Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Denis Kotnik

Podatkovni tokovi in rezervoarsko vzorčenje pri napovedovanju proizvodnje sončnih elektrarn

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Matjaž Kukar

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

"You are old when you are not ready for changes."

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
CRISP-DM	Cross Industry Standard	industrijski standard procesa
ALADIN	Aire Limitée, Adaptation Dynamique, Développement International	omejeno območje — dinamična adaptacija, mednarodno sodelovanje
OVE	renewable energy sources	obnovljivi viri energije
ARMA model	autoregresive moving average model	avtoregresijski model drsečih sredin
ARIMA model	autoregresive integrated moving average model	avtoregresijski integrirani model drsečih sredin
ARIMAX model	autoregresive integrated moving average model with explanatory variable	avtoregresijski integrirani model drsečih sredin z neodvisnimi spremenljivkami

Povzetek

Naslov: Podatkovni tokovi in rezervoarsko vzorčenje pri napovedovanju proizvodnje sončnih elektrarn

Sončna, vetrna in vodna energija kot obnovljivi viri energije vzbujajo vedno večjo pozornost, saj je njihov vpliv na okolje, v primerjavi s fosilnimi gorivi, mnogo manjši. V elektroenergetskih sistemih učinkovito shranjevanje električne energije skoraj ni mogoče, zato so se distributerji primorani ukvarjati s problemom ohranjanja ravnovesja med porabo in proizvodnjo električne energije. Kvalitetno napovedovanje proizvodnje in porabe električne energije omenjeni problem močno olajša. Delo obravnava kratkoročno napovedovanje proizvodnje električne energije sončnih elektrarn na območju primorske Slovenije na podlagi vremenskih napovedi, pri čemer se podatke obravnava kot *podatkovni tok*. Za napovedovanje se uporablja in primerja klasične algoritme strojnega učenja in algoritme, ki se iz podatkov učijo *sproti*. Dopolni se jih z algoritmom *ADWIN*, ki s *prilagodljivimi drsečimi okni* in zaznavanjem *sprememb koncepta* vzdržuje *vzorec* zadnjih primerov. Uporablja se tudi algoritem *rezervoarskega vzorčenja z eksponentnim staranjem elementov*, s katerim se vzdržuje vzorec iz celotnega podatkovnega toka. S sprotnimi modeli naučenimi na vzorcu podatkovnega toka so bile pridobljene uporabne napovedi in povsem primerljivi rezultati s sorodnimi deli.

Ključne besede

podatkovno rudarjenje, podatkovni tokovi, sprememba koncepta, rezervoarsko vzorčenje, sončne elektrarne, napovedovanje proizvodnje

Abstract

Title: Data streams and reservoir sampling for predicting production of solar power plants

In the electrical power systems the efficient storage of electricity is almost impossible, therefore the electrical distributors are forced to deal with the problem of maintaining a balance between consumption and production of electricity. Quality forecasts of electricity production and consumption make this problem easier. This thesis deals with the short-term forecasting of electricity production from solar power plants for the Primorska region in Slovenia, whereby data is treated as a *data stream*. Attributes used for this predictions are usually obtained from weather forecast model. Classical machine learning algorithms as well as algorithms that are capable of *online/incremental* learning are being used for forecasting power production and mutually comparison. Machine learning algorithms are being upgraded with *ADWIN* algorithm, which detects *concept drifts* and maintains a sample of the last examples using *adaptive size sliding window*. A *reservoir sampling algorithm with exponential decay* of older elements is also being used to maintain a sample from the entire data stream. Useful predictions with a performance comparable to other results have been obtained with online algorithms learned on the sample of the data stream.

Keywords

data mining, data streams, concept drift, reservoir sampling, solar power plants, production forecasting

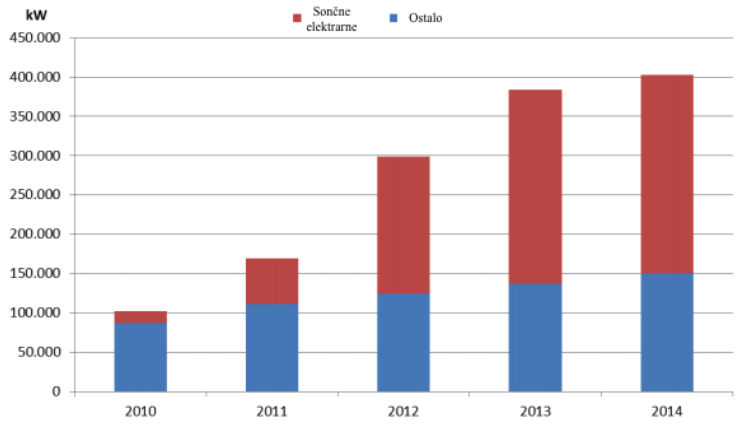
Poglavje 1

Uvod

Slovenija je v zadnjih letih doživela velik razmah v proizvodnji električne energije iz obnovljivih virov energije (OVE) in samostojnih enot za pridobivanje toplotne in električne energije (SPTE). Leta 2009 je zaživela nova podporna shema za tovrstno proizvodnjo. Od takrat je bilo v slovensko distribucijsko omrežje priklopljenih več kot 3.500 novih proizvodnih naprav s skupno instalirano močjo blizu 400 MW. Od vseh naprav, ki so bile do leta 2015 v podporni shemi, se jih je skoraj polovico predstavljalo kot sončne elektrarne (če gledamo po nazivni — priključni moči). Po številu je ta delež takrat dosegal skoraj 90 % [1].

Sončna, vetrna in vodna energija kot obnovljivi viri energije vzbujajo vedno večjo pozornost, saj je njihov vpliv na okolje, v primerjavi s fosilnimi gorivi, mnogo manjši. Ambiciozni načrti glede nadaljnjega zmanjševanja emisij CO_2 nakazujejo, da se bo vključevanje obnovljivih virov še nadalje spodbujalo. Pretvorba sončne v električno energijo je mogoča s fotonapetostnimi moduli. Ker cene slednjih strmo padajo, je sončna energija trenutno eden izmed najbolj uporabljenih obnovljivih virov energije [2]. Na trgu z električno energijo ter pri sistemskih operaterjih se je tako močno povečala pozornost problematiki napovedovanja proizvodnje oddane električne energije. Cena obnovljivih virov je zaradi težke integracije v obstoječe električno omrežje še vedno visoka, predvsem kadar gre za velik delež proizvodnje iz časovno oz.

vremensko pogojenih virov, kot so sončno sevanje, veter, plimovanje, morski valovi itd. Proizvajalci električne energije so v teh primerih pogosto primorani v spremembo obstoječe ali celo gradnjo nove infrastrukture ter razvoj in širjenje novih sistemov in režimov za upravljanje z električnimi omrežji [3].



Slika 1.1: Naraščanje moči naprav OVE in SPTE v podporni shemi v obdobju 2010-2014 [1].

Da bi OVE uspešno integrirali v elektroenergetski sistem, je pomembna natančna napoved proizvodnje oddane moči. Slednja je v veliki meri odvisna od napovedi meteoroloških spremenljivk in je ključna za stabilnost omrežja, torej za ravnotežje oddane in prejete moči v sistemu. Ker se meteorološki parametri lahko v zelo kratkem času spremenijo (npr. intenzivnost vetra, oblačnost in s tem količina sončnega sevanja), je zelo pomembno kratkoročno napovedovanje proizvodnje [4].

1.1 Cilji in motivacija

Distributerji so na prostem trgu električne energije posredniki med elektrarnami, od katerih električno energijo kupujejo, in odjemalci (njihovimi strankami), katerim jo prodajajo. V elektroenergetskih sistemih učinkovito shranjevanje električne energije skoraj ni mogoče — proizvedena električna

energija mora biti v vsakem trenutku enaka porabljeni — zato so se ponudniki električne energije primorane ukvarjati s problemom ohranjanja ravnovesja med porabo in proizvodnjo električne energije. Z natančnejšemi napovedmi porabe in proizvodnje električne energije se lahko zmanjša riziko preobremenitve sistema, kjer lahko pride do trajnih okvar in poškodb v omrežju. Če preobremenitev v električnem omrežju odjemalcem povzroči škodo, lahko tožijo elektro distributerje, katerim to predstavlja (velik) finančni problem. Elektro distributerji lahko z dobro napovedano proizvodnjo prilagajajo svoja vzdrževalna dela tako, da z odklopom določenega območja ene ali več transformatorskih postaj, povzročijo čim manjšo izgubo lastnikom sončnih elektrarn [5].

Uspešnost napovedovanja predstavlja pomemben faktor pri operaterjih trga z električno energijo, še posebej z vidika vzdrževanja ravnovesja med oddano in prejeto močjo. Oddana moč je moč, ki jo oddajo oz. prodajo odjemalcem, prejeta moč pa je moč, katero prejmejo oz. odkupijo običajno od elektrarn. Elektrarne želijo čim hitreje vedeti, koliko energije morajo proizvesti, saj se lahko na napovedano spremembo deloma pripravijo. Zaradi tega ponujajo električno energijo z nižjo ceno, če se jo zakupi vnaprej - hitreje kot se jo zakupi, cenejša je. Distributerji pa želijo zato kupiti energijo čim hitreje in tako ceneje.

Tako kot na vsakem prostem trgu, se tudi na trgu električne energije cena določa na podlagi ponudbe in povpraševanja akterjev na tržišču. Ker točne napovedi porabe omogočajo distributerjem obratovanje z nižjimi stroški, je to področje pomembno tudi z ekonomskega vidika [6]. Kvalitetno napovedovanje proizvodnje in porabe električne energije omenjeni problem olajša, kar doprinese k nižjim stroškom predvsem za distributerje (ki napovedi koristijo za regulacijo napetosti v omrežju in določitev cen), posledično pa tudi za odjemalce električne energije. Največji vpliv na tržne razmere na slovenskem trgu za dan vnaprej se kažejo pri obratovanju sončnih elektrarn (Vir: www.borzen.si).

Citirano [1]:
"Ključen vpliv je seveda čutiti na kratkoročnem trgu (trgu za dan vnaprej in znotraj dne), kjer vremenske razmere in posledično proizvodnja iz obnovljivih virov energije predstavljajo temeljni dejavnik pri oblikovanju cen. Zaradi tega vpliva prihaja do velikih cenovnih razponov med posameznimi dnevi in celo urami, na določenih trgih pa prihaja tudi do pojava negativnih cen. Vpliv porasta deleža proizvodnje iz obnovljivih virov energije je seveda čutiti tudi na dolgoročnem trgu, saj gre za sistemsko spremembo in ne kratkotrajne vplive. Tako je bila leta 2012 povprečna pasovna cena na trgu za dan vnaprej BSP še 53,15 EUR/MWh, leta 2014 pa komaj 40,43 EUR/MWh".

Proizvodnja električne energije je odvisna od mnogih vremenskih dejavnikov, kot so difuzna svetloba, megla, prehodne oblačnosti, koncentracija prahu v zraku, umazanija na panelih, senčenja, temperatura, sončni kot, okvare ... zato elektrarne težko napovedo proizvodnjo za dalj časa vnaprej. Napovedi se običajno delijo na dolgoročne (celoletne), srednjeročne (časovno obdobje meseca) ter kratkoročne (za dan vnaprej in znotraj dne). V magistrski nalogi se lotevamo problema kratkoročnega napovedovanja proizvodnje električne energije na podlagi podatkov o vremenskih napovedi [1].

1.2 Pregled področja

V zadnjem času se uveljavlja področje raziskovanja podatkovnega rudarjenja tudi na podlagi podatkovnih tokov. Glavni motiv so čedalje večje količine v realnem času zajetih podatkov, kot so podatki iz merilnikov, vremenski podatki, lokacijski (GPS) podatki, internetni promet, e-pošta, telefonski pogovori, podatki s pametnih naprav, kliki uporabnikov na spletnih straneh itd.

V literaturi je mogoče zaslediti veliko metod in načinov za napovedovanje tako o proizvodnji kot o porabi električne energije. Najpogostejši pristopi pri napovedovanju se nanašajo na uporabo že znanih algoritmov strojnega učenja, predvsem umetnih nevronskih mrež in časovnih vrst.

Literature, kjer podatke za problem napovedovanja proizvodnje ali porabe električne energije obravnavajo kot podatkovni tok, smo zasledili le malo. Zato smo v nadaljevanju predstavili nekaj raziskav v Sloveniji in tujini. V njih napovedujejo s klasičnimi metodami strojnega učenja, bodisi z metodami primernimi za delo s podatkovnimi tokovi. Poudariti želimo, da je zaradi razlik v člankih, kot so časovna obdobja (kdaj se je napovedovalo), dolžine napovednega intervala, lokacije elektrarn, kriterijskih funkcij za merjenje uspešnosti napovedi itd. medsebojna primerjava rezultatov raziskav težka. Posledično smo primerjave v poglavju 4 predstavili predvsem opisno.

1.2.1 Raziskave napovedovanja električne energije iz podatkovnih tokov v Sloveniji

V [7] in kasneje v doktorski disertaciji z Inštituta Jožef Stefan [8] so avtorji predstavili regresijska drevesa za sprotno učenje iz spremenljivih podatkovnih tokov. Avtorji navajajo, da je njihov pristop prvi, ki v celoti obravnava tematiko regresijskih algoritmov, ki se lahko iz spremenljivih podatkovnih tokov učijo sproti.

V okviru del s Fakultete za računalništvo in informatiko na Univerzi v Ljubljani [9, 10] se avtorja Jan Kraljič in Jaka Demšar ukvarjata s problemi podatkovnih tokov in spremembami koncepta. Poleg tega se lotijo napovedovanja porabe električne energije z vidika klasifikacije na podlagi podatkovnih tokov. Napovedovali so za naslednji dan na dnevni in urni ravni iz podatkov za leto 2008. Preizkušali so različne velikosti podatkovnih oken in ugotovili, da so najboljši rezultati pri velikosti oken 100 dni in diskretizaciji atributov v 4 razrede. Z umetnimi nevronskimi mrežami so dobili med 5 % in 10 %, z modeli časovnih vrst (ARIMA) pa 3 % relativno napako.

V delu, ki je nastalo na isti fakulteti [5], se avtor Tomaž Tomažič ukvarja predvsem z analizo podatkov in napovedovanjem porabe električne energije sončnih elektrarn s klasičnimi regresijskimi algoritmi strojnega učenja. Iz vremenskih napovedi so napovedali proizvodnjo z normalizirano povprečno absolutno napako (nMAE) 0,035.

V okviru Fakultete za elektrotehniko na Univerzi v Ljubljani je bilo objavljeno delo [4], v katerem so avtorji poskušali z umetnimi nevronskimi mrežami napovedovati 24-urno proizvodnjo 25 sončnih elektrarn. Pridobili so rezultate s povprečno absolutno procentno napako (MAPE) 20,1 % za pretežno oblačen dan oziroma MAPE 2,1 % za pretežno jasen dan. V okviru iste fakultete je nastalo tudi delo [11], za katerega je avtor v letu 2016 prejel Prešernovo nagrado Univerze v Ljubljani. Mehke Takagi-Suge modele so naučili na podatkih o porabi električne energije in meritvah (ne napovedih) vremenskih parametrov iz leta 2010 in 2011 ter z njimi napovedovali porabo v letu 2012. Pridobili so rezultate z MAPE 3,66 %.

V članku [6] so se avtorji osredotočili na verjetnostno napovedovanje porabe z uporabo robustnega regresijskega modela, ki upošteva pretekli potek uporabe električne energije in vremenske podatke. Podatke so pred uporabo filtrirali in z avtokorelacijo analizirali vpliv časa na porabo, metode pa testirali za podatke območja Slovenije. Rezultati kažejo, da so napovedi slabše v zimskem in poletnem obdobju, medtem ko so napovedi najboljše v spomladanskem in jesenskem času. Posledica slabših napovedi je lastnost porabe v teh mesecih, saj je negotovost porabe večja.

V raziskavi [12] primerjajo uspešnost napovedovanja urne proizvodnje električne energije sončnih elektrarn, ki temeljijo na 3 in 5 dnevnih vremenskih napovedih modela ALADIN (ARSO). Produkti modela za obdobje 3 dni vsebujejo urne napovedi o sončnem sevanju, produkti za obdobje 5 dni pa dnevne napovedi o največji in najmanjši temperaturi zraka ter deležu oblačnosti. Sklepa, kateri podatki so bolj pomembni za napovedovanje proizvodnje, avtorji v članku niso zaključili. Ugotovili so, da so urne napovedi sončnega sevanja bolj relevantne v obdobjih z malo padavinami, kot je bilo pričakovano. V obdobjih z veliko padavinami je bila 5 dnevna napoved natančnejša za kumulativno dnevno napovedovanje energije.

Na konferenci Dnevi slovenske informatike [13] in delu Fakultete za matematiko in fiziko na Univerzi v Ljubljani [14] gre zaslediti, da se na področju Slovenije za te namene v praksi uporabljajo predvsem klasične metode

strojnega učenja, kot so *multivariatna linearna regresija*[1], umetne nevronske mreže, metode podpornih vektorjev, metode glavnih komponent in časovne vrste.

1.2.2 Raziskave napovedovanja električne energije iz podatkovnih tokov v svetu

Na področju raziskovanja odkrivanja znanja iz podatkovnih tokov so zelo dejavni raziskovalci z Univerze v Portu na Portugalskem, kjer so med drugim predstavili zelo hitra odločitvena drevesa (angl. *The Very Fast Decision Tree Algorithm*) [15]. Omeniti velja raziskovalce João Gama (Univerza v Portu, Portugalska), Albert Bifet (Univerza v Waikatu, Nova Zelandija) in Ricard Gavaldà (Univerza v Kataloniji, Španija).

V [16], objavljenem leta 2002, je prikazana uporaba več nivojskih umetnih nevronskih mrež za napovedovanje porabe električne energije. Rezultati so v večini testov dosegali relativno napako do 5 % .

V [17] je poudarek na testiranju različnih kriterijskih funkcij z umetnimi nevronskimi mrežami z namenom napovedovanja proizvodnje vetrne energije za 72 h vnaprej iz meritev o hitrosti in smeri vetra. Poleg tega avtorji primerjajo klasični oz. *paketni* in *sprotni* način učenja umetnih nevronskih mrež. Rezultati kažejo, da je sprotni način bolj natančen in primeren za sprotno prilagajanje modelov novim podatkom.

V [18] avtorji okvirno predstavijo arhitekturo lastnega podpornega sistema za pomoč pri odločanju pri trgovanju z električno energijo iz obnovljivih virov, kot so sončne, vodne in vetrne elektrarne. Njihov sistem uporablja inkrementalne umetne nevronske mreže, ki se odločajo na podlagi velikih količin podatkov (*angl. Big Data*) iz senzorjev v omenjenih elektrarnah.

Avtorji se v člankih [19, 20] ukvarjajo z *inkrementalnimi* algoritmi za delo s podatkovnimi tokovi, vendar ne z namenom napovedovanja proizvodnje oz. porabe električne energije. Osredotočijo se na razvrščanje (angl.

[1]Multivariatna linearna regresija je tehnika modeliranja naključnih vektorjev.

clustering) senzorjev (virov podatkov), odločitvene sisteme in sisteme za zaznavanje sprememb v podatkih.

1.3 Metodologija in prispevek naloge

V magistrski nalogi smo najprej predstavili koncept podatkovnih tokov in pristopa strojnega učenja na podatkovnih tokovih s *paketnimi* (angl. *batch*) in sprotnimi (angl. *online*) algoritmi. Zatem smo predstavili *spremembo konceptov* in algoritem ADWIN. Opisali smo tudi pogosto uporabljeno tehniko vzorčenja imenovano *rezervoarsko vzorčenje*.

Eksperimentalna evaluacija je potekala po metodologiji CRISP-DM, ki je splošni industrijski standardni proces za podatkovno rudarjenje. Z opisanimi metodami smo se nato osredotočili na napovedovanje proizvodnje električne energije na 304-ih sončnih elektrarnah na območju primorske Slovenije. Napovedovali smo na podlagi vremenskih napovedi, pridobljenih iz elektrarni najbližje modelske točke modela *ALADIN*[2], in nekaj izpeljanih atributov. V primerjavi z drugimi avtorji smo se omenjenega problema lotili z vidika podatkovnih tokov.

Tabela 1.1: Prikaz različnih načinov modeliranja.

	Paketni	Sprotni
Brez vzorčenja	✓	✓
ADWIN	✓	✓
Rezervoarsko vzorčenje	✓	✓

Proizvodnjo smo modelirali na šest različnih načinov, ki so prikazani v tabeli 1.1. V *paketnem* načinu brez vzorčenja smo učno množico inkrementalno povečevali, katero smo potem na dnevni ravni uporabljali za učenje klasičnih algoritmov strojnega učenja. V *sprotnem* načinu brez vzorčenja smo uporabljali algoritme, ki privzeto podpirajo sprotno (angl. *online*) učenje. Oba

[2]ALADIN je numerični mezo-meteorološki model za napovedovanje vremena. Okvirno smo ga opisali v razdelku 3.3.

načina smo ekskluzivno dopolnili z dvema metodama vzorčenja, s katerima smo zmanjšali količino podatkov, potrebnih za računanje napovednih modelov. Kot prvo uporabljamo algoritem ADWIN, ki s *prilagodljivimi drsečimi oknI* in zaznavanjem sprememb koncepta vzdržuje vzorec zadnjih primerov podatkovnega toka. Kot drugo pa uporabljamo algoritem *rezervoarskega vzorčenja z eksponentno funkcijo*, ki vzdržuje vzorec primerov iz celotnega časovnega obdobja podatkovnega toka.

Poglavje 2

Uporabljene metode in orodja

Vse več je virov, ki podatke generirajo neprestano in z visoko hitrostjo. Tem virom pravimo podatkovni tokovi. Za obdelavo, shranjevanje in odkrivanje zakonitosti iz podatkovnih tokov mnogokrat ne moremo uporabiti klasičnih metod, ki se uporabljajo za statične množice podatkov. Poleg tega imajo podatkovni tokovi lastnosti, da se jim porazdelitev podatkov s časom lahko spreminja in da so lahko potencialno neskončni.

V tem poglavju opišemo metode, ki so primere za delo s podatkovnimi tokovi. Najprej predstavimo pristope in algoritme strojnega učenja ter *spremembo koncepta*, s katerim definiramo omenjeno spreminjanje podatkov. Na koncu opišemo metode *vzorčenja*, s katerimi poskušamo v naslednjem poglavju pridobiti vzorec, ki ima lastnosti potencialno neskončnega podatkovnega toka.

2.1 Podatkovni tokovi

Podatkovni tok je urejeno zaporedje podatkov, ki je lahko neskončno dolgo. Podatki prihajajo neprekinjeno, v realnem času in potencialno z zelo visoko hitrostjo, zaradi česar jih je kot celotno množico težko shraniti v delovni oz. bralno-pisalni pomnilnik (kateri je v primerjavi z velikostjo podatkovnega toka lahko zelo majhen) in obdelovati v realnem času. Porazdelitev podat-

kov v podatkovnem toku se lahko sčasoma spremeni, na zaporedje teh pa ni mogoče vplivati [15]. Primeri podatkovnih tokov so zajem podatkov iz merilnikov, vremenski podatki, lokacijski (GPS) podatki, internetni promet, e-pošta, telefonski pogovori, podatki s pametnih naprav, kliki uporabnikov na spletnih straneh itd.

2.2 Pristopi strojnega učenja na podatkovnih tokovih

Strojno učenje lahko delimo na različne načine [21]:

1. Učenje na podlagi človeškega nadzora ali ne:
   (a) nadzorovano (angl. *supervised*),
   (b) delno nadzorovano (angl. *semisupervised*),
   (c) nenadzorovano (angl. *unsupervised*) in
   (d) spodbujevalno (angl. *reinforcement*) učenje.
2. Napovedovanje diskretnih (klasifikacija ali uvrščanje) ali zveznih (regresija) spremenljivk.
3. Sposobnost učenja algoritmov iz posameznih učnih primerov ali večje množice:
   (a) sprotno in
   (b) paketno učenje.
4. Učenje algoritmov na način, da nov podatek preprosto primerjajo z obstoječimi podatki ali na način, da namesto tega iz obstoječih podatkov izluščijo neko zakonitost oz. naučen model:
   (a) na podlagi primerov (angl. *instance-based*) in
   (b) na podlagi modelov (angl. *model-based*).

Omenili bi, da smo v literaturi zasledili tudi druge (tuje) izraze, ki spadajo v 3. zgornjo alinejo oz. delitev. Za *paketni* oz. *klasični* pristop so to *batch*, *offline* ali *epoch* učenje. Za pristop, kjer se algoritem sproti uči iz posameznih primerov, pa smo zasledili izraze, kot so *incremental*, *online*, *out-of-core* in *sequential learning*. V tem delu bomo uporabljali izraza *paketno* in *sprotno* učenje.

Obstajata dva splošna pristopa za uporabo konceptov strojnega učenja na velikih množicah podatkov oz. podatkovnih tokovih [22]. Prvi pristop teži k maksimalni *izrabi* obstoječih klasičnih metod, drugi pa se ukvarja z iskanjem novih metod, ki so posebej prilagojene velikim množicam podatkov. Pristopa sta okvirno opisana v nadaljevanju.

2.2.1 Paketni pristop

Algoritmi strojnega učenja so se v preteklosti osredotočali na odkrivanje znanj iz majhnega števila primerov, saj je bila na voljo le omejena količina podatkov. Nekateri zelo kompleksni algoritmi so kljub majhnim učnim množicah vračali zelo natančne rezultate. Običajno se domneva, da se celotna množica podatkov shrani v delovni pomnilnik, od koder so algoritmu na voljo. V tuji literaturi se za takšen pristop učenja največkrat uporablja terminologija *epoch* ali *batch learning*, generalno pa ga lahko opišemo s psevdokodo 1. V algoritmu, ki ga opisuje omenjena psevdokoda, ponavljamo koraka 3 in 4 toliko časa, dokler ni zadoščeno nekemu pogoju. Primeri pogojev so lahko:

- napaka neke kriterijske funkcije na učni množici podatkov postane majhna,
- napaka neke kriterijske funkcije na validacijski množici prične naraščati (angl. *early stopping*),
- število korakov preseže neko fiksno število ...

V zadnjem desetletju je potreba po obdelavi večjih količin podatkov motivirala področje podatkovnega rudarjenja z raziskovanjem načinom za

Algoritem 1 Psevdokoda klasičnega učenja

```
1: inicializiraj model
2: repeat
3:     obdelaj vse učne primere
4:     posodobi model
5: until ni zadoščeno nekemu pogoju
```

zmanjšanje časa računanja in porabe pomnilnika, potrebnega za obdelavo velikih množic podatkov. Če je podatkov za pomnilnik preveč, se običajno uporabi *vzorčenje* (podpoglavje 2.7.1). Druga možnost je, da se algoritem zateče k začasnemu shranjevanju na zunanji pomnilni medij. Bistveni cilj pri vsem tem je ustvariti, ali se čim bolj približati učnemu procesu, ki je linearen številu primerov. Ta proces je le nadgradnja klasičnega strojnega učenja — učenje šteje za eno samo, morda zelo drago operacijo, katere rezultat je končen statičen model [22]. Primere je potrebno nekako izbrati in združiti v *serijo*[1], katero se lahko uporabi za učenje klasičnih algoritmov strojnega učenja. Pri napovedovanju novih primerov je zato potrebno izbrati ali združiti prave modele. Slabost tega pristopa je težavna izbira velikosti serije oz. učne množice podatkov. Pri izbiri prevelike serije se proces učenja (kompleksnejših) modelov močno upočasni in zato bi celoten proces obdelave podatkovnega toka v realnem času zastal. Pri izbiri premajhne pa bi lahko dobili model ali serijo modelov z veliko pristranskostjo[2] ali veliko varianco[3], kateri bi pri novih primerih najverjetneje vračal slabe rezultate.

Uporabo klasičnih algoritmov strojnega učenja na podatkovnih tokovih v nekaterih virih poimenujejo *pristop z ovojnico* (anlg. *wrapper approach*) [22]. Cilj pristopa je maksimalna izraba klasičnih algoritmov za namene modeliranja podatkovnih tokov. Primeri uporabe pristopa z ovojnico je moč zaslediti v številnih člankih [24, 25, 26].

[1]Serija je skupina stvari, dogodkov, podatkov itd. s skupnimi lastnostmi, značilnosti, ki si sledijo v določenem časovnem zaporedju.

[2]Pristranskost v strojnem učenju (angl. *bias*) je napaka hipoteze, ki izvira iz učnega algoritma [23].

