Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Grega Kešpret

Ocenjevanje zanesljivosti napovedi pri regresijskem napovedovanju iz podatkovnih tokov

diplomsko delo
na univerzitetnem študiju

Mentor: doc. dr. Zoran Bosnić

Ljubljana, 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01826/2012
Datum: 02.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **GREGA KEŠPRET**

Naslov: **OCENJEVANJE ZANESLJIVOSTI NAPOVEDI PRI REGRESIJSKEM NAPOVEDOVANJU IZ PODATKOVNIH TOKOV**

ESTIMATION OF PREDICTION RELIABILITIES IN REGRESSION MODELLING OF DATA STREAMS

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V strojnem učenju predstavlja modeliranje iz podatkovnih tokov številne izzive: porazdelitev podatkov se spreminja, omejeni smo s procesorskim časom in spominom, problematična pa je tudi količina učnih podatkov. Iz navedenih razlogov izhaja večja motivacija po korekcijskem mehanizmu napovedi kot pa po vnovični gradnji in večkratnem testiranju različnih napovednih modelov s ciljem izbire optimalnega.
Kandidat naj v diplomskem delu izbere podatke primernega podatkovnega toka za reševanje regresijskega problema. Na njem naj oceni uspešnost učenja različnih regresijskih modelov in naj primerja njihove osnovne točnosti s točnostmi istih modelov, ki uporabljajo korekcijo napovedi z oceno zanesljivosti SAbias-s (Bosnić & Kononenko, 2008).

Mentor:

doc. dr. Zoran Bosnić

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
UNIVERZA V LJUBLJANI
1

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Grega Kešpret, z vpisno številko **63060113**, sem avtor diplomskega dela z naslovom:

Ocenjevanje zanesljivosti napovedi pri regresijskem napovedovanju iz podatkovnih tokov

S svojim podpisom zagotavljam, da:

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

V Ljubljani, dne 4. julij 2012 Podpis avtorja:

Zahvala

Za vso podporo in motivacijo v času študija, pa tudi skrb in neizmerno zaupanje se zahvaljujem mami Jasni in očetu Danilu. Hvala tebi, brat Matija, od katerega sem se v življenju veliko naučil. Hvala tudi moji najdražji Katji, ker verjameš vame in mi stojiš ob strani.

Posebna zahvala gre tudi *EU-Japan Centre for Industrial Cooperation* in podjetju *SANYO Electric* za edinstveno priložnost enoletnega življenja na Japonskem in opravljanja prakse. どうもありがとうございました！

Na koncu pa bi se rad zahvalil mentorju doc. dr. Zoranu Bosniću za vse nasvete in strokovno pomoč v času pisanja diplomske naloge, predvsem pa odgovore na moja dolga elektronska sporočila tudi ob vikendih in poznih urah.

„Če moraš napovedovati, napoveduj pogosto.“

–Edgar R. Fiedler

Kazalo

Povzetek

Ključna omejitev pri problemih tradicionalnega strojnega učenja je navadno velikost vzorca in ne viri računske moči. Dandanes viri podatkovnih tokov nepretrgano generirajo ogromne količine podatkov iz nestacionarnih porazdelitev, kar predstavlja odmik od klasičnega modeliranja podatkov. Upoštevati moramo določene omejitve, kot npr. končna velikost pomnilnika pri potencialno neskončni količini podatkov, morebitna majhna računska moč vozlišč, nenadne spremembe v procesu generiranja, zmožnost obdelave v realnem času in druge, ki zahtevajo nove, inkrementalne pristope.

V diplomskem delu se ukvarjamo z izdelavo in vrednotenjem napovednega sistema za porabo električne energije, ki temelji na principu podatkovnih tokov. Najprej smo izdelali metodo za odkrivanje in nadomeščanje anomalij v podatkih, nato pa izdelamo in vrednotimo 8 različnih napovednih modelov glede na napovedno točnost, ki jo dosežejo na realnih podatkih. Poleg tega smo raziskali tudi ocenjevanje zanesljivosti napovedi in popravljanje napovedanih vrednosti na podlagi teh ocen. Na koncu predstavimo eksperimentalne rezultate na resničnih podatkih 11 podatkovnih tokov različnih območij zvezne države New York v ZDA in komentiramo smotrnost uporabe ocen zanesljivosti CNK in SAbias za popravljanje prvotnih napovedi.

Ključne besede:

strojno učenje, ocene zanesljivosti, zanesljivost, napaka napovedi, podatkovni tok, regresija, napovedovanje

Abstract

In traditional problems of machine learning, usually the key restriction is the size of sample and not so much computational power. Nowadays, data stream sources continuously generate huge amounts of data from non-stationary distributions, so modelling the data in traditional ways is becoming obsolete. There are certain restrictions like finite size of memory with potentially unlimited amount of data, possible low computational power of nodes, sudden changes in generation process, ability to handle data in real-time and others, which require new, incremental approaches.

In this thesis we develop and evaluate prediction system of electricity consumption based on data streams ideas. First we developed method for detecting and correcting anomalies in the data, and then implemented and evaluated 8 different prediction models based on their prediction accuracy on real data. Besides that, we also researched prediction reliability estimates and correction of prediction based on those measures. In the end, we present experimental results that were obtained using real data of 11 data streams of different areas of New York state in the USA. We also discuss the feasibility of using reliability estimates CNK and SAbias to correct initial predictions.

Keywords:

machine learning, reliability estimates, reliability, prediction error, data stream, regression, prediction

Poglavje 1

Uvod

V preteklosti je bilo potrebno ustvariti algoritme za napovedovanje na majhnih količinah podatkov. Iz tega se je razvila znanstvena veja *strojnega učenja*, ki se tradicionalno ukvarja s problemom, kako zgraditi najboljši možni model za dani omejen nabor podatkov. Rezultat takšnega učenja je nato v splošnem *statični* model, ki ga uporabljamo kot pomoč pri sprejemanju odločitev na novih, nam neznanih podatkih. Ključna omejitev pri problemih takega tipa je navadno velikost vzorca in ne viri računske moči.

Če je bilo v preteklosti težko priti do podatkov in so bili nabori podatkov posledično majhni, pa je napredek tehnologije v zadnjih letih poskrbel za podatkovno eksplozijo, saj je podatkov naenkrat več, kot jih zmoremo obdelati. Dandanes pridobivamo podatke iz različnih virov, kot so npr. senzorska omrežja, promet TCP/IP, podatki o klikanju povezav, sledenje GPS, podatki o mobilnih klicih idr. Vsi ti viri podatkov nepretrgano generirajo ogromne količine podatkov iz nestacionarnih porazdelitev.

Modeliranje takšnih *podatkovnih tokov* pa predstavlja odmik od tradicionalnega modeliranja podatkov. V tem primeru se proces skozi čas razvija in spreminja, zato enotni statični model ne zadostuje več. Statični model namreč čez čas ne bo več konsistenten z dejanskim stanjem, prav tako pa ne more reagirati na nenadne spremembe. Poleg tega nabor podatkov iz podatkovnega toka ob omejenem pomnilniškem mediju kmalu postane prevelik, da bi lahko uporabljali klasične metode, ki za svoje delovanje potrebujejo celotno zbirko podatkov. Upoštevati moramo tudi dodatne omejitve, kot so npr. morebitna majhna računska moč vozlišč, nenadne spremembe v procesu generiranja, zmožnost obdelave v realnem času in druge. Vse te nove omejitve predstavljajo izziv za izdelavo novih metod in uporabo novih inkrementalnih pristopov, ki bodo primerni za uporabo v takšnih okoljih. V

nadaljevanju opišemo domeno učenja, na kateri smo gradili napovedne modele.

1.1 Razlaga domene učenja

Sistemska poraba električne energije je naključni nestacionarni proces, sestavljen iz tisočih posameznih komponent. Časovna vrsta porabe električne energije ima zanimive lastnosti kot so: trend, dnevni, tedenski in letni sezonski vzorci, zunanji vplivi in morebitne nelinearnosti, zato napovedovanje porabe električne energije predstavlja zanimiv akademski problem, ki ga preučujejo že vrsto let. Iz uporabnega vidika pa prav napovedovanje porabe električne energije predstavlja osnovni in osrednji proces pri planiranju in upravljanju elektroenergetskih podjetij.

Napovedi predstavljajo velik potencial, saj so številne pomembne odločitve pri upravljanju teh podjetij, kot npr. načrtovanje ustvarjanja in kupovanja električne energije, načrtovanje popravil in planiranje energijskih transakcij odvisne prav od točnosti napovedi. Točnost napovedi ima torej za elektroenergetska podjetja velike ekonomske posledice. Ocenjeno je bilo, da je povečanje napake napovedi le za 1% povečalo enoletne operativne stroške nekega elektroenergetskega pojetja v Združenih kraljestvih za 10 milijonov funtov [2].

Napovedovanje porabe električne energije v grobem delimo na kratkoročne, srednjeročne in dolgoročne napovedi. V diplomskem delu se omejimo na kratkoročne napovedi. Obnašanje sistema je pogojeno s številnimi vplivi, navadno pa pri napovedih takega tipa upoštevamo vhodne parametre: *pretekli podatki* (poraba prejšnje ure, prejšnjega dneva in enakega dneva prejšnjega tedna), *koledarske in sezonske spremenljivke* ter *vremenski vplivi* (temperatura, vlažnost, veter in pokritost z oblaki) [3]. Tipični izhod takšnega sistema je ocenjena povprečna poraba električne energije za vsako uro v dnevu.

V diplomskem delu se ukvarjamo z izdelavo in vrednotenjem *napovednega sistema* za porabo električne energije za 11 različnih mest v zvezni državi New York v ZDA, kjer podatke pridobivamo iz senzorjev, podatki pa so zelo šumni. Izdelamo različne napovedne modele, ki jih nato vrednotimo glede na napovedno točnost, ki jo dosežejo. Pri tem ne stremimo k najboljši možni napovedni točnosti *per se*, temveč upoštevamo v danem kontekstu omejitve učenja iz podatkovnih tokov.

1.2 Pregled vsebine

Diplomsko delo obsega 5 poglavij. V naslednjem poglavju so obširno predstavljene uporabljene metode strojnega učenja, s poudarkom na algoritmih regresijskega napovedovanja. Predstavljene so tudi mere uspešnosti v regresiji, metodologija statističnih testov in metodi ocenjevanja zanesljivosti napovedi CNK in SAbias.

V tretjem poglavju predstavimo problem napovedovanja 11 realnih podatkovnih tokov, ki smo jih pridobili iz javnodostopnih podatkov o porabi električne energije v zvezni državi New York v ZDA in opišemo izzive, ki jih takšen sistem predstavlja.

V četrtem poglavju prikažemo visokonivojsko shemo izdelanega napovednega sistema in opišemo njegove sestavne dele, s poudarkom na odstranjevanju anomalij v podatkih, izbiranju podmnožice atributnega prostora in učenjem parametrov danega modela (npr. topologija nevronske mreže). V tem poglavju prav tako bolj podrobno orišemo algoritma za dva preizkušena pristopa k napovedovanju: inkrementalnega in periodičnega in podamo formulo za popravljanje napovedi s pomočjo ocen zanesljivosti.

V zadnjem poglavju predstavimo in komentiramo rezultate simuliranja napovedovanja za oba pristopa (periodični, inkrementalni) za različne algoritme strojnega učenja na 11 različnih podatkovnih tokovih, poleg originalnih napovedi pa komentiramo tudi rezultate popravljenih napovedi s pomočjo ocen zanesljivosti CNK in SAbias.

Poglavje 2

Pregled uporabljenih metod

2.1 Učenje iz podatkovnih tokov

Klasične metode, znane tudi kot paketni (*batch*) pristopi delujejo pod predpostavko, da imajo na voljo celotni nabor podatkov. Pri tem so običajne metode za ocenjevanje modelov uporaba *prečnega preverjanja* (angl. *cross-validation*) in njegovih izpeljank. Vendar pa v kontekstu podatkovnih tokov, kjer je podatkov lahko praktično neskončno in kjer se porazdelitev generiranja odvisne spremenljivke skozi čas spreminja, prečno preverjanje ni več smotrna izbira.

Metode za učenje iz podatkovnih tokov imajo v nasprotju s klasičnimi metodami določene zahteve, pogojene z omejitvami, naštetimi v poglavju 1. V tabeli 4.1 povzemamo glavne razlike [14]:

	Klasične metode	**Podatkovni tok**
Čas izvajanja algoritma	neomejen	omejen
Število branj posameznega primera	veliko	enkratno
Dovoljena poraba pomnilnika	neomejena	omejena
Točnost napovedi	natančna	približna

Tabela 2.1: Primerjava med klasičnimi metodami strojnega učenja in metodami, prilagojenimi za delo s podatkovnimi tokovi

V zvezi z učenjem iz podatkovnih tokov se omenjajo trije pristopi [14]:

- **periodični pristop** pomeni, da model po določenem času ponovno zgradimo,

- **inkrementalni pristop** predstavlja *posodabljanje* modela vsakič, ko na vhod pridejo novi podatki,
- **odzivni pristop** pa pomeni, da spremljamo spremembe in model ponovno zgradimo takrat, ko ne ustreza več podatkom.

2.1.1 Drseča okna

Včasih ne želimo računati statistik nad celotno zgodovino podatkovnega toka, temveč le nad *nedavno* zgodovino. Takrat uporabimo metode *drsečih oken*, vendar pa mora biti naš pomnilnik dovolj velik, da lahko shrani *vse* elemente v drsečem oknu [14]. Drseča okna so tudi eden izmed načinov *pozabljanja*.

2.2 Modeli za regresijsko napovedovanje

V splošnem lahko metode strojnega učenja delimo glede na tip ciljne spremenljivke, ki jo v danem problemu modeliramo. Če je to spremenljivka diskretnih vrednosti, govorimo o klasifikaciji. Če pa ciljna (napovedovana) spremenljivka zavzema numerične vrednosti, govorimo o **regresiji**. V tej nalogi se omejimo izključno na regresijske napovedne modele, saj je ciljna spremenljivka, ki jo napovedujemo numerično izražena poraba električne energije.

Naloga regresijskega **napovednega modela** je, da pri znanih vrednostih atributov na vhodu določi neznano vrednost odvisne zvezne spremenljivke na izhodu oziroma, da se nauči funkcijo $\hat{y} = f(\vec{x})$. Učni algoritem torej na že videnih primerih izgradi napovedni model, ki ga nato uporabimo za napovedovanje novih, še nevidenih primerov (slika 2.1).

