UNIVERZA V LJUBLJANI

Fakulteta za strojništvo

Zmanjševanje nepopolnosti proizvodnih podatkov z metodo strojnega učenja

Magistrsko delo magistrskega študijskega programa II. stopnje Strojništvo

Diko Purić

Ljubljana, februar 2018

UNIVERZA V LJUBLJANI

Fakulteta za strojništvo

Zmanjševanje nepopolnosti proizvodnih podatkov z metodo strojnega učenja

Magistrsko delo magistrskega študijskega programa II. stopnje Strojništvo

Diko Purić

Mentor: doc. dr. Rok Vrabič, univ. dipl. inž.
Somentor: prof. dr. Peter Butala, univ. dipl. inž.

Ljubljana, februar 2018

Kandidat
Diko PURIĆ

MAGISTRSKI ŠTUDIJSKI PROGRAM II. STOPNJE: **MAG II/492**

NASLOV TEME: **Zmanjševanje nepopolnosti proizvodnih podatkov z metodo strojnega učenja**

V sodobnih proizvodnih sistemih se zbirajo podatki na vseh nivojih, od upravljanja do proizvodnih procesov. Podatki so na nivoju upravljanja proizvodnje (MES) pogosto vneseni ročno in so zato netočni ali nepopolni. Nepopolnost podatkov pa predstavlja problem za nadaljnje metode analitike in napovedovanja. Cilj magistrskega dela je dopolnitev manjkajočih podatkov o časih trajanja proizvodnih operacij z razvojem metode, ki temelji na strojnem učenju z nevronskimi mrežami, in preizkus razvite metode na realnih podatkih proizvodnje na naročilo.
V magistrskem delu preglejte področje obvladovanja nepopolnosti proizvodnih podatkov in preučite metode za njeno zmanjšanje. Opišite strukturo proizvodnih podatkov, pridobljenih iz proizvodnega podjetja, in identificirajte nepopolnosti. Opišite princip delovanja nevronskih mrež ter njihove značilnosti, parametre in hiperparametre. Na osnovi regresije z nevronskimi mrežami razvijte metodo dopolnjevanja manjkajočih vrednosti. Iz podatkov izberite značilke in argumentirajte njihov izbor. Ovrednotite in komentirajte izbiro parametrov ter hiperparametrov nevronske mreže. Prikažite uporabo metode na realnih podatkih in ovrednotite njeno uspešnost. Načrtajte smernice za uporabo razvite metode za napovedovanje trajanja še ne izvedenih operacij.

Magistrsko delo je treba oddati v jezikovno in terminološko pravilnem slovenskem jeziku. Rok za oddajo tega dela je šest mesecev od dneva prevzema.

Mentor

doc. dr. Rok Vrabič, univ. dipl. inž.

Somentor

prof. dr. Peter Butala, univ. dipl. inž.

Predsednik diplomske komisije

prof. dr. Sašo Medved, univ. dipl. inž.

Podpisani sem delo
prevzel v Ljubljani

dne 12.1.2018

Podpis .....

UNIVERZA V LJUBLJANI • FAKULTETA ZA STROJNIŠTVO •

Prodekan za pedagoško dejavnost
II. in III. stopnje

izr.prof.dr. Andrej Kitanovski, univ.dipl.inž

Zahvala

Zahvaljujem se mentorju doc. dr. Roku Vrabiču, da mi je omogočil obravnavanje želene tematike ter za pomoč in usmerjanje pri izdelavi magistrskega dela. Hvala tudi prof. dr. Petru Butali za podporo in priložnost izdelave magistrskega dela pod okriljem laboratorija LAKOS. Prav tako se zahvaljujem Dominiku Kozjeku za ideje in nasvete pri delu s podatki in strojnim učenjem. Na koncu še hvala družini in vsem ostalim, ki so me pri delu podpirali.

Spodaj podpisani/-a Diko Purić študent/-ka Fakultete za strojništvo Univerze v Ljubljani, z vpisno številko 23152146, avtor/-ica pisnega zaključnega dela študija z naslovom: Zmanjševanje nepopolnosti proizvodnih podatkov z metodo strojnega učenja,

IZJAVLJAM,

1.* (a) da je pisno zaključno delo študija rezultat mojega samostojnega dela;

b) da je pisno zaključno delo študija rezultat lastnega dela več kandidatov in izpolnjuje pogoje, ki jih Statut UL določa za skupna zaključna dela študija ter je v zahtevanem deležu rezultat mojega samostojnega dela;

2. da je tiskana oblika pisnega zaključnega dela študija istovetna elektronski obliki pisnega zaključnega dela študija;

3. da sem pridobil/-a vsa potrebna dovoljenja za uporabo podatkov in avtorskih del v pisnem zaključnem delu študija in jih v pisnem zaključnem delu študija jasno označil/-a;

4. da sem pri pripravi pisnega zaključnega dela študija ravnal/-a v skladu z etičnimi načeli in, kjer je to potrebno, za raziskavo pridobil/-a soglasje etične komisije;

5. da soglašam z uporabo elektronske oblike pisnega zaključnega dela študija za preverjanje podobnosti vsebine z drugimi deli s programsko opremo za preverjanje podobnosti vsebine, ki je povezana s študijskim informacijskim sistemom članice;

6. da na UL neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve avtorskega dela v elektronski obliki, pravico reproduciranja ter pravico dajanja pisnega zaključnega dela študija na voljo javnosti na svetovnem spletu preko Repozitorija UL;

7. da dovoljujem objavo svojih osebnih podatkov, ki so navedeni v pisnem zaključnem delu študija in tej izjavi, skupaj z objavo pisnega zaključnega dela študija;

8. da dovoljujem uporabo mojega rojstnega datuma v zapisu COBISS.

V Ljubljani, 3. 2. 2018 Podpis avtorja/-ice: D. Purić

* Obkrožite varianto a) ali b).

Izvleček

UDK 004.8:658.5(043.2)

Tek. štev.: MAG II/492

Zmanjševanje nepopolnosti proizvodnih podatkov z metodo strojnega učenja

Diko Purić

Ključne besede: nepopolni podatki
kakovost podatkov
proizvodni podatki
strojno učenje
nevronske mreže
pripisovalne metode

V modernih proizvodnih sistemih se zbirajo velike količine podatkov. Spremljanje vseh izvorov podatkov ter njihovo obvladovanje prek številnih informacijskih sistemov je postala zahtevna naloga, pri kateri prihaja do napak. Te lahko povzročajo nepopolnost in posledično manjšo kakovost podatkov. Manjkajoče vrednosti smo zapolnili z napovedmi metode strojnega učenja, imenovane nevronske mreže. Izkazale so se za učinkovit način iskanja vzorcev v podatkih. S stališča točnosti in vnašanja pristranskosti so pridobljeni rezultati precej boljši od najpogosteje uporabljenih pripisovalnih metod.

Abstract

UDC 004.8:658.5(043.2)

No.: MAG II/492

Reducing the incompleteness of manufacturing data with a machine learning method

Diko Purić

Key words: incomplete data
data quality
manufacturing data
machine learning
neural networks
imputation methods

In modern manufacturing systems, large amounts of data are being collected. Monitoring sources of data and managing that data through many information systems is a challenging task that results in errors. These reduce data quality, the source of which is often data incompleteness. We filled the missing values with the predictions of a machine learning method called neural network. They proved effective in search of data patterns. From a point of view of accuracy and inputting bias the results are better than those from other frequently used imputation methods.

Kazalo

Kazalo slik

Kazalo preglednic

Seznam uporabljenih simbolov

Oznaka	Enota	Pomen
a	/	prava vrednost
$\hat{a}$	/	napovedana vrednost
L	/	funkcija napake
n	/	število napovedi
t	/	netransformirano trajanje
t'	/	transformirano trajanje
w	/	utež
x	/	vhod v nevron
y	/	izhod iz nevrona
φ	/	aktivacijska funkcija
Indeksi		
i		indeks povezave
j		indeks napovedi

Seznam uporabljenih okrajšav

Okrajšava	Pomen
ADAM	prilagodljiv moment (angl. *Adaptive Moment*)
CAE	računalniško podprto inženirstvo (angl. *Computer Aided Engineering*)
CAM	računalniško podprta proizvodnja (angl. *Computer Aided Manufacturing*)
CAD	računalniško podprto načrtovanje (angl. *Computer Aided Design*)
CNC	računalniški numerični nazdor (angl. *Computer Numerical Control*)
CuDNN	cuda knjižnica globokih nevronskih mrež (angl. *Cuda Deep Neural Network library*)
ERP	poslovno transakcijski informacijski sistem (angl. *Enterprise Eesource Planning*)
HN	Huberjeva napaka
LAKOS	laboratorij za tehnično kibernetiko, obdelovalne sisteme in računalniško tehnologijo
MAR	naključno manjkanje (angl. *Missing At Random*)
MCAR	popolnoma naključno manjkanje (angl. *Missing Completely At Random*)
MES	proizvodni informacijski sistem (angl. *Manufacturing Execution System*)
MNAR	manjkanje ni naključno (angl. *Missing Not At Random*)
MTBF	povprečni čas med okvarami (angl. *Mean Time Between Failures*)
NaN	ni številka (angl. *Not A Number*)
PAN	povprečna absolutna napaka
PAON	povprečna absolutna odstotkovna napaka
PDM	upravljanje podatkov o izdelkih (angl. *Product Data Management*)
PID	proporcionalno integralni diferencialni krmilnik (angl. *Proportional Integral Derivative controller*)
PKLN	povprečna kvadrirana logaritemska napaka
PKN	povprečna kvadrirana napaka
PLC	programabilni logični krmilnik (angl. *Programmable Logic Controller*)
PLM	upravljanje življenjskega cikla izdelkov (angl. *Product Lifecycle Management*)
R2N	R2 napaka
RAM	delovni pomnilnik (angl. *Random Access Memory*)
RMSProp	korenjen povprečen kvadriran pogon (angl. *Root Mean Square Prop*)
SAP	spremljanje aktivnosti podjetja (angl. *Systems, Applications and Products in Data Processing*)
SCADA	informacijski sistem za nadzor in zbiranje podatkov (angl. *Supervisory Control And Data Acquisition*)
SGD	stohastični gradientni spust (angl. *Stochastic Gradient Descent*)
WBS	delovna struktura (angl. *Work Breakdown Structure*)

XOR	izključno ali (angl. *Exclusive OR*)

1 Uvod

1.1 Ozadje problema

V modernih proizvodnih sistemih se zbirajo velike količine podatkov. Preko informacijskih sistemov, kot so ERP, MES in SCADA, se shranjujejo v podatkovne baze. Taki podatki so lahko nekakovostni zaradi mnogih razlogov. Netočnost, neusklajenost, nepopolnost in nepravočasnost so glavni vzroki slabe kakovosti podatkov. Naš poudarek je na nepopolnosti podatkovnih nizov. Manjkajoče vrednosti v podatke vnašajo pristranskost, onemogočajo izvajanje različnih analiz ter povzročajo še mnogo drugih težav. Če iz podatkov z manjkajočimi vrednostmi na neki način pridemo do popolnih podatkov, to podjetju predstavlja veliko dodano vrednost. Take podatke je lažje analizirati in na njih izvajati različne metode strojnega učenja, kar podjetjem zagotavlja novo znanje o proizvodnem procesu.

Nepopolnost proizvodnih podatkov je lahko posledica nedokončanega dela, okvare opreme, napak pri ročnem beleženju, izgube podatkov in drugih vzrokov. V želji, da lahko podatke kar se da dobro uporabimo, moramo iz njih nepopolnost odstraniti. To lahko storimo na dva načina. Pri prvem načinu del podatkov, ki vsebuje manjkajoče vrednosti, iz podatkov odstranimo. Drugi način pa praznim celicam vrednosti pripiše. Izkaže se, da je drugi način s stališča nadaljnje analize ponavadi boljši. Metode, ki zaznavajo kompleksne vzorce v podatkih, lahko te prenesejo na nepopolni del podatkov in s tem v njih vnašajo minimalno pristranskost. Kljub temu se zaradi enostavnosti implementacije še vedno veliko uporablja prvi način. Nevronske mreže, ki na mnogih področjih dosegajo izjemno dobre rezultate, so metoda strojnega učenja, ki jo lahko uporabimo za prepoznavanje vzorcev. Preko številnih hiperparametrov, kot so aktivacijska funkcija, optimizacijski algoritem, funkcija napake, topologija mreže in drugih, lahko na delovanje nevronskih mrež bistveno vplivamo. To je koristno, saj pri njihovi izbiri neprestano sprejemamo odločitve o tem, kakšne želimo, da so pripisane vrednosti.

1.2 Cilji

Pogosto se nepopolnost podatkov zmanjšuje z zelo enostavnimi metodami, med katerimi je najbolj pogosta brisanje primerov z manjkajočimi vrednostmi. Naš cilj pa je te vrednosti

napovedati in z napovedmi zapolniti prazne celice v podatkih. To bomo naredili z metodo strojnega učenja, imenovano nevronske mreže. Atribut napovedovanja bo čas izvedbe operacije. S tem delom v podatke ne želimo vnašati pristranskosti. To lahko pomeni, da bomo zaradi morebitnih netočnih vhodnih podatkov morali namesto dejanskih napovedati netočne vrednosti. Da lahko dosežemo ta cilj, moramo najprej spoznati omenjene tematike. Pogledali si bomo nadzor proizvodnje in nevronske mreže, kjer želimo pridobiti poglobljeno znanje o njihovem delovanju in optimizaciji. Spoznati tudi želimo, kako se lotiti problema zmanjševanja nepopolnosti podatkov z metodo strojnega učenja, da bo delo bolje izvedeno.

1.3 Struktura naloge

Magistrska naloga je sestavljena iz 6 delov. V prvem delu so opisani kakovost, nepopolnost in izboljševanje podatkov ter razlaga konceptov in trajanj v proizvodnji, pomembnih za nadaljnjo razumevanje. V drugem delu so v prostor umeščene in opisane nevronske mreže, njihovo delovanje ter hiperparametri. Tretji del predstavlja potek dela na strojnem učenju. V četrtem delu sledi razlaga podatkovnih vrst, opis podatkov, s katerimi smo se v nadaljevanju ukvarjali, ter sistemov, ki jih upravljajo. Peti del predstavlja opis našega dela, strukturiran na način, definiran v tretjem delu. V šestem delu so zapisani in razloženi pridobljeni rezultati.

2 Izboljšava kakovosti podatkov o trajanju operacij

2.1 Kakovost podatkov

Pomembnost podatkov v današnjem svetu ni več vprašljiva. Ne le da podatki podjetjem zagotavljajo veliko konkurenčno prednost, mnogi na njih gledajo kot na kapital podjetja. Podatki pa niso dovolj, njihova uporabnost je v veliki meri odvisna od njihove kakovosti. Od kakovosti analiziranih podatkov je odvisna kakovost odločitev, sprejetih na vseh področjih podjetja. V proizvodnih podjetjih se podatki uporabljajo za povečevanje kakovosti izdelkov, analizo napak, optimizacijo in planiranje procesov ter mnoge druge namene [1].

Količina podatkov se povečuje, s tem pa se povečuje tudi težavnost zagotavljanja dobre kakovosti podatkov. Zbiranje, shranjevanje in preučevanje velikih količin podatkov je kompleksen proces, kjer nastopi mnogo priložnosti, da se njihova kakovost poslabša. Spremljanje procesa v podjetjih in prepoznavanje točk, kjer so potrebne izboljšave procesa, sta ključnega pomena za izboljšanje kakovosti podatkov. Izboljšav kakovosti se lahko lotimo z različnimi računskimi metodami, a je morda še bolj pomembno najti vzroke, zaradi katerih do manj kakovostnih podatkov prihaja in jih odpraviti [1].

Pri spremljanju kakovosti podatkov opazujemo več faktorjev. Spodaj so našteti štirje, čeprav jih nekateri avtorji navajajo še več. To so točnost, usklajenost, popolnost in pravočasnost podatkov [1].

Točnost podatkov govori o stopnji, s katero podatki predstavljajo realno stanje, ki ga opisujejo. Tu se je mogoče navezovati na točnost različnih stvari. Opazujemo lahko merilno opremo ter kakšne so izmerjene vrednosti v primerjavi z dejanskim stanjem. Lahko pa opazujemo človeško sposobnost za zagotavljanje pravih podatkov. Primer je ujemanje imen, podanih v anketah, z dejanskimi imeni (razlog za odstop je lahko napaka pri črkovanju, laž, sprememba imena itd.) [2].

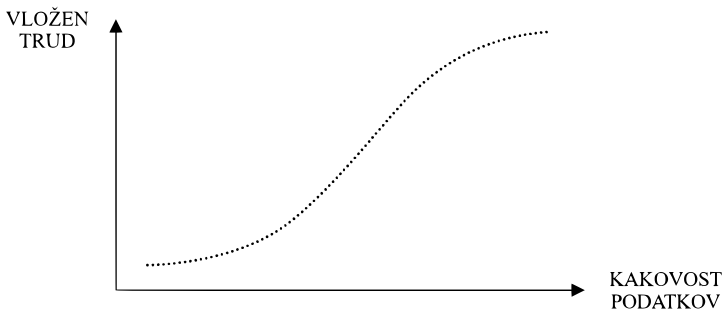
Usklajenost govori o tem, kako se podatki iz enega niza ujemajo s podatki iz drugih nizov. Vrednosti iz različnih nizov si ne smejo kontradiktirati in morajo opisovati isti fenomen. To pa še ne pomeni, da so vrednosti pravilne oziroma točne. Ni nujno, da usklajenost spremljamo na nivoju nizov podatkov. O usklajenosti lahko govorimo tudi pri primerjavi vrednosti znotraj niza podatkov ter usklajenosti v času. Primer usklajenih podatkov je, če se seštevek števila zaposlenih na posameznih lokacijah podjetja (zapisana v ločenih nizih podatkov) ujema s številom zaposlenih celotnega podjetja [2].

Popolnost podatkov govori o tem, ali imajo določeni atributi pripisane vrednosti. Nekateri atributi ne potrebujejo vrednosti, saj ta v določenih primerih ne obstaja. Tam prazna celica ni problematična. Primer takega atributa pa je lahko drugi priimek. Če ima oseba le en priimek, bo celica ostala prazna. Veliko atributov je takih, da pripisano vrednost morajo imeti in ne smejo vsebovati praznih celic. Primer takega atributa je naslov za dostavo pri podjetju, ki dostavlja pakete. Če naslov ni zapisan, paket ne more biti dostavljen. Pri takih kritičnih atributih je popolnost nujna [2].

Pravočasnost govori o tem, kdaj lahko do podatkov dostopamo. Merimo jo s pretečenim časom med tem, kdaj podatke potrebujemo in kdaj jih lahko uporabimo. Primer področja, kjer je to zelo pomembno, so finančni trgi. Tam se atributi, kot so cene vrednostnih papirjev, zelo hitro spreminjajo. Njihova vrednost pa je kritičnega pomena za odločitev o morebitnem nakupu [2].

Dodatni faktorji kakovosti podatkov, ki jih lahko zasledimo v literaturi, so skladnost, integriteta in edinstvenost.

Čeprav nam intuicija govori, da povečanje kakovosti podatkov vodi k izboljšanju odločitev, sprejetih na njihovi podlagi, ta vpliv za zdaj še ni najbolj proučen. Zelo pomembno bi bilo, da vodstvo v podjetju bolje razume kakovost podatkov in tako lažje vrednoti njen vpliv [1]. V delu Hezana et al. [3] ugotavljajo, da lahko slaba kakovost podatkov povzroča povečanje stroškov in tveganja ter zmanjšanje prihodkov in zaupanja. Odvisnost med kakovostjo podatkov ter njenimi pozitivnimi vplivi na podjetja pa ni linearna. Slika 2.1 prikazuje odvisnost med kakovostjo podatkov in trudom, potrebnim za njihovo izboljšavo. Kot nam govori ekonomska teorija o zmanjševanju donosnosti, je treba najti ravnovesje med kakovostjo podatkov in stroški, ki nam jih delo, povezano z njihovo izboljšavo, povzroči. Obstaja torej točka, do katere se podatke splača izboljševati, in točka, od katere naprej to ni več smiselno [1].



Slika 2.1: Odvisnost med kakovostjo podatkov in vloženim trudom [1]

2.2 Nepopolni podatki

Manjkajoče vrednosti lahko najdemo v večini nizov podatkov. Nekatere celice v podatkih nimajo predpisane vrednosti in so prazne. To predstavlja veliko težavo iz več razlogov. Manjkajoče vrednosti lahko v niz podatkov vnesejo pristranskost, mu zmanjšajo statistično moč, hkrati pa na takem nizu podatkov ne moremo izvajati mnogih metod strojnega učenja [3]. Še posebej so lahko škodljive, če porazdelitev manjkajočih vrednosti ni enakomerna in ne poznamo mehanizma, ki bi jih razložil [4]. Za najboljše rezultate mora biti niz podatkov brez manjkajočih vrednosti. Takim podatkom pravimo popolni podatki.

V literaturi se o nepopolnih podatkih piše že desetletja, vendar pa se je moderni pristop začel leta 1976 z Rubinovim teoretičnim okvirjem za reševanje težave manjkajočih vrednosti. V zadnjih 20. letih se je število raziskav na področju obravnave nepopolnih podatkov precej povečalo. Razvite so metode za delo z manjkajočimi vrednostmi, vendar njihova uporaba v realnih aplikacijah še ni najbolj razširjena in porazdeljena po različnih disciplinah, ki delajo s podatki [5].

Najpogosteje opažena težava nepopolnih podatkov je ta, da na njih mnogih vrst analiz ne moremo opraviti, saj taki podatki povzročajo napake. Na njih mnogo algoritmov preprosto ne deluje. To sicer ne velja za vse algoritme, vsekakor pa smo pri izbiri algoritmov, odpornih proti manjkajočim vrednostim, precej omejeni. Primer takih algoritmov so klasifikacijska in regresijska drevesa ter metoda najbližjih sosedov [6].

2.2.1 Nastanek

Razlogov za nastanek nepopolnih podatkov je mnogo. Težava ni prisotna le v proizvodnem inženirstvu, ampak v praktično vseh disciplinah, kjer se ukvarjajo s podatki. Raziskave kažejo, da je pri disciplinah, kot sta ekonomija in medicina, med 40 in 80 % primerov takih, kjer imajo raziskovalci opravka z nepopolnimi podatki [4].

Dejstvo, da so podatki lahko nepopolni, velja za vse discipline. Razlogi, da do tega prihaja, pa so za posamezne discipline specifični. Ena od možnosti za nastanek nepopolnih podatkov je, da del podatkov nikoli ni obstajal. Sem spadajo na primer neodgovorjena vprašanja v anketah, nedokončano delo in ne vnašanje podatkov zaposlenih, prava možnost ni bila na izbiro, odpoved opreme, ki beleži podatke itd. Mogoče je tudi to, da so podatki obstajali, a jih v obravnavanem nizu ni več. Primeri so lahko izguba podatkov ali pa nameren izbris. Slednje se lahko zgodi zaradi omejitev glede razkritja dela podatkov, vrednosti so bile narobe vnesene, podatki niso bili dovolj natančni, vsebovali so preveč šuma in podobno. Prepoznavanje vzrokov nastanka nepopolnih podatkov je ključno za izbiro metode za njihovo odpravljanje [3,7].

2.2.2 Vrste

Vrsta nepopolnih podatkov lahko močno vpliva na kakovost rezultatov različnih algoritmov na teh podatkih. Načinov razvrščanja je več, vendar jih največkrat razdelimo med tri kategorije. To so:
- popolnoma naključno manjkanje (angl. *Missing Completely At Random*) oziroma kratica MCAR,
- naključno manjkanje (angl. *Missing At Random*) oziroma kratica MAR,
- manjkanje ni naključno (angl. *Missing Not At Random*) oziroma kratica MNAR.

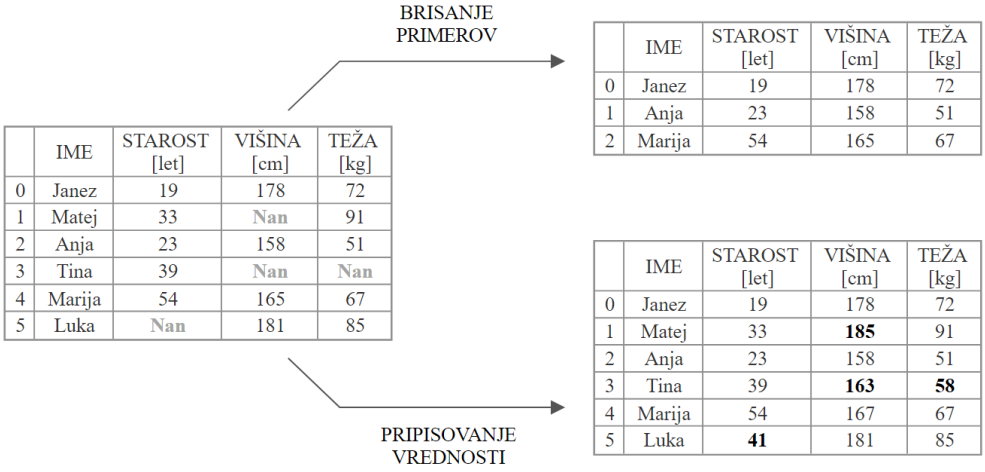
O **MCAR** govorimo, kadar se manjkajoče vrednosti pojavljajo naključno in ne moremo prepoznati vzorca, ki mu sledijo. Verjetnost, da je določena vrednost manjkajoča, je torej neodvisna od same sebe in od ostalih vrednosti v podatkih. Vpliv MCAR na rezultate analiz, narejenih na teh podatkih, je v veliki meri odvisen od porazdelitve manjkajočih vrednosti po podatkih. Bolj kot je porazdelitev enakomerna, manj pristranskosti vnese v podatke. Razlogi za manjkajoče vrednosti so lahko na primer: ročno vnašanje podatkov, odpoved opreme in izguba podatkov pri prenašanju. Primer je lahko manjkajoča telefonska številka osebe, izgubljen ali uničen vzorec za analizo ipd. [3,7].

O **MAR** govorimo, kadar lahko prepoznamo vzorec, ki mu manjkajoče vrednosti sledijo. Verjetnost, da je določena vrednost manjkajoča, je odvisna od preostalih vrednosti v podatkih, ni pa odvisna sama od sebe. Primer so lahko vrednosti, ki manjkajo, kadar manjka tudi neka druga vrednost ali pa neka druga vrednost zavzame na primer ekstremno vrednost. Prav tako o tej vrsti govorimo, ko manjka vrednost, kot je višina dohodka osebe. To vrednost je mogoče oceniti na podlagi drugih spremenljivk, kot so poklic, starost in zaposlovalec [3,7].

O **MNAR** govorimo, kadar ne moremo prepoznati vzorca, kateremu manjkajoče vrednosti sledijo. Razloga za to pa sta dva. Verjetnost, da je določena vrednost manjkajoča, je odvisna od same manjkajoče vrednosti ali pa od vrednosti, ki je nismo opazovali. Da je odvisna sama od sebe, se lahko zgodi, ko nastopi vrednost izven definiranega intervala. Težava, da je odvisna od manjkajoče vrednosti, pa je ta, da tistega atributa nismo opazovali. Ta vrsta je precej problematična, saj jo hitro zamenjamo z MCAR. Vzorca ne najdemo, čeprav bi bilo to mogoče kot pri MAR [3,7].

2.2.3 Metode odpravljanja

Nepopolni podatki so nezaželeni, zato se želimo manjkajočih vrednosti znebiti. To lahko naredimo na dva načina. Prvi način je brisanje primerov, kjer nastopajo manjkajoče vrednosti. Pri drugem načinu pa praznim celicam vrednost pripišemo. Oba načina prikazuje slika 2.2.



	IME	STAROST [let]	VIŠINA [cm]	TEŽA [kg]
0	Janez	19	178	72
1	Matej	33	Nan	91
2	Anja	23	158	51
3	Tina	39	Nan	Nan
4	Marija	54	165	67
5	Luka	Nan	181	85

	IME	STAROST [let]	VIŠINA [cm]	TEŽA [kg]
0	Janez	19	178	72
1	Anja	23	158	51
2	Marija	54	165	67

	IME	STAROST [let]	VIŠINA [cm]	TEŽA [kg]
0	Janez	19	178	72
1	Matej	33	**185**	91
2	Anja	23	158	51
3	Tina	39	**163**	**58**
4	Marija	54	167	67
5	Luka	**41**	181	85

Slika 2.2: Primer nepopolnih podatkov in metode njihovega odpravljanja

Brisanje

Najpreprostejša strategija za odpravljanje manjkajočih vrednosti je, da primere, kjer nastopi manjkajoča vrednost, preprosto izbrišemo. Če pogledamo sliko 2.2, bi to pomenilo, da bi vrstice, v katerih nastopa NaN (oziroma manjkajoča vrednost), izbrisali. Rezultat takega dela je prikazan na sliki 2.2 desno zgoraj. Taka strategija je pogosto uporabljena, vendar ni vedno najboljša izbira. Primerna je za primere, ko je moč množice (angl. *cardinality*), iz katere brišemo, relativno majhna in ko so manjkajoče vrednosti homogeno porazdeljene. Če so manjkajoče vrednosti zbrane v eni spremenljivki ali pa bi njihovo brisanje povzročilo izgubo velike količine podatkov, pa ta strategija ni najboljša [7].

Pripisovanje vrednosti

Metodam pripisovanja vrednosti praznim celicam pravimo tudi pripisovalne metode. Te na podlagi obstoječih podatkov izračunajo vrednosti, ki jih nato vpišemo v prazne celice. Pri tem je treba poudariti, da jim je nemogoče pripisati popolnoma nevtralne vrednosti, ki bi zagotavljale enako zanesljivost, kot če manjkajočih vrednosti ne bi bilo. Verjetno bomo spremenili karakteristike niza podatkov, kot je na primer povprečna vrednost določene spremenljivke ali pa njena varianca [6]. Pomembno je, da poznamo razlog za manjkajoče vrednosti in njihovo vrsto, saj v nasprotnem primeru lahko pride do velike pristranskosti pri dopolnjevanju. Izbira pripisovalne metode ima torej velik vpliv na rezultate modelov in analiz, narejenih na popolnih podatkih [7].

Pripisovalnih metod je veliko, med seboj pa se močno razlikujejo. Med najbolj enostavnimi in najpogosteje uporabljenimi so pripisovanje povprečne vrednosti, pripisovanje mediane in pripisovanje najbolj pogoste vrednosti. Posebne razlage te metode ne potrebujejo. Za primer vzemimo pripisovanje povprečne vrednosti. Iz podatkov vzamemo izbrano spremenljivko in iz vrednosti, ki za to spremenljivko obstajajo, izračunamo povprečje. To povprečje nato pripišemo vsem praznim celicam za to spremenljivko. S tem ohranimo enako povprečno vrednost izbrane spremenljivke, zmanjšamo pa njeno varianco. Taka metoda je lahko seveda močno problematična, če porazdelitev izbrane spremenljivke ni normalna. Včasih se praznim celicam pripiše neka

konstantna vrednost, ki se loči od ostalih vrednosti za izbrano spremenljivko. Taka vrednost je lahko na primer 0. Zgornje metode so dokaj preproste za implementacijo in računsko precej nezahtevne, obstajajo pa tudi metode, ki dosegajo večjo natančnost in omogočajo večjo fleksibilnost. Ena izmed njih je Hot Deck pripisovanje, ki računa razdalje med vrednostmi spremenljivk posameznih primerov. Prazni celici opazovanega primera nato pripiše vrednost te spremenljivke za primer, ki mu je najbližje. Prav tako pogosta je metoda več pripisovanj (angl. *multiple imputations*). Ta niz podatkov razmnoži na več enakih kopij, nato pa vsem nizom podatkov zapolni prazne celice. Metode pripisovanja vrednosti tem praznim celicam so lahko različne ali pa enake. Na koncu pripisane vrednosti iz različnih nizov med seboj primerjamo in izračunamo končno vrednost. Uporabljene metode so še iterativno pripisovanje, pripisovanje z oceno največje verjetnosti, pripisovanje z nevronskimi mrežami in še mnoge druge [6-8].