[3]Varianca v strojnem učenju je napaka hipoteze, ki izvira iz učnih podatkov [23].

Način pristopa z ovojnico lahko obdela večje količine podatkov, vendar še vedno ne reši problema *neprekinjene dobave* novih podatkov — podatkovnega toka. Modela, ki je bil predhodno naučen, z novimi podatki običajno ni mogoče posodobiti. Namesto tega je potrebno celoten postopek učenja ponoviti z vključenimi novimi primeri.

2.2.2 Sprotni pristop

Algoritmi za delo s podatkovnimi tokovi so se pojavili kot odgovor na problem *neprekinjene dobave novih podatkov*. Namenjeni so obdelavi podatkov v realnem času, katerih velikosti so lahko velikokrat večje kot je velikost pomnilnika.

Osnovne predpostavke oz. lastnosti metod za delo s podatkovnimi tokovi so [27]:

1. Algoritem lahko obdeluje primer za primerom. Ne potrebuje končne (učne) množice podatkov, da bi pridobil novo uporabno znanje.

2. Vsak primer naj bi bil obdelan le enkrat.

3. Podatkovne strukture, ki jih uporablja algoritem, naj bi potrebovale fiksno količino prostora.

4. Algoritem lahko kadarkoli uporabimo za napovedovanje oz. lahko kadarkoli poda svoje najnovejše znanje, katero naj bi bilo najboljše.

Algoritmi, ki so zgornjim predpostavkam prilagojeni, ali pa so taki že *po svoji naravi* (npr. naivni Bayesov klasifikator), so še posebej primerni za podatkovne tokove oz. velike množice podatkov. Običajno jim pravimo sprotni ali *inkrementalni* algoritmi.

Sprotno učenje ima več prednosti pred paketnim. Potencialno je bolj robustno, saj se napake ali manjkajoči podatki učne množice popravljajo sproti med samim učenjem. Če so med učenjem sprotnega modela s starimi podatki

Algoritem 2 Psevdokoda sprotnega učenja

```
1: inicializiraj model
2: repeat
3:     obdelaj en učni primer
4:     posodobi model
5: until je vsak učni primer obdelan (vsaj) enkrat
```

naenkrat na voljo novejši podatki, lahko učenje nemudoma prekinemo in nadaljujemo z novejšimi. Še ena zelo pomembna prednost sprotnega učenja je sprotno prilagajanje novim trendom in porazdelitvam v podatkih [28].

Uspešnost in hitrost konvergence učenja paketnih in sprotnih algoritmov (še posebej umetnih nevronskih mrež [29] in v nadaljevanju opisanega gradientnega sestopa) je močno povečana, če so vhodi v te skalirani[4] na neko območje ali standardizirani[5]. Omenjene transformacije podatkov se pri klasičnem strojnem učenju (angl. *feature scaling*) običajno izvede v fazi predobdelave, saj je za to potreben eden ali več statističnih kazalcev, kot so npr. standardni odklon, povprečje, največja ter najmanjša vrednost (npr. pri skaliranju z uporabo enačbe (3.3)) ... neke množice podatkov. Pri delu s spremenljivimi podatkovnimi tokovi so takšne sprotne transformacije podatkov problematične, saj težko natančno določimo razpon vrednosti spremenljivk, porazdelitev slednjih pa se lahko s časom spreminja. S temi problemi se ukvarjajo v [30, 31, 32, 33, 34]. Kako smo se mi lotili omenjenega problema, smo opisali v poglavju 3.5.

[4]Skaliranje v strojnem učenju (angl. *min-max scaling*) je postopek preslikave neke množice podatkov na neko območje, običajno na $[0, 1]$ ali $[-1, 1]$. Namesto *skaliranja* se pogosto uporablja izraz *normalizacija*.

[5]Standardizacija spremenljivk je postopek, s katerim vrednosti spremenljivke transformiramo in sicer tako, da od vsake vrednosti spremenljivke odštejemo aritmetično sredino μ in delimo z njenim standardnim odklonom σ. Nova standardizirana spremenljivka ima tako aritmetično sredino $\mu = 0$ in standardni odklon $\sigma = 1$.

Hoeffdingova drevesa

Učenje odločitvenih dreves iz podatkovnih tokov je eden težjih problemov, s katerimi se sooča skupnost v domeni podatkovnega rudarjenja [15]. Običajna odločitvena in regresijska drevesa se niso sposobna učiti sproti, zato za uporabo na podatkovnih tokovih niso primerna. Za te namene so avtorji v članku [35] predstavili *Hoeffdingova drevesa*. Ta izkoriščajo dejstvo, da je relativno malo primerov potrebnih za statistično utemeljeno odločitev za deljenje lista (širitvi drevesa z odločitvenim vozliščem). Ime so dobili po *Hoeffdingovi meji*, ki je kriterij za delitev. Določena je z enačbo (2.1). Meja zagotavlja, da se z verjetnostjo $1 - \delta$ pravo povprečje naključne spremenljivke z intervalom (zalogo vrednosti) R po n primerih ne bo razlikovalo od ocenjenega povprečja za več kot ε. Torej, če je razlika kriterijskih funkcij (npr. *indeks Gini*) atributov A_i in A_j večja od ε, lahko z verjetnostjo $1 - \delta$ trdimo, da je A_i atribut primeren za deljenje.

$$\varepsilon = \sqrt{\frac{R^2 \; ln(1/\delta)}{2n}} \tag{2.1}$$

Najbolj znan in trenutno najsodobnejši algoritem za gradnjo Hoeffdingovih dreves je *VFDT* (*Very Fast Decision Tree Algorithm*). Originalna različica VFDT algoritma je namenjena klasifikaciji primerov iz (statičnih) podatkovnih tokov, v katerih se porazdelitev podatkov s časom ne spreminja. V [36] predstavijo algoritem *CVFDT*, ki je primeren za klasifikacijo na podlagi podatkovnih tokov, v katerih se porazdelitev lahko spreminja. Obstaja še več različic oz. nadgradenj algoritma VFDT, npr. *VFDTc* [37], ki osnovno različico algoritma nadgradi s podporo zveznim atributom (pri čemer še vedno uporablja diskretizacijo) in močnejšimi tehnikami klasifikacije v listih dreves.

Omenjeni algoritmi so v svoji osnovni različici namenjeni problemom klasifikacije na (spremenljivih) podatkovnih tokovih. Poudariti želimo, da se v večjem deležu literature, katero smo obravnavali za namene tega dela, torej v povezavi s strojnim učenjem na podatkovnih tokovih, ukvarjajo s problemi uvrščanja oz. klasifikacije. Strokovnih in znanstvenih člankov ter knjig, v

katerih se ukvarjajo z regresijo na podatkovnih tokovih, smo našli relativno malo. Med temi bi omenili članek [7] in doktorsko disertacijo [8], kjer med prvimi predstavijo oz. v celoti obravnavajo regresijski problem in sprotno učenje iz spremenljivih podatkovnih tokov.

2.3 Ocenjevanje uspešnosti učenja na podatkovnih tokovih

Pri uporabi strojnega učenja nas zanima, kako uspešno bi izvajalni algoritem z zgrajeno teorijo oz. neko hipotezo reševal nove primere. Za te namene se običajno uporablja kritična spremenljivka *uspešnost* ali njej enakovredna obratna *napaka*. Če imamo regresijski problem, želimo vedeti, kako natančne bodo napovedane vrednosti *odvisne spremenljivke*[6] oz. s kakšnim zaupanjem lahko verjamemo vrednostim, ki nam jih model vrne. V ta namen se pogosto uporabljajo mere uspešnosti, kot so *povprečna absolutna napaka* (MAE; enačba (3.4)), *povprečna kvadratna napaka* (MSE), *povprečna napaka* (MBE) in druge, opisane v poglavju 3.5.

V strojnem učenju običajno podatke razdelimo na dve množici. Na *učno* za učenje in *testno množico* za testiranje uspešnosti algoritma. Pri paketnem učenju je takšen postopek deljenja podatkov deloma odvisen od velikosti množice podatkov. Majhne množice podatkov, ki vsebujejo nekaj tisoč primerov, so primerne za postopke, kot so večkratno učenje enega algoritma, prečna preverjanja ... , ki podatke večkrat delijo na učno in testno množico, jih izrabijo maksimalno oz. iz podatkov izluščijo največ informacij. Z večanjem velikosti množice podatkov zaradi praktične omejitve časa in prostora (v računalniku se pri paketnem učenju podatki običajno hranijo v delovnem pomnilniku) ti postopki postanejo neuporabni. Manjšanje ponovitev učenja, iteracij, števila podmnožic pri prečnem preverjanju ... so zato običajna praksa pri paketnem učenju na velikih množicah podatkov.

[6]Odvisna spremenljivka ali *kriterij* je spremenljivka, katero napovedujemo. Običajno se jo označuje kot y.

Algoritmi za delo s podatkovnimi tokovi imajo glede ocenjevanja točnosti in deljenja podatkov na učno in testno množico drugačne zahteve kot klasični paketni algoritmi. Namesto da se soočajo s problemom, kako iz podatkov izluščiti največ informacij, upoštevajo dejstvo, da se podatki lahko skozi čas spremenijo - *sprememba koncepta*. Ukvarjajo se tudi s problemom, kako izmeriti uspešnost algoritma skozi čas. Za grafično predstavitev slednjega se najpogosteje uporablja graf uspešnosti/napak skozi čas/število primerov (angl. *learning curve*). Na osi x se prikaže čas/število primerov, na osi y pa neka kriterijska funkcija oz. uspešnost. Sprejeta praksa je, da se tak graf uporablja za okvirno primerjavo različnih algoritmov za delo na podatkovnih tokovih; običajno se primerja uspešnost v določenem času, hitrost naraščanja uspešnosti v začetku učenja in vztrajnost uspešnosti skozi čas (da ostane enaka ali narašča).

Ocenjevanje učenja na podatkovnih tokovih je relativno novo in še zdaleč ne tako dobro raziskano področje kot pri klasičnem strojnem učenju. Dobro znana in najpogosteje uporabljena postopka delitve podatkov in evalvacije algoritmov [15, 22], ki pripomoreta k reševanju zgoraj omenjenih problemov, sta opisana v nadaljevanju (razdelka 2.3.1 in 2.3.2).

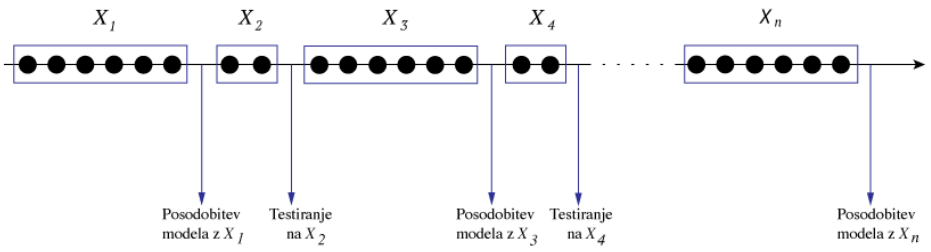
2.3.1 Izločanje neodvisne testne množice

Ko *k-kratno prečno preverjanje* (angl. *k-fold cross validating*) pri klasičnem oz. *paketnem* učenju postane preveč časovno ali prostorsko zahtevno, se namesto izločanja k neodvisnih testnih množic pogosto uporabi izločanje le ene neodvisne testne množice. Metoda je uporabna predvsem za direktno primerjavo različnih študij oz. za primere, ko je razmerje med učno in testno množico vnaprej definirano. Metoda se je tako uveljavila že v *paketnem*, uporablja pa se tudi pri *sprotnem* učenju.

Če želimo v podatkovnem toku oceniti uspešnost algoritma skozi čas, model uporabljamo periodično: nekaj primerov uporabimo kot učno množico za učenje modela in nato naslednje kot testno (angl. *holdout*) množico, na kateri model uporabimo za napovedovanje oz. ocenjevanje uspešnosti. Testne

primere se lahko po uporabi za napovedovanje oz. ocenjevanje uspešnosti uporabi kot učne. Slednje je še posebej učinkovito, ko imamo na voljo malo podatkov.

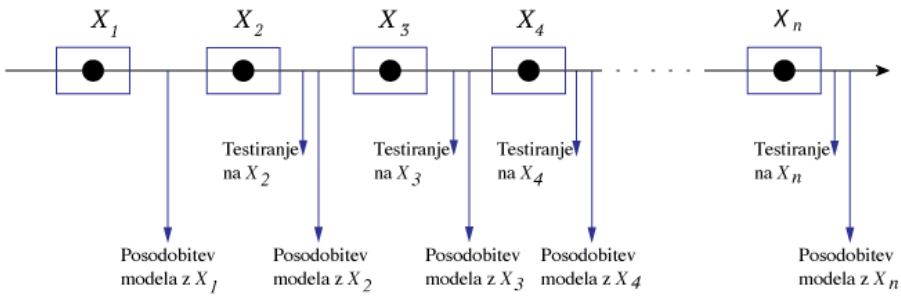
Možen vir te testne množice so lahko tudi najnovejši primeri iz podatkovnega toka, kateri še niso bili uporabljeni za učenje modela. Ta postopek je priporočljiv v scenarijih, ko prihaja do spremembe koncepta, saj tako na nek način izmerimo sposobnost prilagajanja modela najnovejšemu trendu podatkov.



Slika 2.1: Shema postopka izločanja neodvisne testne množice. Primera neodvisnih testnih množic na sliki sta X_2 in X_4.

2.3.2 Prepletanje testiranja in učenja

Alternativni pristop ocenjevanja učenja algoritmov na podatkovnem toku je *prepletanje testiranja in učenja* (angl. *Interleaved Test-Then-Train* ali *Prequential*) [38]. Vsak posamezni primer podatkovnega toka se najprej uporabi za testiranje oz. napovedovanje in ocenjevanje uspešnosti, nato pa za učenje modela. Na ta način je model vedno uporabljen na podatkih, ki jih še nikoli ni *videl*. Naj bo $L(y_i, \hat{y}_i)$ funkcija mere uspešnosti med napovedjo $\hat{y}_i$ in dejansko vrednostjo y_i v podatkovnem toku. Napako prepletanja testiranja in učenja oz. *ocenjeno pričakovano napako* (angl. *prequential error*) v času i opišemo kot vsoto vrednosti funkcij mere uspešnosti do časa i deljeno s številom primerov v podatkovnem toku do časa i. Z drugimi besedami gre



Slika 2.2: Shema postopka prepletanje testiranja in učenja. Potem ko množico $X_2, \ldots, X_n$ uporabimo za testiranje, jo uporabimo za posodobitev modela.

za povprečno napako do časa i. Definiramo jo z enačbo (2.2) [39].

$$P_e(i) = \frac{1}{i} \sum_{k=1}^{i} L(y_k, \hat{y_k}) \tag{2.2}$$

Prednost tega pristopa pred *izločanjem neodvisne testne množice* je večji izkoristek podatkov, ki so nam na voljo za učenje. Prav tako nam ta pristop zagotavlja neprekinjen graf uspešnosti algoritma skozi celoten čas. Dodatna prednost je tudi zmožnost, da lahko izračunano uspešnost, oz. napako uporabimo v realnem času, npr. v našem primeru kot vhod v algoritem ADWIN. Omeniti velja, da za uporabo tega pristopa ni nujno, da poznamo popolnoma vse prave vrednosti y_i v podatkovnem toku — primere z manjkajočimi pravimi vrednostmi y_i lahko izpustimo [15]. Slabost tega pristopa je, da je težko natančno izmeriti čas, ki je potreben za učenje in testiranje algoritma. Poleg tega o pravi uspešnosti algoritma v določenem času (še posebej v začetku učenja) težko sklepamo. Trenutne izboljšave je zaradi trenutnih napak težko zaznati. Tako je uporabnost uspešnosti oz. napak v določenem času vprašljiva, vendar ta učinek sčasoma oz. s številom primerov postane nepomemben [15, 22]. Na uspešnost algoritma vpliva tudi, v nadaljevanju opisana, *sprememba koncepta* v podatkih. Kako ta vpliva na uspešnost naših algoritmov, smo opisali v poglavju 4.

Pristop omogoča posodobitev statistike ocenjevanje učenja z vsakim (novim) primerom iz podatkovnega toka. Če smo omejeni s prostorom, se lahko uporabi rezervoarsko vzorčenje, ali pa se rezultate uspešnosti shranjuje kot pri pristopu izločanja neodvisne testne množice — samo na vsakih nekaj primerov.

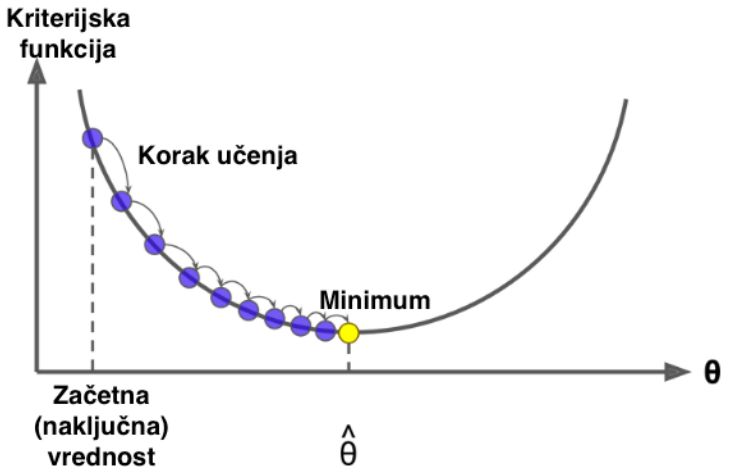
2.4 Stohastični gradientni spust

Glavni namen učenja modelov pri strojnem učenju je poiskati parametre (največkrat so to koeficienti, katere označujemo s θ), ki minimizirajo (ali maksimizirajo) neko *kriterijsko funkcijo*. Te parametre se običajno išče na *analitičen* ali *numerični* način. Prvi način vrne točne rezultate, vendar se ga zaradi računske zahtevnosti ne uporablja pogosto, saj običajno vsebuje *drage* matematične operacije, kot je npr. iskanje inverza (lahko zelo velike) matrike. En tak primer je *normalna enačba* kot analitična rešitev linearne regresije s kriterijsko funkcijo najmanjših kvadratov (enačba (2.3)), kjer je X matrika vrednosti vseh atributov vseh primerov v učni množici — v vsaki vrstici je en učni primer. V omenjeni enačbi y predstavlja vektor vrednosti odvisnih spremenljivk.

$$\theta = (X^T X)^{-1} X^T y \tag{2.3}$$

Druga, precej pogosteje uporabljena možnost, je numerični način pridobitve parametrov oz. uporaba optimizacijskih metod, ki vrnejo približno rešitev. Primeri takih metod so L-BFGS [40], Adam [41], Adadelta [42], gradientni spust (angl. *gradient descent*) itd. V nadaljevanju smo opisali paketni (angl. *batch*) gradientni spust in njegovo *stohastično* različico, ki je za razliko od paketne različice primerna tudi za velike količine podatkov in je najbolj uporabljena optimizacijska metoda v sprotnih algoritmih [43, 44].

Gradientni spust (angl. *gradient descent*) je zelo generična optimizacijska metoda, s katero se lahko najde optimalno rešitev v širokem naboru problemov. Splošna ideja metode je iterativno posodabljanje parametrov



Slika 2.3: Gradientni spust [21].

(θ) z namenom minimiziranja (ali maksimiziranja) neke kriterijske funkcije. Postopek se začne z naključno inicializacijo (angl. *random initialization*) vektorja parametrov θ. Iterativno se nato izračuna lokalni gradient kriterijske funkcije glede na θ, postopek pa se z naslednjo iteracijo nadaljuje v smeri padajočega gradienta. Postopek se običajno ustavi, ko je gradient blizu 0 oz. manjši od neke tolerance ε. Pomemben parameter v metodi gradientnega spusta je korak učenja (angl. *learning rate*, običajno označen z grško črko η), s katerim reguliramo velikost koraka v smeri minimuma. Napačna izbira vrednosti slednjega lahko privede do tega, da metoda ne bo našla dobre rešitve.

Vsaka iteracija paketnega gradientnega spusta vsebuje zahtevno računanje povprečja gradientov kriterijske funkcije $Q(z_i, \theta_t)$ skozi celotno učno množico (število vseh primerov v učni množici je označeno z n, posamezen primer pa z z_i). Za to je potrebna ustrezna in lahko tudi zelo velika količina računalniškega pomnilnika, kar je glavni problem te različice metode.

Algorithm 1: Psevdokoda paketnega gradientnega spusta

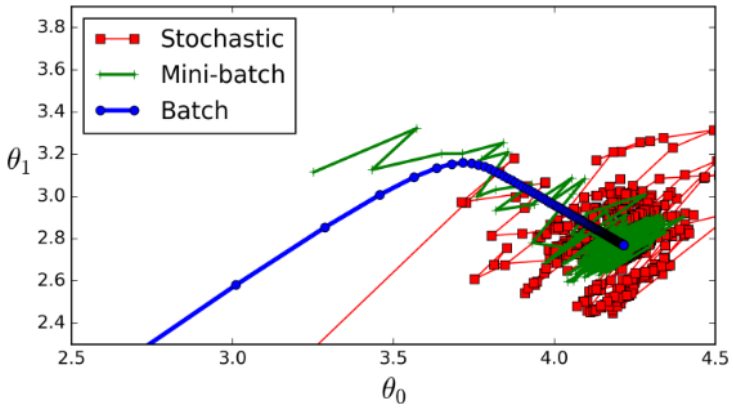
inicializiraj θ na naključne vrednost;
naključno premešaj učno množico;
for $t = 1, \ldots,$ število iteracij **do**
  $\theta_{t+1} = \theta_t - \eta_t \nabla_\theta \hat{J}_L(\theta_t) = \theta_t - \eta_t \frac{1}{n} \Sigma_{i=1}^{n} \nabla_\theta Q(z_i, \theta_t)$
end

Stohastična (imenovana tudi *iterativna* ali sprotna različica gradientnega spusta (angl. kratica *SGD*) je drastična poenostavitev paketne različice. Namesto računanja povprečja gradientov kriterijske funkcije nad celotno učno množico, se v vsaki iteraciji računa gradient kriterijske funkcije enega (lahko naključnega) primera. Poenostavitev metode temelji na upanju, da naključni šum ne bo vplival na povprečno vedenje algoritma. Stohastična različica je tako primerna za uporabo nad novo prihajajočimi podatki oz. podatkovnimi tokovi, tudi takimi, v katerih se porazdelitev podatkov lahko spreminja [45, 46].

Algorithm 2: Psevdokoda stohastičnega gradientnega spusta

inicializiraj θ na naključne vrednost;
for $t = 1, \ldots,$ število iteracij **do**
  naključno premešaj učno množico;
  for $i = 1, \ldots,$ n **do**
    $\theta_{t+1} = \theta_t - \eta_t \nabla_\theta Q(z_i, \theta_t)$
  end
end

Delovanje stohastičnega gradientnega spusta je, za razliko od njegove paketne različice, bolj naključno: namesto da se kriterijska funkcija postopoma zmanjšuje in enakomerno približuje minimumu, bo naključno skakala in se zmanjševala le v povprečju, kar prikazuje slika 2.4. Ko se bo približala minimumu, bo prav tako naključno skakala, kar pa je lahko tudi dobra lastnost — metoda ima tako, v primerjavi s paketno različico, več možnosti, da se izogne lokalnemu minimumu in najde globalnega [21].



Slika 2.4: Poti različic gradientnega spusta [21].

Namesto računanja povprečja gradientov na celotni učni množici (paketni gradientni spust) ali na enem primeru (stohastični gradientni spust), se ta lahko računa na manjši (lahko naključni; vendar ne celotni) množici primerov. Slednja metoda se imenuje *mini-batch* gradientni spust. Glavna prednost te metode pred stohastično različico je, da lahko z uporabo strojne optimizacije (npr. uporabo GPU) precej pospešimo učenje (računanje matričnih operacij) in da omogoča določeno stopnjo paralelizacije [21, 46].

Vse različice gradientnega spusta so zelo občutljive na različne skale vhodnih podatkov, kar pomeni, da se uspešnost in hitrost konvergence lahko precej izboljša z normalizacijo ali standardizacijo atributov.

2.5 Časovne vrste

Časovna vrsta je zaporedje podatkov $x_1, x_2, ..., x_n$, izmerjenih v zaporednih časovnih trenutkih, med katerimi so običajno enaki časovni razmiki. S časovnimi vrstami opazujemo dinamiko pojavov ter razvoj podatkov skozi čas, iščemo zakonitosti v podatkih in predvidevamo o nadaljnjem razvoju dogodkov. Uporabljajo se pri procesiranju signalov, prepoznavanju vzorcev, napovedovanju vremena, napovedovanju cen blaga, finančni matematiki, astronomiji itd. Časovne vrste se najbolje prikaže kot graf, kjer je na abscisni osi

prikazan čas, na ordinatni pa vrednosti časovne vrste [47].

Na podatke v časovnih vrstah vplivajo raznorazni pojavi. Z analizo časovnih vrst skušamo iz časovnih vrst razbrati naslednje vrste sprememb [14]:

- *trend*[7], ki predstavlja dolgoročno obnašanje serije (npr. ali proizvodnja električne energije s časom na dolgi rok upada ali se zvišuje), katerega smer se lahko spreminja in katerega funkcija ni nujno, da je linearna,
- kratkoročne sezonske komponente,
- dolgoročne sezonsko komponento, ki predstavlja pojavljanje v konstantnih obdobjih (npr. proizvodnja sončnih elektrarn je v poletnih mesecih običajno večja kot v zimskih),
- in naključno komponento, ki je stohastična, je ne moremo pojasniti s sistematičnimi gibanji in vsebuje posamične vplive (npr. pri proizvodnji sončnih elektrarn bi se to pokazalo ob sončevem mrku).

Za časovno vrsto ni nujno, da vsebuje vse zgoraj omenjene komponente. Stacionarnost pomeni, da vrednosti preučevane časovne vrste nihajo neodvisno od časa okrog konstantnega povprečja, poleg tega pa mora tudi varianca, ki meri velikost teh nihanj, ostati nespremenjena v času. Za ugotavljanje stacionarnosti časovnih vrst se preučuje tudi avtokorelacijo oz. vrednosti koeficientov avtokorelacije [48]. V praksi so časovne vrste največkrat nestacionarne. Stacionarnost je v korelaciji z dolžino časovne vrste. Stacionarni procesi so največkrat opisani s srednjo vrednostjo, varianco in avtokorelacijsko funkcijo. V kratkem časovnem obdobju so nestacionarni procesi pogostejši kot v daljšem. Glavni lastnosti nestacionarnih časovnih vrst sta trend in sezonski vpliv, katera morata biti identificirana in odstranjena [4].

Podatkovni tok je urejeno zaporedje podatkov, kjer je čas, za razliko od zaporedja podatkov, za analizo lahko tudi nepomemben. Pri časovnih vrstah gre za časovno urejeno zaporedje, v katerem ima vsak podatek svoj časovni žig. Čas je poleg zaporedja podatkov pomemben element pri analizi vrst.

[7]Trend je značilnost pojava glede na spreminjanje v daljšem časovnem obdobju.