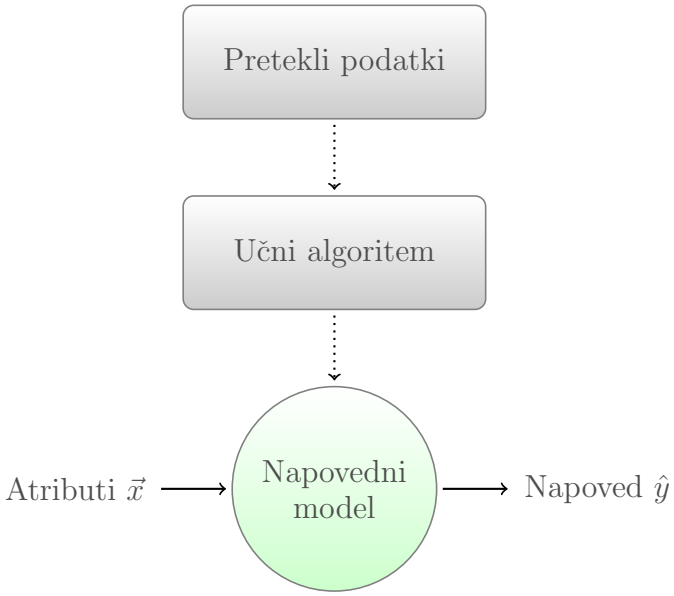
V nadaljevanju so podrobneje opisani regresijski algoritmi, ki smo jih uporabili za testiranja v raziskovalnem delu diplomske naloge.

2.2.1 Linearna regresija z več spremenljivkami

Linearna regresija z več spremenljivkami aproksimira izhodno vrednost y z linearno funkcijo in je definirana kot:

$$\widehat{y}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_m x_m \tag{2.1}$$

V enačbi (2.1) so θ_i neznani **parametri modela**, m število atributov, x_i vhodne spremenljivke in $\widehat{y}(x)$ izhodna vrednost. Pomembno je poudariti, da je izhod



Slika 2.1: Shematični prikaz procesa nadzorovanega učenja in napovedovanja

linearno odvisen od vhodnih spremenljivk, vendar to še ne pomeni, da sama spremenljivka ne more biti preslikana z uporabo nelinearne funkcije (npr. $\log(x)$).

Naloga faze učenja je, da najdemo take parametre θ, ki bodo $\widehat{y}(x)$ čimbolj približali y (pravi vrednosti) vsaj za učno zbirko podatkov.

Iterativni algoritem (paketna izvedba)

Definiramo funkcijo, ki meri za dano kombinacijo vrednosti θ, kako blizu je $\widehat{y}(x^{(i)})$ pravi vrednosti $y^{(i)}$. Definiramo **kriterijsko funkcijo (cost function)**:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(\widehat{y}_\theta(x^{(i)}) - y^{(i)})^2 \tag{2.2}$$

kjer je n število učnih primerov.

Iterativni algoritem ***gradientni spust*** (angl. *batch gradient descent*) najde takšne parametre θ, ki minimizirajo kriterijsko funkcijo (2.2) preko vseh učnih primerov. V algoritmu 2.1 predstavlja n število učnih primerov, m število atributov in α hitrost gradientnega spusta.

Algoritem 2.1: Gradientni spust

repeat
  for *every attribute* $j \leftarrow 1$ **to** m **do**
    $\theta_j \leftarrow \theta_j + \alpha \sum_{i=1}^{n} (y^{(i)} - \widehat{y}_\theta(x^{(i)}))x_j^{(i)}$
  end
until *convergence*

Iterativni algoritem (inkrementalna izvedba)

Algoritem 2.1 mora za en en korak pri spremembi parametrov modela θ pregledati vse učne primere, kar je lahko zelo drago, če je n velik. **Stohastični** ali ***inkrementalni gradientni spust*** (angl. *stochastic gradient descent*) pa deluje inkrementalno in torej lahko začne posodabljati parametre θ takoj, ne šele po pregledu vseh učnih primerov. Po pregledu vsakega učnega primera glede na njegovo napako posodobi parametre θ, zato je bolj primeren za probleme učenja iz podatkovnih tokov.

Algoritem 2.2: Stohastični gradientni spust

for *every example* $i \leftarrow 1$ **to** n **do**
  for *every attribute* $j \leftarrow 1$ **to** m **do**
    $\theta_j \leftarrow \theta_j + \alpha(y^{(i)} - \widehat{y}_\theta(x^{(i)}))x_j^{(i)}$
  end
end

Algoritem 2.2 ima to slabost, da morda nikoli ne "konvergira" do minimuma funkcije (2.2), kar lahko povzroči slabšo napovedno točnost takšnih (inkrementalnih) modelov.

Analitična rešitev

Izkaže se, da za paketno izvedbo iterativnega algoritma obstaja analitična rešitev, ki reši problem minimizacije funkcije (2.2) eksplicitno. Včasih analitični rešitvi pravimo tudi "normalne enačbe". Parametre modela tako lahko izračunamo s pomočjo naslednje formule:

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

kjer je X matrika vhodnih spremenljivk velikosti [n x m], $\vec{y}$ vektor pravih vrednosti velikosti [n x 1] in Θ vektor parametrov modela velikosti [n x 1]. Rešitev normalnih enačb obstaja le v primeru, če je matrika $X^T X$ obrnljiva (njena determinanta mora biti neničelna).

2.2.2 Posplošeni aditivni modeli

Posplošeni aditivni model (angl. *generalized additive model*) sestavljata dva dela: (1) aditivni model in (2) posplošeni linearni model.

Aditivni modeli

Enačba linearne regresije (2.1) v aditivnem modelu postane:

$$\widehat{y}(x) = \theta_0 + f_1(x_1) + f_2(x_2) + ... + f_m(x_m) \tag{2.4}$$

Z drugimi besedami, posamezne konstantne parametre za vsak atribut nadomestimo z (neparametričnimi) funkcijami $f_i(x_i)$.

Posplošeni linearni model

V posplošenem linearnem modelu dovoljujemo povezovanje atributov.

$$\widehat{y}(x) = g\left(\theta_0 + \theta_1(x_1) + \theta_2(x_2) + ... + \theta_m(x_m)\right) \tag{2.5}$$

Inverz funkcije g imenujemo *povezovalna funkcija* (angl. *link function*).

$$g^{-1}(\overline{\widehat{y}(x)}) = \theta_0 + \theta_1(x_1) + \theta_2(x_2) + ... + \theta_m(x_m) \tag{2.6}$$

V enačbi (2.6) predstavlja $\overline{h(x)}$ pričakovano vrednost h(x).

Če združimo vsebino enačb (2.4) in (2.6) dobimo enačbo, ki opisuje posplošene aditivne modele.

$$g^{-1}(\overline{\widehat{y}(x)}) = \sum_{i=1}^{m} \left(f_i(x_i)\right) \tag{2.7}$$

Najpogosteje za funkcije f_i vzamemo *gladilne funkcije* (angl. *scatterplot smoothing functions*) atributov x_i ali kakšno drugo funkcijo iz družine neparametričnih funkcij.

2.2.3 Regresijska drevesa

Linearna regresija je **globalni model**, kjer ena sama napovedna formula velja za cel prostor podatkov. Ko imajo podatki veliko atributov, ki so med seboj odvisni in kjer so te odvisnosti kompleksne in nelinearne, je pogosto težko najti dober globalni model. Alternativni pristop k nelinearni regresiji je, da prostor razdelimo na več manjših podprostorov, kjer so odvisnosti med atributi bolj obvladljive. Ta postopek rekurzivno ponavljamo, dokler niso regije tako majhne, da na podatkih v njih lahko učimo preproste modele (konstanta, linearni model, k-najbližjih sosedov).

Vsako **končno vozlišče** drevesa tako zgrajenega drevesa (list) predstavlja celico v razdeljenem prostoru, vsa nekončna vozlišča predstavljajo atribute, povezave med vozlišči pa vrednosti. Atribute, ki na posameznem nivoju razdelijo prostor, izbiramo na podlagi njihove kvalitete (pogosto npr. kriterij zmanjševanja variance v listu). Prednost regresijskih dreves je navadno, da so zelo razumljiva in hitra, vendar so mnogokrat premalo natančna.

Ko smo regresijsko drevo zgradili, nato v času napovedovanja (slika 2.1) na podlagi vhodnih atributov X potujemo od korena drevesa navzdol po ustreznih povezavah do nekega lista. Vrednost ciljne spremenljivke nato napovemo s funkcijo, ki se nahaja v listu (najpogosteje konstanta - povprečje vrednosti primerov v listu oziroma linearni model).

2.2.4 Naključni gozdovi

Osnovna ideja naključnih gozdov izhaja iz idej učenja ansamblov (*ensemble learning*). Za neko domeno zgradimo več *preprostih modelov* bodisi z spreminjanjem parametrov gradnje bodisi z različnim izbiranjem učnih primerov in na ta način dobimo boljši sestavljeni model.

Naključni gozdovi namreč pri svojem delovanju uporabljajo metodo *bagging* in naključno izbiro atributov za gradnjo zbirke regresijskih dreves s kontrolirano varianco. Tako pri gradnji posameznega regresijskega drevesa na vsakem koraku pri izbiri atribura za delitev izbiramo le med naključno podmnožico vseh atributov. Takšno regresijsko drevo nato pokriva določen segment problemskega podprostora, zato pri računanju končnega rezultata povprečimo napovedi vseh dreves v naključnem gozdu. S tem tudi stabiliziramo varianco in pristranskost končnega modela.

2.2.5 Metoda podpornih vektorjev

Metoda podpornih vektorjev (angl. support vector machines) je prilagojena različica prvotnega algoritma metode podpornih vektorjev, ki deluje na klasifikacijskih problemih. Osnovna ideja klasifikacijske različice metode podpornih vektorjev je poiskati takšno hiperravnino, ki v prostoru prostoru prvotnih (ali spremenjenih) atributov optimalno ločuje razrede ciljne spremenljivke. Optimalna hiperravnina je tista hiperravnina, ki je enako oddaljena od najbližjih primerov različnih razredov. Učne primere, ki so v prostoru najbliže optimalni hiperravnini, imenujemo *podporni vektorji*. Razdalja med hiperravnino in podpornimi vektorji se imenuje *rob*. Optimalno hiperravnino izberemo tako, da maksimiziramo to razdaljo (rob). To predstavlja kvadratični optimizacijski problem, ki ga lahko rešimo z uporabo hitrih algoritmov, kot je npr. SVD [1].

Pri regresijski različici metode podpornih vektorjev, imenovani tudi ε-SVM definiramo problem nekoliko drugače. Primerov, ki padejo znotraj roba, ne upoštevamo pri računanju kriterijske funkcije za optimizacijo modela.

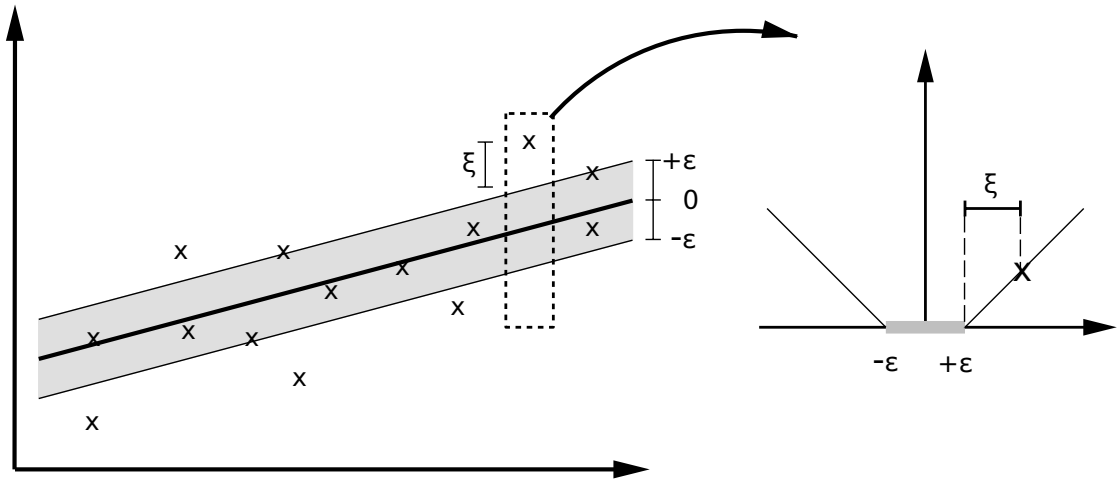
$$|\xi|_\varepsilon = \begin{cases} 0 & |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{sicer.} \end{cases} \tag{2.8}$$

Slika 2.2 [9] prikazuje, da le primeri izven sive regije prispevajo k ceni kriterijske funkcije (2.8). Z drugimi besedami, napake primerov, ki so manjše od ε nas ne zanimajo, vendar pa deviacije, večje kot ε, niso sprejemljive. Problem iskanja regresijske hiperravnine je torej optimizacijski problem kriterijske funkcije, kjer želimo čimbolj zmanjšati napake, ki jih naredi regresijska spremenljivka v primerjavi s pravo vrednostjo.

Poleg uporabe podpornih vektorjev je druga pomembna ideja metode podpornih vektorjev uporaba nelinearnih transformacij – t.i. jedrnih funkcij. S pomočjo jedrnih funkcij (polinomska funkcija, radialna funkcija, sigmoidna funkcija) vrednosti atributov transformiramo iz danega atributnega prostora v kompleksnejši prostor, kjer je lahko optimizacijski problem bolje rešljiv.

2.2.6 Umetne nevronske mreže

Umetne nevronske mreže so formalizem, ki v svojem delovanju z uporabo majhnih neodvisnih gradnikov, imenovanih *nevroni*, posnema lastnosti biološkega živčnega sistema. Nevroni so medsebojno povezani v prepleteno strukturo, imenovano *nevronska mreža*. Najpogosteje so med seboj povezani zaporedno v več plasteh ali



Slika 2.2: Metoda podpornih vektorjev: primeri znotraj roba ne vplivajo na kriterijsko funkcijo

nivojih, kar pomeni, da je vhod nevrona na nivoju $k+1$ lahko le izhod nevrona na nivoju k.

