UNIVERZA V MARIBORU

FAKULTETA ZA NARAVOSLOVJE IN MATEMATIKO

Oddelek za matematiko in računalništvo

DIPLOMSKO DELO

Sara Colja

Maribor, 2011

UNIVERZA V MARIBORU

FAKULTETA ZA NARAVOSLOVJE IN MATEMATIKO

Oddelek za matematiko in računalništvo

Diplomsko delo

ALGORITMI IN TEHNIKE PODATKOVNEGA RUDARJENJA NA BAZI PROCESNIH PARAMETROV

Mentor:

doc. dr. Dominik Benkovič

Somentor:

dr. Peter Cvahte

Kandidatka:

Sara Colja

Maribor, 2011

ZAHVALA

Citat
Uspeh ni nikoli dokončen,
neuspeh nikoli usoden.
(Bob Dore)

Zahvaljujem se mentorju doc. dr. Dominiku Benkoviču in somentorju dr. Petru Cvahtetu za nasvete, strokovno pomoč in vodenje pri nastajanju in oblikovanju diplomskega dela.

Posebna hvala tudi odgovornim v podjetju Impol, ki so mi omogočili pridobivanje potrebnih podatkov in informacij za praktični del diplomskega dela.

Nenazadnje pa hvala tudi mojim staršem za pomoč, podporo in potrpežljivost v času študija.

Vsem iskreno hvala.

UNIVERZA V MARIBORU

FAKULTETA ZA NARAVOSLOVJE IN MATEMATIKO

IZJAVA

Podpisana Sara Colja, rojena 08. januar 1988, študentka Fakultete za naravoslovje in matematiko Univerze v Mariboru, študijskega programa nepedagoška matematika, izjavljam, da je diplomsko delo z naslovom

ALGORITMI IN TEHNIKE PODATKOVNEGA RUDARJENJA NA BAZI PROCESNIH PARAMETROV

pri mentorju doc. dr. Dominiku Benkoviču avtorsko delo. V diplomskem delu so uporabljeni viri in literatura korektno navedeni; teksti niso uporabljeni brez navedbe avtorjev.

Maribor, 12. april 2011

Sara Colja

Algoritmi in tehnike podatkovnega rudarjenja na bazi procesnih parametrov

program diplomskega dela

V diplomskem delu obravnavajte algoritme in tehnike podatkovnega rudarjenja na bazi procesnih in tehnoloških parametrov. Preverite različne rešitve in predstavite njihovo učinkovitost v smislu iskanja povezav vplivov na kvaliteto izdelkov in uporabe za industrijske aplikacije. Na primeru analizirajte vpliv kemijske sestave na mehanske lastnosti aluminijaste zlitine.

Osnovni viri:

1. C. Baragoin, C. M. Andersen, S. Bayerl, G. Bent, J. Lee, C. Schommer, *Mining Your Own Business in Retail*, Redbooks, California, 2001.

2. M. J. A. Berry, G. Linoff, *Mastering Data Mining: The Art and Science of Customer Relationship Management*, Wiley Computer Publishing, Canada, 2000.

3. J. Reinschmidt, R. Bhattacharya, P. Harris, A. Karanasos, *Mining Relational and Nonrealational Data with IBM Intelligent Miner for Data Using Oracle, SPSS, and SAS As Sample Data Source*, Redbooks, California, 1998.

4. I. H. Witten, E. Frank , *Data Mining: Practical Machine Learning Tools and Techniques, Second edition*, Morgan Kaufmann Publishers, San Francisco, 2005.

doc. dr. Dominik Benkovič

dr. Peter Cvahte

COLJA, S.: Algoritmi in tehnike podatkovnega rudarjenja na bazi procesnih parametrov.
Diplomsko delo, Univerza v Mariboru, Fakulteta za naravoslovje in matematiko, Oddelek za matematiko in računalništvo, 2011.

IZVLEČEK

V diplomskem delu je predstavljen proces podatkovnega rudarjenja, njegovi algoritmi, tehnike in uporaba v praksi. V prvem delu se seznanimo s teorijo podatkovnega rudarjenja. Omenjene so tehnike podatkovnega rudarjenja in nekateri najbolj znani algoritmi. Podrobneje je predstavljen algoritem nevronskih mrež, ki se uporabi v praktičnem primeru. V drugem delu je po korakih splošne metode podatkovnega rudarjenja, predstavljene v prvem delu, predstavljen konkreten poslovni problem, ki ga rešujemo s podatkovnim rudarjenjem.

Na bazah podatkov podjetja Impol sta zgrajena modela za iskanje povezav med kemijsko sestavo zlitine EN AW-7075 (interna oznaka PD30) in njenimi mehanskimi lastnostmi. Po združitvi različnih baz in agregiranju podatkov je bilo uporabljenih 675 množic zgodovinskih podatkov za zlitino PD30. Model je bil zgrajen z orodjem SPSS Modeler, s feed-forward nevronsko mrežo in vzvratnim širjenjem napake. Naučeni nevronski mreži napovedujeta mehanske lastnosti *napetost tečenja ($R_{0,2}$), natezna trdnost (R_m)* in *raztezek (A)*, kot funkcijo procesnih parametrov. Točnost napovedi modela nevronske mreže za napetost tečenja je $84,8\%$, točnost napovedi modela za natezno trdnost in raztezek pa $91,8\%$.

S predstavljenima modeloma nevronskih mrež je pokazano, da lahko podjetje Impol razvije model za ocenjevanje končnih mehanskih lastnosti, kot funkcijo procesnih parametrov. S tem je omogočena optimizacija procesne poti glede na produktivnost in kvaliteto.

Ključne besede: podatkovno rudarjenje, nevronska mreža,
odločitvena drevesa, vzvratno širjenje napake,
metoda padajočih gradientov.

Math. Subj. Class. (2010): 62M45, 62-07, 68T05.

COLJA, S.: Data Mining Techniques and Algorithms on Database of Process Parameters.
Graduation Thesis, University of Maribor, Faculty of Natural Scicences and Mathematics, Department of Mathematics and Computer Science, 2011.

ABSTRACT

In the graduation thesis we presented the data mining process, data mining techniques and algorithms on database of process parameters. The thesis begins with a short introduction of data mining process. We described the best know techniques and algorithms of data mining. In more details we presented algorithms of neural networks, wich we used to work on practical business problem. In the second part is described through the generic data mining method an alternative approach to the physical modeling, the artificial intelligence approach, based on the neural networks.

Data for data mining process were collected in company Impol. After merge different database and aggregate data, 675 sets of complete history data were collected for alloy EN AW-7075 (internal use Impol as PD30). For building the neural network model we used SPSS tool, with one of most popural architecture Multilayer Feedforward with Backpropagation learning. This neural networks are capable of predicting *yield strength ($R_{0,2}$), tensile strength (R_m)* and *elongation (A)* as function of process parameters. The accuracy of neural network model for yield strength is $84,8\%$, the accuracy of neural network model for tensile strength and elongation is $91,8\%$.

With the represented models of neural network we show, that the company Impol can develop a model for estimation of the final product properties as a function of the process parameters. This allows optimizing the process path with respect to productivity and quality in the perspective.

Keywords: data mining, neural networks,
decision trees, backpropagation learning,
method of gradient descent.

Math. Subj. Class. (2010): 62M45, 62-07, 68T05.

Kazalo

Uvod

Poslovni svet in poslovanje v le-tem postaja vse bolj kompleksno. Potrebna je iniciativnost in nenehno izpopolnjevanje, da se kljubuje konkurenčnosti na trgu, hkrati pa se prihrani in zagotovi kakovost storitev. Nenehno se povečuje količina podatkov s katerimi podjetja poslujejo, povečuje se število poslovnih spremenljivk, ki vplivajo na poslovne odločitve, postopki odločanja pa so čedalje bolj zapleteni in nepredvidljivi. Tako se pojavi vprašanje, kako koristno uporabiti podatke, ki se leta skladiščijo, hkrati pa veljajo za poslovno skrivnost, saj so konkurenci nedostopni.

V zadnjih desetletjih, ko je razvoj računalništva in informatike bistveno napredoval, se je začela razvijati umetna inteligenca, ena vodilnih znanosti v informacijski tehnologiji. Področje umetne inteligence, ki se ukvarja s podatki je podatkovno rudarjenje. Rudarjenje velja za ključen korak v procesu odkrivanja zakonitosti v podatkih. Gre za uporabo specializiranih algoritmov, s katerimi iz zbrane množice podatkov izluščimo zanimive ter uporabne vzorce in modele.

Z ozirom na problematiko neizkoriščenosti velikih podatkovnih baz podjetij je tema diplomske naloge podatkovno rudarjenje in njegova uporaba v praksi. Naloga je razdeljena na dva dela - teoretični in praktični del. V teoretičnem delu je predstavljen proces podatkovnega rudarjenja, njegove tehnike in algoritmi. Podrobneje je opisan algoritem, ki se kasneje uporabi v praktičnem delu in pa splošna metoda vpeljave rudarjenja v prakso. V praktičnem delu je obravnavan primer podatkovnega rudarjenja za podjetje Impol. To je podjetje, ki se ukvarja s predelovanjem aluminija v izdelke in polizdelke. V njihovi bazi podatkov so skladiščeni vsi podatki naročil, izdelave in testiranja kakovosti izdelkov. Iz danih podatkov smo iskali zvezo med mehanskimi lastnostmi zlitine in njeno kemijsko sestavo. Problem smo reševali z algoritmom nevronskih mrež s pomočjo katerega smo dobili enačbo, na podlagi katere lahko napovemo mehanske lastnosti zlitine, če poznamo njene kemijske lastnosti.

Poglavje 1

Podatkovno rudarjenje

Podatki so eden najbolj uporabnih stranskih produktov računalniškega napredka, ki s svojim obsegom, željo po zniževanju stroškov in hkratno izredno rastjo zmogljivosti računalniške opreme, povzroča nenehno rast števila in velikosti podatkovnih baz. Gradnja in vzdrževanje podatkovnih baz zahteva veliko časa in truda. Kljub temu pa se le redko zavedamo prave vrednosti in polnega potenciala teh dragocenih virov, saj večini organizacij primanjkuje poznavanje učinkovite pretvorbe razpoložljivih podatkov v uporabno znanje.

Za reševanje poslovnih problemov proizvodnje ter za pregled podatkov v podatkovni bazi se običajno uporabljajo statistična orodja. Delujejo na podlagi uporabe tehnik, ki povzemajo podatke v statistične ocene, ki jih je mogoče razložiti ne da bi poznali podrobnosti vsakega podatka podatkovne baze. Razlaga pridobljenih rezultatov zahteva določeno mero statističnega znanja. Običajno za preučevanje podatkov uporabljamo statistične metode kot so *korelacijska analiza*, *faktorska analiza* in *regresijska analiza.*

V nasprotju s statistično analizo podatkovno rudarjenje analizira vse relevantne podatke iz podatkovne baze in povzame skrite vzorce. Podatkovno rudarjenje je razširitev statistične analize, ki temelji na tehnikah in disciplinah uporabljenih v statistični analizi. Algoritmi podatkovnega rudarjenja avtomatizirajo časovno zamudne postopke, skozi katere bi morali, če bi želeli priti do istih rezultatov s klasično statistično analizo.

1.1 Kaj je podatkovno rudarjenje?

Podatkovno rudarjenje lahko smatramo kot vejo matematike, ki se v praksi uporablja za pridobivanje koristnih informacij in znanj, predvsem za reševanje konkretnih poslovnih problemov. Zaradi raznolikosti uporabe še vedno nimamo enotne definicije podatkovnega rudarjenja. Navedimo nekaj definicij, ki jih najdemo v literaturi:

Definicija 1.1 *(Vir: [9])*
Podatkovno rudarjenje je krovni pojem, ki opisuje proces odkrivanja vzorcev, asociacij, sprememb, anomalij ter statistično signifikantnih struktur in dogodkov v podatkih.

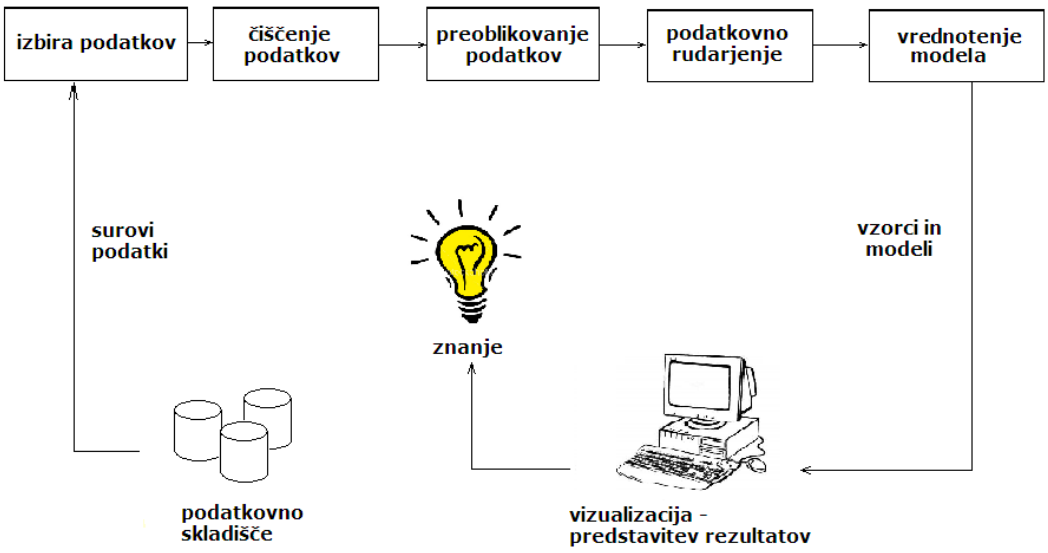
Definicija 1.2 *(Vir: [1])*
Podatkovno rudarjenje je raziskovanje in analiziranje velike količine podatkov z avtomatičnimi ali polavtomatičnimi postopki, za odkrivanje predhodno neznanih, zanimivih odvisnosti.

Definicija 1.3 *(Vir: [1])*
Podatkovno rudarjenje je netrivialen izvleček natančnih, prej neznanih, potencialno uporabnih informacij iz podatkov.

Definicija 1.4 *(Vir: [8])*
Podatkovno rudarjenje je sistematično iskanje informacij v veliki količini podatkov.

Podatkovno rudarjenje kot ga poznamo danes ni staro več kot 20 let, njegove začetke pa lahko zasledimo že v zgodnjem razvoju umetne inteligence v letih 1950. V tem obdobju se je razvilo prepoznavanje vzorcev in sklepanje iz vzorcev, kar je temeljni gradnik podatkovnega rudarjenja. Kot znanstvene aplikacije so se uporabljale tehnike, ki jih danes poznamo pod imenom podatkovno rudarjenje. V zadnjih letih uporaba podatkovnih baz narašča skupaj s potrebo po razumevanju in interpretaciji teh podatkov. Dostopnost do relativno poceni računalnikov je povzročila eksplozijo v uporabi podatkovnega rudarjenja v najrazličnejše poslovne namene.

Osnovni proces podatkovnega rudarjenja prikazuje slika 1.1. Na sliki je proces rudarjenja prikazan sekvenčno, vendar pa se celoten proces odkrivanja znanja zmeraj odvija iterativno. Včasih ga je treba večkrat ponoviti in se iz katere faze vrniti nazaj v predhodno. Odkrivanje koristnih informacij v večini primerov zahteva uporabo različnih metod, vsaka posamezna metoda pa še več poskusov. Proces podatkovnega rudarjenja bo podrobneje opisan v poglavju 5.



Slika 1.1: Osnovni proces podatkovnega rudarjenja

Omenimo še razliko med podatkovnim rudarjenjem in razpoznavanjem vzorcev. *Razpoznavanje vzorcev* je usmerjeno v konstrukcijo točno določenega modela razvrščanja - klasifikatorja. Model razvrščanja je preslikava med množico vhodnih spremenljivk (atributov) $x_1, x_2, ..., x_n$ in izhodno spremenljivko y. Vrednost te preslikave predstavlja eno izmed možnih odločitev oziroma odločitvenih razredov $\omega_1, ..., \omega_m$. Pri tem ni pomemben način pridobitve znanja. Klasifikator ne poda skoraj nič ali pa izredno malo vpogleda v svoj način sprejemanja odločitev. Običajno so pravila odločanja vgrajena v strukturo in je zato ugotavljanje pravil po katerih deluje izredno zapleteno. Pri *podatkovnem rudarjenju* pa odkrivanje znanja ni le sama klasifikacija. Hkrati uporabnik odkriva tudi smiselno znanje s katerim se pride do predlagane odločitve. Torej modeli, ki se uporabljajo za odkrivanje znanj, predstavljajo pravila odločanja v obliki, ki uporabniku omogočajo preverjanje predlaganih odločitev in izvedbo nadaljnjih analiz.

1.2 Vzorci

Vzorci so opisani s karakterističnimi lastnostmi - atributi $x = (a_1, ..., a_t)$. Atributi so lahko nominalni, diskretni ali zvezni. Diskretni in zvezni podatki vsebujejo urejenost med vredno-

stmi in se uvrščajo pod ordinalni podatkovni tip, nominalni podatki ne vsebujejo takšne ureditve. Diskretne in zvezne podatke opišemo z intervali v zveznem spektru vrednosti. Število diskretnih podatkov je ponavadi omejeno na končno množico, medtem ko je lahko zveznih podatkov neskončno veliko.

Vzorec je iz geometrijskega vidika točka v t-dimenzionalnem prostoru. Razdelitev množice glede na razrede ustreza definiranju odločitvenih hiperploskev ali več hiperploskvam v tem prostoru.

Množica podatkov se razdeli na učno množico U in testno množico T. Algoritem pridobivanja znanja ima dostop le do učne množice vzorcev s pomočjo katere mora ustvariti hipotezo. Testna množica se uporablja izključno v namene testiranja kakovosti dobljene hipoteze in podaja predvideno predikcijsko natančnost za še nevidne vzorce. Zato je zaželeno, da je $U \cap T = \oslash$.

1.2.1 Diskretizacija

Kot že rečeno so lahko podatki katere preučujemo podani v različnih oblikah, najpogosteje v obliki diskretnih ali zveznih vrednosti. Določene metode za konstrukcijo klasifikatorja niso sposobne sprejemati zveznih podatkov zato je v takšnem primeru potrebno vse zvezne vrednosti preslikati v diskretni prostor, čemur pravimo *diskretizacija.*

Vrednosti, ki jih lahko zavzema zvezni atribut so običajno opisane kot interval vrednosti $[SpodnjaMeja, ZgornjaMeja]$. Ta interval vrednosti se razdeli na podintervale na katere nato gledamo kot na disketne vrednosti. Določitev mej podintervalov pa je lahko ključna za uspešnost metode.

Metode diskretizacije

Učinkovitost posamezne metode ocenimo glede na *enostavnost, konsistenčnost in natančnost.* V idealnem primeru je diskretizacija najkvalitetnejša v vseh treh pogojih, v praksi pa je izbira ocenitve metode odvisna tudi od potreb in zahtev uporabnika.

