Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Denis Kotnik

Prilagodljivo kratkoročno napovedovanje lokalnih vremenskih parametrov

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Mentor: doc. dr. Matjaž Kukar

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Kandidat naj preizkusi, ali je možno z uporabo regresijskih metod strojnega učenja, predvsem nevronskih mrež in linearne regresije, popraviti oziroma izboljšati kratkoročne napoved hitrosti vetra za zelo omejena področja v Sloveniji. Pri delu naj uporabi podatke meritev cestno-vremenskih postaj Družbe za avtoceste v Republiki Sloveniji in kratkoročne napovedne podatke sistema INCA Agencije Republike Slovenije za okolje. Razvito metodologijo naj preveri tudi na drugih, bolj predvidljivih vremenskih parametrih (temperatura). Rezultate naj ustrezno ovrednoti v primerjavi z uveljavljenim meteorološkim modelom.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Denis Kotnik, z vpisno številko **63100078**, sem avtor diplomskega dela z naslovom:

Prilagodljivo kratkoročno napovedovanje lokalnih vremenskih parametrov

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Matjaža Kukarja,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 1. septembra 2014 Podpis avtorja:

Zahvala

Zahvaljujem se mentorju doc. dr. Matjažu Kukarju za vodenje po čim pravilnejši poti diplomskega dela, za strokovno pomoč, vložen čas in za vse, kar je pripomoglo k zaključku diplomske naloge.
Za navdih nad delom sem hvaležen oddelku Okolje v podjetju CGS plus. Še posebej se zahvaljujem dr. Roku Kršmancu za vložen čas v pomoč meni, pa tudi za to, da mi je povedal za Tagorjeve besede. O teh sem veliko razmišljal. Brez ustreznih podatkov delo ne bi nastalo, zaradi česar se zahvaljujem Agenciji Republike Slovenije za okolje in Direkciji Republike Slovenije za avtoceste. Navsezadnje se iz srca zahvaljujem svoji družini, ki mi je s podporo in ljubeznijo omogočila študij.

Najdaljša je pot, ki te pripelje najbližje k sebi,
in najtežja je vaja, ki rodi najpreprostejši napev.

—Rabindranath Tagore

Kazalo

Slike

Seznam uporabljenih kratic

kratica	angleško	slovensko
CRISP-DM	Cross Industry Standard Process for Data Mining	industrijski standard procesa odkrivanja znanj iz podatkov
ALADIN[1]	Aire Limitée, Adaptation Dynamique, Développement International	omejeno območje - dinamična adaptacija, mednarodno sodelovanje
INCA[2]	Integrated Nowcasting Through Comprehensive Analysis	integrirano kratkoročno napovedovanje s celovito analizo
CVP	road weather station	cestnovremenska postaja

[1]ALADIN - projekt, katerega cilj je z modeliranjem izboljšati meteorološke, hidrološke in okoljske napovedne storitve in opozorila.

[2]INCA - sistem za analizo in kratkoročno napovedovanje vremenskih parametrov. Cilj sistema je izboljšanje zdajšnjih (do 4 ure) ter zelo kratkoročnih (do 12 ur) in dodajanje vrednosti oziroma popravkov kratkoročnih (do 72 ur) numeričnih napovedi že obstoječih modelov.

Povzetek

Cilj diplomskega dela je preizkusiti, ali lahko z uporabo regresijskih metod ali umetnih nevronskih mrež popravimo oziroma izboljšamo napoved hitrosti vetra za določeno območje Slovenije. Za ta namen smo uporabili podatke meritev, ki smo jih pridobili s cestnovremenskih postaj *Družbe za avtoceste v Republiki Sloveniji* in napovedne podatke sistema *INCA-CE Agencije Republike Slovenije za okolje*. Omenjene podatke smo za potrebe modeliranja primerno pripravili s programskim jezikom *Python*. V programskem okolju *R* smo z uporabo omenjenih podatkov ustvarili parameter *napaka*, katerega smo definirali kot razliko med napovedano vrednostjo za čas $t + \Delta t$ in izmerjeno vrednostjo v času $t + \Delta t$. Slednjega smo z uporabo regresijskih metod in umetnih nevronskih mrež napovedovali za do 11 ur vnaprej, ga odšteli ustreznim prvotnim napovedim, rezultate pa vizualizirali. Ugotovili smo, da lahko z uporabo omenjenih metod popravimo napoved hitrosti vetra za leto 2014 za do 1 m/s v prvih napovednih urah. Prav tako smo ugotovili, da kompleksnost umetnih nevronskih mrež v naši domeni ne opraviči uporabo teh oziroma, da lahko za te namene uporabimo enostavnejše regresijske metode.

Ključne besede: nevronska mreža, napaka, podatki, vremenski parameter, napoved, hitrost vetra, vrednost, cestnovremenska postaja, meteorologija.

Abstract

The basic goal of this research is to explore if we could improve the weather forecast parameters such as wind speed for specific area of Slovenia, with the regression methods and artificial neural networks. We utilized measurements data, which we obtained from road-weather stations of *Direkcija Republike Slovenije za avtoceste* and forecast data of *INCA-CE system of Agencija Republike Slovenije za okolje*. We used the Python programming language for the purpose of data preparation process which is needed for modeling process. In the R programming environment we created the parameter *error*, which we defined as the difference between the predicted value for time $t + \Delta t$ and the measured value in time $t+\Delta t$. We predicted the error for subsequent 11 hours with the usage of regression methods and artificial neural networks, then we subtracted it from the INCA-CE predictions and visualised the results. We came to the conclusion that wind speed forecasts for 2014 could be corrected by up to 1 m/s in the early predicted hours with the usage of mentioned methods. We also have found that is better to use simpler regression methods than neural networks due to the complexity of those.

Keywords: neural network, error, data, weather parameter, forecast, wind speed, value, road weather station, meteorology.

Poglavje 1

Uvod

V današnjih časih so računalniške metode ključnega pomena pri napovedovanju vremenskih parametrov. Slednje se dosega s pomočjo fizikalno-matematičnih modelov, ki so bili razviti s strani meteorologov in fizikov, pa tudi z metodami strojnega učenja [17].

V Sloveniji zelo kratkoročne vremenske napovedi modelira sistem *INCA*, katere smo v kombinaciji s podatki pridobljenimi s cestnovremenskih postaj *Družbe za avtoceste v Republiki Sloveniji* uporabili za modeliranje.

V diplomskem delu smo se osredotočili na kvaliteto kratkoročnega in zelo lokalnega napovedovanja hitrosti vetra oziroma izboljšanje le te z uporabo metod strojnega učenja (nevronskih mrež, regresijskih metod ...). Opravili smo analizo podatkov, vizualno in kvantitativno.
Za izbor in pripravo ustreznih podatkov smo uporabili programski jezik *Python*, analize ter modeliranje pa smo opravljali v programskem okolju *R*, kjer so z ustreznimi programskimi paketi podprte ustrezne tehnike modeliranja, opisane v nadaljevanju.
Rezultate uporabe nevronskih mrež smo primerjali z rezultati uporabe linearne regresije in ugotovili, da kompleksnost in prilagodljivost nevronskih mrež v naši domene ne opraviči uporabe le teh.

V poglavju 2 je najprej opisana meteorologija, njene vrste ter katere metode se uporabljajo pri napovedi vremenskih parametrov. Podrobneje sta opisana tudi v te namene v Sloveniji najpogosteje uporabljena modela ALADIN in INCA. V istem poglavju je razloženo kakšni so bili prvotno pridobljeni podatki, v okviru tega pa so opisane tudi cestnovremenske postaje, katere so bile glavni vir podatkov uporabljenih za namene tega dela.

V *poglavju 3* je opisan programski jezik Python ter programsko okolje R, umetne nevronske mreže, metodologija CRISP-DM in ostale *tehnike*, katere so bile uporabljene ob nastajanju tega dela.

Četrto poglavje je sestavljeno iz sklopa *Razumevanja problema*, kjer je opisan problem ter možnosti uporabe rešitev tega v industriji. V sklopih *Razumevanje podatkov* in *Priprava podatkov* so podrobneje opisani podatki ter način priprave teh s programskih jezikom *Python*. Zatem v sklopu *Modeliranje in evalvacija* sledi opis načina učenja metod, nastavitev parametrov in način testiranja v programskem okolju *R*.

Peto poglavje prikazuje analizo in vizualizacijo uporablenih podatkov in rezultatov uporabljenih metod. Osredotočili smo se na primerjavo rezultatov linearne regresije in nevronskih mrež, primerjali pa smo tudi rezultate različno nastavljenih parametrov za uporabo slednjih.

Diplomsko delo se zaključi z razlago najpomembejših ugotovitev ter sklepov tega dela v *poglavju 6*. Sledi mu še opis morebitnih nadaljnih raziskav ter morebitna implementacija rešitve v industrijo.

Poglavje 2

Napovedovanje vremenskih parametrov

2.1 Meteorologija

Meteorologija je veda, ki raziskuje procese in pojave v atmosferi ter napoveduje vreme. Ime je uvedel grški filozof Aristotel že približno 300 let pred našim štetjem. Meteorologija je del geofizike, ki skupaj z geologijo, geodezijo in geografijo sestavlja skupino geo-znanosti, katera pa proučuje naš planet. Po eni strani vsebuje teoretične veje meteorologije in z njo povezane klimatologije ter po drugi strani nekatere veje operativne, vsakodnevne meteorološke dejavnosti. Njeni izsledki so neposredno uporabni v vsakdanjem življenju.

Iz želje po uporabnosti so se tudi razvile posebne specializirane veje meteorologije [22]. Teoretični veji meteorologije sta predvsem *dinamična* in *fizikalna* meteorologija. Prva, *dinamična meteorologija* z uporabo splošnih zakonov dinamike, prilagojenih za ozračje, razlaga dogajanja v njem. Predvsem z obravnavo sil, gibanj, energijskih prehodov in drugih procesov razlaga vzroke za gibanje zraka in z njim povezane vremenske spremembe. *Fizikalna meteorologija* se ukvarja s termodinamiko ozračja, s sevanjem, z dogajanji v oblakih in med oblačnimi delci, z električnimi in optičnimi pojavi v ozračju

in podobno [22].

Klimatologija je sorazmerno stara veda, ki opisuje in proučuje klimo. Ukvarja se s študijem vremenskih elementov in pojavov ter njihovimi statističnimi značilnostmi v daljšem časovnem obdobju. Razlaga tudi procese, ki klimo oblikujejo ter obravnava vzroke in posledice klimatskih sprememb. Osnovno klimatsko opazovalno obdobje naj bi bilo dolgo vsaj 30 let in je tudi osnova klimatskih napovedi.

Prognostična meteorologija napoveduje vreme. Pri tem se je v preteklosti naslanjala na metode sinoptične meteorologije. Na osnovi istočasnih opazovanj spremlja vremenska dogajanja nad velikim delom zameljske površine in omogoča vremenske napovedi z ekstrapolacijo premikov in dogajanj v ozračju. V drugi polovici XX. stoletja pa je sinoptično metodo presegla *metoda numeričnega napovedavanja vremena*: z numeričnimi modeli izračunava resitve enačb, ki opisujejo dogajanja v ozračju, in tako na osnovi izmerjenih podatkov daje osnovo za napoved vremena.

Nekatere operativne veje, ki pomagajo opazovati vreme, so v pomoč teoretičnim vejam meteorologije. Ena se ukvarja z *opazovanji in merilnimi inštrumenti v meteorologiji*. Razvoj ustreznih inštrumentov mora biti tak, da z njimi izmerimo vrednosti, ki so res reprezentativne za določeno uporabo. Tako je npr. termometer z zelo majhno teplotno kapaciteto uporaben za merjenje drobnih temperaturnih sprememb v turbolentnem okolju, tisti z veliko toplotno kapaciteto pa nudi reprezentativno vrednost temperature za neko širše območje. Nekatere merilne tehnike in metode obravnave in na njih temelječi podatki so sorazmerno zapletene in precej zaključene celote, zato sta npr. s*atelitska meteorologija* in *radarska meteorologija* že kar precej samostojni veji meteorologije [25].

Med *aplikativnimi vejami meteorologije* spada *biometeorologija*, ki preočuje vpliv meteoroloških parametrov na žive organizme, to je na rastline, živali in človeka. Z njo sta tesno povezani *medicinska meteorologija* ter *agrometeorologija*.

Meteorologija skuša pomagati z nasveti tudi prometu in prispeva k skrbi za

večjo varnost. Letalski promet, plovba na morju ter gradnja večjih objektov so tako odvisne tudi od *letalske* ter *pomorske* in *gradbene meteorologije*.

2.2 Metode napovedovanja

Napovedovanje vremenskih parametrov je ena najbolj znanih uporabnih nalog meteorologije.

Kmalu po prvi svetovni vojni je bil vremenoslovec Lewis RichardsonkLewis Fry Richardson (∗1881 — †1953) je bil angleški matematik, fizik, meteorolog in psiholog. prepričan, da lahko z matematiko napove vreme, ker v ozračju veljajo fizikalni zakoni. Toda formule so bile prezapletene in računanje s številkami tako zamudno, da so vremenske fronte že prešle, preden so napovedovalci lahko končali svoje računanje. Poleg tega je Richardson uporabljal vremensko odčitavanje, ki so ga opravljali v šesturnih razmikih. Z nastopom računalnikov pa je bilo dolgo izračunavanje mogoče pospešiti. Vremenoslovci so uporabljali Richardsonovo računanje in razvili zapleten številčni model – serijo matematičnih enačb, v katerih so vsi znani fizikalni zakoni, ki vplivajo na vreme.

Danes napovedovanju vremena složijo različne metode, vsem pa je skupno to, da so deterministične[1]. Predpostavljajo namreč, da je bodoče stanje v atmosferi odvisno od začetnega - sedanjega oziroma preteklega stanja. Med začetnim stanjem in stanjem atmosfere v prihodnosti so v tem primeru povezave, ki jih lahko izrazimo v obliki zakonov, postopkov, fizikalnih enačb in numeričnih algoritmov.
Različnih vremenskih sistemov (npr. ciklonov ali neviht) zaradi njihovih različnih značilnih življenskih časov in prostorskih razsežnosti ne moremo spremljati in napovedovati na enak način. Glede na čas vremenske napo-

[1]Determinizem - nauk, po katerem se vse razvija po objektivnih zakonih, brez naključij.

vedi lahko tudi značilnosti prognostičnega[2] procesa razdelimo na naslednje napovedi [25]:

- zdajšnje in zelo kratkoročne (do 6 ur),
- kratkoročne (do 36 ur),
- srednjeročne (do 96 ur) in
- dolgoročne (do 10 dni).

Podpora različnim vrstam prognostičnih procesov zahteva različen pristop, različne tehnike in tudi podatke - torej različno zasnovane tehnološke in druge podsisteme.

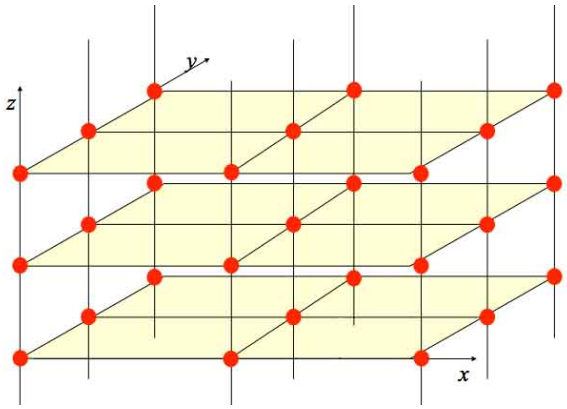
V Sloveniji se *Agencija Republike Slovenije za okolje* med drugim ukvarja tudi s stalnim spremljanjam podnebnih razmer in s prognozo vremenskih parametrov z uporabo modelov opisanih v nadaljevanju.

Sledi kratek opis *metode numeričnega napovedovanja vremena*, na podlagi katere delujeta modela ALADIN in INCA. Poleg te obstaja tudi *sinoptična*[3] *metoda za analizo in prognozo vremena*, ki je zgodovinsko prva deterministična metoda za napovedovanje bodočega stanja atmosfere. Slednja ima podsisteme kot so *sistem meritev, opazovanj in izmenjave podatkov*, *diagnoza stanja v atmosferi*, *prognoza bodočega stanja v atmosferi* in *interpelacija bodočih meteoroloških polj in oblikovanje napovedi vremena* [19].

Citirano iz revije *Vetrnica* ([33]):
Numeričnega modeliranja vremena si ne moremo zamisliti brez zmogljivih računalnikov, kajti število računskih operacij, ki jih je potrebno izvesti za simulacijo vremena s pomočjo numeričnega modela, je ogromno. Prve računalniške simulacije dogajanj v atmosferi so Edwarda N. Lorenza[4]*, ki jih je izvedel leta*

[2]Prognoza - napoved, predvidevanje, zlasti strokovnjaka.
[3]*Synopsis* v grščini pomeni pregled, pogled na celoto.
[4]Edward Norton Lorenz (∗1917 — †2008) je bil ameriški matematik, meteorolog in pionir teorije kaosa.

1964 na takrat najhitrejšem IBM računalniku, nameščenem na Tehnološkem institutu v Massachusett-u, napeljale k razvoju teorije kaosa. Ugotovil je, da majhna napaka, ki nastane pri shranjevanju vmesnih rezultatov simulacije z manjšim številom decimalnih mest, pripelje po določenem času do povsem druge vremenske napovedi. Odkril je tako imenovani metuljev efekt[5]*. Cilj numeričnega napovedovanja vremena je poznati tridimenzionalna polja različnih meteoroloških spremenljivk nad nekim območjem v točkah, ki predstavljajo pravilno mrežo (slika 2.1). Razporeditev točk v mrežo je določena z značilnostmi numerične prognoze, z ločljivostjo (horizontalno in vertikalno razdaljo med točkami) modelskega prostora in z značilnostmi koordinatnega sistema. Trenutno stanje analiziranega meteorološkega polja je torej predstavljeno s tridimenzionalno strukturo, matriko podatkov.*

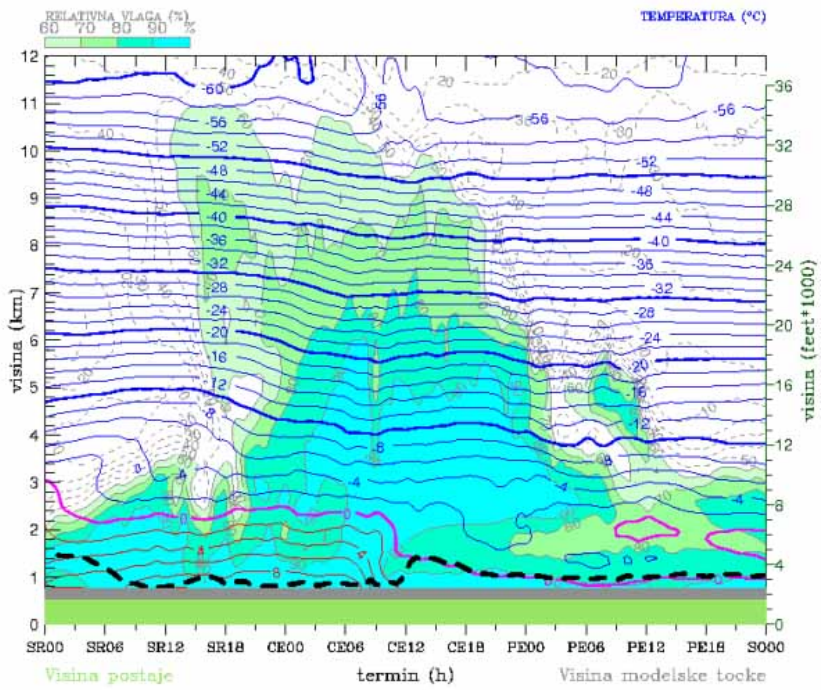


Slika 2.1: Tridimenzionalna struktura točk, s katerimi je predstavljeno stanje meteorološkega polja [35].

Prva naloga numerične analize je torej ta, da mora prostorsko nepravilno razporejene meritve preračunati v pravilno analitično mrežo. Pri tem se uporabljajo interpolacijski in statistični postopki. Druga naloga numerične analize pa je, da je treba v analizirana polja vključiti tudi nesinhrona opazovanja (na primer opazovanja s satelitov). Prvo nalogo opravljajo po vseh glavnih opazovalnih terminih, drugo pa ob prispetju novih podatkov. Obe

[5]Poenostavljena razlaga metuljevega efekta pravi, da lahko vsak že zelo majhen učinek sčasoma vodi v velike spremembe.

nalogi skupaj, poleg postopka kontrole, verifikacije in usklajevanja izmerjenih vrednosti, v okviru numerične analize (diagnoze) vremenskega stanja opravljajo tako imenovani *asimilacijski model*. Po opravljeni numerični analizi lahko tridimenzionalne matrike meteoroloških polj tudi izrišemo v obliki horizontalnih ali vertikalnih presekov, ki so v mnogočem enaki sinoptičnim kartam ali presekom [35, 25].



Slika 2.2: Primer vertikalnega časovnega preseka napovedi modela ALADIN za potek višinskega profila temperature (barvne krivulje) in relativne vlažnosti zraka (črtkane krivulje in obarvana območja). Vir slike: [24].