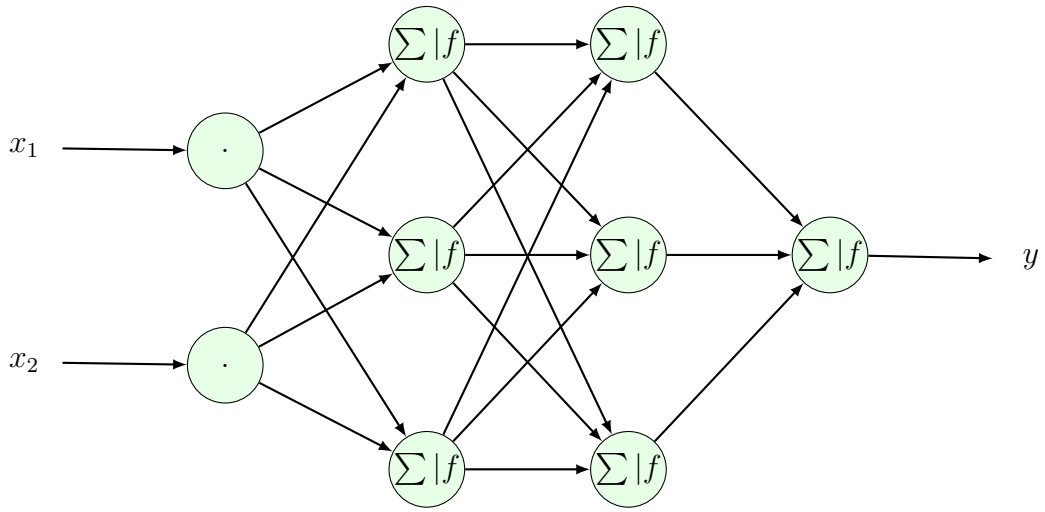
Umetna nevronska mreža ima vhodni nivo, enega ali več skritih nivojev in izhodni nivo, ki najpogosteje vsebuje le en nevron. Število nevronov na vhodnem nivoju je pri regresijskem napovedovanju pogojeno s številom atributov, število skritih nivojev in nevronov v posameznem skritem nivoju je predmet izbire *topologije* nevronske mreže, izhodni nevron pa predstavlja ciljno (odvisno) spremenljivko, ki jo napovedujemo.

Posamezen nevron računa preprosto funkcijo oblike:

$$x_{out} = f\left(\sum_i w_i x_i + w_{bias}\right) \tag{2.9}$$

Funkcijo f v enačbi (2.9) imenujemo *aktivacijska funkcija* in je ponavadi definirana kot neka funkcija iz družine sigmoidnih funkcij. Vidimo torej, da posamezen gradnik v umetni nevronski mreži (nevron) računa uteženo vsoto svojih vhodov, kar v izhod na koncu preslika s pomočjo aktivacijske funkcije. Slika 2.3 prikazuje primer topologije umetne nevronske mreže preden poženemo algoritem učenja.

V fazi učenja mora algoritem nastaviti vrednosti uteži na posameznih povezavah, da se mreža nauči pravilno (čimbolj točno) preslikati vhodne vrednosti atributov v izhodne vrednosti. Eden najbolj razširjenih algoritmov za to se imenuje *algoritem z vzvratnim razširjanjem napake* (angl. *backpropagation*).



Slika 2.3: Umetna nevronska mreža z dvema nevronoma na vhodnem nivoju, dvema skritima nivojema po tri nevrone in enim izhodnim nivojem z enim nevronom

Učenje po tej metodi poteka tako, da so uteži v prvem koraku nastavljene naključno. Mreža nato izračuna izhod za dani primer, ki ga pokažemo na vhodu. Nato nadaljujemo od izhoda mreže proti vhodu tako, da na vsakem nivoju izračunamo razliko do želene vrednosti in ustrezno popravimo uteži tako, da to razliko zmanjšamo. Proces se nadaljuje vse nazaj do prvega skritega nivoja. Ta postopek ponavljamo v več *epohah*, kar lahko naredi algoritem zelo počasen.

2.3 Mere uspešnosti v regresiji

Večina mer uspešnosti v regresiji, ki ocenjujejo določen napovedni model je osnovanih na razliki med pravo in napovedano vrednostjo. Uspešnost preverjamo na podatkih, ki niso bili uporabljeni za učenje modela in pravimo, da je model uspešnejši, če napove vrednost, ki je bližja pravi vrednosti in manj uspešen, če je njegova napoved bolj oddaljena od prave vrednosti [1].

Srednja kvadratna napaka

Srednja kvadratna napaka (angl. *mean squared error*) je definirana kot povprečje kvadratov razlik med napovedanimi vrednostmi $\hat{y}_i$ in resničnimi vrednostmi y_i.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.10}$$

Ponavadi želimo, da je napaka izražena v enakih enotah kot podatki, na katerih je bila izračunana. Takrat izračunamo koren srednje kvadratne napake RMSE.

$$RMSE = \sqrt{MSE} \tag{2.11}$$

Srednja absolutna napaka

Srednja absolutna napaka (angl. *mean absolute error*) je definirana kot povprečje absolutnih vrednosti razlik med napovedanimi vrednostmi $\hat{y}_i$ in resničnimi vrednostmi y_i.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.12}$$

Srednja absolutna napaka v odstotkih

Srednja absolutna napaka v odstotkih (angl. *mean absolute percentage error*) je odstotkovno izražena vrednost povprečja absolutnih vrednosti razlik med napovedanimi vrednostmi $\hat{y}_i$ in resničnimi vrednostmi y_i.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{2.13}$$

Mere MSE, $RMSE$ in MAE imajo to slabost, da je njihova absolutna vrednost odvisna od nabora podatkov in torej vrednosti med seboj niso primerljive, če so bile izračunane na različnih podatkih (z različnimi zalogami vrednosti). To slabost odpravlja mera $MAPE$ (2.13).

Pearsonov korelacijski koeficient

Pearsonov korelacijski koeficient (angl. *Pearson's correlation coefficient*) meri statistično korelacijo (linearno odvisnost) med dvema spremenljivkama X in Y in ima zalogo vrednosti na intervalu $[-1, 1]$.

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{2.14}$$

Uporabimo ga lahko tudi kot mero uspešnosti v regresiji, in sicer tako, da izračunamo korelacijo med napovedanimi vrednostmi $\hat{y}_i$ in resničnimi vrednostmi y_i. Takrat postavimo $X := \hat{y}$ in $Y := y$.

V nasprotju z ostalimi do sedaj opisanimi merami uspešnosti v regresiji želimo, ki morajo biti čimmanjše, želimo da je Pearsonov korelacijski koeficient čimvečji. Večja, kot je korelacija med odvisno spremenljivko in napovedano vrednostjo, bolj uspešen je napovedni model.

Mere uspešnosti v podatkovnih tokovih

Ko ravnamo s podatkovnimi tokovi, ponavadi navadne mere uspešnosti (MSE, RMSE, MAPE itd.) začenjajo izgubljati svoj pomen, saj se ocena napake pri večanju števila primerov ustali in ima torej vsak nov primer vedno manjšo težo pri spreminjanju skupne ocene napake. Vendar pa so opisane mere uspešnosti uporabne, kadar ocenjujemo uspešnost modela pri izračunanih napovedih za izbrano zgodovinsko testno množico, enako kot to počnemo pri neinkrementalnem učenju. Zato v predstavljenih rezultatih za potrebe primerjanja s paketnimi pristopi še vedno računamo korelacijski koeficient, napako RMSE in napako MAPE. Kot dodatni način evalvacije izvedemo tudi evalvacijo s specifičnimi metrikami za podatkovne tokove, ki jih opisujemo v nadaljevanju.

Eden od možnih načinov za merjenje, kako se točnost modela spreminja skozi čas v problemih podatkovnih tokov je uporaba tako imenovanih *bledečih faktorjev* (angl. *fading factors*), ki poudarjajo obnašanje modela na najbolj svežih podatkih [4, 15], na primer *alfa*-bledeča srednja kvadratna napaka (αMSE). Definiramo jo z rekurzivno formulo

$$\begin{aligned} s_i &= (\widehat{y} - y)^2 + \alpha \times s_{i-1} \\ n_i &= 1 + \alpha \times n_{i-1} \\ \alpha MSE_i &= \frac{s_i}{n_i} \end{aligned} \tag{2.15}$$

kjer sta $\widehat{y}$ in y napovedana in prava vrednost in α parameter pozabljanja (utež). Začetne vrednosti v rekurziji nastavimo na $n_0 = 0$, $s_0 = 1$ in utež α na $0.01^{1/velikost\_okna}$ [5]. Tako dobimo skozi čas spreminjajočo se vrsto $\alpha MSE(t)$. Za primerjanje algoritma A in algoritma B uporabimo statistiko Q [10]:

$$Q_i(A, B) = \log_2 \left(\frac{\alpha MSE_i^A}{\alpha MSE_i^B} \right) \tag{2.16}$$

kjer se nadpisana A in B nanašata na model A in model B in ne na potence. Statistika Q (2.16) je prav tako izračunana v vseh točkah in je odvisna od časa. Ko je njena vrednost negativna, je boljši algoritem A, ko je pozitivna, je boljši algoritem B.

2.4 Metodi ocenjevanja zanesljivosti napovedi

Nekatere napovedi, ki jih generirajo napovedni modeli so lahko bolj zanesljive, druge manj zanesljive. Klasične mere uspešnosti v regresiji (razdelek 2.3) ocenjujejo model s kumulativno vrednostjo napak napovedi za vse testne primere. Kot take ne povedo nič o pričakovani napaki posamezne napovedi za še neviden primer na vhodu. Iz tega razloga niso primerne za uporabo pri ocenjevanju zanesljivosti napovedi.

Ocene posameznih napovedi so najpogosteje vgrajene v sam algoritem napovednega modela. Na primer Gammerman, Vovk in Vapnik [13] razširijo metodo podpornih vektorjev (razdelek 2.2.5) tako, da izračuna tudi ocene *zaupanja* in *verodostojnosti*. Poznana je tudi razširitev večnivojske umetne nevronske mreže (razdelek 2.2.6) z dodatnim izhodnim nevronom, ki napoveduje varianco v okolici napovedanega primera, kar lahko uporabimo kot oceno zanesljivosti [1].

Zgoraj omenjeni pristopi so zelo specifični, saj delujejo natanko na enem napovednem modelu. V nadaljevanju opišemo dve metodi ocenjevanja zanesljivosti napovedi, ki sta splošni in delujeta z uporabo kateregakoli modela (t.i. *black-box pristop*).

2.4.1 CNK - metoda, ki temelji na podlagi lokalnih napovedi

Ocena zanesljivosti CNK (angl. *C neighbours minus K*) za dani primer poda lokalno oceno o napaki napovedi [5]. Naj bo $L = \{(x_1, C_1), (x_2, C_2), ..., (x_n, C_n)\}$ učna množica primerov, kjer so $x_i, i = 1...n$ vektorji atributov primerov in $C_i, i = 1...n$ prave vrednosti izhodne spremenljivke.

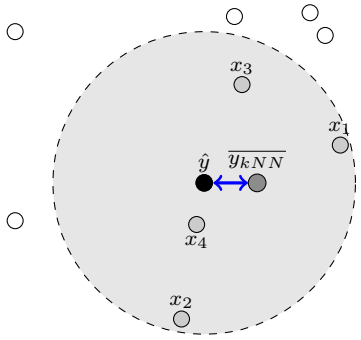
Na tem mestu poenotimo terminologijo, ki jo uporabljamo v tem diplomskem delu in se nekoliko razlikuje od terminologije, ki jo uporabljajo v [5, 6]. Vektor atributov primera brez ciljne spremenljivke označujemo z $\vec{x}$ (v izvorni literaturi označeno z $(x, \_)$), vrednost izhodne spremenljivke označujemo z y (v izvorni literaturi označeno s C) in napovedano vrednost izhodne spremenljivke z $\hat{y}$ (v izvorni literaturi označeno s K).

Ko na vhod v naš napovedni model dobimo nov primer $\vec{x}$ z uporabo enega od algoritmov, opisanih v razdelku 2.2, napovemo izhodno spremenljivko $\hat{y}$.

Ocena CNK je definirana kot razlika med srednjo vrednostjo izhodne spremenljivke y *k najbližjih sosedov* in napovedjo za dani primer.

$$CNK = \frac{\sum_{i=1}^{k} y_i}{k} - \hat{y} \tag{2.17}$$

V enačbi (2.17) predstavlja k število najbližjih sosedov, $\hat{y}$ napoved ciljne spremenljivke za dani primer in y_i prave vrednosti ciljne spremenljivke za izbrano množico k najbližjih sosedov. Ocena CNK je zelo občutljiva na lokalni šum v podatkih, ki pa ga poskušamo obvladovati z ustrezno izbiro parametra k.



Slika 2.4: Prikaz delovanja ocene CNK z iskanjem 4 najbližjih sosedov (k=4). Z x_1 do x_4 so označeni primeri najbližjih sosedov, z modro pa ocena CNK kot razlika med srednjo vrednostjo njihove izhodne spremenljivke in napovedjo za dani primer

2.4.2 SAbias - metoda, ki temelji na analizi občutljivosti

Metoda SAbias [6] temelji na analizi občutljivosti (angl. *sensitivity analysis*) modela v okolici napovedovanega primera. Osnovna ideja metode SAbias je ugotoviti, kako se napoved danega primera spreminja, če v njegovi okolici povzročimo majhne spremembe. Da bi ocenili zanesljivost napovedi za dani primer, uporabljamo naslednji postopek:

1. z uporabo enega od algoritmov, opisanih v razdelku 2.2, napovemo izhodno spremenljivko $\hat{y}$,

2. v množico učnih primerov dodamo ravnokar videni primer z nekoliko spremenjeno odvisno spremenljivko $\hat{y} \pm \varepsilon(l_{max} - l_{min})$ in enakim vektorjem atributov,

3. ponovno učimo model na razširjeni učni množici in naredimo napoved $\hat{y_\varepsilon}$ s spremenjenim modelom za isti primer.

Izbira parametra ε vpliva na to, kako veliko spremembo povzročimo v okolici danega primera. V predhodnih delih [7] so uporabljali izbiro vrednosti $\varepsilon \in E, E = \{0.01, 0.1, 0.5, 1.0, 2.0\}$. l_{max} in l_{min} predstavljata zgornjo in spodnjo mejo izhodne spremenljivke v učni množici, $\hat{y}$ prvotno napoved in $\hat{y_\varepsilon}$ napoved, ki jo naredimo po ponovnem učenju na spremenjeni učni množici.

$$SAbias = \frac{\sum_{\varepsilon \in E} (\hat{y_\varepsilon} - \hat{y}) + (\hat{y_{-\varepsilon}} - \hat{y})}{2|E|} \tag{2.18}$$

Ocena SAbias (enačba (2.18)) je predznačena ocena zanesljivosti lokalne napovedi, ki jo izračunamo kot povprečje razdalj med prvotnimi in popravljenimi napovedmi. Ideja ocene SAbias temelji na predpostavki, da so manj zanesljive tiste napovedi, kjer že majhna sprememba v okolici danega primera povzroči veliko razliko v napovedi ciljne vrednosti in obratno, bolj zanesljive tiste napovedi, kjer majhne spremembe v okolici danega primera ne povzročajo drastičnih sprememb v napovedih.

Bosnić [7] algoritme razdeli glede na to, če pred modeliranjem razdelijo učni prostor ali ne. In sicer na *preproste* (linearna regresija, lokalno utežena regresija) in *kompleksne* (regresijska drevesa, umetne nevronske mreže, metode podpornih vektorjev). Izkaže se, da je metoda SAbias še posebej primerna pri uporabi kompleksnih algoritmov in ni posebej primerna pri uporabi preprostih algoritmov.