- **Ekvidistančna diskretizacija** - kreiranje vnaprej določenega števila enako velikih podintervalov. Če je interval razdeljen na k enakih delov je znotraj intervala $k-1$ delitvenih točk, ki so meje podintervalov.
- **Diskretizacija intervala glede na frekvenco vzorcev** - interval se razdeli na vnaprej določeno število podintervalov pri čemer vsak izmed njih vsebuje enako število vzorcev.

Obe opisani metodi ne upoštevata porazdelitvene vrednosti atributa. Temu se izognemo, če uporabimo katero od naslednjih metod:

- **Pragovna diskretizacija** - interval vrednosti se razdeli na dva dela, pri tem se upošteva lega učnih vzorcev na tem intervalu. Učni vzorci morajo biti razvrščeni po velikosti glede na vrednost določenega atributa. Oznaka za vrednost zveznega atributa pri i-tem učnem vzorcu v urejenem zaporedju je a_i, a_{i+1} pa je vrednost zveznega atributa pri $(i+1)$ učnem vzorcu v zaporedju. Kandidati za prag so vse točke t_i, ki ležijo na aritmetični sredini a_i in a_{i+1}, kjer i teče od 1 pa do števila učnih vzorcev zaporedja, urejenega po naraščajoči vrednosti zveznega atributa. V zanki se preverijo vsi kandidati za prag, ki se ocenijo s funkcijo podobno metriki čistosti.

$$t_i = \frac{a_i + a_{i+1}}{2}$$

- **Dinamična diskretizacija** - gre za dinamično prilagajanje pozicijam učnih vzorcev v intervalu. Čezmernemu prilagajanju učnih vzorcev se izognemo z ustrezno nastavitvijo parametrov - *% tolerance*, ki pove koliko učnih vzorcev enega razreda se lahko nahaja v podintervalu drugega razreda in *faktorjem z*, ki pove kolikokrat več mora biti učnih vzorcev v zunanjih dveh podintervalih trojčka, da lahko združimo tri podintervale v enega.

1.2.2 Manjkajoči podatki

Pri zbiranju podatkov se lahko v realnem okolju zgodi, da v določenem trenutku ni mogoče izmeriti vseh vrednosti atributov vzorca. Konkretne meritve so lahko nesmiselne, nekatere pa nenamerno izpuščene (šum). Tako pride do manjkajočih podatkov v vzorcu. Večina metod učenja znanja v osnovi ni sposobna obdelati manjkajočih vrednosti zato ta vzorec preprosto izpustijo. To pa lahko privede do neuporabnosti zbranih vzorcev saj se lahko zgodi, da v vsakem vzorcu manjka kakšna vrednost atributa. Zato se pred uporabo metode učenja znanja običajno izvede pretvorba manjkajočih podatkov na enega od sledečih načinov:

- Manjkajoča vrednost atributa se zamenja s srednjo vrednostjo, ki ima enak standardni odklon atributa kot v ostalih vzorcih;

  Slabost: direktna korelacija med odločitvenim razredom in izračunano vrednostjo

- Manjkajoča vrednost atributa se izračuna s pomočjo regresije;

  Slabost: kompleksnost uporabe regresije

- Manjkajoča vrednost se smatra za posebno vrednost, ki se lahko uporabi kot pogoj pri predikciji.

  Slabost: možnost povečanja kompleksnosti rešitve

1.3 Vrednotenje modela

Glede na vzorec podatkov učne množice U in glede na izbiro algoritma dobimo *model.* To so pravila za napovedovanje, ki smo jih pridobili iz učnih podatkov. Pri gradnji modela je ključnega pomena enostavnost modela saj so takšni modeli običajno najbolj smiselni in splošni, prav tako pa jih je najlažje interpretirati. To zadeva princip Ockhamove britve, ki med več modeli, ki ustrezajo danim podatkom, vedno izbere tistega, ki je najbolj enostaven med njimi. Da bi lahko odkrili skrite relacije med podatki je bistvenega pomena zgraditi model, ki je na podlagi vrste neodvisnih spremenljivk (atributov) sposoben določiti vrednost odvisne spremenljivke in pri tem minimalizirati napako odločanja.

Za določitev kvalitete modela se meri natančnost modela. Merjenje natančnosti se izvede s statističnimi metodami, ki povedo koliko je model točen za zgodovino - učne podatke. Kot rezultat dobimo *stopnjo zanesljivosti,* ki nam pove v kolikšni meri lahko zaupamo modelu v napovedi svežih podatkov. Za določitev kvalitete modela je ključnega pomena testiranje zgrajenega klasifikatorja, ki se izvede nad podatki, ki niso bili vključeni v procesu učenja.

Delitev osnovne baze podatkov na učno in testno množico je naključna, običajno v razmerju *70% podatkov za učenje, 30% podatkov za testiranje.* Hkrati pa je pomembna porazdelitev odločitvenih razredov v učni množici. Najpogosteje se uporablja naravna porazdelitev, v učni množici se obdrži enaka zastopanost učnih razredov (v odstotkih), kot se pojavi v celotni bazi podatkov.

1.3.1 Indikatorji kvalitete klasifikatorja

Natančnost klasifikatorja

Natančnost klasifikatorja je verjetnost pravilne klasifikacije vzorca. Določamo jo ločeno na učni in testni množici. Natančnost na učni množici pove v kolikšni meri se je klasifikator uspel naučiti danega koncepta. Ključne informacije o natančnosti napovedovanja pa nam poda natančnost klasificiranja testnih vzorcev. Natančnost izračunamo kot

$$natančnost = \frac{število\ pravilno\ klasificiranih\ objektov}{število\ vseh\ objektov}$$

$$StopnjaNapake = 1 - natančnost$$

Skupna natančnost klasifikatorja ni zadosten indikator kvalitete naučenega. Lahko se namreč zgodi, da naučen klasifikator klasificira testne objekte z visoko skupno natančnostjo, natančnost klasificiranja določenega odločitvenega razreda pa je zelo nizka. Zato se pogosto računa *povprečna natančnost*, ki pove mnogo več o kvaliteti. Povprečno natančnosti izračunamo kot

$$natančnost_c = \frac{\textit{število pravilno klasificiranih objektov razreda } c}{\textit{število vseh objektov razreda } c}$$

$$PovprečnaNatančnostRazredov = \frac{\sum_i natančnost_i}{\text{število razredov}}$$

Kvaliteto klasifikatorja grafično predstavimo z grafi vrednotenja. Ti vizualno prikažejo obnašanje modela pri napovedovanju rezultatov. Podatke sortirajo na podlagi napovedane vrednosti in stopnje zanesljivosti modela. Razbijejo jih v skupine enakih velikosti (kvartile) in nato določijo vrednosti posameznega kvartila glede na kriterij določanja (vrsto grafa). Rezultati se shranijo kot posebna vrednost oziroma razpon vrednosti, ki kažejo uspešnost napovedi.

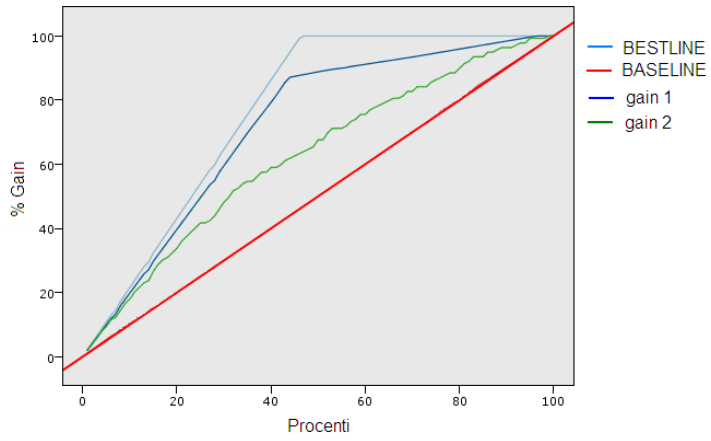
Gain krivulja

Lift krivulja je mera za efektivnost napovedi modela, ki se izračuna kot razmerje med dejanskimi vrednostmi in napovedjo modela. Natančnost modela se določi na podlagi ploščine med lift krivuljo in osnovno premico. Večja je ploščina, boljši je model. Torej višja krivulja kaže na boljši model, še posebej če je zelo na levi strani. Kumulativna *gain krivulja* je vrsta lift krivulje s procentno skalo. Vedno se začne pri 0% in konča pri 100%. Narašča z leve proti desni. V dobrem modelu bo krivulja strmo rastla do 100% in bo nato čimbolj ravna. Model, ki nam ne poda nobene napovedi bo prikazovala naraščajoča diagonala z leve proti desni.

Gain krivulja je opredeljena kot delež vseh zadetkov, ki so se pojavili v vsakem kvartilu, torej

$$\frac{\text{število vseh zadetkov v kvartilu}}{\text{skupno število zadetkov}} \cdot 100\%$$

V graf običajno vrišemo tudi *osnovno krivuljo* (baseline) in *najboljšo krivuljo* (bestline). Prva kaže naključno razdelitev zadetkov, kjer je stopnja zaupanja nerelevantna, druga pa kaže popolno napoved (zanesljivost), kjer so zadetki 100%.



Slika 1.2: Gain krivulja

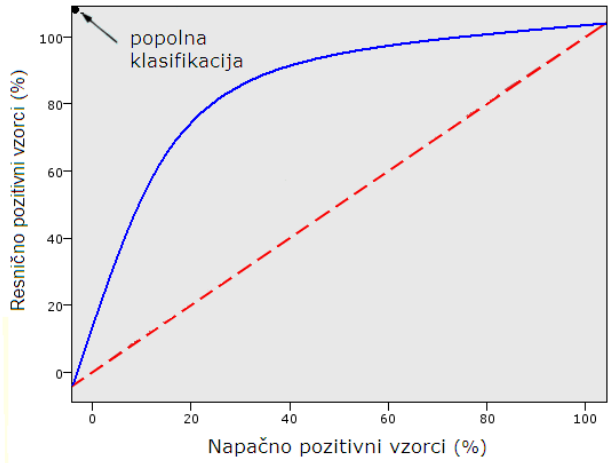
ROC-krivulja

ROC-analiza (*ang. Receiver Operating Characteristic*) je alternativna metoda evalvacije klasifikatorja, ki jo predstavimo z grafom G, ki predstavlja odvisnost med odstotkom napačno pozitivnih vzorcev (os x) proti odstotkom resnično pozitivnih vzorcev (os y) za dan problem. Graf G imenujemo *ROC-krivulja*. ROC-krivulja je torej grafični prikaz pravilno klasificiranih pozitivnih vzorcev proti napačno klasificiranim pozitivnim vzorcem za družino klasifikatorjev. Z drugimi besedami, ROC-graf prikazuje relativen kompromis med izboljšavo (odstotek resnično pozitivnih vzorcev) in stroški (odstotek napačno pozitivnih vzorcev). Za oceno kvalitete nas zanima področje pod krivuljo.

Nekatere točke v ROC-prostoru imajo poseben pomen. Točka $(0, 0)$ predstavlja klasifikator, ki nikoli ni podal pozitivne klasifikacije. Tak klasifikator ne naredi nobene napake v klasifikaciji pozitivnih vzorcev, nikoli pa niti ne doseže nobene resnične pozitivne klasifikacije. Medtem pa točka $(0, 1)$ predstavlja klasifikator s popolno klasifikacijo. Diagonala $y = x$ predstavlja v ROC-prostoru popolnoma naključno klasifikacijo. Vsak klasifikator, ki se nahaja pod diagonalo $y = x$ deluje slabše kot naključje. V splošnem lahko rečemo, da je ena točka v ROC-prostoru boljša od druge, če se nahaja severozahodno od druge, saj je odstotek resnično pozitivnih vzorcev višji, odstotek napačno pozitivnih vzorcev pa nižji.

ROC-krivulja je dvodimenzionalen opis učinkovitosti klasifikatorja, za primerjavo različnih

klasifikatorjev pa je preprosteje, da je podana *ROC-učinkovitost* v eni sami skalarni vrednosti, ki opisuje pričakovano učinkovitost klasifikatorja. Pogosto uporabljena metoda je izračun površine področja pod ROC-krivuljo (AUC). AUC je področje enotskega kvadrata zato je vrednost vedno med 0 in 1, pri čemer noben realni klasifikator ne bi smel imeti $AUC < 0.5$. AUC klasifikatorja je ekvivalentna verjetnosti, da bo klasifikator razvrstil naključno izbran pozitiven vzorec višje kot naključno izbran negativni vzorec.



Slika 1.3: ROC krivulja

Poglavje 2

Tehnike podatkovnega rudarjenja

Skozi leta razvoja podatkovnega rudarjenja se je razvilo ogromno tehnik za pridobivanje informacij iz velikih množic podatkov. Kasnejša združitev teh tehnik pod skupno ime je privedla do določenih zmešnjav razumevanja podatkovnega rudarjenja.

2.1 Tipi tehnik podatkovnega rudarjenja

Tehnike podatkovnega rudarjenja delimo v dve ključni kategoriji:

- Napovedno podatkovno rudarjenje
- Opisno podatkovno rudarjenje

2.1.1 Napovedno podatkovno rudarjenje

Napovedno podatkovno rudarjenje uporablja vrste tehnik, ki v podatkih iščejo povezave med točno določenimi spremenljivkami (imenujemo jih *ciljne spremenljivke*) in ostalimi neodvisnimi ali pojasnjevalnimi spremenljivkami.

Klasifikacija

Klasifikacija je postopek s katerim analiziramo opazovane enote in jih razporedimo v vnaprej oblikovane razrede, v odvisnosti od določene ciljne spremenljivke. Kadar je nov podatek klasificiran, tehnika določi razred in verjetnost, da podatek pripada temu razredu. Tehniko klasifikacije vključujejo odločitvena drevesa, nevronske mreže in radialne funkcije (RBF).

Napovedovanje

Napovedovanje je postopek pri katerem se opazovane enote klasificirajo na podlagi vrednosti spremenljivk v prihodnosti oziroma na podlagi pričakovanega dogajanja v prihodnosti. Najbolj poznane tehnike napovedovanja vključujejo linearno in polinomsko regresijo. Podatkovno rudarjenje nadgradi te tehnike v tehnike, kot so nevronske mreže in RBF napovedovanje.

2.1.2 Opisno podatkovno rudarjenje

Opisno podatkovno rudarjenje uporablja vrste tehnik, ki znotraj podatkov odkrijejo vzorce (korelacije, trende, skupine, anomalije), ki povzemajo (skrite) povezave in razmerja v podatkih.

Grupiranje

Grupiranje je vrsta tehnik, ki združuje podatke na podlagi podobnosti. Obstaja več tehnik grupiranja, vsaka od njih pa ima drugačen pristop za odkrivanje skupin znotraj podatkov.

Analiza povezanosti

Analiza povezanosti opisuje družino tehnik, ki odkrivajo zveze med podatki. Najbolj poznan tip analize povezanosti je analiza nakupovalne košarice. V tem primeru so podatki dobrine, ki jih stranke kupujejo. Dobrine iste transakcije so v eni košarici. Analiza nakupovalne košarice odkriva kombinacije dobrin, ki so jih kupile različne stranke. Z združevanjem (povezovanjem) napravi sliko dobrin, ki se običajno prodajajo skupaj. Če pomislimo na nakupovalno košarico kot na skupino podatkov, lahko uporabimo tehniko analize povezanosti.

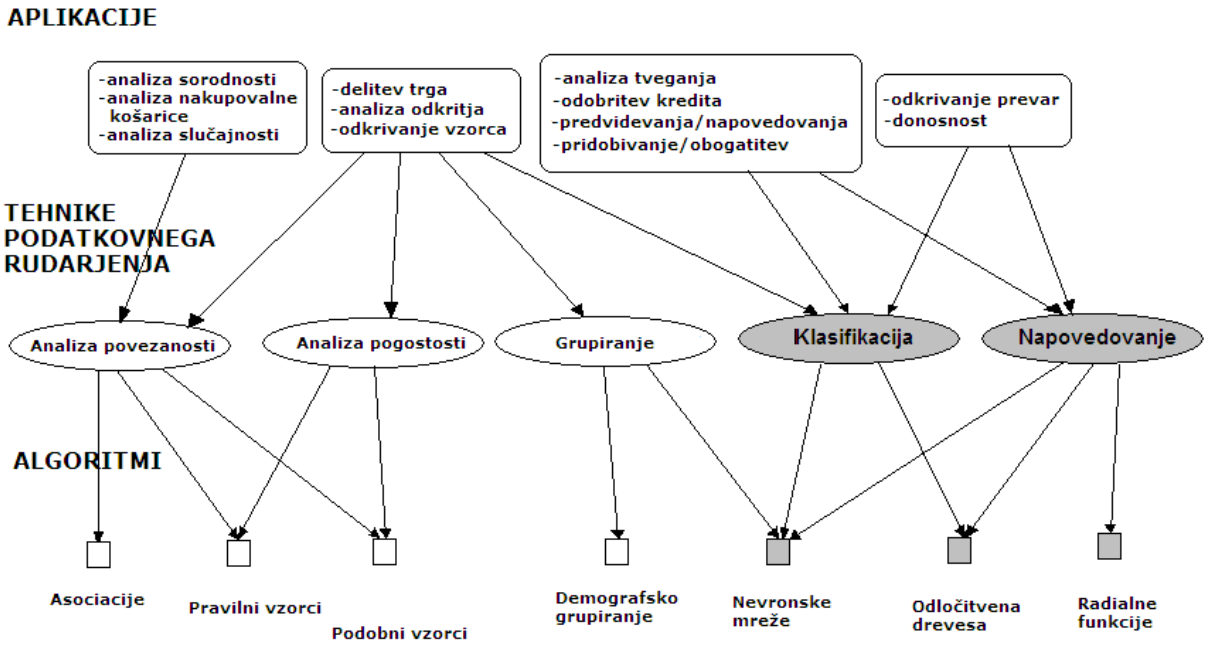
Analiza pogostosti

Analiza pogostosti združuje tehnike podatkovnega rudarjenja, ki uporabljajo časovno urejene podatke ali urejene množice podatkov. Ta tehnika podatkovnega rudarjenja poskuša odkriti podobna zaporedja ali podzaporedja v urejenih podatkih.

2.2 Raznolikost uporabe podatkovnega rudarjenja

Podatkovno rudarjenje lahko uporabljamo v različne namene. V praksi se uporabljajo preproste aplikacije, ki za reševanje posebnih poslovnih problemov vsebujejo tehnike rudarjenja.

Slika 2.1 ilustrira različne uporabe podatkovnega rudarjenja.



Slika 2.1: Od aplikacij do algoritma

Poglavje 3

Algoritmi podatkovnega rudarjenja

Na kratko omenimo nekaj algoritmov podatkovnega rudarjenja. V praktičnem delu diplomskega dela bomo rudarili z algoritmom nevronskih mrež, zato je ta algoritem podrobneje predstavljen v naslednjem poglavju.