Citirano ([25]):

Numerična prognoza je zasnovana na postopku za reševanje sistema parcialnih diferencialnih enačb, ki opisujejo spremembe posameznih meteoroloških spremenljivk. Enačbe so zapis osnovnih fizikalnih zakonov (Newtonov zakon o ohranitvi energije, enačba stanja . . .) za pline (zrak in vodno paro), ki sestavljajo zemeljsko atmosfero. Skupini postopkov in algoritmov za reševanje tega sistema enačb pravimo meteorološki model. Rezultat numerične prognoze

so polja meteoroloških spremenljivk v prostoru (na izbranih nivojih) in ob izbranih časih prognoze. Za izdelavo vremenske napovedi (npr. za regijo veliko 100 x 100 km) in za nekaj dni vnaprej je potrebno, da meteorolog-prognostik dobljena prognostična polja interpretira, iz njih izlušči potek vremena in vremenske pojave in za ugotovljeno stanje napiše besedilo vremenske napovedi.
Z numerično analizo ni mogoče začeti takoj ob opazovalnem terminu, pač pa je potrebno počakati nekaj ur, da so podatki iz svetovne opazovalne mreže zbrani, preverji in izmenjani. Razen tega je postopek numerične analize in prognoze vremena računalniško zahteven, zanj je potrebno precej (do nekaj ur) računskega časa. Za mnoge uporabnike in za nekatera dogajanja v atmosferi je tako dolg čas bistveno predolg. Za napovedovanje vremenskih dogajanj za nekaj ur vnaprej, za manjše in vnaprej določene lokacije, so se zato razvile posebne opazovalne, merske in prognostične metode, ki jih poimenujemo s skupnim imenom zelo kratkoročna in zdajšnja napoved vremena. Uporabniki teh so različni: zračni promet in letališča, cestni in pomorski promet, vojska, ukrepanja ob raznih naravnih katastrofah, industrijski kompleksi ... Na podlagi hitro pridobljenih podatkov, ki so osnovno sredstvo pri uporabi teh metod, je mogoče z metodami zdajšnje prognoze napovedati tudi bodoča stanja v atmosferi. Pri tem je pomembno, da so postopki zdajšnje prognoze tako hitri, da je čas, potreben za meritve in za izdelavo prognoze, tako kratek, da je prognoza pripravljena bistveno hitreje, kot pa potekajo procesi v naravi. Iz tega sledi, da morajo biti posamezni uporabljeni algoritmi v omenjenih metodah dovolj hitri in preprosti (največkrat zadoščajo že linearni algoritmi). Pogosto so postopki zdajšnje prognoze vezani na rezultate numerične prognoze. Primer: za zdajšnjo prognozo danes dopoldne uporabljamo prognostična polja za ta termin (danes dopoldne), ki so narejena na podlagi analize meritev od včeraj opoldne. Numerična prognoza priskrbi scenarij razvoja vremena, ki ga dopolnjujemo in verificiramo z metodami zelo kratkoročne prognoze.

2.3 Trenutne aktualne rešitve

2.3.1 ALADIN

Projekt ALADIN (*Aire Limitée Adaptation dynamique Développement InterNational*) je eden najpomembnejših mednarodnih projektov madžarske meteorološke službe na področju numerične napovedi vremena. Projekta je leta 1990 pričela francoska državna meteorološka služba *Météo-France* z namenom razvijanja numeričnega modela za napoved vremena za omejeno, manjše območje. Cilj projekta je izboljšanje meteoroloških, hidroloških in okoljskih opozoril in napovednih storitev, ki jih podpirajo člani projekta [29].

Slovenska meteorološka služba že od leta 1997 sodeluje v skupini ALADIN, kjer poleg francoske državne meteorološke službe aktivno sodelujejo še meteorološke službe Alžirije, Avstrije, Belgije, Bolgarije, Češke republike, Hrvaške, Madžarske, Maroka, Poljske, Portugalske, Romunije, Slovaške, Tunizije in Turčije. Znotraj skupine ALADIN je tudi pomembna in nekoliko drugače organizirana skupina RC LACE (*Regional Cooperation for Limited Area Modeling in Central Europe*; meteorološke službe Avstrije, Češke republike, Hrvaške, Madžarske, Romunije, Slovaške in Slovenije), ki si še dodatno prizadeva na podlagi skupnih prispevkov izboljšati in optimizirati operativne procese na področju numerične napovedi vremena [23].

Danes model ALADIN tvori preko 3 milijone vrstic programske kode (programska jezika FORTRAN [*FORmule TRANslation* - prevajanje enačb] ter C++), v Sloveniji pa se izvaja na gruči računalnikov oziroma računalniku visoke zmogljivosti *Nimbus* (od leta 2007) z operacijskim sistemom *Linux*, ki se nahaja v strežniških prostorih *Agencije Republike Slovenije za okolje*. Po zaslugi interneta in poljudne grafične predstavitve so z njegovimi rezultati redno seznanjeni vsi, ki jih vreme zanima, ali so od njega odvisni. Tako ima pomembno vlogo tudi pri promociji meteorologije v Sloveniji [12].

Osnova modela ALADIN so fizikalne enačbe, ki opisujejo obnašanje nestisljivega zraka v vlažnem ozračju na vrteči se Zemlji: Navier-Stokesova enačba, kontinuitetna enačba in energijska enačba. Iz njih se dobi sistem diferencialnih enačb, ki nima analitične rečitve. Lahko se ga rešuje le s pomočjo numeričnih metod. Enačbe so zato prilagojene takemu reševanju. V ta namen se ozračje nad območjem, kjer se z modelom simulira bodoči razvoj vremena, preslika v tridimezionalno mrežo računskih točk. V vsaki mrežni točki se nato računa hitrost gibanja zraka (veter), temperaturo, vlago, pritisk in druge količine, kot so vsebnost oblačnih in padavinskih delcev ... Za rešitev sistema diferencialnih enačb nad izbranim geografskim območjem je potrebno začetno stanje modelskih spremenljivk - začetne pogoje in njihove vrednosti na mejah računskega območja - robne pogoje. Rezultati globalnega modela ARPEGE[6] so se uporabili za začetne in robne pogoje ALADIN/LACE, ti rezultati pa so služili za začetne in robne pogoje ALADIN-SI [24].

Pred pričetkom izvajanja modela je potrebno izbrati zemljepisno območje, način priprave začetnih in robnih pogojev, primerno zahtevnost opisa procesov, vir začetnih in robnih pogojev ter urnik opravil. Pri tem je pomembna integracija teh sistemov v celoto in to, da sistem nadzorovano in neprestano teče, saj z naraščajočo prostorsko, horizontalno in vertikalno ter časovno ločljivostjo modela strmo rastejo tudi zahteve po diskovnem prostoru za obdelavo in hranjenje podatkov. Danes samo dnevna produkcija modela ALADIN zahteva okrog 120 GB prostora, ob dvakratnem povečanju horizontalne prostorske ločljivosti pa se ta številka poveča za štirikrat [12, 24].

Napovedi se pripravljajo ob 00, 06, 12 in 18 UTC[7], pri tem pa se teži k čim krajšim časovnim zamikom, ki jih določa razpoložljivost robnih pogojev modela gostitelja (ARPEGE ali IFS[8]). Takoj, ko je na voljo tri urna napoved

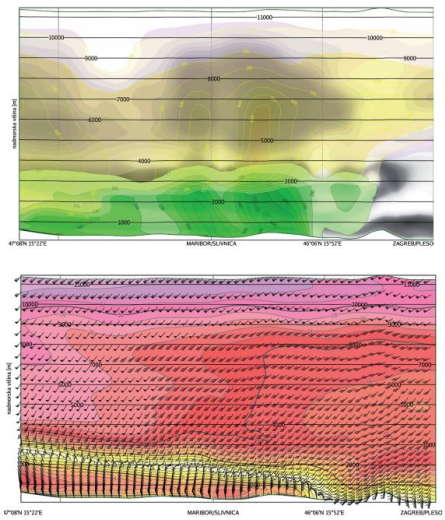
[6] ARPEGE (*Action de Recherche Petite Echelle Grande Echelle*) - raziskovalni/razvojni projekt za majhne in velike skale, ime globalnega modela francoske državne meteorološke službe, znan Landinov parfum; ime izhaja iz arpeggio, ki je način igranja akordov v glasbi.

[7] UTC - univerzalni koordinirani čas (*Universar Time Coordinated*).

[8] IFS (*Integrated Forecasting System*) - ECMWF-jev globalni meteorološki model.

globalnega modela, se priprav začetno stanje s pomočjo analize z lokalno asimilacijo, pri kateri se uporabijo opazovanja, ki so na voljo do tedaj. Analizi sledi izračun 54-urne napovedi, kjer je časovni korak integracije 3 minute. Med potekom integracije se še naprej sproti prenaša potrebne robne pogoje. Vsako uro se izpisujejo tridimenzionalna polja meteoroloških spremenljivk. V povprečju se računanje napovedi prične ob 3:00, 9:50, 14:40, 22:20 UTC. Računanje napovedi je ob uporabi 160 računskih jeder končano v 40 minutah.

Modelske izračune je nato potrebno predstaviti v primerni obliki. Zato se že med računanjem napovedi sproti pripravljajo grafični produkti, ki prikazujejo stanje meteoroloških spremenljivk ali izvedenih parametrov na izbranem območju. Pripravljajo se tudi izbrani časovne in vertikalni preseke (*slika 2.3*) v grafični obliki. Produkti so na voljo na internih spletnih pregledovalnikih, ožji izbor pa je prikazan tudi na spletnih straneh državne meteorološke službe (*www.meteo.si*).



Slika 2.3: Vertikalni krajevni presek za 7.10.2011 ob 10h v liniji Maribor - Zagreb. Na zgornji sliki so prikazane količine oblačne vode (vijolična), ledenih kristalčkov v oblakih (siva), dežja (zelena) in snega (rumena), na spodnji pa horizontalna hitrost (barvno polje) in smer vetra (smer zastavic). Vir slike: [24].

2.3.2 INCA

Skupna značilnost modelov za numerično napovedovanje vremena je, da njihove napake vrednosti zelo kratkoročnih napovednih (do 6 ur) niso bistveno manjše od napak kratkoročnih napovednih (od 12 do 24 ur). To je zato, ker se omenjeni modeli zaženejo z vgrajenimi področji analize, ki so močno omejena z dinamiko in fiziko modela in se lahko bistveno razlikujejo od opazovanih oziroma merjenih vrednosti vremenskih parametrov. Poleg tega njihova omejena horizontalna ločljivost (tipično reda 10 km) ne omogoča reprodukcijo vseh manjših vremenskih pojavov, ki so tipični za neko specifično lokacijo [9].
Za dopolnitev teh modelov oziroma izboljšanje napovedi je bil razvit sistem INCA (*the Integrated Nowcasting through Comprehensive Analysis*), ki temelji na analizi opazovanih oziroma merjenih in napovednih podatkov.

INCA je sistem za analizo in kratkoročno napovedovanje vremenskih parametrov, prikazanih v tabeli 2.1. Cilj sistema je izboljšanje zdajšnjih (do 4 ure) ter zelo kratkoročnih (do 12 ur) in dodajanje vrednosti oziroma popravkov kratkoročnih (do 72 ur) numeričnih napovedi že obstoječih modelov.

Citirano iz revije *Vetrnica* ([33]):
Sistem uporablja kot prvi približek stanja v atmosferi prostorska polja meteoroloških spremenljivk numeričnega meteorološkega modela (npr. ALADIN), nato pa s pomočjo interpolacijskih metod ob upoštevanju določenih fizikalnih zakonitosti izračunava 3-dimenzionalno fizikalno konsistentno analizo v visoki krajevni ločljivosti (1 km), v katero vključuje širok spekter različnih meritev (podatke s klasičnih in avtomatskih meteoroloških postaj, radarske in satelitske podatke in še druge razpoložljive podatke izven državne meteorološke mreže). Ta analiza je nato osnova za izračun nekaterih diagnostičnih polj kot tudi za kratkoročno napoved meteoroloških spremenljivk za 12 ur naprej. Pri

Tabela 2.1: Vremenski parametri modela INCA za analizo in napoved, njihovi vhodi ter interval osveževanja: NWP = izhod numeričnega modela, SFC = meritve s CVP, RAD = radarski podatki, SAT = satelitski podatki. Ostale kratice označujejo parametre znotraj INCA modela.

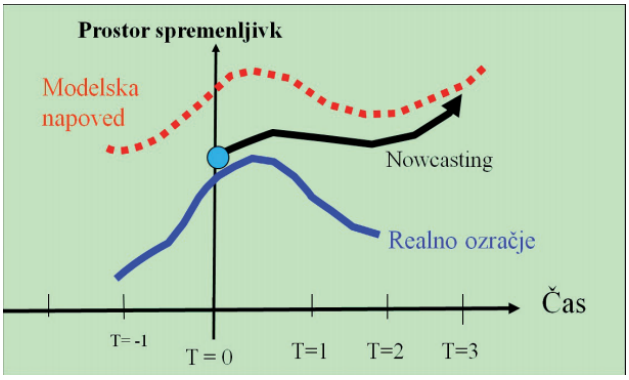
Parametri	**Vhod za analizo**	**Vhod za napoved**	**Osveževanje**
Temperatura	NWP,SFC,PP	NWP,SFC,PP,CC	1 ura
Vlažnost	NWP,SFC,PP	NWP,SFC,PP	1 ura
Veter	NWP,SFC	NWP,SFC	1 ura
Padavine	SFC,RAD	NWP,SFC,RAD	15 min
Tip padavin	SFC,ZS,TT,TG	NWP,ZS,TT,TG	15 min
Oblačnost	SFC,SAT	NWP,SFC,SAT	15 min
Globalno sevanje	SFC,CC	NWP,SFC,CC	1 ura
Sneženje	SFT,TT,QQ	NWP,SFC,TT,QQ	1 ura
Temp. tal	SFC,TT	NWP,SFC,TT	1 ura

tem nowcasting[9] *del izračuna, ki temelji pretežno na ekstrapolacijskih metodah, postopno prehaja v izračun numeričnega meteorološkega modela, kot je prikazano na sliki 2* (na sliki 2.4)*. Ključno pri tem je, da so izračuni dovolj hitri, da so lahko analize in napovedi dostopne praktično v realnem času ali zelo blizu realnega časa, kar omogoča tudi pogosto obnavljanje napovedi ob dostopnosti novih podatkov.*

2.4 Cestnovremenske postaje - vir podatkov

Cestnovremenske postaje (v nadaljevanju *CVP*) so skupek specialnih senzorjev za merjenje meteoroloških parametrov, strojne opreme za zbiranje podatkov ter pošiljanje le teh v nek strežnik ali odjemalec. Ker je njihov namen zbiranje podatkov o stanju vremena in cestišča, se skoraj vedno nahajajo tik ob cesti, kjer imajo poleg senzorjev tudi patentirane aktivne ter pasivne talne sonde za merjenje stanja cestišč. Aktivna sonda v enakomernih ciklih izpeljuje umetno podhladitev in segrevanje mešanice tekočine na tleh

[9] *Nowcasting* je angleška beseda, ki se sklicuje na *now* in *forecasting*, v meteorologiji na (zdajšnjo) napoved za običajno do 6 ur vnaprej.



Slika 2.4: Shematični prikaz *nowcasting* izračuna napovedi, ki postopno prehaja v modelsko napoved. Vir slike: [33].

z namenom simulacije procesa tvorjenja ledu vnaprej, medtem ko pasivna sonda to izpeljuje z uporabo matematičnih modelov. Sonde so vgrajene v cesto oziroma vozišče. Glede na predlog standarda [6], naj bi CVP izvajale naslednje meritve:

- stanje cestišča (navedenih je 5 osnovnih stanj),
- debelina vodnega filma,
- temperatura cestišča,
- temperatura zraka,
- temperatura rosišča,
- temperatura zmrzišča,
- relativna vlažnost zraka,
- čas zaznavanja padavin,
- tip padavin (tekoče, trdne),
- intenziteta padavin,
- debelina snežne odeje,

- hitrost vetra,
- sunki vetra,
- smer vetra,
- vidljivost.

Hitrost vetra se običajno meri 10 m nad tlemi, daleč od visokih ovir. Inštrument za merjenje hitrosti vetra se imenuje anemometer. Vrteči se del klasičnega anemometra je sestavljen iz treh ali štirih skodelic, pritrjenih na os. Hitrost vrtenja osi je sorazmerna s silo, s katero deluje veter na skodelice. Na samodejnih postajah so v uporabi tudi ultrazvočni anemometri, ki merijo hitrost ultrazvoka med tremi ali štirimi senzorji in posredno hitrost vetra.

CVP za zgorajo omenjene parametre uporabljajo kompleksne algoritme za izračun posameznih vrednosti, ki izhajajo iz izmerjenih podatkov. Občasno se namreč pojavijo takšne vremenske razmere, ki jih z osnovnimi meritvami ni moč zaznati oziroma dovolj dobro izmeriti. Zato so razviti posebni algoritmi, ki nam kot rezultat podajo določene alarme npr. za možnost nastanka poledice od določenih pogojih. S pomočjo CVP zaznavamo temperaturo, stanje ter slanost cestišča in na podlagi teh podatkov se izvajajo določeni posegi na cestišču, kot so soljenje, pluženje ... Na ta način je omogočeno učinkovitejše izvajanje teh aktivnosti zimskega vzdrževanja cest [15].

Citirano ([16]):
Nameščajo se na vnaprej določene lokacije, predvsem na izpostavljene odseke cest, viadukte in mostove. Izbira CVP se določi glede na potrebe pri zagotavljanju cestne varnosti na določenih odsekih ter objektih. Splošno znano dejstvo je, da se objekti kot so viadukti in mostovi hitreje ohlajajo in so zato veliko bolj izpostavljeni pojavu poledice od ostalih delov cestišč. Na podlagi tega dejstva so se izvajale postavitve in izbire lokacij CVP, pri tem pa je v dodatno pomoč mnenje lokalnih vzdrževalcev avtoceste. Le-ti na podlagi pogostih pojavov vremenskih nevšečnosti predlagajo postavitev CVP, pomembne

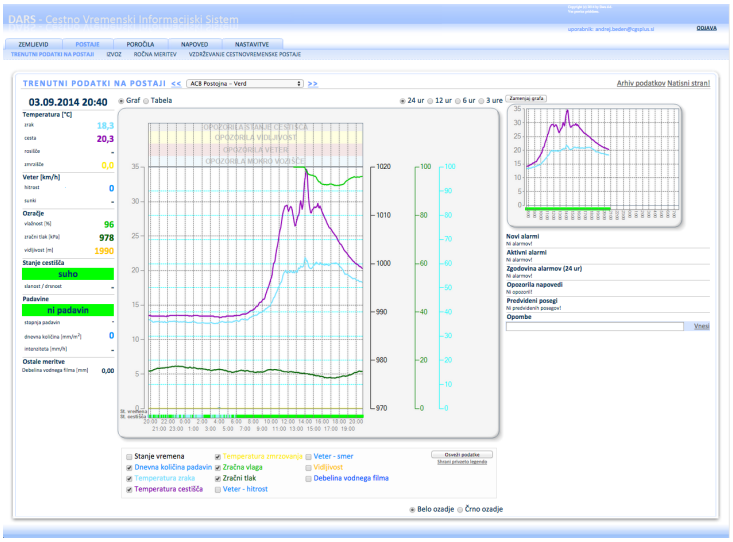
so predvsem mikrolokacije senzorjev (cestni senzorji ter senzorji hitrosti vetra na Primorskem).
Pomembno vlogo pri razvoju enotnega sistema CVP igrajo tudi posamezni senzorji, katerih postavitev in izbira je odvisna od gostote izpostavljenosti podatka na določenem območju ter od mikrolokacije. Zelo pomembno vlogo pri vsem tem je, da so vsi dodatno implementirani senzorji ter postaje kompatibilni z že zgrajenim sistemom CVP.

Slika 2.5: Cestnovremenska postaja Trebnje.

Slovensko avtocestno omrežje vključuje okoli 32 vremenskih postaj s senzorji različnih proizvajalcev, med katerimi so najpogostejši proizvajalcev *Boschung*, *Lufft* in *Vaisala*. Zelo pomemben faktor pri pravilnem delovanju CVP in verodostojnosti podatkov je njihovo vzdrževanje. Potrebni so redni pregledi, običajno pred in po zimskem letnem času, po potrebi pa tudi vmesni vzdrževalni posegi oziroma servisi. Podatke na prednastavljen inter-

val zbirajo in preko optičnega ali mobilnega omrežja pošiljajo v operativni center oziroma informacijski sistem, kjer jih spremljajo operaterji ter po potrebi obdelajo in posredujejo svojim klientom. Za vsak cestno informacijski sistem je pomembno, da ne deluje le na lokalni ravni, temveč omogoča povezovanje večjega števila cestnovremenskih postaj v enovit sistem, ki običajno omogoča:

- pregled nad lokacijami CVP,
- pregled stanj CVP (napajalna napetost, čas zadnjega prenosa podatkov na strežnik ...),
- avtomatsko generiranje opozoril (poledica, povišana stopnja mokrote, visoki sunki vetra, izpad CVP ...),
- pregled nad trenutnimi vrednostmi parametrov,
- pregled nad zgodovinskimi vrednostmi parametrov v obliki grafov,
- izvoz zgodovinskih podatkov ...



Slika 2.6: Prikaz podatkov CVP Verd v cestnovremenskem informacijskem sistemu Družbe za avtoceste v Republiki Sloveniji.

Poglavje 3

Uporabljene metode in orodja

CRISP-DM je eden najpogosteje uporabljenih procesov za namen modeliranja v industriji. Zaradi njegovega preprostega, jasnega in varnega načina prehajanja med fazami smo ga uporabili za namene tega diplomskega dela. Programski jezik *Python* smo zaradi dobre podpore ustreznim programskim knjižnicam uporabili za pripravo podatkov. Te smo nato s statističnim programskim okoljem *R* uporabili za analizo in uporabo na linearni regresiji in nevronskih mrežah. Potek uporabe orodij je prikazan v shemi.