2.5.1 Avtoregresijski model

Časovne vrste so običajno sestavljene iz medsebojno odvisnih elementov, katere se opiše s serijo specifičnih elementov iz preteklosti. To se imenuje avtoregresijski proces. *Avtoregresijski model* (angl. *autoregresive model — AR model*) upošteva vrednosti prejšnjih deviacij iste časovne vrste ter naključne napake ε_t. Opišemo ga z enačbo (2.4) [49], kjer je c konstanta, φ_i parameter modela, ε_t naključna napaka v času t, X_t deviacije (nihanja) okoli srednje vrednosti v času t in p red modela oz. stopnja avtoregresijskega procesa [49]. Običajno se uporablja notacija AR(p), kjer p predstavlja stopnjo avtoregresijskega procesa.

$$X_t = c + \Sigma_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t \tag{2.4}$$

2.5.2 Model drsečih sredin

Model drsečih sredin (angl. *moving average model — MA model*) se uporablja za računanje trenutne vrednosti spremenljivke X_t. Drseča sredina oz. povprečje je linearno odvisno od spremenljivke ε_t iz enačbe (2.4). Definiramo ga z enačbo (2.5) [50, 49], kjer je q red modela. Zanj pa se običajno uporablja notacija MA(q).

$$X_t = \mu + \varepsilon_t + \Sigma_{i=1}^{q} \theta_i \varepsilon_{t-i} \tag{2.5}$$

2.5.3 Avtoregresijski model drsečih sredin

Avtoregresijski model drsečih sredin (angl. *autoregresive moving average model — ARMA model*) je sestavljen iz avtoregresijskega modela in modela drsečih sredin. Model opišemo z enačbo (2.6) [50] ali z notacijo ARMA(p, q). Uporablja se ga za stacionarne časovne vrste.

$$X_t = \Sigma_{i=1}^{p} \varphi_i X_{t-i} + \Sigma_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t + c \tag{2.6}$$

2.5.4 Avtoregresijski integrirani model drsečih sredin

Časovne vrste so velikokrat nestacionarne (in z izrazitim sezonskim vzorčenjem), torej ne nihajo okoli nespremenljive srednje vrednosti. Za ta namen se uporablja *avtoregresijski integrirani model drsečih sredin* (angl. *autoregressive integrated moving average model — ARIMA model*), ki predstavlja posplošitev modela ARMA(p, q) s številom transformacij d nad vhodnimi podatki, da ti ustrezajo začetnim predpostavkam za modeliranje. Zanj se običajno uporablja notacija ARIMA(p, d, q). Omenjeni model definiramo z enačbo (2.7) [50], kjer Δ označuje operator za razliko, oz. ΔX_t pomeni razlika med X_t in X_{t-1}. Z d povemo, kolikokrat izvedemo to operacijo. Primer za $d = 2$: $\Delta^2 X_t = \Delta\Delta X_t = (X_t - X_{t-1}) - (X_{t-1} - X_{t-2})$. Za p in q se običajno vzame vrednosti manjši od 2, za d pa vrednosti 0, 1 ali 2.

$$\Delta^d X_t = \Sigma_{i=1}^{p} \varphi_i \Delta^d X_{t-i} + \Sigma_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t + c \tag{2.7}$$

2.5.5 Avtoregresijski integrirani model drsečih sredin z neodvisnimi spremenljivkami

Avtoregresijski integrirani model drsečih sredin z neodvisnimi spremenljivkami (angl. *Autoregressive Integrated Moving Average with Explanatory Variable model — ARIMAX model*) razširja model ARIMA tako, da upošteva še neodvisne spremenljivke. Definira ga enačba (2.8), kjer so $Y_{i,t}$ neodvisne spremenljivke v času t in β_i njim pripadajoči koeficienti [51, 52]. Model se običajno uporablja, ko se sezonska komponenta s časom spreminja.

$$\Delta^d X_t = \Sigma_{i=1}^{p} \varphi_i \Delta^d X_{t-i} + \Sigma_{i=1}^{q} \theta_i \varepsilon_{t-i} + \Sigma_{i=1}^{M} \beta_i Y_{i,t} + \varepsilon_t + c \tag{2.8}$$

Za namene magistrskega dela smo uporabljali ARIMA in ARIMAX modele, ki so dostopni v knjižnici *Pyflux*[8].

[8]Pyflux je odprto kodna programska knjižnica s podporo modeliranju s časovnimi vrstami v programskem jeziku Python. Dostopna je na: `http://www.pyflux.com/`

2.6 Sprememba koncepta

Ena izmed ključnih lastnosti podatkovnih tokov, ki jih loči od statičnih množic podatkov, je možnost, da vključujejo *spremembo koncepta* (angl. *concept drift*) — porazdelitev, ki generira primere v podatkovnem toku, se spremeni. Vir podatkov S_1 s porazdelitvijo Π_{S_1} se nepredvidljivo zamenja z virom S_2, ki ima porazdelitev Π_{S_2}. Ker se domneva, da je sprememba koncepta nepredvidljiva, se občasna sezonskost oz. ponavljanje dogodkov v podatkih običajno ne obravnava kot sprememba koncepta. Osnovna predpostavka pri problemih, ki se ukvarjajo s spremembami koncepta, je negotovost glede prihodnosti - o viru ciljnega primera lahko le domnevamo, ocenjujemo ali predvidevamo, vendar brez kakršne koli gotovosti [53].

Avtorji v članku [54] navajajo tri načine, na katere lahko v podatkih pri strojnem učenju pride do spremembe koncepta skozi čas, kjer je k število razredov pri klasifikacijskem problemu. Spremenijo se lahko:

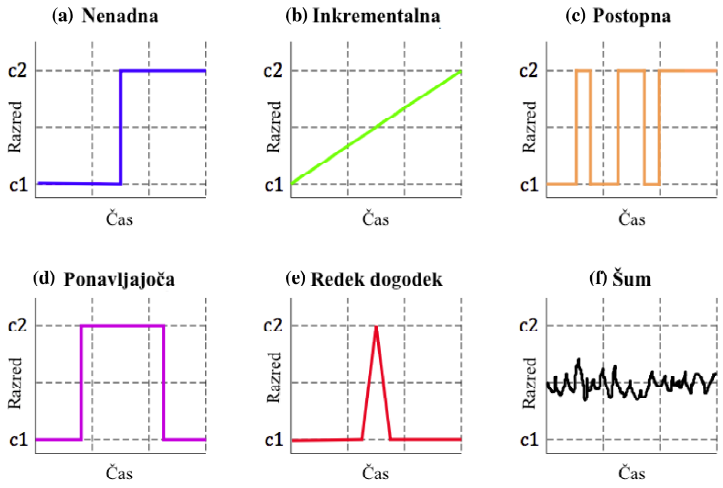
1. apriorne verjetnosti razredov $P(c_1), P(c_2), ..., P(c_k)$ oz. porazdelitev zvezne odvisne spremenljivke y,
2. porazdelitve razredov $P(X|c_i), i = 1, \ldots, k$ oz. porazdelitve zveznih atributov X,
3. aposteriorne porazdelitve vsebovanosti v razredih $P(c_i|X), i = 1, \ldots, k$ oz. relacija med odvisno y in neodvisnimi zveznimi spremenljivkami X.

V prvi točki gre za spremembo vrednosti odvisne spremenljivke, pri drugi pa za spremembo vrednosti neodvisnih spremenljivk, na podlagi katerih je nek primer klasificiran v razred c_i, ali na podlagi katerih je določena vrednost y. Porazdelitev neodvisnih spremenljivk se lahko spremeni celo tako, da sama sprememba nima vpliva na vsebovanost primera v nek razred oz. odvisno spremenljivko y - ta ostane enaka. Temu se v literaturi pravi tudi *virtualna sprememba koncepta* (angl. *virtual concept drift*). V tretji točki gre za spremembo vsebovanosti v nek razred c_i pri vrednostih neodvisnih spremenljivk X oz. za spremembo relacije med odvisno $\hat{y}$ in neodvisnimi spremenljivkami

X. Temu spravimo tudi *prava sprememba koncepta* (angl. *real concept drift*) [55].

Vsaka izmed zgoraj omenjenih sprememb se lahko zgodi na šest različnih načinov, katere prikazuje slika 2.5. Graf (a) na sliki prikazuje nenadno in nepovratno spremembo razreda spremenljivke, (angleško imenovano tudi *concept shift* [15]). Takšne spremembe je najlažje zaznati, primer pri katerem lahko do tega pride, pa je sprememba sezone pri prodaji. Grafa (b) in (c) prikazujeta inkrementalno in postopno spremembo, pri čemer je drugo težje zaznati, saj se primeri iz novega koncepta dalj časa mešajo s primeri iz starega. Primer za inkrementalno spremembo je dvig cen izdelkov zaradi inflacije, primer za postopno pa je počasno, a vztrajno prilagajanje klasifikatorja elektronske pošte oz. *spam* filtra. Ko novo prihajajočo elektronsko pošto ločuje med *uporabniku nekoristno (spam)* in *uporabniku koristno* pošto, pri čemer se interes uporabnika s časom spreminja. Poznamo še ponavljajočo spremembo (graf (d)), za katero je značilno da se stari koncepti neperiodično ponavljajo. V nekateri literaturi jo omenjajo kot *lokalna sprememba koncepta* (angl. *local concept drift*). Graf (e) predstavlja nek dogodek, ki se zgodi redko in nepričakovano ter se ga lahko obravnava kot osamelca (angl. *outlier*). V podatkovnih tokovih se te redke dogodke običajno ignorira, razen če je namen iskanje osamelcev oz. anomalij. Primeri teh dogodkov so izpadi merilnikov, ekstremne/napačne/nesmiselne vrednosti, ki jih vračajo merilniki, pri napovedovanju vremena je to lahko sončev mrk itd. Zadnji graf (f) prikazuje šum oz. naključne spremembe koncepta, katere je potrebno filtrirati oz. odstraniti. Šum se ne sme obravnavati kot sprememba koncepta, ker je zanemarljivo nihanje, ki nima kakršne koli veze s spremembo porazdelitve vira.

Omeniti velja, da zgoraj omenjeni načini spremembe koncepta niso ekskluzivni in se v praksi pogosto prekrivajo oz. mešajo. Ker so podatkovni tokovi lahko potencialno neskončni, je število različnih porazdelitev, ki se pojavijo v viru podatkovnega toka in načinov prehajanja med njimi, lahko neskončno.



Slika 2.5: Tipi sprememb koncepta [55].

Obstaja precej metod za detekcijo spremembe koncepta. Večina jih deluje na principu *opazovanja indikatorjev uspešnosti učnega algoritma* ali s *primerjanjem porazdelitev dveh oken*: referenčnega, ki povzema informacije starih podatkov in aktivnega, ki povzema informacije najnovejših primerov. Za namene tega magistrskega dela smo uporabili algoritem *ADWIN*, ki spada v drugo skupino in je opisan v nadaljevanju. V primerjavi z ostalimi metodami velja se večkrat izkaže za robustnejšega in zanesljivejšega [56, 26]. Omenimo nekaj ostalih metod iz prve od omenjenih skupin: *algoritem CUSUM* [57], ki vrne opozorilo, ko se povprečje vhodnih podatkov (običajno so to napake) signifikantno razlikuje od 0, *Page-Hinkley test* [58], *FLORA* [59], *Early Drift Detection Method* [60] itd. Znan algoritem iz druge skupine je *statistično obvladovanje procesov* (angl. *Statistical Process Control - SPC*) [61].

2.7 Pridobivanje podatkov iz podatkovnih tokov

Za sprotno (oz. rekurzivno) računanje povprečja in standardnega odklona vrednosti v podatkovnem toku se običajno uporabljata formuli (2.9) in (2.10), pri čemer je potrebno v spominu shranjevati le tri vrednosti: število elementov v toku n, vsoto vrednosti $\Sigma_{i=1}^{n} x_i$ in vsoto kvadratov vrednosti $\Sigma_{i=1}^{n} x_i^2$.

$$\bar{x}_1 := x_1; \quad \bar{x}_n = \frac{(n-1)\bar{x}_{n-1} + x_n}{n} \tag{2.9}$$

$$\sigma_n = \sqrt{\frac{1}{n-1}(\Sigma_{i=1}^{n} x_i^2 - \frac{1}{n}(\Sigma_{i=1}^{n} x_i)^2)}; \quad n > 1 \tag{2.10}$$

Zaradi omejene količine delovnega pomnilnika kompleksnejša analiza vseh podatkov v neomejenem (neskončnem) podatkovnem toku pogosto ni mogoča. Ko zaradi tega točni rezultati niso dosegljivi, so sprejemljive tehnike, ki vračajo približne rezultate. Pri delu s podatkovnimi tokovi se uporabljajo metode zmanjševanja količine podatkov in drseča okna [15]. Najpogosteje uporabljene metode zmanjševanja količine podatkov so med drugimi vzorčenje, histogrami in valčna transformacija (angl. *wavelet transformation*).

2.7.1 Vzorčenje

Vzorčenje je pogosto uporabljena praksa oz. postopek izbire neke podmnožice, s katero lahko ocenimo lastnosti celotne množice podatkov. Na podatkovnih tokovih se lahko uporablja za oceno statističnih značilk (npr. standardni odklon in matematično upanje) vrednosti v toku.

Metode vzorčenja lahko drastično zmanjšajo količino podatkov potrebnih za analizo oz. oceno vrednosti in s tem tudi računsko zahtevnost. Vendar so lahko tudi vir napak. Pridobitev *reprezentativnega* vzorca, torej podmnožice podatkov, ki ima približno enake lastnosti celotne množice, je namreč glavna težava. Uporaba vzorčenja na podatkovnih tokovih za namene odkrivanja

napak, sprememb, anomalij, osamelcev itd. je tako lahko problematična, saj ne obravnavamo celotnega toka.

Večina tehnik vzorčenja iz statistike zahteva vnaprej znano velikost množice oz. dolžino podatkovnega toka, zaradi česar jih je za uporabo na podatkovnih tokovih potrebno spremeniti. Ključna problema pri tem sta *velikost vzorca* in *tehnika vzorčenja* [15].

Rezervoarsko vzorčenje

Najenostavnejša vrsta vzorčenja je *naključno vzorčenje*, pri čemer ima vsak element enako verjetnost, da je izbran. Tehnika *rezervoarskega vzorčenja* (angl. *reservoir sampling*) [62] je klasičen algoritem za sprotno naključno vzorčenje in vzdrževanje naključnega izbora. Osnovna ideja je vzdrževanje vzorca velikosti k, ki se imenuje *rezervoar*. Element, ki je v podatkovnem toku n-ti po vrsti, ima verjetnost $\frac{k}{n}$, da je zamenjan z elementom iz rezervoarja. Tako se verjetnost dodajanja novega elementa v rezervoar s časom oz. številom primerov zmanjšuje.

Analizirajmo enostaven primer, kjer je $k = 1$: ko iz podatkovnega toka pride prvi element, je ta v rezervoarju obdržan z verjetnostjo $\frac{1}{1} = 1$. Ko pride drugi element, ga v rezervoar dodamo z verjetnostjo $\frac{1}{2}$. Torej je verjetnost, da v rezervoarju obdržimo prvi element, enaka $(1 - \frac{1}{2}) \times 1 = \frac{1}{2}$. Ko pride tretji element, ga v rezervoar dodamo z verjetnostjo $\frac{1}{3}$. Prvi in drugi element imata pri tem verjetnost $(1-\frac{1}{3}) \times \frac{1}{2} = \frac{2}{3} \times \frac{1}{2} = \frac{1}{3}$, da se ohranita v rezervoarju (pri tem je $\frac{1}{2}$ verjetnost, da je bil element v rezervoarju ohranjen, ko je iz podatkovnega toka prišel drugi element). Verjetnost, da se i-ti element v podatkovnem toku dolžine n ohrani v rezervoarju, je podana z enačbo (2.11).

$$\frac{1}{2} \times \frac{2}{3} \times \cdots \times \frac{i}{i+1} \times \cdots \times \frac{n-2}{n-1} \times \frac{n-1}{n} = \frac{1}{n} \tag{2.11}$$

Časovna zahtevnost za pridobitev naključnega vzorca velikosti k iz podatkovnega toka dolžine n z uporabo algoritma rezervoarskega vzorčenja je $O(n)$ (oziroma $O(1)$ za vsak element), prostorska pa $O(k)$. Če „mečemo kovanec" na vsakih nekaj in ne za vsak element v podatkovnem toku, lahko

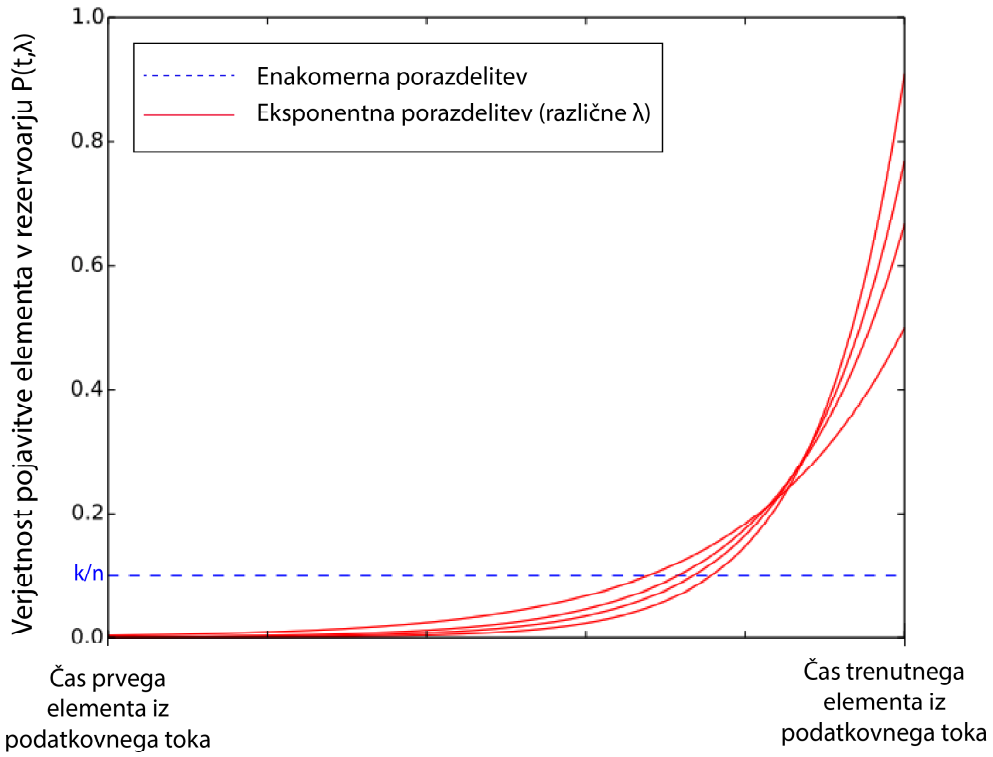
dodatno pohitrimo algoritem. Klasično rezervoarsko vzorčenje je s psevdokodo opisano v algoritmu 3.

Algorithm 3: Rezervoarsko vzorčenje.

```
Vhod: S: podatkovni tok vrednosti,
      k: velikost rezervoarja;
vstavi prvih k elementov v rezervoar;
for v ∈ S do
    naj bo i indeks (število po vrsti) elementa v;
    generiraj naključno celoštevilsko število M ∈ [1, i];
    if M ≤ k then
        vstavi v v rezervoar;
        izbriši naključen element iz rezervoarja;
    end
end
```

Rezervoarsko vzorčenje ohranja vzorec, katerega elementi so enakomerno porazdeljeni skozi zgodovino podatkovnega toka. Če se podatki v toku s časoma spreminjajo, lahko starejši elementi v rezervoarju postajajo manj relevantni. Posledično želimo novejše elemente v rezervoarju ohranjati z večjo verjetnostjo kot starejše. Avtor v članku [63] metodo rezervoarskega vzorčenja nadgradi z eksponentno funkcijo, s katero ohranja vzorec oz. rezervoar, v katerem so elementi skozi zgodovino podatkovnega toka porazdeljeni približno eksponentno — novejši element ima večjo verjetnost pojavitve v rezervoarju kot starejši; starejšim elementom se verjetnost ohranitve v rezervoarju s časoma zmanjšuje, kar prikazuje graf na sliki 2.6.

Naj ima vsak element iz podatkovnega toka svojo časovno značko. Eksponentna funkcija povezana s časom r-tega elementa v rezervoarju, je v času prihoda n-tega elementa ($r \leq n$) iz podatkovnega toka definirana s $f(r, n)$ (enačba (2.12)) oz. povezana z verjetnostjo $P(r, n)$ (enačba (2.13)) [63]. Funkcija $P(r, n)$ je proporcionalna funkciji $f(r, n)$; C je konstanta za normalizacijo. Funkcija $f(r, n)$ z naraščanjem n monotono pada (pri nekem



Slika 2.6: Graf različnih porazdelitvenih funkcij vzorčenja. k označuje velikost rezervoarja, n pa velikost podatkovnega toka.

fiksnem r), z naraščanjem r pa monotono narašča (pri nekem fiksnem n). Slednje omogoča, da imajo novejši elementi podatkovnega toka večjo verjetnost, da bodo vzorčeni. S parametrom λ definiramo hitrost naraščanja funkcije $f(r, n)$ proti novejšim elementom in zavzema vrednost iz območja $[0, 1]$.

$$f(r, n) = e^{-\lambda(n-r)} \tag{2.12}$$

$$P(r, n) = C \cdot f(r, n) \tag{2.13}$$

Kot smo že omenili, je časovna zahtevnost rezervoarskega vzorčenja z enakomerno porazdelitvijo $O(1)$ za vsak element iz podatkovnega toka. V primeru rezervoarskega vzorčenja z eksponentno funkcijo je potrebno elementom v rezervoarju vsakič (ob vsakem novem elementu iz podatkovnega toka) ponovno preračunati porazdelitev (da bi se sprememba odražala v funkciji $f(r, n)$) ne glede na to, ali bo element ohranjen v rezervoarju ali ne. Časovna zahtevnost je posledično $\Omega(k)$ za vsak element iz podatkovnega toka, kjer je k velikost rezervoarja.

V [64] predstavijo dodatne izboljšave in pohitritve rezervoarskega vzorčenja z eksponentno funkcijo, kot so enostavnejše in bolj učinkovito računanje porazdelitve, drugačen in bolj učinkovit način merjenja starosti elementov v rezervoarju itd.

2.7.2 Drseča okna

Večinoma nas bolj kot statistika vseh podatkov zanima statistika le najnovejših podatkov v podatkovnem toku. Za ta namen se uporabljajo *drseča okna fiksne* ali *prilagodljive* velikosti, ki delujejo na podoben princip kot podatkovna struktura vrsta (*FIFO* oz. *first-in first-out*): ko je v okno vstavljen element z indeksom i, je istočasno odstranjen element z indeksom $i - j$, kjer je j velikost okna. Za računanje statistike okna in odstranjevanje starih elementov iz okna je potrebno hraniti vse elemente okna [15].

Slika 2.7: Koncept drsečega okna [15].

ADWIN

ADWIN (angl. *ADaptive sliding WINdow*) [26] je algoritem, ki z uporabo prilagodljivih drsečih oken zaznava spremembe (angl. *drift detection*) in ocenjuje statistično upanje oziroma povprečje v podatkovnem toku bitov ali realnih števil. Algoritem v oknu ohranja le nedavno videne elemente, velikost okna pa se spreminja. Ima lastnost, da je največja dolžina okna statistično skladna s hipotezo, da v povprečni vrednosti elementov znotraj okna ni prišlo do sprememb. Starejši del okna je odstranjen, če in samo če se njegova povprečna vrednost preveč razlikuje od povprečne vrednosti preostalega dela okna. Slednje ima dve posledici [15]:

1. ob omenjenem dogodku algoritem vrne signal, ki pomeni spremembo v podatkovnem toku,

2. algoritem lahko kadar koli vrne povprečno vrednost elementov znotraj okna oz. oceno povprečne vrednosti v podatkovnem toku.

Vhoda v algoritem sta stopnja zaupanja $\delta \in (0, 1)$ in zaporedje realnih števil $x_1, x_2, \ldots, x_t$ (oz. tok podatkov). S stopnjo zaupanja določamo zaupanje v izhod algoritma — večja kot je, manj je sprememb koncepta, katerih algoritem ne zazna, vendar z večanjem stopnje povečujemo delež število lažnih pozitivnih primerov (angl. *false positives - FP*). Vsaka vrednost x_t je na voljo samo v času t in pripada neki porazdelitvi D_t, ki se s časom lahko spremeni. Predpostavlja se, da je x_t v mejah $[0, 1]$, kar se ob predpostavljanju neke zgornje in spodnje meje lahko doseže z normalizacijo.

ADWIN uporablja prilagodljivo drseče okno W, v katerem hrani najnovejše podatke x_i. Ideja algoritma je enostavna: ko je razlika povprečij elementov dveh *dovolj velikih* podoken okna W *dovolj velika*, lahko sklepamo, da so pričakovane vrednosti elementov podoken dovolj različne in starejši del okna oz. starejše podokno odstranimo. Z drugimi besedami: velikost okna W naj bo čim večja, obenem pa skladna z domnevo, da je razlika povprečij katerih koli podoken manjša od ε. V enačbi (2.14) [22] je n velikost okna W, n_o in n_1 velikosti podoken okna W (tako je $n_0 + n_1 = n$), σ_W^2 pa varianca vseh elementov v oknu W. Simbol ln označuje naravni logaritem, simbol $\cup$ pa vsa različna deljenja okna W na dve podokni W_0 in W_1.

$$\begin{aligned} m &= \frac{1}{\frac{1}{n_0} + \frac{1}{n_1}} \\ \delta' &= \frac{\delta}{ln(n)} \\ \varepsilon &= \sqrt{\frac{2}{m} \cdot \sigma_W^2 \cdot ln\frac{2}{\delta'}} + \frac{2}{3 \cdot m} \cdot ln\frac{2}{\delta'} \end{aligned} \tag{2.14}$$

Avtorji v [26] predstavljajo dve različici algoritma ADWIN. Pri prvi imenovani ADWIN0 gre za prostorsko in časovno zahteven algoritem, saj okno W naivno (na vse možne kombinacije) razdeli na podokni W_0 in W_1, vsebino okna pa eksplicitno hrani v pomnilniku. Pri drugi različici imenovani ADWIN2 uporabljajo posebno podatkovno strukturo imenovano *eksponentni histogram* [65], zaradi katere izboljšajo časovno in prostorsko zahtevnost algoritma iz O($|W|$) na O(log$|W|$) ob vsakem vstavljanju elementa, kjer $|W|$ označuje velikost okna. ADWIN2 je opisan z algoritmom 4.

Algorithm 4: ADWIN2

Vhod: stopnja zaupanja δ; podatkovni tok x_t

Izhod: signal ob spremembi koncepta, velikost okna, pričakovana vrednost v podatkovnem toku x_t

```
1 inicializiraj W kot prazen seznam histogramov;
2 inicializiraj VELIKOST, VARIANCA in VSOTA;
3 inicializiraj DRIFT ← False;
4 for t > 0 do
5 |   SetInput(x_t, W);
6 |   vrni μ̂_W kot VSOTA / VELIKOST in DRIFT;
  end
```

Funkcija SetInput *(*element e, seznam W*)*

```
1 |   InsertElement(e, W);
  |   izračunaj ε;
3 |   while |μ̂_{W_0} − μ̂_{W_1}| >= ε za vsako delitev okna W na
  |    W = W_0 ∪ W_1 do
4 |   |   DeleteElement(W);
  |   end
```

Funkcija InsertElement *(*element e, seznam W*)*

```
1 |   inicializiraj nov histogram b z elementom e in kapaciteto 1;
2 |   W ← W ∪ {b} (dodaj e v glavo W);
3 |   posodobi VELIKOST, VARIANCA in VSOTA;
4 |   CompressBuckets(W);
```

Funkcija DeleteElement *(*seznam W*)*

```
1 |   odstrani histogram iz repa seznama W;
2 |   posodobi VELIKOST, VARIANCA in VSOTA;
3 |   DRIFT ← True;
```

Funkcija CompressBuckets *(*seznam W*)*

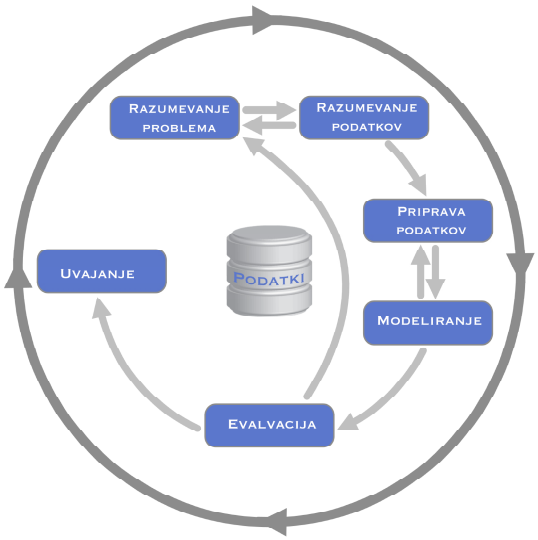
```
1 |   foreach histogram ∈ W.sort (sortiranje naraščujoče glede na
  |    kapaciteto histogramov) do
2 |   |   if če je več kot M histogramov enake velikosti then
3 |   |   |   CompressBuckets(podseznam seznama W, ki še ni bil
  |   |   |    obiskan);
  |   |   end
  |   end
```

Poglavje 3

Eksperimentalna evaluacija

3.1 CRISP-DM

CRISP-DM [66] 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 magistrskega dela. Poglavje je strukturirano skladno s fazami tega procesa, razvidnimi na sliki 3.1 in opisanimi v nadaljevanju.



Slika 3.1: Faze procesa CRISP-DM.

3.2 Razumevanje problema

Kot omenjamo v poglavju 1, lahko kvalitetno napovedovanje proizvodnje električne energije doprinese k nižjim stroškom predvsem za distributerje (ki napovedi koristijo za regulacijo napetosti v omrežju in določitev cen), posledično pa tudi za odjemalce električne energije. V tem poglavju se osredotočimo na napovedovanje proizvodnje električne energije sončnih elektrarn.