3.1 Odločitvena drevesa

3.1.1 Splošno o odločitvenih drevesih

Odločitvena drevesa spadajo v napovedno podatkovno rudarjenje, torej imamo znan ciljni atribut. So simbolična metoda strojnega učenja pri kateri so učni vzorci predstavljeni kot par *(lastnosti,odločitev). Lastnosti* so opisane kot vektor več atributov, ki najbolje predstavijo posamezen vzorec. Izbira atributov je odvisna od učne množice in zmožnosti opravljanja meritev. *Odločitev* je lastnost, ki je znana v vzorcih učne množice, ne pa tudi pri vzorcih o katerih bomo kasneje s pomočjo odločitvenega drevesa sprejemali odločitve. Običajno je odločitvena lastnost lastnost, ki se ne da izmeriti (npr. dogodek, ki se bo zgodil v prihodnosti) ali pa je njena meritev povezana z velikimi stroški in visoko časovno zahtevnostjo. Odločitev mora biti predstavljena kot končni nabor možnih diskretnih vrednosti (pri nevronskih mrežah ni tega pogoja), sicer se pred samo izgradnjo drevesa izvede diskretizacija.

Odločitvena drevesa tako uporabljamo za:

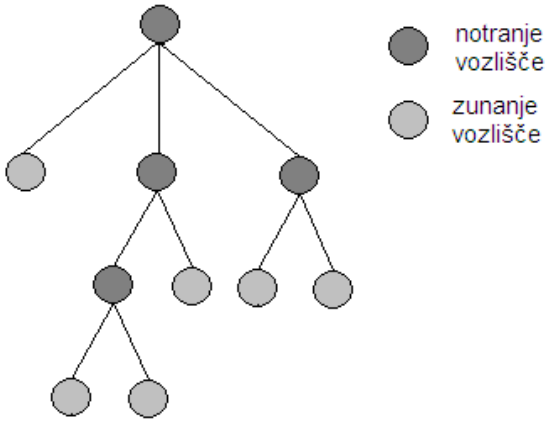
- napoved dogodkov v prihodnosti;
- iskanje alternativnih možnosti za dosego cilja, ki bodo skrajšale čas, znižale stroške ali omogočile doseganje želenih rezultatov.

Odločitvena drevesa iščejo pravila in podobnosti med že rešenimi vzorci, da lahko na osnovi že videnega odločajo o novih, nerešeni vzorcih. Cilj uporabe odločitvenih dreves tako ni le zbiranje znanja rešenih vzorcev, ampak poiskati splošna pravila, ki bi veljala za čimveč vzorcev. Pri tem se poskuša držati pravila Ockhamove britve: *Izmed vseh enakovrednih rešitev (hipotez), je najboljša najenostavnejša.* Pri gradnji dreves stremimo k čim kompaktnejšim drevesom, saj ta običajno zajemajo splošnejše znanje v primerjavi z obširnejšimi drevesi. V želji po minimaliziranju je potrebno paziti, da ne pretiravamo in dobimo preveč posplošenih dreves. Preveč posplošeno drevo lahko vpliva na natančnost pri uporabi svežih podatkov.

3.1.2 Gradnja odločitvenih dreves

Odločitveno drevo se zgradi s pomočjo učne množice, ki je sestavljena iz učnih vzorcev. Lastnosti vzorcev so opisane z množico atributov - lastnosti in izidom (razredom, odločitvijo) v katerega spada. Razredi se med seboj izključujejo, torej en učni vzorec lahko pripada le enemu razredu, vzorci, ki so opisani z enakim vektorjem atributov pa ne morejo imeti različnih odločitev. To nam zagotovi konsistentne učne vzorce. Atributi učne množice so lahko diskretni ali zvezni, pri čemer je potrebno slednje preslikati v diskretno obliko, kar naredimo s pomočjo diskretizacije.

Drevo sestavljajo vozlišča in povezave. Ločimo dve vrsti vozlišč - zunanja in notranja. Vsako notranje vozlišče vsebuje pogoj, ki testira vrednosti atributov in tako razdeli učno množico na manjše podmnožice. Zunanja vozlišča so listi ali terminali in so označeni z razredi. Vozlišča so povezana s povezavami. Povezave so označene z različnimi izidi testov, ki so



Slika 3.1: Odločitveno drevo

bili izvedeni v izvornem vozlišču. Število povezav, ki izhajajo iz vozlišča je tako odvisno od števila možnih izidov testa.

Generiranje odločitvenega drevesa iz učne množice imenujemo *indukcija* odločitvenega drevesa. Indukcija se začne s praznim drevesom in celotno množico učnih vzorcev. Na vsakem koraku s pomočjo metrike čistosti izberemo atribut, ki na poti do trenutnega vozlišča še ni bil uporabljen ter na podlagi možnih vrednosti atributa razdelimo učno množico. Indukcija se izvaja dokler ni zadoščeno pogojem za končanje gradnje odločitvenega drevesa. Pogoji pa so naslednji:

a) v določenem vozlišču učni vzorci pripadajo istemu razredu (imajo isto odločitev);

b) odstotek vzorcev, ki pripadajo večinskemu razredu v določenem vozlišču je večji ali enak toleranci predhodnega klestenja;

c) zmanjkalo je atributov - na poti do vozlišča smo porabili vse atribute, učni vzorci tega vozlišča pa ne pripadajo istemu razredu.

Pri nezadostnem številu učnih vzorcev ali pri prisotnosti šuma običajno pride do čezmernega prilagajanja učnim vzorcem. Rezultat čezmernega prilagajanja so velika odločitvena drevesa, ki vsebujejo precej nepomembnih vej, saj se odločitveno drevo poskuša prilagoditi vsem učnim vzorcem, tudi tistim z napako. Problema nepomembnih vej se lotimo s klestenjem.

3.1.3 Metrike čistosti

Metrike čistosti za izbiro atributov so najpomembnejši faktor pri gradnji odločitvenega drevesa. Njihova naloga je pregledati vse atribute na poti do trenutnega vozlišča, ki še niso bili uporabljeni. Med njimi morajo izbrati tistega, ki najbolj enolično razdeli učno množico vzorcev. Idealni atribut bi učno množico razdelil tako, da bi učni vzorci z enakimi odločitvami prišli pod enak interval ali diskretno vrednost.

Vpeljimo osnovne oznake, ki se uporabljajo v metrikah čistosti. Naj bo S množica učnih vzorcev, opisanih z A atributi in C vzorci. V je število vrednosti, ki jih zavzema dani atribut. Naj bo $n_{..}$ število vzorcev v učni množici; $n_{i.}$ število učnih vzorcev, ki pripadajo razredu C_i; $n_{.j}$ pa število učnih vzorcev, ki imajo j-to vrednost danega atributa. n_{ij} naj označuje število učnih vzorcev, ki pripadajo razredu C_i in imajo j-to vrednost danega atributa. Na učni množici lahko opišemo naslednje verjetnosti $p_{ij} = \frac{n_{ij}}{n_{..}}$, $p_{i.} = \frac{n_{i.}}{n_{..}}$, $p_{.j} = \frac{n_{.j}}{n_{..}}$ in $p_{i|j} = \frac{n_{ij}}{n_{.j}}$.

Navedimo nekaj najpogostejših metrik čistosti: (Vir: [9])

Entropija, informacijski prirastek

Informacijski prirastek je ena najbolj znanih in najpogosteje uporabljenih metrik čistosti. Izvira iz entropije informacijske teorije, ki meri nezanesljivost sporočila kot vir informacij. Več informacij vsebuje sporočilo, nižja je vrednost entropije.

Entropija E atributa A poljubnega učnega vzorca w z možnimi diskretnimi vrednostmi izhodnega atributa $a_1, a_2, ..., a_m$ in verjetnostjo porazdelitve $p(A(w)0a_i)$ je definirana kot

$$E_A = -\sum_j p_{.j}\ log_2 p_{.j}$$

Naj bo E_C entropija porazdelitve razredov, E_A entropija vrednosti danega atributa in E_{AC} entropija združene porazdelitve odločitvenih razredov in vrednosti atributov:

$$E_C = -\sum_j p_{i.}\ log_2 p_{i.}$$

$$E_A = -\sum_j p_{.j}\ log_2 p_{.j}$$

$$E_{CA} = -\sum_i \sum_j p_{ij}\ log_2 p_{ij}$$

Pričakovana entropija porazdelitve razredov glede na atribut A je definirana kot

$$E_{C|A} = E_{CA} - E_A.$$

$E_{C|A}$ prikazuje zmanjšanje entropije, ki ga lahko pričakujemo v primeru, da se atribut A izbere v danem vozlišču za delitveni kriterij.

Informacijski prirastek je definiran kot

$$I_{gain}(A) = E_C - E_{C|A}.$$

V vsakem notranjem vozlišču se izbere atribut z najvišjo vrednostjo I_{gain}.

Gini indeks

Za ocenitev informacijske vrednosti atributov v procesu indukcije odločitvenih dreves in za gradnjo dreves v statistiki se uporablja *gini indeks.*. Gini indeks atributa A je definiran kot:

$$Gini(A) = -\sum_j p_{\cdot j} \sum_i p_{i|j}^2 - \sum_i p_{i\cdot}^2$$

Atribut z najvišjo vrednostjo gini indeksa se izbere za delitveni kriterij v danem vozlišču. Gini indeks ima tendenco h gradnji uravnoteženih dreves.

Hi-kvadrat

Hi-kvadrat testi se uporabljajo za testiranje signifikance v statistiki. Test primerja pričakovano frekvenco e_{ij} z dejansko frekvenco n_{ij} tistih učnih vzorcev, ki pripadajo odločitvenemu razredu C_i in imajo j-to vrednost danega atributa. Definiran je kot

$$\chi^2(A) = \sum_i \sum_j \frac{(e_{ij} - n_{ij})^2}{e_{ij}}, \text{ kjer je } e_{ij} = \frac{n_{\cdot j} n_{i\cdot}}{n_{\cdot\cdot}}.$$

Višja je vrednost χ^2, bolj čista je delitev. Za delitveni kriterij v danem vozlišču se torej izbere atribut z najvišjo vrednostjo χ^2.

3.1.4 Klestenje

Klestenje odločitvenih dreves je postopek s katerim se ublaži oziroma odpravi posledice čezmernega prilagajanja učnim vzorcem. Do čezmernega prilagajanja pride ob prisotnosti šuma, ko se algoritem za gradnjo poskuša prilagoditi temu šumu in se nauči napačne podatke in pa pri pomanjkanju učnih vzorcev, ko algoritem zaradi premalo podatkov ne more izvleči splošnega znanja. Osnovna ideja klestenja je onemogočiti tvorbo oziroma omogočiti odstranitev nepomembnih vej in s tem omejiti velikost odločitvenega drevesa.

V splošnem ločimo med *predhodnim klestenjem* in *naknadnim klestenjem* dreves.

Predhodno klestenje

Predhodno klestenje se izvaja pred samim postopkom gradnje odločitvenega drevesa. Gradnjo nepomembnih vej drevesa prepreči predčasna ustavitev gradnje. Zato predhodnemu klestenju rečemo tudi *kriterij za končanje gradnje drevesa* ali pa *toleranca*, saj z njegovo

uporabo povečamo ohlapnost kriterija za končanje gradnje drevesa in postavitev lista. Prednost je možnost natančne določitve stopnje klestenja. Običajno se določi prag v odstotkih, ki pove, koliko učnih vzorcev z izidom različnim od večine, se zanemari. S tem se neposredno vpliva na velikost zgrajenega drevesa, saj bi se brez predhodnega klestenja namesto lista ustvarilo vozlišče, ki pa bi učno množico delilo dalje.

Naknadno klestenje

Pri naknadnem klestenju imamo že zgrajeno odločitveno drevo, ki mu celotne veje nadomeščamo z vozlišči - listi. V načinu odločanja katere veje bomo oklestili ločimo dva pristopa. Prvi pristop *naknadno klestenje z zmanjševanjem napake* vključuje množico za klestenje s pomočjo katere ocenimo napake v vozliščih. Drugi pristop pa je *naknadno klestenje s statično oceno napake*, kjer napako v vozliščih približno ocenimo s statističnimi metodami.

3.2 Časovne vrste

Časovne vrste spadajo v opisno podatkovno rudarjenje v področje analize pogostosti. *Časovna vrsta* je niz podatkov, ki jih zabeležimo oz. izmerimo v zaporednih časovnih trenutkih ali v zaporednih časovnih intervalih. V prvem primeru govorimo o časovni vrsti v diskretnem času oz. trenutni časovni vrsti, ko opazovanja beležimo zvezno preko nekega časovnega intervala, pa govorimo o časovni vrsti v zveznem času.

S preučevanjem časovnih vrst se ukvarja posebna veja statistike, *analiza časovnih vrst.* Pri analizi časovnih vrst se preučujejo tehnike za risanje motenj vrste. Pri tem je potrebno določiti hipotetičen verjetnostni model, ki predstavlja podatke. Ko je enkrat razvit zadovoljiv model ga lahko uporabimo v različne namene. Model lahko uporabimo za opis podatkov in tako neko časovno vrsto predstavimo kot vsoto potrebnih trendov ter naključnih in sezonskih členov. Uporaba modelov časovnih vrst ponavadi vključuje ločitev šuma, testiranje hipotez, napovedovanje ene vrste z opazovanjem neke druge vrste in nadzorovanje prihodnjih vrednosti vrste s pomočjo določenih parametrov.

3.2.1 Verjetnostni model časovne vrste

Pomemben del analize časovne vrste je izbira primernega verjetnostnega modela ali družine modelov za dane podatke. Pri izbiri primernega modela, navadno sledimo naslednjim korakom:

1. Grafično predstavimo vrsto X_t in poiščemo glavne značilnosti grafa. Še posebej smo pozorni na trend, sezonsko komponento, očitne spremembe v obnašanju in na kakšno (po vrednosti) oddaljeno opazovanje. Če se pojavijo kakšne prekinitve v vrsti, kot npr. nenadna sprememba vrednosti, je priporočljivo vrsto razdeliti v homogene odseke in vsakega posebej analizirati. Če se pojavijo kakšna opazovanja, ki so oddaljena od ostalih, poiščemo vzrok za izstopajoč člen (mogoče je bilo opazovanje nepravilno izmerjeno).

2. S pregledom grafa vidimo, če lahko podatke predstavimo z modelom klasične razčlembe, ki je oblike
$$X_t = m_t + s_t + Y_t,$$
kjer je m_t počasi spreminjajoča se funkcija imenovana *trend*, s_t funkcija z znano periodo d imenovana *sezona* in Y_t *naključna spremenljivka ostanka*, ki je stacionarna. Če valovanje sezone in ostanek naraščata, moramo predhodno preoblikovati podatke, da so lažje združljivi z modelom.

3. Odstranimo trende in sezonske komponente, da dobimo stacionaren preostanek. Včasih je potrebno opraviti predhodne transformacije podatkov. Obstaja kar nekaj postopkov, kako lahko odstranimo trend in sezonske komponente. Nekatere vsebujejo ocenjevanje komponent in odstranitev le-teh iz podatkov, spet druge so odvisne od raznolikosti podatkov. Cilj je pridelati stacionarno vrsto, njene vrednosti pa označimo kot ostanke, ne glede na to katero metodo smo uporabili.

4. Izberemo tak model, da ustreza ostankom. Pri tem uporabljamo različne primere statistike, vključno z avtokorelacijsko funkcijo.

5. Napovedovanje dosežemo z napovedovanjem ostankov in nato z invertiranjem transformacij, ki smo jih uporabili pri odstranjevanju trenda in sezone, da dosežemo napovedovanje prvotne časovne vrste.

3.3 Grupiranje

Grupiranje je metoda, ki spada v opisno podatkovno rudarjenje. Je proces deljenja na skupine oziroma segmente s podobnimi lastnostmi. Za delitev in določitev skupin se uporabljajo statistične in matematične metode. Algoritmi temeljijo na matematični razdalji ali normi. V osnovi se določi norma, algoritmi pa nato znotraj skupine minimalizirajo razdalje med člani in maksimalizirajo razdalje med posameznimi skupinami. Običajno se uporablja Evklidska norma, v kompleksnejših primerih pa tudi ostale znane norme. Pred samim grupiranjem se napravi standardizacija spremenljivk, ki poenoti skale spremenljivk.

Tako se ne more zgoditi, da bi nekatere spremenljivke imele večji vpliv na merjenje razdalje kot druge.

Grupiranje ne poda ene same rešitve, hkrati pa je postopek iterativen. Končni rezultat določimo s pomočjo poskušanja in na osnovi generiranih pravil. Grupiranje ponavadi niti ne poda števila skupin, ampak jih je potrebno vnaprej definirati. Prav tako ne kreira skupin iste velikosti, dobra stvar pa je, da ni odvisno od tipa spremenljivk. Tipična razlika od metod klasifikacije je v tem, da ni ciljnega atributa. Torej so vsi atributi enakovredni. En atribut oziroma ena spremenljivka predstavlja eno dimenzijo, tako dobimo pri veliko atributih veliko kompleksnost.

Poznamo različne tipe grupiranja:

- *Hierarhično* - sprva je vsak podatek v svoji skupini, nato se podatki podmnožic grupirajo v isto skupino. Metoda dobro deluje na majhni množici podatkov.
- *K-means* - na začetku sami določimo število skupin. Algoritem izbere naključne točke prostora in jih določi kot sredine nekih skupin. Na osnovi razdalj določi člane skupine in novo sredino. Postopek ponavlja, dokler se središča skupin spreminjajo.
- *Two step* - algoritem sam izračuna optimalno število skupin in na osnovi razdalj določi njene člane. Sprejema nominalne, ordinalne in številne podatke, spremenljivke pa morajo biti nekorelirane in normalno porazdeljene. Zelo učinkovita metoda pri veliko podatkih.

Grupiranje se v praksi ponavadi uporablja v povezavi s klasifikacijo. Atributi se združijo v skupine, te pa se uvozijo v analizo klasifikacije. Pomembno pravilo je, da se pri združevanju teh dveh metod v analizi grupiranja ne vključi ciljna spremenljivka klasifikacije, saj bi to lahko privedlo do napačno naučenega znanja.

3.4 Asociacije

Pod pojmom asociacije razumemo ugotavljanje dejstev na podlagi katerih določene stvari spadajo skupaj. Metoda izhaja iz področja opisnega podatkovnega rudarjenja, povezovalnih (asociativnih) pravil. Slednja se uporabljajo pri analizi velikih podatkovnih baz za iskanje vzorcev, relacij. S povezovalnimi pravili lahko odkrijemo potencialno zanimive zakonitosti med podatki. Najbolj znan primer uporabe je analiza nakupovalne košarice.

3.4.1 Povezovalna pravila

Povezovalna pravila so pravila oblike $X \rightarrow Y$, kjer sta X in Y neprazni, tuji množici postavk. Namen učenja povezovalnih pravil je na bazi podatkov poiskati zanimive relacije med atributi, ki lahko doprinesejo nova spoznanja o samih podatkih.

Postopek iskanja povezovalnih pravil je sestavljen iz dveh faz:

1. iskanje pogostih n-teric
2. generiranje povezovalnih pravil na podlagi pogostih n-teric

V prvi fazi iščemo množice predmetov z želenimi lastnostmi. Rezultat te faze je množica n-teric (množica množic predmetov), ki imajo podporo večjo ali enako dani spodnji meji. V drugi fazi se generirajo pravila glede na n-terice prve faze.