3.1 Python

Python je objektno orientiran visokonivojski programski jezik, kar pomeni, da omogoča visok nivo abstrakcije od računalniške aparaturne opreme in operacijskega sistema. Ustvaril ga je *Guido van Rossum* leta 1990. Jezik je dobil ime po priljubljeni televizijski nanizanki *Leteči cirkus Montyja Pythona* (*Monty Python's Flying Circus*). Razvit je bil kot odprtokodni projekt, ki ga je upravljala neprofitna organizacija Python Software Foundation [14].

Ima popolnoma dinamične podatkovne tipe in samodejno upravlja s pomnilnikom, zaradi česar je podoben programskim jezikom *Perl*, *Ruby*, *Scheme*, *Smallatk* in *Tcl*. Visok nivo abstrakcije pomeni, da je jezik relativno hitro

naučljiv in enostaven, zato je še posebej primeren tako za začetnike kot tudi za izkušene programerje. Opazimo ga v širokem spektru uporabe: v njem so pisani nekateri sistemski ukazi v operacijskem sistemu Linux, v njem je vedno več spletnih aplikacij (med njimi je najopaznejša *Youtube* podjetja *Google*), kot zanimivost pa velja omeniti, da so bili v njem izdelani vizualni učinki filma *Vojna zvezd* (*Star Wars*) [5].

Jezik temelji na filozofiji povzeti iz dokumenta *PEP 20 (The Zen of Python)*, ki vsebuje aforizme[1] kot so:

- lepo je boljše od grdega,
- eksplicitno je boljše od implicitnega,
- enostavno je boljše od kompleksnega,
- kompleksno je boljše od zapletenega,
- berljivost šteje.

Za potrebe tega diplomskega dela se uporablja Python 64-bitna različica 2.7.7, ki je bila zadnjič posodobljena maja 2014. Posebej gre omeniti uporabljeni programski knjižnici *Numpy 1.8* ter *Pandas 0.14.0*. Prva skrbi za podporo numeričnim operacijam, več-dimenzionalnim tabelam oziroma objektom, med drugim pa tudi povezovanju s podatkovnimi bazami. Druga, odprtokodna knjižnica Pandas, pa nudi podporo enostavni ter visoko zmogljivostni uporabi različnih podatkovnih struktur ter analizi podatkov.

3.2 R

R je polnopravni programski jezik oziroma okolje z radikalno drugačnim pristopom do obdelave velikih, kompleksnih množic podatkov ter statistično

[1] Aforizem je zgoščeno, duhovito izražena globoka misel, domislica, oziroma resnica.

obdelavo. Gre za odprtokodni GNU projekt[2], podprt na več operacijskih sistemih. Njegov razvoj je odvisen od svetovne razvojne skupnosti oziroma prispevkov tistih, kateri ga največ uporabljajo. Med temi je največ statistikov, podatkovnih rudarjev ter analitikov. Jezik vključuje razne statistične in grafične tehnike, vključno z linearnim in nelinearnim modeliranjem, klasičnimi statističnimi testi, časovnimi analizami, razvrščanji, povezovanji ... Je enostavno razširljiv preko funkcij in drugih razširitev, kot so programski paketi ter knjižnice, po katerih je znana aktivna R skupnost. Ker je R interpretiran jezik, uporabniki do njega običajno dostopajo s pomočjo ukazne vrstice. Če uporabnik vnese *2+2*, mu računalnik odgovori s *4*. Podpira postopkovno programiranje s funkcijami in objektno orientirano programiranje s splošnimi oziroma generičnimi funkcijami [30].

Diplomsko delo obsega uporabo R programske knjižnice *neuralnet* avtorjev *Stefan Fritisch*, *Frauke Guenther* ter *Marc Suling*. Uporablja se različica 1.32, zadnjič posodobljena julija 2014. Knjižnica je namenjena podpori učenju nevronskih mrež, katere so opisane v nadaljevanju dela, z različnimi algoritmi[3].

3.3 Umetne nevronske mreže

Umetne nevronske mreže, najpogosteje klicane samo *nevronske mreže*, delujejo po principu človeških možganov, kateri procesirajo popolnoma drugače kot običajen digitalni računalnik. Največji približek možganom bi bil visoko kompleksen, nelinearen in paralelni računalnik.

Možgani imajo sposobnost organiziranja svojih strukturnih sestavin imenovanih *nevroni* na način, da so sposobni izvajati določena procesiranja, kot

[2]GNU projekt je brezplačna programska oprema, projekt za množično sodelovanje, napovedan leta 1983 s strani Richarda Stallmana z MIT.

[3]Algoritem je navodilo, s katerim rešujemo nek problem, običajno zapisano kot seznam korakov, ki nas pripeljejo do rešitve problema.

so razpoznava vzorcev v vidu, dojemanje, nadzor nad motoriko telesa ..., velikokrat hitreje kot trenutno najhitrejši digitalni računalnik na svetu. Ko se človek rodi, imajo njegovi možgani sposobnost ustvarjanja svojih *povezav med nevroni* oziroma svojih *pravil*, katerim lahko rečemo *izkušnje*. Največ teh pravil se ustvari v prveh dveh letih človekovega življenja, vsekakor pa se ustvarjanje nadaljuje tudi po tem obdobju. *Razvijajoči se nevron* je sinonim za *plastičnost možganov*: plastičnost dovoljuje razvijajočemu se živčnemu sistemu prilagajanje na okolje. Ta omenjena lastnost je tako pri človeških možganih eden bistvenih dejavnikov, ki vplivajo na obnašanje nevronov kot informacijsko-procesne enote [11].

Podobne elemente vsebujejo tudi nevronske mreže, tem elementom pa pravimo *umetni nevroni*. Nevronska mreža je stroj, ustvarjen za modeliranje na način, ki je najbolj podoben načinu, po katerem človeški možgani opravijo točno določeno nalogo ali interes. Običajno je implementirana v obliki digitalnega programa na računalniku. Da bi dosegli dobre rezultate, nevronske mreže temeljijo na ogromnem medomrežnem povezovanju preprostih celic za računanje oziroma prej omenjenih nevronih ali procesnih enot. Tako lahko nevronsko mrežo definiramo kot *prilagodljiv stroj* [11]:

Nevronska mreža je masovno paralelni porazdeljen procesor, ki ga sestavljajo preproste procesne enote in ki ima naravno nagnjenost za shranjevanje izkustvenega znanja ter uporabo le tega. Na človeške možgane spominja v dveh pogledih:

1. *Znanje je pridobljeno s procesom učenja s pomočjo omrežja povezav iz svojih okolj.*

2. *Povezave med nevroni, znane kot sinaptične uteži, se uporabljajo za shranjevanje pridobljenega znanja.*

Umetni nevron je model, ki je zasnovan kot približek biološkemu nevronu.

Dendrite[4] pri biološkem nevronu nadomestijo vhodi (x_i), ki imajo vsak svojo utež (w_i). Če imamo v obravnavi več nevronov, potem se uteži povezav nevrona k zapišejo kot w_{ki}. Izhod nevrona k, ki predstavlja vsoto vhodov, pomnoženih z ustreznimi utežmi, imenujemo tudi *aktivacija* (u_k). Aktivacija nevrona je vhod v *aktivacijsko funkcijo* (*angl. activation function*), včasih imenovano tudi *funkcija stiskanja*, ki omeji amplitudo izhoda nevrona. Vrednosti izhoda tipično pripadajo intervalu [0,1] ali intervalu [-1,1]. Izhod nevrona k označimo z y_k. Model nevrona k vsebuje tudi zunanji parameter, *prag* (Θ_k; *angl. threshold*), ki predstavlja nivo signala, pri katerem se sproži nevron. Nevron opišemo z naslednjim parom enačb [34]:

$$u_k = \sum_{j=1}^{S} w_{kj} x_j \tag{3.1}$$

$$y_k = \varphi(u_k - \Theta_k), \tag{3.2}$$

pri čemer so x_1, x_2, ..., x_S vhodni signali, w_{k1}, w_{k2}, ..., w_{kS} sinaptične uteži nevrona, u_k je aktivacija, Θ_k je prag, φ je aktivacijska funkcija in y_k je izhodni signal nevrona. Če vpeljemo $w_{k0} = \Theta_k$ in $x_0 = -1$ ter zapišemo $v_k = u_k - \Theta_k$, lahko preoblikujemo zgornji enačbi kot [34]:

$$V_k = \sum_{j=0}^{S} w_{kj} x_j \tag{3.3}$$

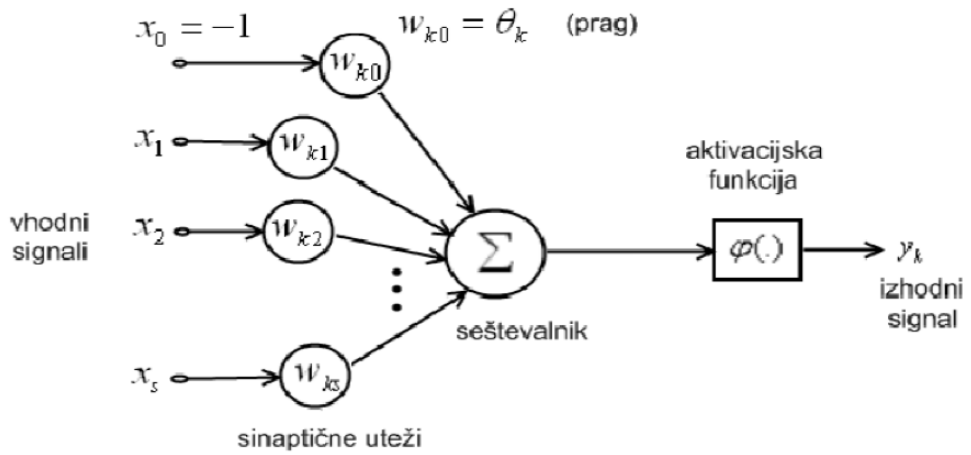
$$y_k = \varphi(v_k) \tag{3.4}$$

Model umetnega nevrona je prikazan na *sliki 3.1.*

Obstajajo naslednji tipi aktivacijske funkcije [34]:

1. **Pragovna (stopničasta) funkcija (*threshold function*)**

[4]Dendrite so razvejane projekcije nevronov, ki delujejo tako, da širijo elektrokemično stimulacijo prejeto od drugih živčnih celic v celico telesa.



Slika 3.1: Model umetnega nevrona.

$$\varphi(v) = \begin{cases} 1, & v \geqslant 1 \\ 0, & v < 0 \end{cases} \tag{3.5}$$

Tedaj je izhod iz nevrona (izpustimo indeks k):

$$y = \varphi(v) = \varphi(u - \theta) = \begin{cases} 1, & u \geqslant \theta \ oziroma \ v \geqslant 0 \\ 0, & u < \theta \ oziroma \ v < 0 \end{cases} \tag{3.6}$$

Nevronu s pragovno aktivacijsko funkcijo rečemo *McCulloch-Pittsov model*, mnogi avtorji pa tej obliki umetnega nevrona s to aktivacijsko funkcijo pravijo *pragovna logična enota* (*threshold logic unit – TLU*). Uporablja se za klasifikacijo linearno ločljivih vzorcev.

2. **Odsekovna linearna funkcija**

Slednjo opišemo z naslednjo enačbo:

$$\varphi(v) = \begin{cases} 1, & u \geqslant \frac{1}{2} \\ v + \frac{1}{2}, & -\frac{1}{2} < v < \frac{1}{2} \\ 0, & v \leqslant -\frac{1}{2} \end{cases} \tag{3.7}$$

3. **Sigmoidna funkcija**
   Slednja je daleč najbolj običajna oblika aktivacijske funkcije pri nevronskih mrežah. Omenimo tri tipe sigmoidne funkcije [34]:

   (a) **Logistična funkcija** (*Logistic function*):

$$\varphi(v) = \sigma(v) = \frac{1}{1 + e^{-av}} \tag{3.8}$$

   (b) **Funkcija signum**:

$$\varphi(v) = sng(v) = \begin{cases} 1, & v > 0 \\ 0, & v = 0 \\ -1, & v < 0 \end{cases} \tag{3.9}$$

   (c) **Funkcija hiperbolični tangens**:

$$\varphi(v) = tanh\frac{v}{2} = \frac{1 - e^{-v}}{1 + e^{-v}} \tag{3.10}$$

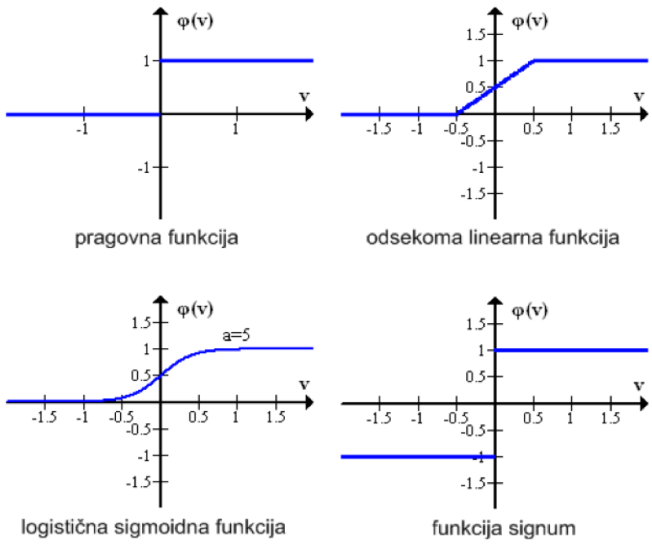
Oblike omenjenih funkcij so prikazane na *sliki 3.2*.

Ena najpomembnejših lastnosti nevronske mreže je zmožnost učenja in izboljševanja svoje učinkovitosti skozi proces učenja. V grobem bi lahko rekli, da ta proces nastavi nevronsko mrežo tako, da na določeno množico vhodov reagira z določeno množico izhodov. Učenje poteka kot iterativni proces prilagajanja, ki se izvaja na utežeh in pragovih glede na določeno učno pravilo. V povezavi z nevronskimi mrežami definiramo učenje na naslednji način [11]:

Učenje je proces, pri katerem se prosti parametri nevronske mreže prilagodijo skozi nenehen proces vzpodbude iz okolja, v katerega je mreža vložena. Tip učenja določa način, po katerem se spreminjajo parametri.

Ta definicija vključuje naslednje zaporedje dogodkov:

1. Nevronsko mrežo vzpodbuja okolica.



Slika 3.2: Oblike aktivacijskih funkcij.

2. Nevronska mreža se spremeni zaradi te vzpodbude.

3. Nevronska mreža odgovori na nek način okolju zaradi sprememb, ki se zgodijo v njeni notranji zgradbi.

Naj bo $w_{kj}(n)$ vrednost sinaptične uteži w_{kj} v času n. Prilagodimo sinaptično utež $w_{kj}(n)$ z $\Delta w_{kj}(n)$ tako, da dobimo osveženo vrednost $w_{kj}(n+1)$:

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n) \tag{3.11}$$

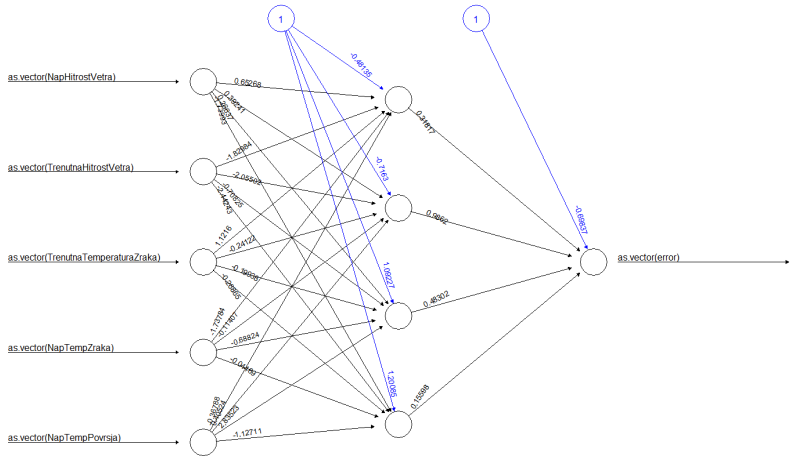
Prilagoditev $\Delta w_{kj}(n)$ se izračuna kot rezultat vzpodbude okolice (*dogodek 1*). Osvežena vrednost $w_{kj}(n+1)$ predstavlja spremembo v mreži kot rezultat te vzpodbude (*dogodek 2*). *Dogodek 3* se izvrši, ko se izračuna odgovor nove mreže, ki deluje z osveženo vrednostjo parametra $w_{kj}(n+1)$.
Opisano množico pravil imenujemo *algoritem učenja* (*learning algorithm*). Obstaja več algoritmov za učenje nevronskih mrež, ki se v osnovi razlikujejo po načinu prilagoditve sinaptične uteži w_{kj}. Najbolj tradicionalen je *algoritem vzratno širjenje napake* (*engl. backpropagation algorithm*. Ta mrežo uči tako, da približuje napako vrednosti nič, napaka pa je razlika med odvisno

spremenljivko in izhodom. V osnovi uporablja metodo *padajočih gradientov* oziroma *delta pravilo*, katerega namen je posodabljanje uteži v *fazi backpropagation* ter s tem zmanjševanje napake E. Prvi korak je *inicializacija*, kjer se vse uteži, vključno s pragovi w_0, nastavijo z naključno funkcijo na območje med -0.5 in 0.5. Drugi korak, *zanka*, pa skrbi za ponavljanje postopka, dokler se vsi vzorci ne klasificirajo pravilno. Mreža se uči in se ji predstavlja učne vzorce dokler ni izpolnjen pogoj za konec.
Za namene diplomskega dela smo uporabljali izboljšano različino omenjenega algoritma imenovano *resilient backpropagation* (*Martin Riedmiller* in *Heinrich Braun*, 1992). Slednji algoritem je eden med najhitrejših pri posodabljanju uteži. Osnovni princip teži k spreminjanju vrednosti vsake uteži glede na obnašanje zaporedja delnih derivatov. Podrobnejši opis najdemo v dokumentaciji avtorja, primerjavo z drugimi algoritmi pa v člankih [13, 1, 26, 3].

Citirano([4]): *Glede na vrsto topologije ločimo večnivojske in enonivojske nevronske mreže. Prve so organizirane po nivojih (vhodni, skriti, izhodni) in ne dovoljujejo povezav med nevroni na istem nivoju. Druge ne ločujejo nevronov po nivojih in dovoljujejo povezave med vsemi nevroni.*
Glede na usmerjenost povezav ločimo nevronske mreže s povezavami nazaj in nevronske mreže brez povezav nazaj. Prve dovoljujejo povezave v obe smeri, pri povezavah drugih pa izhodi enega nevrona pomenijo vhode drugega nevrona, torej se vrednosti vedno prenašajo le z leve proti desni.
Glede na način učenja ločimo nadzorovano učenje in nenadzorovano učenje. Nadzorovano učenje ima nekakšnega učitelja, ki mreži poda pravilen odgovor, medtem ko pri nenadzorovanem učenju ni primerjav med dejanskim in pričakovanim izidom.

Za potrebe diplomskega dela smo uporabili *večnivojske nevronske mreže*, *brez povezav nazaj* z *nadzorovanim učenjem* (v učni množici imamo podan parameter *napaka*). Predstavnikom takih nevronskim mrež pravimo *Feedforward nevronske mreže*, poznamo pa še *Hopfieldove nevronske mreže* (eno-

nivojske, s povezavami nazaj, nadzorovano učenje) in *Kohonenove nevronske mreže* (večnivojske, brez povezav nazaj, nenadzorovano učenje).

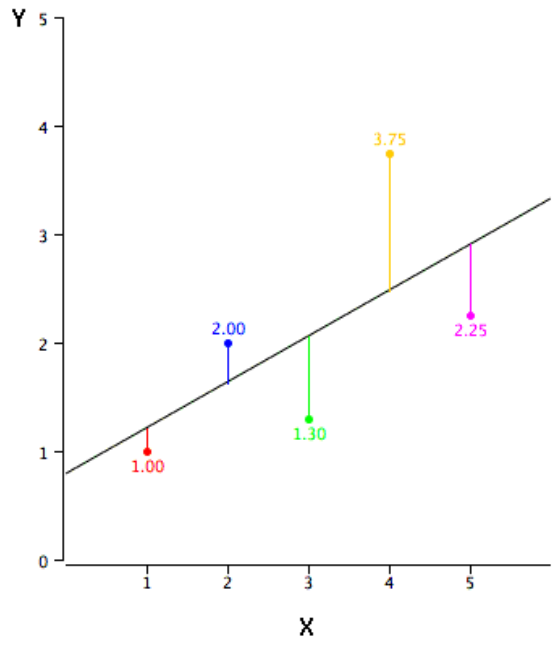


Slika 3.3: Izrisana shema nevronske mreže v programskem okolju *R*. Mreža ima 1 skriti nivo (večnivojska) ter v njem 4 skrite nevrone s povezavami samo naprej. Vhodnih parametrov je 5, izhodni 1 (*napaka*).

3.4 Linearna regresija

V *enostavni linearni regresiji* napovedujemo rezultate prve spremenljivke iz rezultatov druge. Spremenljivka, za katero napovedujemo se imenuje *kriterij* (odvisna spremenljivka) in se omenja kot Y. Spremenljivka na podlagi katere napovedujemo se imenuje *prediktor* (neodvisna spremenljivka), katera se omenja kot X. Ko imamo samo en prediktor, se napovedna metoda imenuje *enostavna linearna regresija*.

Linearna regresija teži k iskanju najustreznejše premice, ki poteka čim bližje čim večim vrednostim oziroma točkam na grafu. Najbolje prilegajoča se premica se imenuje *regresijska premica*. Kriterij za to je običajno tak, da mora *vsota kvadratov napake med dejansko in napovedano vrednostjo* biti minimalna.



Slika 3.4: Primer linearne regresije. Črna črta se imenuje *regresijska premica*, ki predstavlja napovedane vrednosti, barvne točke pa predstavljajo prave vrednosti.