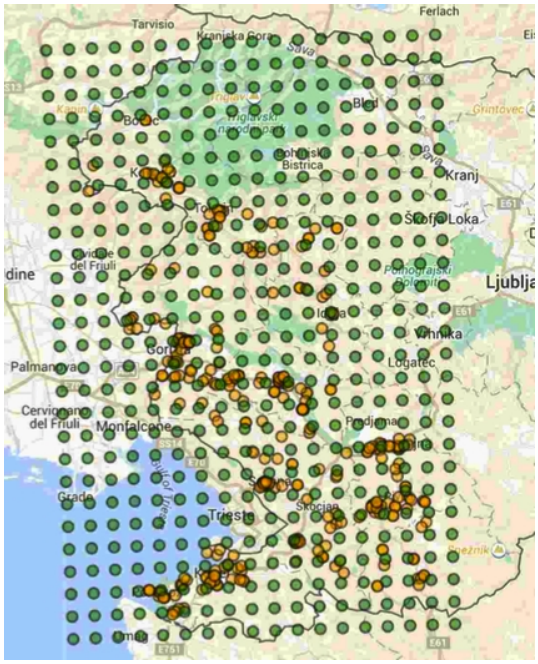
Naj opišemo primer scenarija uporabe metod v praksi, ki jih uporabljamo v tem delu, pred tem pa definirajmo točke v času:

- $t + \Delta t$ — čas, za katerega model ALADIN napoveduje vrednosti vremenskih parametrov in za katerega z modeli strojnega učenja napovedujemo proizvodnjo električne energije,
- t — čas, ko se je izvedla napoved modela ALADIN za čas $t+\Delta t$ oz. čas, v katerem napovedujemo proizvodnjo električne energije za čas $t + \Delta t$,
- Δt — število ur od časa t (ko izvedemo napoved) do časa $t + \Delta t$ (časa, za katerega se napoveduje); $\Delta t \leq 24$.

V času t pridobimo vremenske napovedi modela ALADIN. Za napovedovanje proizvodnje za neko elektrarno se osredotočimo na vremenske napovedi najbližje modelske točke tej elektrarni. Na podlagi teh podatkov in podatkov o pretekli proizvodnji neke elektrarne izračunamo izpeljane atribute. Tako pridobimo atribute $X_{t+\Delta t}$, katere uporabimo kot vhod v naš model. Ta na podlagi atributov izračuna napovedi proizvodnje električne energije $\hat{y}_{t+\Delta t}$. Ko kasneje pridobimo prave meritve o proizvodnji $y_{t+\Delta t}$ na neki elektrarni, jih skupaj z $X_{t+\Delta t}$ uporabimo za posodobitev modela iste elektrarne.

3.3 Razumevanje podatkov

Projekt ALADIN (*Aire Limitée Adaptation dynamique Développement International*) je eden najpomembnejših mednarodnih projektov francoske meteorološke službe na področju numerične napovedi vremena. Cilj projekta je iz-



Slika 3.2: Z zeleno so označene modelske točke, z oranžno barvo pa sončne elektrarne na območju primorske regije v Sloveniji [5].

boljšanje meteoroloških, hidroloških in okoljskih opozoril in napovednih storitev, ki jih podpirajo člani projekta [67, 68]. 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. Tako ima pomembno vlogo tudi pri promociji meteorologije v Sloveniji [68].

V magistrskem delu uporabljamo podatke proizvodenj sončnih elektrarn in vremenskih napovedi modela ALADIN. Prve nam je priskrbelo podjetje Elektro Primorska, druge pa Agencije Republike Slovenije za okolje - ARSO. Podatki pokrivajo celotno območje primorske regije v Sloveniji, kar je razvidno iz slike 3.2. Uporabljali smo zgodovinske podatke za časovno obdobje iz let 2011, 2012, 2013 in 2014, katere smo hranili v podatkovni bazi MySQL. Vsi podatki skupaj, ki smo jih za omenjena časovna obdobja prejeli od pod-

jetij in shranili v podatkovno bazo, obsegajo 69.669.766 zapisov.

Tu bi omenili, da je bila podrobna analiza omenjenih podatkov opravljana v okviru dela z naslovom *Napovedovanje dnevne proizvodnje električne energije sončnih elektrarn* [5]. V delu so med drugim podrobno analizirali napake v meritvah sončnih elektrarn, pomembnosti atributov za modeliranje, raziskali vpliv več modelskih točk modela ALADIN na določeno lokacijo elektrarne itd. Poleg omenjenega smo se iz dela naslonili tudi na postopke čiščenja in priprave ter na način shranjevanja podatkov in izpeljavo novih atributov.

3.4 Priprava podatkov

Za čiščenje in pripravo zgoraj opisanih podatkov ter izpeljavo novih atributov za namene modeliranja smo uporabljali postopke, ki so bili razviti v okviru prej omenjenega dela [5], kjer so proizvodnjo napovedovali za 211 sončnih elektrarn in za obdobje le dveh let. V nadaljevanju smo opisali le okvirne postopke pred priprave podatkov. Natančnejše utemeljitve in podrobne analize podatkov so opisane v omenjenem delu. Po uporabi teh postopkov smo izluščili podatke za modeliranje 304 sončnih elektrarn (na voljo smo imeli podatke o proizvodnji za 345 elektrarn).

3.4.1 Proizvodnja elektrarn

Spoznavanje in priprava podatkov

Podatke smo po podpisu izjave o varovanju podatkov pridobili od podjetja Elektro Primorska. Obsegajo 345 elektrarn od leta 2011 do vključno leta 2015 in zavzemajo 2,7 GB prostora. Za vsako elektrarno smo pridobili podatke o:

- številki merilnega mesta,
- datumu priklopa,
- naslovu (kraj in hišna številka) in/ali lokaciji,
- velikostnem razredu proizvodne naprave,

- količini letne proizvedene energije in
- nazivni moči proizvodne naprave (priključna moč).

Meritve za obdobje od 1.1.2011 do 16.11.2015 so shranjene na vsakih 15 minut, torej imamo za vsak dan za vsako elektrarno 96 meritev. Osnovna enota prejetih meritev je kilovat (kW[1]) ali pa kilovat ura (kWh[2]), odvisno od starosti števca na elektrarni. Tiste meritve, ki so v kilovatnih urah, strogo naraščajo (akumulirajo). Omeniti velja, da se meritve ne pošiljajo v realnem času, ampak se običajno pošljejo vsako uro (torej 4 meritve), ali pa enkrat na dan, odvisno od števca in velikosti elektrarne.

Točnih koordinat vseh elektrarn podjetje Elektro Primorska nima. Za nekatere elektrarne smo dobili koordinate v geografskih kartezičnih koordinatah Gauss-Krugerjevega sistema, katere smo zaradi konsistentnosti s koordinatami modela ALADIN pretvorili v sistem WGS84[3]. Za nekatere elektrarne smo pridobili koordinate najbližje transformatorske postaje, ki so od elektrarne lahko oddaljene največ 300 metrov. Elektrarnam z manjkajočimi koordinatami smo le-te pridobili z naslovov in hišno številko ter z javno dostopnimi storitvami zemljevidov Bing, Google in Nominatim (dostopnimi v okviru projekta *OpenStreetMap*[4]).

Nekatere elektrarne so imele podatke v akumulacijski obliki in njihova vrednost proizvodnje ni predstavljala trenutne, ampak akumulacijo proizvodnje od začetka postavitve elektrarne čez vsa leta. Take elektrarne smo ročno označili in podatke pretvorili v ne akumulacijsko obliko.

[1]Vat oz. kilovat (1 kW = 1000 W) je izpeljana enota za moč, toplotni tok, svetlobni tok in druge oblike energijskega toka. 1 W je enak 1 J/s, kar pri električnih enotah ustreza 1 V·A.

[2]Kilovatna ura (kWh) je fizikalna enota za delo in energijo, enaka 3.600.000 J. Ena kilovatna ura ustreza delu, ki ga opravi porabnik z močjo 1000 W v času 1 ure.

[3]WGS84 je različica koordinatnega sistema, določena leta 1984. Določena je tako, da se najbolje prilega Zemlji v celoti, torej da je vsota odmikov površja Zemlje od referenčnega elipsoida minimalna. Uporabljajo ga sistemi GPS.

[4]OpenStreetMap je odprto kodni projekt, ki podpira storitve v povezavi z zemljevidi. Dostopen je na: `www.openstreetmap.org`

Tabela 3.1: Kode za statuse podatkov.

status	opis
0	validiran
1	nevalidiran
2	nadomesten podatek
3	popravljen podatek
4	simulirani podatki
5	agregirani podatki
6	manjkajoči podatek
7	napaka po validaciji
8	napaka

Obravnavanje napak

Pri vsaki meritvi je zabeležen status meritve, ki ima lahko eno izmed devetih diskretnih vrednosti napisane v tabeli 3.1. Opis vsake je podrobneje opisan v [5] in dokumentu *Standardizirani merilni in obračunski podatki*[5], ki določa standardizirani zapis podatkov in način posredovanja merilnih in obračunskih podatkov s strani izvajalcev nalog Sistemskega operaterja distribucijskega omrežja (SODO) na celotnem področju Republike Slovenije. Kot napačne podatke smo označili tiste, ki imajo v tabeli 3.1 status različen od 0 (enačba (3.1)).

$$Napaka = \begin{cases} Ne, & \text{če je status} = 0, \\ Da, & \text{sicer.} \end{cases} \tag{3.1}$$

Lastnik elektrarne mora za pridobitev soglasja podati priključno moč elektrarne. To pomeni, da ima lahko elektrarna le manjšo proizvodnjo od priključne moči in navedene naj ne bi presegla. Ugotovili smo, da imajo

[5]Dokument je dostopen na: `https://www.sodo.si/_files/903/Standardizirani_merilni_in_obracunski_podatki_V2_mar_2015.pdf`.

Tabela 3.2: Število napačnih meritev (po enačbi (3.1)) na vseh elektrarnah po posameznih obdobjih.

Obdobje	Število napačnih meritev
1.1.2011—1.1.2012	7.389.288
1.1.2012—1.1.2013	4.147.164
1.1.2013—1.1.2014	1.106.638
1.1.2014—1.1.2015	185.031

nekatere elektrarne meritve proizvodnje tudi nekajkrat večje od priključne moči verjetno zato, ker lastniki elektrarn podatke oz. meritve prirejajo. Te smo označili kot napačne.

Dneve proizvodenj elektrarn kateri imajo več kot 15 meritev napačnih (spomnimo, da imamo na dan za eno elektrarno zabeleženih 96 meritev), smo označili kot *slabe*. Nato smo podatke s kodo različno od 0 iz tabele 3.1 skupaj s tistimi, katere smo sami označili kot napačne, nadomestili s podatki linearne interpolacije, pri čemer nismo upoštevali/interpolirali dni, katere smo označili kot *slabe*. Tako smo se izognili problemu neuspešne/slabe interpolacije. Pri tem smo uporabljali orodje *scipy*[6]. Še enkrat omenimo, da je podrobna analiza napak v podatkih opisana v [5]. Tam tudi navajajo, da bi se moralo število napačnih meritev v zadnjih letih zmanjšati, saj se je veliko števcev na elektrarnah moderniziralo, zato se je informacijski sistem shranjevanja izboljšal. To smo opazili tudi na naših podatkih in prikazali v tabeli 3.2. Ugotovili smo, da je število napak (to je meritev, ki je po enačbi 3.1 označena kot napačna) na vseh elektrarnah v letu 2015 okoli 40-krat manj kot v letu 2011.

Korigirano priključno moč elektrarn smo ponovno izračunali glede na največjo proizvodnjo elektrarne v celotnem obdobju, saj bodo tako porazdelitve proizvodnje elektrarn bolj primerljive — priključno moč namreč upo-

[6]Scipy je odprto kodna programska knjižnica za programski jezik Python, ki skrbi za podporo matematičnim operacijam in metodam, ki se uporabljajo pri inženiringu. Dostopna je na: `www.scipy.org`

rabljamo pri računanju mer uspešnosti. Na koncu smo odstranili tiste elektrarne, kjer je veljavnih dni manj kot 150, 15-minutne proizvodnje pa smo povprečili, da smo pridobili enourno proizvodnjo.

3.4.2 Vremenske napovedi

Spoznavanje in priprava podatkov

Podatke o vremenskih napovedih modela ALADIN za leta 2011, 2012, 2013 in 2014 smo pridobili od Agencije Republike Slovenije za okolje - ARSO. Zaradi pomanjkanja diskovnega polja ARSO zgodovinskih podatkov ne hrani, hrani pa začetne pogoje. Modelske napovedi so tako ponovno izračunali iz začetnih pogojev. Za leto 2015 vremenske napovedi žal nismo uspeli pridobiti, ker ponovni izračuni še niso bili končani.

Vremenske napovedi za leti 2011 in 2012 smo pridobili iz različice modela ALADIN, katera se je uporabljal še do nedaljnega. Pred kratkim so model posodobili, zato smo napovedi za leti 2013 in 2014 pridobili iz te posodobljene različice modela. Podatke smo pridobili v 40.180 *.csv*[7] datotekah. Vsaka datoteka vsebuje napovedi vremenskih parametrov za neko uro v omenjenem obdobju. Napovedi, ki jih uporabljamo za namene tega dela, se generirajo enkrat na dan za 30 ur vnaprej (torej 30 datotek), zavzemajo 450 MB prostora in obsegajo 486 modelskih točk v ločljivosti 4,4 km. Slednje pomeni, da vrednost v eni modelski točki predstavlja povprečno vrednost za površino 19,36 km^2. Modelska točka za nek čas vsebuje napovedi za naslednje vremenske parametre (v oglatih oklepajih so napisane enote):

- zemljepisna širina [stopinj],
- zemljepisna dolžina [stopinj],
- temperatura zraka na dveh metrih [K],
- relativna zračna vlaga na dveh metrih [%],

[7]Csv (angl. *Comma Separated Values*) je oblika zapisa podatkov običajno ločenih z vejico.

- oblačnost ali delež pokritosti [%],
- u komponenta vetra [m/s],
- v komponenta vetra [m/s],
- kumulativna količina padavin od izračuna napovedi do določene ure napovedi $[l/m^2]$,
- kumulativno globalno kratkovalovno sevanje pri tleh $[J/m^2]$,
- kumulativno neto kratkovalovno sevanje pri tleh $[J/m^2]$,
- kumulativno neto kratkovalovno sevanje na vrhu atmosfere $[J/m^2]$,
- kumulativno neto dolgovalovno sevanje pri tleh $[J/m^2]$,
- kumulativno globalno dolgovalovno sevanje pri tleh $[J/m^2]$,
- geopotencial $[J/kg]$ in
- kopno / morje [0 / 1].

Postopek priprave podatkov vremenskih napovedi je bil precej manj obsežen kot postopek priprave podatkov o proizvodnji elektrarn. Napovedi kumulativnih vremenskih parametrov (izmed prej naštetih so to npr. količina padavin od izračuna napovedi do določene ure napovedi, neto kratkovalovno sevanje pri tleh itd.) so bile v akumulacijski obliki, katere smo pretvorili v trenutne vrednosti — za določeno uro v dnevu. Kot atribute za modeliranje smo uporabili napovedi vseh vremenskih parametrov, razen geopotenciala ter indeksa za kopno in morje, saj bi ta dva atributa imela varianco 0.

3.4.3 Izpeljava novih atributov

Poleg atributov omenjenih v prejšnjem podpoglavju, smo glede na predznanje o napovedovanju električne energije in [69, 70, 71] za namene modeliranja pridobili ali izpeljali tudi naslednje atribute:

- ura v dnevu
- proizvodnja na prejšnji dan ob isti uri,
- proizvodnja na podoben dan ob isti uri,

- dolžina dneva,
- indeks sončnega sevanja G in
- razmerje med neto kratkovalovnim sevanjem in indeksom sončnega sevanja.

Uro v dnevu in proizvodnjo prejšnjega dne ob isti uri smo smo pridobili na trivialen način. Da smo pridobili drugega od zgoraj omenjenih atributov, smo morali poiskati podoben dan. Tega iščemo vedno samo v preteklosti ($d_p \leq d_t$, kjer je d_p podoben in d_t trenuten dan), izračunali pa smo ga na podlagi vremenskih napovedi z normalizirano neuteženo evklidsko razdaljo za vsako modelsko točko. Dolžino dneva smo izračunali z urno razliko med sončnim zahodom in sončnim vzhodom. Indeks sončnega sevanja pri čistem nebu G smo izračunali z enačbo (3.2) [69]. Gre za matematično izračunane vrednosti, ki so odvisne od položaja v koordinatnem sistemu Zemlje (z zemljepisno dolžino in zemljepisno širino (ϕ)), časa (dan v letu d), sončne deklinacije (kot med zveznico Zemlja-Sonce in ravnino ekvatorja označen kot δ), urnega kota (h_s) in solarne konstante (G_0), katere vrednost je 1367 W/m^2. V množico atributov smo dodali še razmerje med neto kratkovalovnim sevanjem in indeksom sončnega sevanja (N_{ir}/G).

$$G = G_0 \left(1 + 0,033 \cos\left(\frac{360d}{365}\right)\right) \Big(\cos(\delta)\cos(h_s)\cos(\phi) + \sin(\delta)\sin(\phi)\Big) \quad (3.2)$$

3.5 Modeliranje in evalvacija

Glavni cilj faze modeliranja nam je bil napovedovanje proizvodnje sončnih elektrarn. Celotno fazo smo opravljali v programskem okolju Python na računalniku z Linux OS in s procesorjem *Intel Xeon X5650.* Gre za procesor s 6 fizičnimi jedri in frekvenco 2,66 GHz. Uporabljali smo 10 navideznih jeder, tako da smo lahko hkrati modelirali proizvodnjo za 10 sončnih elektrarn.

3.5.1 Uporabljeni atributi in normalizacija

Podatke vremenskih napovedi za neko elektrarno smo vedno pridobili iz elektrarni najbližje modelske točke modela ALADIN. Proizvodnjo sončnih elektrarn $\hat{y}_{t+\Delta t}$ smo napovedovali iz 19 zveznih atributov, med katerimi jih je večina vremenskih napovedi modela ALADIN za čas $t + \Delta t$:

- Δt (ura v dnevu),
- razmerje med neto kratkovalovnim sevanjem in indeksom sončnega sevanja,
- dolžina dneve,
- indeks sončnega sevanja G,
- proizvodnja na podoben dan ob isti uri,
- proizvodnja na prejšnji dan ob isti uri,
- temperatura zraka,
- vlažnost,
- u komponenta vetra,
- v komponenta vetra,
- oblačnost,
- kumulativna količina padavin,
- kumulativno globalno kratkovalovno sevanje pri tleh,
- kumulativno neto kratkovalovno sevanje pri tleh,
- kumulativno neto kratkovalovno sevanje na vrhu atmosfere,
- kumulativno neto dolgovalovno sevanje pri tleh,
- kumulativno globalno dolgovalovno sevanje pri tleh.

V podpoglavju 2.2.2 opisujemo prednosti transformacije podatkov pred učenjem. Rekordne vrednosti precej izmerjenih vremenskih parametrov na območju Slovenije se lahko pridobi npr. iz meteoroloških biltenov (objavljenih na spletni strani ARSO), zato bi lahko za največje in najmanjše vrednosti pri transformaciji podatkov predpostavili te (ali malo višje/nižje od

teh). Lahko bi se omejili tudi na neko smiselno območje. Mi smo pri transformaciji podatkov uporabili največje in najmanjše vrednosti, ki so nam bile na voljo med podatki, ki smo jih pridobili. Vse atribute oz. neodvisne spremenljivke smo z enačbo (3.3) skalirali na zaprti interval $[0, 1]$. V enačbi je *x'* nova transformirana, x stara vrednost, *max(x)* ter *min(x)* pa največja oziroma najmanjša vrednost iz množice ne netransformiranih vrednosti. Na vrednosti izven zaprtega intervala med modeliranjem nismo naleteli.

$$x' = \begin{cases} \frac{x - min(x)}{max(x) - min(x)}, & \text{če je } max(x) \neq min(x), \\ 0, & \text{sicer.} \end{cases} \tag{3.3}$$

3.5.2 Uporabljene metode

Algoritmi strojnega učenja

Za napovedovanje proizvodnje sončnih elektrarn uporabljamo algoritme strojnega učenja, ki so na voljo v programskih knjižnicah *scikit-learn*[8], Pyflux in *Vowpal Wabbit*[9]. Modelirali smo v programskem jeziku Python.

V nadaljevanju naštejemo algoritme, katere smo uporabljali. V oklepajih so izrazi, katere najdemo v uradni dokumentaciji in katere bomo uporabljali v nadaljevanju. Iz knjižnice scikit-learn uporabljamo algoritem naključnih gozdov (*random forest*), večnivojski perceptron (*MLPRegressor*), algoritem *Online Passive-Aggressive* [72] (*PassiveAggressiveRegressor*) in stohastični gradientni spust z metodo najmanjših kvadratov (*SGDRegressor*). Pri paketnem učenju smo vsem modelom nastavili parameter *warm start=false*, pri sprotnem učenju pa *warm start=true*. Slednji parameter namreč pred

[8]Scikit-learn je odprto kodna programska knjižnica s podporo strojnemu učenju za programski jezik Python. Dostopna je na: `www.scikit-learn.org`.

[9]Vowpall Wabbit (VW) je odprto kodni programski paket s podporo sprotnemu učenju z vmesniki pa programske jezike Python, C++, Java, R ... Avtorji paketa sta med drugimi raziskovalna organizacija *Yahoo! Research* in podjetje *Microsoft*. Dostopen je na: `https://github.com/JohnLangford/vowpal_wabbit`.

učenjem nastavi začetne vrednosti iskanih parametrov na vrednosti, katere je pridobil po zadnjem učenju. Modelom, katerim se da nastaviti parameter *learning rate*, smo slednjega nastavili na *constant*, ker želimo, da se korak učenja (angl. *learning rate*) s časom oz. številom iteracij učenja ne spreminja, saj je naš podatkovni tok potencialno neskončen. Povsod smo uporabili reguralizacijo *Ridge* oz. *L2* z različno nastavljenim regularizacijskim parametrom. Vrednosti ostalih parametrov so privzete.

Iz knjižnice Vowpall Wabbit uporabljamo umetne nevronske mreže (*neural network VW*). Uporabljamo jih s parametrov *holdout off*, ki onemogoči metodo izločanje neodvisne testne množice (podpoglavje 2.3.1). Z notacijo *arch* označujemo arhitekturo umetnih nevronskih mrež (primer: arch=(50,50) pomeni, da ima nevronska mreža 2 skrita nivoja s po 50 skritih nevronov na nivo). Poleg regularizacije *Ridge* smo poskusili tudi s tehniko *dropout*. Uporabljamo jo na vseh nevronskih mrežah razen na tisti, ki ima le 24 skritih nevronov. Ostali parametri so nastavljeni na privzete vrednosti.

Algoritem ADWIN

Uporabljamo učinkovitejšo različico algoritma ADWIN (opisano v podpoglavju 2.7.2, kjer je imenovana kot ADWIN2) implementirano v programskem jeziku Python. Implementacija algoritma je učinkovita oz. skladna z omenjeno časovno in prostorsko učinkovitostjo ter je odvisna le od programske knjižnice *Numpy*[10].

Vhod v ADWIN je poleg podatkovnega toka še stopnja zaupanja δ. Avtorji v člankih [26, 73] uporabljajo vrednosti 0,05, 0,1 in 0,3, v članku [74] pa omenijo, da je priporočena vrednost 0,2. Na nekaj elektrarnah smo preizkusili vrednosti 0,1 in 0,2, razlike pa so bile zanemarljive v prid prvi vrednosti, zato smo se odločili, da bomo uporabljali $\delta = 0{,}1$.

[10]Numpy je programska knjižnica za programski jezik Python, ki skrbi za podporo numeričnim operacijam, več-dimenzionalnim tabelam itd. Dostopna je na: `www.numpy.org`

Algoritem rezervoarskega vzorčenja

Za namene dela uporabljamo izboljšano različico rezervoarskega vzorčenja z eksponentno funkcijo, katero smo omenili v podpoglavju 2.7.1 in je implementirana v programskem modulu oz. knjižnici *AppMetrics*[11]. Implementacija vzorčenja v omenjenem paketu je privzeto taka, da lahko v rezervoarju hrani le števila. Modificirali smo jo tako, da smo spremenili dve vrstici izvorne kode in s tem omogočili vzorčenje poljubnih podatkovnih tipov oz. elementov.

Rezervoarju, katerega vzdržuje algoritem rezervoarskega vzorčenja z eksponentno funkcijo, smo velikost nastavili na 2000. Slednje pomeni, da v rezervoarju hranimo podatke za skupno obdobje 83 dni. Povprečna velikost okna, ki ga vzdržuje algoritem ADWIN, je v našem primeru največkrat znašala približno 2000 elementov, zato smo velikost rezervoarja nastavili na enako vrednost. Avtor je v [9] omenil, da velikost okna za obdobje 100 dni daje najboljše rezultate, kar je podobna velikost kot v našem primeru. Na osnovi priporočil avtorja uporabljamo privzeto vrednost $\lambda = 0{,}015$.

3.5.3 Ocenjevanje uspešnosti

Posamezna elektrarna

Za ocenjevanje uspešnosti učenja modela na posamezni elektrarni smo uporabljali *prepletanje testiranja in učenja* (poglavje 2.3.2), torej smo z modeli najprej napovedovali, na koncu pa jih posodobili z novimi primeri iz podatkovnega toka (kot opišemo v nadaljevanju, vedno napovedujemo in posodabljamo z množico veliko 24 primerov, to je en dan). Poudariti velja, da modelov nismo nikoli učili s podatki iz prihodnosti ali z njimi napovedovali na podlagi atributov primera, katerega smo že uporabili za učenje.

Kot glavno mero uspešnosti na posamezni elektrarni smo uporabljali različne povprečne napake do časa oz. do primera i (enačba (2.2)). Torej

[11] AppMetrics je modul za programski jezik Python, ki skrbi za podporo metodam in podatkovnim strukturam primernim za sprotno računanje statistike. Dostopen je na: `https://pypi.python.org/pypi/AppMetrics/`.

smo sproti računali povprečne napako in sicer povprečno absolutno napako (MAE, enačba (3.4)) oz. normalizirano povprečno absolutno napako (nMAE, enačba (3.5)). Slednjo smo uporabili zaradi lažje primerjave uspešnosti algoritmov, dobili pa smo jo tako, da smo MAE delili z maksimalno močjo elektrarne. V omenjenih enačbah je n število vseh primerov v testni množici, y_i prava in $\hat{y}_i$ napovedana vrednost ter PS maksimalna moč elektrarne.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.4}$$

$$nMAE = \frac{MAE}{PS} \tag{3.5}$$

Vse elektrarne skupaj

Vse napovedi vseh modelov na vseh elektrarnah smo shranjevali za kasnejše analize. Na koncu smo želeli ugotoviti, kako smo s posameznim modelom uspešni pri napovedovanju na vseh sončnih elektrarnah oz. pridobiti nekakšno generalno uspešnost nekega modela. Če bi za to vzeli povprečje nMAE vseh elektrarn, bi dobili napačen rezultat, saj napake na neki elektrarni normaliziramo z največjo močjo te elektrarne, največje moči pa se od elektrarne do elektrarne običajno razlikujejo.