Omenili smo že, da je izbira pripisovalne metode odvisna od uporabljenih podatkov. Dobro poznavanje podatkov je zato ključnega pomena. Kljub temu pa so Garciarena et al. [7] ugotovili, da se v splošnem bolj kompleksne metode pripisovanja (metoda večih pripisovanj in Hot Deck pripisovanje) izkažejo za boljše kot najbolj osnovne metode (pripisovanje povprečne vrednosti, mediane in najbolj pogoste vrednosti). Nevronske mreže so se izkazale za zelo učinkovite, še posebej, kadar je delež manjkajočih vrednosti velik, in ko manjkajoče vrednosti pripadajo več spremenljivkam [3].

2.3 Nadzor in vodenje proizvodnje

Razumevanje podatkov, s katerimi imamo opravka, je zelo pomembno. Zato moramo najprej spoznati nadzor in vodenje proizvodnje, pri čemer imamo več nalog. To so zajemanje podatkov o opravljenosti, ukrepanje v primeru odstopanj, zaključevanje delovnih nalogov in upravljanje materiala. Da bi vse skupaj lažje razumeli, bomo v nadaljevanju najprej opisali delovni nalog, operacije in pretočne čase, nato pa bomo natančneje opredelili zajemanje podatkov o opravljenosti [9].

2.3.1 Delovni nalog

Delovni nalog je dokument, ki v proizvodnem podjetju naroča, kdaj naj bodo določeni izdelki, sklopi ali sestavni deli izdelani. Nanj se knjižijo vsi stroški, povezani z izdelavo in montažo izdelkov, sklopov oziroma sestavnih delov. Vsebuje splošne informacije o samem delovnem nalogu, o operacijah za izvedbo in uporabljenih materialih.

Informacije o delovnem nalogu vsebujejo edinstveno številko delovnega naloga, vrsto, status in pomembnost delovnega naloga. Prav tako so tam zapisane informacije o naročniku, količini, stroškovnem mestu in načrtovanih časih, terminih in stroških. Vsaka operacija je v delovnem nalogu zapisana v svojo vrstico. Vsebuje unikatno številko operacije, opis, delovno mesto izvedbe, delavca, orodje ter mesto predaje obdelovanca. Podatki o materialih so zapisani za vsako komponento v novo vrstico. Tam najdemo identifikacijo komponente, informacije o količinah, termin izdaje in delovno mesto. Del delovnega naloga pa so tudi operacijski nalogi, ki govorijo o posameznih operacijah za

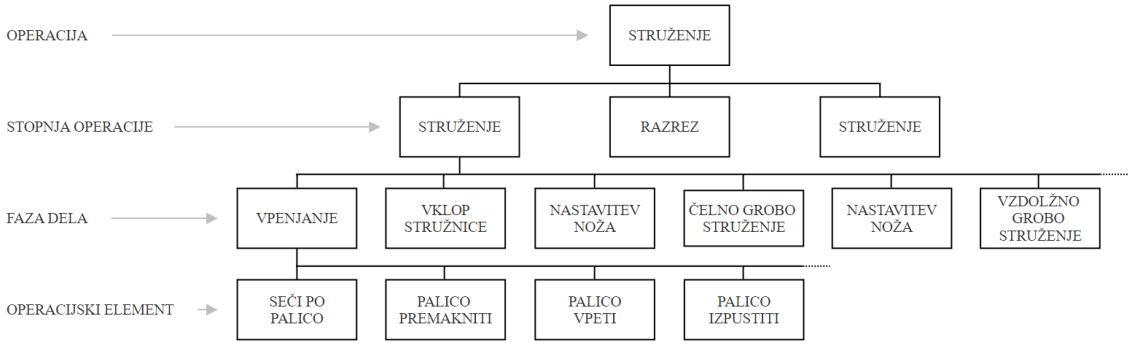
izdelavo izdelka, o posamezni sestavi oziroma o sestavnem delu. Delovni nalog se zaključi, ko je izvedena zadnja operacija in so vsi ustrezni izdelki predani v skladišče [9].

Proizvodno podjetje ima navadno opravka z več naročili naenkrat, kar pomeni, da obstaja tudi več delovnih nalogov. Ti so lahko v proizvodnji prosti ali pa povezani. Pri prostih delovnih nalogih med njimi ni povezav. Rezultati teh nalogov se skladiščijo v končno skladišče ali pa v vmesno skladišče, in sicer neodvisno od ostalih nalogov. Kadar pa so nalogi povezani, se njihovi rezultati večinoma vgrajujejo v nadrejene naloge in le redko v vmesna skladišča. Razlika med tema vrstama nalogov je tudi v načinu razvrščanja. Prosti so razvrščeni le glede na zahtevane roke, medtem ko so povezani odvisni tudi od povezav med njimi [9].

2.3.2 Operacije

V nadaljevanju bodo nekoliko bolj podrobno predstavljene operacije. Gre za delo, ki ga opravi delavec (oziroma skupina delavcev) na nekem delovnem mestu z nekim delovnim pripomočkom. Primer je lahko struženje delavca v strugarni s pomočjo stružnice.

Poznamo več vrst operacij, in sicer tehnološke, transportne, kontrolne, skladiščne operacije ter zastoje. O tehnoloških operacijah govorimo, kadar predmete dela sestavljamo, razstavljamo ali pa jim spreminjamo fizikalne oziroma kemične lastnosti. Transportne so tiste, kjer predmete dela premikamo med delovnimi mesti (premik znotraj delovnega mesta ni transportna operacija). Pri kontrolnih ugotavljamo kakovost in količino predmetov dela. Skladiščne operacije služijo varovanemu shranjevanju predmetov dela. O zastojih govorimo takrat, ko obdelovanec čaka na nezavarovanem mestu zaradi zasedenosti delovnega sistema, h kateremu je namenjen.



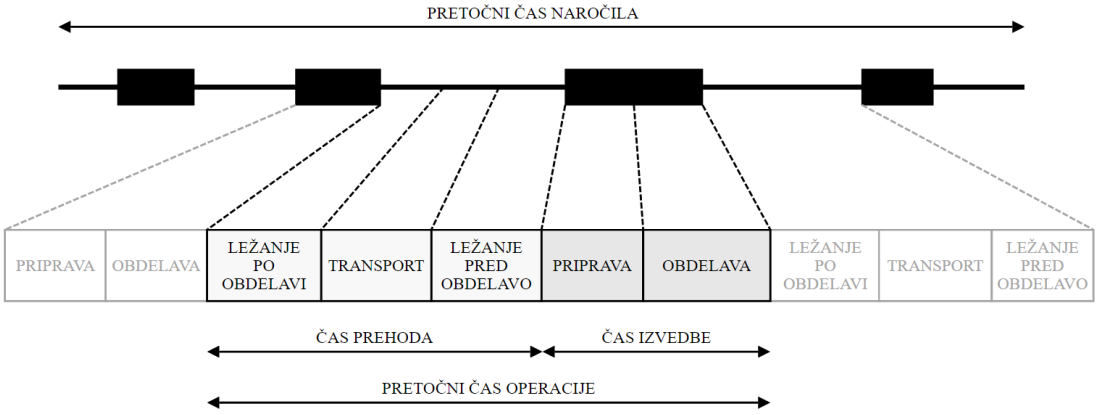
Slika 2.3 Delitev operacije [9]

Podobno kot je delovni nalog razčlenjen na več operacij, lahko tudi operacije razdelimo še naprej. Operacije naprej delimo na stopnje operacij, te na faze dela, na koncu pa še na operacijske elemente, kar prikazuje slika 2.3. Stopnja operacije je del operacije, pri katerem izvedba poteka na določenem delovnem mestu, z nekim delovnim sredstvom in pri nekem vpetju. Faza dela je del stopnje operacije in ga definira zaporedje operacijskih

elementov. Operacijski element je del faze dela, ki ga naprej ni mogoče deliti (manjših delov ne moremo opisati in jim pripisati časa izvajanja) [9].

2.3.3 Pretočni čas

Poznamo dve vrsti pretočnega časa. Prvi je pretočni čas naročila, ki predstavlja trajanje od izdaje prvega materiala iz skladišča pa do skladiščenja končnega izdelka. Drugi pa je pretočni čas operacije, ki predstavlja najmanjšo enoto pretočnega časa naročila. Kot prikazuje slika 2.4, ga sestavljajo čas ležanja po izvedbi predhodne operacije, čas transporta, čas ležanja pred izvedbo opazovane operacije, čas priprave in čas obdelave [9].



Slika 2.4 Struktura pretočnega časa naročila [9]

2.3.4 Zajemanje podatkov o opravljenosti

V proizvodnji zajemanje podatkov poteka tako, da opravljeni akciji sledi beleženje tega dosežka na neki delovni dokument. Poglejmo si nekaj primerov transakcij, ki se beležijo takoj po dogodku. Pri delu z materialom so to lahko: material, izdan iz skladišča, začetek transporta, končan transport. Pri delu na delovnem mestu poznamo: orodje, prevzeto iz skladišča, začetek priprave delovnega mesta, konec priprave delovnega mesta, začetek obdelave, konec obdelave, štetje dobrih kosov, predaja dobrih kosov v skladišče, orodje, vrnjeno v skladišče. Delavcu pa lahko sledimo pri: času prihoda na delo, odhoda iz dela, odhoda na odmor, prihoda iz odmora.

Podatke lahko zajemamo ročno ali avtomatsko. Pri ročnem zajemanju podatke zavedemo na papirno dokumentacijo. Nato jo kopiramo in razvrstimo po oddelkih, ki potrebujejo zbrane informacije. Ročno beleženje proizvodnim podjetjem predstavlja več težav. Gre za počasen proces, kjer pogosto prihaja do napak. Napake so lahko v obliki nezapisanih vrednosti, narobe zapisanih vrednosti ali »le« zakasnitve. Slednje povzročajo neažurnost informacij, kar zmanjšuje njihovo vrednost. V modernih sistemih se zato uporablja

avtomatsko zajemanje. Prvo avtomatsko zajemanje je bilo v obliki mehanskih števcev, registrskih ur in podobno. Danes pa se uporablja predvsem računalniška tehnika, umeščena v samo okolje. To so raznovrstni terminali, elektronski merilni pretvorniki, paneli, aktivni računalniki in mnogi drugi [9].

3 Nevronske mreže za napovedovanje

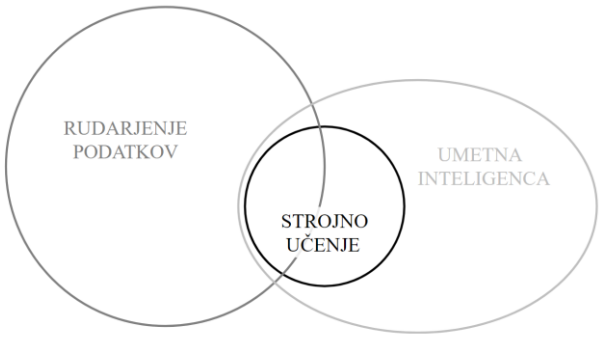
Podatki in učenje
Proizvodni procesi v današnjih podjetjih so vedno bolj kompleksni. Obsežnost sistemov, zapletene omejitve, negotovo okolje, zahteve po prilagodljivosti, hitrosti in točnosti to le še povečujejo. Posledično so vedno bolj kompleksni tudi podatkovni sistemi. Veliko tradicionalnih metod za obdelavo in analizo podatkov preprosto ni več primernih. Pri sodobnem delu s podatki se pogosto srečujemo z izrazi, kot so rudarjenje podatkov, umetna inteligenca in strojno učenje, ki se v proizvodnih sistemih vedno več uporabljajo [10]. Omenjeni koncepti nimajo ostrih mej in se med seboj precej prepletajo, zato jih je pogosto težko definirati.

Rudarjenje podatkov
Rudarjenje podatkov je multidisciplinarno področje, ki obsega vse od statistike, baz podatkov, umetne inteligence, strojnega učenja, teorije informacij, vizualizacije podatkov in še več. Cilj podatkovnega rudarjenja je pridobitev implicitnih, prej neznanih in potencialno uporabnih vzorcev in zakonitosti v podatkih [11]. Primeri pogosto uporabljenih tehnik rudarjenja podatkov so klasifikacija, gručenje, asociacija in analiza časovne vrste. Prav tako uporabljene metode pri rudarjenju podatkov so tudi metode strojnega učenja, ki spadajo v področje umetne inteligence [12].

Umetna inteligenca
Raziskovanje umetne inteligence se ukvarja s področjem, kjer naprava izkazuje kognitivne sposobnosti. Za umetno inteligenco označujemo dejavnosti, kjer naprava posnema inteligentna vedenja, ki jih običajno kažemo ljudje. Navdih za tako obnašanje pa lahko izhaja iz področij, kot so računalništvo, matematika ali statistika [13]. Pristopa k umetni inteligenci sta dva. Simbolični oziroma konvencionalni sledi logičnim operacijam in pravilom, primer česar so Bayesove mreže. Sub-simbolni pristop oziroma računska inteligenca pa sledi biološkim mehanizmom in uporablja način od spodaj navzgor. Primeri tega pristopa so nevronske mreže, genetski algoritmi in mehki sistemi [12].

Strojno učenje
Strojno učenje je podzvrst umetne inteligence. Njegova glavna značilnost je, da napravo naučimo, kako se učiti in je ne programiramo za posamične naloge. Taki algoritmi se na podatkih učijo in na njih delajo napovedi. Strojno učenje delimo na tri kategorije. Pri nadzorovanem učenju napravi predstavimo vhodne podatke in definiramo želen izhod. Cilj takega učenja je, da se na podlagi učnih primerov algoritem nauči in je sposoben napovedovati izide za še ne poznane primere. O nenadzorovanem učenju govorimo, kadar napravi predstavimo le vhodne podatke, ta pa mora brez človeškega nadzora prepoznati strukture v podatkih. Pri vzpodbujevalnem učenju (včasih mu pravimo okrepitveno učenje) se naprava obnaša kot agenti v interakciji z okoljem. Na podlagi teh iteracij se uči, kakšno obnašanje prinaša želene nagrade [13].

Na sliki 3.1 vidimo shemo, ki prikazuje povezanost in prekrivanje zgoraj opisanih terminov. Strojno učenje bomo pri »klasičnem« načinu prepoznavanja vzorcev uporabili le posredno. Teh vzorcev namreč ne bomo uporabili za odločanje, temveč za izboljšanje kakovosti podatkov samih.



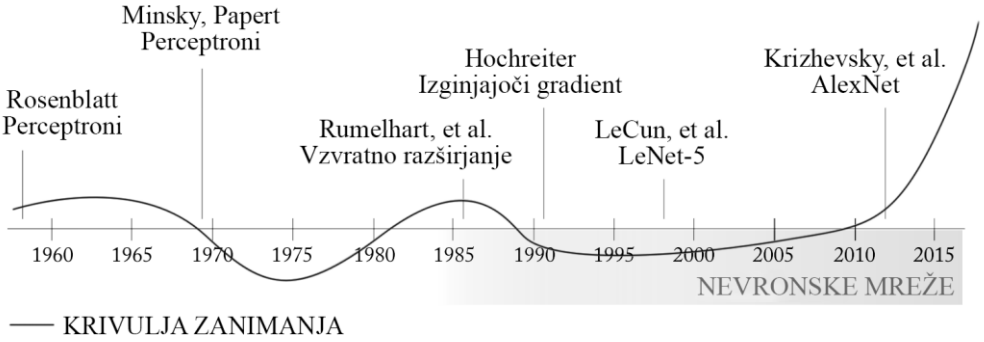
Slika 3.1: Shematski prikaz področij, povezanih s podatki in učenjem

3.1 Nevronske mreže

Metodo strojnega učenja, imenovano nevronske mreže, si lahko predstavljamo kot umetno izdelano predstavitev možganov. Mrežo sestavljajo številni majhni, med seboj povezani elementi, ki omogočajo kompleksno vedenje. Gradnja nevronske mreže po delčkih s strani človeka presega naše sposobnosti, zato se to počne s posrednimi metodami [14].

Izjemno priljubljenost nevronskih mrež v zadnjih letih pripisujemo trem dejavnikom. To so razpoložljivost velikih količin podatkov, povečanje računalniške moči in njene dostopnosti ter znanstveni napredek na področju učinkovitosti numeričnega preračunavanja. Take mreže na številnih področjih po uspešnosti ne presegajo le klasičnih metod, ampak tudi človeka samega. Strokovno znanje tega področja se je zaradi uspešnosti hitro premaknilo iz akademskih krogov tudi v tehnološka podjetja, kjer je postalo pomemben del sodobne programske opreme [13].

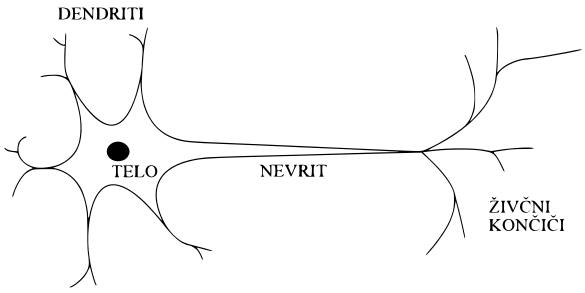
Presenetljivo je, kako daleč v preteklost segajo začetki raziskovanja nevronskih mrež. Uspešnost, ki jo dosegajo danes, je plod desetletij raziskovanja. Prvi poskusi posnemanja možganov segajo v 50. leta prejšnjega stoletja, ko je bila prvič predstavljena ideja umetnega nevrona, imenovana »perceptron«. Med raziskovalci pa se raziskovanje na tem področju ni preveč uveljavilo, predvsem zaradi tehničnih omejitev enega perceptrona. Ugotovili so namreč, da se en perceptron ni zmožen naučiti niti osnovne logične funkcije, kot je XOR. Nov val raziskovanja se je začel šele v 80. letih, ko so bile predstavljene večje in bolj zapletene hierarhične mreže, nove tehnike nastavljanja in paralelna distribuirana obdelava. Tudi ta pa je usahnil, saj je bila računska moč takratne opreme za reševanje realnih problemov še premajhna. V naslednjih desetletjih pa je področje računalništva doživelo velike spremembe. Naraščanje računalniške moči je sledilo Moorovemu zakonu in težave, ki so se včasih zdele nepremostljive, so naenkrat postale rešljive. Nekateri avtorji omenjajo celo nastanek samo-ojačitvene zanke med podatki, algoritmi in računsko močjo. Ko napreduje eno od treh področij, napredujeta tudi drugi dve. Slika 3.2 prikazuje prej opisane valove in pa avtorje prispevkov, ki so najbolj vplivali na to dogajanje [14].



Slika 3.2: Nevronske mreže v preteklosti [14]

Navdih za metodo strojnega učenja, imenovano nevronske mreže, izhaja iz naravnih nevronskih mrež, ki jih najdemo v možganih. Možgani so ogromna mreža med seboj povezanih preprostih gradnikov, imenovanih nevroni. Misli, spomin in percepcija so produkt velikega števila interakcij med temi elementi mreže [14].

Temeljna enota nevronskih mrež je nevron. To je celica z nalogo pošiljanja elektrokemičnih signalov drugim delom živčnega sistema, katerega del so tudi drugi nevroni. Nevron v grobem sestavljajo telo oziroma soma, nevrint oziroma akson, dendriti in živčni končiči (slika 3.3). Telo vsebuje jedro, metabolični mehanizem in druge organele, ki jih najdemo v celicah. Dentriti sprejemajo signale drugih nevronov in se obnašajo kot nekakšen vhod v nevron. Nevrit pa je zadolžen za oddajo signalov preko živčnih končičev nevrona naprej po živčevju [14,15].



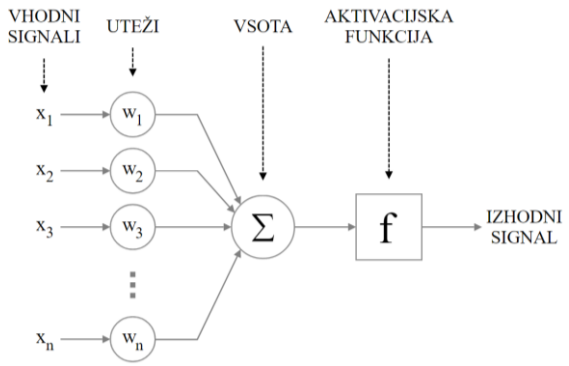
Slika 3.3: Nevron [15]

Komunikacija med nevroni je električni fenomen. Razlika električnih potencialov na nasprotnih straneh membran posameznih nevronov je posledica ionov. Nevron vse signale, ki jih sprejema iz drugih nevronov, akumulira. Ko nabrana razlika potencialov preseže določen prag, se ta v nevronu lahko še ojača ter prenese na drugo stran, kjer je nevron v interakciji z naslednjimi nevroni [14].

Delovanje umetnih nevronskih mrež ni enako, a ima veliko skupnega z naravnimi nevronskimi mrežami. Temeljijo na matematičnih operacijah, ki način delovanja naravnih nevronskih mrež poenostavljajo in prilagodijo obdelavi na modernih mikroprocesorjih. Poudariti je potrebno, da tu govorimo o računalniških modelih, navdihnjenjih s strani nevronov za reševanje arbitrarnih problemov. Obstajajo modeli, ki replicirajo naravne nevrone, namen katerih je razlaga in razumevanje delovanja naravnih nevronov. V tem magistrskem delu se ukvarjamo s prvo kategorijo [14].

3.1.1 Sestava mreže

Umetnemu nevronu, ki ga prikazuje slika 3.4, pravimo tudi perceptron. Njegova sestava je v osnovi podobna naravnemu nevronu. Ima eno ali več vhodnih povezav z nalogo zbiranja numeričnih signalov iz predhodnih nevronov. Vsaki povezavi je dodeljena utež. Na drugi strani je še ena ali več izhodnih povezav, ki peljejo izhodni signal do naslednjih nevronov. Vmes so signali sešteti in pomnoženi z aktivacijsko funkcijo. Ta določa prag, pri katerem pride do aktivacije ter do intervala vrednosti izhodnega signala [15].



Slika 3.4: Umetni nevron [15]

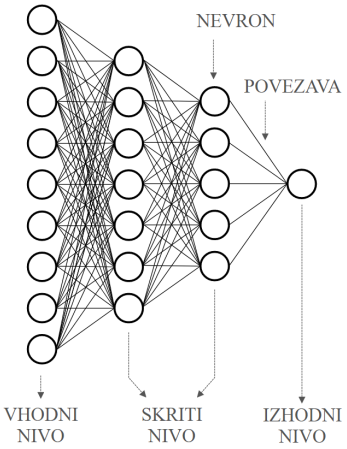
Umetni nevron lahko v matematičnem smislu zapišemo, kot kaže enačba (3.1). V enačbi vidimo, da gre le za seštevek obteženih vhodov, pomnožen z nelinearno aktivacijsko funkcijo. Uteži predstavlja oznaka w_i, vhode v nevron označuje x_i, izhod je y, aktivacijska funkcija pa φ.

$$y = \varphi\left(\sum_i w_i\, x_i\right) \tag{3.1}$$

Enostavnost nevrona kaže na enega ključnih konceptov nevronskih mrež. Njihova sposobnost popisovanja kompleksnega obnašanja ni posledica zapletenega nevrona, temveč skupka obnašanj enostavnih elementov [14].

Spodaj je predstavljena osnovna in ena najpopularnejših oblik nevronske mreže, ki ji pravimo večnivojski perceptron ali usmerjena nevronska mreža. Primer take mreže je na sliki 3.5. Mreža je iz vzporednih nivojev nevronov sestavljena tako, da veljajo naslednji pogoji:
- vsak nevron je povezan z vsemi na naslednjem nivoju,
- med nevroni na istem nivoju ni povezav,
- med nesosednjimi nivoji ni povezav.

Število nivojev ter nevronov v posameznem nivoju je odvisno od problema, ki ga rešujemo. Nevroni na levi predstavljajo vhode za nevrone na njihovi desni. Številu nivojev mreže pogosto pravimo globina, številu nevronov v nivoju pa širina. Nivo v mreži predstavljajo uteži vhodnih signalov ter aktivacijska funkcija pred izhodom. Na sliki 3.5 torej vidimo tronivojsko mrežo. Čeprav se uporablja izraz vhodni nivo, tam dejansko ne gre za nivo, saj nima uteži in aktivacijske funkcije. Vhodni nivo torej ne prispeva h globini mreže. Definira ga število značilk vhodnih podatkov. Izhodni nivo na desni pa dodaja h globini mreže. Število nevronov na njem je odvisno od obravnavanega primera. Nivoji v sredini ali skriti nivoji so poljubno izbrani. Izbira močno vpliva na kompleksnost mreže in kompleksnost obnašanja, ki ga taka mreža lahko popisuje. Vizualne predstavitve, kot je na sliki 3.5, so koristne za lažje razumevanje, vseeno pa lahko model reduciramo do matematične oblike [14].



Slika 3.5: Shematska predstava nevronske mreže

3.1.2 Učenje

Nevronske mreže lahko uporabljamo za nadzorovano ali nenadzorovano učenje. V tej magistrski nalogi bo poudarek na nadzorovanem učenju, kjer nevronsko mrežo uporabljamo za napovedovanje določene spremenljivke. Mreži zagotovimo vhodne podatke, ta pa na izhodu napove vrednost te spremenljivke. Pred tem moramo mrežo naučiti. Učenje mreže poteka na učnih podatkih. To so podatki, v katerih so vrednosti tako vhodov kot tudi izhodov iz mreže. Uteži se na podlagi teh podatkov prilagodijo tako, da mreža čim bolje popiše odvisnost med vhodnimi in izhodnimi podatki.

Mehanizmu, ki to izvaja, pravimo vzvratni postopek učenja. Ponavadi so na začetku utežem mreže pripisane naključne vrednosti. Mreži predstavimo vhodne podatke, ta pa na podlagi pripisanih uteži izračuna izhodno vrednost. Ker pa iz učnih podatkov poznamo pravo izhodno vrednost, ki bi se morala pojaviti na izhodu, lahko ti dve vrednosti primerjamo. Funkciji, ki to počne, pravimo funkcija napake. Ta napaka je nato prenesena v vzvratni smeri na vhod mreže. Tam se na podlagi njene vrednosti s pomočjo optimizacijskega algoritma uteži prilagodijo na tak način, da bo napaka pri naslednjem prehodu čim manjša. Proces se ponavlja toliko časa, dokler z napovedovanjem mreže nismo zadovoljni [13,15].

3.1.3 Generalizacija

Z učenjem se mreža nauči aproksimirati izhodne vrednosti, zagotovljene v učnem nizu. Namen mreže pa ni aproksimirati te vrednosti, temveč vrednosti, ki jih v učnem nizu ni. Model, ki dobro generalizira, bo dobro deloval tako na učnih podatkih kot na novih podatkih. Obstajajo trije pogoji, ki morajo biti izpolnjeni za dobro generalizacijo:

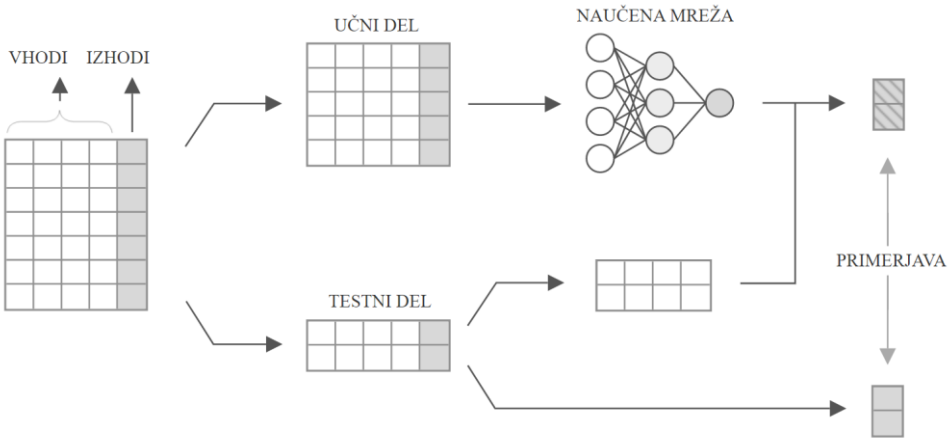
- Učni podatki vsebujejo dovolj informacij, da lahko obstaja matematična funkcija, ki povezuje vhode z izhodi. Od nevronske mreže ni mogoče pričakovati, da se nauči funkcijo, ki je iz podatkov teoretično ni mogoče najti. Zbrati zadostno količino podatkov in najti dobre vhodne značilke, je velikokrat bolj pomembno in zamudno kot učenje mreže.
- Funkcija, ki povezuje vhode z izhodi, mora biti relativno gladka. To pomeni, da majhna sprememba vhoda pripelje do majhne spremembe izhoda. Nekatere vrste nevronskih mrež se lahko naučijo tudi nezveznosti, a to ne velja za vse. Če je funkcija zelo groba, kot to velja za na primer enkriptirane zapise, rezultatov mreže na podlagi takih podatkov ne moremo generalizirati.
- Učni primeri morajo reprezentativno predstavljati vse scenarije, na katere želimo generalizirati. Če uporabimo statistično terminologijo, mora biti vzorec reprezentativen za celotno populacijo. Želimo si, da so napovedi posledica interpolacije in ne ekstrapolacije [16].

3.1.4 Veljavnost

Obstaja več metod za ugotavljanje sposobnosti modela za generalizacijo. Nekateri viri navajajo uporabo statističnih metod za analizo vzorca in populacije, a se te pri nevronskih

mrežah ne uporabljajo veliko. Največ se uporabljata metodi razdelitve in prečnega preverjanja veljavnosti.

Metoda razdelitve je najbolj pogosto uporabljena pri delu z nevronskimi mrežami. Podatke razdeli na učni in testni del, kot kaže slika 3.6. Model učimo na učnem delu podatkov, testnega dela pa za učenje ne uporabimo. Naučeno mrežo izpostavimo vhodnemu delu testnih podatkov in za njih dobimo napovedane izhode. Napovedi primerjamo z izhodnimi vrednostmi testnega dela in dobimo nepristransko oceno napake pri generalizaciji. Slaba stran te metode je, da razdelitev podatkov zmanjša količino podatkov za učenje in testiranje.



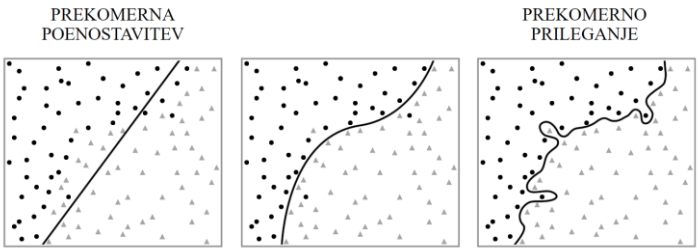
Slika 3.6: Metoda razdelitve

Metoda prečnega preverjanja temelji na podobnem principu kot metoda razdelitve. Predpostavimo, da podatke razdelimo na pet delov, čeprav je v splošnem število delov poljubno (ponavadi 3, 5, 10). V prvi iteraciji uporabimo prvi del kot testni del podatkov, vse ostale dele pa kot učni del. To ponovimo petkrat, pri čemer je testni del vsakič nov del. Tako dobimo 5 rezultatov, katerih povprečna vrednost predstavlja uspešnost modela. Shema prečnega preverjanja je na sliki 3.7. Težava pri tem načinu validacije je ta, da mora učenje modela potekati večkrat, kar je časovno in računsko potratno [16].