Na *grafu 3.4* črna premica predstavlja regresijsko premico oziroma napovedane vrednosti, barvne točke pa prave vrednosti.
Enačba za regresijsko premico je tako

$$Y' = a + kX, \tag{3.12}$$

kjer je Y' napovedovana vrednost, k naklon regresijske premice in a presečišče le te z ordinatno osjo, ki nam pove kolikšna bi bila napovedana vrednost kriterija, če bi bili vsi prediktorji enaki nič.

Z *multiplo regresijo* napovedujemo vrednost *kriterija* (odvisne spremenljivke) na osnovi dveh ali več *prediktorjev* (neodvisnih spremenljivk). Iščemo linearno kombinacijo prediktorjev, katere vrednosti bodo čim bliže vrednostim kriterija. Uteži, s katerimi obtežimo prediktorje, imenujemo *regresijski*

nagibi. Tej nam povedo, za koliko se poveča napovedana vrednost kriterija, če se vrednost prediktorja poveča za eno enoto, vrednosti ostalih prediktorjev pa se ne spremenijo. Slednji model lahko zapišemo z enačbo

$$Y_i' = a + \sum_{j=1}^{P} k_j X_{ij}, \tag{3.13}$$

kjer je Y_i' napovedana vrednost, k_j regresijska utež oziroma nagib za prediktor j in a presečišče z ordinatno osjo.

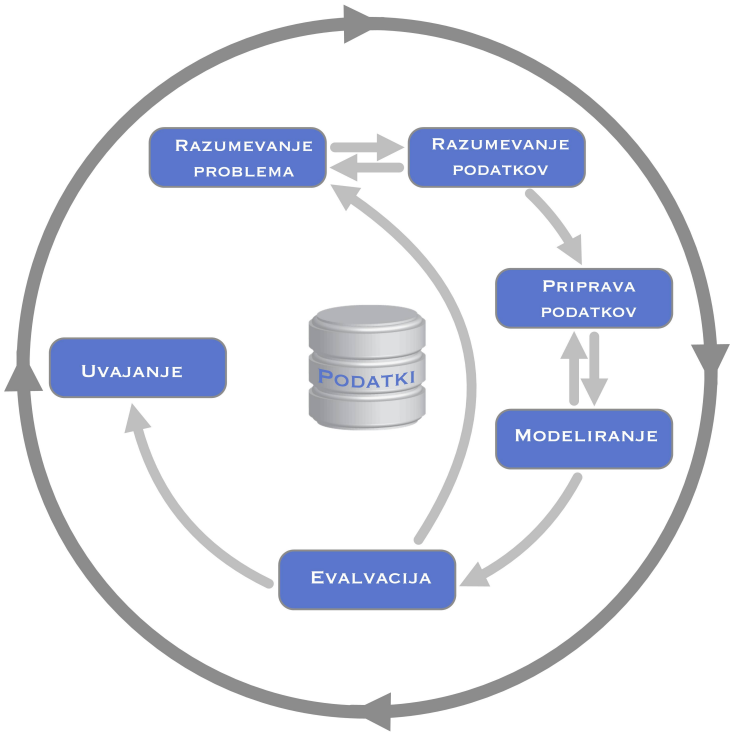
V tem delu smo rezultate napovedi *multiple regresije* primerjali z rezultati umetnih nevronskim mrež.

3.5 CRISP-DM

CRISP-DM (*Cross Industry Standard Process for Data Mining*) je procesni model, ki opisuje najpogosteje uporabljene pristope, ki jih uporabljajo strokovnjaki pri reševanju problemov povezanih s podatkovnim rudarjenjem in zagotavlja strukturiran pristop do načrtovanja projekta. Metodologija je robustna in že dobro uveljavljena.

Življenski cikel je sestavljen iz 6 faz, ki so prikazane v *sliki 3.5.* Zaporedje faz ni togo oziroma premikanje naprej in nazaj med fazami je priporočljivo. Puščice na omenjeni sliki nakazujejo na najpomembnejše odvisnosti med fazami. Ni nujno, da se podatkovno rudarjenje konča v zadnji fazi, v uvajanju. Spoznanja po tem procesu lahko sprožijo nov pogled na problem, novo razumevanje problema, bolj specifična poslovna vprašanja ...

Prva faza, faza *razumevanja problema* se osredotoča na razumevanje ciljev ter zahtev problema iz poslovne perspektive, obenem pa poskuša ta spoznanja pretvoriti v problem, na katerega lahko gledamo iz perspektive podatkovnega rudarjenja.
Naslednje faza, faza *razumevanja podatkov* se prične z zbiranjem prvih podat-



Slika 3.5: Faze procesa CRISP-DM.

kov ter aktivnostmi, ki pomagajo pri seznanitvi s temi podatki, ocenjevanju kakovosti podatkov, odkrivanju prvih vpogledov v podatke, odkrivanju zanimivih podmnožic ... s čimer se tvorijo hipoteze o skritih informacijah.
Faza *pripravljanja podatkov* pokriva vse aktivnosti potrebne za izdelavo končne množice podatkov (podatki, ki se bodo uporabljali pri modeliranju) iz prvotne, surove množice. Te naloge se po nobenem predpisanem zaporedju lahko izvedejo večkrat. Vsebujejo izbor tabel, atributov ter vnosov kot tudi morebitne transformacije in čiščenje podatkov.
Modeliranje vključuje izbor in uporabo raznih tehnik modeliranja ter njihovo optimizacijo, ki je odvisna od problemske domene ter spoznanj prejšnjih faz. Običajno lahko za vsak problem najdemo več tehnik modeliranja, izmed katerih ima lahko neka tehnika potrebo po specifični obliki podatkov. Če se vidi potreba po izboru take tehnike, je potrebna ponovna ustrezna priprava podatkov oziroma prejšnja faza.

V fazi *evalvacije* je že na voljo eden ali več modelov, ki so s perspektive analize podatkov najbolj ustrezni za nek problem. Pred nadaljevanjem v naslednjo fazo je potrebno oceniti in pregledati, ali model pravilno doseže svoje rezultate, ter če rezultati modela zadostujejo potrebam problema iz poslovne perspektivne.

Izbor oziroma oblikovanje modela generalno še ni zadnja faza projekta. Če je cilj modela odkrivanje novih znanj iz podatkov, je potrebno ta znanja organizirati ter predstaviti stranki projekta na način, da bo stranka znala ta znanja uporabiti. Omenjene naloge so glavni cilj zadnje faze, faze *uvajanja* [18].

Poglavje 4

Testna metodologija

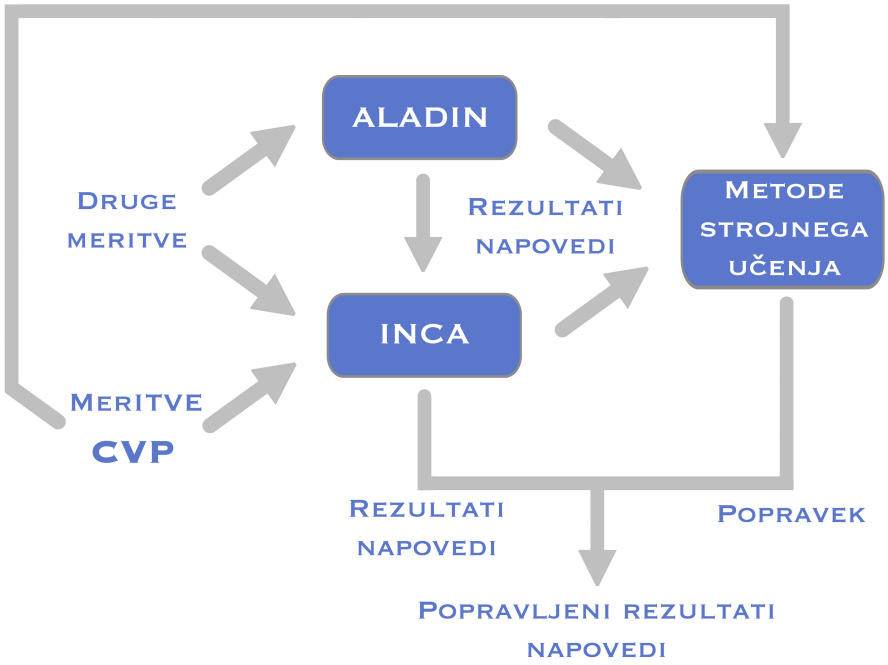
4.1 Razumevanje problema

V prej opisanih napovednih modelih ALADIN ter INCA je vključenih ogromno vremenskih parametrov ter fizikalnih modelov, ki upoštevajo okolico, tipografijo ... V obsegu diplomskega dela nismo poskušali razviti novega abstraktnega modela za napoved parametrov, vendar smo se osredotočili na naknadno izboljšanje lokalnih napovedi hitrosti vetra. Posledično nas je zanimala razlika med napovedano vrednostjo parametrov modelov za nek čas in izmerjeno vrednostjo parametrov v času, za katerega se napoveduje. Slednje smo definirali kot *napaka*.

Ker so merjeni podatki pridobljeni s pomočjo prej opisanih cestnovremenskih postaj, katere se nahajajo ob večjih slovenskih cestah, lahko rezultati dela koristijo gospodarstvu, še posebej cestnim vzdrževalnim službam. Napoved vetrovnih parametrov smo izboljševali s pomočjo pravtako prej opisanih umetnih nevronskih mrež, ki so že preizkušene na podobnih modelih ter se za te namene zadovoljivo obnesejo.

Napovedne vrednosti nam je zagotovila *Agencija Republike Slovenija za okolje*. Večino teh smo pridobili s produkta INCA, nekateri (napoved zračnega

tlaka ter sevalnih parametrov) pa so na voljo samo kot rezultat produkta ALADIN, zato smo uporabili tudi napovedne vrednosti slednjega. Merjene vrednosti vremenskih parametrov smo pridobili s prej opisanih cestnovremenskih postaj, ki so last *Družbe za avtoceste Republike Slovenije*.



Slika 4.1: Shema rešitve.

4.2 Razumevanje podatkov

Podatki izmerjenih vremenskih parametrov, ki smo jih v tem delu uporabljali, so pridobljeni s prej opisanih cestno vremenskih postaj (CVP) in se nahajajo na strežniku v navideznem zasebnem omrežju[1] *Družbe za avtoceste Republike*

[1]Navidezno zasebno omrežje (angl. *virtual private network - VPN*) je podomrežje interneta, do katerega imajo dostop samo določeni uporabniki. Svojo identiteto običajno dokažejo z digitalnim certifikatom.

Slovenije. Nahajajo se v podatkovni bazi na strežniku z nameščenim Microsoftovim operacijskim sistemom *Windows Server 2012 R2*. V isti podatkovni bazi so shranjeni tudi napovedni podatki modela INCA.

Podatki se z vsake CVP na strežnik prenašajo v intervalih približno 5 minut. Za časovno zaokroževanje na polne čase pri tem ni poskrbljeno, zato se prenosi podatkov v teh intervalih lahko zgodijo ob vsakem času, naprimer ob 12:02, 12:06 ...

Tabela 4.1: Tabela časovnih intervalov merjenih podatkov za CVP.

čas	**kraj CVP**
1.1.2014 00:01	Hrušica
1.1.2014 00:06	Hrušica
1.1.2014 00:12	Hrušica
⋮	⋮
1.1.2014 00:00	Novo mesto
1.1.2014 00:04	Novo mesto
⋮	⋮

Za prenos INCA napovednih podatkov v podatkovno bazo na strežniku je poskrbljeno s časovnim zaokroževanjem na polno uro, zato so nam ob vsaki polni uri na voljo napovedni podatki za vsako od 11 ur vnaprej za vsako CVP. Primeri intervalov napovednih oziroma merjenih podatkov so prikazani v *tabeli 4.1* oziroma *tabeli 4.2*.

Podatki merjenih vrednosti in podatki napovedanih vrednosti se na strežniku v podatkovni bazi nahajajo v ločenih tabelah, z različnimi atributi. V Microsoftovem orodju *SQL Server Management Studio 2012* je bilo potrebno

Tabela 4.2: Tabela časovnih intervalov INCA napovednih podatkov za CVP.

čas	**število ur napovedi za vnaprej**	**kraj CVP**
1.1.2014 00:00	1	Hrušica
1.1.2014 00:00	2	Hrušica
⋮	⋮	⋮
1.1.2014 00:00	11	Hrušica
1.1.2014 01:00	1	Hrušica
⋮	⋮	⋮
1.1.2014 01:00	11	Hrušica
⋮	⋮	⋮
1.1.2014 00:00	1	Novo mesto
⋮	⋮	⋮

željene podatke z naslednjimi SQL[2] sestavljenimi stavki pretvoriti v CSV[3] obliko ter jih prenesti na lokalen računalnik:

```
# @IDpostaje je spremenljivka, ki doloca ID CVP
SELECT *
FROM [DARS_CVIS].[dbo].[Mes_Value]
WHERE fk_Station_Mes_ID IN (
        SELECT ID
        FROM [DARS_CVIS].[dbo].[Station_mes]
        WHERE fk_Measurement_ID = 15
        AND fk_Station_ID = @IDpostaje
)
ORDER BY Mes_DateTime ASC
```

```
# @IDpostaje je spremenljivka, ki doloca ID CVP
SELECT *
```

[2]SQL ali strukturirani povpraševalni jezik za delo s podatkovnimi bazami (angl. *Structured Query Language*) je najbolj razširjen in standardiziran povpraševalni jezik za delo s podatkovnimi zbirkami, s programskimi stavki, ki posnemajo ukaze v naravnem jeziku.

[3]CSV (angl. *Comma Separated Values*) je oblika zapisa podatkov. Največkrat so zapisani v obliki stolpcev, ločenimi z vejicami.

```
FROM [DARS_CVIS].[dbo].[WeatherForecast]
WHERE fk_Station_ID = @IDpostaje
ORDER BY DateTime ASC
```

4.3 Priprava podatkov

V pričetku te faze smo že imeli na voljo podatke v dveh ločenih CSV datotekah za vsako CVP in sicer napovedne ter merjene. Cilj postopka *priprave podatkov* je bil za vsako CVP pripraviti podatke v obliko, bi bo primerna za modeliranje. V naši problemski domeni so v ta namen prišle vpoštev ločene datoteke za vsako napovedno uro posebej (od 1. do 11. ure), torej 11 datotek v obliki CSV z merjenimi in napovednimi podatki urejenimi po času meritev.

Ker merjeni pododatki niso bili vedno izmerjeni ob polnih urah, smo za vsako uro izbrali podatek, ki je bil izmerjen približno ob polni uri oziroma katerega toleranca[4] okoli polne ure obsega 3 minute. Isti postopek smo uporabili pri izbiri podatka meritve v času $t + \Delta t$.

Med postopkom naslednje faze, *faze modeliranja*, smo se na *fazo priprave podatkov* iterativno vračali. Po zaključku slednje faze smo s pomočjo prej opisanega programskega jezika *Python* ter podporo programskimi knjižnicami *Pandas* ter *Numpy* datoteke za vsako CVP ustvarili po postopku opisanim s psevdokodo[5] v *algoritmu 1*. Vrstice v omenjenih ustvarjenih datotekah so po postopku zaokroževanja oziroma po koncu *faze priprave podatkov* sestavljene iz podatkov:

[4]Toleranca je dovoljena razlika med nominalno in dejansko vrednostjo neke vrednosti, količine ali kakovosti.

[5]Psévdokóda je način, kako predstaviti algoritem, ki sicer upošteva pomenoslovje, ni pa nujno skladenjsko pravilno in je namenjen izključno tolmačenju. Ni omejena na določen programski jezik.

Algoritem 1: Priprava podatkov za modeliranje

```
Vhod: datoteki z meritvenimi in napovednimi podatki;
      število ur za napoved vnaprej
Izhod: 11 datotek sestavljenih iz specifičnih podatkov
1  datotekaMeritve ← datoteka z meritvenimi podatki;
2  datotekaNapovedi ← datoteka z napovednimi podatki;
3  for steviloUrVnaprej ← 1 to območje(1:11) do
4      izhodnaDatoteka ← ∅;
5      for i ← 1 to območje(1:datotekaMeritve.steviloVrstic) do
6          vrsticaMeritve ← datotekaMerive[i];
7          časMeritve ← vrstica.časMeritve;
8          if okoliPolneUre(časMeritve) then
9              časMeritve ← zaokroziNaPolnoUro(časMeritve);
10             vrsticaMeritveVnaprej ← Pan-
               das(datotekaMeritve[zaokroziNaPolnoUro(časMeritve)]
               == časMeritve + številoUrVnaprej);
11             vrsticaNapovedi ← Pandas(datotekaNapovedi[časNapovedi]
               == časMeritve and datotekaNapovedi[številoUrVnaprej]
               == številoUrVnaprej);
12             if vrsticaMeritveVnaprej and vrsticaNapovedi then
13                 izhodnaVrstica ← vrsticaMeritve +
                   vrsticaMeritveVnaprej + vrsticaNapovedi;
14                 izhodnaDatoteka.pripniNaKonec(izhodnaVrstica);
15             end
16         end
17     end
18     urediPoČasu(izhodnaDatoteka);
19     shraniNaMedij(izhodnaDatoteka);
20 end
```

- $t + \Delta t$ — časa za katerega model INCA napoveduje vrednosti vremenskih parametrov,
- t — časa, ko se je izvedla napoved modela INCA za čas $t + \Delta t$ oziroma časa meritve določenih vremenskih parametrov,
- Δt — število ur od časa, ko se je izvedla napoved modela INCA (t) do časa, za katerega se napoveduje ($t + \Delta t$),
- napovedanih vrednosti vremenskih parametrov modela INCA za čas $t + \Delta t$,
- izmerjenih vrednosti vremenskih parametrov v času t,
- izmerjenih vrednosti vremenskih parametrov v času $t + \Delta t$.

4.4 Modeliranje in evalvacija

Glavni cilj *faze modeliranja* nam je bil modeliranje oziroma napovedovanje *popravka* oziroma *korekcije* napovedi hitrosti vetra, katero smo nato odšteli od napovedane vrednosti hitrosti vetra, rezultate pa primerjali. Celotno fazo smo opravljali v okolju *R*.
Najprej smo z ustreznimi programskimi funkcijami v 11 spremenljivk prebrali 11 *CSV* datotek s podatki za neko CVP, ki smo jih v prejšnji fazi ustvarili s programskim jezikom *Python*.

Da bi izračunali natančnost napovedi hitrosti vetra modela INCA za vsako napovedno uro posebej, smo *napako* definirali kot razliko med napovedano vrednostjo v času t za nek čas $t + \Delta t$ in dejansko izmerjeno vrednostjo v času $t + \Delta t$, kar lahko zapišemo z *enačbo 4.1*.

$$napaka = INCA\ napoved(t;\ za\ t + \Delta t) - meritev(t + \Delta t) \tag{4.1}$$

Vsaki vrstici (sestava vrstice je opisana v prejšnjem poglavju) v podatkih smo dodali parameter *napaka* izračunanega z *enačbo 4.1*.

Ugotovili smo, da so izmerjeni podatki za pritisk v podatkovni bazi shranjeni v enoti *hektopascal (hPa)*, napovedni pa v *pascal (Pa)*, zato smo slednje zaradi boljšega pregleda nad podatki delili s 100 (1 hPa $\equiv$ 100 Pa).

4.4.1 Učenje

Za učenje nevronskih mrež kot tudi za pridobitev regresijskih prediktorjev (od sedaj bomo temu pravili tudi učenje) je bilo potrebno pripraviti *učno množico* s čim več koristnimi podatki. Za slednjo smo uporabili podatke za obdobje od leta 2009 do 2013. Množica je vsebovala vse parametre, ki so nam bili na voljo, med katerimi je tudi prej izračunan parameter *napaka*.

Učenje je potekalo ločeno na podatkih za vsako napovedno uro (od 1 do 11) in za vsako CVP posebej. Z uporabo modeliranja smo napovedovali parameter *napaka* oziroma *korekcija*, zato mu lahko rečemo *kriterij*. Kot *prediktorje* smo uporabili parametre na desni strani znaka $\sim$ oziroma *kače* v naslednjem zapisu:

korekcija $\sim$ *napovedana hitrost vetra + trenutna hitrost vetra + napovedana temperatura zraka + trenutna temperatura zraka + napovedana temperatura površja*

Kot kompromis smo slednje parametre izbrali na podlagi razlag o vplivu na veter v knjigi *Osnove meteorologije za naravoslovce in tehnike* ([25]).

V članku ([20]) avtorji navajajo, da je uspešnost in hitrost konvergence nevronskih mrež močno povečana, če so vhodi v te normalizirani, zato smo vse vrednosti podatkov normalizirali z *enačbo 4.2*, kjer je *x'* nova normali-

zirana (v območju [0,1]), x stara vrednost, *max(X)* ter *min(X)* pa največja oziroma najmanjša vrednost iz množice nenormaliziranih vrednosti.

$$x' = \frac{x - min(X)}{max(X) - min(X)} \tag{4.2}$$

Arhitekturo mrež smo nastavili po priporočilih člankov in testiranjih na podatkih za več CVP:

- Število skritih nivojev (*Hidden layers*) smo nastavili na 1. Slednje zadostuje za veliko večino problemov, saj lahko že z 1 skritim nivojem poljubno natančno aproksimiramo skoraj katerokoli funkcijo. Z nevronskimi mrežami smo poskusili tudi simulirati linearno regresijo, kar smo dosegli z uporabo nevronske mreže z 0 skritimi nivoji. Rezultati so vidni v *grafu 5.7* [32].