Enačba (3.6) definira seštevek vseh napak, enačba (3.7) pa seštevek vseh absolutnih napak skozi celotno obdobje na neki elektrarni e. Z enačbo (3.8) dobimo seštevek maksimalne moči (katera se sicer na posamezni elektrarni ne spreminja; ta je označena s PS) skozi celotno obdobje na neki elektrarni e. Normalizirano povprečno absolutno napako (nMAE) za nek model na vseh elektrarnah izračunamo z enačbo (3.10), normalizirano povprečno napako (nMBE) pa z enačbo (3.9), kjer je m število vseh elektrarn. Zadnja enačba nam pove, ali z nekim modelom v povprečju napovedujemo preveč ali premalo.

$$E(e) = \sum_{i}(y_{i,e} - \hat{y}_{i,e}) \tag{3.6}$$

$$AE(e) = \sum_i |y_{i,e} - \hat{y}_{i,e}| \tag{3.7}$$

$$P(e) = \sum_i PS_{i,e} \tag{3.8}$$

$$nMBE = \frac{\frac{1}{m}\sum_{e\in el.}^{m} E(e)}{\frac{1}{m}\sum_{e\in el.}^{m} P(e)} = \frac{\sum_{e\in el.}^{m} E(e)}{\sum_{e\in el.}^{m} P(e)} \tag{3.9}$$

$$nMAE = \frac{\frac{1}{m}\sum_{e\in el.}^{m} AE(e)}{\frac{1}{m}\sum_{e\in el.}^{m} P(e)} = \frac{\sum_{e\in el.}^{m} AE(e)}{\sum_{e\in el.}^{m} P(e)} \tag{3.10}$$

Primerjava parov algoritmov na podatkovnem toku

Ocenjena uspešnost posameznih algoritmov nam služi za primerjavo uspešnosti algoritmov. Če je razlika uspešnosti posameznih algoritmov relativno majhna, je primerjava lahko statistično neznačilna, še posebej če je standardna napaka posameznih ocen uspešnosti relativno velika [23]. Razlikovati želimo med statistično značilnimi in neznačilnimi razlikami v napovedih oz. rezultatih eksperimentov. V te namene uporabljamo različne statistične teste.

Test Diebold-Mariano

Na področju napovedovanja časovnih vrst se za primerjavo napovednih modelov pogosto uporablja *test Diebold-Mariano* [75]. V ekonomiji se ga pogosto uporablja za testiranje ali sta dva algoritma za napovedovanje časovnih vrst enako uspešna [76, 77, 78, 79] . S testom Diebold-Mariano (DM) želimo dokazati statistično značilno razliko med uspešnostjo ene in druge množice napovedi na *isti* domeni torej razliko med uspešnostjo dveh algoritmov za napovedovanje časovnih vrst na *istem* podatkovnem toku. Za test moramo imeti na voljo prave vrednosti $\{y_i;\ i = 1, \ldots, n\}$, napovedi prvega $\{\hat{y}_{A,i};\ i = 1, \ldots, n\}$ in drugega algoritma $\{\hat{y}_{B,i};\ i = 1, \ldots, n\}$, kjer je n število vseh napovedi oz. primerov v testni množici. Naj bo $L^A(y_i, \hat{y}_{A,i})$ oz. $L^B(y_i, \hat{y}_{B,i})$ funkcija mere

uspešnosti med napovedjo prvega oz. drugega algoritma in dejansko vrednostjo na primeru/trenutku i, pri čemer lahko uporabimo skoraj katerokoli funkcijo mere uspešnosti, npr. kvadrat povprečne napake (MSE), povprečno absolutno napako (MAE) itd. Z enačbo (3.11) definirajmo razliko teh dveh mer uspešnosti.

$$d_i = L^A(y_i, \hat{y}_{A,i}) - L^B(y_i, \hat{y}_{B,i}) \tag{3.11}$$

S testom DM testiramo ničelno hipotezo 3.12 proti alternativni hipotezi 3.13. Torej poskušamo ovreči ničelno hipotezo ki pravi, da sta dva algoritma enako uspešna pri napovedovanju v vsakem primeru i.

$$H_0 : E(d_i) = 0, \forall i \tag{3.12}$$

$$H_A : E(d_i) \neq 0 \tag{3.13}$$

Gre za dvosmerni test, pri katerem ničelno hipotezo zavržemo, če je izračunana statistika DM izven intervala $[-z_{\alpha/2}, z_{\alpha/2}]$, torej če drži pogoj v enačbi 3.14, pri čemer je $z_{\alpha/2}$ zgornja (pozitivna) vrednost standardiziranega odklona (iz tabele normalne porazdelitve) s stopnjo zaupanja $1 - \alpha$.

$$|DM| > z_{\alpha/2} \tag{3.14}$$

Torej pri podani stopnji zaupanja 95 % ($\alpha = 0.05$) hipotezo H_0 zavržemo, če dobimo vrednost DM izven intervala $[-1.96, +1.96]$. S testom DM pridobimo tudi *p-vrednost*[12]. Zelo velika ali zelo majhna (negativna) DM statistika in majhna p-vrednost pomenita, da je malo verjetno, da hipotezo H_0 lahko zavrnemo.

Ker primerjamo več parov algoritmov na istem podatkovnem toku, moramo uporabiti *Bonferronijevo korekcijo* [80], ki je za celotno primerjavo mnogo strožja. Z Bonferronijevo korekcijo zavržemo ničelno hipotezo (da sta algoritma enako uspešna) le, če je razlika med algoritmoma dovolj velika.

[12]V grobem rečeno je p-vrednost verjetnost, da je bila zavrnitev hipoteze H_0 napačna.

Korekcija zahteva strožjo vrednost α_B za dosego enake stopnje značilnosti α. Če N krat ponovimo poskus (ali če imamo N različnih algoritmov) z značilnim rezultatom stopnje α_B, je skupna stopnja značilnosti definirana z enačbo (3.15) [23]. Torej ničelno hipotezo zavrnemo pod pogojem iz enačbe 3.16. Npr., za 12 primerjav ($N = 12$) in zahtevo, da je stopnja značilnosti $\alpha = 0{,}05$, je Bonferronijeva korekcija stopnje značilnosti $\alpha_B = 0{,}0041$.

$$\alpha_B = 1 - (1 - \alpha)^{\frac{1}{N}} \approx \frac{\alpha}{N} \tag{3.15}$$

$$|DM| > z_{\alpha_B/2} \tag{3.16}$$

Statistika Q

Avtorji v članku [81] kot metodo za primerjavo uspešnosti algoritmov na podatkovnih tokovih predlagajo *statistiko Q*. Gre za učinkovito in inkrementalno metodo, s katero lahko sproti primerjamo uspešnost dveh algoritmov na enem podatkovnem toku. Naj bosta $P_e^A(i)$ in $P_e^B(i)$ povprečni meri uspešnosti do primera/trenutka i za algoritem A oziroma B. Tudi tukaj lahko uporabimo skoraj katerokoli mero uspešnosti, npr. kvadrat povprečne napake (MSE), povprečno absolutno napako (MAE) itd. Statistiko Q definiramo z enačbo (3.17).

$$Q_i(A, B) = log(\frac{P_e^A(i)}{P_e^B(i)}) = log(P_e^A(i)) - log(P_e^B(i)) \tag{3.17}$$

Statistika Q v trenutku i (Q_i) nam govori o relativni uspešnosti obeh algoritmov v primeru/trenutku i, njegovo vrednost pa interpretiramo kot moč razlike (torej nam pove, koliko je en napovedni model boljši od drugega). Vrednosti $Q_i(A, B)$ v času i pomenijo:

- $Q_i(A, B) < 0$: A je boljši od B,
- $Q_i(A, B) = 0$: ni razlike,
- $Q_i(A, B) > 0$: B je boljši od A.

3.5.4 Modeliranje z različnimi pristopi

V končni fazi smo modelirali na predpripravljeni *statični* množici podatkov, vendar je vse uporabljene postopke priprave podatkov mogoče uporabiti na podatkih, ki prihajajo v realnem času oz. na podatkovnih tokovih. Podatke pripravljene za modeliranje, smo shranjevali v podatkovni bazi, nad katero smo izvajali poizvedbe med procesom modeliranja. Na ta način smo simulirali podatkovni tok.

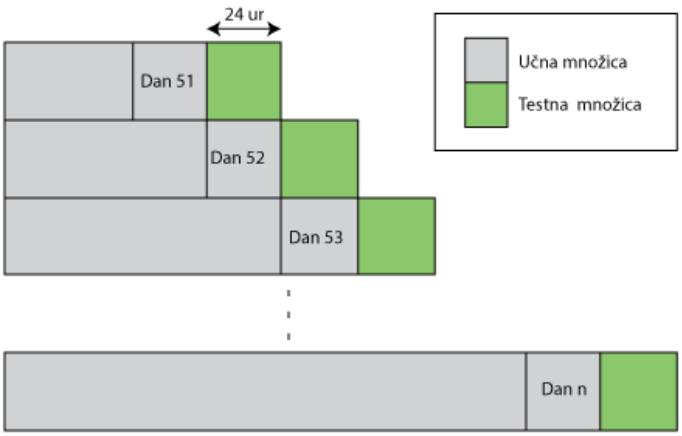
Modeliranje poteka za vsako sončno elektrarno posebej, torej smo pridobili 304 zgoraj omenjenih mer uspešnosti skozi celotno obdobje.

Proizvodnjo električne energije smo napovedovali tako s klasičnimi algoritmi strojnega učenja kot z algoritmi, ki so delu s podatkovnimi tokovi že prilagojeni. Zaradi različnih načinov uporabe prvih in drugih smo modelirali s 6 različnimi pristopi, ki so opisani v nadaljevanju.

Ne glede na velikost množice podatkov elektrarne, uporabimo prvih 50 dni proizvodnje za začetek učenja vseh modelov oz. za tako imenovani *hladni zagon*. V članku [82] so za začetek učenja modela uporabljali množico velikosti največ 25 dni, vendar so imeli podatke le za eno leto. Ker pridobimo vremenske napovedi za 1 dan vnaprej, postopek modeliranja v vseh v nadaljevanju opisanih pristopih izvajamo na tako imenovanih *mini-batch* množicah, ki vsebujejo 24 primerov — po 1 primer za vsako uro, torej skupaj za 1 dan. Napovedovali smo za 24 ur vnaprej, torej vsakič smo napovedali 24 vrednosti.

Paketni pristop brez vzorčenja

Pristop je prilagojen klasičnim algoritmom strojnega učenja, v poglavju 2.2.1 omenjen kot pristop z ovojnico. Ko z modelom napovemo za naslednjih 24 ur, ga zavržemo in ponovno naučimo na učni množici, v katero smo dodali primere iz trenutnega dne — iz dne, za katerega smo pred tem napovedovali. Učno množico tako inkrementalno povečujemo. To pomeni, da se na podatkih za obdobje npr. 100 dni zgradi 100 modelov (če hladnega zagona ne upoštevamo) oz. da ima model vedno znanje, ki ga je pridobil na vseh



Slika 3.3: Shema paketnega pristopa.

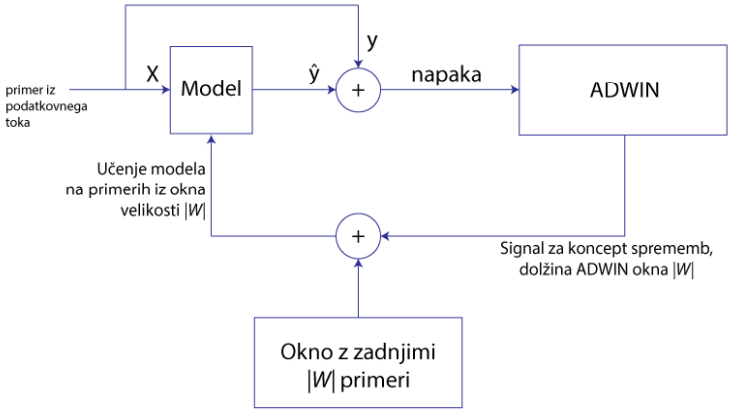
podatkih do sedaj.

Paketni pristop z vzorčenjem ADWIN

Tudi tu gre za pristop z ovojnico. Z modelom naučenim v hladnem zagonu, napovemo za naslednji dan, nato pa iz napovedi in prave vrednosti izračunamo absolutno napako (oz. 24 napak, po eno za vsako napovedano uro) definirano z enačbo (3.18), kjer je $\hat{y}$ napovedana, y pa prava vrednost proizvodnje električne energije za neko elektrarno.

$$napaka = |y - \hat{y}| \tag{3.18}$$

Izračunane napake nato podamo kot vhod v algoritem ADWIN. Če ta v podatkovnem toku oz. v podanih napakah zazna spremembo koncepta, model zavržemo oz. ponovno naučimo na $|W|$ zadnjih podatkih iz podatkovnega toka, kjer je $|W|$ nova velikost okna, ki ga vzdržuje oz. vrne algoritem ADWIN. Pri tem zadnjih $|W|$ primerov hranimo eksplicitno v svojem oknu oz. podatkovni strukturi FIFO — v njo vedno dodajamo primere na konec vrste, samo ob spremembi koncepta pa jih iz začetka vrste odstranimo toliko, da v njej ostane $|W|$ elementov (kolikor je nova velikost ADWIN okna).



Slika 3.4: Shema paketnega pristopa z ADWIN algoritmom.

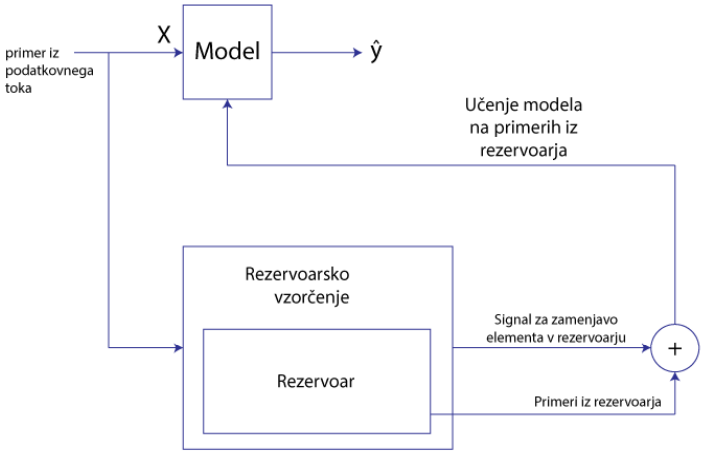
Model naj bi tako vseboval znanje, pridobljeno iz zadnjega trenda podatkov.

Paketni pristop z rezervoarskim vzorčenjem

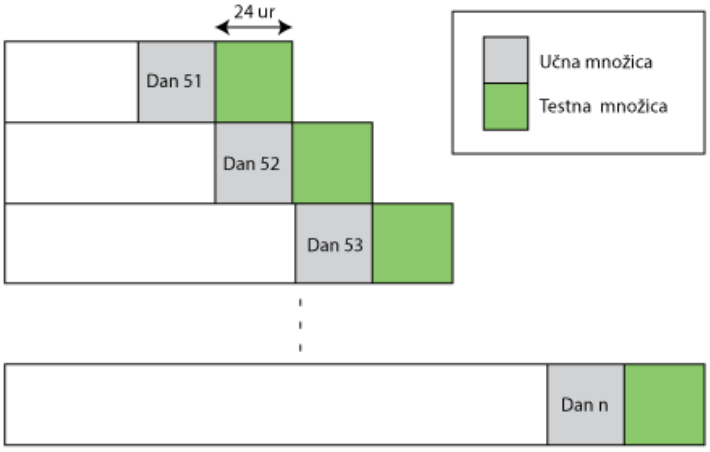
V tem pristopu uporabljamo rezervoarsko vzorčenje z eksponentno funkcijo, opisano v poglavju 2.7.1. Primere iz podatkovnega toka poskušamo sproti vstavljati v rezervoar. Če se to zgodi oz. če pride do zamenjave novega elementa iz podatkovnega toka z elementom v rezervoarju, model zavržemo oz. ponovno naučimo na elementih iz rezervoarja. Model naj bi s tem pristopom vseboval znanje pridobljeno iz vzorca, v katerem je verjetnost ohranitve novejšega elementa (iz celotnega obdobja podatkovnega toka) večja kot starejšega.

Sprotni pristop brez vzorčenja

V tem pristopu uporabljamo algoritme, ki so prilagojeni za delo s podatkovnimi tokovi. Ko z modelom napovemo za naslednjih 24 ur, ga posodobimo s primerom trenutnega dne — s primerom, za katerega smo pred tem napovedovali. Tako ima model vedno znanje, ki ga je pridobil iz vseh podatkov do sedaj.



Slika 3.5: Shema paketnega pristopa z rezervoarskim vzorčenjem.



Slika 3.6: Shema sprotnega pristopa.

Sprotni pristop z vzorčenjem ADWIN

Gre za sprotno učenje modelov z novimi primeri v kombinaciji z uporabo algoritma ADWIN. Uporaba slednjega je podobna, kot v paketnem pristopu z algoritmom ADWIN. Torej če ta v podatkovnem toku oz. v podanih napakah zazna spremembo koncepta, model zavržemo oz. ponovno naučimo na $|W|$ zadnjih podatkih iz podatkovnega toka, kjer je $|W|$ nova velikost okna, ki ga vzdržuje algoritem ADWIN. V primeru da pa ADWIN ne zazna spremembe koncepta, model le posodobimo z novim primerom (oz. s 24 primeri za trenutni dan).

Sprotni pristop z rezervoarskim vzorčenjem

Gre za sprotno učenje modelov z novimi primeri v kombinaciji z uporabo rezervoarskega vzorčenja z eksponentno funkcijo. Slednjega uporabljamo na podoben način kot pri paketnem pristopu z rezervoarskim vzorčenjem. Torej če pride do zamenjave v rezervoarju, model zavržemo in ponovno naučimo na primerih iz rezervoarja. Če pa ne pride do zamenjave, model le posodobimo z novim primerom (oz. s 24 primeri za trenutni dan).

Poglavje 4

Rezultati

V podjetju E3, ki je hčerinsko podjetje podjetja Elektro Primorska, s trenutnimi napovedmi odjema in proizvodnje vseh elektrarn (vključno s hidroelektrarnami, ki imajo zelo predvidljivo napoved proizvodnje) na tedenskem trgu dosegajo v letnem profilu napako do 3 % [5]. To pomeni, da so napovedi sončnih elektrarn z uspešnostjo nMAE nekaj nad 3 %, kot jih dosegajo naši modeli, pravzaprav zelo dober rezultat.

Modelirali smo z modeli z različno nastavljenimi parametri. Z nekaterimi kombinacijami algoritmov in parametrov smo dobili relativno slabe ali nesmiselne rezultate, katere smo v nadaljnji analizi izpustili. Kot primer izpostavimo časovne vrste ARIMA in ARIMAX s parametri *(p=1, d=0, q=1)* ali *($d \neq 0$)*, katere so vračale nesmiselne ali nenatančne napovedi. Polja z "/" v nekaterih grafih označujejo model, ki nekega pristopa modeliranja (privzeto) ne podpira.

Pri ocenjevanju uspešnosti nekega modela na vseh elektrarnah skupaj smo se v nadaljevanju osredotočali na mero uspešnosti nMAE (enačba (3.5)), pri posamezni elektrarni pa na povprečno nMAE do časa oz. primera i (enačba (2.2)).

4.1 Vse elektrarne skupaj

V tabeli 4.1 so prikazani rezultati modeliranja predstavljeni kot povprečje normaliziranih srednjih absolutnih napak (enačba (3.10)) vseh 304 sončnih elektrarn. Najboljše rezultate nMAE (0,0337) dosegamo z modelom naključni gozdovi in paketnim pristopom modeliranja, generalno slabe rezultate pa s časovnimi vrstami ARIMA (do 0,189). Uspešni smo bili tudi z umetnimi nevronskimi mrežami iz paketa Vowpal Wabbit, s katerimi smo pri sprotnem načinu modeliranja z dodanim algoritmom rezervoarskega vzorčenja dobili napako do 0,0363.

Pri osredotočanju na vse tri paketne načine opazimo, da modeli v kombinacijami z algoritmom ADWIN dajejo slabše rezultate, razen v primeru časovnih vrst. Podobno velja za modele v kombinaciji z rez. vzorčenjem, s tem da se tej obnesejo bolje kot modeli z algoritmom ADWIN. Razlog, da z rezervoarskim vzorčenjem dobimo boljše rezultate kot z ADWIN, bi lahko bil v tem, da starejši primeri iz podatkovnega toka, kateri se v vzorcu ohranijo z rezervoarskim vzorčenjem, na nek način naredijo model bolj splošen oz. manj prilagojen na najnovejše podatke. ADWIN namreč vzdržuje vzorec, ki je sestavljen samo iz zadnjih primerov podatkovnega toka. S časovnimi vrstami ARIMAX smo pridobili precej boljše rezultate kot z ARIMA, iz česar lahko sklepamo o pomembnosti atributov oz. neodvisnih spremenljivk — časovne vrste ARIMAX namreč upoštevajo tudi te.

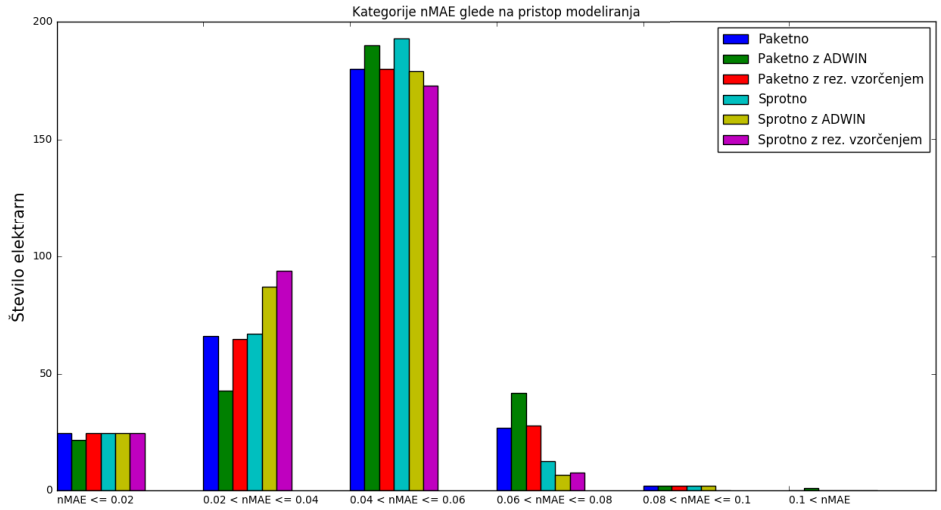
Če se osredotočimo na vse tri sprotne pristope modeliranja pa opazimo, da se modeli brez rezervoarskega vzorčenja ali ADWIN vedno obnesejo slabše kot v kombinaciji z omenjenimi algoritmi. Glede na rezultate v člankih [26, 83], v katerih testirajo inkrementalne modele z algoritmom ADWIN, smo pričakovali podobne rezultate.

Tabela 4.1: Povprečna normalizirana absolutna napaka na vseh elektrarnah z različnimi modeli in pristopi.

Model	Paketni	Paketni z ADWIN	Paketni z rez. vzorč.	Sprotni	Sprotni z ADWIN	Sprotni z rez. vzorč.
Random forest ($n = 40$)	0,0337	0,0376	0,0351	/	/	/
MLPRegressor ($L2\ \alpha = 0.001, SGD, arch = (24, 24), \eta_0 = 0.00001$)	0,0415	0,045	0,0428	0,0446	0,0431	0,0423
MLPRegressor ($L2\ \alpha = 0.001, SGD, arch = (50, 50), \eta_0 = 0.00001$)	0,0416	0,045	0,0427	0,0446	0,043	0,0423
MLPRegressor ($L2\ \alpha = 0.01, SGD, arch = (50, 50), \eta_0 = 0.00001$)	0,0483	0,0451	0,045	0,0441	0,043	0,0423
PassiveAggressive Regressor ($\epsilon = 0.001, C = 0.1$)	0,0453	0,0528	0,0475	0,0414	0,04	0,039
PassiveAggressive Regressor ($\epsilon = 0.001, C = 0.01$)	0,0449	0,052	0,0468	0,0412	0,038	0,0374
SGDRegressor ($L2\ \alpha = 0.01, \eta_0 = 0.01$)	0,0469	0,0477	0,0459	0,0471	0,0468	0,0466
SGDRegressor ($L2\ \alpha = 0.1, \eta_0 = 0.01$)	0,041	0,045	0,0431	0,0588	0,059	0,0588
ARIMAX (p=1, d=0, q=2)	0,051	0,048	0,054	/	/	/
ARIMAX (p=2, d=0, q=1)	0,067	0,061	0,068	/	/	/
ARIMA (p=1, d=0, q=2)	0,145	0,118	0,146	/	/	/
ARIMA (p=2, d=0, q=1)	0,187	0,161	0,189	/	/	/
neural network (VW) ($L2\ \alpha = 0.001, \eta_0 = 0.5, arch = (24)$)	/	/	/	0,0391	0,0372	0,0363
neural network (VW) ($L2\ \alpha = 0.001, \eta_0 = 0.75, arch = (50)$)	/	/	/	0,0424	0,0405	0,0398
neural network (VW) ($L2\ \alpha = 0.01, \eta_0 = 0.1, arch = (50)$)	/	/	/	0,0413	0,0386	0,0387
neural network (VW) ($L2\ \alpha = 0.01, \eta_0 = 0.5, arch = (50)$)	/	/	/	0,0427	0,0411	0,04

Če se osredotočimo na tiste modele, ki podpirajo paketni in sprotni pristop modeliranja ter primerjamo rezultate med obema pristopoma, vidimo, da so rezultati modelov brez dodatnih algoritmov vzorčenja ponekod v prid

paketnemu pristopu, ponekod pa sprotnemu. Modeli z dodanimi algoritmi vzorčenja pa so v sprotnem vedno bolj uspešni kot v paketnem pristopu. Iz tega in prejšnjega odstavka bi lahko sklepali, da se algoritme vzorčenja v našem primeru splača dodati le sprotnim modelom. Vendar če primerjamo rezultate uspešnosti algoritma naključni gozdovi brez (nMAE 0,0337) in z rezervoarskim vzorčenjem (nMAE 0,0351) ter upoštevamo čas učenja algoritmov, bi lahko sklepali tudi drugače. Kot opišemo v razdelku 4.4, vzorčenje precej pripomore k času učenja algoritmov.



Slika 4.1: Število elektrarn po kategorijah nMAE za vsak pristop modeliranja.

Graf 4.1 s stolpčnimi diagrami prikazuje število sončnih elektrarn v posamezni kategoriji nMAE in za posamezen pristop modeliranja. Opazimo, da na največ elektrarnah (preko 170) pridobimo nMAE med 0,04 in 0,06. Na relativno malo elektrarnah (okoli 25) smo uspešni z nMAE pod 0,02.

Tabela A.1 v dodatku tega dela prikazuje napako nMBE. Iz te napake vidimo, da precej modelov ki smo jih uporabljali pri napovedovanju električne energije, napoveduje do približno 0,5 % preveliko proizvodnjo glede na skupno maksimalno močjo elektrarn.

Poskusili smo tudi z ansambelsko napovedjo. Kot slednjo smo uporabljali povprečno vrednost napovedi vseh modelov, s katerimi smo modelirali. Torej če smo za neko postajo in neko točko v času z modelom naključnih gozdov napovedali vrednost 5 kW, z modelom umetne nevronske mreže pa 7 kW, smo kot ansambelsko napoved vzeli 6 kW. Uporaba algoritmov rezervoarskega vzorčenja in ADWIN se v tem primeru nanaša na posamezne modele. V tabeli 4.2 so prikazani rezultati pridobljeni iz povprečenja napovedi vseh modelov, ki smo jih sproti uporabljali pri vsakem pristopu modeliranja. Komentarji rezultatov iz prejšnjega odstavka se lahko nanašajo tudi na to tabelo.

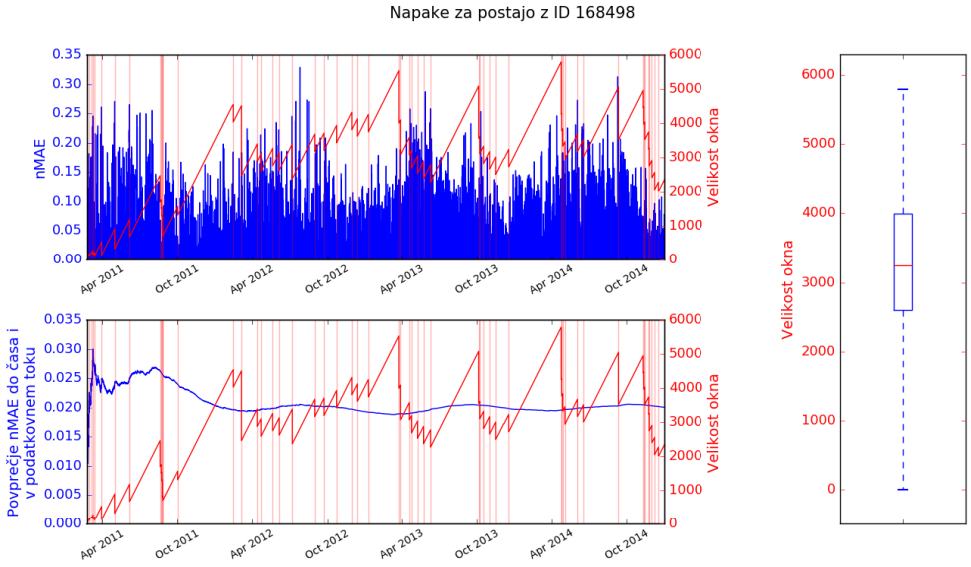
Tabela 4.2: Rezultati pridobljeni iz povprečenja napovedi vseh modelov, ki smo jih sproti uporabljali pri vsakem pristopu modeliranja.