Slika 3.7: Metoda prečnega preverjanja

3.1.5 Prileganje

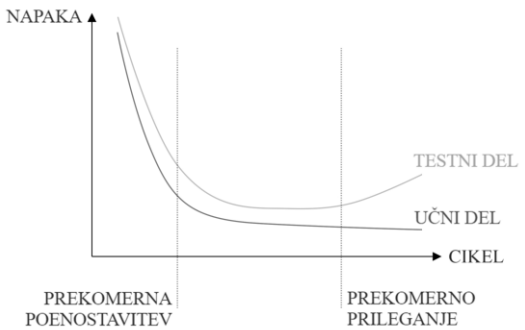
Pri učenju mreže moramo venomer paziti na premajhno in prekomerno prileganje podatkom. Model, ki ni dovolj kompleksen, ni zmožen zaznati funkcije, ki jo niz podatkov popisuje. Tak model ne popiše dobro učnih podatkov in ga ni mogoče generalizirati na nove podatke. Pravimo, da je prišlo do prekomerne poenostavitve, kar je prikazano na sliki 3.8 levo. Model, ki je preveč kompleksen, pa se podatkom lahko prekomerno prilagodi in preveč dobro popiše tudi šum v podatkih. Temu pravimo prekomerno prileganje oziroma prenasičenje modela, kar je prikazano na sliki 3.8 desno [16].



Slika 3.8: Shematska slika prileganja podatkom

Posebej nevarno je prenasičenje, saj ga je pogosto težko zaznati. Nevarnost prenasičenja se pojavi, če sta izpolnjena dva pogoja:
- Število vhodnih značilk je veliko v primerjavi s številom učnih primerov. Ponavadi želimo imeti vsaj desetkrat več učnih primerov kot vhodnih značilk.
- Obstaja velika medsebojna odvisnost med vhodnimi značilkami [16].

V primeru nevronskih mrež je prenasičenje mogoče zaznati z izrisovanjem napake na testnem in učnem delu med učenjem. To prikazuje slika 3.9. Na začetku je napaka velika, saj model še ni dovolj prilagojen podatkom. Napaka na učnem in testnem delu se nato ob zadovoljivih nastavitvah mreže začne zmanjševati. Če se napaka za učni del še naprej zmanjšuje, napaka za testni del pa se na neki točki začne povečevati, lahko predvidevamo, da prihaja do prenasičenja [13]. Prenasičanje lahko preprečimo tako, da učenje ustavimo dovolj hitro. Drugi načini preprečevanja so še omejevanje uteži, izpuščanje nevronov, združevanje modelov in dodajanje zašumljenih podatkov [16].



Slika 3.9: Spreminjanje napake na učnem in testnem delu podatkov pri učenju [13]

3.1.6 Vrste mrež

V prejšnjih poglavjih je bil predstavljen koncept usmerjene nevronske mreže, a poznamo več vrst mreženja. Teh je mnogo, zato bodo v magistrskem delu predstavljene le nekatere. Spodaj je predstavljenih nekaj najpopularnejših metod, ki se uporabljajo danes. Pomembno pa se je zavedati, da je vpliv vrste mreže omejen. Izkaže se, da so že najpreprostejši načini mreženja teoretično sposobni popisati kakršno koli funkcijo. Pri izbiri vrste mreže je treba sprejemati kompromise. Nekatere je lažje učiti, druge potrebujejo manj računskega časa, tretje pa so bolj odporne proti šumu v podatkih.

Konvolucijske mreže se večinoma uporabljajo za analizo slik. Ena glavnih težav usmerjenih nevronskih mrež pri analizi slik je, da je količina vhodnih podatkov zelo velika. Vsako sliko predstavlja tridimenzionalna matrika slikovnih točk (širina, višina in barva). Struktura in delovanje konvolucijskih mrež je zapletena. Na tej točki omenimo samo, da s pomočjo manjših filtrov, napetih čez sliko in konvolucijo, pridemo do sistema, ki je sposoben prepoznati kompleksne vzorce v slikah [15].

Povratne mreže uporabljamo predvsem tam, kjer so med vhodi velike odvisnosti. Najpogosteje jih uporabljamo v kombinaciji z jezikom (tekst in govor). Ideja povratnih nevronskih mrež je, da v mreži obstajajo povezave tudi nazaj. To lahko pomeni, da gre povezava na enega od prejšnjih nivojev ali pa, da nevron svoj izhod pelje sebi na vhod. To mreži doda neke vrste spomin in ji omogoči, da se nauči prepoznavanja in reprodukcije zaporedij in časovnega predvidevanja [14,15].

Avtoenkoderji so simetrične mreže, kjer je na vhodu in izhodu mreže isti vzorec podatkov. Ko se mreža uči, funkcija napake meri, kako dobro mreža rekreira vhode. Izhod takih mrež postane aproksimacija vhoda. Avtoenkoderske mreže se pogosto uporabljajo za kompresijo podatkov [14,15].

O **globokih nevronskih mrežah** oziroma globokem učenju je danes veliko govora. Tu ne gre za poseben način mreženja, glavna lastnost je dejansko globina (število nivojev) mreže. Globoke mreže so lahko usmerjene, konvolucijske, povratne ali katere druge vrste. Število nivojev, potrebnih, da nevronski mreži pravimo globoka nevronska mreža, se spreminja. Pred nekaj leti je bilo 10 nivojev dovolj, danes pa med globoke mreže navadno uvrščamo mreže s 100 in več nivoji [13].

3.1.7 Parametri in hiperparametri

V literaturi pogosto prihaja do napak pri imenovanju parametrov in hiperparametrov mreže. Parametri mreže so tisti, katerih vrednost mora model poznati za napovedovanje. So notranje spremenljivke modela, ki se izračunajo iz podatkov. Pri nevronskih mrežah so to uteži, pripisane posameznim signalom. Hiperparametri pa so modelu predpisane spremenljivke, preden optimiziramo njegove parametre. So spremenljivke, katerih vrednosti iz podatkov ne moremo dobiti. Navadno so izbrane s strani osebe, ki model ustvarja. Njihovih vrednosti ne moremo poznati, lahko jih le ocenimo, kopiramo iz drugih del ali pa iščemo s poskušanjem. Na izbiro hiperparametrov lahko gledamo kot izbor modela. V nadaljevanju so opisani najpomembnejši hiperparametri nevronskih mrež [17].

3.1.8 Aktivacijska funkcija

Aktivacijska funkcija ima v mreži pomembno vlogo in je nujno potrebna. Zamislimo si, da namesto nje v enačbi nevrona nastopa identiteta. Tako bi vrednosti signalov na posameznih nivojih izgledale, kot kažeta enačbi (3.2) in (3.3). Če ju združimo, dobimo enačbo (3.4). Ker so uteži konstante, jo lahko zapišemo tudi v obliki (3.5).

$$x_1 = w_1\, x_0 \tag{3.2}$$

$$x_2 = w_2\, x_1 \tag{3.3}$$

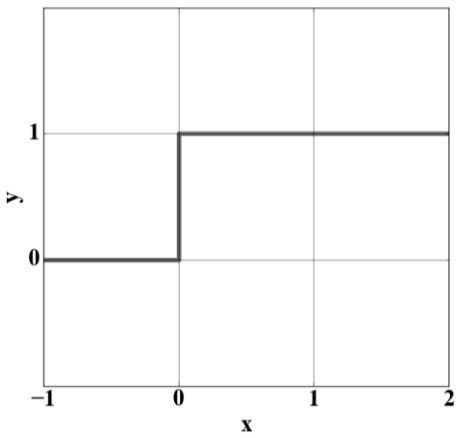
$$x_2 = w_2\, (w_1\, x_0) \tag{3.4}$$

$$x_2 = w'\, x_0 \tag{3.5}$$

Enačba (3.5) pa je enake oblike kot izhodni enačbi. Izkaže se, da kombinacija linearnih transformacij vedno vrne linearno transformacijo. V nevron moramo torej vgraditi nelinearno funkcijo, saj bi bilo v nasprotnem primeru vseeno, ali je nevronska mreža globoka ali pa ima le en nevron. To težavo je mogoče rešiti z zelo enostavnimi nelinearnimi funkcijami. V njih pride do odločitve, v kakšni obliki bo nevron poslal signal naprej (signal lahko sploh ni posredovan naprej). Spodaj je opisanih nekaj primerov.

Prva in najbolj osnovna je koračna funkcija (angl. *step function*). Njena enačba je zapisana v (3.6), njen graf pa lahko vidimo na sliki 3.10. Definiran ima prag, ki ga more vsota obteženih signalov preseči, da vrne 1, sicer vrne 0 [14]. Oznaka φ predstavlja aktivacijsko funkcijo, oznaka x pa neodvisno spremenljivko.

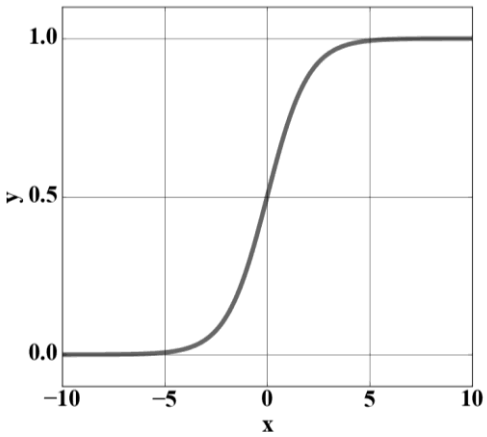
$$\varphi(x) = \begin{cases} 0;\ \text{za } x < 0 \\ 1;\ \text{za } x \geq 0 \end{cases} \tag{3.6}$$



Slika 3.10: Koračna funkcija

Izkaže se, da koračna funkcija ni najboljša, saj se mreža pri njeni uporabi ne more učiti malo po malo. Prehod med 0 in 1 more biti za tako učenje bolj gladek. V modernih nevronskih mrežah jo pogosto nadomešča sigmoidna funkcija (angl. *sigmoid function*). Gre za zvezno funkcijo, zapisano v enačbi (3.7) in prikazano na sliki 3.11 [13]**.**

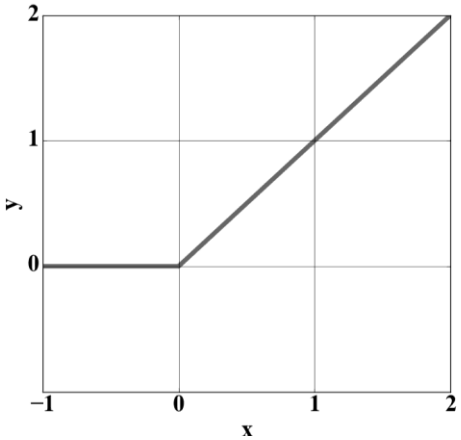
$$\varphi(x) = \frac{1}{1 + e^{-x}} \tag{3.7}$$



Slika 3.11: Sigmoidna funkcija

Pogosto se uporablja tudi strmina oziroma funkcija ReLU (angl. *rectified linear unit*). Ta preprosta funkcija pogosto daje zelo dobre eksperimentalne rezultate. Za razliko od prejšnjih funkcij njena zaloga vrednosti ni med 0 in 1, temveč med 0 in neskončno. Zapišemo jo, kot kaže enačba (3.8), na sliki 3.12 pa lahko vidimo njen graf.

$$\varphi(x) = \begin{cases} 0; \text{ za } x < 0 \\ x; \text{ za } x \geq 0 \end{cases} \tag{3.8}$$



Slika 3.12: Funkcija ReLU

Možnosti pri izbiri aktivacijske funkcije je seveda mnogo in nismo omejeni le z opisanimi. Med drugimi so to tangenshiperbolična, softmax, softplus in softsign funkcija [13]**.**

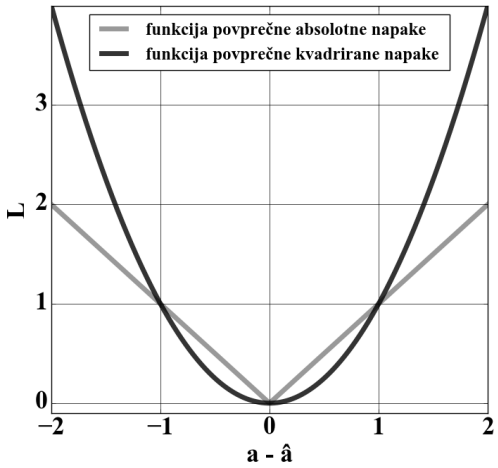
3.1.9 Funkcija napake

Namen funkcije napake je primerjava napovedane vrednosti na izhodu s pravo vrednostjo. Napovedano vrednost nam generira nevronska mreža, pravo vrednost pa najdemo v učnih podatkih. Treba je najti funkcijo, ki bo minimizirana, ko bosta vrednosti enaki. Izbira uporabljene funkcije je odvisna od obravnavanega problema [14]. Spodaj je opisanih nekaj funkcij, uporabljenih pri regresiji in pri klasifikaciji.

Najpogosteje uporabljeni funkciji napake za regresijski problem sta povprečna absolutna napaka (angl. *mean absolute error*), zapisana v enačbi (3.9), in povprečna kvadrirana napaka (angl. *mean squared error*), zapisana v enačbi (3.10). V enačbah oznaka a predstavlja pravo vrednost, oznaka $\hat{a}$ pa napovedano vrednost. Število napovedi označuje n, oznaka j pa posamezno napoved. Prva je manj občutljiva na šum in izstopajoče točke, druga pa izstopajoče točke in velike napake bolj kaznuje. Druga je povsod diferenciabilna, kar pomaga pri računanju gradienta pri optimizacijskem algoritmu. Obe funkciji za eno napoved prikazuje slika 3.13.

$$L = \frac{1}{n}\sum_{j=1}^{n} |a_j - \hat{a}_j| \tag{3.9}$$

$$L = \frac{1}{n}\sum_{j=1}^{n} (a_j - \hat{a}_j)^2 \tag{3.10}$$

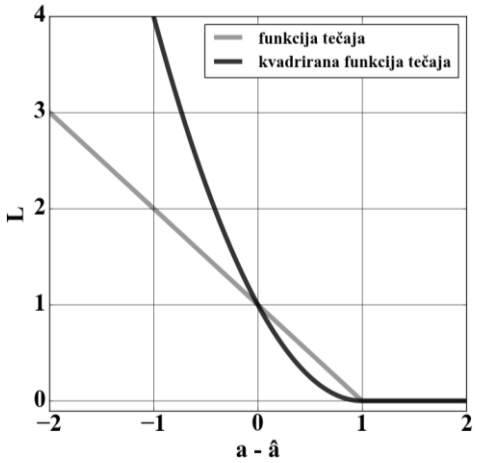


Slika 3.13: Funkcija povprečne absolutne napake in povprečne kvadrirane napake

Za napovedovanje kategoričnih vrednosti pa zgornji funkciji ne bi bili primerni. Eden od razlogov je ordinalnost. Pri kategoričnih vrednostih si 0 in 1 nista nič bolj podobni kot 0 in 5 [14]. Primera funkcij napake pri klasifikaciji sta funkcija tečaja (angl. *hinge function*), zapisana z enačbo (3.11), in kvadrirana funkcija tečaja (angl. *squared hinge function*), zapisana z enačbo (3.12). Obe funkciji za eno napoved prikazuje slika 3.14.

$$L = \frac{1}{n}\sum_{j=1}^{n} \max\left(0,\ 1 - a_j\,\hat{a}_j\right) \tag{3.11}$$

$$L = \frac{1}{n}\sum_{j=1}^{n} \left(\max\left(0,\ 1 - a_j\,\hat{a}_j\right)\right)^2 \tag{3.12}$$



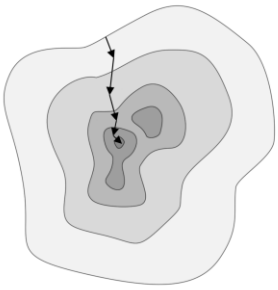
Slika 3.14: Funkcija tečaja in kvadrirana funkcija tečaja

Predstavljenih je bilo le nekaj funkcij napake. Sicer jih za različne namene poznamo še mnogo. Med pomembnejšimi so še povprečna kvadratna logaritemska napaka (angl. *mean squared logarithmic error*), napaka L2, povprečna absolutna procentualna napaka (angl. *mean absolute percentage error*), Kullback Leibler divergenca (angl. *Kullback Leibler divergence*), križna entropija (angl. *cross entropy*), Poissonova funkcija in kosinusna bližina (angl. *cosine proximity*) [18].

3.1.10 Optimizacijski algoritem

Nevronska mreža poda napoved ter jo s pomočjo funkcije napake primerja s pravo vrednostjo. Nato je treba mrežo prilagoditi tako, da bodo napovedi naslednjič bolj natančne. To počne optimizacijski algoritem. Uteži mreže prilagodi na tak način, da se funkcija napake minimizira.

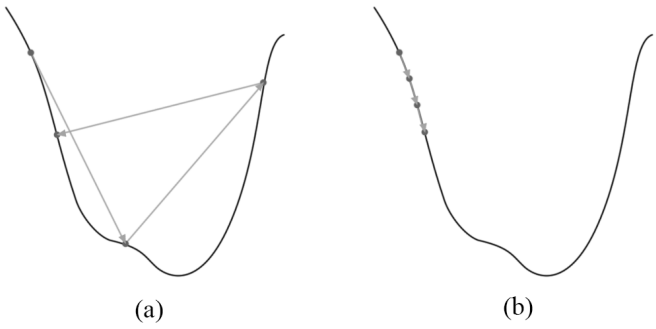
Najpomembnejši optimizacijski algoritmi temeljijo na ideji gradientnega spusta. Če si funkcijo napake predstavljamo kot površino v prostoru, lahko njen lokalni minimum najdemo tako, da se po površini sprehajamo navzdol, dokler je to mogoče. Tak spust prikazuje slika 3.15. Gradient nam pove smer, v katero moramo spremeniti napoved, ne pa tudi, kako spremeniti uteži. Te običajno spremenimo proporcionalno stopnji, s katero so prispevale k napovedi. Uteži signalov, ki so napoved približale pravi vrednosti, povečamo, tiste, ki so storile obratno, pa zmanjšamo [14].



Slika 3.15: Vizualizacija gradientnega spusta [14]

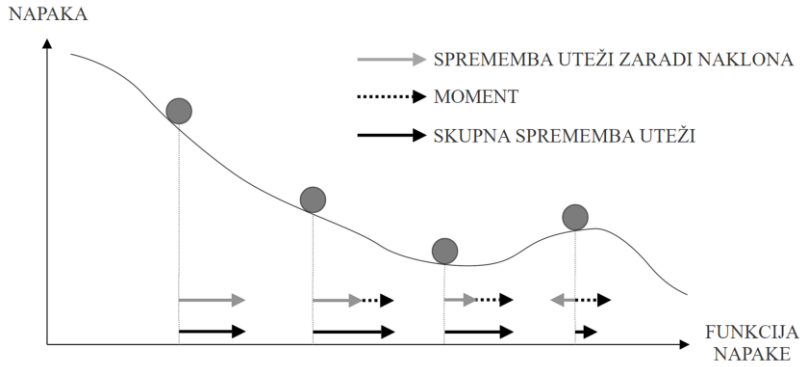
Eden najbolj priljubljenih optimizacijskih algoritmov je SGD oziroma stohastični gradientni spust. Uteži se pri tem načinu prilagajajo za vsak učni primer. Pogosto prilagajanje sicer povzroča hitro konvergenco proti minimumu funkcije napake, a povzroča veliko nihanje te vrednosti, kar je težavno, saj ne vemo, točno kdaj učenje ustaviti. Za SGD velja, da je hitrost učenja enaka za vse uteži in se ne spreminja med procesom učenja. Bolj napredni optimizacijski algoritmi, kot sta RMSProp in ADAM, pa hitrost učenja prilagajajo, kar nam pogosto producira boljše rezultate, do katerih pride tudi hitereje [19].

Hitrost učenja govori o tem, kako hitro se premikamo proti optimalnim vrednostim uteži. Majhne spremembe uteži oziroma počasno učenje je lahko problematično, saj rešitev prepočasi konvergira proti minimumu funkcije napake. Pri velikih korakih oziroma hitrem učenju pa lahko optimalne parametre preskočimo, skačemo okrog minimuma ali celo popolnoma zaidemo s poti. Obe možnosti sta prikazani na sliki 3.16. Kot omenjeno v prejšnjem poglavju, imajo nekateri optimizacijski algoritmi konstantno hitrost učenja, drugim pa se lahko ta med učenjem spreminja [20].



Slika 3.16: Primer velike (a) in majhne (b) hitrosti učenja [13]

Moment je enostaven koncept, s katerim lahko povečamo hitrosti konvergence in zmanjšamo nihanje funkcije napake. Pri spreminjanju uteži spremembi dodamo še delež spremembe te uteži iz prejšnje iteracije. Tako pride do akumulacije uteži v tisti smeri, kamor se uteži pretežno spreminjajo. To pomaga pri grobem ugotavljanju smeri spusta in je koristno tudi zato, da lahko preskočimo majhen »hribček« funkcije napake (glejte sliko 3.17) [21].

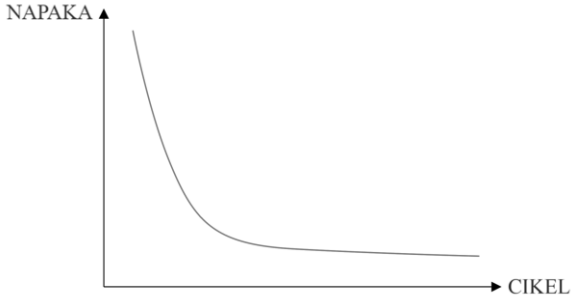


Slika 3.17: Moment [13]

Tako hitrost učenja kot moment uvrščamo med hiperparametre, saj ju mora izvajalec pogosto določiti sam. Nekateri moderni optimizacijski algoritmi pa ju prilagajajo že sami, zato je poskušanja in iskanja primernih vrednosti vedno manj.

3.1.11 Število ciklov in velikost paketa

To sta hiperparametra, ki močno vplivata na čas učenja mreže. Število ciklov nam pove, kolikokrat je mreža izpostavljena nizu podatkov. Velikost paketa pa je število učnih primerov, ki jih mreža vidi, preden pride do spremembe uteži. Pomembno je, da ne pride do mešanja z izrazom iteracija. Število iteracij nam pove število prehodov paketa učnih primerov skozi mrežo. Predstavljajmo si, da imamo 200 učnih primerov in je velikost paketa nastavljena na 100. Za izvedbo enega cikla sta tako potrebni 2 iteraciji. Slika 3.18 prikazuje zmanjševanje napake v odvisnosti od ciklov [13].

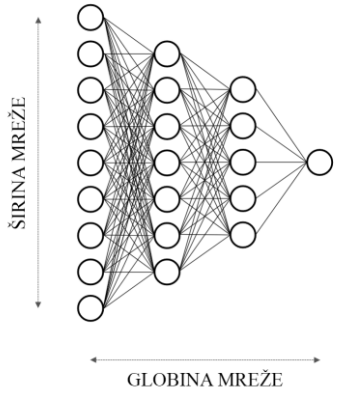


Slika 3.18: Spreminjanje napake pri učenju

Povečanje števila ciklov sorazmerno poveča čas učenja mreže. Učenje mreže s 300 cikli bo trajalo desetkrat dlje kot učenje mreže s 30 cikli. Povečanje velikosti paketa pa bo povečalo zasedenost računalniškega pomnilnika.

3.1.12 Število nevronov in nivojev

O globini in širini mreže oziroma njeni topologiji smo govorili že v poglavju *3.1.1 Sestava mreže*. Slika 3.19 prikazuje primer topologije mreže. Ta lahko močno vpliva na rezultate naše analize. Povečevanje širine in globine mreže poveča kompleksnost modela, kar poveča tudi računski čas, saj je treba optimizirati več parametrov. Število potrebnih nivojev in nevronov za želen rezultat mreže je močno odvisno od karakteristik problema.

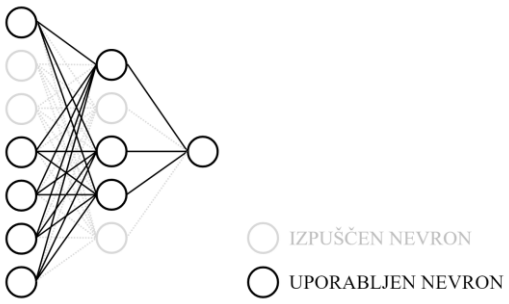


Slika 3.19: Širina in globina mreže

Mreža vedno vsebuje en vhodni nivo. Število nevronov v tem nivoju je enako številu značilk v podatkih. Izhodni nivo je prav tako en sam. Število nevronov je odvisno od problema. Če se ukvarjamo z regresijo, je en nevron na izhodu, če pa gre za klasifikacijski problem, potem je število nevronov ponavadi enako številu razredov klasifikacije. Večjo težavo predstavljata izbira števila skritih nivojev ter njihovih nevronov. Za nekatere primere skritega nivoja sploh ne potrebujemo, saj linearni modeli zadoščajo. Veliko število primerov bo model z enim ali dvema skritima nivojema popisal dovolj dobro. V literaturi obstaja veliko metod, s katerimi naj bi ocenili število skritih nivojev in nevronov, vendar pa so večinoma nepopolne in niso sprejete s strani širše raziskovalne skupnosti. Izbira je odvisna od številnih dejavnikov, ki jih je težko strniti v preprost nasvet. Število nevronov na nekem nivoju je odvisno od števila nevronov v vseh ostalih nivojih, števila učnih primerov, količine šuma, kompleksnosti učnega primera, aktivacijske funkcije, optimizacijskega algoritma in še mnogo drugih stvari [16]. Težavo ponavadi rešujemo kar s poskušanjem, bolj vešči pa tudi z intuicijo.

3.1.13 Izpuščanje nevronov

Izpuščanje nevronov je ena izmed tehnik regularizacije nevronskih mrež, ki do določene mere preprečuje prenasičenje. Deluje tako, da iz mreže naključno izključuje nevrone (glejte sliko 3.20). Izločeni nevroni pri določenem prehodu podatkov ne sodelujejo, zato njihova »pomoč« pri napovedi začasno ni prisotna. Na ta način so preostali nevroni prisiljeni povečati svoj vpliv na napoved. Delež izpuščenih nevronov definiramo s številko med 0 in 1. V prvem primeru ne bo izpuščenih nevronov, v drugem pa bodo vsi. V praksi se uporabljajo vrednosti do približno 0,5 [15].



Slika 3.20: Shematski prikaz izpuščanja nevronov

3.1.14 Inicializacija in omejevanje uteži

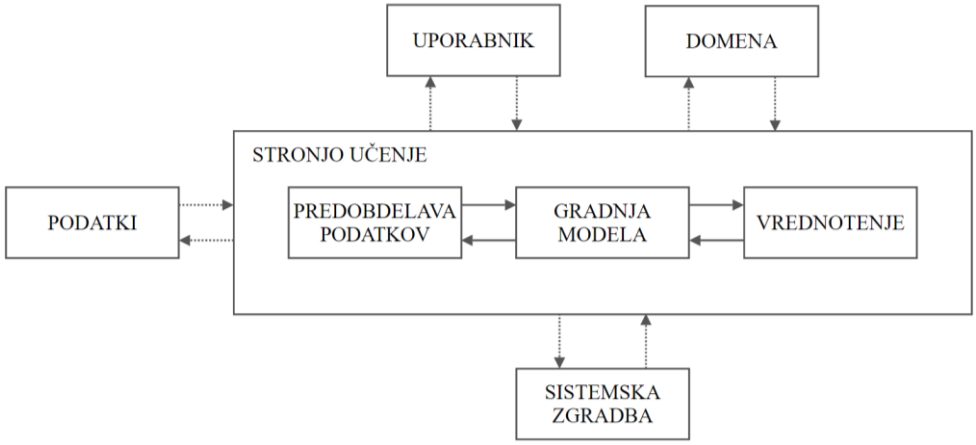
Uteži signalov v nevronski mreži se z učenjem spreminjajo. Vseeno pa jim moramo pred začetkom učenja predpisati vrednost. Obstaja več načinov predpisovanja vrednosti utežem, čemur pravimo inicializacija uteži. Način, ki se je pogosto uporabljal v preteklosti, je, da vsem utežem predpišemo vrednost nič. Danes je ta način manj pogost, saj se je izkazalo, da je bolje, če so začetne uteži majhne, a ne enake [14]. Bolj pogost je naključen izbor iz majhnega intervala. Porazdelitev izbranih vrednosti pa je lahko enakomerna ali pa sledi na primer normalni porazdelitvi [13].

Utežem lahko predpišemo tudi maksimalno vrednost, ki jo lahko zavzamejo [13]. Temu pravimo omejevanje uteži. Določene vhodne značilke k napovedi nevronske mreže pripomorejo bolj kot druge, kar je posledica bolj obteženih signalov teh značilk. To je do določene mere zaželeno, včasih pa privede do prevelikega prilagajanja podatkom. Z omejevanjem uteži lahko to težavo rešimo.

3.2 Strojno učenje z nevronskimi mrežami

Za zmanjševanje nepopolnosti podatkov smo se odločili uporabiti eno izmed metod strojnega učenja. Pri delu na tako kompleksnem, interaktivnem in pogosto iterativnem procesu se to pogosto razdeli na več manjših sklopov [22,23]. V literaturi najdemo mnogo

različnih razdelitev dela pri strojnem učenju, ki imajo večinoma veliko podobnosti. Naše delo si lahko predstavljamo, kot shematsko prikazuje blokovni diagram na sliki 3.21.



Slika 3.21: Blokovna shema dela s strojnim učenjem [24]

Glavni del predstavlja strojno učenje, pri katerem se ukvarjamo s predobdelavo podatkov, gradnjo modela in vrednotenjem. Podroben pogled na povezave pri strojnem učenju razkrije, da povezave potekajo v obe smeri. Razlog za to je, da proces ne teče le iz leve proti desni, temveč je ponavadi iterativen. Sklop na desni nam namreč lahko razkrije nov pogled oziroma razumevanje sklopa na levi, zaradi česar se nanj vrnemo in nato tam opravimo dodatno delo. Temu glavnemu okvirju dodamo še štiri druge komponente. To so podatki, uporabnik, domena in zgradba sistema. Tudi tu so povezave dvosmerne, kar nakazuje na soodvisnost sklopov. Te štiri komponente namreč vplivajo na strojno učenje, prav tako pa strojno učenje vpliva na njih [24].

3.2.1 Podatki

Vrsta podatkov, njihova količina in njihova heterogenost so dejavniki, ki močno vplivajo na strojno učenje. Vplivajo na izbiro metode strojnega učenja, način predobdelave podatkov, način predstavitve rezultatov in še marsikaj. Pri analizi na primer masivnih podatkov (angl. *big data*) mora biti veliko pozornosti posvečene učinkovitosti procesa. Pogosto morajo biti metode strojnega učenja temu prilagojene [24]. Podatki primarno predstavljajo strojnemu učenju vhod, kar velja tudi za naš primer zmanjševanja nepopolnosti podatkov. Ko pa z njim napovemo manjkajoče vrednosti, te postanejo del podatkov. Povezava med podatki in strojnim učenjem torej poteka v obeh smereh.

3.2.2 Uporabnik

Uporabnik je v interakciji s strojnim učenjem tako, da v proces vključuje znanje, zahteve sistema ter osebne želje. Podaja povratne informacije o uporabnosti ter izkorišča rezultate

za izboljševanje modela strojnega učenja kot tudi procesa samega. Končni uporabnik namreč ni edini, ki tak sistem vzpostavlja. Tisti, ki sistem postavljajo, sprejemajo največ odločitev, kot so način zajemanja podatkov, specifike strojnega učenja in celo vrednotenje rezultatov. Vključevanje uporabnika sistema v ta proces privede do bolj učinkovitih sistemov z boljšo uporabniško izkušnjo [24].