Naloga učnega algoritma je poiskati vsa povezovalna pravila, ki so dovolj pogosta, kar izračunamo s *podporo* in zanesljiva, kar izračunamo z *zaupanjem*. To sta običajni meri za zanimivost povezovalnega pravila. Podpora je delež transakcij v katerih se pojavijo vsi predmeti, nastopajoči v X in Y, zaupanje pa delež transakcij med transakcijami z X, ki vsebujejo tudi Y. Meri sta objektivni, saj temeljita na statističnih zakonitostih v podatkih.

$$podpora(X \rightarrow Y) = \frac{|\{T \,|X \cup Y \subseteq T, T \in DB\}|}{|DB|}$$

$$zaupanje(X \rightarrow Y) = \frac{|\{T \,|X \cup Y \subseteq T, T \in DB\}|}{|\{T \,|X \subseteq T, T \in DB\}|}$$

Obstaja podobnost med povezovalnimi pravili in klasifikacijskimi pravili. Slednja se razlikujejo v tem, da v sklepnem delu nastopa samo ena postavka, ki je razred, saj so klasifikacijska pravila namenjena klasifikaciji. Povezovalna pravila so razširitev klasifikacijskih.

Poglavje 4

Nevronske mreže

Veda o umetnih nevronskih mrežah se uveljavlja šele v zadnjih desetletjih čeprav začetki segajo že v zgodnja štirideseta leta. Že leta 1943 je bil predstavljen matematični model živčne celice oziroma nevrona. Ta model umetnega nevrona je zasnovan na predpostavki, da so vhodni signali uteženi, sestavljeni signal pa je odvisen od produkta vhodnih vrednosti in uteži. Sestavljeni signal potuje do posebne preklopne oziroma aktivacijske funkcije. Izhod aktivacijske funkcije je tudi izhod umetnega nevrona. Takšen model umetnega nevrona se je v skoraj nespremenjeni obliki ohranil do danes.

4.1 Definicija nevronske mreže

Nevronska mreža je sistem povezanih vozlišč oziroma enot imenovanih nevroni, katerih model funkcionalnosti temelji na modelu delovanja možganskih celic. Procesna sposobnost oziroma znanje nevronske mreže je vsebovano v povezavah med nevroni, ki so jim dodeljene številske uteži. Prave vrednosti uteži izpeljemo s postopkom učenja nevronske mreže na množici učnih primerov. Učenje nevronske mreže se izvaja z algoritmom učenja, ki spreminja uteži na povezavah med nevroni in s tem "oblikuje" znanje, ki bo shranjeno.

4.2 Matematični model nevrona

Osnovni gradnik nevronskih mrež je *nevron*. Operacija, ki jo nevron opravi ima dve komponenti - vhodno in izhodno. Prva komponenta nevrona k, *vhodna funkcija* s_k, je linearna

komponenta, ki predstavlja seštevek produktov vhodnih signalov in uteži:

$$s_k = \sum_{i=1}^{n} w_{ki} \cdot x_i \tag{4.1}$$

Druga komponenta, *aktivacijska funkcija* φ, ima za vhod izhod iz prve komponente. Izhod druge komponente je tudi izhod nevrona:

$$y_k = \varphi(s_k - \theta_k) \tag{4.2}$$

- $x_1, ...x_n$ **vhodni signali** - atributi, ki vstopajo v mrežo in so lahko izključno numerični,
- $w_{k1}, ...w_{kn}$ **sinaptične uteži k-tega nevrona** - izražajo moč podatka, ki prihaja preko povezave iz enega na drugi nivo,
- s_k **aktivacija**,
- θ_k **prag** - nivo signala pri katerem se sproži nevron,
- φ **aktivacijska funkcija** - kombinira vhode v vmesno vrednost,
- y_k **izhodni signal nevrona** - preslika izhod aktivacije v izhod nevrona.

Model nevrona k vsebuje *prag* θ, ki predstavlja nivo signala pri katerem se sproži nevron. Prag θ je lahko tudi zunanji parameter. Vpeljimo oznako

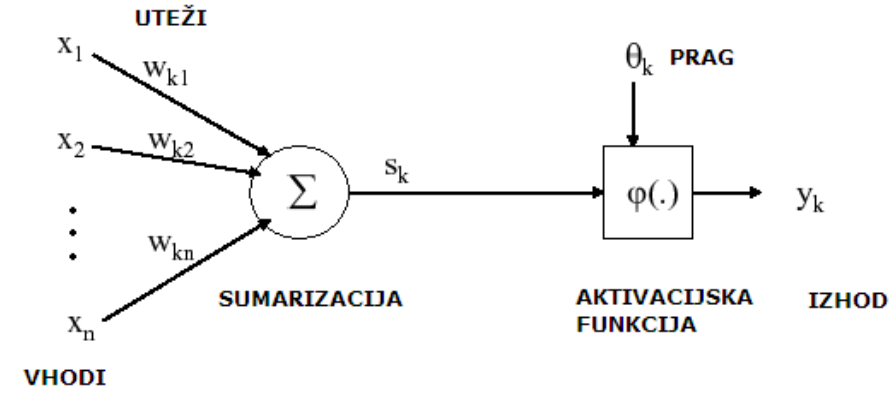
$$a_k = s_k - \theta_k$$

Če dodamo vhod $x_0 = -1$ in postane prag sinaptična utež w_{k0}, lahko zapišemo prenos signala skozi nevron z enačbama

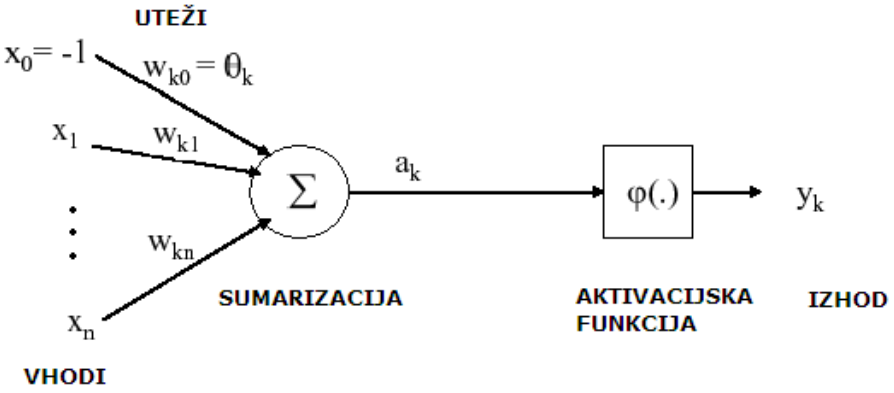
$$a_k = \sum_{i=0}^{n} w_{ki} \cdot x_i \tag{4.3}$$

$$y_k = \varphi(a_k) \tag{4.4}$$

Sliki 4.1 in 4.2 prikazujeta zgradbo umetnega nevrona brez in s pragom kot vhodnim parametrom. Modela sta matematično ekvivalentna.



Slika 4.1: Zgradba umetnega nevrona - prag ni vhodni parameter



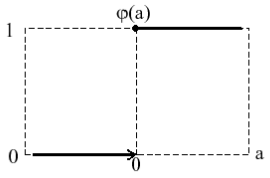
Slika 4.2: Zgradba umetnega nevrona - prag je vhodni parameter

4.2.1 Aktivacijske funkcije

Aktivacijske funkcije so ponavadi omejene monotone funkcije. Te so lahko zvezne ali nezvezne. Navedimo nekaj najpogostejših aktivacijskih funkcij:

1. **Pragovna ali stopničasta funkcija**

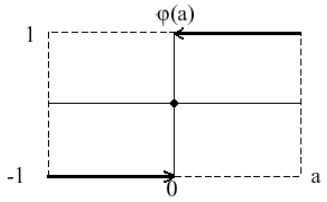
$$\varphi(a) = \begin{cases} 1 & ; \quad a \geq 0 \\ 0 & ; \quad a < 0 \end{cases}$$



Slika 4.3: Pragovna funkcija

2. **Funkcija signum**

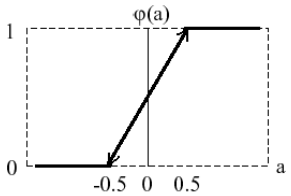
$$\varphi(a) = \begin{cases} 1 & ; \quad a > 0 \\ 0 & ; \quad a = 0 \\ -1 & ; \quad a < 0 \end{cases}$$



Slika 4.4: Funkcija signum

3. **Odsekoma linearna funkcija**

$$\varphi(a) = \begin{cases} 1 & ; \quad a \geq \theta \\ a + \theta & ; \quad -\theta < a < \theta \\ 0 & ; \quad a \leq -\theta \end{cases}$$



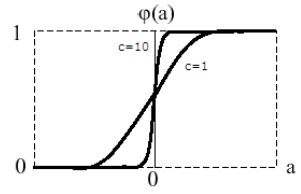
Slika 4.5: Odsekoma linearna funkcija za $\theta = 0.5$

4. **Sigmoidne funkcije** so funkcije v obliki črke 'S'. To so najobičajnejše aktivacijske funkcije nevronskih mrež. Navedimo dve najbolj pogosti:

   (a) **Logistična funkcija** je definirana z enačbo

   $$\varphi(a) = \frac{1}{1 + e^{-ca}},$$

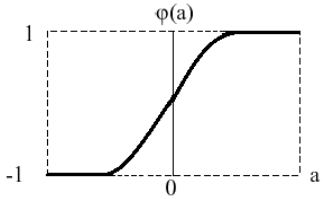
   kjer je c *parameter naklona.* Male vrednosti c krivuljo sploščijo, velike pa jo naredijo bolj strmo. V limiti, ko gre $c \to \infty$, dobimo pragovno funkcijo. Logistična funkcija zvezno zavzame vrednosti med 0 in 1 in je za razliko od pragovne funkcije odvedljiva.



Slika 4.6: Logistična funkcija

   (b) **Funkcija hiperbolični tangens** je definirana z enačbo

   $$\varphi(a) = tanh(a) = \frac{e^{2a} - 1}{e^{2a} + 1}$$



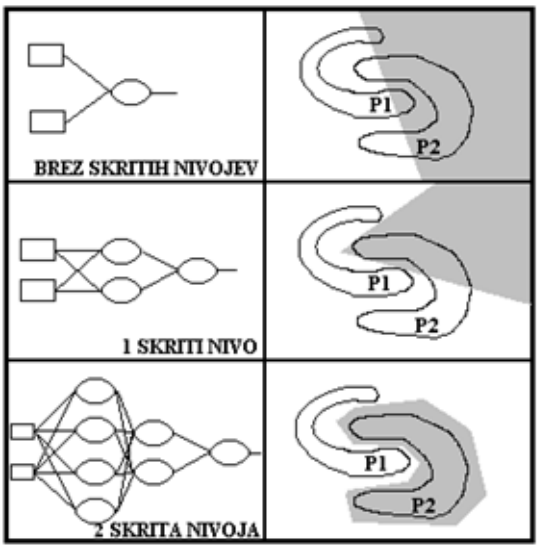
Slika 4.7: Funkcija hiperbolični tangens

Opazimo, da imajo različne aktivacijske funkcije različne dopustne vrednosti izhoda, ki se gibljejo na intervalu $[0, 1]$ ali $[-1, 1]$. Aktivacijsko funkcijo lahko torej izberemo glede na želene vrednosti izhoda nevrona.

4.3 Nevronska mreža

Nevronska mreža je množica paralelno povezanih nevronov, ki istočasno dobijo signal in ga obdelajo z računalniškimi operacijami. Nevronsko mrežo, kakor nevron, sestavljajo vhodi, izhodi in uteži, poleg tega pa mreža lahko vsebuje tudi *skrite nivoje*, kjer so uteženi nevroni z aktivacijsko funkcijo. Umetni nevron obdela vhode in jih posreduje skozi en izhod, ki je z utežmi vezan na druge nevrone. Nastane povezana mrežna struktura sestavljena iz nivojev ali plasti. En nivo je vedno vhoden, ta skrbi za vstopanje informacij v model, en nivo je izhoden in posreduje rezultate delovanja mreže. Ostali nivoji so skriti in omogočajo klasifikacijo objektov.

Zmogljivosti mrež ocenjujemo glede na število nivojev. Nevronske mreže, ki nimajo skritih nivojev lahko ločijo prostor le na dve polravnini (v več dimenzijah na dva hiperprostora). Takšne nevronske mreže imenujemo *perceptroni.* Nevronske mreže z enim skritim nivojem lahko omejijo poljubno odprto ali zaprto konveksno območje. Nevronske mreže z dvema skritima nivojema pa lahko omejijo poljubno območje.



Slika 4.8: Zmogljivost nevronskih mrež v odvisnosti od števila nivojev

(Slika vir: [9])

Če si nevronsko mrežo predstavimo grafično nam nevron loči n-dimenzionalen prostor vhodnih spremenljivk s hiperravnino. Uteži nevrona predstavljajo parametre hiperravnine in s spreminjanjem le-teh spreminjamo lego hiperravnine. Cilj učenja je najti takšne parametre (uteži), da bo hiperravnina s svojo lego ločevala prostor vzorcev tako, da bodo na eni strani le vzorci z enako iskano lastnostjo.

Zmogljivost nevronskih mrež je seveda precej večja od zmogljivosti samega nevrona. Številu in načinu povezave nevronov v mrežo pravimo *topologija nevronske mreže.*

4.4 Topologija nevronskih mrež

4.4.1 Lastnosti nevronskih mrež:

1. Glede na **vrsto topologije** ločimo *večnivojske in enonivojske* nevronske mreže. Prve so organizirane po nivojih (vhodni, skriti, izhodni) in ne dovoljujejo povezav med nevroni na istem nivoju. Druge ne ločujejo nevronov po nivojih in dovoljujejo povezave med vsemi nevroni.

2. Glege na **usmerjenost povezav** ločimo *nevronske mreže s povezavami nazaj in nevronske mreže brez povezav nazaj.* Prve dovoljujejo povezave v obe smeri, pri povezavah drugih pa izhodi enega nevrona pomenijo vhode drugega nevrona, torej se vrednosti vedno prenašajo le z leve proti desni.

3. Glede na **način učenja** ločimo *nadzorovano učenje in nenadzorovano učenje.* Nadzorovano učenje ima nekakšnega učitelja, ki mreži poda pravilen odgovor, medtem ko pri nenadzorovanem učenju ni primerjave med dejanskim in pričakovanim izhodom.

Predstavniki nevronskih mrež:

Feed-forward nevronske mreže - večnivojske, brez povezav nazaj, nadzorovano učenje.

Hopfieldove nevronske mreže - enonivojske, s povezavami nazaj, nadzorovano učenje.

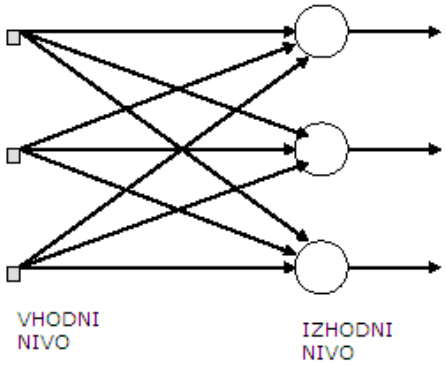
Kohonenove nevronske mreže - večnivojske, brez povezav nazaj, nenadzorovano učenje.

4.4.2 Vrste nevronskih mrež:

Enonivojska nevronska mreža brez povezav nazaj

Najpreprostejša nevronska mreža ima en vhodni nivo in en izhodni nivo. Primer enonivojske mreže brez povezav nazaj je perceptron. Perceptron je linearen, če je aktivacijska funkcija

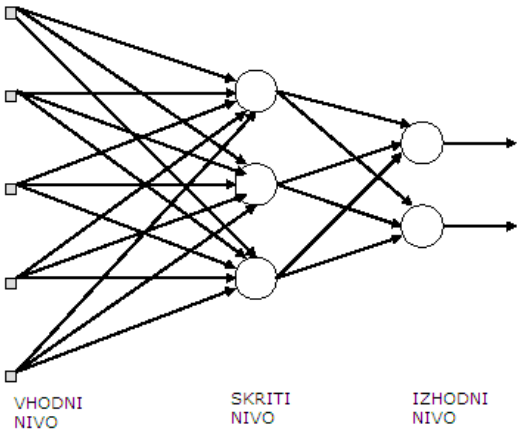
linearna in nelinearen, če je aktivacijska funkcija nelinearna funkcija. Linearen perceptron je v statistiki ekvivalenten linearni regresiji.



Slika 4.9: Enonivojska nevronska mreža brez povezav nazaj - perceptron

Večnivojska nevronska mreža brez povezav nazaj

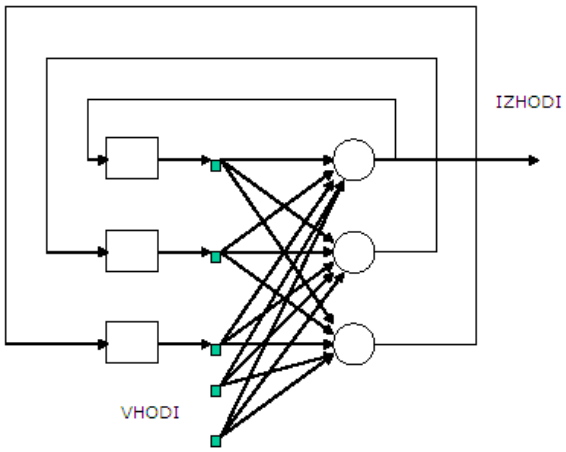
Večnivojska nevronska mreža brez povezav nazaj je najbolj priljubljena oblika nevronske mreže. Mreža vsebuje poljubno število vhodov, vsaj en skriti nivo, v vsaki skriti plasti določeno število nevronov, uteži na vhodih in v skritih plasteh. Mreža je polno povezana, če je vsak nevron v nevronski mreži povezan z vsemi nevroni na naslednjem nivoju.



Slika 4.10: Polno povezana večnivojska nevronska mreža brez povezav nazaj

Nevronska mreža s povezavami nazaj

Nevronske mreže s povezavami nazaj se ločijo od nevronskih mrež brez povezav nazaj po tem, da vsebujejo vsaj eno povratno povezavo. Povratna povezava lahko izvira iz skrite plasti ali pa iz izhodne plasti mreže. Povratne povezave bistveno izboljšajo sposobnost učenja in delovanja nevronske mreže, algoritmi učenja pa so zato kompleksnejši.