Mera uspešnosti	Paketni	Paketni z ADWIN	Paketni z rez. vzorč.	Sprotni	Sprotni z ADWIN	Sprotni z rez. vzorč.
nMAE	0,0403	0,0436	0,0412	0,0412	0,0397	0,0393
nMBE	-0,0048	-0,0033	-0,0034	-0,00118	-0,0014	-0,0015

4.2 Posamezna elektrarna

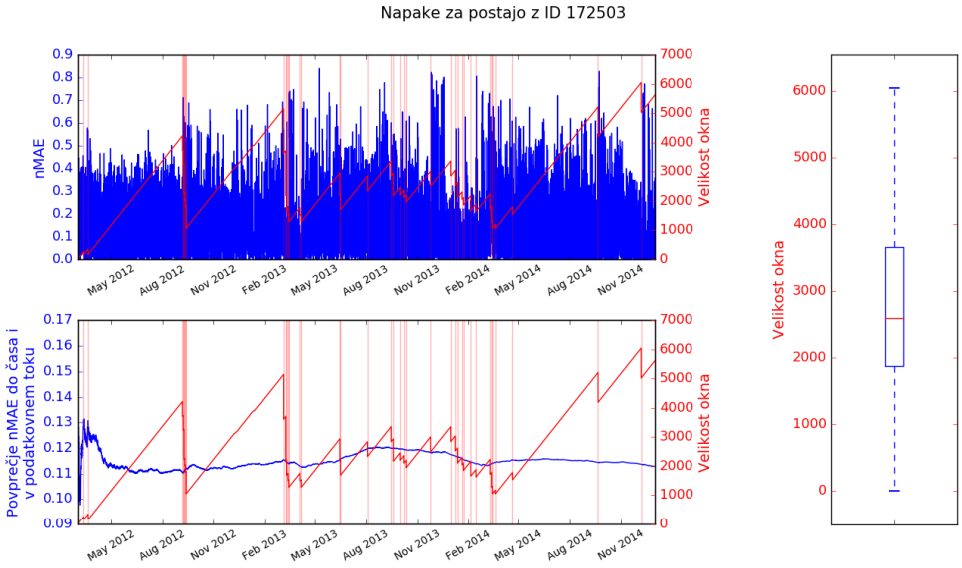
V nadaljevanju se bomo osredotočili na rezultate pridobljene na posamezni elektrarni za nek pristop modeliranja. Prikazali smo jih na grafih z napakami nMAE skozi celotno obdobje in povprečno nMAE do časa oz. primera i. Na nekaterih grafih so prikazane tudi velikosti prilagodljivega drsečega okna, katerega je vzdrževal algoritem ADWIN.

Iz grafa 4.2 poleg nMAE in povprečnega nMAE vidimo tudi velikosti prilagodljivega drsečega okna pri paketnem načinu modeliranja z ADWIN. Povprečna velikost okna pri modeliranju elektrarne z ID 168498 je okoli 3200 primerov, kar je 133 dni. ADWIN je zaznal 54 sprememb koncepta. Opazimo, da se okno največ skrči v obdobjih okoli oktobra 2011, oktobra 2012, oktobra 2013 in spet oktobra 2014 oz. ko je bilo v oknu okoli 5500 primerov. Z omenjenimi dogodki lahko morda povežemo vpliv letnih časov na vreme in s tem na proizvodnjo električne energije — model naučen na podatkih iz

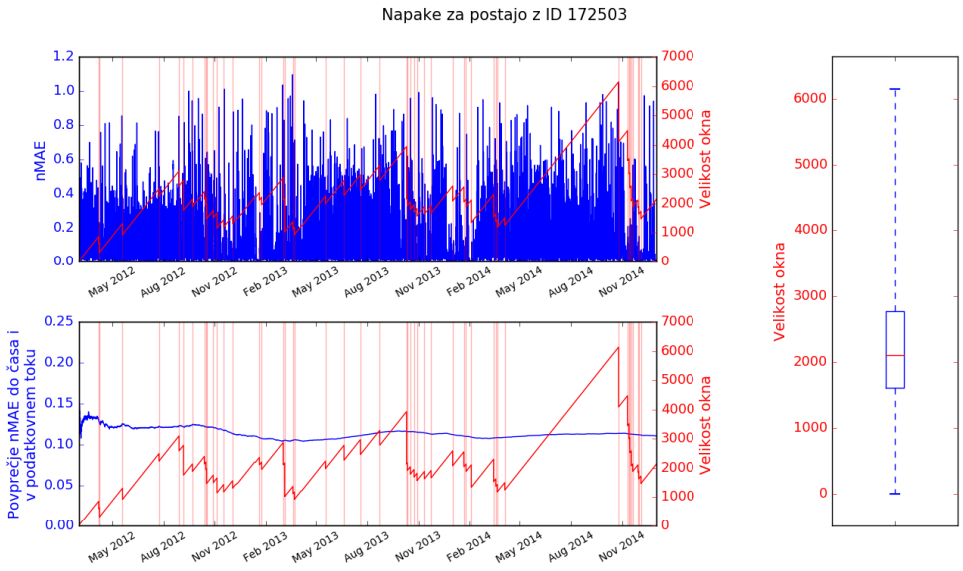


Slika 4.2: Graf nMAE skozi čas za elektrarno z ID 168498 pri paketnem načinu modeliranja z ADWIN, kjer je povprečen nMAE relativno nizek oz. 0,02. Rezultati so izračunani iz napovedi modela Random Forest ($n = 40$) in algoritma ADWIN, kateri je zaznal 54 sprememb koncepta.

zimskega obdobja je manj natančen pri napovedovanju v poletnem obdobju, zato se napaka, ki je vhod v ADWIN, lahko poveča. Graf 4.3 prikazuje isti pristop modeliranja na elektrarni z ID 172503, kjer smo pridobili relativno visoko nMAE (0,112). Pri tem je ADWIN zaznal 38 sprememb koncepta. Zanimivo je naraščanje povprečne nMAE do oktobra 2013 in padanje po januarju 2014, ko ADWIN zazna spremembo. Vzrok za to bi lahko bili podatki (meritve) slabe kakovosti.



Slika 4.3: Graf nMAE skozi čas za postajo z ID 172503 pri paketnem načinu modeliranja z ADWIN, kjer je povprečen nMAE relativno visok oz. 0,112. Rezultati so izračunani iz napovedi modela SGDRegressor ($L2$ $\alpha = 0.1, \eta_0 = 0.01$) in algoritma ADWIN, kateri je zaznal 38 sprememb koncepta.

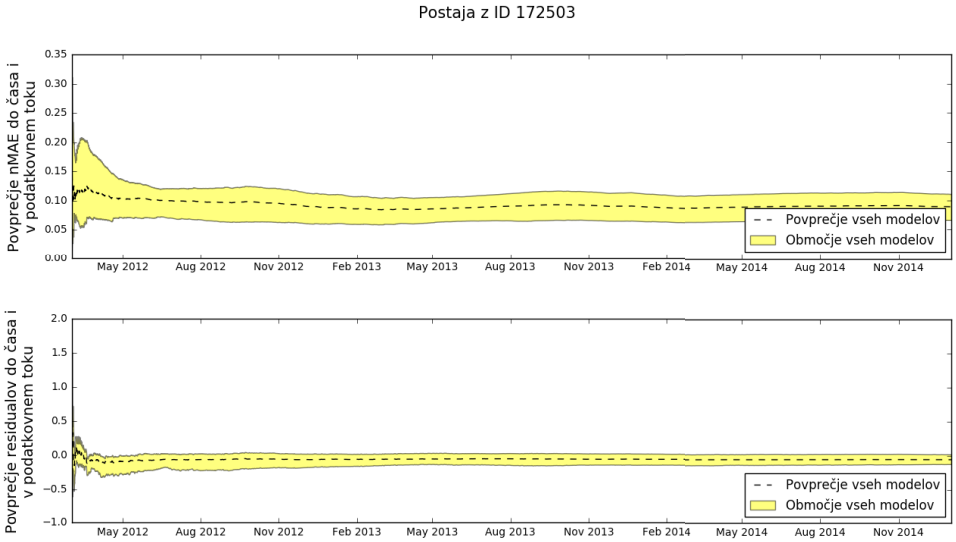


Slika 4.4: Graf nMAE skozi čas za postajo z ID 172503 pri sprotnem načinu modeliranja z ADWIN, kjer je povprečen nMAE relativno visok oz. 0,11. Rezultati so izračunani iz napovedi modela SGDRegressor ($L2$ $\alpha = 0.1, \eta_0 = 0.01$) in algoritma ADWIN, kateri je zaznal 48 sprememb koncepta.

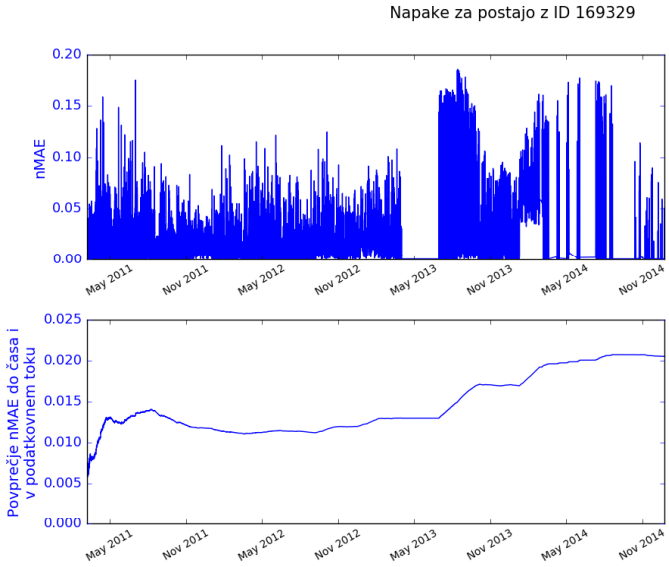
Rezultati oz. nMAE sprotnega načina z algoritmom ADWIN so razvidni na grafu 4.4. Gre za elektrarno z ID 172503, na kateri smo dobili relativno visok nMAE (0,11). Opazimo, da se povprečen nMAE s časom skoraj ne spreminja, velikost okna pa se povečuje tja do okoli 6050. Tu je ADWIN zaznal 48 sprememb koncepta, kateri pa bistveno ne vplivajo na napako modela. Na grafu 4.5 vidimo, da se na isti elektrarni in istem pristopu modeliranja *residuali*[1] ostalih modelov s časom tudi bistveno ne spreminjajo.

Graf 4.6 prikazuje modeliranje elektrarne, na kateri je bilo od aprila 2013 naprej ogromno meritev manjkajočih ali s proizvodnjo 0. Menimo, da je v tem obdobju šlo za neko okvaro na elektrarni. Tu modeliramo s paketnim načinom in modelom naključnih gozdov. Opazimo, da napaka po izpadu

[1]Residual je razlika med pravo vrednostjo y in napovedano vrednostjo $\hat{y}$. Običajno želimo, da so residuali normalno porazdeljeni okoli vrednosti 0, saj to pomeni, da z modeli ne napovedujemo preveč ali premalo.

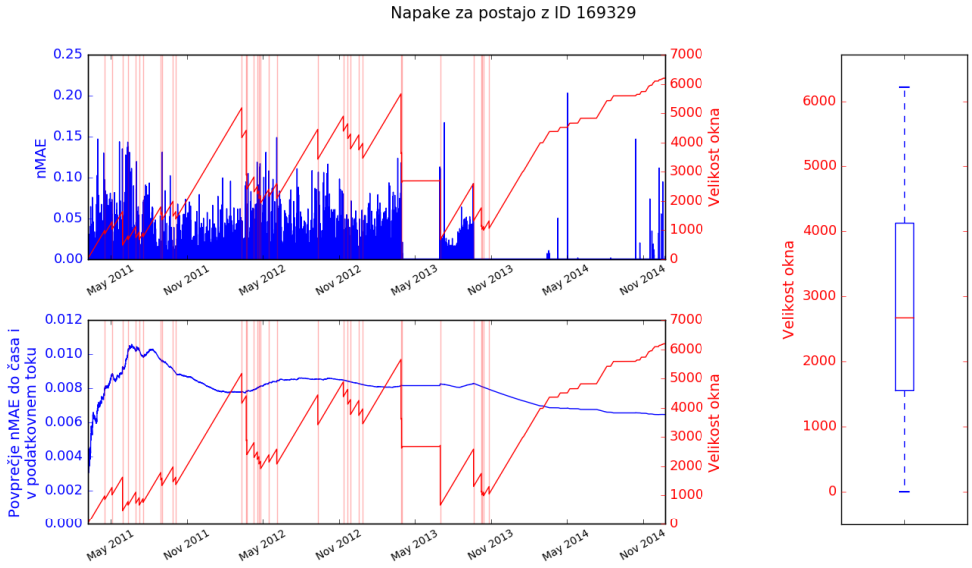


Slika 4.5: Primer območja nMAE in residualov vseh modelov skozi čas za postajo z ID 172503 pri sprotnem načinu modeliranja z ADWIN, kjer je povprečen nMAE relativno visok oz. 0,11.

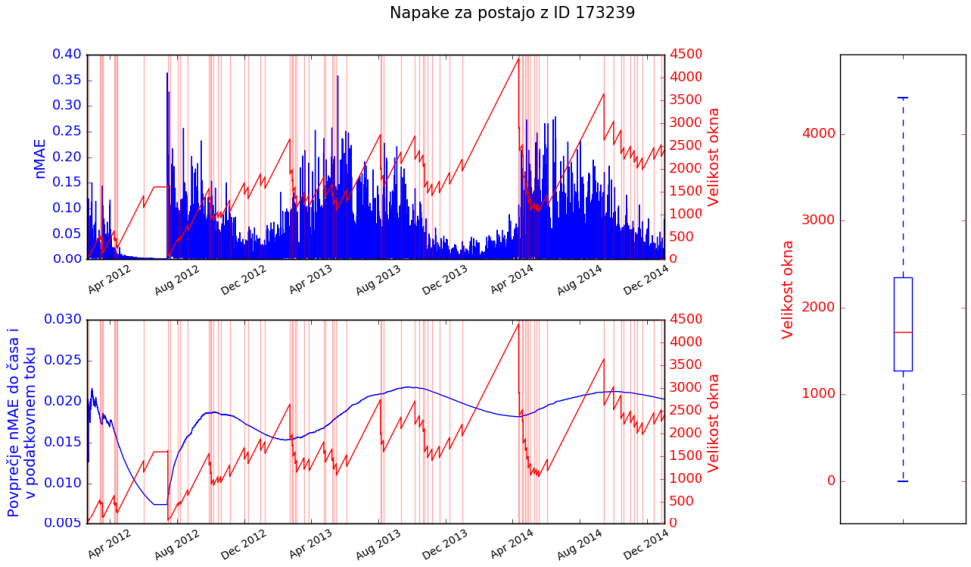


Slika 4.6: Graf nMAE skozi čas za elektrarno z ID 169329 pri paketnem načinu modeliranja, kjer je povprečen nMAE relativno nizek oz. 0,02. Rezultati so izračunani iz napovedi modela Random Forest ($n = 40$). Izpad meritev je viden v mesecih od aprila 2013 dalje.

podatkov drastično naraste, saj statični model vsebuje znanje še iz obdobja, ko z elektrarno domnevno ni bilo težav. Graf 4.7 prikazuje modeliranje na isti postaji, z istim modelom in paketnim principom z dodanim algoritmom ADWIN. Slednji drastično pripomore k uspešnosti modela naključnih gozdov, saj se ta nauči na svežih podatkih — na podatkih kjer je ogromno meritev s proizvodnjo 0. Tu je sprememba koncepta in občutljivost algoritma ADWIN zelo opazna. Na isto elektrarno in paketni princip modeliranja (brez ADWIN) se nanaša tudi graf A.4 (v dodatku A), iz katerega je razviden upad uspešnosti modelov v obdobju domnevne okvare elektrarne.



Slika 4.7: Graf nMAE skozi čas za elektrarno z ID 169329 pri paketnem načinu modeliranja z ADWIN, kjer je povprečen nMAE ekstremno nizek oz. 0,01. Rezultati so izračunani iz napovedi modela Random Forest ($n = 40$) in algoritma ADWIN, kateri je zaznal 34 sprememb koncepta.



Slika 4.8: Primer napak skozi čas za postajo z ID 168498 pri sprotnem načinu modeliranja z ADWIN, kjer je povprečen nMAE relativno nizek oz. 0,02. Rezultati so izračunani iz napovedi modela neural network (VW) ($L2\ \alpha = 0.01, \eta_0 = 0.1, arch = (50)$) in algoritma ADWIN, kateri je zaznal 74 sprememb koncepta.

Na grafu 4.8 pa lahko opazimo relativno visoka nihanja v povprečni napaki v obdobju. V časih julij 2012, marec 2013 in aprila 2014 se vidi negativni vpliv algoritma ADWIN na uspešnost modelov. Ko ADWIN zazna spremembo koncepta, napaka bistveno naraste. To se zgodi verjetno zaradi zmanjšanja velikosti okna. Posledice tega bi morda zmanjšali tako, da bi algoritmu ADWIN določili (spremenili implementacijo) najmanjšo velikost okna.

Na precej grafih (tudi tistih iz dodatka A) vidimo naraščanje in padanje napak nMAE in povprečna nMAE do časa i, kar ima verjetno povezavo z letnimi časi. Kot smo omenili v pregledu področja, se avtorji v članku [6] osredotočijo na napovedovanje porabe električne energije na področju Slovenije in ugotovijo, da so napovedi slabše v zimskem in poletnem obdobju medtem, ko so napovedi najboljše v spomladanskem in jesenskem času.

Opazili smo, da smo s precej modeli najboljše rezultate pridobili na elek-

trarni z ID 168498. Gre za elektrarno iz vasi Dobravlje v Vipavski dolini, za katero nismo imeli nobenih manjkajočih podatkov. Z nekaj modeli pa smo najslabše napovedovali za elektrarno z ID 172503. Tu gre za elektrarno iz okolice Idrije, za katero smo imeli na voljo podatke iz obdobja treh let. Špekulativno lahko rečemo, da na razlike v uspešnosti napovedovanja med omenjenima elektrarnama vpliva sama lokacija elektrarn (bližina obale, hribi) in stabilnejše vreme.

4.3 Medsebojne primerjave algoritmov

Kot smo že omenili, smo napovedi modelov shranjevali za kasnejše analize, zato smo za primerjavo algoritmov lahko uporabili statistični test Diebold-Mariano (DM). Uporabili smo stopnjo značilnosti $\alpha = 0{,}05$. Kot ničelno hipotezo smo postavili hipotezo, da sta primerjana algoritma enako uspešna. Zavrnili smo jo pod pogojem $|DM| > z_{\alpha_B/2} = 2{,}88$ oz. $p \leq \alpha_B/2 = 0{,}00205$ (pri računanju Bonferronijevega popravka smo uporabili $n = 12$). Alternativna hipoteza pa pravi, da je uspešnost primerjanih algoritmov različna. Kot funkcijo mer uspešnosti smo uporabili povprečno absolutno napako (MAE). Za primerjavo posameznih algoritmov smo uporabili tudi statistiko Q.

V tabelah in slikah v nadaljevanju za označbo algoritmov uporabljamo oznake iz tabele 4.3.

Tabela 4.3: Označbe algoritmov. Urejenost teh nima pomena.

Oznaka	Ime algoritma
0	RandomForest ($n = 40$)
1	MLPRegressor ($L2\ \alpha = 0.001, SGD, arch = (24, 24), \eta_0 = 0.00001$)
2	MLPRegressor ($L2\ \alpha = 0.001, SGD, arch = (50, 50), \eta_0 = 0.00001$)
3	MLPRegressor ($L2\ \alpha = 0.01, SGD, arch = (50, 50), \eta_0 = 0.00001$)
4	PassiveAggressive Regressor ($\epsilon = 0.001, C = 0.1$)
5	PassiveAggressive Regressor ($\epsilon = 0.001, C = 0.01$)
6	SGDRegressor ($L2\ \alpha = 0.01, \eta_0 = 0.01$)
7	SGDRegressor ($L2\ \alpha = 0.1, \eta_0 = 0.01$)
8	neural network (VW) ($L2\ \alpha = 0.001, \eta_0 = 0.5, arch = (24)$)
9	neural network (VW) ($L2\ \alpha = 0.001, \eta_0 = 0.75, arch = (50)$)
10	neural network (VW) ($L2\ \alpha = 0.01, \eta_0 = 0.1, arch = (50)$)
11	neural network (VW) ($L2\ \alpha = 0.01, \eta_0 = 0.5, arch = (50)$)

4.3.1 Vse elektrarne skupaj

Pri primerjavi parov algoritmov na vseh elektrarnah skupaj smo kot prave y_i in napovedane proizvodnje $\hat{y}_i$ vzeli vsoto prave oz. napovedane proizvodnje vseh elektrarn, ki so obratovale v trenutku i (oz. za katere imamo v trenutku i meritve in napovedi).

V vsakem pristopu modeliranja smo primerjali vsak algoritem z vsakim. V tabelah 4.4, 4.5 in 4.6 so prikazane p-vrednosti, ki smo jih pridobili s statističnim testom DM. S krepko pisavo so označeni primeri, kjer nismo

Tabela 4.4: P-vrednosti testa DM pri primerjavi parov algoritmov pri sprotnem pristopu.

	1	2	3	4	5	6	7	8	9	10	11
1	**1,00**										
2	0,00	**1,00**									
3	0,00	**0,0629**	**1,00**								
4	0,00	0,00	0,00	**1,00**							
5	0,00	0,00	0,00	0,00	**1,00**						
6	0,00	0,00	0,00	0,00	0,00	**1,00**					
7	0,00	0,00	0,00	0,00	0,00	0,00	**1,00**				
8	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**1,00**			
9	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**1,00**		
10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**1,00**	
11	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**0,00621**	0,00	**1,00**

Tabela 4.5: P-vrednosti testa DM pri primerjavi parov algoritmov pri sprotnem pristopu z ADWIN.

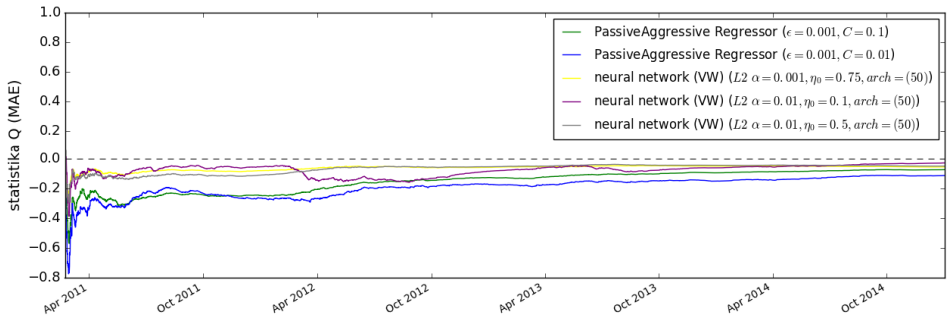
	1	2	3	4	5	6	7	8	9	10	11
1	1,00										
2	**0,7203**	**1,00**									
3	**0,1014**	**0,0363**	**1,00**								
4	0,00	0,00	0,00	**1,00**							
5	0,00	0,00	0,00	**0,8283**	**1,00**						
6	0,00	0,00	0,00	0,00	0,00	**1,00**					
7	0,00	0,00	0,00	0,00	0,00	0,00	**1,00**				
8	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**1,00**			
9	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**1,00**		
10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**1,00**	
11	0,00	0,00	0,00	0,00081	**0,0268**	0,00	0,00	0,00011	0,00	0,00	**1,00**

Tabela 4.6: P-vrednosti pridobljene iz statistike DM pri primerjavi parov algoritmov pri sprotnem pristopu z rezervoarskim vzorčenjem.

	1	2	3	4	5	6	7	8	9	10	11
1	1,00										
2	**0,0792**	**1,00**									
3	**0,3309**	**0,00365**	**1,00**								
4	0,00	0,00	0,00	**1,00**							
5	0,00	0,00	0,00	0,00	**1,00**						
6	0,00	0,00	0,00	0,00	0,00	**1,00**					
7	0,00	0,00	0,00	0,00	0,00	0,00	**1,00**				
8	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**1,00**			
9	0,00	0,00	0,00	0,00	**0,00527**	0,00	0,00	0,00	**1,00**		
10	0,00	0,00	0,00	0,00	0,00047	0,00	0,00	0,00	0,00	**1,00**	
11	0,00	0,00	0,00	**0,04463**	0,00	0,00	0,00	0,00	0,00	0,00	**1,00**

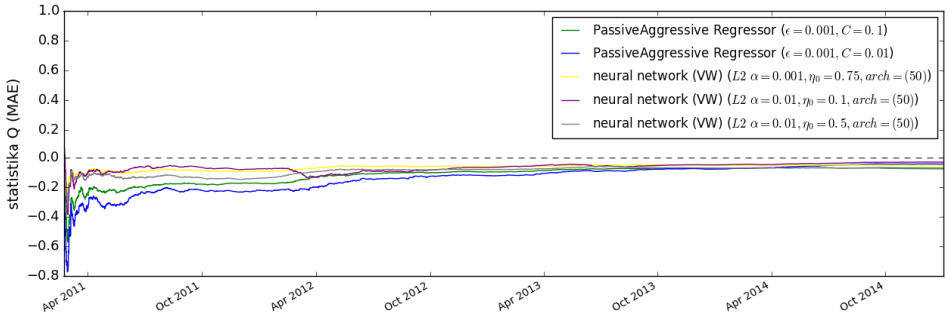
uspešno zavrnili ničelne hipoteze pri upoštevanju Bonferronijeve korekcije.

Opazimo, da razlike med uspešnostjo niso statistično značilne predvsem pri primerjavi istih algoritmov z različno nastavljenimi parametri, s katerimi smo dobili zelo podoben rezultat nMAE. Največjo p-vrednost (0,8283) smo pridobili pri primerjanju algoritmov 4 in 5 (iz tabele 4.3) pri sprotnem načinu z ADWIN. Statističnih rezultatov, ki smo jih pridobili s primerjanjem parov algoritmov v vseh treh paketnih pristopih, v delu nismo prikazovali — pri vseh parih primerjav smo ničelno hipotezo ovrgli, razen pri primerjanju algoritmov 1 in 2 pri paketnem pristopu z ADWIN, kjer smo pridobili p-vrednost 0,82. Z različno nastavljenimi umetnimi nevronskimi mrežami iz paketa Vowpal Wabbit smo pri sprotnih pristopih pridobili zelo podobne rezultate nMAE, vendar so uspešnosti glede na test DM skoraj povsod statistično značilno različne.



Slika 4.9: Primerjava parov algoritmov s statistiko Q pri sprotnem pristopu modeliranja na vseh elektrarnah. Neural network (VW) ($L2\ \alpha = 0.001, \eta_0 = 0.5, arch = (24)$) primerjamo z ostalimi petimi najuspešnejšimi algoritmi (po kriteriju nMAE) pri modeliranju v omenjenem pristopu.

S statistiko Q smo primerjali najuspešnejši algoritem (v enačbi 3.17 je ta označen z A) s petimi najuspešnejšimi algoritmi (v enačbi 3.17 so ti označeni z B) na vseh elektrarnah. Pri tem smo kot mero uspešnosti uporabili povprečno absolutno napako (MAE). Na grafu 4.9 vidimo primerjavo algoritmov pri sprotnem pristopu modeliranja. Opazimo, da je razlika v uspešnosti med



Slika 4.10: Primerjava parov algoritmov s statistiko Q pri sprotnem pristopu modeliranja z rezervoarskim vzorčenjem na vseh elektrarnah. Neural network (VW) ($L2\ \alpha = 0.001, \eta_0 = 0.5, arch = (24)$) primerjamo z ostalimi petimi najuspešnejšimi algoritmi (po kriteriju nMAE) pri modeliranju v omenjenem pristopu.

podobnima topologijama umetnih nevronskih mrež iz paketa Vowpal Wabbit majhna. Graf 4.10 prikazuje primerjavo s statistiko Q pri sprotnem načinu modeliranja z rezervoarskim vzorčenjem. Opazimo, da je razlika med najuspešnejšim in ostalimi algoritmi s časoma manjša kot pri sprotnem modeliranju brez rezervoarskega vzorčenja. Iz tega bi lahko sklepali, da rezervoarsko vzorčenje pripomore k uspešnosti algoritmov. Grafe ostalih pristopov modeliranja smo prikazali v dodatku A.