2.5 Statistični testi

Statistični test je postopek za odločanje, če je hipoteza o numeričnih vrednosti populacije pravilna ali napačna. Problem opredelimo tako, da vprašanje postavimo kot dve nasprotujoči si trditvi ali hipotezi, med katerima se lahko odločamo: **ničelna hipoteza** H_0 proti **alternativni hipotezi** H_1. Rezultat statističnega testa je lahko bodisi:

- zavrni ničelno hipotezo H_0 v prid alternativni hipotezi H_1,

- ne zavrni ničelne hipoteze H_0.

Kot je razvidno iz napisanega zgoraj, ničelno hipotezo obravnavamo nekoliko drugače. Ničelna hipoteza namreč predstavlja trditev o lastnosti populacije, ki jo testiramo in za katero predpostavimo, da drži. Ničelno hipotezo želimo z danim statističnim testom ovreči. Če naš statistični test ne zavrne ničelne hipoteze, to še ne pomeni, da je ničelna hipoteza pravilna, le da ob danem vzorcu ne moremo z določeno verjetnostjo ovreči te trditve.

S **stopnjo značilnosti testa** α označimo verjetnost, da zavrnemo ničelno hipoteza v prid alternativni hipotezi, čeprav je ničelna hipoteza resnična (napaka 1. vrste). V vseh statističnih testih, ki jih izvajamo v poglavju 5 privzamemo vrednost $\alpha = 0.05$.

Ničelno hipotezo zavrnemo v prid alternativni hipotezi, če je **p-vrednost** statističnega testa (verjetnost ničelne hipoteze) manjša od α (izbrana stopnja značilnosti statističnega testa). Manjša kot je p-vrednost statističnega testa, bolj prepričani smo lahko o tem, da ničelna hipoteza res ni resnična in jo torej lahko ovržemo.

V eksperimentalnem delu (poglavje 5) smo uporabljali dva neodvisna testa za testiranje normalnosti statistike Q(prvotna napoved, popravljena napoved) (enačba 2.16), in sicer testa **Kolmogorov-Smirnov** ter **Anderson-Darling**. Po zavrnitvi normalnosti omenjene statistike smo uporabili **Wilcoxonov** enostranski parametrični test za testiranje, če je statistika Q manjša od 0 (kar bi pomenilo, da je boljša prvotna, nepopravljena napoved).

Poglavje 3

Opis problema in podatkov

Neodvisni sistemski operater distribucijskega omrežja električne energije (NYISO) od leta 2001 naprej za zvezno državo New York v ZDA na spletu [20] dnevno objavlja podatke o porabi električne energije. Podatke zajemajo senzorji vsakih 5 minut, vrednosti pa so nato agregirane po območjih. Distribucijsko omrežje zvezne države New York je razdeljeno na 11 območij (slika 3.1). Celotna množica podatkov za obdobje 2001-2012 tako zajema 13.346.802 primerov.

Ti podatki za problem napovedovanja predstavljajo številne izzive:

1. podatke sestavlja 11 različnih časovnih vrst z **različnimi (nestacionarnimi) porazdelitvami**,

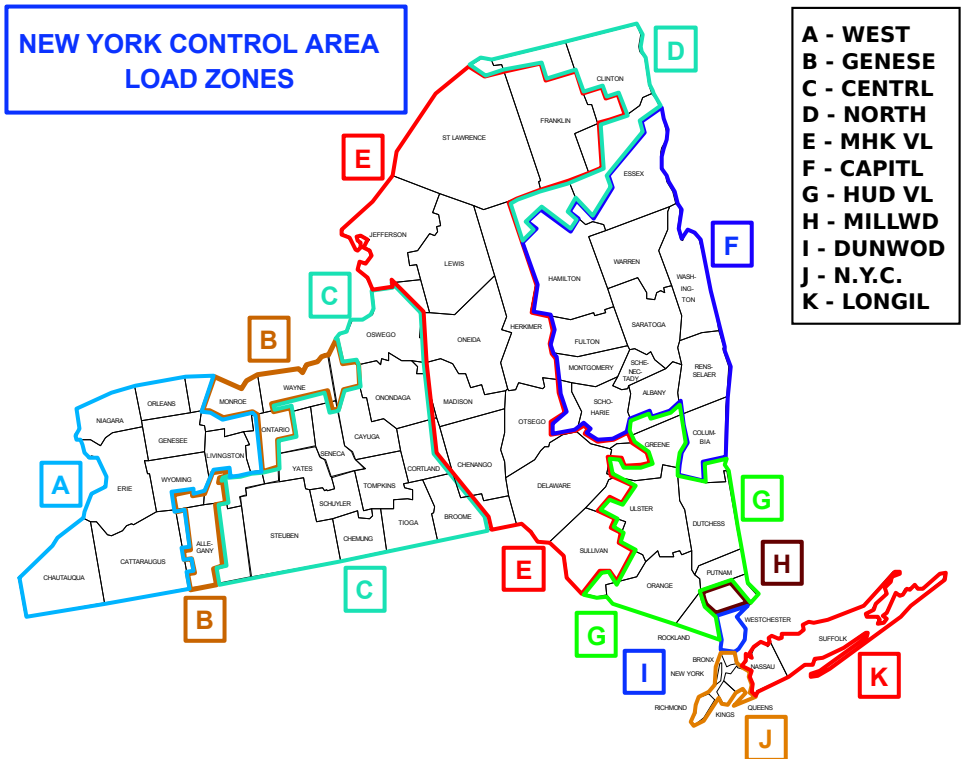
2. podatki so nepopolni, opravka imamo z **neregularnimi časovnimi vrstami** in **manjkajočimi vrednostmi**,

3. senzorji včasih odpovejo, kar pomeni, da obstajajo v podatkih **anomalije**,

4. **velika količina** podatkov,

5. **pomanjkanje atributov** za učenje.

Cilj diplomske naloge je izdelati napovedovalni sistem za avtomatsko napovedovanje prihodnje porabe električne energije za vsa območja. Omejimo se na kratkoročno napovedovanje in definiramo dva pristopa: **(i) periodični pristop** napovedovanja porabe prihodnjih 24 ur vsak dan začenši ob polnoči in **(ii) inkrementalni pristop** napovedovanja porabe za enako uro čez 24 ur. Pri implementaciji je potrebno določiti podrobnosti in naslednje parametre:



Slika 3.1: Območja distribucijskega omrežja zvezne države New York v ZDA za katere se zajemajo podatki o porabi električne energije

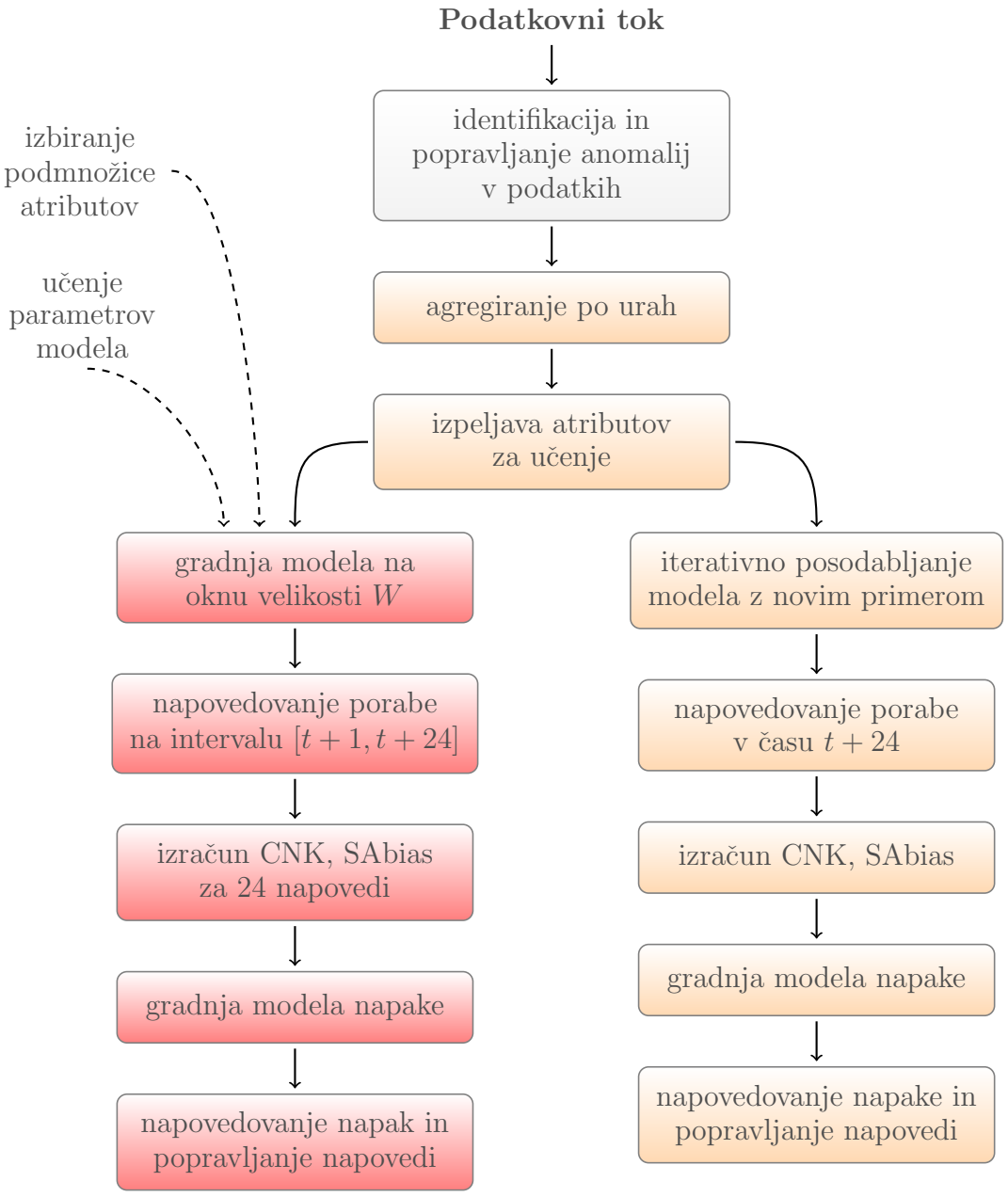
1. **učenje parametrov modela**, npr. pri metodi podpornih vektorjev sta to parametra C in σ,
2. **izbira podmnožice atributov**, ki jih bomo uporabljali v našem napovednem modelu za posamezno območje,
3. izračun ocen zanesljivosti lokalnih napovedi in **popravljanje napovedi**,
4. način **primerjave modelov**.

Poglavje 4

Sistem za kratkoročno napovedovanje porabe električne energije

V tem poglavju opišemo izdelani sistem za kratkoročno napovedovanje porabe električne energije. Če torej povzamemo in strnemo uvodne besede, se osredotočimo na metode, ki delujejo na podatkovnih tokovih - v danem trenutku nimamo na voljo celotnega nabora podatkov, temveč le podmnožico vseh preteklih vrednosti. Ker se porazdelitev skozi čas spreminja, statični model ne zadošča in je potrebno model prilagajati podatkom. Na sliki 4.1 je visokonivojska predstavitev izdelanega napovedovalnega sistema.

Zgoraj v sistem vstopa tok podatkov, za vsako od 11 območij zvezne države New York v ZDA s frekvenco 1 vrednost na približno 5 minut (razdelek 4.1). Včasih na vhodu ne dobimo podatka po 10 minut, včasih ga dobimo že po 4 minutah in včasih se vrednosti podvajajo. Poleg tega podatkovni tok vsebuje anomalije, ki jih nato s pomočjo algoritma, opisanega v razdelku 4.2, identificiramo in popravimo ter podatke nato agregiramo po urah. To pomeni, da to sprva neregularno časovno vrsto pretvorimo v regularno časovno vrsto z večjo periodo (1 ura) oziroma manjšo frekvenco. Osnovna časovna vrsta ima le dva atributa: čas in vrednost porabe ob tem času. Za gradnjo modelov in napovedovanje potrebujemo več atributov, zato dodamo koledarske atribute in atribute zakasnitev prvotne časovne vrste (angl. *lagged variables*). Takšen tok podatkov je nato vhod v bodisi periodični pristop (leva stran na sliki 4.1) bodisi iterativni pristop (desna stran na sliki 4.1).



Slika 4.1: Visokonivojska predstavitev izdelanega napovedovalnega sistema. Prikazana sta (1) **periodični pristop** k napovedovanju (leva stran) in (2) **iterativni pristop** (desna stran). Barva posameznega podsistema prikazuje frekvenco sprememb: bela - 1/5 minut, oranžna - 1/1 ura, rdeča - 1/24 ur.

Pri periodičnem pristopu se frekvenca delovanja zmanjša, saj model gradimo

vsak dan znova na določeni velikosti okna oziroma številu preteklih vrednosti primerov in napovedujemo porabo naslednjih 24 ur. Vse, kar opisujemo do sedaj, se dogaja nenehno oziroma *sproti* (angl. *online*). Izbiranje podmnožice atributov za dani model in učenje parametrov modela (razdelka 4.4 in 4.5) sta podproblema, ki smo se ju zaradi računske kompleksnosti odločili reševati *nesprotno* (angl. *offline*). V sklopu tega pristopa periodično gradimo različne modele in napovedujemo porabo za naslednjih 24 ur. Pri tem računamo drseče okno napak, ki ga posodabljamo vsak dan sproti, ko dobimo pravilne rezultate za predhodnih 24 napovedi. Prav tako vsak dan sproti zračunamo ocene zanesljivosti CNK in SAbias za novih 24 napovedi. Periodično gradimo linearni model napake v odvisnosti od ocen CNK in SAbias in tako za nove vrednosti CNK in SAbias napovemo predvideno napako, ki jo upoštevamo pri popravljanju napovedi. Na izhodu dobimo torej nov podatkovni tok napovedi, kjer imamo 3 napovedi (za vsako 24 vrednosti): prvotno napoved, napoved, popravljeno na podlagi ocene CNK in napoved, popravljeno na podlagi ocene SAbias. Različne napovedne modele, prav tako pa tudi popravljanje napovedi na podlagi ocen zanesljivosti ovrednotimo v poglavju 5.

Pri inkrementalnem pristopu se frekvenca delovanja ne zmanjša, torej sistem deluje z enako frekvenco. V osnovi delujejo podsistemi tega pristopa na enak način, glavna razlika pa je, da modela ne gradimo vsakič znova na nekem oknu predhodnih vrednosti omejene velikosti (*problem izbire velikosti okna*), temveč inkrementalno ali stohastično posodabljamo model z enim samim novim primerom. Napovedujemo nato vsakič za 24 ur vnaprej (le 1 primer) in prav tako ocene CNK in SAbias računamo le za posamezen primer. Tudi tukaj so izhod sistema 3 napovedi (prvotna, popravljena s CNK, popravljena s SAbias), vendar le za 1 vrednosti.