- Večino študij za število skritih nevronov priporoča povprečje vhodnih in izhodnih parametrov ali približno $\frac{2}{3}$ vhodnih nevronov, čemur prištejemo število izhodnih nevronov ali da naj bo število skritih nevronov manjše od dvakratnika števila vhodnih nevronov. Teh imamo v našem primeru 5, izhodnih pa enega (*korekcija*). Opravili smo testiranja z različno števili skritih nevronov, tudi več kot 20, vendar so razlike bile pričakovane oziroma izredno majhne. Uporabili smo 4 skrite nevrone [20, 32].

- V programskem paketu smo lahko izbirali med aktivacijskima funkcijama *hiperbolični tangens* in *logistična* funkcija. Ker ima prva izhodne vrednosti v območju [-1,1] in druga [0,1], smo izbrali slednjo zaradi normalizacije podatkov na to območje [21].

- Na izhodnem nivoju smo lahko uporabili možnost *linearnega izhoda*, kar pomeni, da so izhodne vrednosti lahko v območju $(-\infty, +\infty)$. Ker smo vrednosti vhodnih parametrov normalizirali, možnosti *linearnega izhoda* zato nismo uporabili.

- Vrednosti začetnih uteži smo uporabili privzete. Te se nastavijo na naključne vrednosti, kar je po priporočilih.

- Uporabimo lahko dve različni funkciji za seštevek vsote uteži: *cross-entropy* in *sum of sqared error* (vsota korenov napak). Prva se uporablja za primere, kjer je izhod aktivacijskih funkcij binarna vrednost ali verjetnost, zato smo uporabili drugo. V primeru slednjega bi bila boljša izbira prva [8, 10].

- *Prag* (*angl. threshold*) smo ugotovili s preizkušanjem na validacijski množici. Prenizek ali previsok prag lahko povzroči *overfitting* oziroma *underfitting*. To pomeni, da se nevronska mreža preveč oziroma premalo prilagodi učnim podatkom, kar v večini primerov vpliva na poslabšanje rezultatov uporabe iste nevronske mreže na testnih podatkih. Privzeta pragovna vrednost je bila 0.01, po testiranjih pa smo se odločili za vrednost 0.5. Višja ali nižja vrednost v povprečju doprinese slabše rezultate.

- Na voljo smo imeli naslednje algoritme učenja (v angleških izrazih): *resilient backpropagation with weight backtracking*, *resilient backpropagation without weight backtracking*, in dve različici *globally convergent* algoritma (*sag* in *slr*), ki sta spremenjeni različici drugega. Med algoritmi kljub spreminjanju ostalih parametrov ni bilo signifikantnih razlik.

- Ostale parametre funkcije za uporabo nevronskih mrež v paketu *neuralnet* smo ohranili na privzetih vrednostih, katere so poleg opisov na vpogled na voljo v uradni dokumentaciji [7].

Umetne nevronske mreže so nedeterministične (zaradi inicializacije začetnih uteži na naključne vrednosti), zato so lahko rezultati na istih podatkih različni. Proces učenja smo tako na istih podatkih ponovili z različno naključno inicializacijo uteži, rezultate vsake ponovitve pa shranjevali.

Če bi za vsako od 5-ih ponovitev procesa učenja izbrali drugo množico (ali drugo podmnožico iste množice) podatkov, bi v našem primeru lahko na

končne rezultate vplival tudi parameter *rep* oziroma *število iteracij učenja nevronske mreže*. Slednji bi nevronsko mrežo večkrat naučil na isti trenutno izbrani množici podatkov, rezultate pa povprečil. Zaradi procesa učenja vsake ponovitve na isti množici podatkov omenjeni parameter nima vpliva na rezultate, zato smo ga ohranili na privzeti vrednosti (to je 1).

4.4.2 Testiranje

Testiranje opisanih metod strojnega učenja je potekalo na podatkih iz leta 2014 za vsako CVP in napovedno uro posebej. Rezultate testiranj naučenih metod na slednjih podatkih smo pridobili z ustreznimi funkcijami v *R* okolju (*compute* in *predict*).

Podatke in rezultate metod je bilo potrebno denormalizirati, za kar smo uporabili *enačbo 4.3*, kjer je x normalizirana, x' denormalizirana (v območju [0,1]), *max(X)* ter *min(X)* pa največja oziroma najmanjša vrednost iz množice nenormaliziranih vrednosti.

$$x = x'(max(X) - min(X)) + min(X) \tag{4.3}$$

V nadaljevanju modeliranja smo se osredotočili na mero uspešnosti napovedi hitrosti vetra. Uporabili smo *koren kvadratne povprečne napake* (angl. *root mean squared error* - *RMSE*). Ta je najstrožja mera natančnosti napovedi, ki močneje penalizira (kaznuje) večja odstopanja med dejanskim gibanjem in napovedjo spremenljivke. Definiramo ga z enačbo 4.4.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i' - y_i)^2}{N}}, \tag{4.4}$$

kjer so v našem primeru y_i' napovedane vrednosti za čas $t + \Delta t$, y_i merjene vrednosti v času $t + \Delta t$ in N število vseh primerov v množici merjenih vrednosti hitrosti vetra.

Denormalizirane podatke (*hitrosti vetra v času* $t + \Delta t$ in *napovedane hitrosti vetra za čas* $t + \Delta t$) smo uporabili za izračun RMSE.

Nato smo rezultate uporabe naučenih metod (*korekcija*) na testnih podatkih odšteli *napovedanim hitrostim vetra za čas* $t + \Delta t$ (enačba 4.5), to razliko pa ponovno uporabili v enačbi RMSE.

$$popravljenaNapoved = napoved - korekcija \tag{4.5}$$

Tako smo poleg RMSE izračunanega s *hitrostjo vetra v času* $t + \Delta t$ in *napovedano hitrostjo vetra za čas* $t + \Delta t$ pridobili še RMSE izračunan s *hitrostjo vetra v času* $t + \Delta t$ in *popravljenoNapovedjo* - rezultatom bomo v nadaljevanju pravili *RMSE nepopravljene napovedi* in *RMSE popravljene napovedi.*

Preden smo vhodne parametre pred uporabo normalizirali, smo imeli težave s konvergenco nevronskih mrež, saj do te ni prišlo tudi v zelo velikem *številu iteracij* skozi učno množico. V tem primeru bi lahko postopek prisilno ustavili s parametrom za *največje število iteracij* (*angl. stepmax*). Po normalizaciji podatkov se z nastavljanjem le tega nismo ubadali, saj je do konvergenc na nastavljeni nevronski mreži prišlo po približno od 80 do 200 iteracijah. Omenjeni parameter smo zato ohranili na privzeti vrednosti (100000).

Celoten postopek modeliranja v okolju R je opisan s psevdokodo v *algoritmu 2.*

Algoritem 2: Modeliranje

```
Vhod: 11 datotek s podatki za vsako CVP
Izhod: rezultati v obliki grafov
vsiPodatki ← vseh 11 datotek s podatki;
vsiPodatki ← (vsiPodatki["napovedanaHitrostVetra"] < 40);
napaka ← vsiPodatki["napovedanaHitrostVetra"] - vsiPodatki["hitrostVetraVCasu t + Δt"];
vsiPodatki.Dodaj(napaka);
vsiPodatki["NapZracniPritisk"] ← (vsiPodatki["NapZracniPritisk"] / 100);
AnalizaPodatkov(vsiPodatki);
formula ← napovedanaHitrostVetra + trenutnaHitrostVetra + trenutnaTemperaturaZraka +
napovedanaTemperaturaZraka + napovedanaTemperaturaPovrsja;
Normalizacija(vsiPodatki);
vsiUcniPodatki ← VsiPodatkiKiNisoIzLeta2014(vsiPodatki);
vsiTestniPodatki ← VsiPodatkiIzLeta2014(vsiPodatki);
for poskus ← 1 to Območje(1:5) do
    for steviloUrVnaprej ← 1 to Območje(1:11) do
        ucniPodatki ← (vsiUcniPodatki["steviloUrVnaprej"] == steviloUrVnaprej);
        testniPodatki ← (vsiTestniPodatki[steviloUrVnaprej"] == steviloUrVnaprej);
        hitrostVetraVCasu t + Δt = testniPodatki["hitrostVetraVCasu t + Δt"];
        napovedanaHitrostVetra = testniPodatki["napovedanaHitrostVetra"];
        linearnaRegresija ← NauciLinearnoRegresijo(ucniPodatki,formula);
        korekcijaLinearneRegresije ← TestirajLinearnoRegresijo(linearnaRegresija,
        testniPodatki);
        nevronskaMreža ← NauciNevronskoMrežo(ucniPodatki,formula);
        korekcijaNevronskeMreže ← TestirajNevronskoMrežo(nevronskaMreža, testniPodatki);
        Denormalizacija(hitrostVetraVCasu t + Δt, napovedanaHitrostVetra,
        korekcijaLinearneRegresije, korekcijaNevronskeMreže);
        RMSE[steviloUrVnaprej] ← IzračunRMSE(napovedanaHitrostVetra,
        hitrostVetraVCasu t + Δt);
        napovedanaHitrostVetraPopravljenaZLinearnoRegresijo ← napovedanaHitrostVetra -
        korekcijaLinearneRegresije;
        RMSEpopravljenZLinearnoRegresijo[steviloUrVnaprej] ←
        IzračunRMSE(napovedanaHitrostVetraPopravljenaZLinearnoRegresijo,
        hitrostVetraVCasu t + Δt);
        napovedanaHitrostVetraPopravljenaZNevronskoMrežo ← napovedanaHitrostVetra -
        korekcijaNevronskeMreže;
        RMSEpopravljenZNevronskoMrežo[poskus][steviloUrVnaprej] ←
        IzračunRMSE(napovedanaHitrostVetraPopravljenaZNevronskoMrežo,
        hitrostVetraVCasu t + Δt);
    end
end
PovprečiSŠtevilomPoskusov(RMSEpopravljenZNevronskoMrežo);
PrikažiNaGrafu(RMSE,
RMSEpopravljenZLinearnoRegresijo,RMSEpopravljenZNevronskoMrežo);
```

Poglavje 5

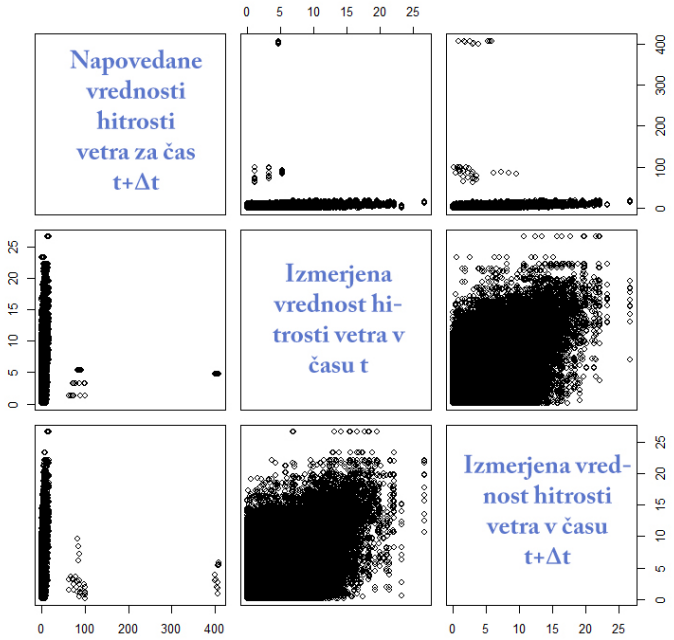
Rezultati

V okolju R smo opravili analizo nekaterih podatkov, rezultate pa grafično prikazali z vgrajenimi ukazi. Za nekatere smo uporabili programsko knjižnico *hexbin* [2], katere namen je združevanje in prikaz podatkov visoke gostote.

Analizirali smo porazdeljenost podatkov, nekatere izmed analiz pa tudi grafično prikazali, kar je razvidno iz *matričnega grafa 5.1*. Matrični graf se največkrat uporablja, kadar želimo ugotoviti medsebojno korelacijo[1] večih spremenljivk. Diagonala grafa prikazuje imena spremenljivk (parametrov), ostala polja pa korelacijo ter razpršenost spremenljivk, imena katerih najdemo na diagonali, oziroma na vertikali in horizontali polja.

Ugotovili smo, da so vse vrednosti vseh parametrov, razen napovednih podatkov za hitrost vetra, razpršene v mejah normale. Izmerjenih vrednosti v času t in $t + \Delta t$ je tako največ v območju med 0 in 15 m/s, sežejo pa vse do 25 m/s. Pravtako smo ugotovili, da v obdobju zadnjih nekaj let napovedni podatki hitrosti vetra za nekatere CVP presežejo vrednost 60 m/s, vse do 400 m/s, kar se natančneje vidi na *grafih 5.2 in 5.3*. Delež omenjenih vrednosti

[1] Korelacija ali korelacijski koeficient je številska mera, ki predstavlja moč linearne povezanosti dveh spremenljivk. Statistična veda s korelacijo v splošnem označuje odvisnost dveh spremenljivk v statistični populaciji ali populacijah.

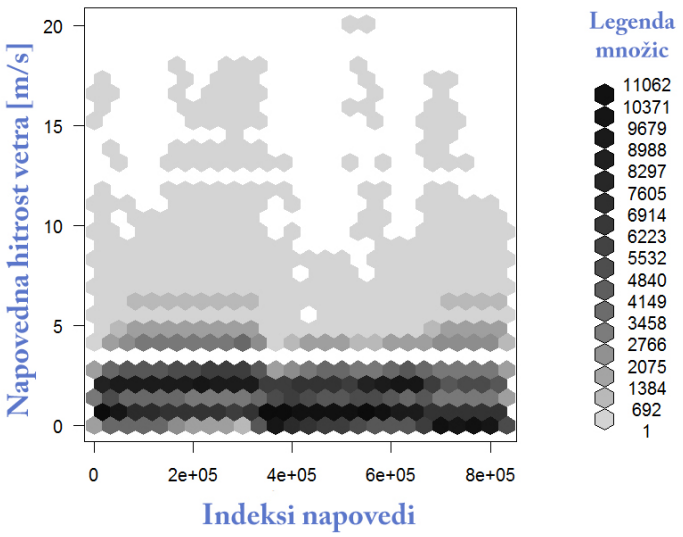


Slika 5.1: Matrični graf razpršenosti nekaterih parametrov za CVP *Črni Kal* za obdobje od januarja 2012 do junija 2014. Slednji prikazuje razpršenost in medsebojno korelacijo parametrov. Diagonala grafa prikazuje imena spremenljivk (parametrov), ostala polja pa korelacijo ter razpršenost spremenljivk, imena katerih najdemo na diagonali oziroma na vertikali in horizontali polja. Enote na vseh oseh so *m/s*.

med podatki za vse CVP je okoli 0.02 %.

Ker na območju Slovenije nikoli ni bilo uradno izmerjenih hitrosti vetra večjih od okoli 60 m/s oziroma 220 km/h [31], smo po pogovoru z meteorologi omenjene podatke označili kot napačne oziroma *šum*. Pred procesom modeliranja smo jih zato izločili, omeniti pa velja, da meteorologi teh anomalij doslej niso opazili.

Zanimala nas je porazdelitev *napake* za vse napovedne ure skupaj (od 1 do 11). Slednja se za različne CVP bistveno ne razlikuje; primer za CVP *Črni Kal* je prikazan v *grafu 5.4*, porazdelitev za vse CVP skupaj vidimo v

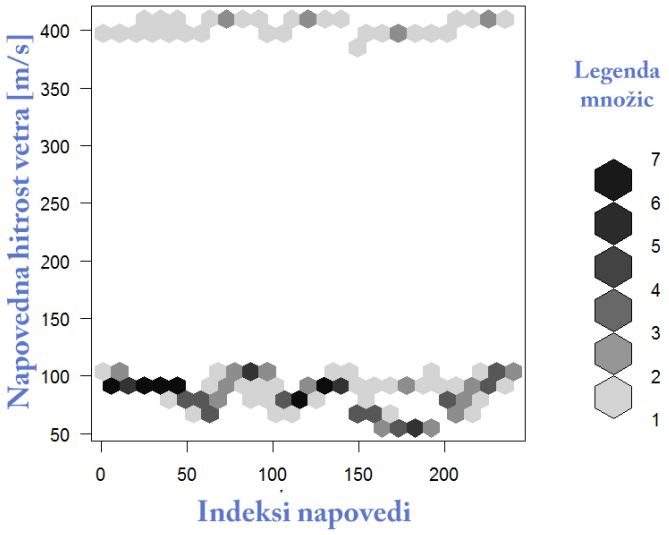


Slika 5.2: Graf prikazuje porazdeljenost napovednih hitrosti vetra, katere so manjše od 40 m/s, za vse CVP za obdobje od januarja 2012 do junija 2014. Intenziteta sivine pomeni število meritev za neko vrednost, kar je razvidno iz legende grafa.

grafu 5.5.
Ugotovimo, da so vrednosti *napake* normalno porazdeljene okoli vrednosti *0 m/s* skozi vso obdobje analize. V *grafu 5.6* lahko vidimo porazdeljenost napake za posamezne napovedne ure. Razvidno je, da je *napaka* za vsako napovedno uro porazdeljena pravtako enakomerno in okoli vrednosti 0 m/s.

Ko smo že omenili, nas je zanimala primerjava rezultatov med linearno regresijo in nevronsko mrežo z 0 skritimi nivoji (ostala arhitektura je ista kot v opisu v *podpoglavju 4.4.1*).

V *grafu 5.7* vidimo, da je za CVP *Črni Kal* korekcija z nevronskimi mrežami v začetnih urah slabša kot korekcija izračunana z linearno regresijo. Rezultati oziroma korekcija tako nastavljene nevronske mreže so od rezultatov linearne regresije slabši na vseh testiranih CVP.

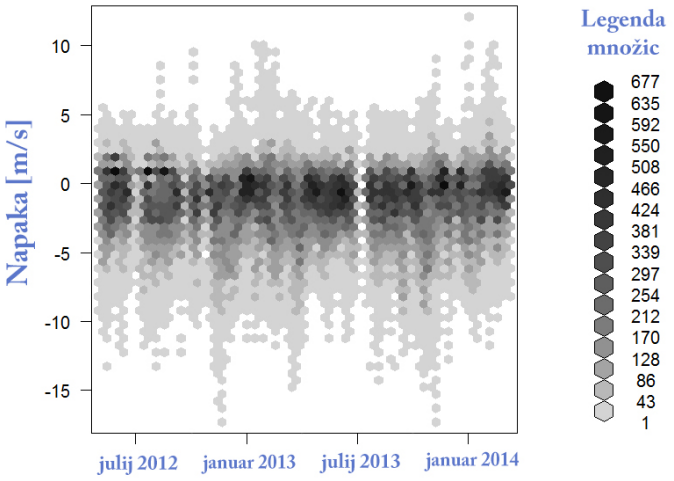


Slika 5.3: Graf prikazuje porazdeljenost napovednih hitrosti vetra, katere so večje od 40 m/s in smo jih pred postopkom modeliranja izločili, za vse CVP za obdobje od januarja 2012 do junija 2014. Intenziteta sivine pomeni število meritev za neko vrednost, kar je razvidno iz legende grafa.

Zanimala nas je primerjava različnih učnih algoritmov, kateri so nam bili na voljo v programski knjižnici *neuralnet*. Iz *grafa 5.8* je razvidno, da razlik med njimi v našem primeru praktično ni oziroma so zanemarljive.

Razlike med različnimi pragovnimi vrednostmi (*engl. threshold*) so za CVP *Črni Kal* razvidne v *grafu 5.9*. Manjše ali večje vrednosti od 0.5 so pri testiranjih običajno doprinesle slabše rezultate, zato smo vsa nadaljnja testiranja opravili s slednjo nastavljeno vrednostjo.

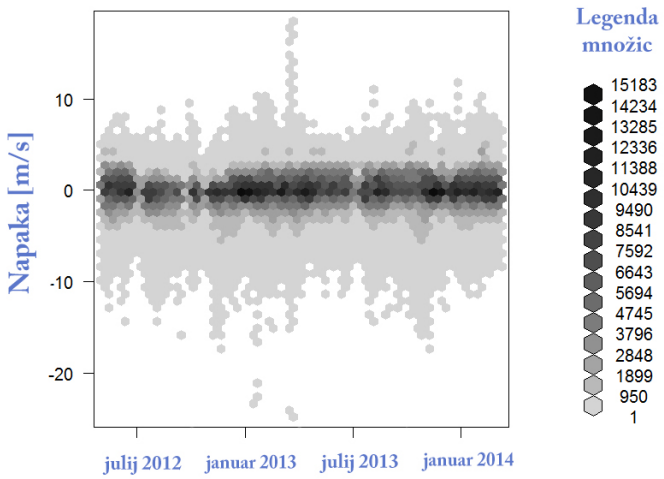
Za *napako* oziroma nepopravljen RMSE hitrosti vetra sistema INCA gre pričakovati, da le ta skupaj z napovedno uro narašča. Na podlagi samo naših



Slika 5.4: Porazdelitev napovedne napake hitrosti vetra za CVP *Črni Kal* za obdobje od januarja 2012 do julija 2014. Intenziteta sivine pomeni število meritev za neko vrednost, kar je razvidno iz legende grafa.

rezultatov lahko sklepamo, da temu ni tako, kasnejšnji pogovor z meteorologi pa je razkril, da je padanje napake z napovedno uro za nekatere CVP normalen pojav. Vsekakor omenjena anomalija odpira možnosti za nadaljnje analize podatkov.