4.3.2 Posamezna elektrarna

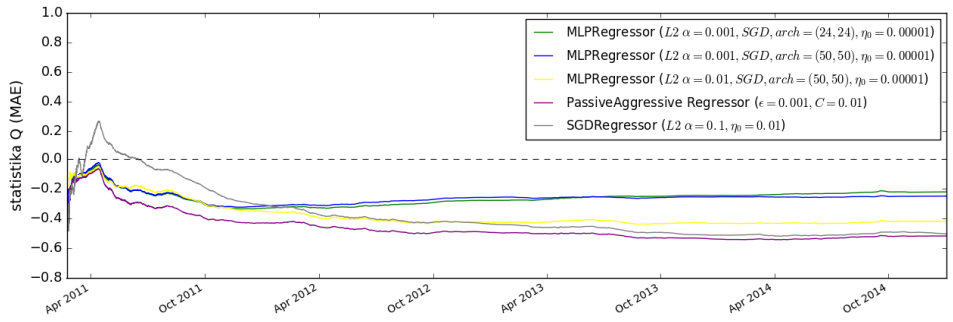
Opravili smo tudi medsebojno primerjavo algoritmov na podatkih elektrarne z ID 168498 — to je elektrarna, na kateri smo s precej modeli pridobili najboljše rezultate.

V vsakem pristopu smo primerjali algoritem, s katerim smo v pristopu pridobili najboljše rezultate nMAE (kar je razvidno v tabeli 4.1), z ostalimi algoritmi. Pri paketnih načinih je to algoritem Random forest s parametrom $n = 40$, pri sprotnem pa neural network (VW) s parametri $L2\ \alpha = 0.001$, $\eta_0 = 0.5$, $arch = (24)$. Prvi algoritem je v tabelah v nadaljevanju označen z ★Random forest, drugi pa z ♦neural network.

Tabela 4.7: P-vrednosti pri primerjanju parov algoritmov z mero uspešnosti MAE s testom DM.

	Paketni (★Random forest)	**Paketni z ADWIN** (★Random forest)	**Paketni z rez. vzorč.** (★Random forest)	**Sprotni** (♦neural network)	**Sprotni z ADWIN** (♦neural network)	**Sprotni z rez. vzorč.** (♦neural network)
★Random forest ($n = 40$)	**1,00**	**1,00**	**1,00**	/	/	/
MLPRegressor ($L2\ \alpha = 0.001, SGD, arch = (24, 24), \eta_0 = 0.00001$)	0,00	0,00	0,00	0,00	0,00	0,00
MLPRegressor ($L2\ \alpha = 0.001, SGD, arch = (50, 50), \eta_0 = 0.00001$)	0,00	0,00	0,00	0,00	0,00	0,00
MLPRegressor ($L2\ \alpha = 0.01, SGD, arch = (50, 50), \eta_0 = 0.00001$)	0,00	0,00	0,00	0,00	0,00	0,00
PassiveAggressive Regressor ($\epsilon = 0.001, C = 0.1$)	0,00	0,00	0,00	0,00	0,00	0,00
PassiveAggressive Regressor ($\epsilon = 0.001, C = 0.01$)	0,00	0,00	0,00	0,00	0,00	0,00
SGDRegressor ($L2\ \alpha = 0.01, \eta_0 = 0.01$)	0,00	0,00	0,00	0,00	0,00	0,00
SGDRegressor ($L2\ \alpha = 0.1, \eta_0 = 0.01$)	0,00	0,00	0,00	0,00	0,00	0,00
ARIMAX (p=1, d=0, q=2)	0,00	0,00	0,00	/	/	/
ARIMAX (p=2, d=0, q=1)	0,00	0,00	0,00	/	/	/
ARIMA (p=1, d=0, q=2)	0,00	0,00	0,00	/	/	/
ARIMA (p=2, d=0, q=1)	0,00	0,00	0,00	/	/	/
♦neural network (VW) ($L2\ \alpha = 0.001, \eta_0 = 0.5, arch = (24)$)	/	/	/	**1,00**	**1,00**	**1,00**
neural network (VW) ($L2\ \alpha = 0.001, \eta_0 = 0.75, arch = (50)$)	/	/	/	0,00	0,00	0,00
neural network (VW) ($L2\ \alpha = 0.01, \eta_0 = 0.1, arch = (50)$)	/	/	/	**0,0108**	0,00	0,00
neural network (VW) ($L2\ \alpha = 0.01, \eta_0 = 0.5, arch = (50)$)	/	/	/	0,00	0,00	0,00

Primerjave so vidne v tabeli 4.7, kjer so za vsak par primerjanih algoritmov napisane p-vrednosti. Pri tem smo uporabili mero uspešnosti MAE (enačba (3.4)). S krepko pisavo so označeni primeri, kjer nismo uspešno zavrnili ničelne hipoteze pri upoštevanju Bonferronijeve korekcije. Opazimo, da samo v enem primeru (če ne upoštevamo primerjav enakega algoritma z enakim) ne zavržemo ničelne hipoteze. Hipoteza bi v tem primeru bila zavrnjena pri pogoju $|DM| > z_{\alpha_B/2} = 2{,}88$.



Slika 4.11: Primerjava parov algoritmov s statistiko Q pri paketnem pristopu modeliranja na elektrarni z ID 168498. Algoritem random forest ($n = 40$) primerjamo s petimi najuspešnejšimi algoritmi pri modeliranju v omenjenem pristopu.

Graf 4.11 prikazuje statistiko Q pri paketnem pristopu modeliranja na elektrarni z ID 168498. Razvidno je, da je v začetnem času algoritem SGDRegressor ($L2\ \alpha = 0.1, \eta_0 = 0.01$) uspešnejši od algoritma Random forest ($n = 40$), saj je $Q(A, B) > 0$.

4.4 Časovna zahtevnost

Zanimal nas je tudi red velikosti časa učenja uporabljenih algoritmov strojnega učenja na podatkih za celotno obdobje (torej od začetka 2011 do konca 2014) ene elektrarne. Ker smo v posameznem pristopu modelirali z več algoritmi hkrati, težko ugotovimo čas učenja za vsakega posebej. Zato smo

se osredotočili na en paketni in en sprotni algoritem strojnega učenja (brez algoritmov ADWIN ali rezervoarskega vzorčenja), s katerima smo dobili najboljše rezultate — naključni gozdovi in umetne nevronske mreže iz paketa Vowpal Wabbit. Za sprotni algoritem smo izmerili konstanten čas učenja: za vsak nov primer iz podatkovnega toka je bilo potrebnih ∼0,5 sekunde za posodobitev modela. Čas učenja paketnega algoritma (na podatkih iz celotnega obdobja) smo izmerili ∼6 minut, čas učenja paketnega algoritma na množici podatkov velikosti 2000 (naša velikost rezervoarja) pa smo izmerili ∼30 sekund.

Poudarimo, da je v tem primeru šlo za učenje ene elektrarne na enem virtualnem procesorju, kjer smo imeli podatke v časovni ločljivosti ene ure. Menimo, da bi bilo vsakodnevno ponovno učenje paketnih algoritmov za 300 in več elektrarn v realnem času, izvedljivo na hitrem več jedrnem računalniku. Množica podatkov bi se v praksi iz dneva v dan povečevala, s tem pa tudi čas izvajanja našega paketnega pristopa. S kombinacijo paketnega pristopa in rezervoarskega vzorčenja bi pridobili konstanten čas učenja. Prav tako z uporabo sprotnih algoritmov. Ker smo s sprotnimi algoritmi pridobili dobre rezultate, menimo, da bi bila uporaba slednjih v primerjavi s paketnimi s stališča časovne zahtevnosti bolj smiselna.

Poglavje 5

Zaključek

5.1 Sklepne ugotovitve

Cilj magistrskega dela je bila uporaba algoritmov za delo s podatkovnimi tokovi in primerjava teh s klasičnimi algoritmi strojnega učenja v domeni napovedovanja proizvodnje električne energije 304-ih sončnih elektrarn na območju primorske Slovenije.

Proizvodnjo elektrarn smo napovedovali na podlagi vremenskih napovedi, pridobljenih iz elektrarni najbližje vremenske točke modela ALADIN in nekaj izpeljanih atributov za 24 ur vnaprej. V tako imenovanem *paketnem* pristopu modeliranja smo na podatkovnih tokovih uporabljali klasične algoritme strojnega učenja. Učno množico smo vsakodnevno povečevali in jo uporabljali za učenje. V drugem pristopu smo uporabljali *sprotne* algoritme, ki se učijo inkrementalno. Oba pristopa smo ekskluzivno dopolnili z dvema metodama vzorčenja, s čimer smo zmanjšali količino podatkov, potrebnih za računanje napovednih modelov. Najprej smo uporabili algoritem ADWIN, ki z uporabo *prilagodljivih drsečih oken* in zaznavanjem *sprememb koncepta* vzdržuje vzorec zadnjih podatkov iz podatkovnega toka. Kot drugo metodo smo uporabili algoritem *rezervoarskega vzorčenja*, s katerim smo poskušali ohraniti reprezentativen vzorec, ki ima enake lastnosti kot podatkovni tok.

Ugotovili smo, da smo se z našimi modeli zelo približali napaki nMAE

0,03. Najboljše rezultate z napako nMAE 0,0337 dobimo z modelom naključni gozdovi in s paketnim načinom modeliranja. Z umetnimi nevronskimi mrežami, ki se lahko učijo sproti, smo v kombinaciji z algoritmom rezervoarskega vzorčenja dobili nMAE 0,0363.

Z algoritmom ADWIN smo v kombinaciji s sprotnim pristopom v povprečju pridobili 1 % boljše rezultate nMAE, kot z istim pristopom brez omenjenega algoritma. Pri paketnem načinu smo z ADWIN v povprečju pridobili 10 % slabše rezultate nMAE, razen pri uporabi časovnih vrst, kjer je ADWIN doprinesel približno 10 % boljše rezultate. Iz velikosti drsečih oken oz. zaznave sprememb koncepta na nekaterih grafih (npr. graf 4.2) lahko sklepamo o vplivu trenutnih letnih časov na napako modelov, ki so bili naučeni na podatkih iz prejšnjega letnega časa. V našem primeru se je algoritem ADWIN izkazal kot dobra izbira samo v kombinaciji s sprotnimi modeli. Algoritem rezervoarskega vzorčenja je sprotnim modelom izboljšal rezultate nMAE v povprečju za 5 %. Pri modelih, ki podpirajo paketni in sprotni pristop modeliranja smo ugotovili, da so modeli v sprotnem pristopu generalno uspešnejši.

Glede na to, da je vreme težko napovedljivo in da naše napovedi proizvodnje električne energije temeljijo na vremenskih napovedih, menimo, da so bili naši modeli uspešni. Poleg tega v kontekstu območja slovenskega električnega sistema naša pridobljena napaka nMAE 0,0337 relativno ni velika, saj sončne elektrarne predstavljajo le $\sim$1 % celotne proizvodnje v Sloveniji [5] in je napovedovanje proizvodnje zanje bistveno zahtevnejše kot npr. za hidroelektrarne.

Naši rezultati so povsem primerljivi z rezultati drugih avtorjev. Za boljšo primerjavo pa ocenjujemo, da bi modele morali testirati na podatkih za enako obdobje, enake lokacije elektrarn in z enakimi kriteriji. V [9, 10] so z umetnimi nevronskimi mrežami pridobili med 5 % in 10 %, z ARIMA modeli pa 3 % napako.

Pri pregledu področja nismo zasledili, da bi pri modeliranju proizvodnje ali porabe električne energije uporabljali algoritme za zaznavanje sprememb koncepta ali algoritme, ki se iz podatkov lahko učijo sproti. Prav tako nismo

zasledili informacije, kako pogosto v praksi posodabljajo modele. Glede na naraščanje števila obnovljivih virov energije oz. količine podatkov menimo, da bo za sprotne algoritme, ki smo jih opisali in uporabljali v tem delu, vse več zanimanja tudi na tem področju. Poleg tega menimo, da bi algoritmi za zaznavanje sprememb koncepta lahko doprinesli k uspešnosti napovedovanja, še posebej v primerih izrednih dogodkov, kot so sončni mrk, okvara (ki bi povzročila pošiljanje napačnih podatkov) ali menjava strojne opreme na sončni elektrarni (npr. solarnega panela) itd.

Naše delo ima lahko velik potencial pri napovedovanju proizvodnje električne energije in s tem k cenejšim obratovalnim stroškom, kar posredno pripomore k prijaznejšemu okolju.

5.2 Možnosti za nadaljnje delo

Kot smo omenili, smo vremenske napovedi za leti 2011 in 2012 pridobili iz stare različice modela ALADIN, podatke za leti 2013 in 2014 pa iz nove oz. posodobljene. Bilo bi smiselno natančneje raziskati, če napovedi iz obeh različic vplivajo na uspešnost napovedovanja proizvodnje električne energije sončnih elektrarn. Metode, ki smo jih uporabljali v tem delu, bi bilo prav tako smiselno testirati na več podatkih oz. na podatkih za leto 2015 in naprej, še posebej zato, ker število napak v podatkih proizvodnje sončnih elektrarn v kasnejših letih upade.

ARSO za zelo kratkoročno (do 12 ur) napovedovanje vremena uporablja tudi model INCA. Model pri svojem izračunu med drugim uporablja tudi bolj podrobno topografijo. Prostorska ločljivost napovedovanja modela je 1 km, napovedi pa se izračunajo vsako uro, zaradi česar bi jih bilo smiselno uporabljati za napovedovanje proizvodnje električne energije za vsako uro. Za namene tega dela smo uporabljali napovedi 88 modelskih točk. Izračunali smo, da eno modelsko točko v povprečju uporablja 2,38 elektrarn. Dobro bi bilo ugotoviti, koliko modelskih točk je sploh smiselno uporabljati za namene napovedovanja oz. raziskati vpliv števila modelskih točk, ki so na voljo, na

uspešnost napovedovanja električne energije. Produkti modelov ALADIN in INCA za podjetja namreč niso brezplačni.

Proizvodnjo električne energije bi lahko napovedovali še z regresijskimi metodami iz programskega paketa *MOA*[1]. Večina algoritmov v omenjenem paketu je sicer za namene klasifikacije, nekaj dodatnih pa jih je primernih tudi za regresijske probleme. V [7, 8] so avtorji leta 2012 kot prvi predstavili regresijska drevesa za sprotno učenje iz spremenljivih podatkovnih tokov, katera bi lahko preizkusili tudi v domeni napovedovanja proizvodnje električne energije.

Omenimo še globoke umetne nevronske mreže (npr. z arhitekturo *LTSM* [84, 85, 86]) in ansambelske metode, kot so naučeno kombiniranje z meta učenjem (angl. *stacking*), povezovanje algoritmov (npr. angl. *gradient boosting*, *adaboost*) itd. Avtor v knjigi [15] opisuje tudi ansambelske metode za uporabo na podatkovnih tokovih.

Preizkusili bi lahko tudi druge metode za zaznavo sprememb koncepta, ki smo jih omenili na koncu poglavja 2.6. V [73] predstavijo novo metodo za zaznavanje spremembe koncepta in jo primerja z algoritmom ADWIN. Rezultati so v prid metodi iz članka. Zaznavanje sprememb koncepta bi lahko preizkusili tudi na npr. neodvisnih spremenljivkah (vremenskih napovedih) oz. na načine, ki so omenjeni v poglavju 2.6.

[1]MOA (Massive Online Analysis) [22] je programski paket algoritmov za podatkovno rudarjenje na podatkovnih tokovih. Napisan je v programskem jeziku Java.

Literatura

[1] D. Copič, B. Rajer, Izzivi pri napovedovanju proizvodnje iz razpršenih virov, v: 12. konferenca slovenskih elektroenergetikov, 2015, dostopno na: `http://www.cigre-cired.si/Images/files/documents/12_konferenca_Portoroz_2015/C5-18_2097.pdf` (pridobljeno 26.1.2018).

[2] K. Sredenšek, Vpliv naklona fotonapetostnih modulov na proizvodnjo električne energije, diplomsko delo, Fakulteta za energetiko, Univerza v Mariboru (2014).

[3] A. Klemenc, Obnovljivi viri energije v Sloveniji - Kratek pregled potencialov, stanja politik in izzivov, v: Regionalni center za okolje, Slovenija, 2015, dostopno na: `http://www.gbc-slovenia.si/wp-content/uploads/2015/10/OBNOVLJIVI-VIRI-V-SLOVENIJI-s-prilogami-1.pdf` (pridobljeno 26.1.2018).

[4] G. Vozelj, Kratkoročno napovedovanje proizvodnje električne energije sončnih elektrarn in malih hidroelektrarn, magistrsko delo, Fakulteta za elektrotehniko, Univerza v Mariboru (2016).

[5] T. Tomažič, Napovedovanje dnevne proizvodnje električne energije sončnih elektrarn, magistrsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani (2016).

[6] Š. Kunstelj, M. Rejc, M. Pantoš, Kratkoročno napovedovanje porabe električne energije po regijah za območje Slovenije, *Elektrotehniški vestnik* **84** (4) (2014) 222–228.

[7] E. Ikonomovska, J. Gama, S. Džeroski, Learning Model Trees from Evolving Data Streams, *Data Min. Knowl. Discov.* **23** (1) (2011) 128–168.

[8] E. Ikonomovska, Algoritmi za učenje regresijskih dreves in ansamblov iz spremenljivih podatkovnih tokov, doktorska disertacija, Inštitut Jožef Stefan, Ljubljana, Slovenija (2012).

[9] J. Kraljič, Napovedovanje podatkovnega toka porabe električne energije, diplomsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani (2011).

[10] J. Demšar, Razlaga napovednih modelov in posameznih napovedi pri inkrementalnem učenju, diplomsko delo, Fakulteta za računalništvo in informatiko, Fakulteta za matematiko in fiziko, Predmet računalništvo in matematika, Univerza v Ljubljani (2012).

[11] G. Černe, Kratkoročno napovedovanje porabe električne energije z uporabo mehkih Takagi-Sugeno modelov, magistrsko delo, Fakulteta za elektrotehniko, Univerza v Mariboru (2016).

[12] J. Vetršek, S. Medved, Prediction of photovoltaic systems production using weather forecasts, *Eurosun: Solar energy for a brighter future* (2012) 14–15.

[13] P. Šparl, D. Kofjač, A. Brezavš, Kvantitativne metode za napovedovanje porabe električne energije: pregled uveljavljenih metod in analiza njihove uporabe v Sloveniji, v: 20. konferenca Dnevi slovenske informatike, 2013, dostopno na: `http://www.meteo-drustvo.si/data/upload/Vetrnica0412_Pod_drobnogledom.pdf` (pridobljeno 26.1.2018).

[14] S. Šmigoc, Primerjava pristopov k napovedovanju porabe električne energije, magistrsko delo, Fakulteta za naravoslovje in matematiko, Univerza v Mariboru (2016).

[15] J. Gama, Knowledge discovery from data streams, CRC Press, Taylor and Francis Group, 2010.

[16] D. Carmona, M. A. Jaramillo, E. Gonzalez, J. A. Alvarez, Electric energy demand forecasting with neural networks, v: IEEE 2002 28th Annual Conference of the Industrial Electronics Society. IECON 02, izd. **3**, 2002, str. 1860–1865.

[17] J. B. Ricardo, J. Gama, M. Vladimiro, Entropy and Correntropy Against Minimum Square Error in Offline and Online Three-Day Ahead Wind Power Forecasting, v: IEEE, izd. **24**, 2009.

[18] M. Ceci, N. Cassavia, R. Corizzo, P. Dicosta, D. Malerba, G. Maria, E. Masciari, C. Pastura, Innovative power operating center management exploiting big data techniques, v: Proceedings of the 18th International Database Engineering; Applications Symposium, IDEAS '14, ACM, 2014, str. 326–329.

[19] P. Rodrigues, J. Gama, A system for analysis and prediction of electricity-load streams, v: Intell. Data Anal., izd. **13**, 2009, str. 477–496.

[20] J. Gama, P. P. Rodrigues, Stream-Based Electricity Load Forecast, v: J. N. Kok, J. Koronacki, R. Lopez de Mantaras, S. Matwin, D. Mladenič, A. Skowron (Eds.), Knowledge Discovery in Databases: PKDD 2007: 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, Warsaw, Poland, September 17-21, 2007. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, str. 446–453.

[21] A. Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques for Building Intelligent Systems, O'Reilly Media, 2017.

[22] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, Data Stream Mining: A Practical Approach, Citeseer, 2011.

[23] M. R. Šikonja, I. Kononenko, Inteligentni sistemi, 1. izd., Fakulteta za računalništvo in informatiko, Ljubljana, 2010.

[24] Fang Chu, C. Zaniolo, Fast and Light Boosting for Adaptive Mining of Data Streams, v: Advances in Knowledge Discovery and Data Mining, Springer Berlin Heidelberg, 2004, str. 282–292.

[25] Haixun Wang, Wei Fan, Philip S. Yu, Jiawei Han, Mining concept drifting data streams using ensemble classifiers, v: International Conference on Knowledge Discovery and Data Mining, 2003, str. 226–235.

[26] A. Bifet, R. Gavaldà, Learning from Time-Changing Data with Adaptive Windowing, v: 2007 SIAM International Conference on Data Mining (SDM'07), 2007, str. 443–448.

[27] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.), Advances in Knowledge Discovery and Data Mining, American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996.

[28] R. Sutton, S. D. Whitehead, Online Learning with Random Representations, *Proceedings of the Tenth Int. Conf. on Machine Learning* (1995) 314–321.

[29] Y. LeCun, L. Bottou, G. B. Orr, K.-R. Muller, Efficient backprop, v: Neural Networks: Tricks of the Trade, izd. **7700** of Lecture Notes in Computer Science, 2012, str. 9–48.

[30] S. Ross, P. Mineiro, J. Langford, Normalized online learning, v: Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, UAI 2013, Bellevue, WA, USA, August 11-15, 2013.

[31] D. Bollegala, Dynamic feature scaling for online learning of binary classifiers, *Knowl.-Based Syst.* **129** (2017) 97–105.

[32] C. B. Do, Q. V. Le, C.-S. Foo, Proximal Regularization for Online and Batch Learning, v: Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, ACM, New York, NY, USA, 2009, str. 257–264.

[33] N. Karampatziakis, J. Langford, Importance weight aware gradient updates, *CoRR* abs/1011.1576.

[34] J. Langford, L. Li, T. Zhang, Sparse Online Learning via Truncated Gradient, *J. Mach. Learn. Res.* **10** (2009) 777–801.

[35] P. Domingos, G. Hulten, Mining High-speed Data Streams, v: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '00, ACM, New York, NY, USA, 2000, str. 71–80.

[36] G. Hulten, L. Spencer, P. Domingos, Mining Time-changing Data Streams, v: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '01, ACM, New York, NY, USA, 2001, str. 97–106.

[37] J. Gama, R. Rocha, P. Medas, Accurate Decision Trees for Mining High-speed Data Streams, v: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '03, ACM, New York, NY, USA, 2003, str. 523–528.

[38] A. P. Dawid, Statistical Theory: The Prequential Approach, *Journal of the Royal Statistical Society* **147** (2) (1984) 278–292.

[39] J. Gama, S. Raquel, P. P. Rodrigues, On Evaluating Stream Learnign Algorithms, *Journal Machine Learning* **90** (3) (2013) 317–346.

[40] R. H. Byrd, P. Lu, J. Nocedal, A Limited Memory Algorithm for Bound Constrained Optimization, *SIAM Journal on Scientific and Statistical Computing* (1995) 1190–1208.

[41] D. P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, *CoRR* abs/1412.6980.

[42] M. D. Zeiler, ADADELTA: An Adaptive Learning Rate Method, *CoRR* abs/1212.5701.

[43] L. Bottou, F. E. Curtis, J. Nocedal, Optimization Methods for Large-Scale Machine Learning, *CoRR* abs/1606.04838.

[44] N. N. Schraudolph, Local gain adaptation in stochastic gradient descent, v: 1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470), izd. **2**, 1999, str. 569–574.

[45] L. Bottou, On-line Learning in Neural Networks, Cambridge University Press, New York, NY, USA, 1998, str. 9–42.

[46] L. Bottou, Large-scale machine learning with stochastic gradient descent, v: Y. Lechevallier, G. Saporta (Eds.), Proceedings of COMPSTAT'2010, Physica-Verlag HD, 2010, str. 177–186.

[47] W. Enders, Applied Econometric Time Series, Wiley Series in Probability and Statistics - Applied Probability and Statistics Section Series, J. Wiley, 2004.

[48] S. Gričar, Analiza dinamike cen v gostinstvu, doktorska disertacija, Fakulteta za management, Univerza na Primorskem (2012).

[49] B. Bizjak, J. Bizjak, J. Voršič, Primeri kratkoročnega in dolgoročnega napovedovanja pretokov moči, v: 10. konferenca slovenskih elektroenergetikov, 2011, dostopno na: `http://www.cigre-cired.si/Images/files/documents/10_konferenca_Ljubljana_2011/2011-CIGRESKC4-06.pdf` (pridobljeno 26.1.2018).

[50] G. E. P. Box, G. Jenkins, Time Series Analysis, Forecasting and Control, Holden-Day, Incorporated, 1990.

[51] A. Andreoni, M. N. Postorino, Time Series Models to Forecast Air Transport Demand: A Study About a Regional Airport, *IFAC Proceedings Volumes* **39** (12) (2006) 101–106, 11th IFAC Symposium on Control in Transportation Systems.

[52] B. H. Andrews, M. D. Dean, R. Swain, C. Cole, Building ARIMA and ARIMAX Models for Predicting Long-Term Disability Benefit Application Rates in the Public/Private Sectors, Master's thesis, University of Southern Maine (Avgust 2013).

[53] I. Žliobaite, Adaptive training set formation, doktorska disertacija, Vilnius University (2010).

[54] M. G. Kelly, D. J. Hand, N. M. Adams, The Impact of Changing Populations on Classifier Performance, v: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '99, ACM, 1999, str. 367–371.

[55] D. Brzezinski, Mining data streams with concept drift, magistrsko delo, Poznan University of Technology, Faculty of Computing Science and Management, Institute of Computing Science (2010).

[56] R. Sebastião, J. Gama, A Study on Change Detection Methods, v: Progress in Artificial Intelligence, 14th Portuguese Conference on Artificial Intelligence, EPIA 2009, 2009.

[57] E. S. Page, Continuous Inspection Schemes, *Biometrika* **41** (1/2) (1954) 100–115.

[58] H. Mouss, D. Mouss, N. Mouss, L. Sefouhi, Test of Page-Hinckley, an approach for fault detection in an agro-alimentary production system, v: 2004 5th Asian Control Conference, izd. **2**, 2004, str. 815–818.

[59] G. Widmer, M. Kubat, Learning in the Presence of Concept Drift and Hidden Contexts, *Machine Learning* **23** (1) (1996) 69–101.

[60] M. Baena-Garcia, J. Campo-Avila, R. Fidalgo-Merino, A. Bifet, R. Gavaldà, R. Bueno, Early Drift Detection Method, *IWKDDS* (2006) 77–86.

[61] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with Drift Detection **8** (2004) 286–295.

[62] J. S. Vitter, Random Sampling with a Reservoir, *ACM Trans. Math. Softw.* **11** (1) (1985) 37–57.

[63] C. C. Aggarwal, On biased reservoir sampling in the presence of stream evolution, v: Proceedings of the 32Nd International Conference on Very Large Data Bases, VLDB '06, VLDB Endowment, 2006, str. 607–618.

[64] G. Cormode, V. Shkapenyuk, D. Srivastava, B. Xu, Forward Decay: A Practical Time Decay Model for Streaming Systems, v: 2009 IEEE 25th International Conference on Data Engineering, 2009, str. 138–149.

[65] M. Datar, A. Gionis, P. Indyk, R. Motwani, Maintaining Stream Statistics over Sliding Windows, Tech. rep., Department of Computer Science, Stanford University (Julij 2001).

[66] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, R. Wirth, CRISP-DM 1.0 Step-by-step data mining guide, dostopno na:`http://www.crisp-dm.org/CRISPWP-0800.pdf`(pridobljeno 26.1.2018).

[67] H. M. Service, ALADIN model, dostopno na: `http://owww.met.hu/en/hmshp.php?almenu_id=homepages&pid=numprog&pri=3&mpx=0` (pridobljeno 26.1.2018).