4.1 Zajem in predprocesiranje podatkov

Podatki so prostodostopni na spletni strani neodvisnega sistemskega operaterja NYISO [20], tako da je bilo samo pridobivanje podatkov trivialno (program *curl* v ukazni lupini). Podatki so shranjeni v petminutnih presledkih za posamezen dan za vsako od enajstih območij, kjer vsaka datoteka v obliki CSV vsebuje za 1 dan podatkov. S pomočjo skripte *perl* smo nato podatke takšne oblike pretvorili v eno samo datoteko v obliki CSV, kjer vsak stolpec predstavlja časovno vrsto enega od 11 območij.

Ker so podatki šumni in nepopolni (točki 2 in 3 v poglavju 3), jih agregiramo po urah. S preprosto agregacijo po urah sicer pretvorimo časovno vrsto iz neregularne v regularno in s tem odpravimo točko 2, vendar pa anomalije v tem primeru

postanejo del agregirane vrednosti in jih v takšni obliki težje odpravimo.

Iz tega vzroka najprej rešujemo problem anomalij, ki ga opišemo v razdelku 4.2. Šele nato, ko identificiramo in nadomestimo anomalije v podatkih, tako očiščene podatke agregiramo po urah (glej tudi sliko 4.1).

4.2 Identifikacija in popravljanje anomalij

V podatkih obstaja več tipov anomalij, in sicer:

1. **neveljavni podatki** - vrednosti, ki niso mogoče; npr. negativne vrednosti porabe električne energije,

2. **osamljene napačne vrednosti** - posamezne vrednosti, ki se tako zelo razlikujejo od ostalih vrednosti v bližini, da jih lahko z določeno gotovostjo opredelimo kot napačne vrednosti (slika 4.2),

3. **napačne vrednosti v serijah** - ponavadi posledica okvare senzorjev ali problemov pri prenosu podatkov. Napačne vrednosti se v tem primeru pojavljajo po več skupaj in zaradi njihove gostote v lokalni okolici to tako popačijo, da jih le s primerjanjem bližnjih točk ne moremo opredeliti kot napačne vrednosti (slika 4.3).

Anomalije tipa 1 in 2 ne predstavljajo prevelikih težav in jih lahko uspešno popravimo že s preprostimi metodami za odpravljanje šuma, npr. zavračanjem vrednosti manjših ali enakih 0 in glajenjem z uporabo *drsečega povprečja* (angl. *moving average*). Vendar pa taki pristopi odpovejo pri anomalijah tipa 3. V nadaljevanju opišemo pristop, ki je na naših podatkih (11 časovnih vrstah iz različnih porazdelitev) uspešno odstranil in nadomestil anomalije v podatkih vseh treh tipov.

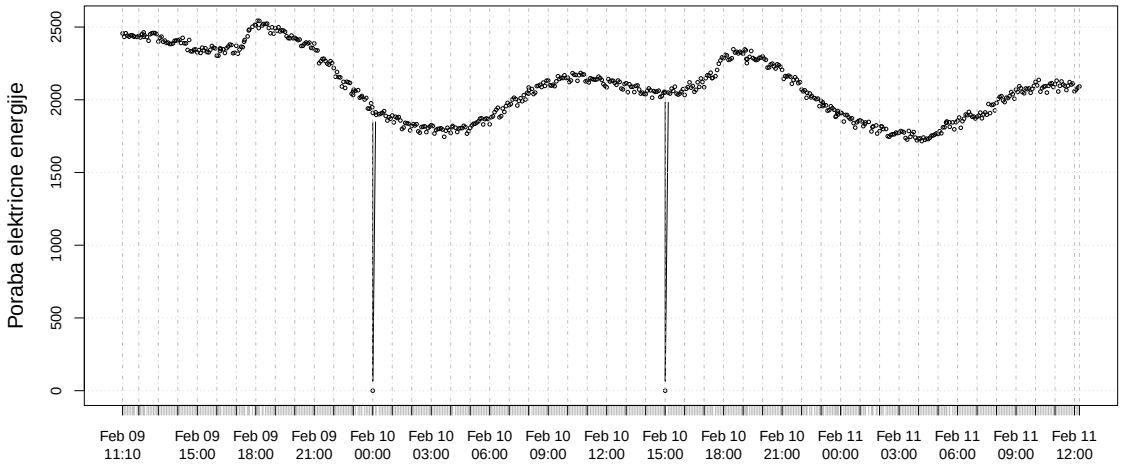
Najprej zgradimo podatkovno zbirko z naslednjimi učnimi podatki:

- **Load** - trenutna poraba,

- **CurrentHourMedian** - mediana zadnje ure,

- **PrevDay1SameHourMedian** - včerajšnja mediana za enako uro,

- **PrevDay2SameHourMedian** - predvčerajšnja mediana za enako uro.

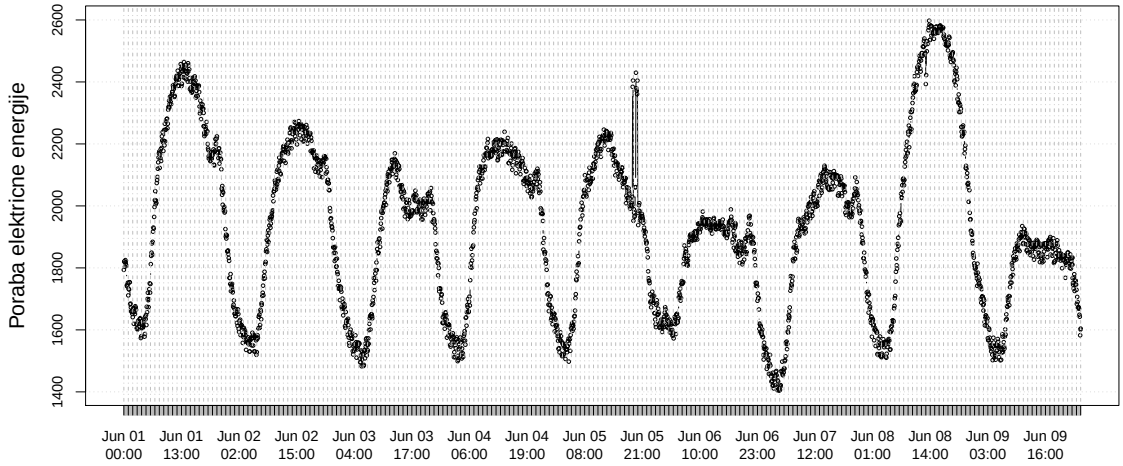


Slika 4.2: Anomalija v podatkih tipa 2 - osamljene napačne vrednosti. 10. februarja okrog polnoči in okrog 15:00 so poročane vrednosti porabe drastično različne od vseh ostalih v neposredni okolici

Nad temi podatki zgradimo model linearne regresije (razdelek 2.2.1). Dolžina okna, ki se je v eksperimentih pokazala za primerno, je 30 dni. Z uporabo *modela odstopanja gibanja povprečja* (angl. *mean-shift outlier model*) izračunamo ostanek po Studentu [1] za posamičen primer pri stopnji značilnosti testa α in ničelni hipotezi, da se premik povprečja ni zgodil. Da bi izračunali vseh n ostankov po Studentu bi lahko n-krat neodvisno zgradili tak model, vendar obstajajo za to računsko hitrejše poti. Ostanki po Studentu so med seboj odvisni, kar predstavlja dodaten problem. To rešujemo z *Bonferronijevo korekcijo p-vrednosti* za največji ostanek po Studentu (po absolutni vrednosti), kjer preprosto delimo vrednost z n [16].

Za računanje Bonferronijevega testa smo uporabili funkcijo *outlierTest* v paketu *car* programskega ogrodja **R**, pri čemer smo parameter *cutoff* nastavili na 0.05. Če s tem testom najdemo primere na zgrajenem modelu linearne regresije (opisanem zgoraj), ki imajo Bonferronijevo p-vrednost manjšo od 0.05, jih odstranimo in nadomestimo z vrednostjo mediane zadnjih treh ur.

[1] *Ostanek po Studentu* (angl. *Studentized residual*) je kvocient *pričakovanega ostanka* (angl. *residual*) z njegovo pričakovano standardno deviacijo.



Slika 4.3: Anomalija v podatkih tipa 3 - napačne vrednosti v serijah. 5. junija med približno 18:00 in 20:00 so poročane vrednosti porabe sicer lokalno podobne, vendar drastično različne od globalnega profila porabe

4.3 Izpeljava atributov za učenje

Iz začetne časovne vrste, kjer poznamo numerično vrednost porabe električne energije in datum ter čas, izpeljemo učno množico z naslednjimi atributi: Skupaj imamo torej 35 atributov za učenje. Pari *sin* in *cos* spremenljivk se uporabljajo kot pomoč napovednemu modelu pri modeliranju periodičnosti (dnevne in tedenske) [11]. Atribute Minus.***xx*** velja podrobneje razložiti. Skupaj imamo 27 takšnih atributov, in sicer 14 atributov (Minus.***24***, Minus.***48***, ..., Minus.***336***), kjer vrednost atributov ustreza: včerajšnji porabi v enaki uri, predvčerajšnji porabi v enaki uri itn. vse do porabe pred 14 dnevi v enaki uri. Vrednosti nadaljnjih 13 atributov (Minus.***25***, Minus.***49***, ..., Minus.***313***) pa predstavljajo porabo na včerajšnji dan eno uro pred trenutno uro, porabo na predvčerajšnji dan eno uro pred trenutno uro itd.

V naslednjem razdelku opišemo *nesprotni* (angl. *offline*) sistem za izbiro podmnožice atributov za posamezen napovedni model in v razdelku 4.5 izbiro parametrov modela.

Atribut	Opis
Load	numerična vrednost porabe električne energije
Weekday	numerična vrednost od 0 do 6, ki opredeljuje dan
Hour	numerična vrednost od 0 do 23, ki opredeljuje uro
Weekend	numerična vrednost 0 ali 1, ki opredeljuje, če je danes vikend ali ne
Minus.$\boldsymbol{xx}$	numerična vrednost porabe električne energije pred $\boldsymbol{xx}$ urami $\boldsymbol{xx} \in \{24, 48, ..., 336, 25, 49, ..., 313\}$
Sin.24	numerična vrednost med 0 in 1, urna perioda
Cos.24	numerična vrednost med 0 in 1, urna perioda
Sin.168	numerična vrednost med 0 in 1, tedenska perioda
Cos.168	numerična vrednost med 0 in 1, tedenska perioda

Tabela 4.1: Opis izpeljanih atributov za uporabo v gradnji modela in napovedovanju

4.4 Izbiranje podmnožice atributov

Iz množice vseh *atributov* želimo za *dani model* izbrati le tisto podmnožico, ki daje najboljše rezultate. Za izbiro najboljših atributov smo uporabili metodo *vzvratne selekcije* (angl. *backwards selection*), kjer Algoritem 4.1 [17] gradi in ocenjuje modele z različnimi podmnožicami atributov in nato na podlagi neke mere uspešnosti (npr. *RMSE*) izbere najboljšo takšno podmnožico. Ime metode izhaja iz načina iskanja v prostoru rešitev, kjer začnemo s polno množico atributov, v naslednjih iteracijah algoritma pa atribute po enega ali več naenkrat odstranjujemo in ponovno gradimo ter ocenjujemo model na teh podmnožicah atributnega prostora.

Ker je število vseh podmnožic atributnega prostora v našem primeru preveliko ($2^{35} = 3.43 \times 10^{10}$), je potrebno uporabiti pametnejšo strategijo iskanja v prostoru rešitev. Algoritem *rekurzivnega odstranjevanja atributov* (angl. *recursive feature elimination*), ki ga uporabljamo v naši rešitvi temelji na rangiranju pomembnosti atributov (vrstica 5) od najbolj pomembnega do najmanj pomembnega, kjer za vsako preizkušano število S_i obržimo le S_i najbolj pomembnih atributov (vrstica 7). Nato ponovno zgradimo model (vrstica 8) na tej (novi) podmnožici atributov in napovedujemo vrednosti nevidenih primerov (vrstica 9). Po odstranitvi določenega števila atributov lahko ponovno rangiramo pomembnost preostalih atributov, lahko pa uporabljamo prvotne ocene. Za naključne gozdove se izkaže [12], da ponovno rangiranje ni koristno. Pri vseh ostalih modelih uporabljamo ponovno rangiranje atributov.

Da bi zmanjšali preveliko prileganje modela podatkom in se izognili *pristranskosti izbire* (angl. *selection bias*) [2] uporabljamo *prečno preverjanje* (angl. *cross-validation*). V vsaki iteraciji (vrstica 3) razdelimo množico primerov na dve disjunktni množici: množico učnih primerov in množico testnih primerov. Prvo uporabljamo za učenje in drugo za napovedi in ocenjevanje uspešnosti modela. Primeri v testni množici se med iteracijami ne prekrivajo, tako da v k iteracijah s testnimi množicami velikosti $\frac{n}{k}$ pokrijemo celotno množico primerov.

Na koncu sledi ocenjevanje uspešnosti modela na različnih podmnožicah atributov in izbiranje optimalne velikosti podmnožice atributov S_i (vrstica 14). Ker imamo več iteracij, končno uspešnost, merjeno z RMSE računamo za posamezno velikost podmnožice atributov S_i kot povprečje ocen RMSE *čez vse iteracije oziroma testne množice podatkov v prečnem preverjanju*. Nato primerjamo med seboj te vrednosti in izberemo velikost podmnožice S_i z minimalno oceno RMSE (4.1).

Ko imamo velikost optimalne podmnožice atributov S_i določeno, je potrebno v zadnjem koraku izbrati, kateri so ti atributi. V primeru, da v vmesnih iteracijah ne računamo ponovno vrstnega reda atributov (vrstica 10), je to kar prvih S_i atributov v urejeni množici atributov, kjer so atributi urejeni po padajoči pomembnosti. Če pa ponovno računamo pomembnost atributov se lahko zgodi, da so v različnih iteracijah prečnega preverjanja zaradi različne porazdelitve vzorcev v podatkovni množici atributi različno ocenjeni. Končno množico atributov nato izberemo tako, da povprečimo ocenjeno pomembnost atributa preko vseh iteracij in atribute razvrstimo od najbolj pomembnega do najmanj pomembnega. Tudi v tem primeru izberemo prvih S_i atributov.

$$RMSE_{opt} = \min_{RMSE} \left\{ \frac{\sum_{j=1}^{k} RMSE(observed_j, predictions_j)}{k} \right. \tag{4.1}$$

V zgornji formuli (4.1) predstavlja k število prečnih preverjanj, $observed_j$ prave vrednosti izhodne spremenljivke v k-ti iteraciji in $predictions_j$ napovedane vrednosti izhodne spremenljivke v k-ti iteraciji prečnega preverjanja.