Slika 4.11: Nevronska mreža s povezavami nazaj

4.5 Učenje nevronskih mrež

Učenje v smislu nevronskih mrež definiramo kot

Definicija 4.1 *Učenje je proces pri katerem se prosti parametri nevronske mreže prilagodijo skozi kontinuiran proces simulacije od okolja, v katerega je vložena mreža. Tip učenja določa način, po katerem se spreminjajo parametri.*

Nevronska mreža potrebuje za učenje, kot vsak algoritem podatkovnega rudarjenja, učne in testne podatke. Ključni so učni podatki na podlagi katerih mreža razporeja uteži, da se približa čimboljšemu izhodu. Na osnovi vhodnih podatkov se postavijo na povezavah uteži, ki se primerjajo s pragom. Vsaka utež v naslednjem nivoju je odvisna od odločitve nevronov v prejšnjem nivoju. Na izhodu nevrona se rezultat primerja s predvidenim rezultatom iz učnega modela. Če je rešitev znotraj dovoljene tolerance se razporeditev uteži in prag shranijo kot ustrezna rešitev, sicer se učenje ponavlja dokler ni dosežen željen rezultat. Pogosto prihaja do problema generalizacije, ko se nevronska mreža nauči točne učne podatke

in ni sposobna sklepati za nove vhode. Kriterij zaustavitve učenja nevronske mreže je lahko podan kot število ciklov učenja ali kot minimalna napaka pri testiranju.

Pri učenju mreže pride vedno do razlike med dejanskim in želenim izhodom (razen pri generalizaciji). Razliko oziroma pomembnost te napake merimo s *funkcijo napake* (error function). Cilj učenja je, da napaka postane tako majhna, da jo lahko zanemarimo. Povedano z matematičnimi besedami: *učenje mreže je iskanje lokalnega minimuma na ploskvi napake - uteži.* Najbolj razširjena funkcija napake je *vsota kvadratnih napak* (sum of squared errors), saj so posamična velika odstopanja bistveno bolj kaznovana kot enakomerno porazdeljena manjša odstopanja. Vsoto kvadratnih napak nevronske mreže izračunamo po enačbi

$$E_{NM} = \frac{1}{2N} \sum_{k=1}^{N} \sum_{j=1}^{m} (t_j^{(k)} - y_j^{(k)})^2,$$

kjer je $j = 1, ..., m$ indeks nevrona v zadnjem sloju, m število nevronov v zadnji plasti, $k = 1, ..., N$ indeks vzorca, N število vseh vzorcev, $t_j^{(k)}$ želeni izhod j-tega nevrona za k-ti vzorec in $y_j^{(k)}$ dejanski izhod j-tega nevrona za k-ti vzorec.

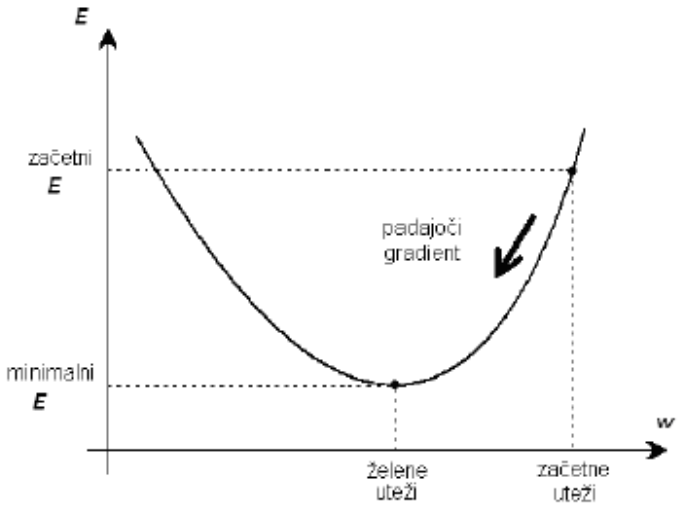
Najpogosteje uporabljene nevronske mreže so feed-forward nevronske mreže, ki so večnivojske, brez povezav nazaj in jih učimo z nadzorovanim učenjem. Njihova značilnost je, da niso dovoljene povezave med nevroni na istem nivoju, niso dovoljene povezave nazaj na prejšnje nivoje in ni dovoljeno preskakovanje enega ali več nivojev pri povezavah nazaj v fazi učenja. V primeru diplomskega dela se bomo ukvarjali s feed-forward mrežo, zato se omejimo na učenje le-te.

Najbolj uveljavljen in sorazmerno preprost algoritem za nadzorovano učenje je algoritem *vzvratno širjenje napake,* ki temelji na metodi padajočih gradientov.

4.5.1 Metoda padajočih gradientov

Gradientni spust ali *metoda padajočih gradientov* je standardni matematični optimizacijski algoritem, ki deluje na podlagi računanja odvodov. Zato mora biti funkcija napake, ki jo želimo minimalizirati z reguliranjem uteži, diferenciabilna.

Ključna lastnost odvoda je, da lahko v vsaki točki določimo strmino in smer krivulje. Če je odvod negativen funkcija pada proti desni, če je odvod pozitiven pada proti levi. Velikost odvoda pove strmino. Gradientni spust je iterativna procedura, ki uporabi te informacije za regulacijo parametrov. Vzame vrednost odvoda jo pomnoži s konstanto η, ki ji pravimo hitrost učenja in odšteje rezultat od dejanske vrednosti parametra. To ponavlja za vse vrednosti parametra dokler ne doseže minimuma. Slika 4.12 predstavlja gradientni spust, pri čemer os x prikazuje parametre, ki jih optimiziramo.



Slika 4.12: Gradientni spust

Naj bo y_j izhodni vektor j-tega nevrona nevronske mreže in t_j vektor pričakovanih rezultatov j-tega nevrona. Definirajmo *napako signala*, ki potuje po nevronski mreži, kot

$$e_j = t_j - y_j$$

Če za funkcijo napake vzamemo vsoto kvadratnih napak je napaka, ki jo naredi nevronska mreža pri klasificiranju j-tega nevrona enaka

$$E_j = \frac{1}{2}e_j^2 = \frac{1}{2}(t_j - y_j)^2$$

Seštevek napak vseh nevronov v zadnji plasti nam poda napako nevronske mreže za en predstavljen vzorec

$$E = \sum_{j=1}^{m} E_j = \frac{1}{2}\sum_{j=1}^{m}(t_j - y_j)^2$$

Definirajmo funkcijo uteži $E = E(w_1, ..., w_n)$. Optimalen utežni vektor najdemo z metodo padajočih gradientov. Padajoči gradient je iskanje minimuma funkcije več spremenljivk $y(x_1, ..., x_n)$ pri čemer se uporabi pravilo

$$\Delta x_i = -\eta \frac{\partial y}{\partial x_i},$$

kjer η predstavlja hitrost učenja. Pravilo pove v katero smer se morajo popravljati neodvisne spremenljivke x, da se zmanjša odvisna spremenljivka y.

Če pretvorimo pravilo v naše oznake dobimo

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}, \tag{4.5}$$

pravilo za izračun spremembe i-te uteži j-tega nevrona.

Izračunajmo parcialni odvod

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial e_j} \cdot \frac{\partial e_j}{\partial y_j} \cdot \frac{\partial y_j}{\partial a_j} \cdot \frac{\partial a_j}{\partial w_{ij}}, \tag{4.6}$$

pri tem je a_j seštevek uteženih vhodov po enačbi 4.3.

Poračunajmo odvode

$$\frac{\partial E}{\partial e_j} = e_j$$

$$\frac{\partial e_j}{\partial y_j} = -1$$

$$\frac{\partial y_j}{\partial a_j} = \varphi'(a_j)$$

$$\frac{\partial a_j}{\partial w_{ij}} = x_i$$

Enačbo 4.6 lahko torej zapišemo kot

$$\frac{\partial E}{\partial w_i} = -e_j \varphi'(a_j) x_i \tag{4.7}$$

Vpeljimo oznako δ_j za *gradient* j-tega nevrona

$$\delta_j = \frac{\partial E}{\partial a_j} = \frac{\partial E}{\partial e_j} \cdot \frac{\partial e_j}{\partial y_j} \cdot \frac{\partial y_j}{\partial a_j} = -e_j \varphi'(a_j)$$

Sedaj lahko zapišemo enačbo za spremembo uteži 4.5 kot

$$\Delta w_i = \eta \delta_j \varphi'(a_j) x_i, \tag{4.8}$$

ki jo imenujemo *delta pravilo.*

4.5.2 Vzvratno širjenje napake

Algoritem vzvratno širjenje napake mrežo uči tako, da približuje napako vrednosti nič, napaka pa je razlika med izhodom in odvisno spremenljivko. Algoritem zahteva veliko ponavljanj zato je časovno zahteven. V osnovi uporablja metodo padajočih gradientov oziroma delta pravilo, ki v fazi backpropagation posodablja uteži. Znotraj algoritma obstajata dve možnosti posodobitve uteži, *paketno spreminjanje uteži* (batch-mode) in *vzorčno učenje* (on-line learning). V paketnem načinu se uteži posodabljajo po izračunu skupne napake mreže (E_{NM}), v vzorčnem načinu pa po izračunu napake vsakega učnega vzorca posebej (E). Najpogosteje se uporablja vzorčni način, ki je preprostejši za inplementacijo in poda efektivnejše rešitve problemov. V nadaljevanju diplomskega dela bomo predstavili algoritem vzvratnega širjenja napake z vzorčnim posodabljanjem uteži.

Cilj je s pomočjo uteži v nevronski mreži zmanjšati napako E. Algoritem ima dva koraka: *inicializacijo* (korak 1), ki vse uteži, vključno s pragovi w_0, izbere z naključno funkcijo z vrednostmi med -0.5 in 0.5; in pa *zanko* (korak 2 in 3), kjer se ponavljajo postopki v notranjosti zanke, dokler se vsi vzorci ne klasificirajo pravilno. Mrežo učimo in ji predstavljamo učne vzorce dokler ni izpolnjen pogoj za konec. Cikel v katerem mreži predstavimo učne vzorce imenujemo *epoha* (epoch).

Algoritem vzvratnega širjenja napake bomo opisali na primeru nevronske mreže z enim skritim nivojem za en učni vzorec. Podobno bi lahko postopek prenesli na nevronske mreže z več skritimi nivoji.

Algoritem vzvratnega širjenja napake

1. V n dimenzionalen vhodni vektor x nevronske mreže vpišemo vrednosti atributov za učni vzorec. Vsak vhod x_i tvori signal na povezavah uteženimi z utežmi w_{ij}.

2. Faza *feed-forward*: računajo se aktivacije a_j za j-ti nevron po enačbi 4.3, za vsak nevron, medtem ko se pomikamo od vhodnega nivoja preko skritih nivojev do izhodnega. Na koncu dobimo po enačbi 4.4 vrednosti komponent za izhodni vektor y, ki predstavlja odločitev.

3. Faza *backpropagation*: primerjava vektorjev y in t. Če sta enaka ali dovolj blizu se vrnemo na začetek zanke, sicer popravimo utež, da se zmanjša razlika med vektorjema y in t, ter se nato vrnemo na začetek zanke. Za popravljanje uteži računamo gradiente, plast za plastjo. Pri tem si pomagamo z učnim pravilom delta.

   Popravljanje uteži v fazi *backpropagation* se izvede po enačbi

$$w_{ij}^{(l+1)} = w_{ij}^{(l)} + \Delta w_{ij}^{(l)}, \tag{4.9}$$

kjer je $\Delta w_{ij}^{(l)}$ popravek, ki ga izračunamo po enačbi

$$\Delta w_{ij}^{(l)} = \eta \delta_j x_i + \alpha \Delta w_{ij}^{(l-1)} \tag{4.10}$$

$w_{ij}^{(l)}$ predstavlja vrednost i-te uteži j-tega nevrona izračunane na l-tem koraku zanke. η in α sta pozitivni konstanti. Prva predstavlja hitrost učenja, druga pa moment ali iteracijo učenja, ki povezuje spremembo med korakoma $(l+1)$ in (l). Na začetku se običajno postavita vrednosti $\eta = 0.5$ in $\alpha = 0.9$, nato pa vrednosti med učenjem spreminjamo, odvisno če želimo povečati ali zmanjšati hitrost konvergence. Izračun gradienta j-tega nevrona δ_j je odvisen od nivoja v katerem se nahajamo. Če je nevron j v izhodnem nivoju se vrednost izračuna po enačbi

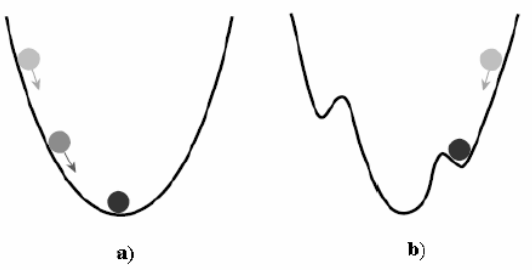
$$\delta_j = (t_j - y_j) \cdot \varphi'(a_j)$$

Če pa se nevron j nahaja v skritem nivoju, računamo na način

$$\delta_j = \varphi'(a_j) \cdot \sum_k \delta_k w'_{jk},$$

kjer parameter k teče obakrat od 1 pa do m, kjer je m število nevronov v izhodnem nivoju nevronske mreže. Uteži w'_{jk} so uteži nevronov v izhodnem nivoju, δ_k pa računamo po enačbi za nevrone v izhodnem nivoju.

Če si napako predstavimo grafično (slika 4.13) je napaka krožec, ki pada proti minimumu. Če je krivulja gladka (slika 4.13 a), algoritem nima težav. Če pa krivulja vsebuje kakšne lokalne minimume (slika 4.13 b), lahko pride do napake delovanja algoritma, saj se napaka lahko ustavi v lokalnem minimumu. Z uporabo momenta α povzročimo zmanjševanje napake proti globalnemu minimumu. Pri izračunu nove uteži na koncu enostavno prištejemo del prilagoditve uteži prejšnjega cikla na istem mestu (enačba 4.10).



Slika 4.13: Grafična predstavitev zmanjševanja napake

Poglavje 5

Splošna metoda podatkovnega rudarjenja

Splošna metoda podatkovnega rudarjenja obsega sedem korakov, to so:

1. Natančno definiranje problema
2. Opredelitev podatkov modela in podatkovnih zahtev
3. Pridobivanje podatkov iz vseh dostopnih skladišč in priprava podatkov
4. Vrednotenje kvalitete podatkov
5. Izbira tehnologije rudarjenja in definiranje poteka rudarjenja
6. Razlaga rezultatov in odkrivanje novih informacij
7. Vpeljava rezultatov in novih znanj v posel

Vidimo lahko, da pretežen del vsega časa pri podatkovnem rudarjenju porabimo za obdelavo podatkov.

5.1 Definiranje poslovnega problema

Prvi korak splošne metode podatkovnega rudarjenja je opredelitev in razlaga poslovnega problema ter ugotovitev kako lahko problem prevedemo v vprašanje ali množico vprašanj, na katera lahko odgovorimo s podatkovnim rudarjenjem.

Zahteve poslovnega problema so:

- Jasen opis problema, ki ga preučujemo;
- Razumevanje podatkov, ki bi naj bili relevantni;
- Vizijo kako uporabiti rezultate podatkovnega rudarjenja v praksi.

Opis problema

Pri opisu problema je zelo dobro vedeti kako deluje podatkovno rudarjenje. To nam pomaga, da zastavimo problem tako, da lahko nanj odgovorimo s pomočjo podatkovnega rudarjenja. Najpomembneje si je zapomniti, da deluje podatkovno rudarjenje na odkrivanju vzorcev in povezav med podatki. Različne aplikacije podatkovnega rudarjenja uporabljajo isti osnovni koncept vendar na različne načine. Problem je torej potrebno zastaviti tako, da bo iskanje vzorcev in povezav med podatki imelo smisel.

Razumevanje podatkov

Hkrati je pri definiranju poslovnega problema potrebno premisliti kdaj so podatki, ki so nam na voljo, zadostni za rešitev problema. Pomembno je že vnaprej prepoznati kdaj dostopni podatki ne vsebujejo informacij, ki bi nam omogočile najti odgovor na postavljeno vprašanje.

Kadar dobimo že definiran problem je poznavanje podatkov v pomoč pri izbiri uporabnih podatkov in pri pretvorbi podatkov, da bodo dejansko uporabni. Ta proces imenujemo *gradnja osnovnega podatkovnega modela*. Uporaba dobrega osnovnega podatkovnega modela je v veliko pomoč pri izvedbi podatkovnega rudarjenja za točno določen poslovni problem.

Uporaba rezultatov

Pri definiranju poslovnega problema je dobro, da vnaprej razmislimo kako bomo uporabili odkrita znanja podatkovnega rudarjenja. Pogosto je poznavanje uporabe rezultatov podatkovnega rudarjenja uporabno in v pomoč pri definiranju problema in za določitev uporabnih podatkov. Torej potrebno je razmisliti kakšne bodo pridobljene informacije, kar je lahko ključ do pretvorbe poslovnega problema v problem podatkovnega rudarjenja.

5.2 Opredelitev podatkov modela za uporabo

Drugi korak splošne metode podatkovnega rudarjenja je opredelitev podatkov, ki se bodo uporabljali v procesu učenja in testiranja. Podatki so ponavadi shranjeni v velikih obsegih

in so skladiščeni za podporo veliko aplikacij. Arhitektura skladišča podatkov ni vedno enaka. V večini primerov je vsaka aplikacija podprta s svojim podatkovnim skladiščem, ki je nadgrajen v rednih intervalih ali ko je nek določen podatek spremenjen. V tej strukturi ima vsako skladišče svoj podatek in shranjuje znanje kako je podatek bil pridobljen, podatkovni zapis, v katere oblike ga lahko pretvorimo,... Podatke in skladišče podatkov skupaj imenujemo *model podatkov.*

Tipični model podatkov opredeljujejo:

- Viri podatkov;
- Tipi podatkov;
- Obseg podatkov;
- Opis podatkov;
- Uporaba podatkov.

Viri podatkov navedejo fizično lokacijo skladišča oz. pridobitve podatkov. *Tip podatkov* opredeli sestavo podatkov (recimo časovni format zapisa podatkov). *Obseg podatkov* našteje tabele ali arhive podatkov in področje, ki ga obsegajo. *Opis podatkov* vsebuje imena in opise teh področij. *Uporaba podatkov* upošteva lastništvo tabel in arhivov, kako jih uporabnik razume in uporabi. Model podatkov vsebuje tudi informacije o tem kdaj so podatki veljavni, kdaj jih je potrebno zamenjati in podobo.

Uporaba splošnega modela podatkov

Definiranje modela podatkov za katerokoli aplikacijo je pogosto kompleksno in definiranje modela podatkov za podatkovno rudarjenje ni nobena izjema. Kadar je model podatkov potreben za podporo aplikaciji, ki ima posebne specifične zahteve, so podatki lahko definirani na način, da se vprašamo o koncu uporabe; kakšne informacije želimo in kakšne so lahko napravljene pretvorbe, da se zadosti tem zahtevam. V primeru podatkovnega rudarjenja je to lahko pravi izziv saj pogosto nismo prepričani katere spremenljivke vzeti na začetku, katere so pomembne in katere so želene. Tako je lahko gradnja modela podatkov za popolnoma novo aplikacijo podatkovnega rudarjenja časovno zelo dolgotrajna.

Druga možnost je, da se vzame splošni model podatkov razvit za reševanje podobnih poslovnih problemov. Takšen tip modela morda ne bo podal vseh informacij, ki smo jih želeli, ponudil pa nam bo spremenljivke, ki bi jih lahko dodatno vključili. Največja prednost uporabe splošnega modela podatkov je, da nas seznani z načinom kako videti uporabo podatkovnega rudarjenja v praksi.