3.2.3 Domena

Znanje o domeni pomaga najti vzorce in povezave, ki jih z analizo podatkov samih ne bi mogli najti. Razlog za to je lahko, da količina učnih podatkov ni zadostna ali pa ti niso reprezentativni. Pridobitev primernih podatkov je včasih iz različnih razlogov celo nemogoča. Domena nam pri tem pomaga tako, da z njenim poznavanjem lažje postavljamo hipoteze, pripravljamo podatke, prilagajamo učne cilje in spreminjamo samo iskanje. Najdeni vzorci pa na koncu posodobijo in povečajo znanje o domeni [24].

3.2.4 Sistemska zgradba

Sistemska zgradba oziroma programska in strojna oprema sestavljata okolje, v katerem strojno učenje poteka. Ta močno vpliva na to, kako bodo algoritmi za učenje tekli in kako učinkovit bo ta proces. Velika razlika je med sistemom, ki teče na več jedrnem procesorju osebnega računalnika, in porazdeljenim sistemom na več grafičnih karticah. Nezanemarljiv je pa tudi vpliv strojnega učenja na oblikovanje sistemske arhitekture prihodnosti [24].

3.2.5 Strojno učenje

Strojno učenje razdelimo na tri sklope, kot kaže slika 3.21. Prvi sklop je predobdelava, ki je opisan v nadaljevanju. Nekateri avtorji pa pred predobdelavo postavljajo še druge korake, ki jih bomo na kratko opisali.

Prvi v vrsti dodatnih korakov je definicija problema, ki ga rešujemo. To je pomembno, saj, ko natančno razumemo cilj naloge, lažje sprejemamo pravilne odločitve za rešitev problema. Temu lahko sledi zajemanje podatkov [10]. Želimo si zajeti podatke, ki niso močno zašumljeni, pristranski ali celo napačni. V mnogo primerih so podatki že zajeti, zato ta korak spustimo in gremo k razumevanju podatkov [22]. Tu je pomembno poznavanje metode zajemanja podatkov kot tudi znanje o domeni, s katerim si razlagamo pomen zajetih podatkov. Nato sledi priprava podatkov. Do podatkov moramo dostopati in v določenih primerih iz njih izbrati reprezentativni pod-sklop podatkov, ki ga bomo uporabljali pri predobdelavi [23,25].

Predobdelava podatkov
Predobdelava podatkov je pomemben korak, saj z njim zagotovimo, da bodo modeli strojnega učenja na podatkih dobro delovali. Realni »surovi« podatki ponavadi niso

primerni za učenje modelov. Kot omenjeno v poglavju *2.1 Kakovost podatkov*, so lahko netočni, neusklajeni, nepopolni, neskladni itd. Glavna naloga tega koraka je izboljšati kakovost podatkov, jih obdelati in pripraviti za uporabo pri modelih strojnega učenja. To je korak, za katerega je pogosto potrebno več časa in drugih virov, kot za samo metodo strojnega učenja. Predobdelavo delimo na čiščenje in transformacijo podatkov.

Čiščenje podatkov zajema delo s šumom v podatkih, odstranjevanje izoliranih točk, delo z manjkajočimi vrednostmi, reševanje nekonsistentnosti in upoštevanje časovnih zaporedij [10]. Pri tem delu moramo veliko pozornosti posvetiti načinu čiščenja podatkov. Glajenje šuma lahko iz podatkov odstrani pomembne vzorce. Odstranjevanje izoliranih točk lahko v podatke vnese pristranskost. Izbira metode za delo z manjkajočimi vrednostmi lahko močno zmanjša statistično moč niza podatkov. Vseeno pa je ta korak izjemnega pomena in ga ne smemo preprosto izpustiti. Pristranskost podatkov bi bila v tem primeru lahko večja, saj je nekonsistentnost podatkov in njihovo podvajanje lahko še bolj škodljivo [24].

Transformacija podatkov zagotavlja, da dobimo niz podatkov, ki je v ustrezni obliki za uporabo z metodami strojnega učenja in omogoča njihovo dobro delovanje. Pomembna je redukcija podatkov. Pri njej želimo odstraniti nepotrebne atribute in tako zmanjšati količino podatkov. Pravilno reducirani podatki pa ne smejo izgubiti pomembnih informacij. Rezultati, pridobljeni na reduciranih podatkih, bi morali biti enaki ali vsaj podobni rezultatom, pridobljenih na podlagi vseh podatkov, a bi do njih morali priti hitreje [10,26]. Reducirane podatke želimo nato spraviti v obliko, da bodo kar najbolj koristni za izbrano metodo strojnega učenja. Temu pravimo tudi generacija značilk. Rezultati strojnega učenja so močno odvisni od ustvarjenih značilk. Tu ima pomembno vlogo znanje o domeni, saj brez tega težko vemo, kakšne značilke pozitivno vplivajo na napovedovanje [24]. Izbira značilk je odvisna tudi od uporabljene metode strojnega učenja. Za uporabo nekaterih metod moramo kategorične vrednosti preslikati v numerične, za uporabo drugih pa moramo kontinuirane vrednosti diskretizirati (preslikati v kategorije, ki se med seboj ne prekrivajo) [24]. Nekaterim metodam močno pomaga, če vrednosti vseh atributov zajemajo enak interval, drugim pomaga, če podatke centriramo, tretjim, če jih logaritmiramo in tako naprej [10,25]. Končni transformirani podatki morajo biti integrirani. To pomeni, da združujejo vse vire, vrste podatkov in značilk v eno bazo podatkov ali eno datoteko [10]. Taki podatki so nato pripravljeni na nadaljnje delo.

Gradnja modela

Pri tem koraku želimo iz podatkov izvleči uporabno znanje s pomočjo samodejnega računalniškega procesa. To naredimo z izbiro metode strojnega učenja in ustreznega algoritma za izvedbo izbrane metode ter načinom predstavitve rezultatov te analize [23].

Izbira uporabljene metode temelji na uskladitvi ciljev, ki jih želimo izpolniti, in ciljev, ki jih je izbrana metoda sposobna izpolniti. Pretehtati moramo, kakšen model in kakšni hiperparametri bodo skupaj zagotavljali rezultate, ki jih od tega koraka pričakujemo. Kot že omenjeno, nekateri modeli namreč bolje delujejo s kategoričnimi spremenljivkami, drugi pa z numeričnimi spremenljivkami [22]. Poznamo več vrst metod strojnega učenja, ki jih ponavadi delimo na napovedne in opisne. Pri napovednih metodah gre za napovedovanje na podlagi podatkov iz preteklosti. Regresijske metode napovedujejo na zvezni skali, pri klasifikacijskih metodah pa napovedujemo kategorične vrednosti. Pri opisnih metodah pa v podatkih iščemo odnose in vzorce. Gručenje razvrsti primere v

skupine med seboj podobnih podatkov, asociacija pa identificira skupine, ki se pogosto pojavljajo skupaj. Raziskave kažejo, da se pri delu s proizvodnimi podatki in pri analizi kakovosti proizvodnega procesa najpogosteje uporablja regresija. Ta je uporabljena v skoraj polovici primerov. Klasifikacija, uporabljena v četrtini primerov, je druga najpogostejša. V približno 15 % je uporabljeno gručenje, medtem ko je asociacija uporabljena zelo redko [25].

Izbran model moramo nato implementirati in ga naučiti na podatkih. Prvi ustvarjen model skoraj zagotovo ne ob optimalen za reševanje našega problema, zato želimo napovedovanje iterativno izboljševati. To lahko naredimo s tem, da spreminjamo vhodne podatke, ki jih modelu zagotovimo. Morda ugotovimo, da jih je bolj smiselno transformirati na drug način, izbrati druge značilke in podobno. Napovedovanje pa lahko izboljšamo tudi s spremembo nastavitev modela. Privzete nastavitve modela po vsej verjetnosti našega problema z našimi podatki ne rešujejo najbolje. Zato se lotimo optimizacije hiperparametrov. V nadaljevanju je opisanih nekaj pogostih načinov tega dela.

Mrežno iskanje je najpreprostejša oblika iskanja. Za več hiperparametrov definiramo niz vrednosti, ki naj jih zavzamejo. Za vsako kombinacijo hiperparametrov je ustvarjen in naučen nov model. Izmed vseh rezultatov poiščemo najboljšega in pripadajoči najboljši model. Težava te metode je, da je zelo dolgotrajna. Če spreminjamo 10 vrednosti 5 hiperparametrom, moramo model učiti 100.000-krat. Če enkratno učenje modela traja le 5 minut, celotno iskane tako traja 1 leto [27].

Naključno iskanje je v principu podobno mrežnemu, le da izmed množice vseh kombinacij naključno izberemo podmnožico kombinacij. Za zgornji primer lahko vzamemo le 2000 kombinacij in iskanje traja namesto 1 leto le 1 teden. Pri popolnoma naključnem iskanju so si določene kombinacije lahko med seboj preveč podobne, zato se pogosto uporablja »kvazi« naključno iskanje, ki ta problem rešuje. Tak način ima včasih težave pri izbiri kombinacij, kadar upoštevamo veliko število hiperparametrov [27].

Ročno iskanje pomeni, da kombinacije hiperparametrov preskušamo eno za drugo sami. To nam vzame veliko časa, saj moramo biti pri iskanju optimalne kombinacije ves čas prisotni. Velika prednost pa je to, da sproti spremljamo rezultate in se učimo na napakah. Pri vsaki iteraciji analiziramo prejšnje rezultate in se lažje odločamo, kaj poskusiti naslednjič. Pri tem razvijamo intuicijo, kakšne hiperparametre kdaj uporabiti [27].

Bayesovo iskanje lahko temelji na različnih algoritmih, ki so v nekaterih pogledih podobni ročnemu iskanju. Pri iskanju najboljše kombinacije hiperparametrov se navezujejo na rezultate preteklih poskusov. Na njihovi podlagi predvidevajo, kakšni bodo rezultati za še ne analizirano kombinacijo in sami predlagajo naslednjo kombinacijo hiperparametrov. Težava teh je, da še niso najbolj razširjeni, zaradi česar je njihova implementacija pogosto zahtevna [27].

Izbrana kombinacija hiperparametrov in uporabljeni podatki nam definirajo parametre modela. Te lahko shranimo za uporabo, da nam modela ni treba ponovno učiti. Ker je učenje lahko dolgotrajen proces, se včasih splača uporabiti že uporabljene modele, ki so jih naredili drugi raziskovalci [26].

3.2.6 Vrednotenje

Po zaključeni gradnji modela moramo rezultate, ki jih dobimo, vrednotiti. To počnemo s stališča točnosti, zanesljivosti, potrebnega časa in potrebnih virov [25]. Pri tem se lahko v vsakem trenutku vrednotenja vračamo na prejšnje korake in stvari popravljamo [22]. Ne gre namreč za linearno zaporedje korakov, temveč lahko nastopijo zanke med poljubnima dvema korakoma. Pri vrednotenju je treba rezultate interpretirati ter jih postaviti v kontekst. S tem namenom vzorce v podatkih, sam model in podatke vizualiziramo [23].

Temu sledi še implementacija, česar ponavadi ne štejemo pod strojno učenje. Pridobljeno znanje lahko uporabimo neposredno, lahko ga implementiramo v drug sistem ali pa ga le dokumentiramo in poročamo tistim, ki jih to znanje zanima. Včasih se na strojno učenje gleda tudi bolj ozko, kot je opisano v tem magistrskem delu. Nekateri štejejo k strojnemu učenju le del, ko gradimo model. To je morda smiselno v idealnih in nadzorovanih okoliščinah, a so vsi ostali koraki za reševanje realnih problemov prav tako pomembni in včasih celo težji za izvedbo [22].

4 Opis podatkov

Za potrebe strojnega učenja moramo podatke dobro razumeti. V tem poglavju so s tem namenom opisane vrste podatkov in njihove lastnosti, pomembne za strojno učenje. Prav tako pa moramo dobro poznati konkretne podatke, ki jih pri delu uporabljamo, in njihov izvor. To je predstavljeno v drugem delu tega poglavja.

4.1 Vrste podatkov

Podatke delimo na več načinov. Eden od načinov delitve je na strukturirane in nestrukturirane podatke.

Strukturirani podatki so tisti, ki sledijo vnaprej določeni shemi oziroma strukturi. Tipične predstavnike takih podatkov najdemo v bazah podatkov. Tam najdemo informacije (ponavadi v obliki teksta), ki so razvrščene v stolpce in vrstice. Ustvariti bazo podatkov, ki ustreza vsem zahtevam našega sistema, ni lahka naloga, še posebej, ker jo je treba definirati pred zapolnitvijo s podatki [28,29].

Nestrukturirani podatki pa so tisti, ki ne sledijo neki vnaprej definirani shemi oziroma strukturi. Niso shranjeni v vrstice in stolpce baze podatkov, saj bi bilo to zelo težko izvedljivo, čeprav ni nemogoče. Primeri takih podatkov so elektronska pošta, pdf-datoteke, video posnetki, objave na družbenih omrežjih in mnogi drugi [28,29].

Nestrukturirani podatki niso privzeto slabši od strukturiranih. V nestrukturiranih podatkih lahko najdemo več povezav in relacij med informacijami. Takih podatkov je tudi mnogo več, kar je v mnogih primerih ključnega pomena. Kljub temu pa mnoge metode strojnega učenja zahtevajo strukturirane podatke [28,29].

Pri pisanju magistrske naloge smo imeli opravka s strukturiranimi podatki v obliki teksta, shranjenega v bazi podatkov. Celice take baze podatkov zapolnjujejo različne vrednosti, ki jih lahko uvrščamo med numerične ali kategorične podatke.

4.1.1 Numerični podatki

Numerični ali kvantitativni podatki so tisti, ki so merljivi. Na njih lahko izvajamo matematične operacije, kot sta seštevanje ter množenje, podatki pa bodo ohranjali numerični pomen. Naprej jih delimo na zvezne in diskretne.

Diskretni podatki predstavljajo nekaj, kar lahko štejemo. Primeri so lahko število izdelkov, število zaposlenih, število barv in tako naprej. Število mogočih vrednosti je lahko omejeno ali neomejeno. Pri 10 metih kocke je število primerov, ko pade 6, omejeno z 0 in 10. Število potrebnih metov, da pade število 6 tisočkrat, pa je navzgor neomejeno.

Zvezni podatki predstavljajo meritve. Primeri so temperatura, dolžina, teža in drugi. Njihove vrednosti ne moremo prešteti, lahko pa so prav tako omejene in neomejene. Volumen goriva, natočen v avto, je zvezni podatek. Če je velikost rezervoarja 50 litrov, je ta omejen med 0 in 50 litri. Vrednosti, ki pa jih lahko volumen zavzame med 0 in 50, pa je neskončno mnogo [30].

4.1.2 Kategorični podatki

Kategorični ali opisni podatki predstavljajo karakteristike na podlagi kategorij. Primeri so spol, državljanstvo in najljubše sadje. Takim podatkom lahko pripišemo numerične vrednosti, a te nimajo matematičnega pomena. Če jabolku pripišemo vrednost 1, banani 2, malini pa 3, vrednost 2,7 ne bi imela smisla. Nakazovala bi večjo podobnost malini kot jabolku, čeprav taka vrednost sploh ne more obstajati.

Ordinalni podatki so posebna vrsta kategoričnih podatkov. Tudi tu podatke ločimo v kategorije, a imajo vrednosti, ki predstavljajo kategorije, določen pomen. Če lahko kakovost izdelka ocenimo s slab, povprečen in dober (oziroma 1, 2 in 3), vidimo, da so kategorije povezane. Povprečna vrednost 100 ocen bo imela smisel, kar za kategorične podatke v klasičnem smislu ne velja [30].

4.1.3 Vrste podatkov in strojno učenje

Različne metode strojnega učenja za delovanje zahtevajo različne podatke. Nekatere potrebujejo kategorične vrednosti, druge pa numerične. Pri napovedovanju s pomočjo strojnega učenja je izbira metode odvisna od vrste napovedane spremenljivke na izhodu. Kadar napovedujemo numerične vrednosti, uporabljamo regresijske metode, ko napovedujemo kategorične vrednosti, pa uporabljamo klasifikacijske metode. Vrsta vhodnih podatkov je za določene metode strojnega učenja poljubna, včasih pa je določeno vrsto podatkov treba zagotoviti. To predstavlja težavo, saj na vrsto podatkov pogosto nimamo vpliva. Potrebne so torej metode za pretvarjanje iz numeričnih v kategorične podatke in obratno. V nadaljevanju je predstavljenih nekaj pogosto uporabljenih metod za tako transformacijo.

Celoštevilsko kodiranje vsaki kategoriji pripiše celo število. Rdeči barvi na primer pripišemo 1, modri 2 in zeleni 3. Lastnost celih števil je, da so urejena zaporedno, kar med njimi izkazuje relacije. Število 1 je bližje številu 2 kot številu 6. Take odnose med števili lahko nekatere metode strojnega učenja razumejo in jih izkoristijo. To je koristno za kategorije ordinalnih podatkov, kot so slab, povprečen in dober, ki so prav tako med seboj povezane. Manj pa je primerno za nepovezane kategorije, kot so prej omenjene jabolko, banana in malina [31].

One-hot kodiranje uporabimo, ko ni opaznih relacij med kategorijami in si nekatere med seboj niso nič bolj podobne kot druge. Deluje tako, da za vsako kategorijo ustvarimo vektor. V polja tega vektorja nato vpisujemo binarne vrednosti. Vrednost 1 je v polje zapisana, ko to predstavlja kategorijo, v kateri leži. Vrednost 0 pa je pripisana vsem ostalim poljem. Slika 4.1 prikazuje one-hot kodiran primer za rdečo, modro in rumeno barvo [31].

	rdeč	moder	rumen
1. primer	[1,	0,	0]
2. primer	[0,	1,	0]
3. primer	[0,	0,	1]

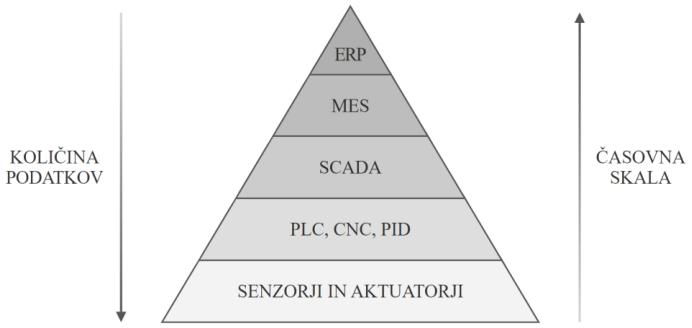
Slika 4.1: One-hot kodiranje

Združevanje v enake bloke je ena najenostavnejših metod diskretizacije podatkov. Opazovane numerične podatke razdeli v izbrano število intervalov enake velikosti. Take kategorije moramo ustvariti za vsak atribut posebej. Podatke o ocenah kakovosti izdelkov, katerih vrednosti so med 0 in 5, lahko razdelimo v kategorije slab, povprečen in dober. Oceni 0 in 1 preslikamo v kategorijo slab, oceni 2 in 3 v povprečen ter oceni 4 in 5 v kategorijo dober [32].

Metoda enake frekvence diskretizira podatke tako, da je v vsaki kategoriji enako število primerov. Če imamo n primerov in jih želimo razdeliti v k intervalov, potem bo v vsakem intervalu n/k primerov. Problem lahko predstavljajo zadnji primeri v podatkih. Te lahko povzročijo, da n/k ni celo število, kar pomeni, da vsi intervali ne bodo imeli enakega števila primerov [33].

4.2 Podatki v proizvodnih podjetjih

V proizvodnih podjetjih imajo podatki pomembno vlogo, zato je pravilno vodenje le-teh na vseh nivojih podjetja velikega pomena. Potrebni so zanesljivi in točni podatki, ki so dosegljivi kjerkoli in kadarkoli. Predstavljeni morajo biti pravemu osebju, v pravem trenutku in na pravi način. S tem namenom so v modernih tovarnah implementirani informacijski sistemi, ki zaposlenim nudijo pomoč in podporo pri odločanju [34]. Te sisteme pogosto predstavimo v avtomatizacijski piramidi, prikazani na sliki 4.2.



Slika 4.2: Avtomatizacijska piramida [34]

ERP oziroma poslovno transakcijski informacijski sistem (angl. *enterprise resource planning*) najdemo na vrhu piramide. Je informacijski sistem, ki ga uporabljajo vodilni in nosilci odločanja v podjetju. V podjetju sledi mnogim stvarem, kot so denarni kapital, materiali, projekti, delovne naloge, naročila, plačila, proizvodne zmogljivosti in mnoge druge. Časovna skala, na kateri ta sistem deluje, je najdaljša od vseh v piramidi (od dneva pa do tednov in mesecev). Pogosto uporabljeni ERP-sistemi so SAP, Navision in Oracle [35][36].

MES oziroma proizvodni informacijski sistem (angl. *manufacturing execution system*) je namenjen spremljanju dogajanja na nivoju proizvodnje v realnem času in pomoči pri odločanju o optimizaciji proizvodnega procesa. Predstavlja povezavo med ERP in SCADA sistemoma. Izkaže se namreč, da ERP-sistemi področja proizvajanja ne pokrivajo dovolj dobro. Kljub temu pa MES potrebuje pametne podsisteme, s katerimi se povezuje, komunicira in mu zagotavljajo podatke. Njegove funkcije so med drugim razporejanje operacij, sledenje projektom in delovnim nalogom, načrtovanje proizvodnih virov in njihov nadzor, kontrola kakovosti ter planiranje vzdrževanja [34,36].

SCADA oziroma informacijski sistem za nadzor in zbiranje podatkov (angl. *supervisory control and data acquisition*) je zadolžen za oddaljeno dostopanje do podatkov iz različnih lokalnih kontrolnih modulov (moduli so lahko od različnih proizvajalcev, saj povezovanje poteka po standardiziranem protokolu). Zbiranje podatkov iz različnih virov poteka preko komunikacijskega omrežja, zato lahko veliki SCADA-sistemi spominjajo na distribuirane kontrolne sisteme. Kontrolira lahko več lokacij preko velikih razdalj. Zbrane podatke procesira in vrača preproste ukaze, kot sta zagon ali zaustavitev naprave. SCADA je primerna za odločitve nižjega nivoja, ne zagotavlja pa holističnega pogleda na proizvodnjo, kar bi pomagalo pri odločitvah vodilnih uslužbencev [34,36].

PLC, **CNC** in **PID** predstavljajo člene na nivoju krmilnikov. Na tem nivoju gre za nadzor in krmiljenje posameznih senzorjev in aktuatorjev. V višjih nivojih smo govorili bolj o informacijskih sistemih, tu pa imamo opravka z veliko strojne opreme. PLC ali programabilni logični krmilnik je robusten industrijski kontroler, ki temelji na podobni elektroniki, kot jo najdemo v računalnikih. Neprestano spremlja stanje vhodnih naprav in se odloča na podlagi predpisanega programa. Podobno deluje tudi CNC, glavna razlika je, da je PLC izdelan za širok spekter aplikacij, CNC pa je bolj namenski. Najpogosteje se CNC uporablja za krmiljenje več osnih naprav. PID krmili s pomočjo povratne zanke, ki neprestano spremlja napako oziroma razliko med dejanskim in želenim stanjem vhoda.

Sestavljajo ga proporcionalni (P), integralni (I) in diferencialni del (D), kar mu daje tudi ime.

Senzorji in **aktuatorji** predstavljajo spodnji nivo avtomatizacijske piramide. Na tem nivoju prihaja do generacije podatkov ter izvedbe ukazov na delovnih sistemih. Senzorji oziroma merilna zaznavala spremljajo spremembe v njihovi okolici in informacije o tem pošiljajo drugim komponentam. Aktuatorji pa na podlagi vhodnih signalov iz drugih komponent izvajajo premikanje sistema. Časovna skala, na kateri se te operacije izvajajo, je majhna. Ponavadi govorimo o mikrosekundah, včasih pa celo še manj [34,37].

Poleg naštetih potrebujejo proizvodna podjetja še mnogo drugih informacijskih sistemov, ki so v pomoč pri delu in odločanju. To so med drugim še razvojno inženirski (CAD/CAM, CAE), dokumentacijski (PDM oziroma PLM) in pisarniško administrativni informacijski sistemi (na primer Microsoft Office) [34].

4.3 Podatki podjetja Litostroj Power

Podatki, uporabljeni pri izdelavi te magistrske naloge, so pridobljeni v podjetju Litostroj Power. Gre za družbo, ki ima sedež v Ljubljani in zaposluje približno 400 ljudi. Proizvajajo predvsem vodne turbine, črpalke ter industrijsko in preoblikovalno opremo [38]. Njihovo proizvodnjo delimo na varilnico, pločevinarno in obdelave. Implementiranih imajo več informacijskih sistemov, ki jim omogočajo spremljanje, zajemanje in analizo podatkov [35]. To so SAP, MikroMES, LiMES in SCADA, ki so v nadaljevanju podrobneje opisani. Podatke iz teh sistemov smo dobili v obliki .sql datoteke. Ta baza podatkov vsebuje 52 tabel, izvoženih iz teh sistemov, za obdobje od leta 2009 do leta 2011.

4.3.1 Informacijski sistemi v Litostroj Power

SAP

SAP je eden od največjih ponudnikov sistemov ERP [39]. Ta sistem vsebuje podatke o planiranju proizvodnje, kot so podatki o materialih, operacijah, delovnih nalogih in delovnih strukturah (WBS) [35]. Spodaj so opisane tabele iz sistema SAP, ki so bile vključene v našo bazo podatkov:

- SAP_Nalogi vsebuje seznam identifikacijskih številk nalogov, materialov, informacije o vrsti naloga, količini, datumu začetka in konca, statusu, spremembah ter druge pomembne informacije, povezane z delovnimi nalogi.
- SAP_Nalogi_Projekt vsebuje seznam identifikacijskih kod projektov ter pripadajočih nazivov.
- SAP_Nalogi_WBS vsebuje kode in nazive WBS ter pripadajoče projekte.
- SAP_Operacije vsebuje seznam unikatnih številk operacij, pripadajoče delovno mesto, identifikacijsko številko naloga, datume planiranega začetka in konca, dolg opis in druge informacije, povezane z operacijami.
- SAP_Operacije_Tekst vsebuje še bolj podroben opis posameznih operacij.
- SAP_Materiali vsebuje identifikacijske številke materialov ter njihov naziv.

LiMES

LiMES oziroma Litostroj Manufacturing Execution Sysem je bil razvit skupaj z laboratorijem LAKOS iz Fakultete za strojništvo na Univerzi v Ljubljani. Sistem je prilagojen individualni in maloserijski proizvodnji, zato je bil testno uveden tudi v podjetju CIMOS. Na njem poteka priprava proizvodnih informacij, daje podporo pri načrtovanju operacij, sprotno zajema podatke o dogodkih, stanju naprav in parametrih procesa. Omogoča odločanje na podlagi ažurnih podatkov, saj spremljanje poteka na nivoju milisekund ob vsaki spremembi dogodka [34,35]. V našo bazo podatkov so bile iz sistema LiMES vključene naslednje tabele, ki pokrivajo le obdelave:

- Del_Mesta_Sled vsebuje identifikacijsko številko in ime delovnega mesta, status, različne datume, trajanja itd.
- Delovna_Mesta vsebuje ime delovnega mesta, opis, stroškovno mesto, obrat, naziv delavnice itd.
- Statusi_DM vsebuje seznam mogočih statusov delovnih mest, njihovo vrsto in opis.
- Operacije_Sled vsebuje identifikacijsko številko operacije, pripadajoče delovno mesto, datume začetka in konca, status itd.
- SAP_Operacije je delna kopija tabele iz sistema SAP, od koder so prepisane planirane operacije v LiMES zaradi prenosa na panele [35].
- Zastoji vsebuje identifikacijske številke različnih vrst zastojev in pripadajoče opise.
- Del_Mesta_Zastoji vsebuje delovna mesta ter zastoje, ki so na določenem delovnem mestu mogoči.
- Dogodki povezuje identifikacijsko številko z nazivom za dogodke iz tabele Operacije_Sled.

MikroMES

MikroMES za razliko od LiMES-a pokriva pločevinarno in varilnico. V obdobju, na katero se nanašajo naši podatki, je bil pri obdelavah implementiran le na petih delovnih mestih. Ta sistem spremlja aktivnosti in operacije manjšega obsega, ne pa tudi prekinitev. Njegova omejitev je tudi ta, da spremljanje poteka na nivoju ur in dni. To pomeni, da je ocena časov približna in ne zagotavlja časovne natančnosti [35]. Tabele, ki so bile vključene v našo bazo podatkov, so:

- uMES_Materialni_Premiki vsebuje identifikacijske številke premikov in pripadajočih materialov, njihove šarze, količine, datume vnosa in premika itd.
- uMES_Potrditve vsebuje kadrovske številke, številke nalogov, operacij, njihovo trajanje ter datum vnosa.
- uMES_Procesi vsebuje oznako procesa in njegov opis.
- uMES_Procesi_Sm povezuje procese s stroškovnimi mesti.
- uMES_Rezijska_Dela vsebuje šifre del, njihove opise ter oznako pripadajočega procesa.
- uMES_Rezijske_Ure vsebuje kadrovske številke, šifre dela, datume vnose in trajanja.

SCADA

V podjetju imajo implementiran tudi sistem SCADA [39]. Podatkov iz tega sistema praktično nimamo. V bazi podatkov najdemo le tabelo SCADA_Stroji, ki povezuje oznake delovnih mest s stroji SCADA ter informacijami o starih in novih panelih.

V podjetju sicer uporabljajo še mnoge druge informacijske sisteme. To so sistemi za razvoj proizvodov (AutoCAD, Pro/Engineer, WSCAD itd.), sistem za evidenco delovnega časa (Špica), Windows Server, elektronski sistem za bančno poslovanje, sistem za izračun potnih nalogov (Access), pisarniško administrativni sistemi (Microsoft Office) in razvojni programski sistemi (Access, Oracle itd.) [39].

4.3.2 Uporabljeni podatki

Cilj magistrske naloge je napovedovanje manjkajočih informacij o trajanju operacij. Informacije o trajanju operacij najdemo v tabeli Operacije, kjer imamo zapise o časovni točki začetka in konca operacij. Na sliki 4.3 je prikazan del tabele Operacije, kjer opazimo, da tabela ni popolna.

StPotrditve	DelovnoMesto	StNaloga	DatumZacetkaDela	DatumKoncaDela	StatusOperacije
1841411	[illegible]	1260198	2010-01-14 12:42:29.960000	2010-01-14 12:42:29.960000	IZP LANS PKKD
1841426	[illegible]	1260198	2010-02-17 16:13:12.353000	2010-02-18 01:58:26.127000	IZP LANS VPLA
1841427	[illegible]	1260198	2010-02-18 12:10:34.763000	NULL	IZP LANS
1841428	[illegible]	1260198	2010-02-21 07:02:25.127000	2010-02-23 19:21:38.183000	IZP LANS VPLA
1841430	[illegible]	1260198	2010-02-24 09:33:28.833000	2010-02-24 09:33:28.833000	IZP LANS PKKD
1841433	[illegible]	1260198	2010-02-26 11:08:46.177000	2010-02-26 14:54:19.520000	IZP LANS VPLA
1841434	[illegible]	1260198	2010-03-01 13:54:48.410000	2010-03-02 17:20:13.387000	IZP LANS VPLA
1841435	[illegible]	1260198	2010-03-02 12:11:11.720000	NULL	IZP LANS
1841437	[illegible]	1260198	2010-03-04 10:55:30.780000	NULL	IZP LANS
1841438	[illegible]	1260198	2010-03-04 10:55:05.647000	NULL	IZP LANS

Slika 4.3: Del podatkov iz tabele Operacije

V tabeli na sliki 4.3 je zapisanih več različnih podatkov. To so številski podatki, tekstovni podatki in datumi. Vrstice tabele predstavljajo posamezne operacije. Temu pravimo tudi primeri oziroma podatkovni zapisi. Stolpci predstavljajo različne atribute podatkov, čemur lahko rečemo tudi polja. V zgornjem primeru so to StPotrditve, DelovnoMesto, StNaloga, DatumZacetkaDela, DatumKoncaDela in StatusOperacije. Celice, ki nimajo predpisane vrednosti, so označene z NULL. Te smo želeli zapolniti z napovedmi. V ta namen pa smo potrebovali podatke iz več tabel. Uporabili smo tabele Operacije, Operacije_Sled, Zastoji in Delovna_Mesta.

