, `https://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf`. (Datum ogleda: 1. 9. 2018.) *(Citirano na straneh 2, 41 in 45.)*

[22] J. R. Quinlan, Induction Of Decision Trees, *Machine learning* **1** (1986) 81-106. *(Citirano na strani 2.)*

[23] C. E. Shannon, A Mathematical Theory of Communication, *The Bell System Technical Journal* **27** (1948) 379–423, 623–656. *(Citirano na straneh 2, 29 in 30.)*

[24] J. Thickstun, *Mercer's theorem*, `https://homes.cs.washington.edu/~thickstn/docs/mercer.pdf`. (Datum ogleda: 26. 12. 2018.) *(Citirano na strani 50.)*

[25] *Bayes theorem*, `https://en.wikipedia.org/wiki/Bayes'_theorem`. (Datum ogleda: 5. 9. 2018.) *(Citirano na strani 19.)*

[26] *Decision Tree Pruning based on Confidence Intervals (as in C4.5)* , `www.cs.bc.edu/~alvarez/ML/statPruning.html`. (Datum ogleda: 6. 1. 2019.) *(Citirano na strani 35.)*

[27] *Kaj pomeni kasko zavarovanje cestnih vozil (AK)?*, `https://www.zav-zdruzenje.si/vprasanja/kaj-pomeni-kasko-zavarovanje-cestnih-vozil-ak/`. (Datum ogleda: 3. 12. 2018.) *(Citirano na strani 5.)*

[28] *Opisi zavarovanj*, `https://www.zav-zdruzenje.si/sredisce-informacij/zavarovalne-vrste/`. (Datum ogleda: 27. 10. 2018.) *(Citirano na strani 5.)*

[29] *Slovensko zavarovalno združenje*, `https://www.zav-zdruzenje.si/sredisce-informacij/slovar-zavarovalnih-izrazov/`. (Datum ogleda: 27. 10. 2018.) *(Citirano na straneh 4 in 6.)*

[30] *Zakon o obveznih zavarovanjih v prometu (ZOZP)*, `https://www.uradni-list.si/glasilo-uradni-list-rs/vsebina/82582`. (Datum ogleda: 13. 12. 2018.) *(Citirano na strani 5.)*

[31] *ZAKON O ZAVAROVALNIŠTVU (ZZavar-1)*, `https://www.uradni-list.si/glasilo-uradni-list-rs/vsebina/12419`. (Datum ogleda: 20. 8. 2018.) *(Citirano na strani 5.)*

[32] *ZAVAROVANJA MOTORNIH VOZIL 2018*, `http://zmv.zav-zdruzenje.si/index.html`. (Datum ogleda: 15. 10. 2018.) *(Citirano na straneh 4 in 6.)*