[68] N. Pristov, J. Cedilnik, ALADIN in RC LACE – predstavitev, nastanek in kratka zgodovina mednarodnega sodelovanja, v: Vetrnica, 2012, dostopno na: `http://www.meteo-drustvo.si/data/upload/Vetrnica0412_Pod_drobnogledom.pdf` (pridobljeno 26.1.2018).

[69] F. Wang, Z. Mi, S. Su, H. Zhao, Short-Term Solar Irradiance Forecasting Model Based on Artificial Neural Network Using Statistical Feature Parameters, *Energies* **5** (5) (2012) 1355–1370.

[70] Y. Ren, P. Suganthan, N. Srikanth, Ensemble methods for wind and solar power forecasting—A state-of-the-art review, *Renewable and Sustainable Energy Reviews* **50** (2015) 82–91.

[71] N. Sharma, P. Sharma, D. Irwin, P. Shenoy, Predicting solar generation from weather forecasts using machine learning, v: 2011 IEEE International Conference on Smart Grid Communications (SmartGridComm), 2011, str. 528–533.

[72] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, Y. Singer, Online Passive-Aggressive Algorithms, *J. Mach. Learn. Res.* **7** (2006) 551–585.

[73] S. Sripirakas, R. Pears, Y. S. Koh, One Pass Concept Change Detection for Data Streams, v: Advances in Knowledge Discovery and Data Mining, Springer Berlin Heidelberg, 2013, str. 461–472.

[74] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A Survey on Concept Drift Adaptation, *ACM Comput. Surv.* **46** (4) (2014) 44:1–44:37.

[75] F. X. Diebold, Comparing Predictive Accuracy, Twenty Years Later: A Personal Perspective on the Use and Abuse of Diebold-Mariano Tests, Tech. rep., National Bureau of Economic Research (September 2012).

[76] M. Nelson, Exchange Rates and Fundamentals: Evidence on Long-Horizon Predictability, *American Economic Revie*w **85** (1) (1995) 201–18.

[77] N. R. Swanson, H. White, Forecasting economic time series using flexible versus fixed specification and linear versus nonlinear econometric models, *International Journal of Forecasting* **13** (4) (1997) 439–461.

[78] V. Corradi, N. R. Swanson, C. Olivetti, Predictive ability with cointegrated variables, *Journal of Econometrics* **104** (2) (2001) 315–358.

[79] P. R. Hansen, A. Lunde, A forecast comparison of volatility models: does anything beat a GARCH(1,1)?, *Journal of Applied Econometrics* **20** (7) (2005) 873–889.

[80] S. L. Salzberg, U. Fayyad, On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach, *Data Mining and Knowledge Discovery* (1997) 317–328.

[81] J. a. Gama, R. Sebastião, P. P. Rodrigues, Issues in Evaluation of Stream Learning Algorithms, v: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, ACM, New York, NY, USA, 2009, str. 329–338.

[82] R. Urraca, J. Antonanzas, M. Alía Martínez, F. J. Ascacibar, F. Antonanzas, Smart baseline models for solar irradiation forecasting **108** (2016) 539–548.

[83] A. Bifet, R. Gavaldà, Adaptive Learning from Evolving Data Streams, v: Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII, IDA '09, Springer-Verlag, Berlin, Heidelberg, 2009, str. 249–260.

[84] L. Xiangang, W. Xihong, Constructing Long Short-Term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition, *CoRR* abs/1410.4281.

[85] K. Amarasinghe, D. L. Marino, M. Manic, Deep neural networks for energy load forecasting, v: 2017 IEEE 26th International Symposium on Industrial Electronics (ISIE), 2017, str. 1483–1488.

[86] H. Zheng, J. Yuan, L. Chen, Short-Term Load Forecasting Using EMD-LSTM Neural Networks with a Xgboost Algorithm for Feature Importance Evaluation **10** (2017) 1168.

Dodatek A

Podrobnejši rezultati napovedi

Tabela A.1 prikazuje nMBE (povprečno napako) modelov pri vseh elektrarnah. Opazimo, da v povprečju napovedujemo okoli 0,5 % preveliko proizvodnjo glede na skupno maksimalno moč elektrarn.

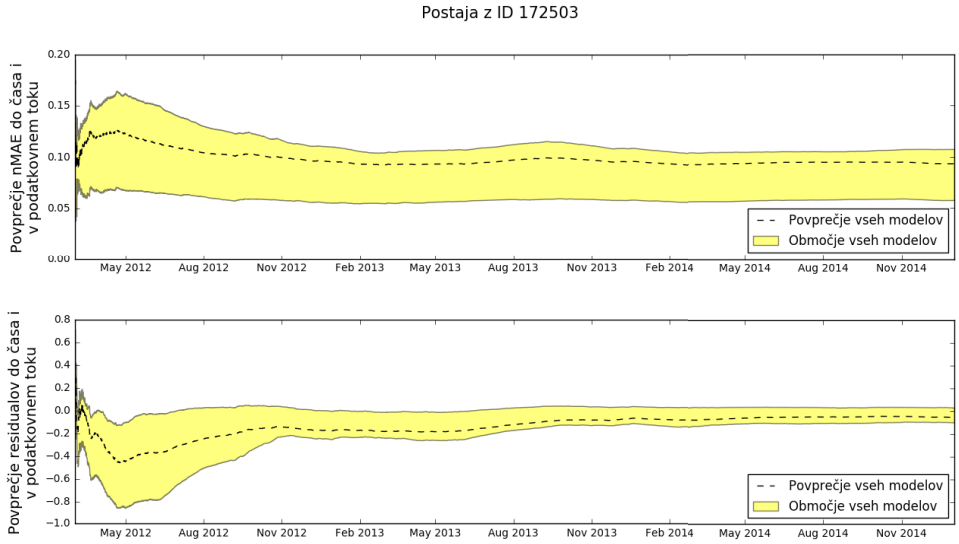
Na grafu A.1 vidimo, da modeli pri paketnem načinu modeliranja v začetnem času v povprečju napovedujejo preveliko, na grafu A.2, s paketnim pristopom in algoritmom ADWIN, pa premajhno proizvodnjo.

Graf A.3 prikazuje primer modeliranja na elektrarni z ID 168498, pri katerem različni modeli dosegajo bistveno drugačno uspešnost — skoraj še enkrat višjo/nižjo.

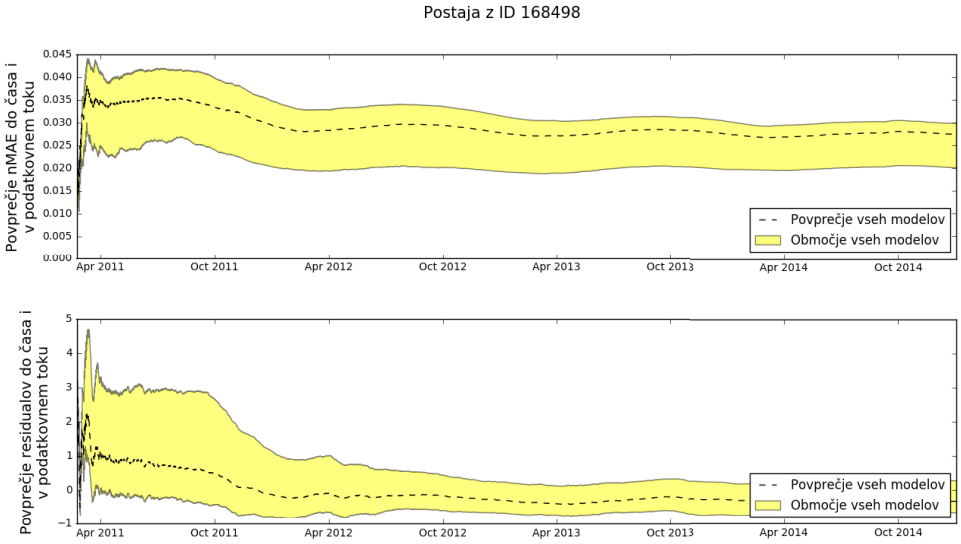
Na grafih A.6, A.7, A.8, A.9 in A.10 je prikazana primerjava parov algoritmov s statistiko Q za elektrarno z ID 168498. Grafi A.11, A.12, A.13 in A.14 pa prikazujejo primerjave algoritmov na vseh elektrarnah. Podrobnosti so napisane v opisih grafov.

Tabela A.1: Povprečen nMBE modeliranj vseh elektrarn z različnimi modeli in pristopi.

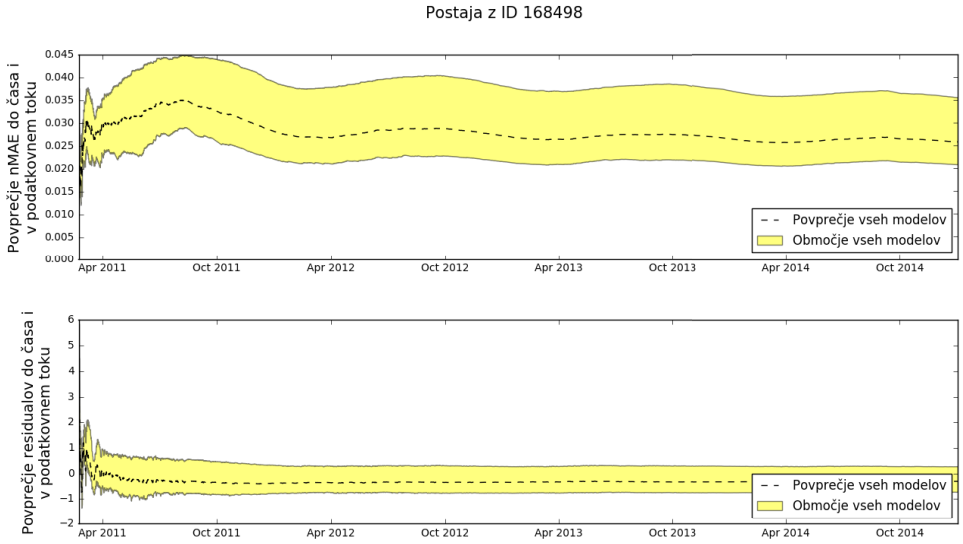
Model	Paketni	Paketni z ADWIN	Paketni z rez. vzorč.	Sprotni	Sprotni z ADWIN	Sprotni z rez. vzorč.
Random forest ($n = 40$)	-0,00187	-0,00061	0,00017	/	/	/
MLPRegressor ($L2\ \alpha = 0.001, SGD, arch = (24, 24), \eta_0 = 0.00001$)	-0,00622	-0,00655	-0,00591	-0,00528	-0,00496	-0,0047
MLPRegressor ($L2\ \alpha = 0.001, SGD, arch = (50, 50), \eta_0 = 0.00001$)	-0,0065	-0.00672	-0,005845	-0,00529	-0,00495	-000468
MLPRegressor ($L2\ \alpha = 0.01, SGD, arch = (50, 50), \eta_0 = 0.00001$)	-0,006	-0,00625	-0,00677	-0,0052947	-0,004926	-0,00469
PassiveAggressive Regressor ($\epsilon = 0.001, C = 0.1$)	-0,00277	0,00589	-0,0004	-0,002719	-0,00238	-0,00236
PassiveAggressive Regressor ($\epsilon = 0.001, C = 0.01$)	-0,00201	0,0037	-0,00059	-0,002445	-0,0021	-0,001474
SGDRegressor ($L2\ \alpha = 0.01, \eta_0 = 0.01$)	-0,00673	-0,00585	-0,00566	-0,002691	-0,00312	-0,00327
SGDRegressor ($L2\ \alpha = 0.1, \eta_0 = 0.01$)	-0,00096	0,0041	-0,00579	0,0033	0,003015	0,00279
ARIMAX (p=1, d=0, q=2)	-0,00662	-0,0051	-0,0082	/	/	/
ARIMAX (p=2, d=0, q=1)	-0,0104	-0,01	-0,0216	/	/	/
ARIMA (p=1, d=0, q=2)	0,0233	0,0171	0,022	/	/	/
ARIMA (p=2, d=0, q=1)	-0,0615	-0,0401	-0,0619	/	/	/
neural network (VW) ($L2\ \alpha = 0.001, \eta_0 = 0.5, arch = (24)$)	/	/	/	0,00172	0,001817	0,00175
neural network (VW) ($L2\ \alpha = 0.001, \eta_0 = 0.75, arch = (50)$)	/	/	/	0,0016	0,000883	0,00026
neural network (VW) ($L2\ \alpha = 0.01, \eta_0 = 0.1, arch = (50)$)	/	/	/	0,00833	0,006192	0,00492
neural network (VW) ($L2\ \alpha = 0.01, \eta_0 = 0.5, arch = (50)$)	/	/	/	-0,0043	-0,00528	-0,00594



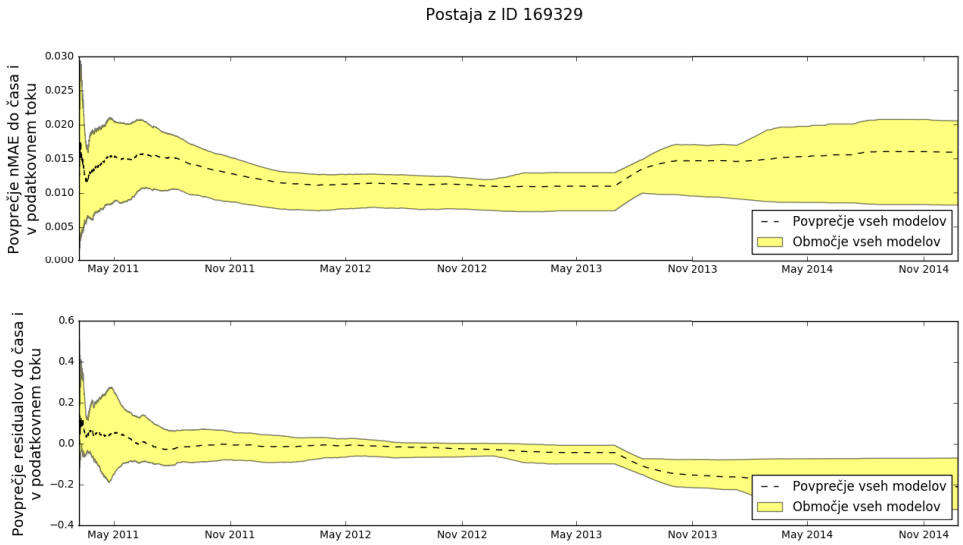
Slika A.1: Graf območja nMAE in residualov vseh modelov skozi čas za postajo z ID 163210 pri paketnem načinu modeliranja.



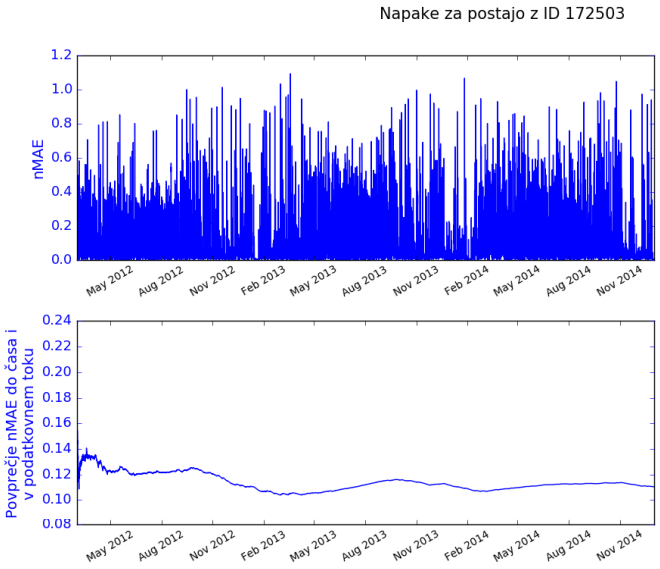
Slika A.2: Primer območja nMAE in residualov vseh modelov skozi čas za postajo z ID 168498 pri paketnem načinu modeliranja z ADWIN, kjer je povprečen nMAE relativno nizek oz. 0,02.



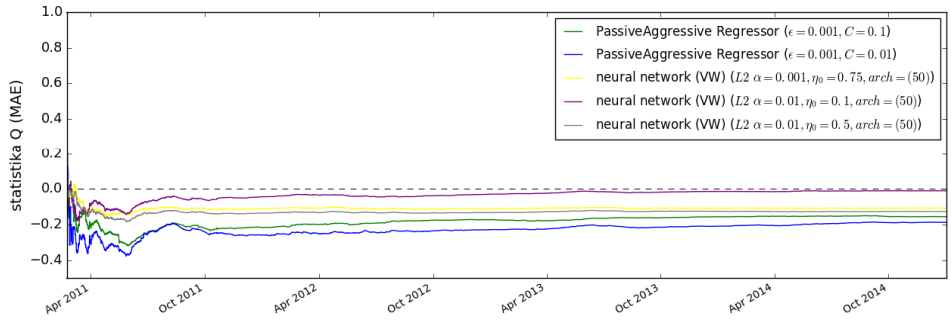
Slika A.3: Primer območja nMAE in residualov vseh modelov skozi čas za postajo z ID 168498 pri sprotnem načinu modeliranja, kjer je povprečen nMAE relativno nizek oz. 0,021.



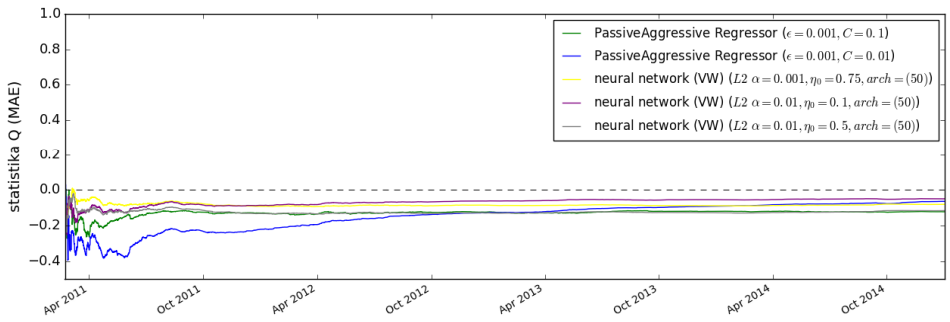
Slika A.4: Graf območja nMAE in residualov vseh modelov skozi čas za elektrarno z ID 168498 pri paketnem načinu modeliranja.



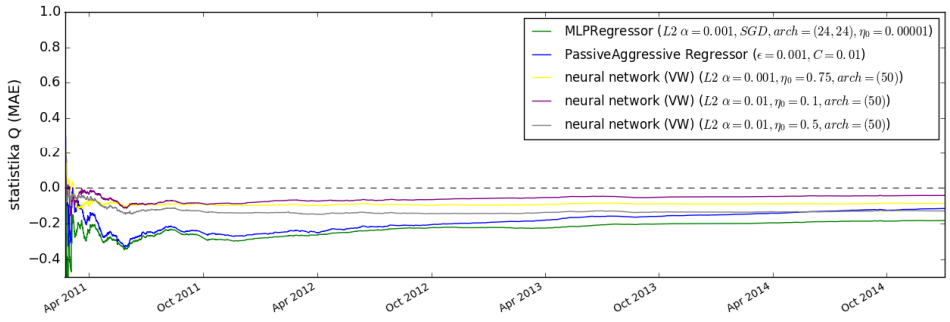
Slika A.5: Primer napak skozi čas za postajo z ID 172503 pri sprotnem načinu modeliranja z rezervoarskim vzorčenjem, kjer je povprečen nMAE relativno visok oz. 0,101. Rezultati so izračunani iz napovedi modela SGDRegressor ($L2\ \alpha = 0.1, \eta_0 = 0.01$). V rezervoarju je prišlo do 14084 zamenjav.



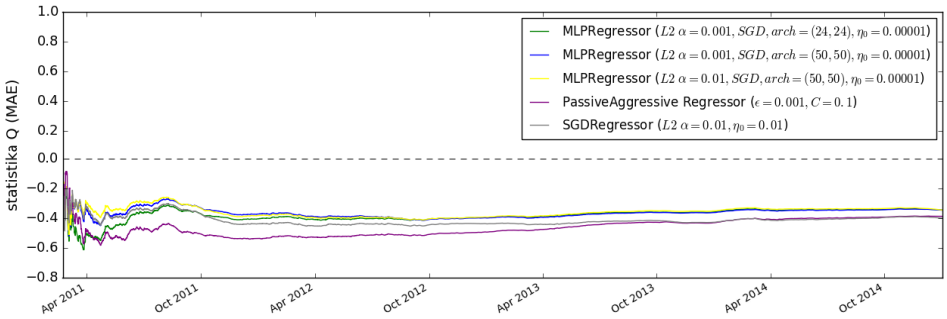
Slika A.6: Primerjava parov algoritmov s statistiko Q pri sprotnem pristopu modeliranja na elektrarni z ID 168498. Algoritem neural network (VW) ($L2\ \alpha = 0.001, \eta_0 = 0.5, arch = (24)$) primerjamo s petimi najuspešnejšimi algoritmi pri modeliranju v omenjenem pristopu.



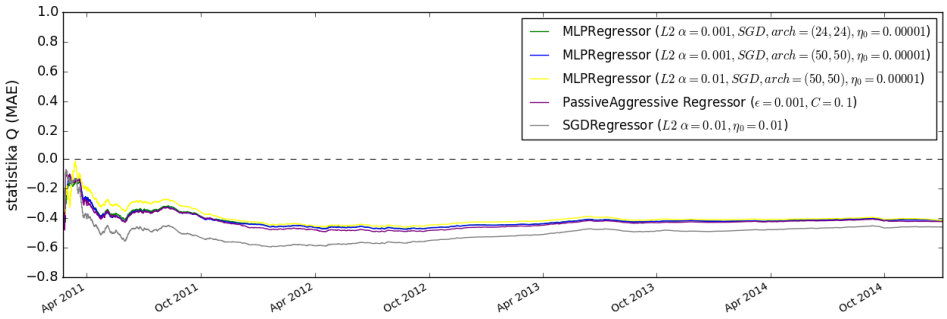
Slika A.7: Primerjava parov algoritmov s statistiko Q pri sprotnem pristopu modeliranja z rezervoarskim vzorčenjem na elektrarni z ID 168498. Algoritem neural network (VW) ($L2\ \alpha = 0.001, \eta_0 = 0.5, arch = (24)$) primerjamo s petimi najuspešnejšimi algoritmi pri modeliranju v omenjenem pristopu.



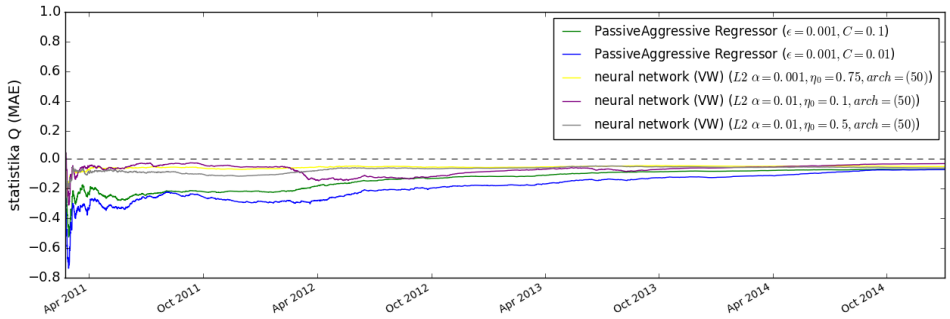
Slika A.8: Primerjava parov algoritmov s statistiko Q pri sprotnem pristopu modeliranja z algoritmom ADWIN na elektrarni z ID 168498. Algoritem neural network (VW) ($L2\ \alpha = 0.001, \eta_0 = 0.5, arch = (24)$) primerjamo s petimi najuspešnejšimi algoritmi pri modeliranju v omenjenem prostopu.



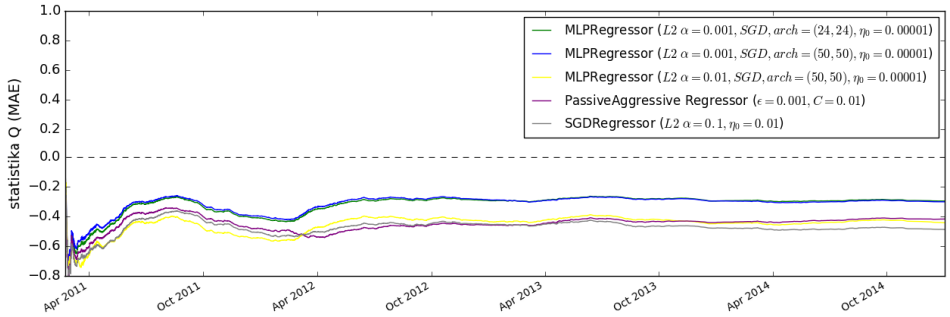
Slika A.9: Primerjava parov algoritmov s statistiko Q pri paketnem pristopu modeliranja z algoritmom ADWIN na elektrarni z ID 168498. Algoritem random forest ($n = 40$) primerjamo s petimi najuspešnejšimi algoritmi pri modeliranju v omenjenem pristopu.



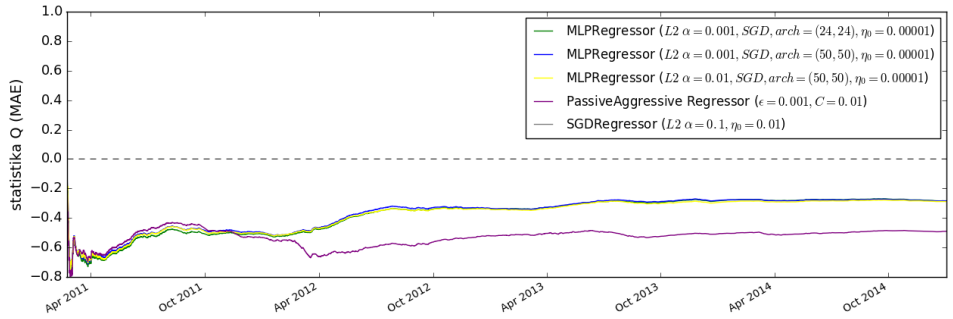
Slika A.10: Primerjava parov algoritmov s statistiko Q pri paketnem pristopu modeliranja z rezervoarskim vzorčenjem na elektrarni z ID 168498. Algoritem random forest ($n = 40$) primerjamo s petimi najuspešnejšimi algoritmi pri modeliranju v omenjenem pristopu.



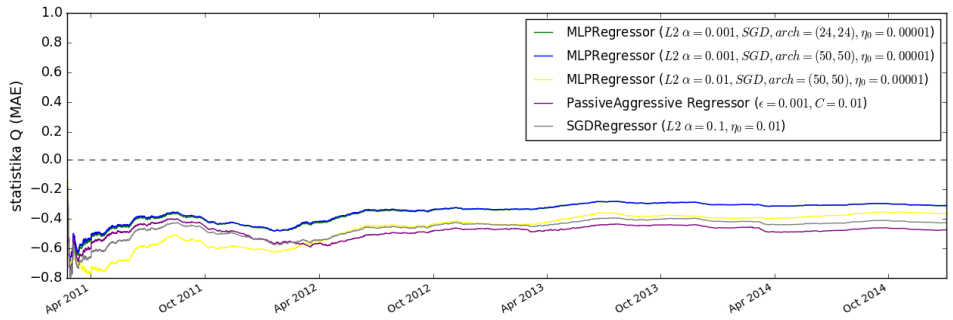
Slika A.11: Primerjava parov algoritmov s statistiko Q pri sprotnem pristopu modeliranja z ADWIN na vseh elektrarnah. Algoritem neural network (VW) ($L2\ \alpha = 0.001, \eta_0 = 0.5, arch = (24)$) primerjamo s petimi najuspešnejšimi algoritmi pri modeliranju v omenjenem pristopu.



Slika A.12: Primerjava parov algoritmov s statistiko Q pri paketnem pristopu modeliranja na vseh elektrarnah. Algoritem random forest ($n = 40$) primerjamo s petimi najuspešnejšimi algoritmi pri modeliranju v omenjenem pristopu.



Slika A.13: Primerjava parov algoritmov s statistiko Q pri paketnem pristopu modeliranja z ADWIN na vseh elektrarnah. Algoritem random forest ($n = 40$) primerjamo s petimi najuspešnejšimi algoritmi pri modeliranju v omenjenem pristopu.



Slika A.14: Primerjava parov algoritmov s statistiko Q pri paketnem pristopu modeliranja z rezervoarskim vzorčenjem na vseh elektrarnah. Algoritem random forest ($n = 40$) primerjamo s petimi najuspešnejšimi algoritmi pri modeliranju v omenjenem pristopu.