5.3 Pridobivanje in preobdelava podatkov

Tretji korak splošne metode podatkovnega rudarjenja je pridobivanje in predelava podatkov, ki predstavljajo model podatkov. Definiran model podatkov priskrbi potrebno strukturo v smislu, da nam poda spremenljivke s katerimi bomo delali, še vedno pa moramo priskrbeti podatke.

Pridobivanje in obdelava podatkov temelji na *prepoznavanju, zbiranju, filtriranju in združevanju* podatkov v format, ki ga zahteva model podatkov in izbrana funkcija rudarjenja. Kadar so pridobivanje in preobdelava podatkov najbolj zamuden del v projektu podatkovnega rudarjenja je potrebno obsežno opisati ključne korake. Kadar pa podatki pridejo iz podatkovnega skladišča je večina tega že opravljenega.

Viri podatkov

Podatkovno rudarjenje ponavadi želi podatke v eni združeni tabeli ali datoteki. Če so zahtevane spremenljivke porazdeljene v večih virih je treba to združevanje oblikovati tako, da je pridobljen vsak niz podatkov. Če so podatki skladiščeni v relacijskih tabelah podatkovne baze je ustvarjanje nove tabele ali podatkovne baze relativno hitro, kompleksno združevanje pa ima lahko pomemben vpliv tudi na baze virov.

Preoblikovanje podatkov

Kadar podatki niso dobljeni iz skladišča podatkov je potrebno izvesti funkcijo preoblikovanja podatkov, ki zajema čiščenje, zbiranje, preoblikovanje in filtriranje. Včasih je potrebno izvesti takšne pretvorbe tudi kadar imamo skladiščene podatke. Orodja podatkovnega rudarjenja običajno omejijo čiščenje podatkov, saj je to specializiran proces, ki pa ga mnogo produktov lahko opravi učinkoviteje. Za združevanje in filtriranje podatkov poznamo veliko različnih načinov, način izberemo odvisno od strukture virov podatkov.

5.4 Vrednotenje modela podatkov

S populacijskim modelom podatkov še vedno ne moremo zagotoviti, da so podatki popolni, točni in relevantni. Zato s četrtim korakom splošne metode podatkovnega rudarjenja opravimo začetno ocenjevanje modela. Koraki vrednotenja so:

- Vizualni pregled - veljavnost izbranih spremenljivk;
- Manjkajoče vrednosti - ravnanje z vrednostmi, ki odstopajo;

- Selekcija spremenljivk - odstranjevanje odvečnih spremenljivk.

Prvi korak *vizualni pregled* obsega brskanje po vhodnih podatkih z vizualnimi orodji. To lahko pripelje do odkritja odstopanj. Primer: Pridružitev napačni tabeli v koraku priprave podatkov se lahko kaže v spremenljivkah, ki vsebujejo vrednosti, katere pripadajo drugemu polju.

Drugi korak se ukvarja z *identifikacijo neskladnosti* in *resolucijo napak.* Čuden zapis najden s prvim korakom je lahko posledica slabega zbiranja podatkov. Veliko odstopanj in manjkajočih vrednosti vrne prirejene vrednosti. Ublažitev odstopanj in pretvorba manjkajočih vrednosti v uporabne podatke precej izboljša kvaliteto podatkov. Največji problem manjkajočih vrednosti je, da nas lahko privedejo do pristranskih rezultatov rudarjenja in nezavednih napak.

Zadnji korak je *končna selekcija spremenljivk* za rudarjenje. Nekatere spremenljivke so lahko odveč, če predstavljajo zelo podobno informacijo in le tratijo čas. Odvisne ali korelirane spremenljivke najdemo s statističnimi testi, kot sta linearna in polinomska regresija. Odvisne spremenljivke je potrebno reducirati z izbiro ene ali z vpeljavo nove spremenljivke, ki korelira ostale s faktorjem ali komponentno analizo. Po statističnem preverjanju ostanejo le spremenljivke, ki imajo smisel in tiste z jasno interpretacijo. Ta korak vsebujejo le preizkušeni modeli podatkov. Selekcija spremenljivk v tej fazi se lahko dejansko izvede le s praktičnimi izkušnjami.

5.5 Izbira tehnike podatkovnega rudarjenja

Peti korak metode podatkovnega rudarjenja je ključni korak - izbira najprimernejše tehnike rudarjenja. Ta korak ne obsega le izbiro primerne tehnike ali mešanice tehnik ampak tudi način v katerem bo tehnika uporabljena. Izziv običajno ni odkriti katero tehniko izbrati ampak kateri algoritem te tehnike uporabiti. Vsaka od tehnik podatkovnega rudarjenja zahteva določeno obliko izbire parametrov, kar pa zahteva izkušnje, kako določena tehnika deluje in kaj različni parametri delajo.

5.6 Interpretacija rezultatov

Interpretacija rezultatov je šesti korak v splošni metodi rudarjenja. Rezultati podatkovnega rudarjenja so lahko včasih kompleksni zato je pomembno, da so predstavljeni tako, da so relativno lahko razumljivi. Pogosto je za interpretacijo rezultatov, kot pomoč, potrebno imeti orodja, ki priskrbijo potrebne statistične informacije za razlago.

5.7 Uporaba rezultatov

Sedmi korak splošne metode podatkovnega rudarjenja je v praksi najpomembnejši korak rudarjenja. Ukvarja se s vprašanjem kako pridobljene informacije uporabiti in pretvoriti v posel. Podatkovno rudarjenje ustvari matematično reprezentacijo podatkov kateri pravimo model. Ta model nas seznani z globino poslovnega problema, hkrati pa se lahko uporabi tudi v nadaljnjih poslovnih procesih v relacijskih bazah podatkov.

Poglavje 6

Podatkovno rudarjenje na primeru

Zlitina je trdna raztopina dveh ali več kovin. Dobimo jo tako, da osnovni kovini dodamo enega ali več *legirnih* elementov. Tako s spremembo kemijske sestave izboljšujemo mehanske lastnosti osnovnim kovinam. Zlitine označujemo po vnaprej določenem standardu z interno nomenklaturo, ki za vsako zlitino pove, v kakšnih mejah se morajo nahajati legirni elementi in nečistoče v zlitini.

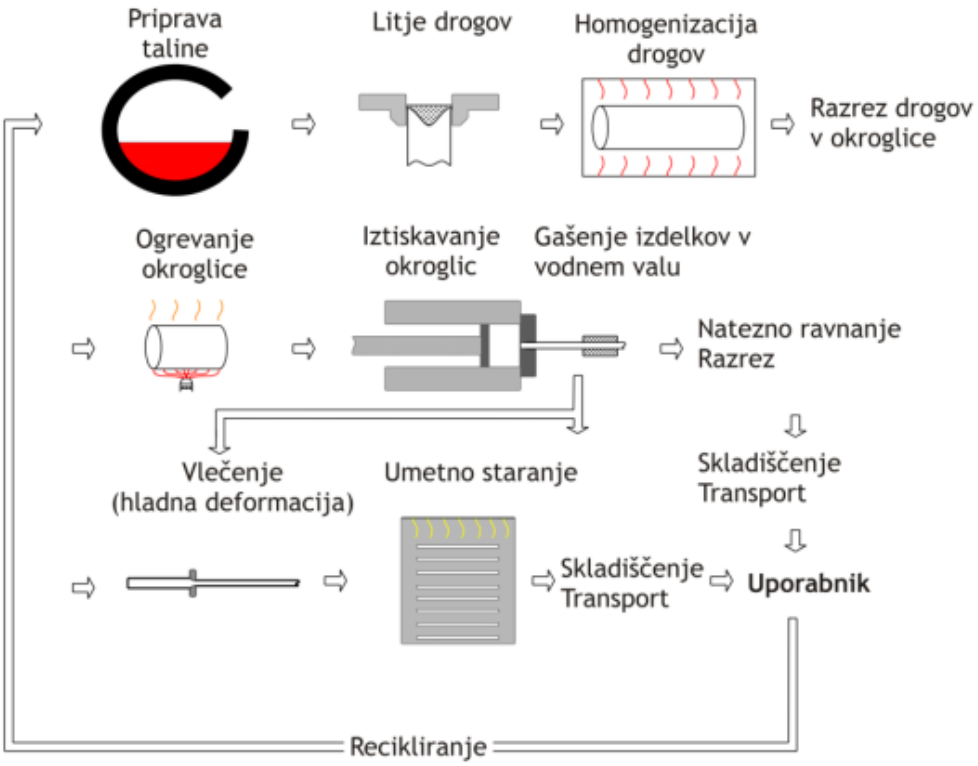
V praktičnem delu diplomskega dela smo obravnavali aluminijevo zlitino EN AW-7075 za katero podjetje Impol uporablja interno oznako PD30. To je zlitina skupine Al-Zn-Mg-Cu (serija 7.000 po klasifikaciji ASTM), *perduali*. Glavni zlitinski elementi so Zn (4-7%), Mg(0,8-3%), Cu(do 2%), Al je osnovna kovina. Stranski zlitinski elementi so Mn, Cr in Zr. Te zlitine so ene najtrših, vendar slabo korozijsko obstojne. Največ se uporabljajo v vojaški industriji za izdelavo strelnega orožja, v letalstvu za razne konstrukcijske elemente ter na avtocestah za podporne ograje.

Šarža je količina materiala, ki se da ob enem polnjenju v peč. Pri tem mešamo povratne materiale različnih zlitin, dodajamo čisti aluminij in različne legirne elemente z namenom, da bo kemijska sestava dobljene šarže ustrezna, oziroma da bomo dobili ustrezno zlitino. Legirne elemente dodajamo v talino po potrebi. Njihova količina je odvisna od kemične sestave vsade (vložka), ki ga zalagamo v talilno peč. Če dodajamo več zlitinsko sorodnega materiala nam ni potrebno dodati veliko legirnih elementov. Legirni elementi so v primerjavi z zlitinami bistveno dražji zato je cilj, da se jih porabi čim manj.

Talina vsake šarža gre v *kemijsko analizo* kjer se s kvantometrom preverijo koncentracije posameznih kemijskih elementov. Če kemijska sestava ni v dovoljenih mejah se dodaja čisti aluminij za redčenje šarže ali legirni elementi. Ko je sestava šarže ustrezna se talina lije. V Impolu lijejo dve obliki in sicer drogove in brame. Drogovi so vhodna oblika za

tipično procesno verigo vročega iztiskavanja palici. To sestavlja več procesov, ki vplivajo na mehanske lastnosti izdelka.

Procesna veriga izdelave iztisnjenih izdelkov iz aluminijastih zlitin je sestavljena iz več tehnoloških operacij pri katerih se v posamezne baze podatkov shranjujejo procesni parametri, ki krmilijo posamezni proces. Izdelavo razdelimo na *livarniški del*, kjer se iz povratnega materiala, čistega aluminija in legirnih elementov izdela ustrezna zlitina, ki se nato lije v drogove in *vroče iztiskavanje*, kjer se drogovi razrežejo v ustrezne dolžine (okroglice), te se nato ogrejejo, iztisnejo in razrežejo na tehnološke dolžine. Nadaljnja tehnološka pot je odvisna od zlitine in želenih mehanskih lastnosti.



Slika 6.1: Procesna veriga izdelave iztisnjenih palic

(Slika vir: [4])

Za določanje mehanskih lastnosti končnega izdelka se uporabi določeno število *preizkušancev,* ki se pošljejo v *mehanski laboratorij*. Preizkušanci so predhodno obdelani, praviloma krogelnega ali pravokotnega prereza. Na celotnem srednjem stanjšanem delu imajo konstanten prerez S_0 in so dolžine L_0. Prerez v glavi je povečan, prehodi so izpeljani s primernim zaobljenjem. V mehanskem laboratoriju se izvede *natezni preizkus* s katerim se ugotavljajo

trdnostne lastnosti kovin in zlitin, napetost tečenja (meja plastičnosti), natezna trdnost in modul elastičnosti. Pri začetnem obremenjevanju se materiali raztezajo elastično, to pomeni, da po razbremenitvi razteg izgine, pri čemer je ta elastični raztezek premosorazmeren z napetostjo (Hookov zakon). Mejno napetost nad katero se začenjajo kovine plastično deformirati imenujemo *napetost tečenja* (nekdaj meja plastičnosti). Določamo jo z napetostjo pri kateri po razbremenitvi ostane določen najmanjši delež trajnega raztezka. Dogovorjeno je, da je za diagrame brez izrazite napetosti tečenja pri $\epsilon = 0,2$ trajne deformacije, napetost tečenja $R_{0,2}$. Ob nadaljnji deformaciji v področje plastičnosti se kovine utrjujejo in je za nadaljnjo deformacijo potrebna večja obremenitev oziroma napetost. Iz največje obremenitve (F_m), ki jo material zdrži, izračunamo *natezno trdnost* $R_m = \frac{F_m}{S_0}$. Natezni preizkus delamo dokler se preizkušanec ne poruši, pri tem pa določimo napetost tečenja, natezno trdnost ter *raztezek*. Po pretrgu združimo pretrgana dela in s pomičnim merilom izmerimo končno merilno dolžino L_u in izračunamo *razteznost* ali *raztezek* ob pretrgu $A = \frac{L_u - L_0}{L_0} = \frac{\Delta L_0}{L_0}$.

V praktičnem delu diplomskega dela smo na kemijskih, mehanskih in ostalih tehnoloških podatkih različnih izdelkov iz zlitine PD30 izvedli algoritem podatkovnega rudarjenja ter poskušali poiskati vpliv kemijske sestave na mehanske lastnosti. Ker sta mehanski lastnosti *napetost tečenja in natezna trdnost* v korelaciji, smo učenje ločili na dva dela in ga izvedli za vsako lastnost posebej, *raztezek* pa pridružili eni od njiju.

6.1 Splošna metoda podatkovnega rudarjenja na primeru

6.1.1 Definiranje poslovnega problema

Obvladovanje procesov v procesni verigi izdelave izdelkov predstavlja največji izziv inženirjem v proizvodnem procesu, kot tudi raziskovalcem, ki rešujejo probleme pri izdelavi izdelkov. Brez povezave procesov v celoto in razumevanje zgodovine procesnih parametrov na lastnosti materiala, je nemogoče ugotoviti vzročne povezave na ciljne karakteristike izdelkov. Pri izdelavi polizdelkov iz zlitin pa je še toliko pomembneje poznavanje vplivov.

Tipično procesno verigo izdelave Al iztiskanih palic sestavlja več procesov, ki vplivajo na lastnosti izdelka, kot so:

- Litje drogov (sestava vsade, kemična sestava zlitine, parametri taljenja, parametri priprave zlitine, parametri litja, parametri homogenizacije)

- Parametri ogrevanja okroglic
- Parametri iztiskavanja (tip procesa (direktno-indirektno iztiskavanje), parametri nastavitve stroja)
- Parametri gašenja
- Parametri termične obdelave po procesu vročega preoblikovanja

Nastavitve procesnih parametrov so seveda pogojene s kupčevimi zahtevami po lastnostih, ki jih mora izdelek izpolnjevati. Te zahteve se nanašajo tako na mehanske lastnosti, kot tudi na zahteve po uniformnosti mikrostrukture izdelkov.

Pri procesih, ki so informacijsko podprti, nastajajo obsežne baze procesnih in tehnoloških parametrov. Pri vplivu posameznega procesnega parametra na kvaliteto izdelka so procesni inženirji, brez sodobnih numeričnih orodij, prepuščeni sami sebi in se lahko zanašajo predvsem na bogate izkušnje iz dela v proizvodnji.

Cilj naloge je preveriti numerične metode na bazah procesnih parametrov in poiskati najučinkovitejšo v smislu iskanja povezav vplivov na kvaliteto izdelkov in uporabe za industrijske aplikacije.

(dr. Peter Cvahte)

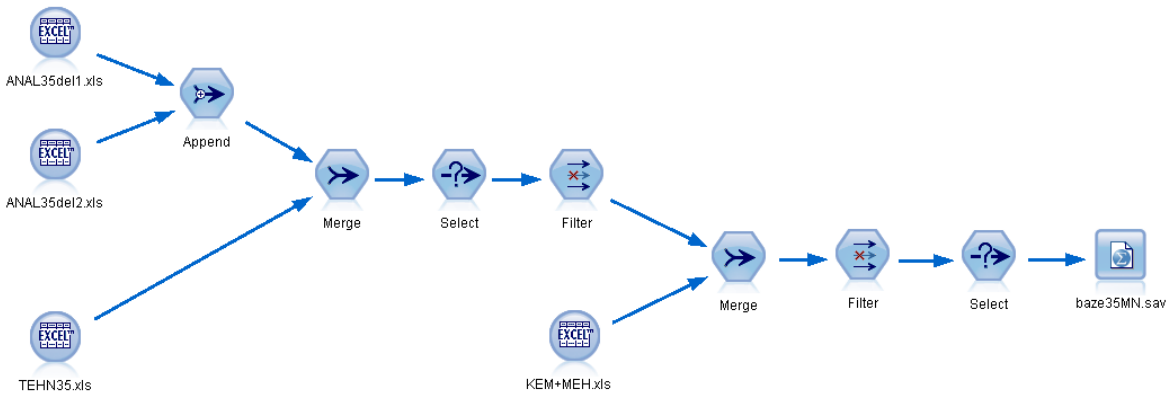
6.1.2 Model podatkov

V procesu izdelave aluminijastih izdelkov na posameznem koraku procesne verige nastajajo baze procesnih parametrov. Baze podatkov, ki so bile uporabljene v analizi so naslednje:

- Kemijska sestava zlitine za posamezne okroglice:
  - Livarniški del procesne verige
  - Procesni parametri: kemijska sestava vložka, legirni elementi
- Parametri litja:
  - Livarniški del procesne verige
  - Procesni parametri: dimenzija droga, hitrost in temperatura litja
- Parametri homogenizacije:
  - Livarniški del procesne verige
  - Procesni parametri: čas in temperatura, hitrost ohlajanja

- Tehnološka baza (v tej bazi so zapisani predvideni parametri termo-mehanske predelave):
  - Vroče iztiskavanje
  - Procesni parametri: temperatura in čas ogrevanja okroglic, hitrost pomika bata, število iztisnjenih žil, iztiskalno razmerje, hitrost gašenja palic
- Baza analognih podatkov (v tej bazi so zapisani izmerjeni realni parametri med termo-mehansko predelavo):
  - Vroče iztiskavanje
  - Procesni parametri: temperatura in čas ogrevanja okroglic, hitrost pomika bata, število iztisnjenih žil, iztiskalno razmerje, hitrost gašenja palic
- Baza mehanskih lastnosti:
  - Mehanski laboratorij
  - Procesni parametri: napetost tečenja, natezna trdnost, raztezek

Iz teh podatkov je bila tvorjena nova baza podatkov zlitine PD30 za leto 2010. Za podatkovno rudarjenje, ki zajema pripravo podatkov in izvedbo algoritma, je bilo uporabljeno orodje SPSS Modeler. Združevanje baz prikazuje slika 6.2.