Povprečno korekcijo za CVP *Črni Kal* vidimo na *grafu 5.10*, za CVP *Ribnik* pa na *grafu 5.13*. Opazimo, da korekcija (to je razlika med nepopravljenim in popravljenim RMSE) niti približno ni konstantna, vendar je za prve napovedne ure večja; v območju med 0.3 in 1 m/s.
Rezultati, kjer je korekcija v povprečju največja, so za CVP *Rupovščica* prikazani v *grafu 5.11*. Opazimo, da je s popravljena napoved natančnejša za do 1.2 m/s v prvih dveh napovednih urah.
Zanimali so nas tudi rezultati za CVP, kjer je korekcija najmanjša. Slednja, za CVP *Savinja most*, je prikazana na *grafu 5.12*, iz katerega je razvidno, da je popravljena napovedna hitrost vetra v povprečju opazno boljša samo za prve 2 napovedne ure.



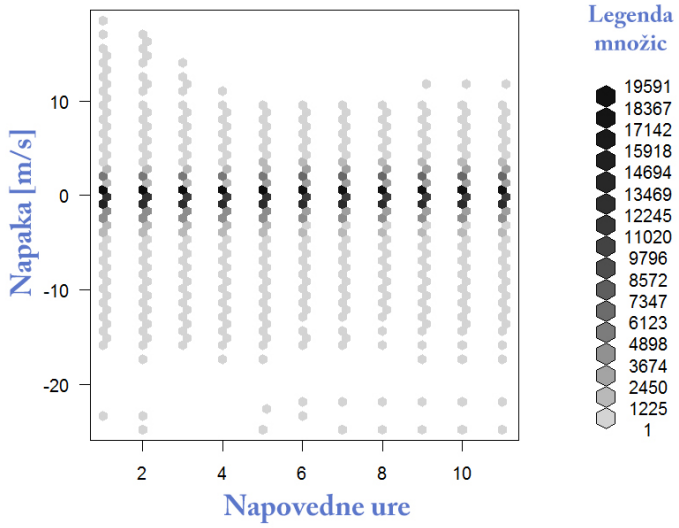
Slika 5.5: Porazdelitev napovedne napake hitrosti vetra za vse CVP za obdobje od januarja 2012 do julija 2014. Intenziteta sivine pomeni število meritev za neko vrednost, kar je razvidno iz legende grafa.

Da bi rezultate posplošili, smo popravljanje napovedi testirali na vseh CVP, katerih podatki so nam bili na voljo, rezultate (RMSE) pa povprečili.

Iz rezultatov prikazanih v *grafu 5.14* pa lahko sklepamo, da je popravljena napoved hitrosti vetra v prvih napovednih urah za povprečno CVP natančnejša od nepopravljene za do 0.8 m/s. V kasnejših napovednih urah korekcija upade tja do 0.4 m/s.

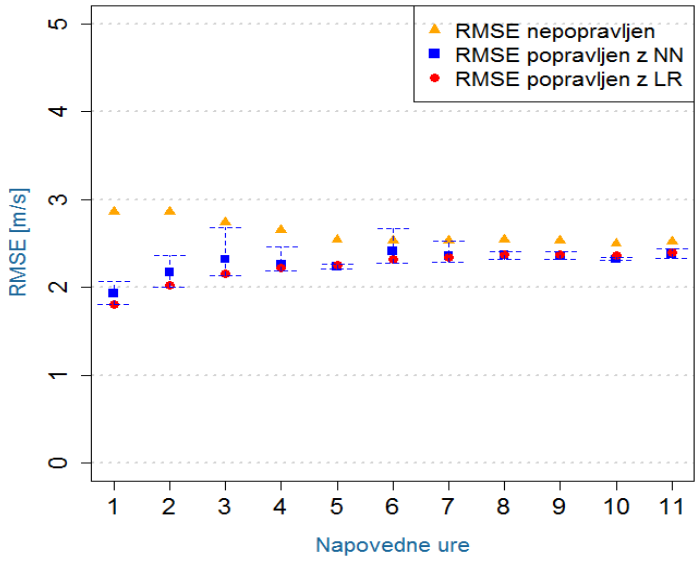
V slednjem poskusu (z več CVP) smo z uporabo linearne regresije tako pridobili naslednjo regresijsko enačbo:

*korekcija = 0.01851 + 0.42224 * napovedana hitrost vetra - 0.45035 * trenutna hitrost vetra - 0.01052 * napovedana temperatura zraka + 0.02665 * trenutna temperatura zraka - 0.02319 * napovedana temperatura površja*

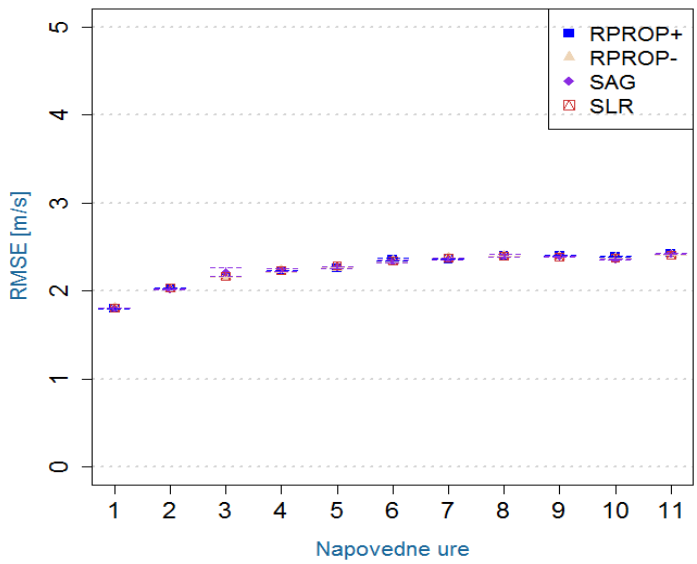


Slika 5.6: Graf porazdeljenosti napake hitrosti vetra za vse CVP za specifične napovedne ure, za katere se napoveduje. Podatki so iz obdobja od januarja 2012 do junija 2014. Intenziteta sivine pomeni število meritev za neko vrednost, kar je razvidno iz legende grafa.

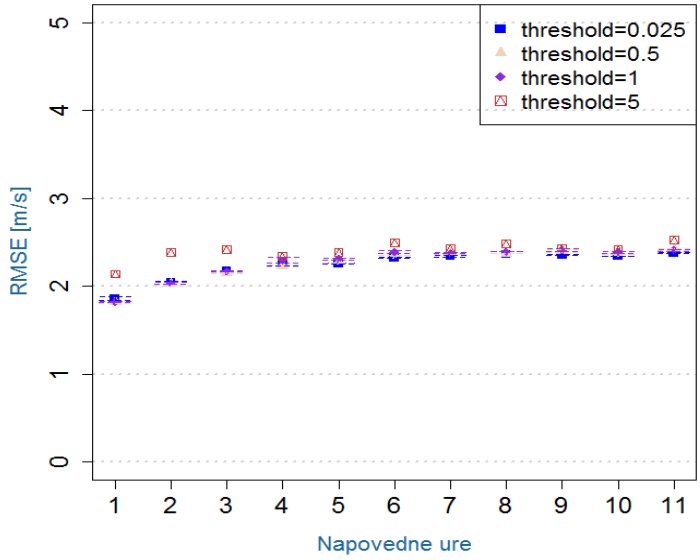
Ker so podatki normalizirani, je iz zgornje enačbe po absolutnih vrednostih pridobljenih koeficientov razvidno, da sta napovedna ter izmerjena hitrost vetra najpomembnejša prediktorja. Da bi se v rezultate bolje prepričali, smo uporabili programsko knjižnico *CORElearn* [28], katere namen je podpora različnim metodam strojnega učenja ter metodam za ocenjevanje atributov. S funkcijo *attrEval* smo analizirali pomembnost parametrov oziroma pridobili naslednje rezultate:
0.999 za napovedno hitrost vetra, 0.432 za trenutno hitrost vetra, 0.124 za trenutno temperaturo zraka, 0.001 za napovedano temperaturo površja in 0.000 za napovedano temperaturo zraka.
Napovedna hitrost vetra, trenutna hitrost vetra in trenutna temperatura zraka so bili z omenjeno funkcijo ocenjeni kot najpomembejši atributi, napovedani temperaturi površja in zraka pa sta bili ocenjeni kot nepomembni.



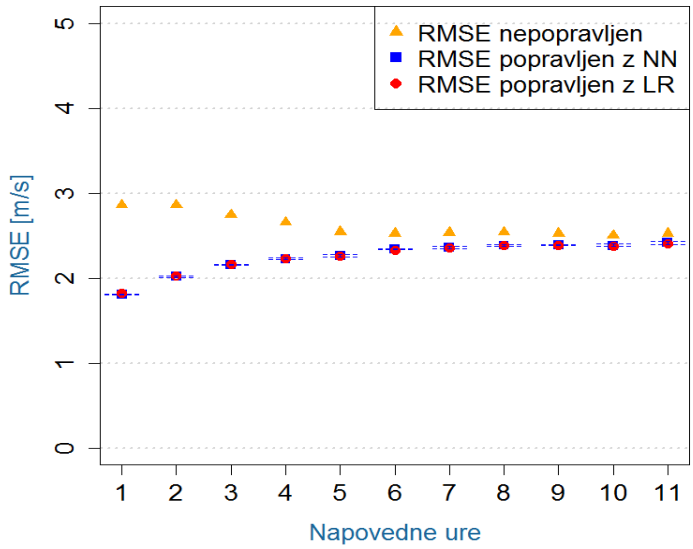
Slika 5.7: Graf prikazuje RMSE nepopravljene hitrosti vetra (oranžne točke) in RMSE popravljene napovedne hitrosti vetra. Slednji je odšteta korekcija pridobljena z nevronskimi mrežami z 0 skritimi nivoji (modre točke) in linearno regresijo (rdeče točke). Modre točke so povprečje rezultatov vseh iteracij napovednih ur, modre črte pa največji oziroma najmanjši rezultat popravljenega RMSE za določeno napovedno uro. Rezultati so prikazani za CVP *Črni Kal*, pridobljeni na podatkih iz obdobja od januarja 2012 do julija 2014.



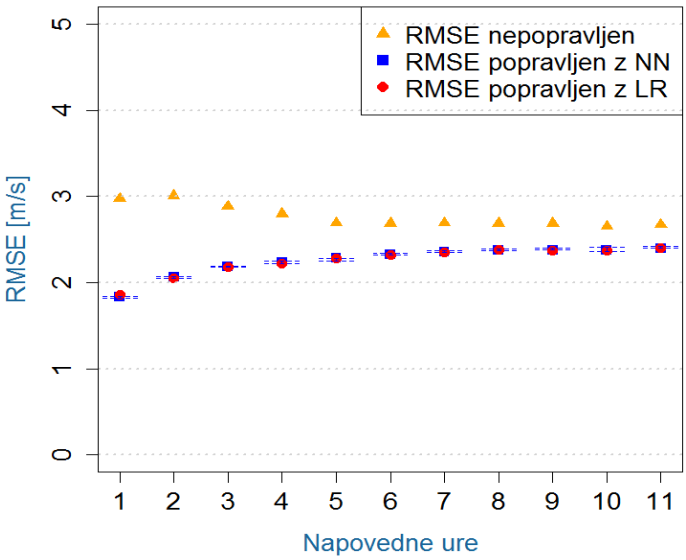
Slika 5.8: Zgornji graf prikazuje RMSE popravljene napovedne hitrosti vetra, katere so bile popravljene z nevronskimi mrežami z različnimi učnimi algoritmi. Rezultati so bili pridobljeni na podatkih CVP *Črni Kal*. RPROP+ je kratica za *resilient backpropagation with weight backtracking*, RPROP- za *resilient backpropagation without weight backtracking*, SAG za *globally convergent algorithm (learningrate associated with the smallest absolute gradient)* in SLR za *globally convergent algorithm (learningrate associated with the smallest learningrate itself)*.



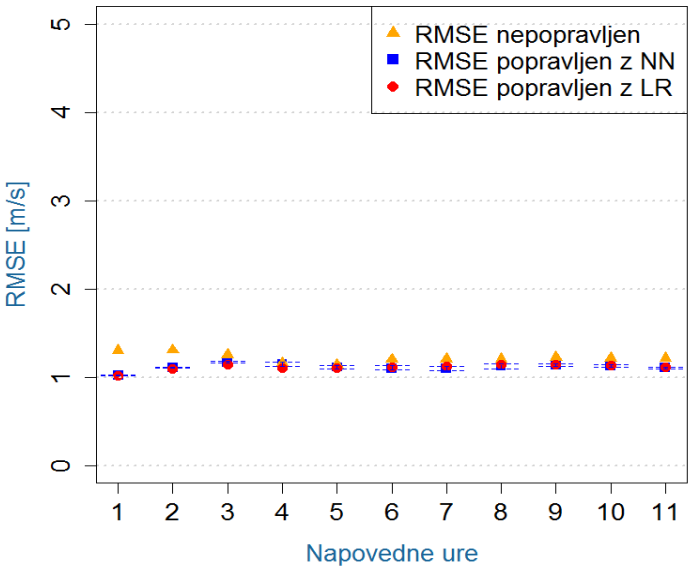
Slika 5.9: RMSE popravljenih hitrosti vetra, popravljenih z nevronskimi mrežami nastavljenimi na različne pragovne vrednosti (*angl. threshold*) so prikazanai na slednjem grafu. Pragovno vrednost smo za kasnejša testiranja nastavili na 0.5.



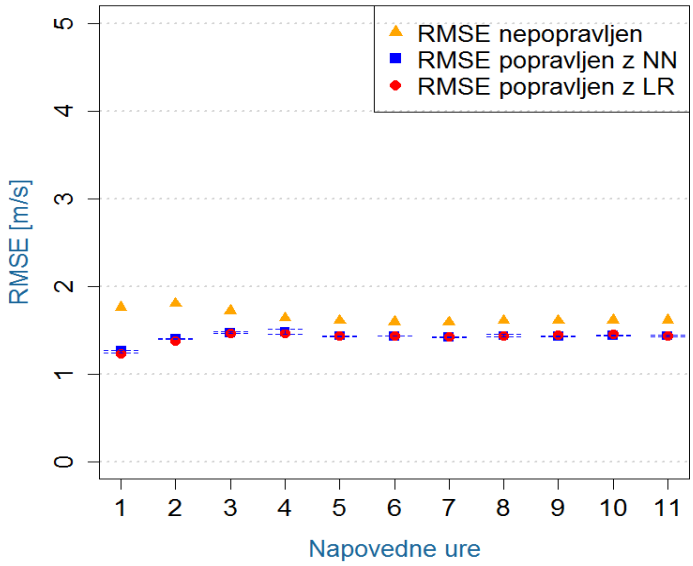
Slika 5.10: Graf prikazuje RMSE nepopravljenih in popravljenih napovednih hitrosti vetra z nevronskimi mrežami in linearno regresijo za CVP *Črni Kal*.



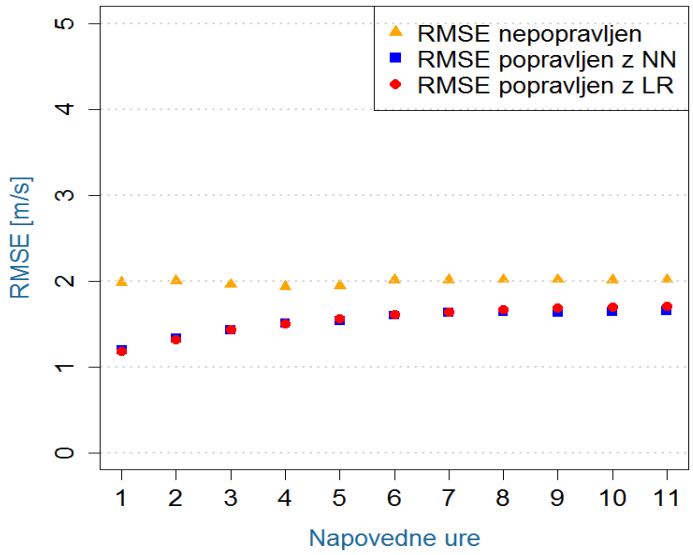
Slika 5.11: Graf prikazuje RMSE nepopravljenih in popravljenih napovednih hitrosti vetra z nevronskimi mrežami in linearno regresijo za CVP *Rupovščica*. Za slednjo CVP smo pridobili največjo povprečno korekcijo.



Slika 5.12: Graf prikazuje RMSE nepopravljenih in popravljenih napovednih hitrosti vetra z nevronskimi mrežami in linearno regresijo za CVP *Savinja most*. Za slednjo CVP smo pridobili najmanjšo povprečno korekcijo.



Slika 5.13: Graf prikazuje RMSE nepopravljenih in popravljenih napovednih hitrosti vetra z nevronskimi mrežami in linearno regresijo za CVP *Ribnik*.



Slika 5.14: Graf prikazuje RMSE nepopravljenih in popravljenih napovednih hitrosti vetra z nevronskimi mrežami in linearno regresijo za vse CVP.

Poglavje 6

Sklepne ugotovitve

Cilj diplomskega dela je bil pokazati, ali lahko z uporabo regresijskih metod ali umetnih nevronskih mrež pridobimo korekcijo za trenutne napovedi hitrosti vetra, s katero bi bile le te bolj natančne kot sicer.

Analiza podatkov je pokazala, da v povprečju 0.02 % napovednih vrednosti hitrosti vetra modela INCA presegajo meje normalnih območij za Slovenijo tja do 400 m/s. Po pregledu datotek, ki jih pripravi omenjeni model za ta čas, smo ugotovili, da gre za napako pri interpretaciji oziroma predstavitvi rezultatov. Omenjene podatke smo izločili.

Slabost nevronske mreže se kaže predvsem v tem, da nima pojasnjevalne moči, ker deluje kot *črna škatla* (*angl. black box*), v katero pošljemo podatke in na izhodu dobimo neke rezultate. Ker točno določenih pravil, ki bi omogočala nastavitev parametrov za optimalno modeliranje nevronskega modela, ni, ampak samo splošna priporočila, smo pri tem bili bolj prepuščeni lastni iznajdljivosti in izkušnjam. Ugotovili smo, da pri uporabi nevronskih mrež v našem primeru optimalno nastavljanje parametrov nima ključnega pomena. Največjo vlogo imata *prag*, katerega smo nastavili na 0.5 in število skritih nivojev. Zaradi sklepa na podlagi člankov in ker so rezultati z 0 skritimi nivoji v našem primeru slabi, smo število slednjih nastavili na 1.

Ob primerjavi nevronskih mrež z linearno regresijo lahko zaključimo, da prve kljub kompleksnosti niso primerne za uporabo pri napovedovanju vre-

menskih parametrov oziroma popravljanju napovedi teh. Zaradi enostavnosti uporabe in časovne zahtevnosti linearne regresije ter nepotrebnim ukvarjanjem z nastavljanjem parametrov, smo se odločili, da je slednja izbira boljša. Pojasniti, zakaj kompleksnejše nevronske mreže delujejo slabše, pa nam zaenkrat ni uspelo.

Rezultati kažejo, da je z uporabo regresijskih metod ali umetnih nevronskih mrež možno izboljšati napovedi hitrosti vetra. Korekcija se za različne CVP razlikuje, za prvih 6 napovednih ur je v povprečju večja kot za ostale ure in sega tja do 1 m/s. Ta se z daljšanjem napovedi zmanjšuje tja do 0.2 m/s.

6.1 Možnosti za nadaljnje delo

Napovedovanje oziroma izboljšanje napovedovanja vremenskih parametrov je zelo široko področje.

Natančnejša analiza podatkov bi morda razkrila več zanimivih podmnožic podatkov, katere lastnosti bi lahko uporabili pri uporabi metod strojnega učenja.

Z metodami strojnega učenja bi se poleg hitrosti vetra lahko osredotočili tudi na napovedi ostalih vremenskih parametrov. Vsekakor bi se tega problema popravljanja napovedovanja lahko lotili z drugačnimi pristopi. Poskusiti bi šlo z drugimi topologijami umetnih nevronskih mrež, drugimi regresijskimi ali klasifikacijskimi metodami kot so regresijska drevesa ali naključni gozdovi ...

V kolikor se bomo odločili rešitev tega diplomskega dela implementirati v gospodarstvo, bo potreben popravek programske kode na način, da se bodo naučene umetne nevronske mreže lahko shranile ter tako ponovno uporabljale za namene popravljanja v realnem času. V *R* programskem okolju bi za to uporabili možnost shranjevanja ter nalaganja delovnega okolja. Čez čas bi po morebitni implementaciji rezultate ponovno primerjali z rezultati nepopravljenih napovedi.