Tabela Operacije

Tabela vsebuje 38 atributov in 53827 primerov. Podatki so združeni iz sistemov SAP in LiMES. Pri našem delu nismo potrebovali vseh atributov. Na sliki 4.4 je prikazanih 6 atributov, ki smo jih pri delu uporabljali. Opisani so spodaj:

- StPotrditve vsebuje unikatno številko posamezne operacije. Te številke smo uporabili za identifikacijo posamezne operacije ter pravilno pripisovanje podatkov iz drugih tabel.
- DelovnoMesto vsebuje naziv delovnega mesta, na katerem je bila operacija izvedena oziroma se je na njem izvajala.
- PlDatumZacetkaOperacije in PlDatumKoncaOperacije govorita o planiranih začetkih in koncih operacije. Ti podatki so kopirani iz sistema SAP (oziroma tabele

SAP_Operacije) in so vedno le na dan natančni. V SAP-u je sicer časovna natančnost večja, vendar so bile bolj natančne informacije pri prenosu odrezane.

- DatumZacetkaDela in DatumKoncaDela podajata informacije o dejanskem začetku in koncu dela na posamezni operaciji. Zavedeni podatki so natančni do milisekunde, saj so sem preneseni iz sistema LiMES. Vrednost DatumKoncaDela je zavedena za 17428 od 53827 operacij. Razlogov za to je več. Operater je bil površen oziroma ga je nekaj med delom zmotilo in zaključka operacije ni zavedel. Operater operacije dejansko ni zaključil. Operacija je bila zaključena, vendar se je ta informacija pri prenosu izgubila. Operacija ostane nezaključena, ker se je zaključila na drugem delovnem sistemu, prvo delovno mesto pa se ukine. Projekt se je prekinil in ostal sredi zastoja, konec zastoja pa je izven časovnega okvirja podatkov.

StPotrditve	DelovnoMesto	PlDatumZacetkaOperacije	PlDatumKoncaOperacije	DatumZacetkaDela	DatumKoncaDela
2257895	[illegible]	2010-04-29 00:00:00.000000	2010-04-29 00:00:00.000000	2010-01-29 12:48:20.263000	2010-01-29 12:48:40.590000
2257899	[illegible]	2010-10-22 00:00:00.000000	2010-10-22 00:00:00.000000	2010-11-03 12:35:48.673000	NULL
2257901	[illegible]	2010-10-25 00:00:00.000000	2010-10-25 00:00:00.000000	2010-11-03 12:36:17.307000	NULL
2257926	[illegible]	2010-10-19 00:00:00.000000	2010-10-19 00:00:00.000000	2010-10-20 12:42:34.100000	NULL
2257929	[illegible]	2010-09-08 00:00:00.000000	2010-09-09 00:00:00.000000	2010-10-18 20:58:27.080000	2010-10-18 20:58:33.810000
2257930	[illegible]	2009-11-20 00:00:00.000000	2009-11-20 00:00:00.000000	2010-01-29 12:46:45.827000	2010-01-29 12:46:56.170000
2257934	[illegible]	2010-10-22 00:00:00.000000	2010-10-22 00:00:00.000000	2010-11-03 12:34:38.810000	NULL

Slika 4.4: Uporabljeni del tabele Operacije

Tabela Operacije_Sled

Je tabela, katere podatki izhajajo iz sistema LiMES. Vsebuje 34 atributov in 48389 primerov. Tu vsak primer ne predstavlja svoje operacije. Različnih operacij je 14061, te pa se lahko pojavijo v več vrsticah. Del te tabele, ki smo ga uporabili pri nadaljnjem delu, je prikazan na sliki 4.5. Iz celotne tabele Operacije_Sled smo uporabili 9 atributov, ki so opisani spodaj:

- StPotrditve opisuje enako kot StPotrditve v tabeli Operacije.
- DelMesto opisuje enako kot DelovnoMesto v tabeli Operacije.
- DatumOperStart in DatumOperEnd opisujeta enako kot DatumZacetkaDela in DatumKoncaDela v tabeli Operacije. Vse operacije v tej tabeli imajo zabeležen njihov konec.
- Zastoj vsebuje kodo zastoja v primeru, da je do zastoja prišlo. V nasprotnem primeru ni zabeleženo nič in vidimo označbo NULL.
- Trajanje vsebuje razliko med začetkom zastoja in nekim drugim delom. Opisuje trajanje do spremembe na operaciji. Informacije so na uro natančne.
- TrajanjeHHMISS predstavlja enako kot Trajanje, le da je najdaljše trajanje 24 ur, nato se ponastavi na nič. Informacije v tem stolpcu so na sekundo natančne.
- TrajanjeZastoj vsebuje trajanje v urah, ko se določena operacija ni izvajala. Kot pri stolpcu Zastoj tudi tukaj najdemo NULL v primeru, da zastoja ni.
- DatumZastojEnd vsebuje informacijo o tem, kdaj se je zastoj končal. Celica z NULL pomeni enako kot pri prejšnjih atributih.

StPotrditve	DelMesto	DatumOperStart	DatumOperEnd	Zastoj	Trajanje	TrajanjeHHMISS	TrajanjeZastoj	DatumZastojEnd
3407112	[redacted]	2010-01-19 06:09:25.967000	2010-01-20 06:32:34.260000	1	24	00:23:08	48	2010-01-21 06:11:38.877000
3407112	[redacted]	2010-01-21 06:11:38.877000	2010-01-21 06:28:48.707000	NULL	0	00:17:09	NULL	NULL
3407138	[redacted]	2010-01-04 11:24:14.927000	2010-01-04 14:37:06.357000	NULL	3	03:12:51	NULL	NULL
3407138	[redacted]	2010-01-04 21:55:35.840000	2010-01-04 21:55:49.377000	NULL	0	00:00:13	NULL	NULL
3407138	[redacted]	2010-01-04 21:55:49.377000	2010-01-04 21:55:56.443000	178	0	00:00:07	28	2010-01-06 02:00:43.707000
3407138	[redacted]	2010-01-06 02:00:43.707000	2010-01-06 05:58:11.747000	NULL	4	03:57:28	NULL	NULL
3407138	[redacted]	2010-01-06 05:58:22.790000	2010-01-06 13:57:29.353000	NULL	8	07:59:06	NULL	NULL

Slika 4.5: Uporabljeni del tabele Operacije_Sled

Zastoji

Tabela Zastoji iz sistema LiMES vsebuje 6 stolpcev in 294 vrstic. Nam sta koristila le 2 stolpca, prikazana na sliki 4.6. V stolpcu Zastoj3 so definirane kode zastojev, v stolpcu OpisZ3 pa pripadajoči opisi zastojev.

Zastoj3	OpisZ3
13	Dodatno čiščenje delovnega mesta
155	Menjava oljnega filtra
156	Menjava zračnega filtra
157	Ostrenje specialnega orodja
160	Nepopolna tehnološka dokumentacija
161	Neustrezen CNC program

Slika 4.6: Uporabljeni del tabele Zastoji

Delovna_Mesta

Tabela Delovna_Mesta je pridobljena iz sistema LiMES in vsebuje seznam vseh delovnih mest. Število vrstic te tabele je 352, vsaka pa ima definiranih 13 atributov. Nas so zanimali le 4 atributi, prikazani na sliki 4.7 in opisani spodaj:

- DelovnoMesto vsebuje unikatno oznako delovnega mesta.
- DelovnoMestoNad vsebuje unikatno oznako nadrejenega delovnega mesta, če ta obstaja.
- Vrsta vsebuje informacijo o vrsti delovnega mesta.
- Opis vsebuje podrobne opise delovnega mesta.

DelovnoMesto	DelovnoMestoNad	Vrsta	Opis
[redacted]	[redacted]	0002	RAZREZ PALIČASTEGA MATERIALA
[redacted]	[redacted]	0002	PLOČEVINARNA - TOPLOTNA OBDELAVA
[redacted]	NULL	0002	OBDELAVA
[redacted]	[redacted]	0002	TEŽKA OBDELAVA
[redacted]	[redacted]	0002	TEŽKA OBDELAVA - TEHNOLOŠKI CENTER
[redacted]	[redacted]	0002	TEŽKA OBDELAVA - STRUŽENJE
[redacted]	[redacted]	0002	TEŽKA OBDELAVA - VRTANJE IN REZKANJE

Slika 4.7: Uporabljeni del tabele Delovna_Mesta

5 Strojno učenje

Eksperimentalno delo pri tej magistrski nalogi zajema strojno učenje. V poglavju *3.2 Strojno učenje z nevronskimi mrežami* je na sliki 3.21 prikazano, kaj vse to delo zajema in kako poteka. V tem poglavju pa bo predstavljeno, kako smo opisane korake izvedli z namenom zmanjševanja nepopolnosti podatkov. Opisani bodo tudi vplivi na ta proces.

Za napovedovanje manjkajočih vrednosti smo izbrali nevronske mreže. Te v zadnjih letih na mnogo področjih doživljajo razcvet, saj njihovo napovedovanje dosega zelo dobre eksperimentalne rezultate. V delu Garciarena et al. [7] ugotavljajo, da dajejo kompleksni modeli za napovedovanje manjkajočih vrednosti boljše rezultate od enostavnih. Ne le da kompleksni modeli manjkajoče vrednosti bolj točno napovejo, podatki, dopolnjeni s kompleksnimi modeli, dajejo boljše rezultate pri nadaljnjih analizah. V kategorijo kompleksnih modelov lahko umestimo tudi nevronske mreže, saj so sposobne zaznati izjemno zapletena razmerja in vzorce v podatkih. V delu Nishanth et al. [3] so raziskovalci ugotovili celo, da se nevronske mreže dobro obnesejo za napovedovanje manjkajočih kategoričnih vrednosti, in to še posebej, kadar je delež manjkajočih vrednosti velik. V naših podatkih o času izvajanja operacij je manjkajočih vrednosti približno dve tretjini, zato menimo, da bi bile lahko nevronske mreže dobra izbira.

5.1 Vplivi na strojno učenje

Na že omenjeni sliki 3.21 najdemo 4 faktorje, ki vplivajo na potek strojnega učenja. To so podatki, uporabnik, domena in zgradba sistema. V nadaljevanju so opisani vplivi omenjenih faktorjev na naš proces.

Podatki, ki smo jih uporabljali, vsebujejo 53827 vrstic, vsaka izmed njih opisuje eno operacijo. Od tega je 17428 primerov takih, kjer čas izvedbe operacije poznamo in 36399 takih, kjer časa izvedbe ne poznamo. Omenjen je že vpliv velikega deleža manjkajočih podatkov na izbiro metode za napovedovanje. Poleg tega so vrste podatkov v tabelah močno vplivale na način njihove predobdelave. Tudi dejstvo, da so bile informacije o operacijah razpršene v več tabelah, je na predobdelavo močno vplivalo.

Uporabniki teh podatkov so lahko zaposleni v podjetju Litostroj kot tudi raziskovalci, ki bodo v prihodnosti na njih izvajali različne analize. V proces odločanja pri tem procesu niso bili vključeni. Kljub tem smo bili pri specifikah napovedovanja zelo pozorni na uporabnost podatkov za nadaljnje analize (več o tem v poglavju *5.2 Opis problema*).

Domena in njeno poznavanje nam je močno koristila pri razumevanju podatkov. Različna trajanja, lastnosti delovnih mest in druga znanja o proizvodnem procesu bi nam bila sicer nedosegljiva. Veliko pomoč je nudila tudi pri generaciji značilk. Lažje smo postavljali hipoteze o tem, kakšne značilke signifikantno vplivajo na čas izvedbe operacije.

Sistemska zgradba vpliva na izbiro vrste nevronske mreže, njeno kompleksnost in način iskanja optimalnih hiperparametrov. Naš sistem predstavlja osebni računalnik z Intel Core i7 2670QM/2.2 GHz procesorjem, NVIDIA GeForce GT 555M grafično kartico, 8 GB RAM polnilnika in teče na 64-bitni različici operacijskega sistema Windows 10. Osebni računalnik za gradnjo in učenje najbolj zapletenih in računsko potratnih nevronskih mrež ni primeren. Enak problem predstavljajo kompleksne metode optimizacije hiperparametrov. Delovanje na grafični kartici je bilo sicer vzpostavljeno, a ta ne podpira knjižnice cuDNN, zaradi česar pogosto delovanje ni bistveno hitrejše od delovanja na procesorju. Na sistemu so za izvajanje strojnega učenja nameščeni MySQL, Python, Anaconda, Theano, Keras in PyCharm.

5.2 Opis problema

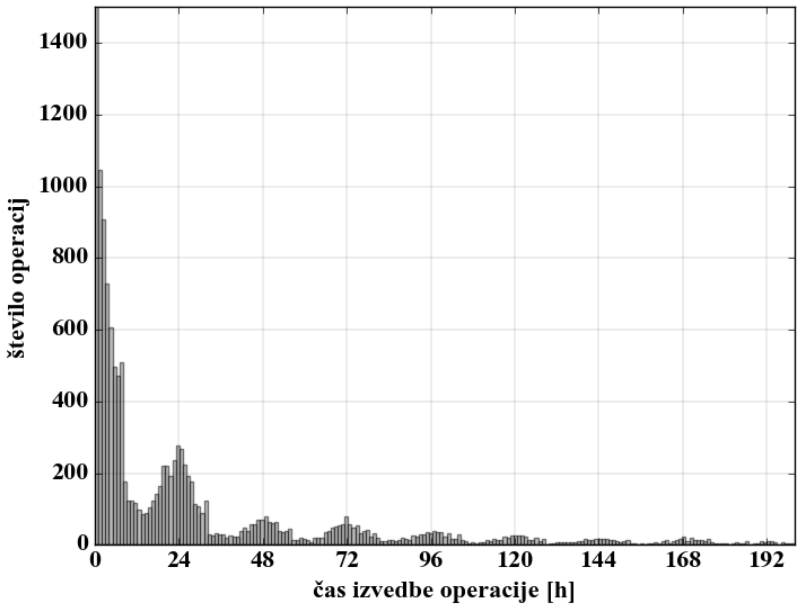
Težava uporabljenih proizvodnih podatkov je njihova nepopolnost. Konkretno nas zanimajo manjkajoče informacije o časovni točki (datumu, uri, minuti, sekundi in mikrosekundi) zaključka izvedbe operacije. Napovedovanje časovne točke je težavno zaradi oblike, v kakršni nastopa. Napovedovati moramo nekaj, česar oblika je za nevronske mreže bolj primerna. Dve izmed možnosti sta napovedovanje UNIX časovne oznake konca izvedbe operacije in napovedovanje trajanja izvedbe operacije. Odločili smo se za slednjo. Pri napovedovanju želimo čim bolj točne napovedi posameznih trajanj in tudi čim večjo podobnost porazdelitev dejanskih in napovedanih trajanj.

Da lahko ustrežemo meriloma točnosti posameznih trajanj in podobnosti porazdelitev, moramo biti pozorni na vnos pristranskosti v podatke. Za ponazoritev si poglejmo trajanja, katerih vrednost je 0. V naših podatkih je takih trajanj veliko. Jasno je, da trajanje operacije ne more biti 0. Kljub temu poznavanju pa naš cilj ni napovedati dejanskega trajanja, katerega vrednost je lahko majhna, a ne ničelna. Če bi napovedali dejansko trajanje, katerega vrednost nikoli ne bi zavzela 0, bi v podatke vnašali pristranskost. Naš cilj je preslikati vzorec iz podatkov na napovedane vrednosti, čeprav vemo, da take vrednosti v resnici ne morejo obstajati. S tem razlogom je veliko razmisleka posvečenega čiščenju, filtriranju in obdelavi podatkov pred njihovo uporabo.

5.3 Zajemanje, razumevanje in priprava podatkov

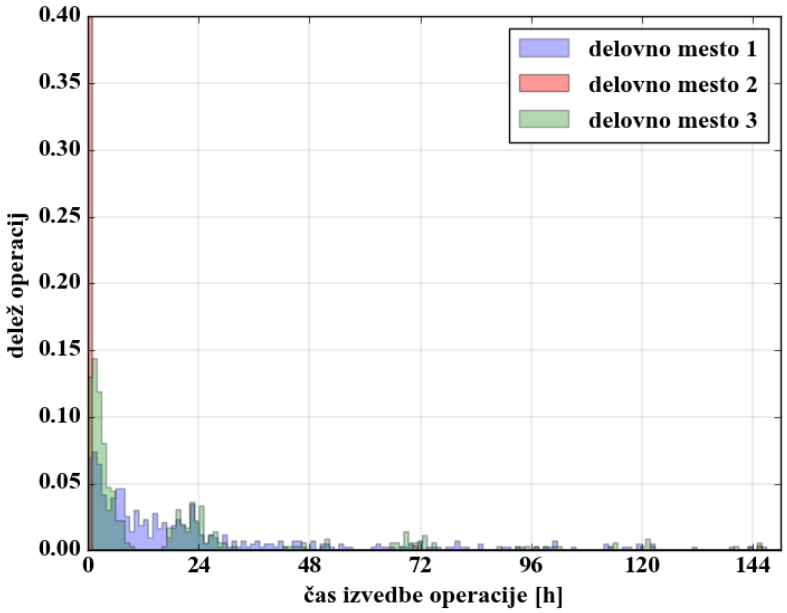
V strogem pomenu ni nujno, da zajemanje, razumevanje in pripravo podatkov uvrščamo med strojno učenje (glejte sliko 3.21). Vseeno pa si poglejmo zadnja dva koraka, saj sta bila pri našem delu pomembna. Zajemanje podatkov je bilo opravljeno s strani podjetja Litostroj Power, zato ta korak v tej nalogi ne bo podrobno opisan.

O podatkih je veliko napisanega že v poglavjih *4.3 Podatki podjetja Litostroj Power* in *5.1 Vplivi na strojno učenje*. Za učenje na podatkih imamo na voljo 17428 primerov, od skupaj 53827 primerov. Za **razumevanje** je najbolj smiselno podatke vizualizirati. To smo naredili s pomočjo JetBrains PyCharm okolja, kamor smo pisali v Python jeziku. Najprej nas je zanimala porazdelitev trajanj vseh operacij, kar prikazuje slika 5.1. Opazimo lahko, da gre skrajno levi stolpec izven meja grafa. Izkaže se, da je v tem stolpcu, ki predstavlja trajanja od 0 ur do 1 ure, približno tretjino vseh operacij. Tudi najdaljša trajanja na sliki 5.1 niso vidna, saj je x-os prekinjena pri 200 urah. Take operacije trajajo nekaj 1000 ur, pri čemer je najdaljše trajanje 10776 ur oziroma nekaj več kot leto dni.



Slika 5.1: Histogram trajanj za izvedbo operacij

Porazdelitev trajanj, ki smo jih pričakovali, je bila popačena normalna porazdelitev, a temu ni tako. Porazdelitev, prikazano na sliki 5.1, si lahko predstavljamo na dva načina. Lahko jo gledamo kot nihajočo eksponentno porazdelitev ali pa porazdelitev, sestavljeno iz več normalnih porazdelitev. Na na sliki 5.1 so trajanja označena vsakih 24 ur, kar kaže na eno izmed značilnosti proizvodnje. Izkaže se, da je v določenih delih dneva opravljenih več operacij kot v drugih delih dneva. Razlog za to je lahko večje število zaposlenih na delu, večja produktivnost delavcev v določenih izmenah, razporeditev delavcev na svoja delovna mesta in podobno. Hipoteza o nihanju eksponentne krivulje se torej zdi smiselna. Hipoteza o sestavljenih normalnih porazdelitvah pa je izhajala iz ideje, da imajo posamezna delovna mesta normalno porazdelitev trajanj, a jo lahko zavržemo. Na to dejstvo kaže slika 5.2.



Slika 5.2: Histogrami trajanj za tri delovna mesta

Na sliki 5.2 vidimo, da tudi posamezna delovna mesta ne sledijo normalni porazdelitvi. Ne le to, njihove porazdelitve so med seboj precej različne. Na delovnem mestu 1 se delež operacij s povečevanjem časa za izvedbo dokaj enakomerno zmanjšuje. Na delovnem mestu 2 vsa trajanja padejo v prvi stolpec, ki vsebuje trajanja od 0 ur do 1. ure. Na delovnem mestu 3 pa vidimo, da delež operacij močno niha v odvisnosti od časa izvedbe operacij. Vse to kaže na bistvene razlike med delovnimi mesti, kar smo lahko v nadaljevanju izkoristili pri generaciji značilk. Nevronski mreži namreč lahko predstavimo dejstvo, da so si delovna mesta med seboj različna v upanju, da bo to znanje pomagalo pri napovedovanju. Na sliki 5.2 so prikazana le 3 delovna mesta, saj v nasprotnem primeru vizualizacija ne bi bila jasna. Slika 5.3 pa prikazuje razlike med 79 delovnimi mesti.

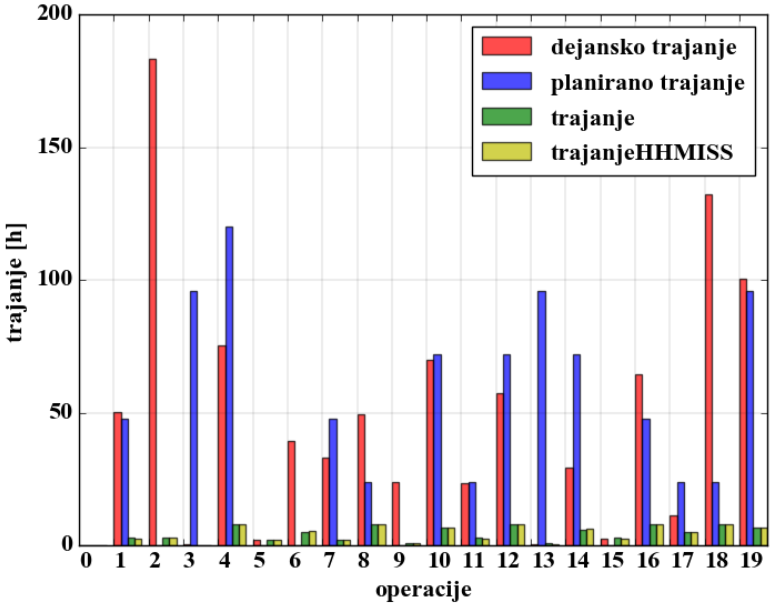
Razlog, da trajanja operacij ne sledijo normalni porazdelitvi, pripisujemo tipu proizvodnje. Litostroj Power namreč deluje na principu maloserijske proizvodnje. Pri njih enakih naročil praktično ni. Temu pravimo tudi »engineer-to-order« proizvodnja, kjer vsakemu naročilu sledi razvoj proizvoda. Menimo, da bo iz tega razloga napovedovanje trajanj težje, saj v podatkih ni tako jasnih vzorcev, kot bi bili na primer v masovni proizvodnji.

Na sliki 5.3 vidimo škatlo z brki za 79 delovnih mest. Vseh delovnih mest je sicer 352, a nekatera izmed njih niso več v uporabi. Izmed teh je 160 delovnih mest takih, da imajo v podatkih zavedeno vsaj eno operacijo. Omenjenih 79 pa je takih, kjer imamo zavedeno vsaj eno operacijo, ki nosi informacijo o času njene izvedbe. Število zavedenih operacij na posameznih delovnih mestih je zelo različno. Pri nekaterih je primerov tako malo, da škatle z brki sploh ni mogoče ustvariti. V tistih vrsticah najdemo le oznake ×, ki sicer predstavljajo izstopajoče točke. Debelejša navpična črta predstavlja mediano, robova škatle predstavljata prvi in tretji kvartil, robova brkov pa predstavljata najkrajše in najdaljše trajanje, ki ju še ne štejemo kot izstopajoči točki. Zaradi zelo različnih trajanj je na horizontalni osi trajanje prikazano v logaritemski skali.

Slika 5.3: Škatle z brki za posamezna delovna mesta

Želeli smo vizualizirati še različna trajanja v podatkih za naključne primere operacij, kar prikazuje slika 5.4. Vsako operacijo predstavljajo štirje stolpci, ki opisujejo različna trajanja. Opazimo lahko, da med dejanskimi in planiranimi trajanji korelacija obstaja. Povezanost med njima ni vidna v vseh primerih (na primer operaciji 2 in 3), v nekaterih primerih pa sta si trajanji zelo podobni (na primer operacije 1, 10, 19). Planirano trajanje je torej prav tako atribut, ki ga je smiselno vzeti za značilko nevronske mreže. Razlog, da ujemanje ni še bolj natančno, je pogosto to, da je planirano trajanje v podatkih le na 24 ur natančno (zavzema lahko torej vrednosti 24 ur, 48 ur, 72 ur in tako dalje). Na drugi strani

pa imamo atributa trajanje in trajanjeHHMISS, ki se zdi, da za značilko ne bosta primerna. Oba vedno zavzameta vrednost med 0 in 24 ur. Take informacije bi nam torej lahko koristile le za operacije, katerih čas izvedbe je krajši od enega dneva. Mi bomo vsa trajanja popisovali z enim modelom, zato nam ta dva atributa ne bosta koristila. Koristna bi bila lahko v primeru, da bi uporabljali različna modela za napovedovanje trajanj do enega dneva in od enega dneva naprej.



Slika 5.4: Primerjava trajanj v podatkih

Za napovedovanje so pomembne tudi informacije o zastojih. Zapisov o zastojih je 45168 in se nanašajo na 14059 različnih operacij. Na eni operaciji je torej pogosto prisotnih več zastojev. Večina informacij o zastojih, ki so v podatkih zabeležene, se nanaša na operacije, katerih čas izvedbe poznamo in ne vsebujejo manjkajočih vrednosti. Pri napovedovanju bo torej večina napovedi trajanj predvidevala, da do zastojev pri teh operacijah ni prišlo. Tu se pokaže, kako neugodni so nepopolni podatki. Mi namreč ne vemo, ali tam do zastojev ni tako pogosto prihajalo ali tam informacije o zastojih manjkajo. Verjetnejša se zdi slednja možnost. Kljub temu smo se odločili, da bomo informacije o zastojih vključili med podatke, ki bodo predstavljali vhode v nevronsko mrežo.

Po procesu razumevanja podatkov smo jih nato **pripravili**. Pri pripravi smo s pomočjo MySQL do podatkov iz različnih tabel dostopali, jih združevali in urejali. Izbrani atributi, združeni v eno tabelo, so bili izvoženi v obliki .csv datoteke. To so StPotrditve, Delovno Mesto, PlDatumZacetkaOperacije, PlDatumKoncaOperacije, DatumZacetkaDela, DatumKoncaDela, Zastoj in DatumZastojEnd.

5.4 Predobdelava

Tudi predobdelava je potekala v programskem okolju JetBrains PyCharm. Razlog za to izbiro je, da je generacija in učenje nevronskih mrež potekalo v istem okolju, zato je bil prehod precej lažji. Uporaba različnih knjižnic, ki jih lahko uporabljamo s programskim jezikom Python, nam je omogočila precejšno poenostavitev dela na predobdelavi podatkov.

5.4.1 Čiščenje podatkov

Pri čiščenju podatkov je bil pomemben del odločitev povezan s filtriranjem podatkov. Pri strojnem učenju namreč pogosto odstranjujemo izolirane točke in nesmiselne vrednosti. Izolirane točke so v našem primeru izjemno dolga trajanja, katerih vrednost presega več 1000 ur. Nesmiselne vrednosti pa so, ko je trajanje enako 0. Izolirane točke so lahko posledica napak pri vnašanju, trajanje z vrednostjo 0 je zagotovo nemogoče. Treba se je bilo odločiti, ali bomo vrstice s takimi primeri iz podatkov odstranili. Pri tem smo se vrnili na definicijo problema. Brisanje teh podatkov oziroma njihovo filtriranje bi skoraj zagotovo povzročilo bolj natančne napovedi časa izvajanja operacij, a to ni naš cilj. Njihovo izločanje bi iz podatkov izbrisalo vzorce, ki jih želimo na napovedi preslikati. To bi povzročilo pristranskost napovedanih vrednosti, kar je za nas nezaželeno. Pridobljeni podatki bodo morda v prihodnosti dodatno analizirani ali uporabljeni pri strojnem učenju, zato filtriranja ne bomo uporabili. Naš cilj je izboljšava kakovosti podatkov in ne napovedovanje točnih časov izvedb operacij.

Pomemben korak pri čiščenju je tudi odpravljanje nepopolnosti v podatkih. S to težavo smo se srečali, čeprav je bil namen našega dela zmanjševanje nepopolnosti v podatkih. Tu se navezujemo predvsem na manjkajoče vrednosti o zastojih. Predvidevamo, da gre za MAR vrsto nepopolnih podatkov. Zdi se namreč, da vzorec, ki mu sledijo manjkajoče vrednosti, lahko najdemo. Velik delež informacij o zastojih je prisoten le za operacije, katerih čas izvedbe poznamo. Mogoče je, da je manjkajoča informacija o zaključku operacije razlog za manjkajočo informacijo o zastoju. Logično bi bilo, da je delež operacij z zastoji podoben za primere, kjer poznamo čas izvajanja operacije in za primere, kjer tega časa ne poznamo. Tega pa z gotovostjo ni mogoče trditi, zato smo se odločili, da podatke pustimo v taki obliki, kot so. V primeru, da bi to želeli preveriti, bi lahko analizirali povprečne čase med okvarami (angl. *mean time between failures*). Preverili bi, na koliko časa na določenem delovnem mestu v povprečju pride do zastoja. Če bi predvidevali, da določenim delovnim mestom zastoji manjkajo, bi jih na podlagi ustvarjenih pravil naključnim operacijam lahko dodajali. S tem skoraj zagotovo ne bi pravilno zadeli trajanja zastoja in tega ne bi pripisali pravi operaciji. Kljub temu pa bi se verjetno porazdelitev napovedi bolj ujemala s porazdelitvijo časov izvajanja operacij, prisotno v podatkih.

Tretji pomembni korak je reševanje nekonsistentnosti. Konsistentnost med tabelami v naših podatkih je bila zagotovljena. Še vedno je bilo združevanje tabel po različnih ključih pogosto problematično, a zaradi konsistentnosti rešljivo. Večinoma smo tabele lahko združevali po unikatni številki, ki definira operacijo. Ta je v tabelah poimenovana StPotrditve.