[2] Pristranskost izbire se lahko zgodi, ko nek atribut naključno močno korelira s ciljno spremenljivko za dan vzorec primerov. Algoritem tak atribut dobro oceni, čeprav se atribut na novem testnem naboru podatkov kasneje izkaže za neuporabnega.

Algoritem 4.1: Izbiranje podmnožice atributov

```
1  Normaliziraj podatke na interval [0, 1]
2  foreach Model do
3      foreach Iteracijo ponovnega vzorčenja do
4          Razdeli podatke na učno in testno množico
5          Izračunaj pomembnost/rang atributov
           // Rekurzivno odstranjevanje atributov
6          foreach Podmnožico atributov S_i, i ∈ to S do
7              Obdrži S_i najbolj pomembnih atributov
8              TuneTrainModelParameters(S_i, Model) // Alg. 4.2
9              Napovej primere v testni množici (uporabi naučene parametre
               modela)
10             [neobvezno] Ponovno izračunaj pomembnost za vsak atribut
11         end
12     end
       // Izberi optimalno podmnožico atributov
13     Izračunaj napako za vse velikosti S_i na testnih množicah
14     Določi najprimernejše število atributov
15     Izberi dokončno podmnožico atributov za končni model
16     Zgradi Model na prvotni množici podatkov z S_i najpomembnejšimi
       atributi (uporabi naučene parametre modela)
17 end
```

4.5 Učenje parametrov modela

Nekateri modeli imajo različne parametre, ki vplivajo na obnašanje in izrazno moč modela. Umetne nevronske mreže so primer takšne vrste modela, kjer so parametri vrsta aktivacijske funkcije, število skritih nivojev in število nevronov. Znano je, da pri umetnih nevronskih mrežah večje število nevronov in skritih nivojev poveča moč modeliranja, kar pomeni, da lahko aproksimirajo bolj zapletene nelinearne funkcije. Iz tega sledi, da bo optimalni nabor parametrov modela za različne podmnožice atributov različen. Nabor podatkov z le 3 atributi bo zahteval različne vrednosti parametrov kot nabor podatkov s 35 atributi (velikost celotnega atributnega prostora v našem primeru).

V prejšnjem razdelku smo opisali postopek izbiranja podmnožice atributov, kjer parametrov modela nismo posebej omenjali. Ker sta izbira podmnožice atributov in učenje parametrov modela povezana, v notranji zanki algoritma 4.1 (vrstica 8) za določeno velikost podmnožice atributov S_i poiščemo dobre vrednosti parametrov modela[3]. Za nekatere modele, na primer linearno regresijo, učenje parametrov modela ni pomembno in torej v teh primerih vrstica 8 v algoritmu 4.1 ne izvede nobene operacije. V drugih primerih, npr. metodo podpornih vektorjev ali umetne nevronske mreže pa ta vrstica sproži Algoritem 4.2. Ta algoritem najde dobre vrednosti parametrov modela med vsemi kombinacijami različnih vrednosti parametrov, ki so specifične za posamezen model.

[3]Pomembno je poudariti, da gre tukaj ponovno za (še eno–notranje) **10-kratno prečno preverjanje**, vendar tukaj že drugič razdelimo množico vseh podatkov. Če v algoritmu 4.1 v srednji zanki npr. 90% vseh originalnih podatkov uporabimo za učno množico pri izbiri podmnožice atributov, nato v algoritmu 4.2 v posamezni iteraciji prečnega preverjanja 90% teh podatkov (oziroma 81% originalnih podatkov) uporabimo za iskanje parametrov modela.

Algoritem 4.2: Učenje parametrov modela

```
Določi nabor vrednosti parametrov modela, ki jih preizkušamo
foreach Nabor parametrov modela do
    foreach Iteracijo ponovnega vzorčenja do
        Iz množice primerov izvzemi nekaj primerov
        Zgradi model na preostanku primerov
        Napovej vrednosti za izvzete primere
    end
    Izračunaj povprečno uspešnost čez vse izvzete primere iteracij
end
Določi optimalni nabor parametrov modela
Zgradi končni model na vseh prvotnih primerih z optimalnim naborom
parametrov
```

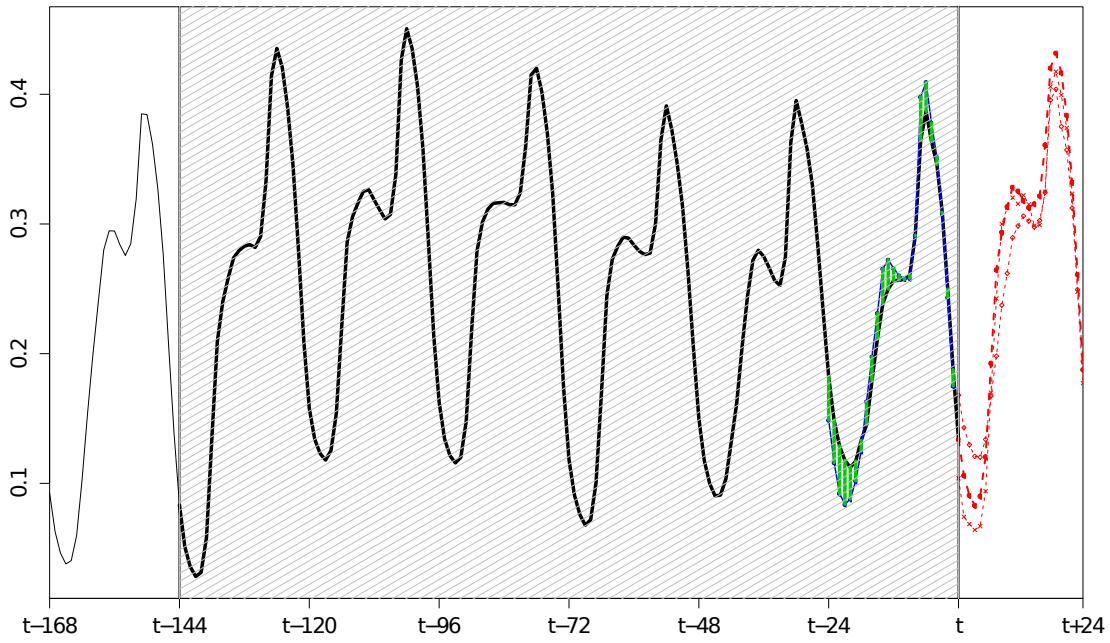
4.6 Periodični pristop napovedovanja

Slika 4.4 prikazuje periodični pristop v času t, ko smo ravnokar premaknili drseče okno za 24 vrednosti naprej (zasenčeno območje). V vsaki iteraciji algoritma izberemo/izpeljemo ustrezno podmnožico atributov za dani model in tok podatkov.[4]

Velikost okna v tem primeru je 144 ur le za ilustrativne namene, v resnici uporabljamo vrednost 336 ur (2 tedna). Nato izračunamo napake napovedi prejšnjega dneva (zelene črte), torej razliko med dejanskimi vrednostmi porabe (črna črta) in napovedmi, ki smo jih naredili v času $t - 24$ (modra črta). Velja opozoriti, da nam je v trenutku t vrednost napovedi, ki smo jih naredili v času $t - 24$, že znana. Nato zgradimo model s predhodno naučenimi parametri[5] in napovemo vrednosti za vsako uro naslednjega dne (rdeča črta). Nato algoritem izračuna ocene zanesljivosti CNK in SAbias, izgradi linearni model napake v odvisnosti od obeh ocen, napove napake v obeh primerih in na koncu popravi vrednosti napovedi prihodnjega dneva, enkrat z ocenami zanesljivosti CNK (rdeča črta) in drugič z ocenami zanesljivosti SAbias (rdeča črta).

[4]Podmnožico atributov smo določili v *nesprotnem načinu* (angl. *offline* mode) za dani algoritem in podatkovni tok (razdelek 4.4).

[5]Parametre smo izbrali predhodno za dani model in podatkovni tok (razdelek 4.5).



Slika 4.4: **Periodični pristop** napovedovanja - napovedujemo porabo električne energije za naslednjih 24 ur. Znotraj zasenčenega območja so podatki, na katerih lahko zgradimo model. Modra črta prikazuje v dejanske pretekle vrednosti, na podlagi katerih lahko v trenutku t izračunamo napake napovedi prejšnjega dne (zelene črte). Tri različne rdeče črte pa prikazujejo prvotno napoved in nato obe popravljeni napovedi (s pomočjo CNK in SAbias).

Algoritem 4.3: Periodični pristop napovedovanja

```
1  foreach Začetek dneva do
2      Izberi ustrezno podmnožico atributov za dani model
3      Premakni drseče okno velikosti W za 24 vrednosti naprej
4      Izračunaj napake napovedi prejšnjega dne
5      Obreži vrsto preteklih CNK in SAbias napovedi na ustrezno dolžino
6      Zgradi model z naučenimi parametri na učni množici
7      Pripravi vrednosti atributov kot vhod v napovedni model
8      Napovej naslednjih 24 vrednosti
9      Izračunaj oceni CNK in SAbias za naslednjih 24 vrednosti
10     Zgradi model napake v odvisnosti od vrednosti ocen CNK in SAbias
11     Napovej napake z obemi modeli in popravi originalno napoved
12     Izračunaj mere uspešnosti
13 end
```

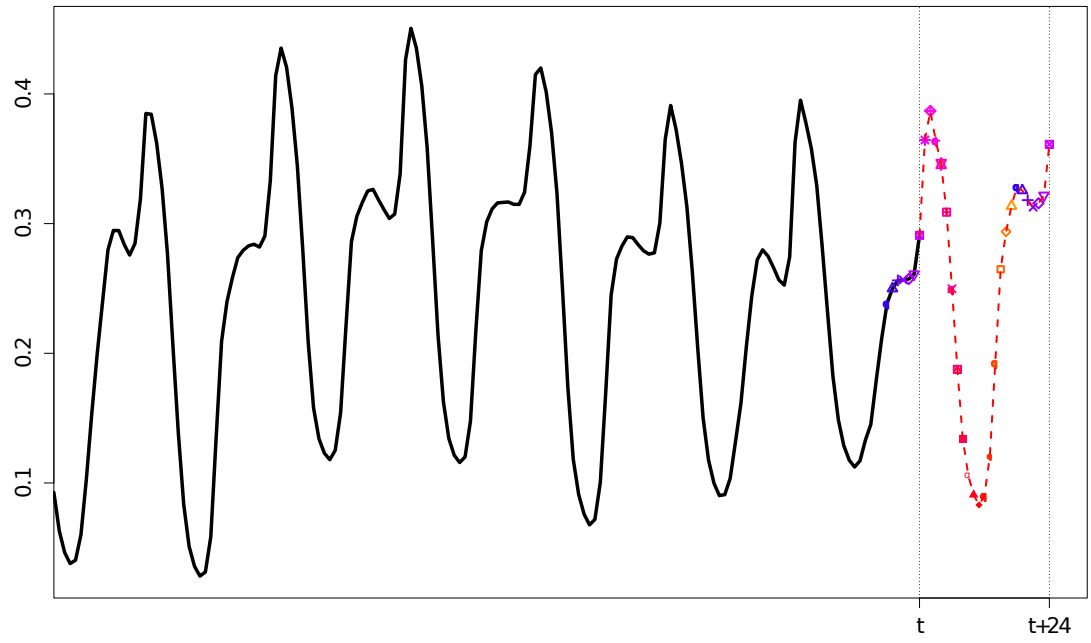
4.7 Inkrementalni pristop

Inkrementalni pristop se najbolj približa zahtevam napovedovanja iz podatkovnih tokov, naštetih v razdelku 2.1, saj preteklih primerov ne držimo v *oknu*, prav tako ne gradimo modela periodično, temveč ga venomer posodabljamo z novim prispelim primerom.

Kot je razvidno iz algoritma 4.4 in slike 4.5, se vedno učimo na primerih do trenutka t, napovedujemo pa posamezni primer v času $t + 24$. Enake omejitve morajo veljati za računanje ocen zanesljivosti CNK in SAbias. Napako napovedi lahko vedno računamo za napovedi, ki smo jih naredili v času $t-24$ in ne napovedi, narejene kasneje.

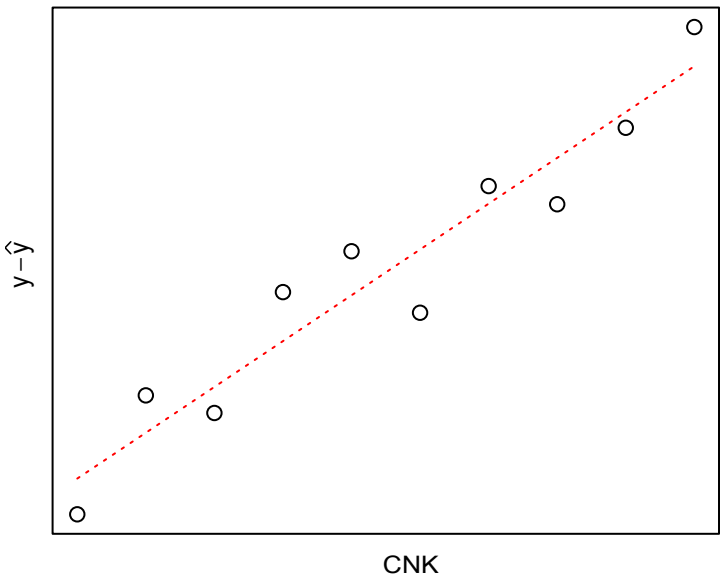
Algoritem 4.4: Inkrementalni pristop napovedovanja

```
1  Zgradi model na začetni učni množici
2  foreach Example do
3      Posodobi model z novim primerom
4      Napovej vrednost t + 24
5      Izračunaj oceni CNK in SAbias za napovedan primer
6      Izračunaj napako napovedi (originalne in popravljenih), narejenih v
       t − 24
7      Dodaj napake v vrste napak in obreži vrste
8      Zgradi modela napake v odvisnosti od ocen CNK in SAbias
9      Napovej napake z obemi modeli in popravi originalno napoved
10     Izračunaj mere uspešnosti
11 end
```



Slika 4.5: Inkrementalni pristop napovedovanja - v vsaki točki t napovedujemo porabo električne energije za točko $t + 24$. Z enako barvo in simbolom sta označeni dve točki v razmiku 24 ur. Za vsako napovedano vrednost v prihodnosti, označeno z neko barvo in simbolom, je vrednost, označena z enako barvo in simbolom 24 ur pred njo zadnja vrednost v času napovedovanja, s katero smo lahko posodobili napovedni model. Poleg tega vrednosti napak predhodnih napovedi v modelu napak vedno zamujajo 24 ur, saj lahko napako, ki jo naredimo pri napovedi ob času $t - 24$ ovrednotimo šele v času t. Tako bo napaka trenutne napovedi v času t ovrednotena šele v času $t + 24$ (enaka barva in simbol).