Dodatek

Programska koda za pripravo podatkov

```

# Denis Kotnik
# Prilagodljivo kratkorocno napovedovanje lokalnih vremenskih parametrov
# Mentor: doc. dr. Matjaz Kukar
# Fakulteta za racunalnistvo in informatiko
# September 2014; v1.5

#######################################
# Knjizice
#######################################
# Pandas data frame (uporabne povezave: http://pandas.pydata.org/pandas-docs/dev/dsintro.html, http://pandas.pydata.org/pandas-docs/dev/10min.html, https://pypi.python.org/pypi/pandas#downloads )
import pandas as pd
# Za numericne operacije
import numpy as np
import random
# Za sistemske funkcije (argumenti ...)
import sys
# Za manipulacijo z datumom (uporabne povezave: http://www.lfd.uci.edu/~gohlke/pythonlibs/#python-dateutil http://www.lfd.uci.edu/~gohlke/pythonlibs/#six )
from datetime import datetime as dt
from datetime import timedelta
```

```
import datetime as dt2
# Za matematicne funkcije
import math

#####################################
# Nastavitve
#####################################
# station_ID = sys.argv[1]
# Za koliko ur naprej nas zanima napoved
houraAhead = int(sys.argv[1])
wind = 15
station_ID = 59 # Crni Kal na primorskem
# Ali zelimo, da so podatki urejeni po uri (True) ali po datumu (False)
orderedByHour = True

# inputFileNameObservationsPreasure = "Mes_Value_18_2_ordered"
inputFileNameObservationsTemp = "Mes_Value_1_59_ordered"
inputFileNameObservationsWind = "Mes_Value_15_59_ordered"
inputFileNameStation = "StationMeasurement"
inputFileNameForecast = "WeatherForecast_59_ordered"

fileExtension = ".csv"

outputFileName = "learningDataTemp_" + str(station_ID) + "_" + str(houraAhead)

# Imena atributov, kateri nas zanimajo in jih hocemo dodati v ucno mnozico
# Paziti je potrebno, da seznami nimajo skupnih parametrov. V tem primeru uporabi funkcijo
# observation.rename(columns={"ID" : "ID_Mes_Value"}, inplace = True)

# ID|fk_Station_Mes_ID|Mes_DateTime|Mes_Value|MeasurementIndex|Prediction
observationParameters = ['Mes_DateTime','Mes_Value']
futureObservationParameters = ['Mes_DateTime','Mes_Value']
```

```
53 # ID|fk_Station_ID|fk_Measurement_ID|MeasurementIndex|
       RowDataOrder|LocalMeasurementCode|DisplayOrder|Active
54 stationParameters = ['fk_Measurement_ID']
55 # ID|fk_Station_ID|Mes_DateTime|MinuteForecast|Parameters
56 # Atribut Parameters mora biti obvezno vsebovan
57 forecastParameters = ['fk_Station_ID','DateTime','
       MinuteForecast','Parameters']
58 currentObservationTempParameters = ['Mes_DateTime','Mes_Value']
59 #currentObservationPreasureParameters = ['Mes_DateTime','
       Mes_Value']
60 dataSeparator = "|"
61 decimalSeparator = ","
62 dateFormat = "%Y-%m-%d %H:%M:%S"
63 ######################################
64 ######################################
65
66 # Date parser
67 def dateParser(date):
68     Mes_DateTime = dt.strptime(date, dateFormat)
69     return Mes_DateTime
70
71 # Preberemo datoteke
72 observationsWind = pd.read_csv(inputFileNameObservationsWind +
       fileExtension, sep = dataSeparator, parse_dates = ["
       Mes_DateTime"], date_parser = dateParser, decimal =
       decimalSeparator)
73 stationMeasurements = pd.read_csv(inputFileNameStation +
       fileExtension, sep = dataSeparator)
74 weatherForecasts = pd.read_csv(inputFileNameForecast +
       fileExtension, sep = dataSeparator)
75 observationsTemp = pd.read_csv(inputFileNameObservationsTemp +
       fileExtension, sep = dataSeparator, decimal =
       decimalSeparator)
76
77 print "#########################################"
78 print "Datoteke uspesno nalozene v pomnilnik."
79 print "#########################################"
80
```

```
81 # Naslednji korak je potreben zaradi velikosti datoteke. UPDATE
       : le pri 32-bitnem Pythonu
82 #observationsWind = pd.concat(observationsWind, ignore_index=
       True)
83 #df = pd.concat(chunk for chunk in observationsWind)
84
85 # Iz tabele StationMeasurement izberemo vrstico ID meritev za
       specificno postajo in specificen parameter (15 = veter)
86 station = stationMeasurements[(stationMeasurements["
       fk_Station_ID"] == station_ID) & (stationMeasurements["
       fk_Measurement_ID"] == wind)]
87 # ID cestnovremenske postaje
88 StationMeasurement_ID = station.iloc[0]["ID"]
89 # Izberemo le parametre, ki nas zanimajo
90 station = pd.DataFrame(station.loc[:, stationParameters])
91
92 NNData = pd.DataFrame()
93
94 lastAddedMeasurement = None
95 # Za vsako meritev
96 vnosov = 0
97 for counter in range(0, observationsWind["Mes_DateTime"].count
       ()):
98     # Meritev z vsemi atributi
99     currentObservation = observationsWind.iloc[[counter]]
100
101     # Atributa meritve: ID fk_Station_Mes_ID ter cas meritve
102     fk_Station_Mes_ID = currentObservation.iloc[0]["
           fk_Station_Mes_ID"]
103     Mes_DateTime = currentObservation.iloc[0]["Mes_DateTime"]
104
105     # Meritev prive v postev, ce je bila izmerjena med XX:00 in
           XX:06
106     # Ce se bo nadgrajevalo z uporabo drsecega okna: http://
           stackoverflow.com/questions/4118526/find-the-closest-
           hour
107     if ((fk_Station_Mes_ID == StationMeasurement_ID) & (int(
           Mes_DateTime.strftime("%M")) <= 6) & (
```

```
        lastAddedMeasurement == None or Mes_DateTime.strftime("%Y-%m-%d %H") != lastAddedMeasurement)):
108         print "Zbiranje podatkov za ID postaje", fk_Station_Mes_ID, "/", station_ID, "za cas", Mes_DateTime
109
110         # Da bomo za vsako uro vzeli le eno meritev
111         lastAddedMeasurement = currentObservation.iloc[0]["Mes_DateTime"].strftime("%Y-%m-%d %H")
112
113         # Vse minute med 00 in 06 zaokrozimo na 00
114         roundedMes_DateTime = dt.strptime(str(Mes_DateTime), dateFormat).replace(minute = 00)
115         # Izberemo le parametre, ki nas zanimajo.
116         currentObservation = pd.DataFrame(currentObservation.loc[:, observationParameters])
117         currentObservation.rename(columns={"Mes_Value" : "TrenutnaHitrostVetra"}, inplace = True)
118         currentObservation.rename(columns={"Mes_DateTime" : "DatumTrenutneMeritveVetra"}, inplace = True)
119
120         currentObservationTemp = observationsTemp[(observationsTemp["Mes_DateTime"] >= str(Mes_DateTime - dt2.timedelta(minutes = 3))) & (observationsTemp["Mes_DateTime"] < str(Mes_DateTime + dt2.timedelta(minutes = 3)))]
121         if len(currentObservationTemp) > 1:
122             currentObservationTemp = currentObservationTemp.iloc[[0]]
123         currentObservationTemp =  pd.DataFrame(currentObservationTemp.loc[:, currentObservationTempParameters])
124         currentObservationTemp.rename(columns={"Mes_Value" : "TrenutnaTemperaturaZraka"}, inplace = True)
125         currentObservationTemp.rename(columns={"Mes_DateTime" : "DatumMeritveTempZraka"}, inplace = True)
126
```

```
127         # Pridobimo izmerjeno vrednost cez n-ur; ter preimenuje
                atribute
128         futureObservation = observationsWind[(observationsWind[
                "Mes_DateTime"] >= str(Mes_DateTime + dt2.timedelta
                (hours = hoursAhead) - dt2.timedelta(minutes = 3)))
                 & (observationsWind["Mes_DateTime"] < str(
                Mes_DateTime + dt2.timedelta(hours = hoursAhead) +
                dt2.timedelta(minutes = 3)))]
129         if len(futureObservation) > 1:
130             futureObservation = futureObservation.iloc[[0]]
131         futureObservation = pd.DataFrame(futureObservation.loc
                [:, futureObservationParameters])
132         futureObservation.rename(columns={"Mes_Value" : "
                HitrostVetraCezN"}, inplace = True)
133         futureObservation.rename(columns={"Mes_DateTime" : "
                DatumMeritveVetraCezN"}, inplace = True)
134         #print "Prihodnja",futureObservation
135
136         # Pridobimo izmerjeno vrednost cez n-ur; ter preimenuje
                atribute
137         futureObservationTemp = observationsTemp[(
                observationsTemp["Mes_DateTime"] >= str(
                Mes_DateTime + dt2.timedelta(hours = hoursAhead) -
                dt2.timedelta(minutes = 3))) & (observationsTemp["
                Mes_DateTime"] < str(Mes_DateTime + dt2.timedelta(
                hours = hoursAhead) + dt2.timedelta(minutes = 3)))]
138         if len(futureObservationTemp) > 1:
139             futureObservationTemp = futureObservationTemp.iloc
                    [[0]]
140         futureObservationTemp = pd.DataFrame(
                futureObservationTemp.loc[:,
                futureObservationParameters])
141         futureObservationTemp.rename(columns={"Mes_Value" : "
                TrenutnaTemperaturaZrakaCezN"}, inplace = True)
142         futureObservationTemp.rename(columns={"Mes_DateTime" :
                "DatumMeritveTempZrakaCezN"}, inplace = True)
143         #print "Prihodnja",futureObservation
144
```

```
145         # Izberemo ustrezno napoved. Gledamo na cas napovedi (
                zamaknjen za n ur nazaj), za koliko casa naprej se
                napoveduje ter ID postaje
146         forecast = weatherForecasts[(weatherForecasts["DateTime
                "] == str(roundedMes_DateTime)) & (weatherForecasts
                ["MinuteForecast"] == (hoursAhead * 60)) & (
                weatherForecasts["fk_Station_ID"] == station_ID)]
147         if len(forecast) > 1:
148             forecast = forecast.iloc[[0]]
149         # Izberemo le parametre, ki nas zanimajo
150         forecast = pd.DataFrame(forecast.loc[:,
                forecastParameters])
151         forecast.rename(columns={"DateTime" : "DatumNapovedi"},
                 inplace = True)
152         forecast.rename(columns={"MinuteForecast" : "
                MinuteNapoved"}, inplace = True)
153
154         # Ce imamo eno napoved za vsako meritev, potem podatke
                zdruzi # and len(currentObservationPreasure) == 1
155         if (len(currentObservation) == 1 and len(station) == 1
                and len(forecast) == 1 and len(futureObservation)
                == 1 and len(currentObservationTemp) == 1 and len(
                futureObservationTemp) == 1):
156             # Parameters
157             print "Nasel"
158             parametri = forecast.iloc[0]["Parameters"].split(',
                    ')
159             #print parametri[0]
160             parametri = {"NapTempZraka": parametri[0],"
                    NapTempRosisca": parametri[1],"NapTempPovrsja":
                     parametri[2],"NapVlaznostZraka": parametri[3],
                    "NapHitrostVetra": parametri[4],"
                    NapKolicinaPadavin": parametri[5],"
                    NapKolicinaTrdnihPadavin": parametri[6],"
                    NapVrstaPadavin": parametri[7],"
                    NapDolgovalovnoSevanje": parametri[8],"
                    NapKratkovalovnoSevanje": parametri[9],"
                    NapZracniPritisk": parametri[10],"
```

```
                    NapDelezOblacnosti": parametri[11],"
                    NapTermalnoSevanje": parametri[12],"
                    NapSolarnoSevanje": parametri[13],"NapOdprtost"
                    : parametri[14]}

            parametri = pd.DataFrame(parametri,index=[0])
            forecast = forecast.drop("Parameters", 1)

            # Reindex first element so that join command will
                work
            currentObservation.index = [1]
            station.index = [1]
            forecast.index = [1]
            parametri.index = [1]
            futureObservation.index = [1]
            currentObservationTemp.index = [1]
            #currentObservationPreasure.index = [1]
            futureObservationTemp.index = [1]

            #currentObservation = currentObservation.join(
                currentObservationPreasure)
            currentObservation = currentObservation.join(
                currentObservationTemp)
            currentObservation = currentObservation.join(
                futureObservation)
            currentObservation = currentObservation.join(
                station)
            currentObservation = currentObservation.join(
                forecast)
            currentObservation = currentObservation.join(
                parametri)
            currentObservation = currentObservation.join(
                futureObservationTemp)

            NNData = NNData.append(currentObservation)
            vnosov += 1

# First reset index and initialise Pandas objects
```

```
#NNData = NNData.reset_index(drop = True)
NNDataFinal = pd.DataFrame()



# Separate data by hour of the day
# First reset index and initialise Pandas objects
NNData = NNData.reset_index(drop = True)
NNDataFinal = NNDataFinal.append(NNData)

if orderedByHour:
    NNData00 = NNData01 = NNData02 = NNData03 = NNData04 =
        NNData05 = NNData06 = NNData07 = NNData08 = NNData09 =
        NNData10 = NNData11 = NNData12 = NNData13 = NNData14 =
        NNData15 = NNData16 = NNData17 = NNData18 = NNData19 =
        NNData20 = NNData21 = NNData22 = NNData23 = pd.
        DataFrame()
    # For every element in NNData
    ko = 0
    for counter in range(0, NNData["DatumTrenutneMeritveVetra"
        ].count()):
        ko += 1
        Mes_DateTime = dateParser(str(NNData.loc[counter, "
            DatumTrenutneMeritveVetra"]))
        print "Urejanje podatkov " + str(Mes_DateTime)
        #print int(Mes_DateTime.strftime("%H"))
        #print type(int(Mes_DateTime.strftime("%H")))
        # Every full hour has its place in Pandas objects
        if (int(Mes_DateTime.strftime("%H")) == 0):
            #NNData00 = NNData00.append(NNData[NNData["
                Mes_DateTime"] == str(Mes_DateTime)])
            NNData00 = NNData00.append(NNData.iloc[[counter]])
        elif (int(Mes_DateTime.strftime("%H")) == 1):
            #NNData01 = NNData01.append(NNData[NNData["
                Mes_DateTime"] == str(Mes_DateTime)])
            NNData01 = NNData01.append(NNData.iloc[[counter]])
        elif (int(Mes_DateTime.strftime("%H")) == 2):
```

```
            #NNData02 = NNData02.append(NNData[NNData["
                Mes_DateTime"] == str(Mes_DateTime)])
            NNData02 = NNData02.append(NNData.iloc[[counter]])
        elif (int(Mes_DateTime.strftime("%H")) == 3):
            #NNData03 = NNData03.append(NNData[NNData["
                Mes_DateTime"] == str(Mes_DateTime)])
            NNData03 = NNData03.append(NNData.iloc[[counter]])
        elif (int(Mes_DateTime.strftime("%H")) == 4):
            #NNData04 = NNData04.append(NNData[NNData["
                Mes_DateTime"] == str(Mes_DateTime)])
            NNData04 = NNData04.append(NNData.iloc[[counter]])
        elif (int(Mes_DateTime.strftime("%H")) == 5):
            #NNData05 = NNData05.append(NNData[NNData["
                Mes_DateTime"] == str(Mes_DateTime)])
            NNData05 = NNData05.append(NNData.iloc[[counter]])
        elif (int(Mes_DateTime.strftime("%H")) == 6):
            #NNData06 = NNData06.append(NNData[NNData["
                Mes_DateTime"] == str(Mes_DateTime)])
            NNData06 = NNData06.append(NNData.iloc[[counter]])
        elif (int(Mes_DateTime.strftime("%H")) == 7):
            #NNData07 = NNData07.append(NNData[NNData["
                Mes_DateTime"] == str(Mes_DateTime)])
            NNData07 = NNData07.append(NNData.iloc[[counter]])
        elif (int(Mes_DateTime.strftime("%H")) == 8):
            #NNData08 = NNData08.append(NNData[NNData["
                Mes_DateTime"] == str(Mes_DateTime)])
            NNData08 = NNData08.append(NNData.iloc[[counter]])
        elif (int(Mes_DateTime.strftime("%H")) == 9):
            #NNData09 = NNData09.append(NNData[NNData["
                Mes_DateTime"] == str(Mes_DateTime)])
            NNData09 = NNData09.append(NNData.iloc[[counter]])
        elif (int(Mes_DateTime.strftime("%H")) == 10):
            #NNData10 = NNData10.append(NNData[NNData["
                Mes_DateTime"] == str(Mes_DateTime)])
            NNData10 = NNData10.append(NNData.iloc[[counter]])
        elif (int(Mes_DateTime.strftime("%H")) == 11):
            #NNData11 = NNData11.append(NNData[NNData["
                Mes_DateTime"] == str(Mes_DateTime)])
```

```
243                 NNData11 = NNData11.append(NNData.iloc[[counter]])
244             elif (int(Mes_DateTime.strftime("%H")) == 12):
245                 #NNData12 = NNData12.append(NNData[NNData["
                        Mes_DateTime"] == str(Mes_DateTime)])
246                 NNData12 = NNData12.append(NNData.iloc[[counter]])
247             elif (int(Mes_DateTime.strftime("%H")) == 13):
248                 #NNData13 = NNData13.append(NNData[NNData["
                        Mes_DateTime"] == str(Mes_DateTime)])
249                 NNData13 = NNData13.append(NNData.iloc[[counter]])
250             elif (int(Mes_DateTime.strftime("%H")) == 14):
251                 #NNData14 = NNData14.append(NNData[NNData["
                        Mes_DateTime"] == str(Mes_DateTime)])
252                 NNData14 = NNData14.append(NNData.iloc[[counter]])
253             elif (int(Mes_DateTime.strftime("%H")) == 15):
254                 #NNData15 = NNData15.append(NNData[NNData["
                        Mes_DateTime"] == str(Mes_DateTime)])
255                 NNData15 = NNData15.append(NNData.iloc[[counter]])
256             elif (int(Mes_DateTime.strftime("%H")) == 16):
257                 #NNData16 = NNData16.append(NNData[NNData["
                        Mes_DateTime"] == str(Mes_DateTime)])
258                 NNData16 = NNData16.append(NNData.iloc[[counter]])
259             elif (int(Mes_DateTime.strftime("%H")) == 17):
260                 #NNData17 = NNData17.append(NNData[NNData["
                        Mes_DateTime"] == str(Mes_DateTime)])
261                 NNData17 = NNData17.append(NNData.iloc[[counter]])
262             elif (int(Mes_DateTime.strftime("%H")) == 18):
263                 #NNData18 = NNData18.append(NNData[NNData["
                        Mes_DateTime"] == str(Mes_DateTime)])
264                 NNData18 = NNData18.append(NNData.iloc[[counter]])
265             elif (int(Mes_DateTime.strftime("%H")) == 19):
266                 #NNData19 = NNData19.append(NNData[NNData["
                        Mes_DateTime"] == str(Mes_DateTime)])
267                 NNData19 = NNData19.append(NNData.iloc[[counter]])
268             elif (int(Mes_DateTime.strftime("%H")) == 20):
269                 #NNData20 = NNData20.append(NNData[NNData["
                        Mes_DateTime"] == str(Mes_DateTime)])
270                 NNData20 = NNData20.append(NNData.iloc[[counter]])
271             elif (int(Mes_DateTime.strftime("%H")) == 21):
```

```
272                 #NNData21 = NNData21.append(NNData[NNData["
                        Mes_DateTime"] == str(Mes_DateTime)])
273                 NNData21 = NNData21.append(NNData.iloc[[counter]])
274             elif (int(Mes_DateTime.strftime("%H")) == 22):
275                 #NNData22 = NNData22.append(NNData[NNData["
                        Mes_DateTime"] == str(Mes_DateTime)])
276                 NNData22 = NNData22.append(NNData.iloc[[counter]])
277             elif (int(Mes_DateTime.strftime("%H")) == 23):
278                 #NNData23 = NNData23.append(NNData[NNData["
                        Mes_DateTime"] == str(Mes_DateTime)])
279                 NNData23 = NNData23.append(NNData.iloc[[counter]])
280
281     NNDataFinal = pd.DataFrame()
282
283     NNDataFinal = NNDataFinal.append(NNData00)
284     NNDataFinal = NNDataFinal.append(NNData01)
285     NNDataFinal = NNDataFinal.append(NNData02)
286     NNDataFinal = NNDataFinal.append(NNData03)
287     NNDataFinal = NNDataFinal.append(NNData04)
288     NNDataFinal = NNDataFinal.append(NNData05)
289     NNDataFinal = NNDataFinal.append(NNData06)
290     NNDataFinal = NNDataFinal.append(NNData07)
291     NNDataFinal = NNDataFinal.append(NNData08)
292     NNDataFinal = NNDataFinal.append(NNData09)
293     NNDataFinal = NNDataFinal.append(NNData10)
294     NNDataFinal = NNDataFinal.append(NNData11)
295     NNDataFinal = NNDataFinal.append(NNData12)
296     NNDataFinal = NNDataFinal.append(NNData13)
297     NNDataFinal = NNDataFinal.append(NNData14)
298     NNDataFinal = NNDataFinal.append(NNData15)
299     NNDataFinal = NNDataFinal.append(NNData16)
300     NNDataFinal = NNDataFinal.append(NNData17)
301     NNDataFinal = NNDataFinal.append(NNData18)
302     NNDataFinal = NNDataFinal.append(NNData19)
303     NNDataFinal = NNDataFinal.append(NNData20)
304     NNDataFinal = NNDataFinal.append(NNData21)
305     NNDataFinal = NNDataFinal.append(NNData22)
306     NNDataFinal = NNDataFinal.append(NNData23)
```

```
307
308 NNDataFinal.to_csv(outputFileName + fileExtension, sep =
        dataSeparator)
309
310 print "#######################################"
311 print "Ucna mnozica uspesno ustvarjena v datoteki " +
        outputFileName + fileExtension + " (" + str(vnosov) + "
        vnosov)"
312 print "#######################################"
```