Iz napisanega je razvidno, da smo bili pri čiščenju podatkov konzervativni. Z namenom, da ne bi vnašali novih pristranskosti, je bilo čiščenju posvečeno veliko razmisleka. Na koncu je pomembno oceniti še, kako reprezentativno učni podatki predstavljajo vse podatke. V tem pogledu sta problematični dve točki. Prva, že omenjena, je v povezavi z zastoji, druga pa v povezavi z delovnimi mesti. O zastojih je povedanega že veliko in zdi se, da podatki o zastojih niso popolnoma reprezentativni. V učnih podatkih, ki zajemajo tretjino primerov, najdemo več kot 95 % vseh zastojev. Vemo tudi, da v učnih podatkih nastopi zastoj na približno vsakih 8 operacij. Iz tega lahko sklepamo, da bodo napovedi za približno 12 % primerov prekratke. Za katere primere to velja, ne znamo oceniti, a se moramo pri rezultatu tega zavedati. Pri drugi točki, povezani z delovnimi mesti, se moramo zavedati, da v učnih podatkih nimajo vsa delovna mesta zavedenega enakega števila operacij. To je razvidno že iz slike 5.3, kjer določena delovna mesta nimajo dovolj podatkov za izris škatle z brki. Eden od načinov reševanja te težave je umetno povečanje števila primerov za določena delovna mesta. Problem te metode je, da imajo nekatera delovna mesta zelo malo primerov, zaradi česar bi jih težko ustrezno umetno razmnožili. Še večja težava pa je, da 81 od 160 delovnih mest nima informacije o dejanskem trajanju operacij, ki na njih potekajo. Za ta delovna mesta ni mogoče umetno povečati števila primerov. Zato smo se odločili, da bomo delovna mesta združevali v skupine podobnih delovnih mest. Ideja je, da bodo v nekaterih skupinah združena delovna mesta z veliko operacijami in tista brez operacij. Tako bomo nevronski mreži zagotovili informacije tudi za delovna mesta brez operacij oziroma povečali količino informacij za delovna mesta z malo primeri. Nevronska mreža se bo posledično lažje naučila, kakšen je vpliv delovnih mest na čas izvedbe operacije.

5.4.2 Transformacija podatkov

Na tej točki naj bi se prvič lotili transformacije podatkov. V teoriji so posamezni atributi do te točke še vedno v obliki, v kakršni so v originalnih podatkih. V praksi pa je proces strojnega učenja iterativen in pogosto skačemo iz enega koraka na drug korak. Tako lahko že v poglavju *5.3 Zajemanje, razumevanje in priprava podatkov* najdemo informacije o časovni točki začetka in konca, pretvorjene v obliko trajanja. Kljub temu povemo, da je bil prvi korak transformacije, ki smo se ga lotili, generacija značilk.

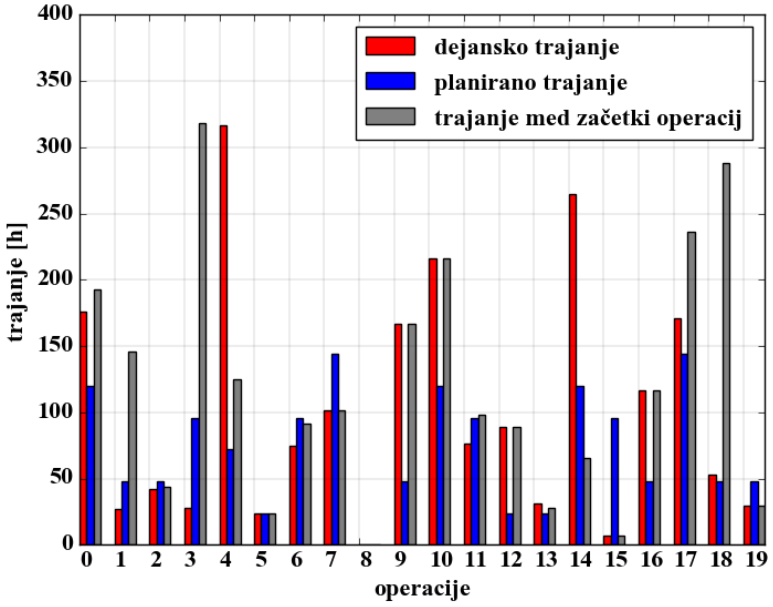
Pri generaciji značilk je treba podatke transformirati v tako obliko, da jih izbrana metoda strojnega učenja lahko čim bolje izkoristi. Nevronske mreže zahtevajo numerične podatke, zato informacije v obliki datumov in kategoričnih vrednosti za njih niso primerne. Spodaj so opisane vse ustvarjene značilke, uporabljene pri učenju nevronske mreže. Pomembno je vedeti, da vnaprej ne vemo, katere vhodne značilke in v kakšni obliki bodo dobro delovale pri napovedovanju. S tem se ukvarjamo, ko je model že zgrajen.

Značilka o planiranem trajanju je izračunana iz atributov PlDatumZacetkaOperacije in PlDatumKoncaOperacije. Pri delu smo si med drugim pomagali s knjižnico datetime. Ta omogoča računanje razlik med časovnimi točkami, pri čemer upošteva različno število dni v mesecu, prestopna leta in podobno.

Značilka o trajanju med začetki operacij je pridobljena iz atributov DatumZačetkaDela in DelovnoMesto. Govori o tem, koliko časa je preteklo od začetka opazovane operacije do začetka naslednje operacije na istem delovnem mestu. Operacije smo najprej razdelili po

delovnih mestih, kjer smo jih nato razvrstili po času začetka dela. Značilka za opazovano operacijo torej predstavlja razliko med časovno točko začetka dela na naslednji operaciji tega delovnega mesta in začetkom dela na opazovani operaciji.

Slika 5.5 prikazuje vrednosti opisanih značilk in dejanskega trajanja naključnih operacij. Na sliki opazimo, da trajanje med začetki operacij pogosto precej dobro popiše dejansko trajanje. Večinoma je nekoliko daljše od dejanskega trajanja, kar je v skladu s teorijo iz poglavja *2.3.3 Pretočni čas*. Trajanje med začetki operacij namreč vključuje še ležanje po obdelavi, transport in ležanje pred obdelavo. Prav tako mu lahko dodamo počitek delavca, opravljanje drugega dela delavca in podobno. Zgodi pa se tudi, da je trajanje med začetki krajše od dejanskega trajanja operacije, kar kažeta operaciji 4 in 14. To lahko razlagamo z vrinjenimi operacijami. Izkaže se, da se lahko na nekem delovnem mestu začne izvajati nova operacija, preden se opazovana zaključi. V primeru vrinjene operacije bo trajanje med začetki seveda krajše, saj bo začetek vrinjene operacije nastopil prej kot konec opazovane operacije.



Slika 5.5: Primerjava značilk o trajanjih

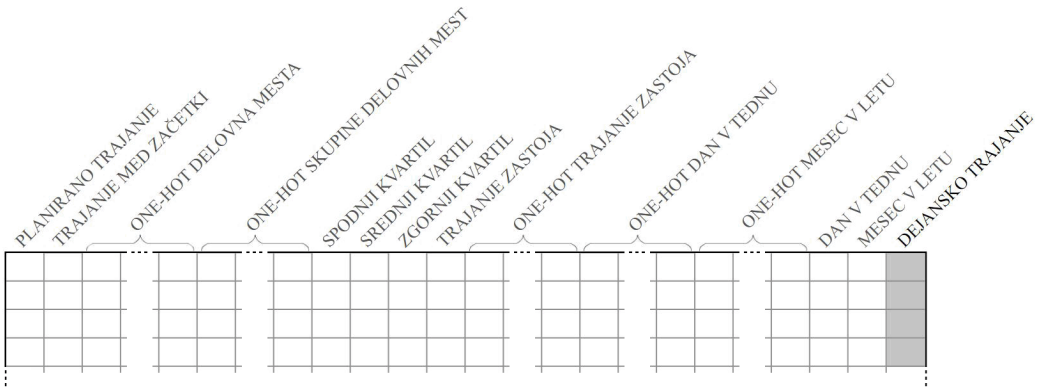
Značilke o delovnem mestu so pridobljene iz atributa DelovnoMesto. Tam najdemo tekstovne oznake delovnih mest, ki jih nevronske mreže kot vhod ne morejo sprejeti. Te kategorične podatke smo transformirali na način, da še vedno popisujejo kategorično naravo podatkov, a so predstavljeni na numeričen način. Odločili smo se za one-hot kodiranje, bolj podrobno opisano v poglavju *4.1.3 Vrste podatkov in strojno učenje*. Ustvarjen je bil vektor, katerega dolžina se ujema s številom kategorij delovnih mest (v našem primeru 160). Za vsako operacijo je vanj vpisano število 1 le na tisto mesto, ki predstavlja delovno mesto, na katerem je bila operacija izvedena. Odločili smo se tudi, da bomo ustvarili še en krajši one-hot vektor. En razlog za to odločitev je opisan na koncu poglavja *5.4.1 Čiščenje podatkov*, drug razlog pa je ta, da preveliko število značilk lahko vodi do prenasičenja modela (glejte poglavje *3.1.5 Prileganje*). To smo storili tako, da smo delovna mesta združili v 67 skupin. Znotraj skupine so med seboj podobna delovna mesta.

Značilke o kvartilih delovnih mest dajejo informacije o porazdelitvi dejanskih trajanj na delovnem mestu opazovane operacije. Za vsako delovno mesto smo zbrali vsa dejanska trajanja ter iz njih izračunali spodnji, srednji in zgornji kvartil. Ti so za posamezna delovna mesta prikazani že na sliki 5.3. Vsaka operacija z dodatkom teh značilk nosi še informacije o kvartilih za delovno mesto, na katerem je izvajana. Ideja je ta, da s tem delno popišemo porazdelitev trajanj na tem delovnem mestu. Kot smo ugotovili že ob sliki 5.2, se porazdelitve trajanj med delovnimi mesti močno razlikujejo.

Značilke o zastojih so pridobljene iz atributa DatumZastojEnd. Izračunana je bila razlika med začetkom dela operacije ter zaključkom zastoja iste operacije. Prva značilka je vsebovala le to informacijo. Želeli pa smo dodati tudi informacijo o vrsti zastoja, zapisano v atributu Zastoj. Težava je, da informacijo o vrsti zastoja najdemo v obliki kategoričnih podatkov. Podobno kot pri značilkah o delovnem mestu smo podatke kodirali v one-hot vektor. Ustvarjen vektor vsebuje 30 vrednosti, izmed katerih so vse, razen ene, 0. Neničelna vrednost opisuje omenjeno trajanje zastoja in je zapisana na mesto, ki predstavlja vrsto tega zastoja. Vektor dolžine 30 predstavlja za nevronsko mrežo 30 vhodnih značilk.

Značilke o datumu začetka dela so pridobljene iz atributa DatumZačetkaDela. Informacije o datumih smo pretvorili v dva one-hot vektorja. Prvi vektor dolžine 7 predstavlja dan v tednu, v katerem se je operacija začela izvajati. Drugi vektor pa je dolžine 12 in predstavlja mesec v letu, v katerem se je operacija začela izvajati. Ideja uporabe teh 19 značilk je, da so zaposleni morda ob ponedeljkih bolj produktivni kot ob nedeljah oziroma da decembra niso tako učinkoviti kot marca. Čeprav so dnevi v tednu in meseci v letu kategorični podatki, pa niso popolnoma nepovezani. Januar je bližje februarju kot avgustu in so podobni ordinalnim podatkom. S tem namenom smo ustvarili še dve značilki. Prva je vsebovala vrednosti od 0 do 6 (od ponedeljka do nedelje), druga pa vrednosti ob 0 do 11 (od januarja do decembra). Katera vrsta predstavitve teh podatkov bo zagotavljala boljše rezultate, pa na tej točki še ni jasno.

Rezultate zgoraj opisanih transformacij je bilo treba integrirati v eno tabelo. Ta tabela vsebuje 53827 vrstic in 285 stolpcev. Del podatkov, uporabljen za učenje nevronske mreže, predstavlja 17428 operacij, izmed katerih je vsaka predstavljena z 284 vhodnimi značilkami. En stolpec je rezerviran za informacijo o dejanskem času izvajanja operacije. Del podatkov, za katerega moramo dejanske čase izvajanja operacije napovedovati, pa prestavlja 36399 vrstic. Te prav tako vsebujejo 284 vhodnih značilk, stolpec o dejanskih časih izvajanja operacij pa je prazen, saj bomo te vrednosti napovedali. Tabela je zaradi velikosti lahko prikazana le shematsko, kot kaže slika 5.6.



Slika 5.6: Vizualizacija tabele z ustvarjenimi značilkami

Iz zgornje tabele smo ustvarili več različic. Vrednosti o trajanjih iz tabele smo na različne načine transformirali v pričakovanju, da bo nevronska mreža lažje prepoznala vzorce v podatkih. Pri prvi različici smo vrednosti korenili, kar predstavlja enačba (5.1). Druga različica je tabela, kjer so vrednosti logaritmirane. Problem predstavljajo vrednosti 0, saj po logaritmiranju zavzamejo vrednost $-\infty$. Ta problem se ponavadi rešuje tako, da se vsem vrednostim prišteje 1, kar kaže enačba (5.2). Tretja različica je podobna drugi, le da namesto logaritma z osnovo 10 uporabimo naravni logaritem. Za to transformacijo je uporabljena enačba (5.3). Četrta in peta različica se od druge in tretje razlikujeta le po prišteti vrednosti. Pogosto najdemo nasvet, da namesto števila 1 prištejemo polovico najmanjše neničelne vrednosti. Ker je v našem primeru najkrajše trajanje $0{,}0014$ ure, sta transformaciji taki, kot kažeta enačbi (5.4) in (5.5). Za šesto različico smo vrednosti transformirali z inverznim hiperboličnim sinusom, kar kaže enačba (5.6). Sedmo različico smo ustvarili tako, da smo vrednosti kvadrirali. To transformacijo opisuje enačba (5.7). Osma različica pa ni transformirana, kar lahko predstavimo, kot kaže enačba (5.8). V spodnjih enačbah predstavlja t netransformirano, t' pa transformirano trajanje.

$$t' = \sqrt{t} \tag{5.1}$$

$$t' = \log_{10}(t + 1) \tag{5.2}$$

$$t' = \ln(t + 1) \tag{5.3}$$

$$t' = \log_{10}(t + 0.0007) \tag{5.4}$$

$$t' = \ln(t + 0.0007) \tag{5.5}$$

$$t' = \operatorname{arcsinh}(t) \tag{5.6}$$

$$t' = t^2 \tag{5.7}$$

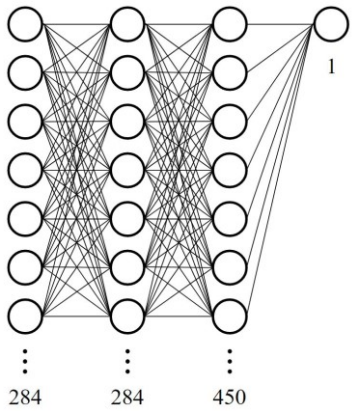
$$t' = t \tag{5.8}$$

5.5 Gradnja modela

Za gradnjo nevronske mreže smo uporabili Python knjižnico Theano. Uporabili smo tudi Keras, ki je knjižnica višjega nivoja in nam grajenje nevronske mreže poenostavi. Okolje, v katerem je potekalo delo, je bilo kot pri prejšnjih korakih JetBrains PyCharm.

Prvi korak je predstavljal odločitev o tem, kakšne vrste mrežo želimo ustvariti. Upoštevati smo morali, da napovedujemo zvezno spremenljivko. Opravka imamo torej z regresijskim problemom. Odločili smo se, da bo ustvarjena usmerjena nevronska mreža (angl. *feed forward neural network*), ki se za take probleme najpogosteje uporablja. Usmerjene mreže so prav tako primerne za delovanje na osebnem računalniku. Zahtevnejše vrste mreže bi bilo verjetno treba implementirati na zmogljivejših sistemih.

Nato smo zgradili prvo verzijo mreže. Shematski prikaz te mreže je prikazan na sliki 5.7. Vhodni nivo ima 284 nevronov, toliko kot je ustvarjenih značilk. Temu sledi prvi skriti nivo z 284 nevroni, nato pa še en skriti nivo s 450 nevroni. Na izhodnem nivoju je le en nevron, saj gre za regresijski problem.



Slika 5.7: Shematski prikaz prve verzije nevronske mreže

Preden smo se lahko lotili izboljševanja napovedovanja, smo se morali odločiti, kaj bo enoznačno in objektivno merilo kakovosti napovedovanja. To določimo z izbiro funkcije napake. Ta je ena izmed hiperparametrov nevronske mreže, zato pravimo, da smo s tem delom začeli proces optimizacije hiperparametrov. Spodaj so naštete vrste napak, katerih funkcije smo testirali:

- povprečna absolutna napaka (angl. *mean absolute error*) oziroma kratica PAN,

- povprečna kvadrirana napaka (angl. *mean squared error*) oziroma kratica PKN,
- povprečna absolutna odstotkovna napaka (angl. *mean absolute percentage error*) oziroma kratica PAON,
- povprečna kvadrirana logaritemska napaka (angl. *mean squared logarithmic error*) ozriroma kratica PKLN,
- R2 napaka (angl. *R2 error*) oziroma kratica R2N,
- Huberjeva napaka ali gladka absolutna napaka (angl. *Huber loss*) oziroma kratica HN.

Prve štiri naštete najdemo definirane že v knjižnici Keras. Zadnji dve smo ustvarili sami.

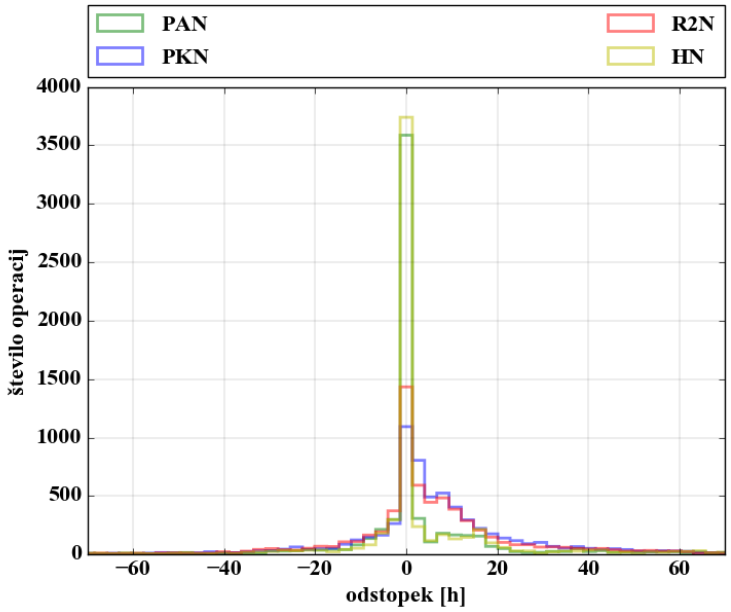
Pred učenjem moramo algoritmu definirati metodo, s katero preverjamo veljavnost. Tu smo se odločili za metodo razdelitve. Dve tretjini podatkov smo uporabili za učni del, eno tretjino pa za testni del. Keras nam po učenju mreže na učnem delu vrne vrednost napake, izračunane na testnem delu podatkov. Teh vrednosti pa med seboj zaradi spreminjanja funkcije napake ne moremo primerjati. S tem namenom smo napovedane vrednosti primerjali s tabelami in vizualizacijami.

Najprej smo napovedi različnih mrež primerjali s pomočjo dveh pogosto uporabljenih kazalcev kakovosti napovedi pri regresiji. To sta povprečna vrednost odstopkov ter povprečna kvadrirana vrednost odstopkov. Za prvo smo izračunali še standardno variacijo, saj si je tako odstopke lažje predstavljati. Odstopek definiramo kot razliko med napovedanim in dejanskim trajanjem. Vse skupaj je prikazano v preglednici 5.1, kamor smo dodali še informacijo o času učenja modela.

Preglednica 5.1: Rezultati primerjave funkcij napake

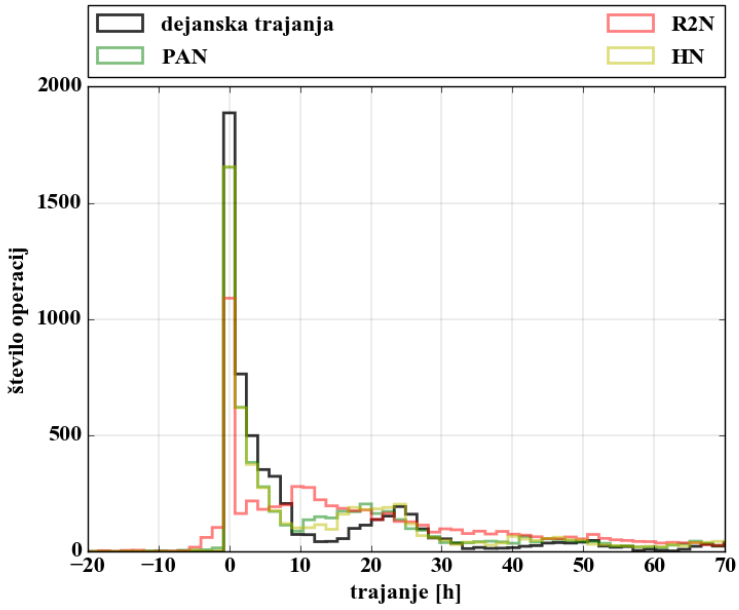
Funkcija napake	Povprečna vrednost odstopkov [h]	Povprečna kvadrirana vrednost odstopkov [h^2]	Standardna deviacija odstopkov [h]	Čas učenja [s]
PAN	**-2,40**	**2498,2**	**49,9**	**192,2**
PKN	4,86	2764,1	52,4	184,2
PAON	-29,57	4356,0	59,0	262,7
PKLN	-26,47	4074,8	58,1	220,6
R2N	2,41	2562,6	50,6	196,8
HN	-2,59	2535,1	50,3	258,0

Ker želimo, da so vse vrednosti v preglednici 6.1 čim manjše, je hitro jasno, da PAON in PKLN nista najbolj primerni. Obe napovedujeta večinoma prekratke vrednosti, kar lahko razberemo iz prvega stolpca preglednice 6.1. Tudi čas učenja je nekoliko višji, čeprav to nima tako bistvene vloge. Zaradi razločnosti vizualizacij v nadaljevanju ne bosta več prikazani. Odstopke lahko tudi vizualiziramo in tako o njih dobimo več informacij. To prikazuje slika 5.8.



Slika 5.8: Primerjava odstopkov za različne funkcije napake

Opazimo lahko, da sta si porazdelitvi PAN in HN ter PKN in R2N med seboj podobni. Slednji dve bolj kaznujeta velike odstopke. Vizualizacija dodatno potrjuje, da PKN ni najboljša izbira. Že v tabeli je v vseh pogledih, razen času učenja, slabša od ostali treh. Zato je na sliki 5.9, ki prikazuje porazdelitev trajanj, ne najdemo več.



Slika 5.9: Primerjava trajanj za različne funkcije napake

Kot omenjeno že v definiciji problema, je za nas poleg točnosti napovedi pomembna tudi njihova porazdelitev. Želimo si, da je porazdelitev napovedanih trajanj čim bolj podobna porazdelitvi dejanskih trajanj. Na sliki 5.9 vidimo, da porazdelitvi dejanskih trajanj najslabše sledi R2N. Najboljša se zdi HN, a je razlika med njo in PAN zelo majhna. Če

pogledamo vrednosti v tabelah, pa lahko damo malo prednost PAN. Zaradi velike podobnosti in pomembnosti izbire smo se odločili, da bomo primerjavo med PAN in HN ponovili po prilagajanju podatkov za boljše napovedovanje. Do takrat bomo uporabljali PAN, saj z uporabo te funkcije učenje poteka nekoliko hitreje.

V naslednjih korakih želimo izboljšati napovedovanje naše nevronske mreže. To lahko storimo na dva načina. Spreminjamo lahko podatke, ki jih modelu zagotovimo. Lahko pa spreminjamo model sam. Naše delo na teh dveh področjih je opisano v nadaljevanju.

5.5.1 Prilagajanje podatkov

Izbor značilk je prvi korak, ki smo se ga na tem mestu lotili. Želeli smo ugotoviti, ali potrebujemo vseh 284 značilk. Morda bo kakovost napovedovanja enaka ali celo boljša, če nekatere izmed njih iz podatkov odstranimo. Manj značilk v splošnem pomeni tudi krajši čas učenja, saj obstaja manj parametrov, ki se jih pri učenju preračunava.

Preglednica 5.2: Rezultati izbora značilk

Ime (število značilk)	Številka primera															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	**15**	16
Planirano trajanje (1)	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Trajanje med začetki (1)	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
One-hot delovna mesta (160)												✓		✓	✓	✓
One-hot skupine delovnih mest (67)													✓	✓	✓	✓
Kvartili trajanj na delovnem mestu (3)								✓								✓
Trajanje zastoja (1)	✓	✓	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓		✓
One-hot trajanje zastoja (30)	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
One-hot dan v tednu (7)										✓	✓				✓	✓
One-hot mesec v letu (12)										✓	✓				✓	✓
Dan v tednu (1)									✓		✓					✓
Mesec v letu (1)									✓		✓					✓
PAN na testnem delu podatkov [h]	15,47	23,32	19,80	18,79	41,60	14,76	19,30	15,68	15,36	15,30	15,44	14,97	15,14	14,49	**14,04**	14,55

Legenda: ✓ značilka je bila vključena v model za ta primer

Preglednica 5.2 prikazuje rezultate za različne kombinacije značilk. Prvi primer smo obravnavali kot osnovni model. Vanj smo nato vključevali oziroma iz njega izključevali značilke. Eksperimentiranje je bilo zasnovano tako, da rezultat vsakega poskusa primerjamo z rezultatom osnovnega modela. Na ta način ugotovimo, katere značilke vplivajo na rezultat pozitivno in katere negativno. Hipoteza je namreč ta, da če na koncu v model vključimo le značilke, ki na rezultat vplivajo pozitivno, bo končni model boljši od modela, ki ima vključene vse značilke.

S primeri 2, 3 in 4 smo raziskovali vplive značilk o planiranem trajanju in trajanju med začetki. Izkazalo se je, da izključitev ene ali obeh značilk negativno vpliva na napovedovanje. S primeri 5, 6 in 7 smo preverjali vplive značilk, ki dajejo informacije o zastojih. Rezultati kažejo na to, da je najbolje, če je v model vključena le značilka trajanje zastoja. To je presenetljivo, saj značilke one-hot trajanje zastoja vsebujejo iste informacije kot trajanje zastoja in dodajajo še informacijo o vrsti zastoja. Razloge za tako obnašanje je težko najti, saj se nevronske mreže obnašajo kot neke vrste »črna škatla«. Ta ne daje informacij o tem, zakaj je obnašanje tako, kot je. Predvidevamo lahko, da je razpršitev informacij o trajanju zastoja na več značilk bolj škodovalo učenju, kot so pomagale dodatne informacije o vrsti zastoja. Mogoče je, da v učnih podatkih ni dovolj informacij za vsako vrsto zastoja, zato se uteži težko primerno prilagodijo. S primerom 8 smo preverjali koristnost značilk kvartili. Izkazalo se je, da te na rezultat vplivajo negativno. S primeri 9, 10 in 11 smo raziskovali vplive značilk o dnevu v tednu in mesecu v letu. Rezultati kažejo na to, da najbolje deluje model z obema one-hot vektorjema. Čeprav si meseci in dnevi sledijo zaporedno, jih je, kot kaže, bolj smiselno obravnavati kot nepovezane. Zanimivo je tudi to, da je napovedovanje slabše, če so v model vključene vse značilke, kot če so modelu zagotovljene posebej. S primeri 12, 13 in 14 pa smo preverjali vplive značilk o delovnih mestih. Rezultati kažejo, da je v modelu smiselno zadržati oba one-hot vektorja. To je ravno obratno od tega, kar se je zgodilo pri raziskovanju prejšnjih treh značilk. Izkazalo se je torej, da je pri izbiri nujno potrebno eksperimentiranje in da je obnašanje težko predvideti. Primer 15 vključuje le značilke, ki so za napovedovanje vplivale pozitivno. Ob primerjavi s primerom 16, ki ima vključene vse značilke, lahko našo hipotezo potrdimo. Povprečna absolutna napaka je namreč padla za približno pol ure.

Povečevanje količine podatkov je naslednji korak, ki smo se ga lotili. Pridobitev večjega števila podatkov za nas ni bila mogoča. Mogoče pa je bilo umetno povečati število podatkov. To smo storili tako, da smo podatke najprej razdelili. Kot pravi metoda razdelitve, smo ustvarili učni in testni del podatkov. Količino učnega dela smo nato umetno povečali. Le razmnoževati primere ni smiselno, saj lahko enak učinek dosežemo s povečevanjem števila ciklov pri učenju. Podatke zato razmnožimo ter dodatnim podatkom dodamo malo šuma. V preglednici 5.3 so rezultati za modele s povečanim številom podatkov. Spreminjali smo tako količino razmnoženih podatkov kot tudi količino dodanega šuma.

V prvi vrstici tabele je primer brez povečevanja količine podatkov. Rezultat PAN za testni del podatkov tega primera se zdi, da bi se moral ujemati s tistim v preglednici 5.2. Do majhne razlike pride, ker smo mrežo ponovno učili. Učenje je namreč do neke mere naključno. To omejujemo tako, da v programu definiramo naključno seme (angl. *random seed*), a se rezultati lahko klub temu med seboj nekoliko razlikujejo.

Vrstice 2, 3, 4 predstavljajo primere, kjer smo količino podatkov podvojili. Razlika med njimi je v količini dodanega šuma. V 3. vrstici je bila trajanjem prišteta naključna vrednost iz intervala, omejenega z vrednostima -1 % trajanja in +1 % trajanja. V vrstici 5 pa je primer, kjer smo število podatkov potrojili.

Preglednica 5.3: Rezultati povečevanja števila primerov

Povečanje števila primerov	Dodan šum [%]	PAN [h]		Čas učenja [s]
		Učni del podatkov	Testni del podatkov	
1x	**0**	**11,90**	**14,01**	**202,2**
2x	0,5	10,80	14,62	451,8
2x	1	11,26	14,20	438,0
2x	2	11,40	14,09	423,9
3x	1	10,35	15,15	797,1

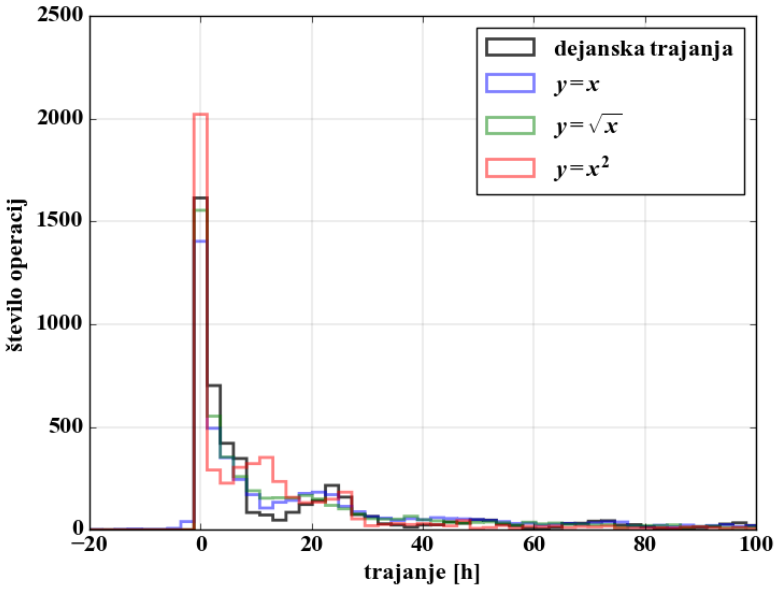
Za učni del podatkov se s povečevanjem količine podatkov vrednost PAN manjša. To je smiselno, saj se količina podatkov, na katerih se uči, povečuje. Zgodi se podobno, kot če bi v kodi povečali število ciklov. Težava pa je v tem, da se PAN za testni del podatkov povečuje. To pa je tisto, kar govori o sposobnosti modela za generalizacijo. Daje nam informacije o tem, kako dober je model za napovedovanje na podatkih, s katerimi še ni imel opravka. Najboljši rezultat PAN na testnem delu podatkov nam daje originalen niz podatkov. Razlog za zmanjševanje PAN na učnem delu in povečevanje PAN na testnem delu je, da se model začne podatkom preveč prilegati. Prenasičen model učne podatke popiše zelo dobro, a ga ne moremo generalizirati še na druge podatke. V poglavju *3.1.5 Prileganje* smo povedali, da si želimo v podatkih vsaj desetkrat več primerov, kot imamo značilk. Naši podatki vsebujejo sedemdesetkrat več primerov kot značilk. Zdi se, da v našem primeru količina podatkov ne predstavlja težave. Povečevanje količine podatkov podaljšuje tudi čas učenja, zato to ni bilo uporabljeno.