4.8 Popravljanje napovedi z ocenami zanesljivosti



Slika 4.6: Popravljanje napovedi z oceno zanesljivosti CNK - gradnja linearnega modela napake v odvisnosti od preteklih ocen zanesljivosti CNK

Osnovna ideja popravljanja napovedi z ocenami zanesljivosti je prikazana v sliki 4.6. Gradimo in obnavljamo linearni model ocen zanesljivosti (npr. CNK in SAbias) v odvisnosti od napake napovedi v tistem trenutku. Najnovejša informacija v takem modelu je jasno lahko le ocena CNK za 24 ur nazaj, saj šele v danem trenutku dobimo povratno informacijo o napaki napovedi, ki smo jo naredili v tistem času. Število primerov, na katerih gradimo tak linearni model je v našem primeru enako velikosti okna za periodični pristop. Na koncu napoved popravimo z uporabo formule:

$$\widehat{y_{CNK_i}} = \widehat{y_i} + f(CNK_i) \tag{4.2}$$

kjer je $f(CNK_i)$ v zgornji enačbi napoved napake linearnega modela za trenutni CNK_i. Enak princip uporabljamo tudi za gradnjo modela napake in popravljanja napovedi z ocenami zanesljivosti SAbias.

Poglavje 5

Eksperimentalna evalvacija

Visokonivojsko predstavitev izdelanega napovednega sistema (slika 4.1) in njene komponente smo predstavili v prejšnjem poglavju, v tem poglavju pa predstavimo rezultate napovednega sistema za podatkovne tokove na podatkih za 11 območij v zvezni državi New York v ZDA in jih komentiramo.

5.1 Izbira podmnožice atributov in parametrov modela

5.1.1 Optimalna velikost podmnožice atributov

V začetnih testiranjih se je izkazalo, da izbira velikosti podmnožice S_i z minimalno oceno $RMSE$, predstavljeno v enačbi (4.1), kljub 10-kratnem prečnem preverjanju vodi do prevelikega prileganja množici podatkov (angl. *overfitting*) in posledično slabših rezultatov pri simulaciji napovedovanja. Zaradi tega smo se odločili, da število atributov v podmnožici atributov izberemo na bolj robusten način, in sicer [17]:

$$RMSE_{tol} = 100 \times \frac{RMSE - RMSE_{opt}}{RMSE_{opt}} \tag{5.1}$$

kjer je napaka $RMSE_{opt}$ izračunana s pomočjo formule (4.1) in predstavlja najmanjšo napako. Nato izberemo takšno najmanjšo podmnožico atributov, ki ima napako $RMSE_{tol}$ manjšo od $1.5 \times RMSE_{opt}$.

V tabeli 5.1 so prikazane izbrane vrednosti števila atributov izbrane podmnožice za vse podatkovne tokove pri uporabljenem algoritmu regresijskih dreves.

Vidimo lahko, da je pri prvotni izbiri podmnožice atributov s pomočjo formule 4.1 (stolpec Nattr(opt)) število izbranih atributov vedno večje ali enako številu izbranih atributov pri izbiri s pomočjo formule 5.1 (stolpec Nattr(tol)), čeprav je razlika v uspešnosti napovedi na testni množici zanemarljiva.

Nabor podatkov	**Nattr(opt)**	**Nattr(tol)**	**RMSE(opt)**	**RMSE(tol)**
WEST	20	20	75.54	75.54
GENESE	20	8	90.62	91.38
CENTRL	15	15	33.53	33.53
NORTH	15	9	44.03	44.18
MHK VL	10	9	54.68	54.97
CAPITL	15	10	117.33	118.01
HUD VL	20	20	49.57	49.57
MILLWD	10	10	33.28	33.28
DUNWOD	34	25	206.76	207.76
N.Y.C.	34	10	21.44	21.72
LONGIL	15	10	68.94	69.91

Tabela 5.1: Primerjava med izbiro podmnožice atributov z $RMSE_{opt}$ in $RMSE_{tol}$ za **regresijska drevesa**. Stolpca Nattr(opt) in Nattr(tol) predstavljata izbrano število atributov za posamezni podatkovni tok, RMSE(opt) in RMSE(tol) pa napako, ki jo taka podmnožica atributov napravi na testnih podmnožicah ko delamo 10-kratno prečno preverjanje na prvih 80 dneh leta 2007.

5.1.2 Nastavitve pri izbiri podmnožice atributov

Izbiranje podmnožice atributov in učenje parametrov modela, ki smo ju opisali v prejšnjem poglavju, smo izvedli na prvih 80 dneh leta 2007. V vseh primerih smo uporabljali 10-kratno prečno preverjanje. Pri linearni regresiji, regresijskih drevesih, naključnih gozdovih in posplošenem aditivnem modelu nismo izvajali učenja parametrov modela, saj zaradi značilnosti omenjenih modelov to ni bilo potrebno ali pa že sam model to implicitno vsebuje (naključni gozdovi). Pri umetnih nevronskih mrežah in metodi podpornih vektorjev smo v notranji zanki algoritma za izbiranje podmnožice atributov uporabljali tudi učenje parametrov modela (algoritem 4.2), pri katerem smo zopet uporabljali 10-kratno prečno preverjanje. To

Algoritem	Max Nattr	Učenje parametrov
Linearna regresija	34[1]	✗
Regresijsko dreveso	34[1]	✗
Naključni gozd	5[2]	✗
Posplošeni aditivni model	5[2]	✗
Metoda podpornih vektorjev	34[1]	✓(C, σ)
Umetna nevronska mreža	10[3]	✓$(size, decay)$

Tabela 5.2: Uporabljene nastavitve pri iskanju podmnožice atributov

[1] Testirane podmnožice atributov: od 1 do 10, 15, 20, 25, 30 in 34 (vsi atributi)

[2] Testirane podmnožice atributov: od 1 do 5 atributov

[3] Testirane podmnožice atributov: od 1 do 10 atributov

prikazuje tabela 5.2. Pri naključnih gozdovih in posplošenem aditivnem modelu smo morali omejiti maksimalno število izbranih atributov na 5 zaradi časovne kompleksnosti pri izvajanju simulacij napovedovanja, pri umetnih nevronskih mrežah pa na 10 atributov.

5.2 Rezultati

Tabela 5.3 podaja rezultate simuliranja periodičnega pristopa za različne algoritme. Algoritme smo testirali na različnih podatkovnih tokovih (11 območij v zvezni državi New York). Število primerov v posameznem podatkovnem toku smo omejili na 8760 (1 leto podatkov). Prav tako smo omejili velikost okna za hranjenje preteklih podatkov na 2 tedna (336 primerov).

Navedene so napake RMSE, napake MAPE in Pearsonov korelacijski koeficient. Ker so napake RMSE izražene v enakih enotah, kot originalni podatki, njihovih vrednosti ne moremo primerjati med seboj za različne podatkovne tokove (vendar pa jih lahko med seboj primerjamo za posamezni podatkovni tok). Zato podajamo tudi napake MAPE, ki so izražene v odstotkih in so tako neodvisne od razpona vrednosti v prvotnem podatkovnem toku. Pearsonov korelacijski koeficient smo računali med pravimi vrednostmi in napovedanimi vrednostmi. Pri napakah RMSE in MAPE je boljša nižja vrednost, pri Pearsonovem korelacijskem koeficientu pa višja.

Ker so (nestacionarne) porazdelitve podatkovnih tokov različne in jih generirajo različni procesi, lahko vidimo, da uspešnost algoritmov strojnega učenja ni povsod enaka. Tako se je izkazalo na primer, da so na podatkovnem toku območja **New York City (N.Y.C.)** prav vsi algoritmi dosegali bistveno boljše rezultate (napaka MAPE okrog 2%) kot recimo na podatkovnem toku območja **MILLWD** (napaka MAPE okrog 11%).

V tabelah 5.3 in 5.4 **odebeljene** vrednosti predstavljajo RMSE nekega algoritma, ki je dosegel najboljši rezultat na danih podatkih območja. Predstavljene so tudi vrednosti napak MAPE ter Pearsonov korelacijski koeficient med pravimi vrednostmi in napovedanimi vrednostmi. Polje je obarvano sivo, če je za dani algoritem na pripadajočem podatkovnem toku popravljena vrednost s pomočjo ocene CNK boljša od prvotne napovedi. Z modro barvo pa so označene vrednosti algoritmov pri pripadajočem podatkovnem toku, če je popravljena vrednost s pomočjo ocene SAbias boljša od prvotne napovedi.

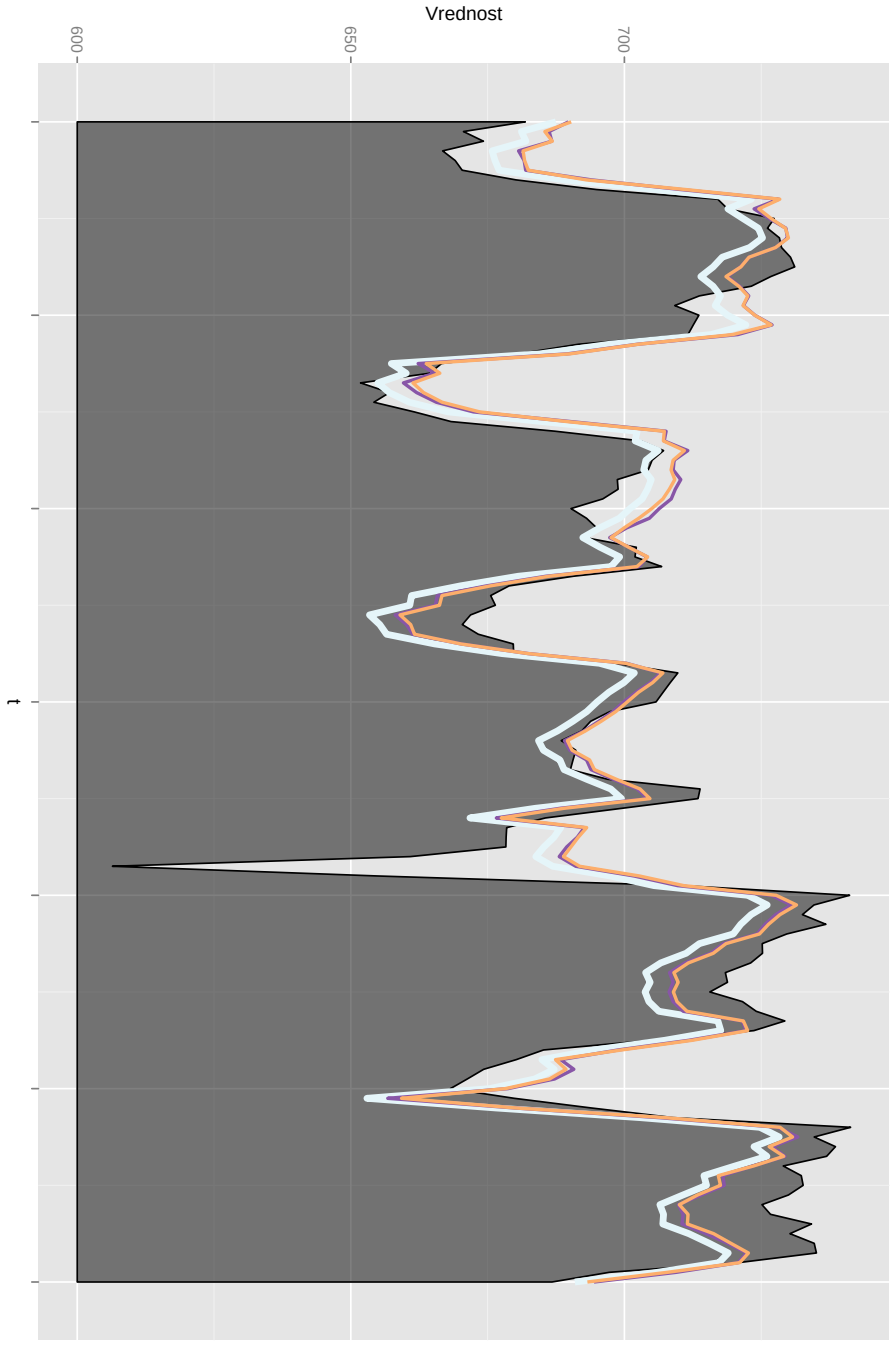
Na sliki 5.1 lahko vidimo podatke električne porabe za pet dni, in sicer prave vrednosti (polna črna barva), vrednosti prvotnih napovedi (bela barva), napovedi, popravljenih s pomočjo ocene CNK (oranžna barva) in napovedi, popravljene s pomočjo ocene SAbias (vijolična barva). Prikazan je podatkovni tok območja N.Y.C. in algoritem regresijskega drevesa, kjer popravljene napovedi dosežejo večjo napovedno točnost od prvotnih. Na sliki se to odraža tako, da so popravljene vrednosti bliže pravim vrednostim, vendar to ne drži v splošnem.