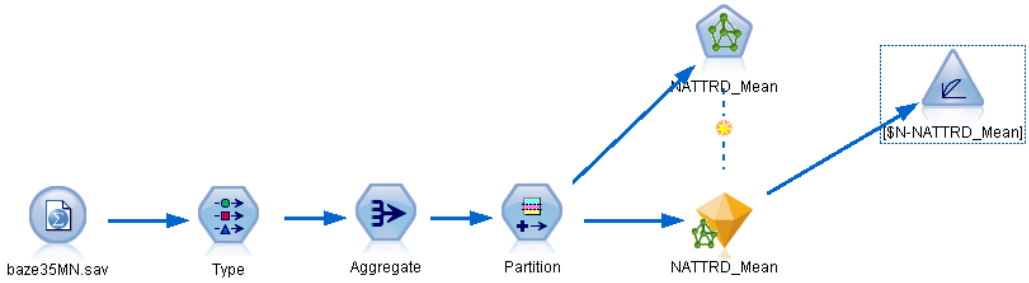


Slika 6.2: Združevanje baz

6.1.3 Preobdelava podatkov

Baze podatkov so bile dobljene v podjetju Impol d.o.o. Model podatkov je bil zgrajen z združevanjem posameznih baz (slika 6.2). Vhodno tabelo je bilo potrebno urediti tako, da je bila pripravljena za rudarjenje, pri tem pa je bilo potrebno paziti, da so se agregirali potrebni podatki.

Praviloma se za izdelavo posameznega izdelka (delovni nalog) uporabi vhodni material ene šarže. Če naročena količina posameznega izdelka presega količino vhodnega materiala iz ene šarže, se v procesu izdelave posameznega izdelka uporabi več šarž iste zlitine. Lahko pa količina vhodnega materiala ene šarže zadošča za izdelavo več različnih izdelkov (delovnih nalogov). Tako je bilo potrebno podatke pravilno agregirati. SPSS vsebuje potrebne gradnike, ki avtomatsko agregirajo, lepijo, filtrirajo. Pripravo podatkov in izvedbo algoritma prikazuje shema 6.3.



Slika 6.3: Priprava podatkov in izvedba algoritma

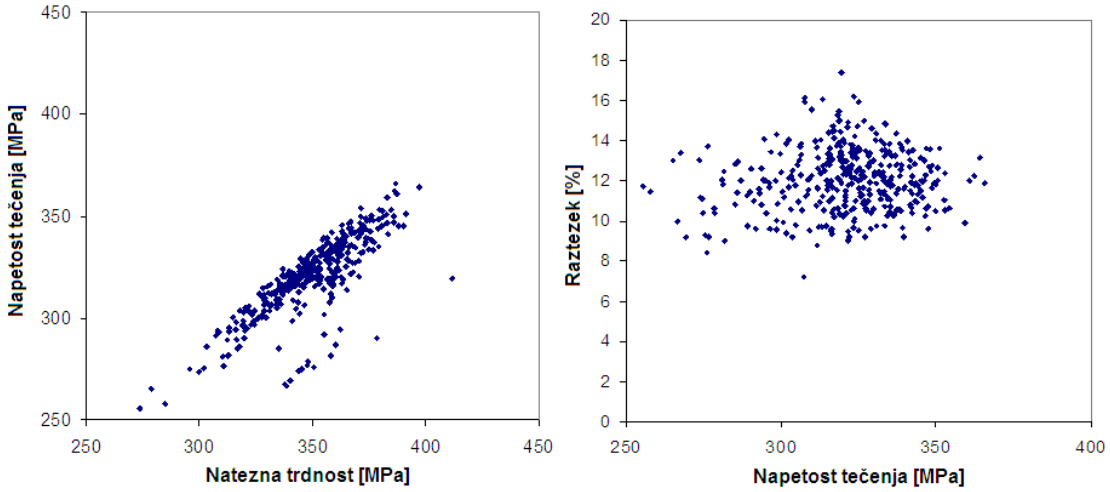
Prvi gradnik *baze35MN.sav* prikazuje vhodno tabelo po tem ko so bile združene vse baze. V gradniku *Type* se je določil tip podatkov (za nevronsko mrežo morajo biti vhodni podatki zvezni), odpravile so se odstopajoče, manjkajoče in ničelne vrednosti. Z gradnikom *Aggregate* so bili agregirani podatki po delovnem nalogu in znotraj tega še po šarži. Za agregirane podatke se je uporabila povprečna vrednost. Z gradnikom *Partition* se je razdelilo podatke na učno množico (70%) in testno množico (30%).

Novonastala združena tabela je vsebovala vse parametre, ki bi lahko vplivali na končne mehanske lastnosti, teh je skupaj 55. V nadaljnjem procesu obdelave je bilo uporabljenih 101.771 zapisov za katere so bili znani vsi podatki.

6.1.4 Vrednotenje modela podatkov

Vsi dobljeni podatki v tabeli niso bili relevantni. S pomočjo SPSS gradnika *Data Audit* se je preverila kvaliteta podatkov. Odkritih je bilo nekaj izstopajočih vrednosti in manjkajočih vrednosti. Z gradnikom *Type* so bile odpravljene nepravilnosti. Podatki so bili prečiščeni pred agregiranjem v eno tabelo (korak 4 pred korakom 3), saj je bilo pri agregiranju uporabljeno povprečje, ki pa ob neočiščenih podatkih ne bi bilo pravilno. Po agregiranju je bilo dobljenjih 675 podatkov, ki so bili pripravljeni na algoritem podatkovnega rudarjenja.

Izhodni parametri analize so *napetost tečenja*, *natezna trdnost* in *raztezek*. Med napetostjo tečenja in natezno trdnostjo obstaja močna korelacija, med napetostjo tečenja in raztezkom ter natezno trdnostjo in raztezkom pa ni nobene očitne korelacije, kar je razvidno iz slike 6.4. Tako je bila analiza ločena na dva dela. V prvem delu je bila kot izhodni parameter izbrana napetost tečenja [MPa], v drugem delu pa raztezek [%] in natezna trdnost [MPa] hkrati.



Slika 6.4: Korelacija med parametri

6.1.5 Izbira tehnike podatkovnega rudarjenja

Cilj je napovedati napetost tečenja, raztezek, in natezno trdnost, ki so ciljne spremenljivke, na podlagi podanih atributov. Vrednosti parametrov so numerične, zvezne vrednosti, tako so bile za algoritem podatkovnega rudarjenja izbrane nevronske mreže, ki popišejo n-dimenzionalen prostor in so zmožne najti zapletene medsebojne povezave, ki jih je sicer

pri tako velikem številu parametrov težko intuitivno obdelati. Pokažejo lahko področja, ki so za procese ugodna in tista, ki bi se jim veljajo med procesi izogibati.

6.1.6 Interpretacija rezultatov

Model nevronske mreže poda kot rezultat povzetek in sliko nevronske mreže, na podlagi katerih lahko ovrednotimo model, ne vidimo pa eksplicitnih enačb s katerimi model napoveduje. Da bi razumeli delovanje mreže in našli enačbo napovedi smo model izvozili v PMML kodo. PMML je standardni jezik za produkcionalizacijo (deploy) modelov. S pomočjo znanja osnovnih standardnih ukazov je mogoče razbrati postopek in zaporedje delovanja modela.

Primer s katerim se srečujemo v diplomskem delu je dokaj kompleksen, zato smo si delovanje mreže predstavili in razjasnili na preprostejšem primeru, katerega smo lahko preverili tudi računsko 'na roko'. Model smo naučili računati kvadratno funkcijo in nato želeli poiskati enačbo po kateri model napoveduje. Iz kode PMML smo pridobili uteži za računanje. Rezultat kvadratov smo poskušali napovedati s pomočjo enačb 4.3 in 4.4. Napoved modela in naši prvotni rezultati se niso ujemali. Po ponovnem preverjanju PMML kode smo ugotovili, da model ne računa z dejanskimi vrednostmi vhodnih spremenljivk. Vse vhodne vrednosti, ki nastopajo v omenjenih enačbah 4.3 in 4.4 se najprej normalizirajo z linearno normalizacijo. Izračunana vrednost se nato denormalizira in poda kot napoved modela nevronske mreže. Po tej ugotovitvi smo lahko zapisali točno enačbo za napoved vrednosti kvadrata danega števila in jo preverili tudi računsko 'na roko'.

Za modela diplomskega dela izračuna 'na roko' nismo preverili, saj bi bil postopek precej dolgotrajen. Na koncu tega dela je kot priloga dodana PMML koda uteži za model napetosti tečenja, s katero je takšen izračun mogoče napraviti.

Na mehanske lastnosti izdelkov iz zlitine PD30 vpliva veliko parametrov. Zaradi obvladljivosti analize in optimalnosti obdelave razpoložljive količine podatkov so bili na podlagi znanih fizikalnih dejstev izbrani le najpomembnejši vhodni parametri. Iz baze podatkov so bili pri analizi uporabljeni naslednji parametri:

$x_1 \cdots$ T_PLIN_Mean - temperatura v plinski peči

$x_2 \cdots$ T_IS_Mean - temperatura v indukcijski peči spredaj

$x_3 \cdots$ T_ISR_Mean - temperatura v indukcijski peči sredina

$x_4 \cdots$ T_IZ_Mean - temperatura v indukcijski peči zadaj

$x_5 \cdots$ V_BAT_Mean - hitrost bata

$x_6 \cdots V\_PUL\_Mean$ - hitrost pullerja

$x_7 \cdots R\_Mean$ - iztiskalno razmerje

$x_8 \cdots Fe\_Mean$ - % železa v zlitini

$x_9 \cdots Si\_Mean$ - % silicija v zlitini

$x_{10} \cdots Mn\_Mean$ - % mangana v zlitini

$x_{11} \cdots Mg\_Mean$ - % magnezija v zlitini

$x_{12} \cdots Cu\_Mean$ - % bakra v zlitini

$x_{13} \cdots Zn\_Mean$ - % cinka v zlitini

$x_{14} \cdots Cr\_Mean$ - % kroma v zlitini

$x_{15} \cdots Ti\_Mean$ - % titana v zlitini

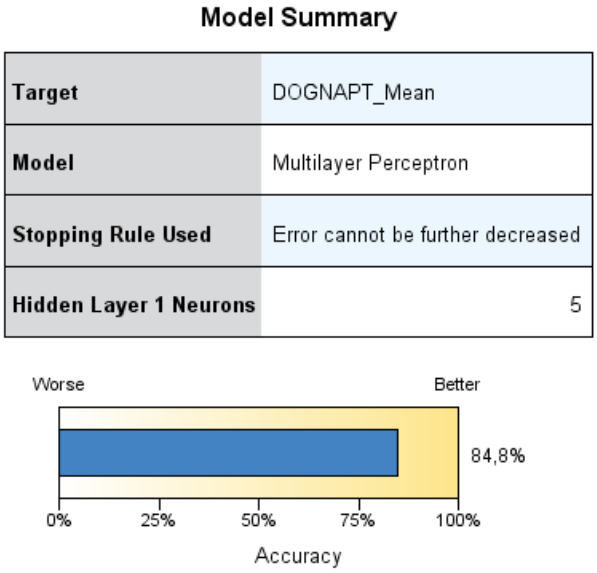
$x_{16} \cdots Pb\_Mean$ - % svinca v zlitini

$x_{17} \cdots Bi\_Mean$ - % bizmunta v zlitini

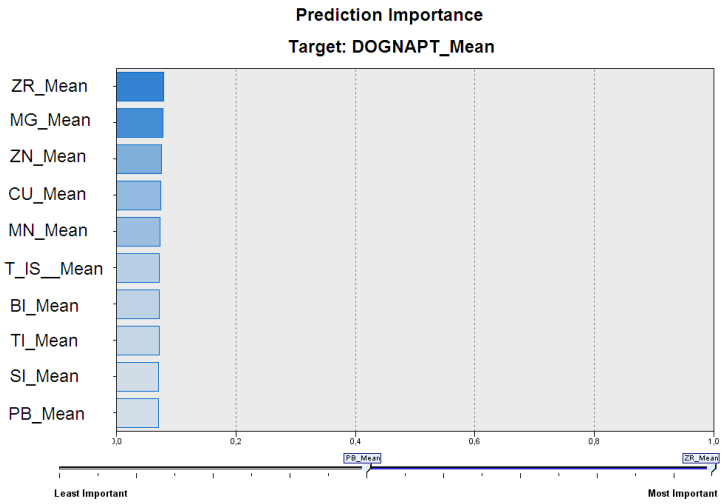
$x_{18} \cdots Zr\_Mean$ - % cirkonija v zlitini

NAPETOST TEČENJA

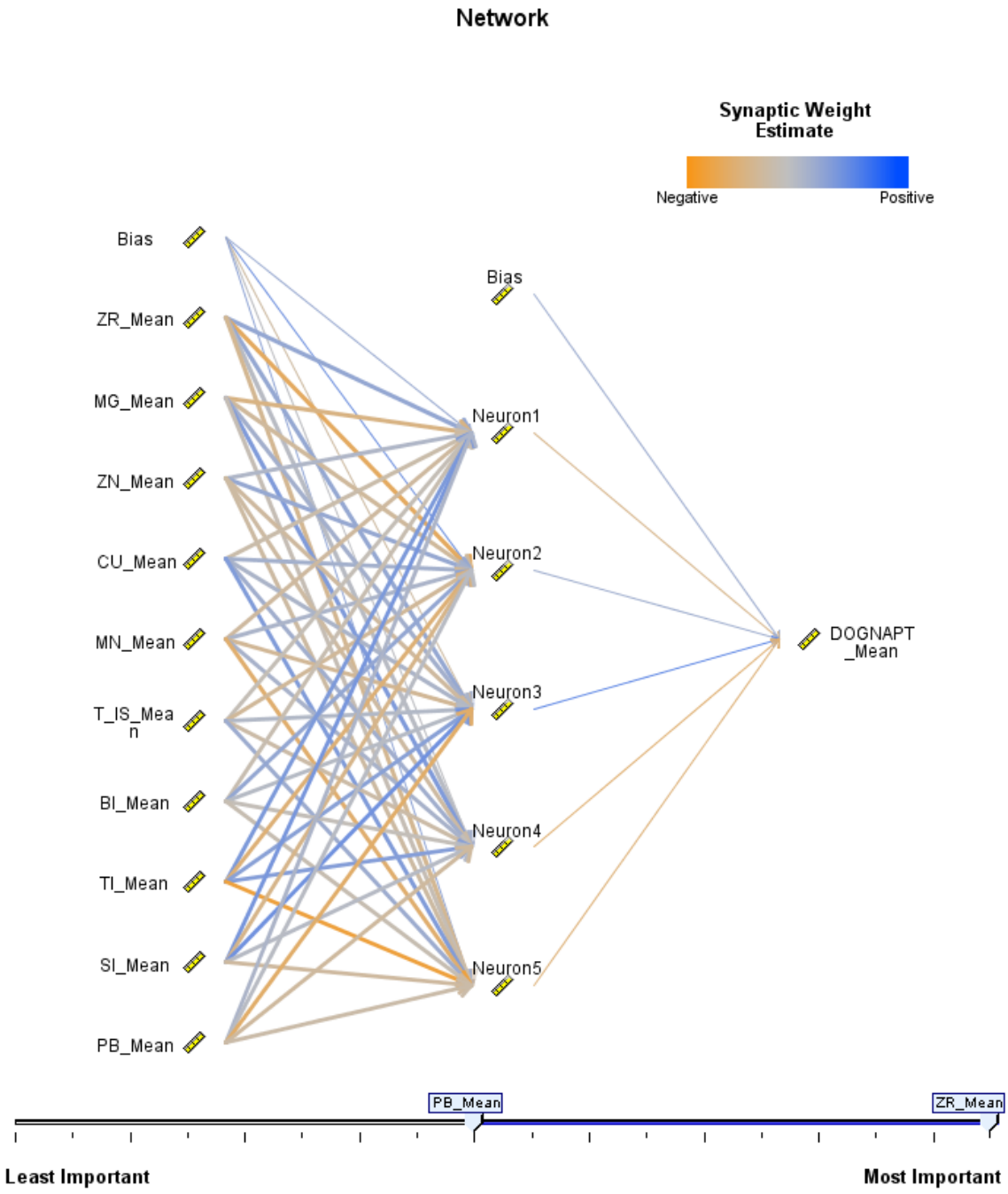
a) Model



Slika 6.5: Povzetek modela



Slika 6.6: Najpomembnejši prediktorji



Slika 6.7: Nevronska mreža

b) Enačba napovedi

Delovanje nevronske mreže opišemo z že znanima enačbama:

$a_k = \sum_{i=0}^{n} w_{ki} \cdot x_i$

$y_k = \varphi(a_k)$

Kot vhodi nevronov so bili izbrani zgoraj opisani parametri $x_1, \ldots x_{18}$. Izbran model je multilayer perception, ki računa uteži po metodi vzvratnega širjenja napake. Aktivacijska funkcija $\varphi(a)$ za računanje skritega nivoja je hiperbolični tangens, za izhod iz nevrona pa identiteta. Izbira aktivacijskih funkcij in vrednosti izračunanih uteži so vidni iz PMML kode, ki je v prilogi. Po ugotovitvah zgoraj te uteži podajo napoved le, če v enačbo vnesemo normalizirane vrednosti parametrov $x_1, \ldots x_{18}$.