Izbira različice ustvarjenih podatkov je bila naslednji korak. V poglavju *5.4.2 Transformacija podatkov* je opisanih 8 ustvarjenih različic podatkov. Rezultatov PAN med seboj ne moremo neposredno primerjati, saj so tudi napovedi posledično v transformirani obliki. Preglednica 5.4 govori o PAN in PKN za vse različice podatkov. Vrednosti v njej so transformirane nazaj v osnovne enote.

Preglednica 5.4: Rezultati različic podatkov

Različica	PAN [h]	PKN [h^2]
$t' = \sqrt{t}$	15,92	4233,0
$t' = \log_{10}(t+1)$	23,47	9499,4
$t' = \ln(t+1)$	19,57	6185,4
$t' = \log_{10}(t+0.0007)$	32,31	66896,4
$t' = \ln(t+0.0007)$	29,65	152009,1
$t' = \operatorname{arcsinh}(t)$	19,52	6544,2
$t' = t^2$	16,50	4676,6
$\boldsymbol{t' = t}$	**13,95**	**3983,8**

Ob pogledu na preglednico 5.4 je jasno, da najboljše rezultate predstavlja zadnja vrstica, kjer gre za netransformirane podatke. Še posebej slabe rezultate so dale različice, kjer so bili podatki logaritmirani. To nas preseneča, saj z logaritmiranjem pridemo do porazdelitve trajanj, ki je bolj podobna normalni porazdelitvi. Ocenjujemo, da je razlog za slabo napoved ta, da se majhna napaka pri antilogaritmiranju lahko močno poveča. Prekratke napovedi postanejo še krajše, predolge napovedi pa še daljše. Za nas pa je pomembna tudi porazdelitev trajanj, zato smo to tudi vizualizirali. Na sliki 5.10 so prikazane porazdelitve 3 najboljših različic iz preglednice 5.4.



Slika 5.10: Porazdelitev trajanj različnih različic podatkov

Slika 5.10 kaže na to, da izmed prikazanih najslabše zadene porazdelitev model s kvadriranimi trajanji. Preveč operacijam napove čas izvedbe med 0. in 1. uro. Prav tako ne zadene dobro že opisanih nihanj vsakih 24 ur. Iz slike 5.10 se zdita modela s korenjenimi trajanji in trajanji brez transformacije približno enako dobra za napovedovanje. Prvi zadene nekoliko več operacij s trajanjem 0, slednji pa nekoliko bolje sledi nihanjem. Na podlagi grafa in tabele smo se odločili, da bomo uporabljali podatke brez transformacije.

5.5.2 Optimizacija hiperparametrov

V poglavju *3.2.5 Strojno učenje* so opisani različni načini optimizacije hiperparametrov. Odločili smo se za kombinacijo ročnega in mrežnega iskanja. Ročno je za naš primer ustrezno, saj je računsko manj zahtevno. To je pomembno zato, ker mrežo učimo na osebnem računalniku. Zagotavlja nam tudi veliko stika z optimizacijo in nam omogoča sprotno spremljanje rezultatov ter generacijo vizualizacij, iz katerih se učimo. Tako delo je sicer časovno potratno, a nam omogoča pridobitev bolj poglobljenega znanja in intuicije. Ročni način je še posebej pomemben pri izbiri tistih hiperparametrov, kjer je mogočih kombinacij veliko (aktivacijska funkcija, število nivojev ter nevronov itd.). Za primere,

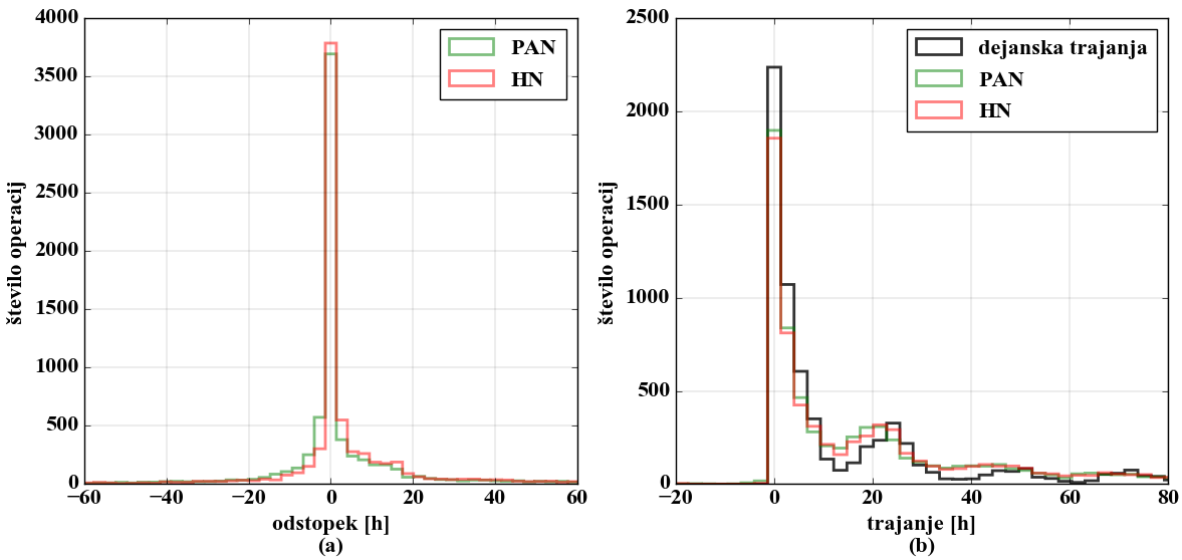
kjer je mogočih kombinacij manj (optimizacijski algoritem, inicializacija uteži itd.), pa je bilo za opazovan hiperparameter uporabljeno mrežno iskanje. To je koristno, saj pri učenju različnih kombinacij ni potrebno, da smo stalno prisotni.

Izbira funkcije napake se je na tej točki še enkrat ponovila. Primerjali smo le PAN in HN. Rezultati analize so v preglednici 5.5 in na sliki 5.11. Vrednosti veljajo za testni del podatkov.

Preglednica 5.5: Rezultati primerjave PAN in HN

Funkcija napake	Povprečna vrednost odstopkov [h]	Povprečna kvadrirana vrednost odstopkov [h^2]	Standardna deviacija odstopkov [h]	Čas učenja [s]
PAN	-2,28	3134,7	55,12	168,4
HN	**2,15**	**2975,8**	**54,21**	**176,9**

Iz preglednice 5.5 razberemo, da se je tokrat HN izkazala nekoliko bolje. Napovedovanje z uporabo te funkcije je bilo nekoliko bolj točno, a smo za učenje potrebovali nekoliko dlje. Slika 5.11 potrjuje, kar nakazuje že tabela. PAN generalno napoveduje nekoliko prekratka trajanja, HN pa nekoliko predolga. Porazdelitev trajanj je v obeh primerih podobna porazdelitvi dejanskih trajanj.



Slika 5.11: Porazdelitev odstopkov (a) in trajanj (b) za PAN in HN

Odločili smo se, da bomo za nadaljevanje uporabljali HN. Prvi razlog je, da tokratni rezultati kažejo na malo boljše napovedovanje z njeno uporabo. Drugi razlog pa je ta, da je HN nekoliko manj občutljiva na izolirane točke.

Izbira optimizacijskega algoritma je naslednji korak optimizacije hiperparametrov. Knjižnica Keras vsebuje 7 optimizacijskih algoritmov. Rezultati testiranja vseh 7 so podani v preglednici 5.6. Do te točke je bil uporabljen optimizacijski algoritem Adam.

Na podlagi preglednice 5.6 je izbira preprosta. Algoritem Adamax daje najboljše rezultate tako s stališča napake kot časovne učinkovitosti. Gre za variacijo priljubljenega algoritma Adam, le da ta temelji na konceptu, imenovanem neskončna norma [40].

Preglednica 5.6: Rezultati različnih optimizacijskih algoritmov

Optimizacijski algoritem	HN [h]	Čas učenja [s]
RMSprop	14,49	78,5
SGD	47,33	54,5
Adagrad	13,26	87,1
Adadelta	13,62	98,4
Adam	13,35	87,6
Adamax	**13,20**	**72,0**
Nadam	14,42	115,7

Izbira načina inicializacije uteži je korak, ki določi, kakšne vrednosti so utežem pripisane pred začetkom učenja. Poskusili smo 8 različnih načinov, katerih rezultati so podani v preglednici 5.7. Do te točke smo uporabljali normalno porazdeljene vrednosti uteži.

Preglednica 5.7: Izbira načina inicializacije uteži

Inicializacija uteži	HN [h]	Čas učenja [s]
Normalna porazdelitev	13,14	69,0
Enakomerna porazdelitev	**12,68**	**68,8**
Lecun enakomerna porazdelitev	13,45	62,5
Ničle	53,92	65,3
Glorot normalna porazdelitev	13,21	67,2
Glorot enakomerna porazdelitev	13,21	75,0
He normalna porazdelitev	13,77	70,4
He enakomerna porazdelitev	13,56	64,5

S stališča točnosti se najbolje obnese enakomerna (angl. *uniform*) porazdelitev. Čeprav je čas učenja pri njej nekoliko daljši, smo se odločili za njeno uporabo, saj to za nas ni kritičnega pomena. Interval vrednosti, za katerega enakomerna porazdelitev velja, ima mejne vrednosti -0,05 in 0,05. Izkazalo se, da teoretični nasveti iz poglavja *3.1.14 Inicializacija in omejevanje uteži* držijo. Ničle so se namreč izkazale za najslabšo izbiro.

Izbira aktivacijske funkcije temelji na dveh eksperimentih. Pri prvem smo spreminjali aktivacijsko funkcijo skritih nivojev, v drugem pa aktivacijsko funkcijo izhodnega nivoja. Izbiramo lahko med 10 različnimi aktivacijskimi funkcijami. Do te točke je bila na skritih nivojih uporabljena funkcija ReLU, na izhodnem pa linearna funkcija. Rezultati eksperimentov so prikazani v preglednicah 5.8 in 5.9.

Preglednica 5.8: Izbira aktivacijske funkcije skritih nivojev

Aktivacijska funkcija	Zaloga vrednosti	HN [h]	Čas učenja [s]
Softmax funkcija	(0,1)	52,79	182,6
Softplus funkcija	(0,∞)	13,22	192,4
Softsign funkcija	(-1,1)	36,08	70,7
Funkcija ReLU	**[0,∞)**	**12,81**	**71,4**
Funkcija hiperboličnega tangensa	(-1,1)	34,46	84,7
Sigmoidna funkcija	(0,1)	37,08	119,3
Ostra sigmoidna funkcija	[0,1]	38,06	83,1
Linearna funkcija	(-∞,∞)	17,95	69,1
Elu funkcija	$(-\alpha, \infty)$	12,95	161,7
Selu funkcija	$(-\lambda\alpha, \infty)$	13,28	175,3

Iz preglednice 5.8 je razvidno, da dobre rezultate zagotavljajo funkcije, katerih zaloga vrednosti je v območju, v kakršnem so tudi trajanja. Softmax funkcija se ne obnese dobro, saj je njena zaloga vrednosti od 0 do 1, naša trajanja pa so med vrednostjo 0 in nekaj 1000 ur. Najboljše rezultate daje že prej uporabljena funkcija ReLU. Presenetljivo dobre rezultate zagotalja tudi linearna funkcija. V tem primeru je na vseh nivojih uporabljena linearna funkcija (tako skritih kot izhodnem). V poglavju *3.1.8 Aktivacijska funkcija* smo omenili, da se samo linearna funkcija ponavadi ne uporablja. Ta je najpogosteje uporabljena le na izhodnem nivoju. V primeru uporabe linearne funkcije na vseh nivojih se nevronska mreža obnaša, kot da vsebuje le en nevron. Gre za napovedovanje, podobno običajni linearni regresiji.

Preglednica 5.9: Izbira aktivacijske funkcije izhodnega nivoja

Aktivacijska funkcija	Zaloga vrednosti	Brez spremembe območja		Sprememba območja	
		HN [h]	Čas učenja [s]	HN [h]	Čas učenja [s]
Softmax funkcija	(0,1)	54,18	79,6	1230,29	111,5
Softplus funkcija	(0,∞)	12,78	91,2	12,92	128,2
Softsign funkcija	(-1,1)	53,91	77,8	158,36	137,2
Funkcija ReLU	[0,∞)	12,68	78,5	12,80	148,9
Funkcija hiperboličnega tangensa	(-1,1)	53,87	85,8	132,23	140,0
Sigmoidna funkcija	(0,1)	53,85	89,3	24,72	137,1
Ostra sigmoidna funkcija	[0,1]	53,85	95,6	43,64	143,1
Linearna funkcija	(-∞,∞)	12,89	78,1	13,01	154,0
Elu funkcija	$(-\alpha, \infty)$	12,86	83,5	13,04	134,2
Selu funkcija	$\boldsymbol{(-\lambda\alpha, \infty)}$	**12,73**	**86,5**	12,84	137,1

Nato smo se lotili še izbire aktivacijske funkcije izhodnega nivoja, kar prikazuje preglednica 5.9. Raziskovanje smo razdelili na dva dela. V prvem delu smo območje podatkov pustili nespremenjeno. V drugem delu pa smo podatke transformirali tako, da se

njihovo območje ujema z zalogo vrednosti aktivacijske funkcije. Nekateri raziskovalci ugotavljajo, da je za določene primere taka transformacija zaželena.

V nekaterih primerih je sprememba območja napovedovanje nekoliko izboljšala (sigmoidna funkcija in ostra sigmoidna funkcija), v večini pa je to rezultate poslabšalo. Kar nekaj primerov zagotavlja rezultate, katerih HN je približno 50 ur. Taka in manjša vrednost HN predstavlja slabo napovedovanje. Tej vrednosti bi bila velikost napake podobna že v primeru, če bi vsem operacijam napovedali dolžino trajanja enako 0. Povprečen čas izvedbe operacije je namreč približno 55 ur. Slab rezultat modelov z določenimi aktivacijskimi funkcijami na izhodu ne preseneča, saj se te za regresijo ne uporabljajo. Softmax se uporablja za klasificirajo v več razredov, sigmoidna funkcija pa za binarno klasifikacijo. Razlog za slabo napovedovanje slednje je lahko zmanjševanje gradienta te funkcije. Pri velikih vrednostih je ta namreč blizu 0, kar je s stališča napovedovanja zvezne spremenljivke slabo.

Na podlagi opisanih rezultatov smo se odločili, da bomo v nadaljevanju za skrite nivoje uporabljali funkcijo ReLU, za izhodni nivo pa funkcijo SeLU. Podatkom območja ne bomo spreminjali, saj nam je to podaljšalo učenje, napovedovanja pa ni izboljšalo.

Izbira števila nivojev ter nevronov je ena najpogosteje uporabljenih načinov optimizacije mreže. Pri izbiri najboljše topologije mreže imamo neskončno možnosti. Ročno preverjanje različnih topologij je potekalo tako, da smo poskusili napovedovanje različnih vrst mrež. Mreže so namreč lahko plitke, a zelo široke, lahko so ozke, a globoke, lahko se izmenično širijo ter ožijo in tako naprej. Mreža do te točke je vsebovala vhodni nivo z 284 vhodi, prvi skriti nivo z 284 nevroni, drugi skriti nivo s 450 nevroni in izhodni nivo z 1 nevronom.

Rezultate prve iteracije prikazuje preglednica 5.10. Iz nje lahko razberemo število skritih nivojev ter število nevronov na posameznem skritem nivoju. Nivoji so med seboj ločeni z vejico. Število vhodov na vhodnem nivoju ostaja 284, na izhodu pa 1 nevron. Nivoji v preglednici 5.10 se vrstijo tako, da skrajno levi sledi vhodnem nivoju, tisti na desni pa stoji pred izhodnim nivojem. Opazovali smo napovedovanje 5 različnih skupin topologij mreže. Prva vsebuje le 1 skriti nivo. Druga vsebuje 2 skrita nivoja in ji spreminjamo število nevronov na prvem skritem nivoju. Tretja je podobna drugi, le da ji spreminjamo število nevronov na drugem skritem nivoju. Četrta, za razliko od prvih treh, predstavlja globoke, a ozke mreže. Peta pa vsebuje primere različnih razširitev in oženj.

Izkazalo se je, da najbolje napovedujejo široke in plitke mreže. Mreže 5, 11 in 17 so s stališča HN najboljše. Ni presenetljivo, da je čas učenja za mrežo 17 precej daljši kot tisti za 5 in 11. Slednji mreži vsebujeta namreč precej manj nevronov, zato je število računskih operacij precej manjše. Te najboljše tri mreže smo vzeli za osnovo eksperimentiranja v naslednji operaciji. Tam smo število nevronov povečevali in zmanjševali v manjšem intervalu v upanju, da najdemo še boljšo topologijo mreže. Rezultati tega dela so prikazani v preglednici 5.11.

Preglednica 5.10: Rezultati prve iteracije iskanja topologije mreže

Vrsta	Številka mreže	Število nevronov v skritih nivojih	HN [h]	Čas učenja [s]
1	1	100	14,43	39,5
	2	250	13,54	48,2
	3	284	13,52	43,9
	4	500	13,03	47,3
	5	**1000**	**12,75**	**70,7**
	6	2000	12,81	142,0
2	7	100, 500	13,04	51,2
	8	250, 500	13,12	71,7
	9	284, 500	12,88	81,8
	10	350, 500	12,88	90,6
	11	**500, 500**	**12,68**	**113,0**
	12	1000, 500	12,73	218,0
3	13	500, 50	12,90	60,9
	14	500, 200	12,97	73,6
	15	500, 500	12,68	113,0
	16	500, 1000	12,65	199,0
	17	**500, 2000**	**12,53**	**351,0**
	18	500, 3000	12,57	487,0
4	19	10, 10, 10, 10, 10, 10	16,54	41,9
	20	30, 30, 30, 30, 30, 30	14,57	45,9
	21	100, 100, 100, 100, 100, 100	13,72	71,3
	22	250, 250, 250, 250, 250, 250	14,22	139,0
	23	500, 500, 500, 500, 500, 500	16,28	373,0
	24	100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100	15,53	105,0
5	25	250, 2000, 1000, 500, 250, 250	14,16	1062,0
	26	250, 1000, 500, 250, 25	13,15	314,0
	27	250, 1000, 250, 100, 25	14,06	201,0
	28	500, 1000, 500, 250, 100	14,60	395,0
	29	250, 25, 250, 25	13,47	54,7
	30	250, 500, 250, 500	13,11	157,0
	31	250, 1000, 250, 1000	13,57	254,0
	32	250, 150, 75, 750	13,67	94,1

Številni rezultati v preglednici 5.11 so prejšnjim rezultatom s stališča napake zelo podobni, saj do razlikovanja pride šele v drugi decimalni vrednosti. To je lahko deloma posledica naključne narave učenja. Kljub temu vrednost HN kaže na to, da mreža 17 iz preglednice 5.10 še vedno zagotavlja najboljše rezultate. Trdimo lahko, da nam povečevanje števila nivojev, nevronov in s tem kompleksnosti ne prinaša nujno izboljšave napovedovanja. Zdi se, da ko dosežemo določen nivo kompleksnosti mreže, je ta sposobna relacije in vzorce v podatkih, kar se da dobro popisati. Nadaljnje povečevanje kompleksnosti za napovedovanje ni nujno smiselno, saj je napovedi take mreže pogosto težje generalizirati.

Preglednica 5.11: Rezultati druge iteracije iskanja topologije mreže

Vrsta	Število nevronov v skritih nivojih	HN [h]	Čas učenja [s]
1	750, 1	12,89	60,9
	1000, 1	12,75	70,7
	1250, 1	12,77	94,8
2	425, 425, 1	12,84	94,6
	500, 500, 1	12,68	113,0
	750, 750, 1	12,56	264,53
	825, 825, 1	12,81	264,4
	1000, 1000, 1	12,77	354,4
3	500, 1500, 1	12,64	312,5
	250, 2000, 1	12,89	225,1
	350, 2000, 1	12,76	241,6
	500, 2000, 1	**12,53**	**351,0**
	650, 2000, 1	12,84	434,3
	750, 2000, 1	12,86	539,2
	500, 2500, 1	12,69	487,9

Izbrana je bila topologija, kjer imamo na vhodu 284 nevronov, nato skrita nivoja s 500 in 2000 nevroni, na izhodnem nivoju pa 1 nevron.

Velikost paketa je naslednji hiperparameter, ki smo si ga želeli optimizirati. Povečevali smo ga vse do največje možne vrednosti 10000, pri čemer so zajeti vsi podatki naenkrat. Razlog, da pri tej vrednosti zajamemo vse primere, je, da smo uporabili metodo razdelitve in je bilo toliko primerov v učnem delu podatkov. Rezultati so podani v preglednici 5.12. Do te točke je bila uporabljena velikost paketa 100.

Preglednica 5.12: Rezultati različnih velikosti paketa

Velikost paketa [primer]	HN [h]	Čas učenja [s]
10	12,68	2676,6
50	12,83	570,1
100	12,57	359,3
150	**12,47**	**302,5**
200	13,21	247,6
500	13,13	200,8
1000	12,96	199,7
10000	16,59	189,5

Opazimo, da se časi učenja s povečevanjem velikosti paketa zmanjšujejo. Razlog je, da je za izvedbo enega cikla potrebnih manj iteracij. S stališča napake najboljše rezultate zagotavlja velikost paketa, ki vključuje 150 primerov. Ta vrednost je bila uporabljena tudi v nadaljevanju optimizacije hiperparametrov.

Izbira hitrosti učenja in števila ciklov sta naslednja hiperparametra na seznamu optimizacije. Ne moremo ju obravnavati ločeno, saj hitrost učenja močno vpliva na

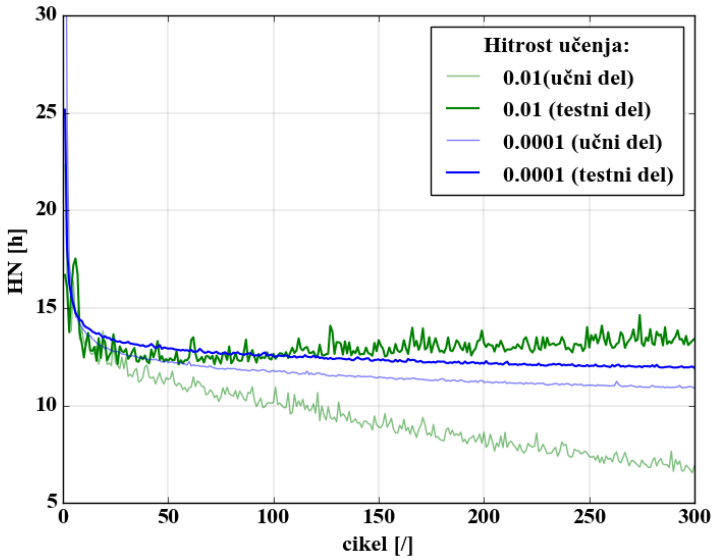
potrebno število ciklov. Privzeta hitrost učenja za optimizacijski algoritem Adamax je 0,002 in je med učenjem konstantna. Pri eksperimentiranju smo uvedli tudi zmanjševanje hitrosti učenja.

Eksperimenta smo se lotili tako, da smo najprej testirali različne hitrosti učenja brez zmanjševanja hitrosti učenja. Rezultati tega dela so v preglednici 5.13. Za različne hitrosti učenja so zapisane pripadajoče minimalne vrednosti HN in številka cikla, pri katerem je do minimuma HN prišlo.

Preglednica 5.13: Rezultati različnih hitrosti učenja

Hitrost učenja	HN [h]	Številka cikla
0,0001	11,82	494
0,00025	**11,81**	**265**
0,0005	11,91	85
0,001	11,89	70
0,002	12,04	84
0,005	12,09	70
0,01	12,12	63

Pri izbiri hitrosti učenja smo omejeni. Pri preveliki hitrosti učenje sploh ne bo delovalo, pri prenizki hitrosti pa bo učenje trajalo izjemno dolgo. Opazimo lahko, da se s povečevanjem hitrosti učenja povečuje tudi HN, zmanjšuje pa se številka cikla, pri katerem je HN minimalna. Za boljšo predstavo omenjenih dejstev je vključena slika 5.12.



Slika 5.12: Primerjava velike in majhne hitrosti učenja

Na sliki 5.12 opazimo med veliko in majhno hitrostjo učenja veliko razlik. Prva je razlika v nihanju krivulje HN ob učenju. V primeru hitrega učenja je sprememba uteži pri vsakem

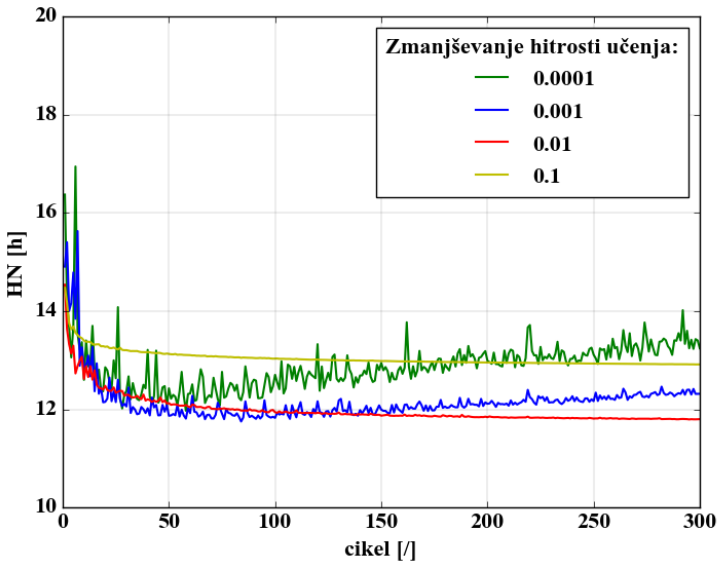
ciklu velika, kar povzroča veliko nihanje. Velike spremembe uteži pri vsakem ciklu pojasnjujejo tudi hiter padec HN za testni del podatkov. Opazimo lahko tudi, da velika hitrost učenja na koncu povzroča slabše rezultate testnega dela. Medtem ko se HN testnega dela pri počasnem učenju vztrajno zmanjšuje, začne pri hitrem učenju ta naraščati. To je posledica prekomernega prilagajanja podatkom. Vidimo namreč, da HN za učni del pada, HN za testni del pa se začne dvigovati (glejte poglavje *3.1.5 Prileganje*).

Pozitivne lastnosti hitrega in počasnega učenja je mogoče združiti s pomočjo sprotnega zmanjševanja hitrosti učenja. S tem smo želeli ujeti začetni hitri padec HN, ki je značilen za hitro učenje. Prav tako pa smo želeli ujeti vztrajno padanje HN brez večjih nihanj, ki ga kaže počasno učenje. Izkaže se, da je začetna vrednost hitrosti učenja manj pomembna. Iz tega razloga je v vseh primerih nastavljena na isto vrednost, spreminjali pa smo zmanjševanje hitrosti učenja. Rezultati so predstavljeni v preglednici 5.14 in na sliki 5.13.

Preglednica 5.14: Rezultati različnega zmanjševanja hitrosti učenja

Hitrost učenja	Zmanjševanje hitrosti učenja	HN [h]	Številka cikla
0,007	0,0001	11,97	57
0,007	0,001	11,78	83
0,007	**0,01**	**11,75**	**585**
0,007	0,1	12,90	?

Iz preglednice 5.14 je razvidno, da ob hitrejšem zmanjševanju hitrosti učenja narašča število potrebnih ciklov, da pridemo do minimuma HN. Pri vrednosti zmanjševanja hitrosti 0,1 namreč te številke sploh ne poznamo. Razlog je preprosto to, da do tega pride po zelo velikem številu ciklov, kar pa pomeni izjemno dolgotrajno učenje. Več informacij o učenju pa razberemo iz slike 5.13.



Slika 5.13: Primerjava različnih vrednosti zmanjševanja hitrosti

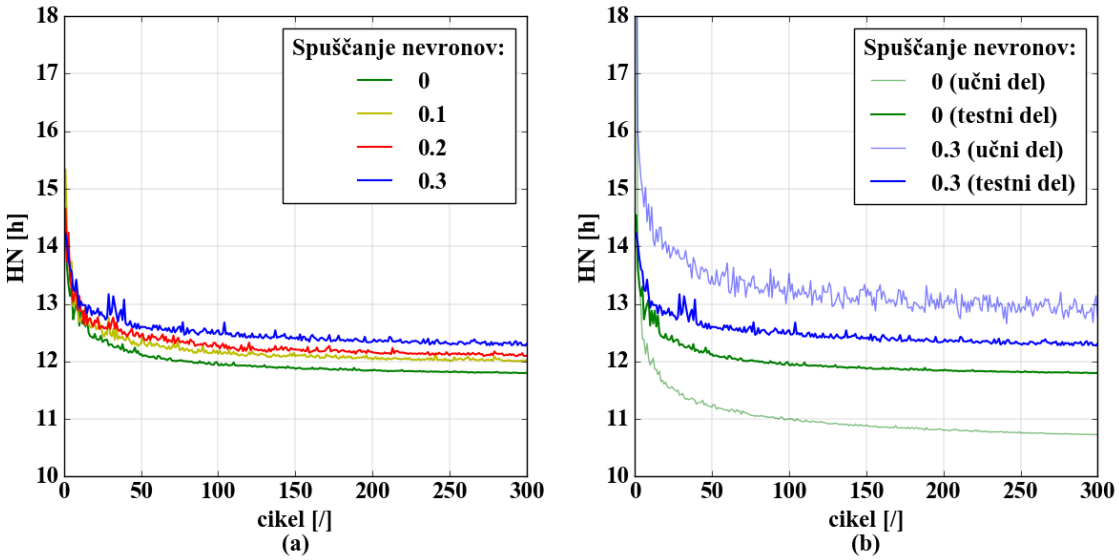
Na sliki 5.13 so prikazane vrednosti HN za testni del podatkov pri različnih vrednostih zmanjševanja hitrosti učenja. Pri prevelikem zmanjševanju opazimo, da funkcija sicer ne niha, a pada prepočasi. Pri premajhnem zmanjševanju pa je rezultat podoben, kot če zmanjševanja sploh nimamo in hitro pride do prenasičenja. Za najboljšo izbiro se izkaže srednja pot, kjer je vrednost 0,01. Ta HN precej hitro zmanjša, nato pa še naprej konstantno pada in pri tem ne niha. Mirna krivulja je pomembna, saj se zmanjša tveganje, da učenje ustavimo ravno ob nepravilnem trenutku.

Odločili smo se, da kljub dolgotrajnemu učenju izberemo hitrost učenja 0,007 in zmanjševanje hitrosti 0,01. Mirna krivulja in predvidljiv rezultat sta bila na tej točki za nas velikega pomena. Izbira števila ciklov, pri katerem smo se odločili učenje ustaviti, je 600.

Izbire omejitve uteži in izpuščanja nevronov smo se lotili, čeprav zaradi zmanjševanja hitrosti učenja težav s prenasičenjem nimamo. Najprej smo testirali izpuščanje nevronov, rezultati pa so prikazani v preglednici 5.15 in na sliki 5.14.