Programska koda za modeliranje

```
# Denis Kotnik
# Prilagodljivo kratkorocno napovedovanje lokalnih vremenskih parametrov
# Mentor: doc. dr. Matjaz Kukar
# Fakulteta za racunalnistvo in informatiko
# September 2014; v1.4


# Nastavi trenutni delovni direktorij
setwd("C:/")

# Kolikokrat ponovimo ucenje in testiranje oz. izberemo drgacno testno ucno mnozico
repeats = 5

# Uvozimo knjizico neuralnet
# http://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf
library(neuralnet)
parameters = c("station","TrenutnaTemperaturaZrakaCezN","DatumMeritveVetraCezN","DatumTrenutneMeritveVetra", "NapHitrostVetra", "HitrostVetraCezN", "TrenutnaHitrostVetra", "hoursAhead","TrenutnaTemperaturaZraka","NapTempZraka","NapTempPovrsja","TrenutniPritisk","NapZracniPritisk")

# Preberemo vse datoteke za vse CVP
data_all = list()
for (stationID in 1:90){
  for (hour in 1:11){
    file_input = paste("learningDataTemp_",stationID,"_",hour,".csv",sep="")

    if(file.exists(file_input)){
      data <- read.csv(file_input, head=TRUE, sep="|", dec=".", na.strings="-")
      hoursAhead = hour
      station = stationID
```

```
29        data <- cbind(data,hoursAhead)
30        data <- cbind(data,station)
31        data = data[complete.cases(data[, parameters]), parameters
            ]
32        data_all <- rbind(data_all,data)
33      }
34    }
35  }
36
37  # Odstranimo vse meritve z napovedmi vecjimi od 40 m/s (
      outlayerje)
38  data_all <- data_all[ which(data_all$NapHitrostVetra < 40), ]
39
40  # Error = regressor => napaka => razlika med napovedano
      hitrostjo vetra za cez n ur in dejansko izmerjeno hitrostjo
      vetra cez n ur
41  error = as.numeric(data_all$NapHitrostVetra) - as.numeric(data_
      all$HitrostVetraCezN)
42  # Dodaj error v data_all
43  data_all = cbind(data_all, error)
44
45  data_all["NapZracniPritisk"] = data_all["NapZracniPritisk"] /
      100
46
47  ######################################
48  # VIZUALIZACIJA ANALIZE PODATKOV
49  #pairs(~NapHitrostVetra+TrenutnaHitrostVetra+
      TrenutnaTemperaturaZraka+TrenutniPritisk+NapTempZraka+
      NapZracniPritisk+NapTemperaturaTal,data=data_all, main="
      Simple Scatterplot Matrix")
50
51  #bin<-hexbin(as.Date(data_all$DatumTrenutneMeritveVetra,format
      ="%Y-%m-%d"), data_all$error, xbins=50)
52
53  #plot(bin, main="Hexagonal Binning")
54  #plot(as.Date(data_all$DatumTrenutneMeritveVetra,format="%Y-%m-%
      d"), data_all$error)
55
```

```
#ggplot( data = data_all , aes( as.Date(data_all$DatumNapovedi,
    format="%Y-%m-%d"), data_all$error)) + geom_line()
#####################################

# Vektor ur (00:00 - 23:00)
hours = sprintf("%02d:00",0:23)

# Formula za ucenje nevronske mreze
formula = "as.vector( error ) ~ as.vector( NapHitrostVetra ) +
    as.vector( TrenutnaHitrostVetra ) + as.vector(
    TrenutnaTemperaturaZraka ) + as.vector( NapTempZraka ) + as.
    vector( NapTempPovrsja )"

# NORMALIZACIJA
maxError = max(data_all$error)
minError = min(data_all$error)
minNapHitrostVetra = min(data_all$NapHitrostVetra)
maxNapHitrostVetra = max(data_all$NapHitrostVetra)
minHitrostVetraCezN = min(data_all$HitrostVetraCezN)
maxHitrostVetraCezN = max(data_all$HitrostVetraCezN)

data_all$error = (data_all$error - minError) / (maxError -
    minError)
data_all$NapHitrostVetra = (data_all$NapHitrostVetra - min(data_
    all$NapHitrostVetra)) / (max(data_all$NapHitrostVetra) - min(
    data_all$NapHitrostVetra))
data_all$HitrostVetraCezN = (data_all$HitrostVetraCezN  - min(
    data_all$HitrostVetraCezN )) / (max(data_all$HitrostVetraCezN
     ) - min(data_all$HitrostVetraCezN ))
data_all$TrenutnaHitrostVetra = (data_all$TrenutnaHitrostVetra -
     min(data_all$TrenutnaHitrostVetra)) / (max(data_all$
    TrenutnaHitrostVetra) - min(data_all$TrenutnaHitrostVetra))
#data_all$hoursAhead = (data_all$hoursAhead - min(data_all$
    hoursAhead)) / (max(data_all$hoursAhead) - min(data_all$
    hoursAhead))
data_all$TrenutniPritisk = (data_all$TrenutniPritisk - min(data_
    all$TrenutniPritisk)) / (max(data_all$TrenutniPritisk) - min(
    data_all$TrenutniPritisk))
```

```
data_all$TrenutnaTemperaturaZraka = (data_all$
    TrenutnaTemperaturaZraka - min(data_all$
    TrenutnaTemperaturaZraka)) / (max(data_all$
    TrenutnaTemperaturaZraka) - min(data_all$
    TrenutnaTemperaturaZraka))
data_all$NapTempZraka = (data_all$NapTempZraka  - min(data_all$
    NapTempZraka )) / (max(data_all$NapTempZraka) - min(data_all$
    NapTempZraka ))
data_all$NapZracniPritisk = (data_all$NapZracniPritisk - min(
    data_all$NapZracniPritisk)) / (max(data_all$NapZracniPritisk)
     - min(data_all$NapZracniPritisk))
data_all$NapTempPovrsja = (data_all$NapTempPovrsja - min(data_
    all$NapTempPovrsja)) / (max(data_all$NapTempPovrsja) - min(
    data_all$NapTempPovrsja))


#data2009 <- data_all[ which(format(as.Date(data_all$
    DatumMeritveVetraCezN),format="%Y") == "2009"), ]
#data2010 <- data_all[ which(format(as.Date(data_all$
    DatumMeritveVetraCezN),format="%Y") == "2010"), ]
#data2011 <- data_all[ which(format(as.Date(data_all$
    DatumMeritveVetraCezN),format="%Y") == "2011"), ]
data2012 <- data_all[ which(format(as.Date(data_all$
    DatumMeritveVetraCezN),format="%Y") == "2012"), ]
data2013 <- data_all[ which(format(as.Date(data_all$
    DatumMeritveVetraCezN),format="%Y") == "2013"), ]
dataVsaLeta = rbind(data2009, data2010,data2011, data2012,
    data2013)
data2014 <- data_all[ which(format(as.Date(data_all$
    DatumMeritveVetraCezN),format="%Y") == "2014"), ]

# Ustvarjanje dataFrame oz. podatkovnega okvirja z vektorji
rmseVector = numeric(11)
rmseNNVector = numeric(11)
rmseLNVector = numeric(11)
differenceVector = numeric(11)
graph_results_data = data.frame(rmseVector, rmseNNVector,
    differenceVector, rmseLNVector)
```

```
99
100 # Tabela za rezultate vseh ponovitev za vsako uro (uporabljena
    za prikaz variance ponovitev za doloceno uro)
101 array_rmse <- array(dim=c(11,repeats))
102 array_rmsenn <- array(dim=c(11,repeats))
103 array_rmselr <- array(dim=c(11,repeats))
104
105 print("Pricenjam...")
106
107 for (r in 1:repeats) {
108
109   for (y in 1:11)
110   {
111     ucnaMnozica = data.frame()
112     testnaMnozica = data.frame()
113
114     #data = subset(data_all, hoursAhead == y)
115     data_ucna = subset(dataVsaLeta, hoursAhead == y)
116     data_testna = subset(data2014, hoursAhead == y)
117
118     # 100 % primerov za v ucno mnozico
119     proportion = 1
120     numberOfCases = round(nrow(data_ucna) * proportion)
121     selected = sample(c(1:nrow(data_ucna)), numberOfCases)
122     ucnaMnozica <- data_ucna[selected, ]
123
124     numberOfCases = round(nrow(data_testna) * proportion)
125     selected = sample(c(1:nrow(data_testna)), numberOfCases)
126
127     testnaMnozica <- data_testna[selected, ]
128
129     # Parametra, uporabljena v enacbi RMSE
130     HitrostVetraCezN = testnaMnozica[,"HitrostVetraCezN"]
131     NapHitrostVetra = testnaMnozica[,"NapHitrostVetra"]
132
133     # V testni mnozici ne sme biti parametra HitrostVetraCezN
134     testnaMnozica = subset(testnaMnozica, select=c("
        NapHitrostVetra","TrenutnaHitrostVetra","
```

```
        TrenutnaTemperaturaZraka","NapTempZraka","NapTempPovrsja"
        ))


    # Ucenje nevronske mreze
    print(paste("Pred izdelavo NN_","Repeat: ",r," Ura: ",y,"_",
        nrow(ucnaMnozica)))
    neuralNetwork <- neuralnet(formula, as.data.frame(
        ucnaMnozica), threshold=0.5, hidden = 4, lifesign="full",
         act.fct="logistic", linear.output=FALSE, algorithm="
        rprop+")

    print("Po izdelavi NN")

    # LINEARNA REGRESIJA NAD TESTNIMI PODATKI
    fit <- lm(error ~ NapHitrostVetra + TrenutnaHitrostVetra +
        TrenutnaTemperaturaZraka + NapTempZraka + NapTempPovrsja,
         data=ucnaMnozica)
    errorLR = predict(fit, testnaMnozica)

    rmse = 0
    rmseNN = 0
    rmseLR = 0

    # Nauceno nevronsko mrezo uporabimo na testnih podatkih ter
        pridobimo napako oz. popravek
    print("Pred uporabo: ")
    results <- compute(neuralNetwork, as.data.frame(
        testnaMnozica))
    errorNN = results$net.result

    # DENORMALIZACIJA
    errorNN2 = (errorNN * (maxError - minError)) + minError
    errorLR2 = (errorLR * (maxError - minError)) + minError
    NapHitrostVetra2 = (NapHitrostVetra * (maxNapHitrostVetra -
        minNapHitrostVetra)) + minNapHitrostVetra
    HitrostVetraCezN2 = (HitrostVetraCezN * (maxHitrostVetraCezN
         - minHitrostVetraCezN)) + minHitrostVetraCezN
```

```
    # RMSE izracunan brez popravka
    rmse <- sqrt(sum((NapHitrostVetra2 - HitrostVetraCezN2)^2) /
        length(HitrostVetraCezN2))


    # RMSE, ki uposteva popravek NN
    NapHitrostVetraNN2 = NapHitrostVetra2 - errorNN2
    rmseNN <- sqrt(sum((NapHitrostVetraNN2 - HitrostVetraCezN2)
        ^2) / length(HitrostVetraCezN2))

    # RMSE, ki uposteva popravek LR
    NapHitrostVetraLR = NapHitrostVetra2 - errorLR2
    rmseLR <- sqrt(sum((NapHitrostVetraLR - HitrostVetraCezN2)
        ^2) / length(HitrostVetraCezN2))

    # Razlika med nepopravljenim rmse in popravljenim rmse
    difference = rmse - minRmseNN

    # Rezultate dodamo v dataFrame oz. podatkovni okvir
    graph_results_data$hoursVector[y] = y
    graph_results_data$rmseVector[y] = graph_results_data$
        rmseVector[y] + rmse
    graph_results_data$rmseNNVector[y] = graph_results_data$
        rmseNNVector[y] + rmseNN
    graph_results_data$rmseLNVector[y] = graph_results_data$
        rmseLNVector[y] + rmseLR
    graph_results_data$differenceVector[y] = graph_results_data$
        differenceVector[y] + difference

    # Rezultate vseh ponovitev shranjujemo v spodnji tabeli
    array_rmse[y,][r] = rmse
    array_rmsenn[y,][r] = minRmseNN
    array_rmselr[y,][r] = rmseLR
  }
  print(paste("Ponovitev #N:------", r))
}
```

```
graph_results_data$rmseVector = graph_results_data$rmseVector / repeats
graph_results_data$rmseNNVector = graph_results_data$rmseNNVector / repeats
graph_results_data$rmseLNVector = graph_results_data$rmseLNVector / repeats
graph_results_data$differenceVector = graph_results_data$differenceVector / repeats

print(paste("Generalni popravljeni RMSE: ", sum(graph_results_data$rmseNNVector) / 11))

CU.up <- apply(array_rmse[, 1:repeats], 1, max)
CU.dn <- apply(array_rmse[, 1:repeats], 1, min)
CUNN.up <- apply(array_rmsenn[, 1:repeats], 1, max)
CUNN.dn <- apply(array_rmsenn[, 1:repeats], 1, min)

################################### Izris grafa
y_range <- ceiling(range(0, max(graph_results_data$rmseVector))[2]) + 1
x_range <- 11
plot(graph_results_data$rmseNNVector,col="blue",ylim=c(0,5),axes=FALSE,ann=FALSE, lwd=2, pch=15, cex=1.2)
lines(graph_results_data$rmseVector, type="o", pch=17, lty=0, col="orange",cex=1.2)
lines(graph_results_data$rmseLNVector, type="o", pch=16, lty=0, col="red",cex=1.2)
grid(0, NULL, col = "lightgray", lty = "dotted", lwd = 2, equilogs = TRUE)
#arrows(1:11,CU.dn,1:11,CU.up,code=3,length=0.15,angle=90,col='red')
arrows(1:11,CUNN.dn,1:11,CUNN.up,code=3,length=0.15,angle=90,col='blue', lty=2)
#arrows(1,sum(graph_results_data$rmseNNVector) / 1,1,sum(graph_results_data$rmseNNVector) / 1, code=2)
#abline(h=sum(graph_results_data$rmseNNVector) / 11, col="purple")
```

```
214 axis(1,at=1:x_range, lab=graph_results_data$hoursVector[1:x_
    range],cex.axis=1.4)
215 axis(2,at=0:5,cex.axis=1.4)
216 box()
217 #lines(mean(graph_results_data$rmseNNVector), type="o", pch=22,
    lty=2, col="red")
218 title(xlab="Napovedne ure", ylab="RMSE [m/s]", col.lab=rgb
    (0.1,0.4,0.6), cex.lab=1.3)
219 legend("topright", y_range, c("RMSE nepopravljen","RMSE
    popravljen z NN","RMSE popravljen z LR"), cex=1.3, col=c("
    orange","blue","red"), pch=c(17,15,16), lty=c(0,0,0))
220 ######################################
```

Literatura

[1] Muntase Abdul and Wahed Salman. *Adaptive learning rate versus resilient backpropagation for numeral recognition*, 2008. Dostopno na: `http://www.iasj.net/iasj?func=fulltext&aId=15445`.

[2] Dan Carr. *Package hexbin*, avgust 2014. Dokumentacija dostopna na: `http://cran.r-project.org/web/packages/hexbin/hexbin.pdf`.

[3] Chen Chien-Sheng. *Resilient Back-propagation Neural Network for Approximation 2-D GDOP*, 2010. Dokumentacija originalnega avtorja algoritma dostopna na: `http://www.iaeng.org/publication/IMECS2010/IMECS2010_pp900-904.pdf`.

[4] Sara Colja. Algoritmi in tehnike podatkovnega rudarjena na bazi procesnih parametrov. Diplomsko delo, Univerza v Mariboru, Fakulteta za naravoslovje in matematiko, 2011.

[5] Janez Demšar. Python za programerje. Dostopno na: `http://trac.lecad.si/vaje/raw-attachment/wiki/python/pythonzaprogramerje.pdf`, 2008.

[6] European Committee for Standardization. Road weather information systems. In *Winter maintenance equipment*. European Committee for Standardization, Bruselj (Belgija), 3 edition, 2009.

[7] Stefan Fritsch and Frauke Guenther. *Package neuralnet*, julij 2014. Dokumentacija dostopna na: `http://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf`.

[8] Pavel Golik, Patrick Doetsch, and Hermann Ney. Cross-entropy vs. squared error training: a theoretical and experimental comparison. Raziskava, Human Language Technology and Pattern Recognition, Computer Science Department, RWTH Aachen University, 52056 Aachen, Germany; Spoken Language Processing Group, LIMSI CNRS, Paris, France. Dostopno na: `http://www-i6.informatik.rwth-aachen.de/publications/download/861/GolikPavelDoetschPatrickNeyHermann--Cross-Entropyvs.SquaredErrorTrainingaTheoreticalExperimentalComparison--2013.pdf`.

[9] T. Haiden, A. Kann, G. Pistotnik, K. Stadlbacher, and C. Wittmann. Integrated nowcasting through comprehensive analysis (INCA), Januar 2010. Dostopno na: `http://www.zamg.ac.at/fix/INCA_system.pdf`.

[10] Greg Hamerly and Tom Mitchell. Intro. to machine learning (csi 5325) lecture 9: neural networks. Zapiski; dostopni na: `http://cs.ecs.baylor.edu/~hamerly/courses/5325_11s/lectures/lecture_09.pdf`.

[11] Simon Haykin. *Neural networks: A comprehensive foundation*. Pearson Education, 2 edition, 1999.

[12] Jerman Jure. Uporaba visoko zmogljivih računalnikov za potrebe numeričnega modeliranja vremena v slovenski meteorološki službi. *Vetrnica*, 2012. Dostopno na: `http://www.meteo-drustvo.si/data/upload/Vetrnica0412_Pod_drobnogledom.pdf`.

[13] Oygur Kisi and Erdal Uncuoglu. *Comparison of three back-propagation training algorithms for two case studies*, 2005. Dostopno na: `http://nopr.niscair.res.in/bitstream/123456789/8460/1/IJEMS%2012(5)%20434-442.pdf`.

[14] Marko Koblar. Programski jezik python. *Moj mikro*, 2013. Dostopno na: `http://www.mojmikro.si/v_praksi/mojster/programski_jezik_python`.

[15] Polona Kolar. Tehnologija reinženiringa zimskega vzdrževanja cest. Diplomsko delo, Univerza v Mariboru, Fakulteta za gradbeništvo, marec 2009.

[16] Marko Korošec. Enovit nadzor nad vremenskim stanjem na slovenskih avtocestah - cestno vremenski informacijski sistem. *Slovenski kongres o cestah in prometu*, page 3, oktober 2006. Dostopno na: `http://www.drc.si/Portals/1/Referati/T4-Korosec.pdf`.

[17] Rok Kršmanc. Napovedovanje meteorološkega stanja vozišča iz preteklih podatkov in vremenskih napovedi. Doktorska disertacija, Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, 2013.

[18] Matjaž Kukar and Igor Kononenko. *Maschine learning and data mining: Introduction to Principes and Algorithms*. Horwood Publishing, 2007.

[19] Manfred Kurtz. *Synoptic Meteorology*. Deutscher Wetterdienst, 2 edition, 1998.

[20] Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus-Robert Muller. Efficient backprop, 1998. Dostopno na: `http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf`.

[21] Sibi P., Jones S.Allwyn, and Siddarth P. Analysis of different activation functions using back propagation neural networks. Raziskava, SASTRA University, Kumbakonam, India, 2005-2013. Dostopno na: `http://www.jatit.org/volumes/Vol47No3/61Vol47No3.pdf`.

[22] Zdravko Petkovšek and Andrej Hočevar. *Meteorologija - osnove in nekatere aplikacije*. Partizanska knjiga, 1988.

[23] Neva Pristov and Jure Cedilnik. ALADIN in RC LACE – predstavitev, nastanek in kratka zgodovina mednarodnega sodelovanja. *Vetrnica*, 2012. Dostopno na: `http://www.meteo-drustvo.si/data/upload/Vetrnica0412_Pod_drobnogledom.pdf`.

[24] Neva Pristov, Jure Cedilnik, Jurij Jerman, and Benedikt Strajnar. Priprava numerične meteorološke napovedi ALADIN-SI. *Vetrnica*, 2012. Dostopno na: `http://www.meteo-drustvo.si/data/upload/Vetrnica0412_Pod_drobnogledom.pdf`.

[25] Jože Rakovec and Tomaž Vrhovec. *Osnove meteorologije za naravoslovce in tehnike*. DMFA, 2007.

[26] Martin Riedmiller. *Rprop - Description and Implementation Details*, 1994. Dokumentacija originalnega avtorja algoritma dostopna na: `http://www.inf.fu-berlin.de/lehre/WS06/Mustererkennung/Paper/rprop.pdf`.

[27] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS*, pages 586–591, 1993.

[28] Marko Robnik-Sikonja and Petr Savicky. *Package CORElearn*, julij 2014. Dokumentacija dostopna na: `http://cran.r-project.org/web/packages/CORElearn/CORElearn.pdf`.

[29] Hungarian Meteorological Service. ALADIN model. Dostopno na: `http://owww.met.hu/en/hmshp.php?almenu_id=homepages&pid=numprog&pri=3&mpx=0`.

[30] David Smith. R is hot. *Revolution Analytics*, pages 1–2, 2010.

[31] Oddelek za klimatologijo Urad za meteorologijo. Slovenski vremenski rekordi. Technical report, Agencija Republike Slovenije za okolje,

2010. Dostopno na: http://www.arso.gov.si/vreme/podnebje/slo_vremenski_rekordi.pdf.

[32] Sarle Warren S, 1997-2002. Dostopno na: ftp://ftp.sas.com/pub/neural/FAQ.html ali na http://www.faqs.org/faqs/ai-faq/neural-nets/part1/preamble.html.

[33] Alenka Šajn Slak, Rok Kršmanc, and Janko Merše. INCA-CE - projekt, ki povezuje meteorološke službe osrednje evrope s končnimi uporabniki. *Vetrnica*, 2012. Dostopno na: http://www.cgsplus.si/Portals/1/Raziskovalno%20razvojna%20dejavnost/clanki/INCA-Vetrnica.pdf.

[34] Študenti. Umetne nevronske mreže. Učno gradivo predmeta inteligentni sistemi, Univerza v Ljubljani, Fakulteta za elektrotehniko, 2010. Dostopno na: http://studentski.net/gradiva/ulj/fel/el1/inteligentni-sistemi.html?r=ulj_fel_el1_ins_sno_nevronske_mreze_02.pdf.

[35] Nedjeljka Žagar. Uvod v numerično napovedovanje vremena. Prezentacija, Univerza v Ljubljani, Fakulteta za matematiko in fiziko, januar 2014. Dostopno na: http://www.fmf.uni-lj.si/~zagarn/downloads/M2013/nwp_osnove_2013.pdf.