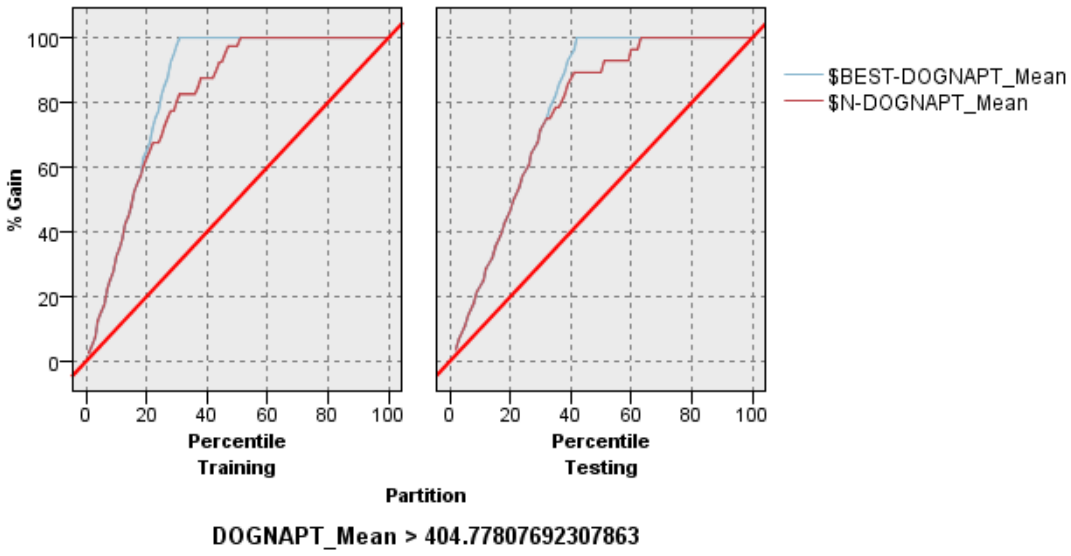
c) Vrednotenje modela

Results for output field DOGNAPT_Mean

Comparing $N-DOGNAPT_Mean with DOGNAPT_Mean

Minimum Error	-210,927
Maximum Error	106,182
Mean Error	0,837
Mean Absolute Error	11,696
Standard Deviation	20,012
Linear Correlation	0,921
Occurrences	675

Slika 6.8: Analiza modela



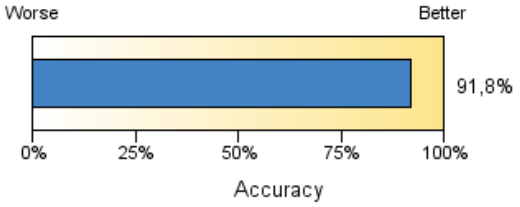
Slika 6.9: Gain krivulja

RAZTEZEK in NATEZNA TRDNOST

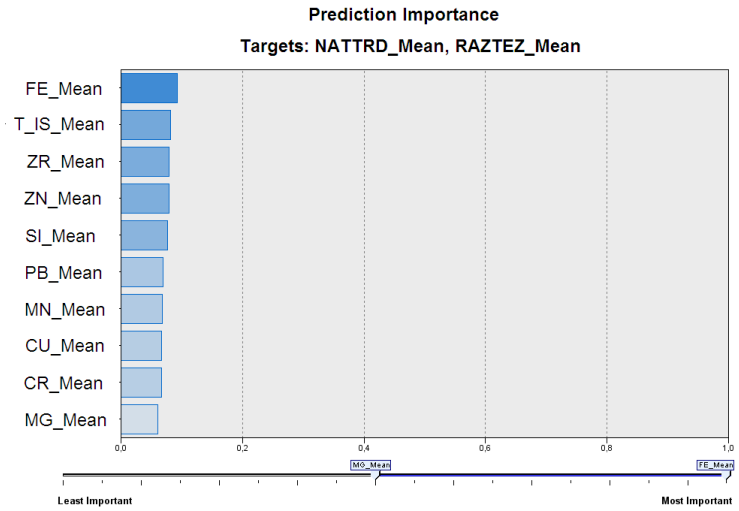
a) Model

Model Summary

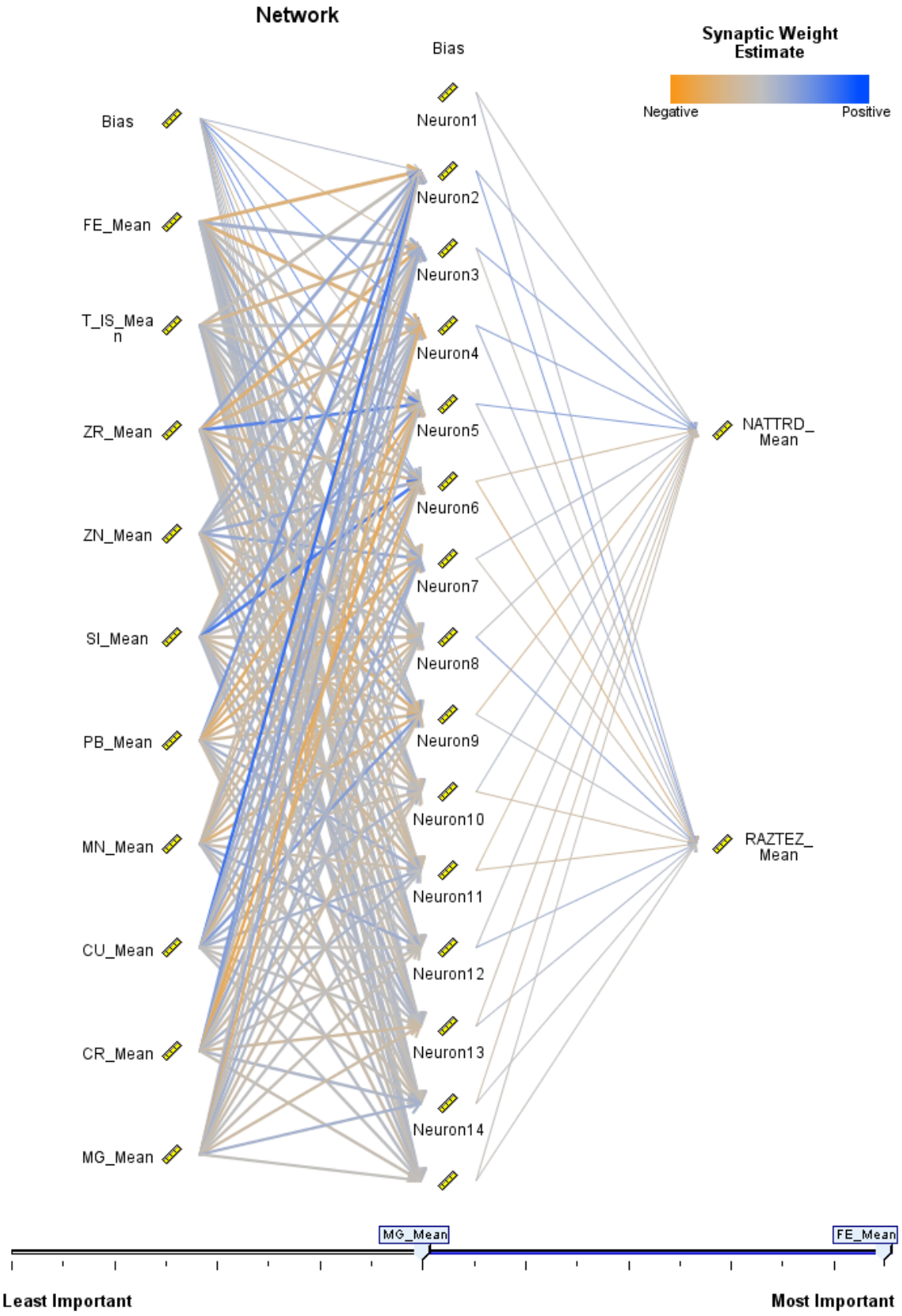
Targets	NATTRD_Mean RAZTEZ_Mean
Model	Multilayer Perceptron
Stopping Rule Used	Error cannot be further decreased
Hidden Layer 1 Neurons	14



Slika 6.10: Povzetek modela



Slika 6.11: Najpomembnejši prediktorji



Slika 6.12: Nevronska mreža

b) Enačba napovedi

Tudi tukaj so bili kot vhodi izbrani zgornji parametri $x_1, \ldots x_{18}$, multilayer perceptron in aktivacijski funkciji hiperbolični tangens za skriti nivo in identiteta za izhodni.

c) Vrednotenje modela

Results for output field NATTRD_Mean

Comparing $N-NATTRD_Mean with NATTRD_Mean

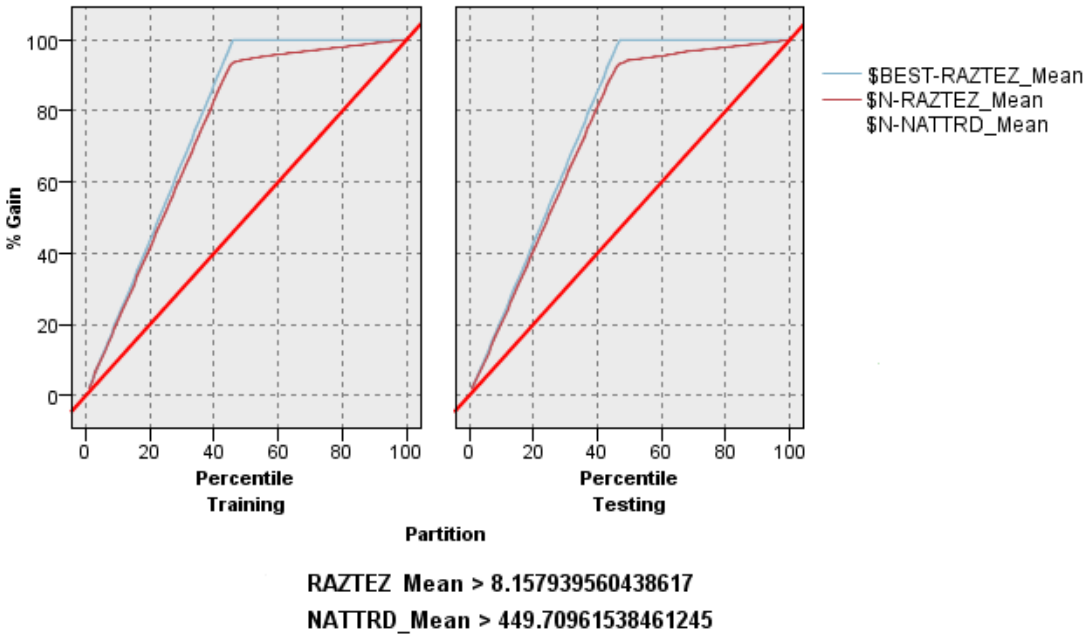
Minimum Error	-140,752
Maximum Error	110,627
Mean Error	0,167
Mean Absolute Error	9,171
Standard Deviation	16,458
Linear Correlation	0,946
Occurrences	675

Results for output field RAZTEZ_Mean

Comparing $N-RAZTEZ_Mean with RAZTEZ_Mean

Minimum Error	-1,487
Maximum Error	1,256
Mean Error	0,007
Mean Absolute Error	0,177
Standard Deviation	0,256
Linear Correlation	0,97
Occurrences	675

Slika 6.13: Analiza modela



Slika 6.14: Gain krivulja

6.1.7 Povzetek in uporaba rezultatov

Napovedovanje mehanskih lastnosti za aluminijske polproizvode na podlagi fizikalnega numeričnega modeliranja celotnega procesa je izjemno zapleteno, predvsem zaradi večnivojskega in večfaznega obnašanja materiala. V diplomskem delu je skozi proces podatkovnega rudarjenja opisano modeliranje z nevronskimi mrežami, kot alternativni pristop k fizikalnemu modeliranju.

Podatki za učenje nevronskih mrež so bili zbrani v podjetju Impol d.o.o. Po združitvi različnih baz in agregiranju podatkov je bilo uporabljenih 675 množic zgodovinskih podatkov za zlitino EN AW-7075 (interna oznaka PD30). Naučeni nevronski mreži lahko napovedujeta mehanske lastnosti *napetost tečenja ($R_{0,2}$), natezna trdnost (R_m)* in *raztezek (A)* kot funkcijo procesnih parametrov. Vhodni parametri nevronskih mrež so kemijske sestave v %: $1-Fe$, $2-Si$, $3-Mn$, $4-Mg$, $5-Cu$, $6-Zn$, $7-Cr$, $8-Ti$, $9-Pb$, $10-Bi$, $11-Zr$ in ostali procesni parametri: 12-temperatura v plinski peči, 13-temperatura v indukcijski peči spredaj, 14-temperatura v indukcijski peči sredina, 15-temperatura v indukcijski peči zadaj, 16-hitrost bata, 17-hitrost pullerja, 18-stiskalno razmerje. Kot izhodni parametri so izbrani napetost tečenja, natezna trdnost in raztezek. Točnost napovedi modela nevronske mreže za napetost tečenja je $84,8\%$, točnost napovedi modela za natezno trdnost in raztezek pa $91,8\%$.

S predstavljenima modeloma nevronskih mrež je pokazano, da lahko podjetje Impol razvije model za ocenjevanje končnih mehanskih lastnosti, kot funkcijo procesnih parametrov. S tem je omogočena optimizacija procesne poti glede na produktivnost in kvaliteto.

Literatura

[1] C. Baragoin, C. M. Andersen, S. Bayerl, G. Bent, J. Lee, C. Schommer, *Mining Your Own Business in Retail*, Redbooks, California, 2001.

[2] M. J. A. Berry, G. Linoff, *Mastering Data Mining: The Art and Science of Customer Relationship Management*, Wiley Computer Publishing, Canada, 2000.

[3] P. J. Brockwell, *Introduction to time series and forecasting*, Springer Science + Business Media, LLC, 2002.

[4] P. Cvahte, *Vezani termomehanski in metalurški procesi med iztiskavanjem Al-zlitin : doktorska disertacija*, Ljubljana, 2009.

[5] A. Dobnikar, *Nevronske mreže: teorija in aplikacije*, Didakta, Radovljica, 1990.

[6] K. D. Lawrence, S. Kudyba, R. K. Klimberg, *Data Mining Methods and Applications*, Auerbach Publications, New York, 2008.

[7] J. Reinschmidt, R. Bhattacharya, P. Harris, A. Karanasos, *Mining Relational and Nonrealational Data with IBM Intelligent Miner for Data Using Oracle, SPSS, and SAS As Sample Data Source*, Redbooks, California, 1998.

[8] I. H. Witten, E. Frank , *Data Mining: Practical Machine Learning Tools and Techniques, Second edition*, Morgan Kaufmann Publishers, San Francisco, 2005.

[9] M. Zorman, V. Podgorelec, M. Lenič, P. Povalej, P. Kokol, A. Tapajner, *Inteligentni sistemi in profesionalni vsakdan*, Univerza v Mariboru, Maribor, 2003.

Priloga

PMML koda uteži za NAPETOST TEČENJA

```
- <NeuralNetwork activationFunction="tanh"
algorithmName="MLP" functionName="regression">
  <Extension extender="spss.com" name="modelID" value="0" />
- <NeuralLayer numberOfNeurons="5">
- <Neuron bias="0.81173141002486" id="18">
  <Con from="0" weight="-0.343740344782638" />
  <Con from="1" weight="-0.672479992372019" />
  <Con from="2" weight="-0.46354218699515" />
  <Con from="3" weight="-0.186582599432619" />
  <Con from="4" weight="-1.1772009938575" />
  <Con from="5" weight="-0.683041525514692" />
  <Con from="6" weight="0.431299076872977" />
  <Con from="7" weight="-0.106898435303416" />
  <Con from="8" weight="1.36510006008112" />
  <Con from="9" weight="1.27837116327612" />
  <Con from="10" weight="0.0122655418038363" />
  <Con from="11" weight="-0.208744281659928" />
  <Con from="12" weight="0.0461150698255143" />
  <Con from="13" weight="0.269725419713087" />
  <Con from="14" weight="-0.0496176663419704" />
  <Con from="15" weight="0.060899171944713" />
  <Con from="16" weight="0.35345033892796" />
  <Con from="17" weight="0.892430721400142" />
  </Neuron>
- <Neuron bias="1.515269554504" id="19">
  <Con from="0" weight="0.890811796618956" />
  <Con from="1" weight="-0.445759468382609" />
```

```
  <Con from="2" weight="0.596853182403659" />
  <Con from="3" weight="0.29836811922169" />
  <Con from="4" weight="-0.739390419892352" />
  <Con from="5" weight="0.437107567193032" />
  <Con from="6" weight="0.167404471528639" />
  <Con from="7" weight="0.00158602728464687" />
  <Con from="8" weight="-1.06207955351241" />
  <Con from="9" weight="-1.39647817881233" />
  <Con from="10" weight="-0.594970803377965" />
  <Con from="11" weight="-0.789998072320312" />
  <Con from="12" weight="-0.568027892317452" />
  <Con from="13" weight="0.254161160684885" />
  <Con from="14" weight="-0.0127721554660887" />
  <Con from="15" weight="-0.276732844814492" />
  <Con from="16" weight="0.844633437184317" />
  <Con from="17" weight="-1.73601952306473" />
  </Neuron>
- <Neuron bias="-0.650454895301956" id="20">
  <Con from="0" weight="0.222390806031347" />
  <Con from="1" weight="-1.54797623244001" />
  <Con from="2" weight="0.433603605645879" />
  <Con from="3" weight="-0.2526286884147" />
  <Con from="4" weight="0.606512209007823" />
  <Con from="5" weight="-0.853696249303062" />
  <Con from="6" weight="-1.60438627117624" />
  <Con from="7" weight="-0.0294376444634329" />
  <Con from="8" weight="1.47177277934209" />
  <Con from="9" weight="1.07555120087873" />
  <Con from="10" weight="0.440193780641727" />
  <Con from="11" weight="0.312416914183609" />
  <Con from="12" weight="0.0681044840352135" />
  <Con from="13" weight="0.268994724120378" />
  <Con from="14" weight="0.0284879892586712" />
  <Con from="15" weight="-0.0236561301326746" />
  <Con from="16" weight="-0.685908298361993" />
  <Con from="17" weight="0.862392900006776" />
  </Neuron>
- <Neuron bias="0.0776245486540568" id="21">
```

```
  <Con from="0" weight="-0.167314607663114" />
  <Con from="1" weight="-1.07477105428919" />
  <Con from="2" weight="0.6478713022447" />
  <Con from="3" weight="0.344847635031972" />
  <Con from="4" weight="1.2361632981982" />
  <Con from="5" weight="0.510375062109962" />
  <Con from="6" weight="-0.702885669454526" />
  <Con from="7" weight="0.0721493927888712" />
  <Con from="8" weight="0.209621647935351" />
  <Con from="9" weight="1.15537491054201" />
  <Con from="10" weight="-0.493902378427576" />
  <Con from="11" weight="-0.529978914348073" />
  <Con from="12" weight="-0.217184642664497" />
  <Con from="13" weight="-0.25788476838275" />
  <Con from="14" weight="0.0932410221720569" />
  <Con from="15" weight="-0.064855964366585" />
  <Con from="16" weight="-0.626851425389713" />
  <Con from="17" weight="0.161491314053544" />
  </Neuron>
- <Neuron bias="0.998903671364493" id="22">
  <Con from="0" weight="-0.311745568033318" />
  <Con from="1" weight="0.632388728358372" />
  <Con from="2" weight="1.21732162832101" />
  <Con from="3" weight="0.156840980698819" />
  <Con from="4" weight="-0.257800574506242" />
  <Con from="5" weight="-1.52050223115109" />
  <Con from="6" weight="-0.501815258092998" />
  <Con from="7" weight="0.00649157586724195" />
  <Con from="8" weight="-0.689427431884163" />
  <Con from="9" weight="-2.21710943864157" />
  <Con from="10" weight="0.597059179398823" />
  <Con from="11" weight="0.724829872537067" />
  <Con from="12" weight="-0.00307128934273279" />
  <Con from="13" weight="-0.230217943705679" />
  <Con from="14" weight="-0.0155405008746353" />
  <Con from="15" weight="-0.00493820690694506" />
  <Con from="16" weight="-0.623106749350197" />
  <Con from="17" weight="-0.920210296714143" />
```

```
  </Neuron>
  </NeuralLayer>
- <NeuralLayer activationFunction="identity" numberOfNeurons="1">
- <Neuron bias="0.725757351889708" id="23">
  <Con from="18" weight="-1.0852429672044" />
  <Con from="19" weight="0.699168128486036" />
  <Con from="20" weight="1.77267168399435" />
  <Con from="21" weight="-1.50742226947986" />
  <Con from="22" weight="-1.39265242190374" />
  </Neuron>
```