Preglednica 5.15: Rezultati izpuščanja nevronov

Izpuščanje nevronov [/]	HN [h]
0	**11,75**
0,1	11,95
0,2	12,04
0,3	12,23



Slika 5.14: Primerjava različnih vrednosti izpuščanja nevronov

V preglednici 5.16 so rezultati omejevanja uteži. Kažejo na to, da omejevanje na napovedovanje ne vpliva. Razlog je, da naša mreža nima težav s prevelikim prileganjem podatkom. Regularizacijske metode posledično nimajo velikega vpliva na rezultate. Preglednica 5.15 jasno kaže na to, da izpuščanje nevronov negativno vpliva na

napovedovanje. Do podobnih zaključkov pridemo tudi ob analizi grafa (a) slike 5.14. V poglavju *3.1.13 Izpuščanje nevronov* je že omenjeno, da je ta tehnika pomembna za preprečevanje prenasičenja. Napovedi, kjer ni prišlo do prenasičenja, lahko z večjo gotovostjo generaliziramo. To pa dobro prikazuje graf (b) na sliki 5.14. Mreža brez izpuščanja nevronov ima na učnem delu podatkov precej nižji HN kot na testnem delu. Model torej za nove podatke ne deluje tako dobro kot za podatke, na katerih se je učil. Mreža z izpuščanjem nevronov pa ima rezultat na testnem delu podatkov celo boljši kot na učnem delu. To pomeni, da se tak model brez težav generalizira. Kljub temu spoznanju sposobnost generalizacije ocenjujemo na podlagi testnih podatkov HN. Pri teh je še vedno bolje deloval model brez izpuščanja nevronov. Odločili smo se torej, da nevronov ne bomo izpuščali.

Praktično identični rezultati za različne vrednosti omejitev uteži pomenijo, da velikih uteži pri nas v mreži ni. Velike uteži so ponavadi posledica velike hitrosti učenja. V primeru, da hitrosti učenja ne bi zmanjševali in bi izbrali hitro učenje, pa bi omejevanje uteži najverjetneje pomagalo. Tudi če bi velikost omejitve zmanjšali pod 1, bi verjetno prišlo do drugačne napovedi. Ocenjujemo pa, da se napovedovanje v tem primeru ne bi izboljšalo, zato z manjšo vrednostjo nismo poskusili.

Preglednica 5.16: Rezultati omejevanja uteži

Omejitev uteži	HN [h]
1	11,75
10	11,75
100	11,76
∞	**11,75**

Omejevanja uteži na podlagi rezultatov v preglednici 5.16 torej nismo uporabili v končni mreži.

Končna mreža je tako vsebovala vhodni nivo z 249 vhodi, dva skrita nivoja s 500 in 2000 nevroni ter izhodni nivo z enim nevronom. V skritih nivojih nevroni uporabljajo ReLU, na izhodu pa aktivacijsko funkcijo SeLu. Uporabljena funkcija napake je Huberjeva napaka. Optimizacijski algoritem je Adamax s hitrostjo učenja 0,007 in zmanjševanjem 0,01. Porazdelitev uteži na začetku je enakomerna. Izpuščanje nevronov ter omejevanje uteži ni uporabljeno. Učenje poteka 600 ciklov pri velikosti paketa 150.

6 Rezultati in diskusija

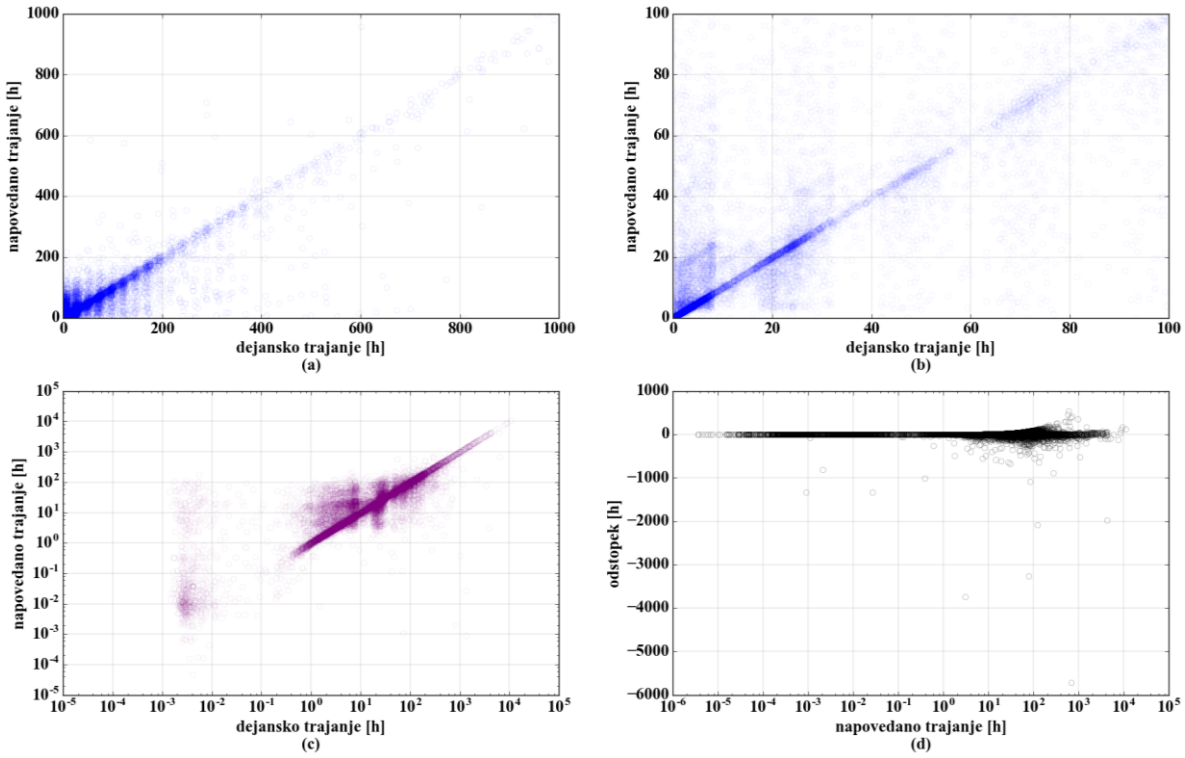
Rezultat našega dela so napovedane vrednosti. Poglejmo, kako se naše napovedi primerjajo z napovedmi enostavnih pripisovalnih metod, opisanih v poglavju *2.2.3 Metode odpravljanja*. Preglednica 6.1 prikazuje sposobnost napovedovanja našega modela ter napovedane vrednosti primerja z vrednostmi, pridobljenimi z metodama pripisovanja povprečne vrednosti in pripisovanja mediane.

Preglednica 6.1: Primerjava z drugimi pripisovalnimi metodami

	Pripisovalnje z nevronsko mrežo	Pripisovanje povprečne vrednosti	Pripisovanje mediane
PAN [h]	12,25	73,95	53,47
PKN [h^2]	5313,1	72633,2	74997,2
R2	0,93	0	-0,03
Kosinusna podobnost	0,96	0,2	0,2
Evklidska razdalja [h]	9608,4	35525,7	36099,2

Najprej analizirajmo rezultate svojega dela. Povprečna absolutna napaka znaša dobrih 12 ur, povprečna kvadrirana napaka pa dobrih 5300 ur^2. Same po sebi je te vrednosti težko oceniti, saj je interval, znotraj katerega napovedana trajanja nastopajo, zelo velik. Več nam pove vrednost R2, ki govori o deležu variabilnosti trajanja, ki je pojasnjen s strani modela. Naš model razlaga 93 % variabilnosti dejanskih trajanj. Drugi dve vrsti pripisovanja, ki pripišeta vsem praznim celicam enako vrednost, variabilnosti dejanskih trajanj jasno ne pojasnjujeta. Vrednosti R2 za ti dve metodi sta zato izjemno majhni. Vrednosti PAN in PKN pa sta zanju, v primerjavi z našimi napovedmi, precej večji. PAN in PKN za obe enostavni metodi pripisovanja nista enaki. Pripisovanje mediane je boljše s stališča PAN, saj ta manj penalizira velike odstopke. Pripisovanje povprečne vrednosti pa je boljše s stališča PKN, saj ta manj penalizira manjše odstopke. S kosinusno podobnostjo in Evklidsko razdaljo smo primerjali vektorje še s stališča smeri in magnitude. Približevanje kosinusne podobnosti številu 1 pomeni boljši rezultat, Evklidsko razdaljo pa želimo čim manjšo. V obeh primerih se naše pripisovanje izkaže za mnogo boljše.

Vidimo, da je naš način zmanjševanja nepopolnosti podatkov s stališča točnosti boljši od enostavnih pripisovalnih metod. Vizualizacije na sliki 6.1 pa prikazujejo, kako sposoben je za napovedovanje.



Slika 6.1: Primerjava napovedanih trajanj z dejanskimi v (a), (b) in (c) ter odstopki v (d)

Grafa (a) in (b) na sliki 6.1 prikazujeta relacijo med dejanskimi in napovedanimi trajanji. V idealnem primeru bi bile napovedi enake dejanskim trajanjem. Na grafu bi v takem primeru videli vse vrednosti na simetrali lihih kvadrantov. Višja kot je vrednost R2, bolj so točke blizu te simetrale. Na teh dveh grafih opazimo tudi vzorce. Na grafu (a) se pojavljajo navpične linije vsakih 24 ur. Pričakovali smo vodoravne linije, ki bi jih pojasnila lastnost značilke o planiranih trajanjih. Ta je natančna le na 24 ur, zato bi lahko napovedi »vlekla« k tem vrednostim. Vendar pa vodoravnih linij ni opaziti, kar pomeni, da značilka o planiranem trajanju ni problematična. Navpične linije pa ne morejo biti posledica značilk, temveč proizvodnega procesa. Razlog, da se vzorci vidijo, je, da je takih trajanj več in ne to, da je varianca napovedi tam večja. Na to dejstvo kaže že slika 5.1, potrjuje pa ga tudi graf (d), kjer navpičnih vzorcev ne vidimo.

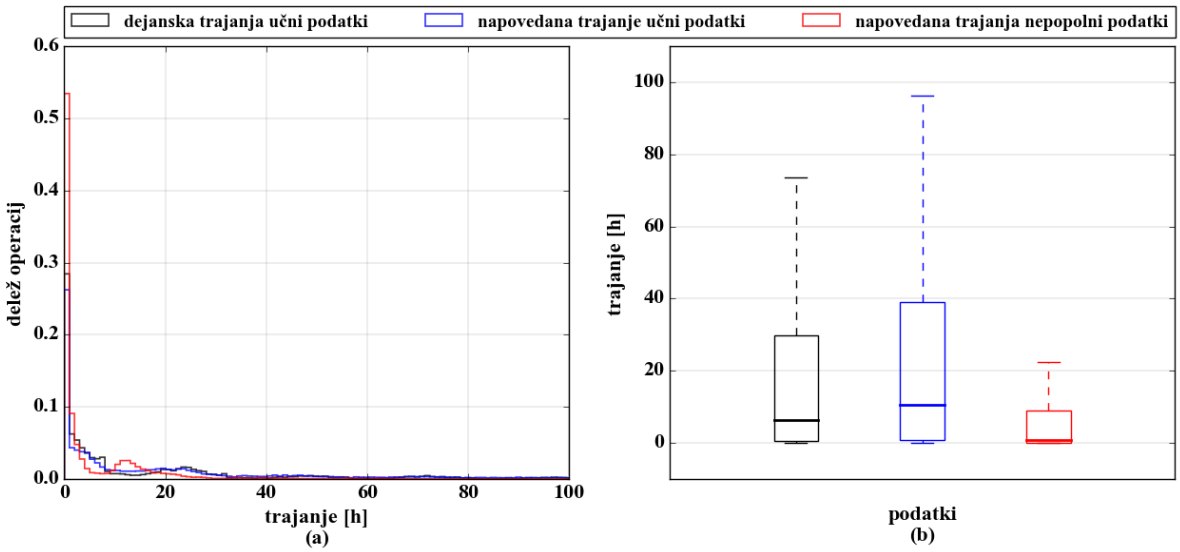
Na grafu (b), kjer so prikazana trajanja desetkrat krajša, pa opazimo navpične vzorce, ki se ločujejo vsakih 8 ur. Operacij s trajanji med 0 in 8 urami je veliko ter jih v glavnem napovedujemo predolge. Trajanj v intervalu od 8 do 16 ur je manj, naše napovedovanje pa ni več tako očitno predolgo. V naslednjem intervalu od 16 do 24 so napovedi pogosto prekratke, v intervalu od 24 do 32 ur pa ponovno večinoma predloge. Ta vzorec se nadaljuje tudi naprej. Ocenjujemo, da je obnašanje tako zato, ker zaposleni začete operacije, če je le mogoče, končajo znotraj svoje izmene (znotraj osmih ur). Število operacij s trajanji od 24 do 32 ur je spet povečano, saj se delavci vračajo na svoja delovna

mesta in zaključujejo nekončane operacije. Okrog mejnika 24 se število trajanj poveča tudi zato, ker morajo biti nekatere operacije končane znotraj 24 ur, kot je to planirano.

Graf (c) prikazuje enako odvisnost kot grafa (a) in (b), le da sta osi v logaritemski skali. Vidimo lahko, da so napovedi do dejanskega trajanja približno 10^1 ure večinoma predolge. Kratka trajanja naš model torej napoveduje predolga. V kombinaciji z grafom (d) pa za interval od 10^1 do 10^3 ur opazimo ravno obratno obnašanje. Odstopek smo definirali kot razliko med napovedanim in dejanskim trajanjem, kar pomeni, da se pojavljajo večinoma prekratke napovedi. Zanimivo je, da operacij s trajanji v intervalu od 10^{-2} do 10^{-1} ure praktično ni. Ocenjujemo, da so to operacije, katerih trajanja so prekratka za beleženje na panelih in prekompleksne za avtomatsko beleženje. Na grafu (c) prvič opazimo tudi vodoravne linije, ki so verjetno posledica pristranskega napovedovanja. Konkretnega vzroka teh linij ne poznamo.

Iz prej opisanega je jasno, da je obnašanje naših napovedi heteroskedastično. Na grafih je mogoče opaziti različne variance na različnih intervalih trajanj. Razlogi za predolgo in prekratko napovedovanje so večinoma že opisani. Na grafu (d) pa lahko opazimo še eno značilnost. Vse operacije, pri katerih je odstopek okrog 1000 ur ali več, imajo napovedi prekratke. Predvidevamo, da je razlog za to odsotnost informacij o zastoju. Le 10 takih operacij z največjimi odstopki je k rezultatu PAN v preglednici 6.1 prineslo približno 1,3 ure.

Slika 6.2 je namenjena primerjavi trajanj v učnih podatkih ter napovedanih trajanj, ki bodo zapolnila prazne celice v nepopolnih podatkih.



Slika 6.2: Porazdelitev trajanj različnih nizov podatkov

Iz slike 6.2 (a) se zdi, da se porazdelitvi dejanskih in napovedanih trajanj učnih podatkov precej ujemata. Graf (b) kaže na to, da so slednja gledano generalno nekoliko daljša. Precej pa se razlikuje porazdelitev napovedi za nepopolne podatke. Graf (a) kaže na to, da se nihaje namesto pri 24 urah zgodi nekoliko prej. Tudi delež trajanj med 0. in 1. uro je večji.

Takih trajanj je v učnih podatkih nekaj manj kot 30 %, pri napovedanih trajanjih za nepopolne podatke pa nekaj več kot 50 %. Iz grafa (b) na sliki 6.2 vidimo, da so napovedana trajanja za nepopolne podatke precej kratka. Razlog za to je, da je informacij o številu zastojev v teh podatkih mnogo manj. Predvidevamo torej, da je za približno 12 % operacij napovedano trajanje prekratko. Z gotovostjo torej ni mogoče trditi, da so rezultati o kakovosti napovedi iz preglednice 6.1 točni tudi za nepopolne podatke. Pomembno pa je razumeti, da rezultati niso taki, kot so le zaradi izbrane metode napovedovanja. Izkaže se, da so v večji meri odvisni od podatkov, na podlagi katerih se vrednosti pripisuje.

Kvantificirano primerjavo, kot jo podaja preglednica 6.1, je za nepopolne podatke težje podati, saj gre za vektorje različnih dolžin. Zavedati se moramo, da ujemanje teh porazdelitev ni nujno. Model je bil dobro naučen, kar lahko sklepamo iz primerjave učnih podatkov. Naš cilj pa je bil preslikati vzorce iz učnih podatkov na nepopolne podatke. Podrobna primerjava porazdelitev ni smiselna, saj gre za različna niza podatkov z drugimi primeri oziroma operacijami. Pri zmanjševanju nepopolnosti podatkov je ključno to, da ni prišlo do prevelikega prileganja podatkom in s tem vnašanja pristranskosti v podatke. To smo zagotovili z zmanjševanjem števila značilk, zmanjševanjem hitrosti učenja in s preverjanjem z omejevanjem uteži in izpuščanjem nevronov.

Napovedovanje manjkajočih trajanj je bilo še posebej težavno zaradi specifik proizvodnje. Pri individualni in maloserijski proizvodnji si težko pomagamo s trajanji podobnih izdelkov, saj te večinoma ne obstajajo. V podjetju Litostroj Power v primeru, da jih zanima manjkajoče trajanje operacije, uporabijo kar trajanje med začetki operacij na določenem delovnem mestu. Ta približek je sicer lahko za določene primere zadovoljiv, a se ne more primerjati z napovedmi našega modela. Na to dejstvo kaže že preglednica 5.2. Prepoznati je namreč potrebno kompleksne vzorce v podatkih, kar pa presega človeške sposobnosti.

Implementacija takega napovedovanja v podjetju ne bi bila zapletena, dokler ostajajo informacijski sistemi in baza podatkov v enaki obliki. Sam proces je mogoče popolnoma avtomatizirati ter ga razširiti tudi na ostale atribute, kjer najdemo manjkajoče vrednosti. Pomembna lastnost takega napovedovanja je, da kljub njegovi kompleksnosti lahko poteka na osebnem računalniku z minimalno intervencijo usposobljenega osebja. Podjetju posledično ne bi bilo treba dodatno investirati v novo opremo.

7 Zaključki

Rezultat našega dela so podatki z zmanjšano nepopolnostjo. Manjkajoče vrednosti nepopolnega niza podatkov podjetja Litostroj Power smo zapolnili z napovedmi modela umetne nevronske mreže. V spodnjem seznamu so zapisane ugotovitve, dosežki in spoznanja, pridobljena v okviru tega dela:

1) Definirali in opisali smo kakovost podatkov, s poudarkom na njihovi nepopolnosti. Primerjali smo različne pripisovalne metode za zmanjševanje nepopolnosti podatkov ter ugotovili, da je napovedovanje manjkajočih vrednosti s pomočjo nevronskih mrež smiselno.
2) Raziskali smo metodo strojnega učenja, imenovano nevronske mreže, jo umestili v prostor ter raziskali njeno delovanje. Ugotovili smo, da lahko s številnimi hiperparametri bistveno vplivamo na način napovedovanja in da so nevronske mreže za napovedovanje trajanj z namenom zmanjšanja nepopolnosti podatkov primerne.
3) Spoznali smo, da narava maloserijske in individualne proizvodnje zaradi majhnega števila podobnih proizvodov otežuje napovedovanje trajanj operacij.
4) Pokazali smo, da se delovna mesta v proizvodnji med seboj precej razlikujejo, tako iz stališča vrste opravljenih operacij, števila operacij, kot tudi porazdelitve trajanj.
5) Pokazali smo, da planirano trajanje, trajanje med začetki izvajanja na delovnem mestu, informacije o delovnem mestu, zastojih ter dnevu in mesecu začetka dela izkazujejo korelacije z dejanskim trajanjem operacije.
6) Ugotovili smo, da vključevanje značilk z informacijami o kvartilih trajanja na določenem delovnem mestu ne pripomore k napovedovanju mreže. Predstavitev informacij o porazdelitvi vrednosti določene kategorije na ta način ni primerna.
7) Ugotovili smo, da je težko oceniti, katere informacije in v kakšni obliki je smiselno vključiti v regresijski model. Zdi se, da je za potrditev kakršnih koli predpostavk pri tem delu treba izvesti preskus.
8) Rezultati kažejo na to, da umetno povečevanje količine podatkov v našem primeru ne izboljšuje napovedovanja.
9) Pokazali smo, da transformacija trajanj iz linearnega časovnega prostora v druge časovne prostore v našem primeru ne izboljšuje napovedovanja.

10) Ugotovili smo, da med Huberjevo napako in povprečno absolutno napako ni velikih razlik, kadar ju uporabljamo kot funkcijo napake. Zdi se, da se bolje obnese Huberjeva napaka, kadar je interval napovedovanih vrednosti velik.
11) Spoznali smo, da daje enakomerna porazdelitev vrednosti začetnih uteži dobre eksperimentalne rezultate. Prav tako smo videli, da ničelne vrednosti uteži ne delujejo dobro, kar se ujema z ugotovitvami drugih raziskovalcev.
12) Pokazali smo, da je pomembno, da se zaloga vrednosti aktivacijske funkcije ujema z intervalom, v katerem do le-te vrednosti prihajajo in iz nje izhajajo.
13) Ugotovili smo, da v našem primeru najboljše rezultate zagotavljajo široke, a plitke mreže.
14) Pokazali smo, da zmanjševanje hitrosti učenja pozitivno vpliva na čas učenja, prenasičenje in rezultat napovedovanja.
15) Ugotovili smo, da omejevanje uteži in izpuščanje nevronov ni smiselno v primeru, ko nimamo težav s prevelikim prileganjem podatkom.
16) Spoznali smo, da na sposobnost modela za napovedovanje močno vplivajo modelu zagotovljeni podatki in njihova oblika. Tudi optimizacija modela ima na napovedovanje vpliv, a je ta manjši in bolj omejen.
17) Pokazali smo, da napovedovanje z nevronskimi mrežami v številnih pogledih prekaša enostavne metode pripisovanja, ki so v praksi najpogosteje uporabljene.
18) Spoznali smo, da je vnašanje pristranskosti ena od glavnih težav pri zmanjševanju nepopolnosti podatkov. Na strani modela nevronske mreže lahko to do neke mere popravljamo, vendar se zdi, da so glavni vzrok za vnašanje pristranskosti v podatke podatki sami.

Z napovedovanjem trajanj in polnjenjem praznih celic v podatkih smo pokazali, da je metoda nevronskih mrež uporabna za zmanjševanje nepopolnosti podatkov. S pravilnim vključevanjem podatkov v model ter pravilnimi nastavitvami modela nevronske mreže je mogoče zmanjšati pristranskost, ki jo vnesemo v niz podatkov.

Predlogi za nadaljnje delo

Možnosti za izboljšanje napovedovanja modela je več. Lahko spremenimo vhodne podatke tako, da jim dodamo nove značilke ali jih predstavimo na drugačen način. Vhodne podatke lahko tudi izboljšamo. Z analizo MTBF je mogoče ugotoviti, ali so informacije o zastojih resnično nepopolne. Na ta način je mogoče približati porazdelitev napovedanih trajanj porazdelitvi učnih podatkov. Mogoče pa je izboljšati tudi model. Morda bi model, ki napoveduje kategorične vrednosti, deloval bolje. Operacije bi uvrščal v intervale trajanj, kamor najverjetneje spadajo. Združevanje modelov je prav tako način, s katerim je mogoče izboljšati napovedovanje. Združili bi lahko regresijski in klasifikacijski model. Združili bi lahko tudi modele, ki napovedujejo v različnih časovnih intervalih. En model bi napovedoval kratka trajanja brez zastojev, drug pa dolga in z zastoji. Takih možnosti je še mnogo.

Zmanjševanje nepopolnosti je bilo narejeno za en atribut. Pridobljena znanja in model lahko uporabimo tudi za napovedovanje manjkajočih vrednosti drugih atributov. To bi bilo smiselno zato, ker so nepopolni podatki vplivali tudi na naše delo. Zanimivo bi bilo videti, kakšno točnost napovedovanja trajanja operacij bi lahko dosegli, če bi pred tem zapolnili prazne celice drugih atributov, ki nam pri napovedovanju trajanj operacij lahko koristijo. Tak način dela je mogoče vključiti tudi v iterativni proces, kjer se vsako iteracijo izboljša napovedovanje manjkajočih vrednosti enega od atributov. Ideja je v tem, da bi se slabe napovedi v prvi iteraciji s povečevanjem iteracij izboljšale.

Model, uporabljen pri pisanju te magistrske naloge, bi se lahko uporabil tudi za napovedovanje trajanja še ne izvedenih operacij. Nastavitve modela in uporabljene značilke bi ostale praktično enake. Razlika je ta, da ne bi imeli informacij o zastojih, zato bi napovedi predvidevale obratovanje brez zastojev. Nekoliko bi lahko spremenili tudi predobdelavo podatkov. V takem primeru si ne bi želeli preslikati vzorca iz učnih na nepopolne podatke, temveč napovedovati dejanska trajanja operacij. To bi vplivalo predvsem na filtriranje podatkov.

Vidimo, da je možnosti za nadaljevanje raziskovanja na tem področju veliko. V prihodnosti bo količina podatkov, zajetih v proizvodnih sistemih, le še večja, s tem pa se bodo povečale tudi potrebe po točnih in zanesljivih metodah izboljševanja njihove kakovosti.

Literatura

[1] B. T. Hazen, F. K. Weigel, J. D. Ezell, B. C. Boehmke, R. V. Bradley: *Toward understanding outcomes associated with data quality improvement*. International Journal of Production Economics 193 (2017) str. 737–747.

[2] D. Loshin: *Monitoring Data Quality Performance Using Data Quality Metrics*. Informatica White Paper 1 (2016) str. 1–22.

[3] K. J. Nishanth, V. Ravi: *Probabilistic neural network based categorical data imputation.* Neurocomputing 218 (2016) str. 17–25.

[4] I. K. McDonough, D. L. Millimet: *Missing data, imputation, and endogeneity*. Journal of Economics 199 (2017) str. 141–155.

[5] C. K. Enders: *Multiple imputation as a flexible tool for missing data handling in clinical research*. Journal of the Korean Statistical Society, 98 (2017) str. 4–18.

[6] J. Brownlee: *How to Handle Missing Data with Python*. Dostopno na: https://machinelearningmastery.com/handle-missing-data-python/, ogled: 9. 1. 2018.

[7] U. Garciarena, R. Santana: *An extensive analysis of the interaction between missing data types, imputation methods, and supervised classifiers.* Expert Systems with Applications 89 (2017) str. 52–65.

[8] G. Beliakov, D. Gómez, S. James, J. Montero, J. T. Rodríguez: *Approaches to learning strictly-stable weights for data with missing values*. Fuzzy Sets Systems 325 (2017) str. 97–113.

[9] J. Kušar, T. Berlec: *Planiranje in krmiljenje proizvodnje, zapiski predavanj*. Fakulteta za strojništvo, Ljubljana, 2017.

[10] Y. Cheng, K. Chen, H. Sun, Y. Zhang, F. Tao: *Data and Knowledge Mining with Big Data towards Smart Production*. Journal of Industrial Information Integration, 1 (2017) str. 1–13.

[11] S. Sumathi, S. N. Sivanandam: *Introduction to Data Mining Principles Objectives.* SCI 20 (2006) 1–20.

[12] K. Hiovská, P. Koncz: *Application of Artificial Intelligence and Data Mining Techniques to Financial Markets*. ACTA VSFS 6 (2012) str. 62–76.

[13] A. Gulli, S. Pal: *Deep Learning with Keras*. Packt Publishing, Birmingham, 2017.

[14] B. Reagen, R. Aolf, P. Whatmough, G.-Y. Wei, D. Brooks: *Deep learning for computer architects*. Morgan&Claypool Publishers, California, 2012.

[15] G. Zaccone, M. R. Karim, A. Menshawy: *Deep Learning With TensorFlow*. Packt Publishing, Birmingham, 2017.

[16] W. Sarle: *Neural Network FAQ*. Dostopno na: ftp://ftp.sas.com/pub/neural/FAQ.html, ogled: 9. 1. 2018.

[17] J. Brownlee: *What Is the Difference Between a a Parameter and a Hyperparameter*. Dostopno na: https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/, ogled: 9. 1. 2018.

[18] I. Changhau: *Loss Functions in Artificial Neural Networks*. Dostopno na: https://isaacchanghau.github.io/2017/06/07/Loss-Functions-in-Artificial-Neural-Networks/, ogled: 9. 1. 2018.

[19] J Brownlee: *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. Dostopno na: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/, ogled: 9. 1. 2018.

[20] S. Ruder: *An overview of gradient descent optimization algorithms*. Eprint Arxiv 1 (2016) str. 1–14.

[21] J. D. McCaffrey: *Neural Network momentum*. Dostopno na: https://jamesmccaffrey.wordpress.com/2017/06/06/neural-network-momentum/, ogled: 9. 1. 2018.

[22] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth: *From Data Mining to Knowledge Discovery in Databases*. AI Magazine 17 (1996) str. 37–55.

[23] F. Gullo: *From Patterns in Data to Knowledge Discovery: What Data Mining Can Do.* Physics Procedia 62 (2015) str. 18–22.

[24] L. Zhou, S. Pan, J. Wang, A. V. Vasilakos: *Machine learning on big data: Opportunities and challenges*. Neurocomputing 237 (2017) str. 350–361.

[25] G. Köksal, I. Batmaz, M. C. Testik: *A review of data mining applications for quality improvement in manufacturing industry*. Expert Systems with Applications 38 (2011) str. 13448–13467.

[26] R. J. Sethi, Y. Gil: *Scientific workflows in data analysis: Bridging expertise across multiple domains.* Future Generation Computer Systems 75 (2017) str. 256–270.

[27] Y. Shevchuk: *Hyperparameter optimization for Neural Networks*. Dostopno na: http://neupy.com/2016/12/17/hyperparameter_optimization_for_neural_networks.html, ogled: 9. 1. 2018.

[28] R. Sint, S. Schaffert, S. Stroka, R. Ferstl: Combining unstructured, fully structured and semi-structured information in semantic wikis. CEUR Workshop Proceedings 464 (2009) str. 73–87.

[29] S. Sherpa: *Structured and Unstructured Data : What is it?* Dostopno na: https://sherpasoftware.com/blog/structured-and-unstructured-data-what-is-it/, ogled: 23. 1. 2018

[30] D. Rumsey: *Statistics For Dummies, 2nd Edition*. Wiley Publishing, Hoboken, 2009.

[31] J Brownlee: *Why One-Hot Encode Data in Machine Learning?* Dostopno na: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/, ogled: 23. 1. 2018.

[32] J. Dougherty, R. Kohavi, M. Sahami: *Supervised and Unsupervised Discretization of Continuous Features*. Machine Learning PTIC 1 (1995) str. 194–202.

[33] S. Y. Jiang, X. Li, Q. Zheng, L. X. Wang: *Approximate equal frequency discretization method*. GCIS 3 (2009) str. 514–518.

[34] P. Butala: *Flexible Manufacturing Systems, zapiski predavanj*. Fakulteta za strojništvo, Ljubljana, 2017.

[35] J. Bozja: *Vodenje proizvodnje na osnovi standardnih kazalnikov: diplomsko delo*. Ljubljana, 2014.

[36] H. Datonis: *The Next Step: IoT for SCADA, ERP & MES*. Dostopno na: http://altizon.com/iot-for-scada-erp-mes/, ogled: 23. 1. 2018.

[37] D. Bračun: *Senzorji in aktuatorji, zapiski predavanj*. Fakulteta za strojništvo, Ljubljana, 2017.

[38] Litostroj Power d.o.o.: *Company profile*. Litostroj Power d.o.o., Ljubljana, 2017.

[39] S. Kastelic: *Prenova ERP sistema v podjetju Litostroj E.I.: diplomsko delo*. Ljubljana, 2007.

[40] D. P. Kingma, J. L. Ba: *Adam: A Method for Stochastic Optimization*. ICLR 1 (2014) str. 1–15.