		SVM	GAM	RF	LR	RT	NN
WEST	*RMSE*	115.233	111.959	111.222	**105.670**	142.003	140.080
	MAPE	0.061	0.057	0.058	0.055	0.074	0.068
	Pearson	0.882	0.894	0.891	0.905	0.826	0.841
GENESE	*RMSE*	153.020	134.584	**128.732**	150.855	174,957	160.891
	MAPE	0.058	0.049	0.047	0.055	0.063	0.058
	Pearson	0.842	0.882	0.889	0.859	0.801	0.846
CENTRL	*RMSE*	85.892	77.734	**73.854**	87.023	98.979	122.780
	MAPE	0.085	0.075	0.071	0.083	0.096	0.113
	Pearson	0.859	0.890	0.898	0.867	0.819	0.770
NORTH	*RMSE*	107.750	100.049	**92.128**	112.316	128.955	132.990
	MAPE	0.064	0.056	0.052	0.063	0.073	0.072
	Pearson	0.855	0.883	0.896	0.863	0.805	0.820
MHK VL	*RMSE*	124.631	**121.287**	123.743	143.992	154.793	166.186
	MAPE	0.068	0.064	0.066	0.073	0.082	0.080
	Pearson	0.868	0.881	0.870	0.847	0.807	0.800
CAPITL	*RMSE*	290.754	**256.358**	267.706	274.273	337.325	337.106
	MAPE	0.075	0.064	0.067	0.067	0.084	0.079
	Pearson	0.886	0.915	0.904	0.904	0.851	0.861
HUD VL	*RMSE*	70.751	**63.748**	68.546	72.177	84.411	75.961
	MAPE	0.060	0.054	0.057	0.060	0.070	0.062
	Pearson	0.890	0.913	0.896	0.892	0.846	0.880
MILLWD	*RMSE*	47.220	44.499	**42.125**	46.065	55.406	53.983
	MAPE	0.118	0.110	0.101	0.111	0.133	0.122
	Pearson	0.826	0.852	0.862	0.843	0.770	0.788
DUNWOD	*RMSE*	602.585	**544.622**	570.902	573.931	739.436	646.009
	MAPE	0.064	0.059	0.060	0.059	0.077	0.065
	Pearson	0.879	0.906	0.892	0.896	0.824	0.873
N.Y.C.	*RMSE*	20.580	21.252	20.835	**19.624**	24.295	20.781
	MAPE	0.020	0.021	0.020	0.019	0.024	0.020
	Pearson	0.927	0.923	0.924	0.934	0.900	0.927
LONGIL	*RMSE*	143.455	121.616	**118.317**	126.811	158.735	162.250
	MAPE	0.057	0.047	0.045	0.049	0.061	0.061
	Pearson	0.835	0.888	0.889	0.876	0.806	0.814
	$\overline{RMSE}$	160.170	**145.246**	147.101	155.703	190.845	183.547
	$\overline{MAPE}$	0.066	0.060	0.059	0.063	0.076	0.073
	$\overline{Pearson}$	0.868	0.893	0.892	0.881	0.823	0.838
#izboljšanj s CNK		4	1	0	0	10	6
#izboljšanj s SAbias		0	0	0	0	11	0

Tabela 5.3: Rezultati napovedovanja za **periodični** pristop za 11 območij in 6 različnih algoritmov strojnega učenja. **Odebeljene** vrednosti so tiste napake RMSE nekega algoritma, ki so dosegle najboljši rezultat na danih podatkih območja. Predstavljene so tudi vrednosti napak MAPE ter Pearsonov korelacijski koeficient med pravimi vrednostmi in napovedanimi vrednostmi. Polje je obarvano sivo, če je za dani algoritem na pripadajočem podatkovnem toku popravljena vrednost s pomočjo ocene CNK boljša od prvotne napovedi. Z modro barvo pa so označene vrednosti algoritmov pri pripadajočem podatkovnem toku, če je popravljena vrednost s pomočjo ocene SAbias boljša od prvotne napovedi.

		LR	NN
WEST	*RMSE*	**95.337**	113.642
	MAPE	0.050	0.059
	Pearson	0.921	0.901
GENESE	*RMSE*	**121.299**	145.289
	MAPE	0.046	0.055
	Pearson	0.902	0.872
CENTRL	*RMSE*	**67.924**	73.308
	MAPE	0.068	0.069
	Pearson	0.915	0.910
NORTH	*RMSE*	**88.207**	103.815
	MAPE	0.051	0.060
	Pearson	0.905	0.880
MHK VL	*RMSE*	**103.048**	121.513
	MAPE	0.057	0.066
	Pearson	0.912	0.891
CAPITL	*RMSE*	**220.454**	251.371
	MAPE	0.058	0.063
	Pearson	0.936	0.923
HUD VL	*RMSE*	**57.446**	80.943
	MAPE	0.049	0.071
	Pearson	0.928	0.878
MILLWD	*RMSE*	**38.522**	46.929
	MAPE	0.094	0.116
	Pearson	0.886	0.857
DUNWOD	*RMSE*	**470.107**	545.410
	MAPE	0.052	0.056
	Pearson	0.928	0.911
N.Y.C.	*RMSE*	**18.319**	23.337
	MAPE	0.018	0.024
	Pearson	0.942	0.911
LONGIL	*RMSE*	**113.408**	129.789
	MAPE	0.045	0.051
	Pearson	0.899	0.879
	$\overline{RMSE}$	**126.734**	148.668
	$\overline{MAPE}$	0.053	0.063
	$\overline{Pearson}$	0.916	0.892
#izboljšanj s CNK		0	10
#izboljšanj s SAbias		0	3

Tabela 5.4: Rezultati napovedovanja za **inkrementalni** pristop za 11 območij in 2 različna algoritma strojnega učenja. **Odebeljene** vrednosti so tiste napake RMSE nekega algoritma, ki so dosegle najboljši rezultat na danih podatkih območja. Predstavljene so tudi vrednosti napak MAPE ter Pearsonov korelacijski koeficient med pravimi vrednostmi in napovedanimi vrednostmi. Polje je obarvano sivo, če je za dani algoritem na pripadajočem podatkovnem toku popravljena vrednost s pomočjo ocene CNK boljša od prvotne napovedi. Z modro barvo pa so označene vrednosti algoritmov pri pripadajočem podatkovnem toku, če je popravljena vrednost s pomočjo ocene SAbias boljša od prvotne napovedi.



Slika 5.1: Prave in napovedane vrednosti električne porabe za podatkovni tok N.Y.C. S **črno** barvo so predstavljene prave vrednosti, **bela** barva predstavlja napovedane vrednosti, **oranžna** barva napovedane vrednosti, popravljene s CNK in **vijolična** napovedane vrednosti, popravljene s SAbias

5.3 Ovrednotenje rezultatov

Izkaže se, da za periodični pristop najbolje delujeta algoritma strojnega učenja posplošenih aditivnih modelov in naključnih gozdov, vendar pa je njuna pomanjkljivost to, da potrebujeta precej časa za gradnjo modela.

Iz rezultatov lahko potrdimo, da je metoda ocenjevanja zanesljivosti SAbias primerna za kompleksne in ne za preproste algoritme, kar se sklada z ugotovitvami v [7]. V tabeli 5.3 (rezultati za periodični pristop) namreč lahko vidimo, da smo pri popravljanju napovedi s pomočjo ocene zanesljivosti SAbias dobivali boljše rezultate (modra barva) napovedi pri regresijskih drevesih. Regresijska drevesa učni prostor pred modeliranjem razdelijo (angl. *partition*) in jih zato umeščamo med kompleksne algoritme [7].

Pri uporabi ocene zanesljivosti CNK za popravljanje napovedi smo boljše rezultate (sivo ozadje) za nekatere podatkovne tokove dobivali pri uporabi algoritmov metod podpornih vektorjev, posplošenih aditivnih modelov, regresijskih dreves in umetnih nevronskih mrež.

Rezultati eksperimentov inkrementalnega pristopa so predstavljeni v tabeli 5.4 na enak način kot rezultati periodičnega pristopa. Opazimo lahko, da so napake pri inkrementalnem pristopu manjše od tistih pri periodičnem pristopu. Razlago gre iskati v *omejitvi velikosti okna* na zadnjih 336 v primeru periodičnega pristopa. V predhodnih testih, ki smo jih opravili s periodičnim pristopom, smo potrdili, da večje okno pri dovolj velikem številu primerov v podatkovnem toku vedno daje boljše rezultate. Ker pri inkrementalnem načinu ne uporabljamo uteženih faktorjev, drsečih oken ali kakšnih drugih načinov pozabljanja, je akumulirano znanje modela bolj povprečeno in daje na dolgi rok boljše rezultate. V primeru t.i. *spremembe učnega koncepta* (angl. *concept drift*), ko se porazdelitev vrednosti v podatkovnem toku drastično spremeni, bi periodični pristop deloval bolje od inkrementalnega.

Popravljanje napovedi z ocenama zanesljivosti CNK in SAbias za nekatere algoritme (SVM, regresijska drevesa, umetne nevronske mreže) na nekaterih podatkovnih tokovih daje boljše rezultate (oziroma manjše napake) od prvotnih napovedi. Vendar naši eksperimenti kažejo, da izboljšane napovedi nikoli ne dosežejo manjše napake za nek podatkovni tok od najboljšega modela na enakih podatkih. Uporaba popravljanja napovedi na podlagi ocen zanesljivosti se torej morda lahko izkaže za uporabno v primerih, ko ne moremo zlahka ali ne želimo zamenjati algoritma, ki je v uporabi. Popravljanje napovedi na podlagi ocen zanesljivosti namreč ne spreminja algoritma, temveč deluje nad rezultati (napovedmi), ki jih algoritem

izračuna. V vseh ostalih primerih je bolj smotrno ugotoviti, kateri model (in s kakšnimi parametri in podmnožico atributov) na tistem naboru podatkov deluje optimalno in nato uporabljati ta model.

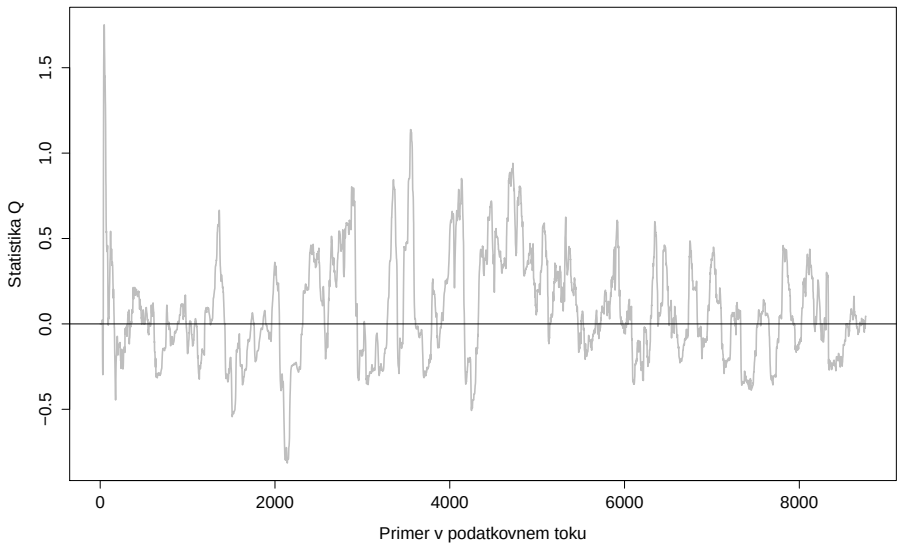
Osnovna težava takšnega napovedovanja je namreč, da imamo na voljo le *opazovane spremenljivke* (angl. *observed variables*), na primer porabo električne energije, ne pa tudi spremenljivk procesa, ki je te opazovane spremenljivke generiral. Daljši, kot je pogled v prihodnost, težje je napovedovati in posledično večje napake bomo delali pri napovedih, saj bo frekvenca pridobivanja povratne informacije naših napovedi nižja in torej osveževanje naših modelov na podlagi resničnih podatkov manj pogosto. Tako so na primer napake, ki jih delamo pri napovedih naslednje ure, bistveno manjše, kot napake, ki jih delamo pri napovedih naslednjega dneva, naslednjega tedna ali še dlje.

5.3.1 Dodatni opravljeni test izboljševanja napovedi

Pri vseh napovedih smo računali tudi popravljene napovedi s pomočjo ocen zanesljivosti CNK in SAbias. Iterativno smo računali tudi ocene αMSE pri prvotnih napovedih, napovedih, popravljenih s CNK in napovedih, popravljenih s SAbias. Nato smo izračunali statistiki:

- Q(prvotna napoved, popravljena napoved s CNK),
- Q(prvotna napoved, popravljena napoved s SAbias).

in uporabili Wilcoxonov enostranski statistični test z ničelno hipotezo, da je srednja vrednost statistike Q manjša od 0, kar bi pomenilo, da je boljša prvotna napoved. Ničelno hipotezo zavrnemo v prid alternativni hipotezi, če je p-vrednost manjša od 0.05. Primer izračuna statistike Q v vseh točkah podatkovnega toka je prikazan v sliki 5.2.



Slika 5.2: Statistika Q za podatkovni tok MILLWD in algoritem SVM - Q(prvotna napoved, popravljena napoved s CNK). Wilcoxonov statistični test pokaže, da je v tem primeru popravljena napoved s CNK boljša od prvotne napovedi.

Poglavje 6

Sklep

V diplomski nalogi smo uporabili različne metode strojnega učenja za gradnjo napovednih modelov, med njimi posplošene aditivne modele, umetne nevronske mreže, metode podpornih vektorjev in druge. Razvili smo metodo detekcije in popravljanja anomalij v podatkih. S pomočjo tehnik ponovnega vzorčenja smo naredili avtomatsko izbiranje optimalne podmnožico atributov in parametrov za različne algoritme. Predstavili in uporabili smo 2 različna pristopa napovedovanja v podatkovnih tokovih: **periodičnega** in **inkrementalnega**. S pomočjo ocenjevanja zanesljivosti napovedi in metod CNK ter SAbias smo originalne napovedi napovednih modelov inkrementalno popravljali in predstavili rezultate eksperimentov za realne podatke.

6.1 Zaključne ugotovitve

Pri implementaciji in eksperimentih napovednega sistema za podatkovne tokove z ocenjevanjem zanesljivosti smo prišli do nekaterih ugotovitev:

1. Za različne podatkovne tokove dobivamo z enakimi metodami različno dobre rezultate, saj so procesi, ki generirajo (nestacionarne) porazdelitve primerov različni. Zaradi tega tudi v poteku avtomatskega izbiranja podmnožice atributov, podmnožica atributov ni vselej enaka.

2. Popravljene napovedi s pomočjo ocen zanesljivosti CNK in SAbias za nekatere algoritme na nekaterih podatkovnih tokovih izboljšuje prvotne napovedi, vendar tako popravljene napovedi nikoli niso boljše od najboljšega napovednega modela za enak podatkovni tok. Z drugimi besedami povedano, če

napovedni model z uporabo določenega algoritma in določene podmnožice atributov ne dosega dovolj dobrih rezultatov napovedi, lahko z uporabo ocenjevanja zanesljivosti napovedi izboljšamo takšne napovedi. Vendar se je potrebno zavedati, da v mnogih primerih takšno popravljanje napovedi v resnici poslabša.

3. Inkrementalni pristop brez tehnik pozabljanja deluje na podatkovnih tokovih, kjer ne prihaja do *spremembe učnega koncepta* (angl. *concept drift*) bolje od periodičnega pristopa, kjer se omejimo na majhno okno preteklih primerov.

4. Eksperimenti in meritve, ki smo jih v sklopu diplomskega dela naredili, še zdaleč niso izčrpne. Čeprav smo metode preizkusili na realnih podatkih iz različnih porazdelitev, bi bilo dobro za boljšo primerljivost različne algoritme z ocenami zanesljivosti testirati z različnimi parametri (npr. parameter k v primeru CNK in nove vrednosti parametra α v primeru SAbias) na podatkih iz novih domen.

6.2 Nadaljnje delo

Implementacija napovednega sistema predstavlja izhodišče, ki bi ga lahko dopolnili in razširili z naslednjimi idejami:

1. Implementacija *adaptivnega drsečega okna* (angl. *adaptive windowing*), na primer algoritma ADWIN [14]. Gre za drseče okno, ki prilagaja svojo velikost glede na povprečje primerov znotraj okna. Tako velikost okna ne ostaja statična kot v našem primeru periodičnega pristopa, temveč se v različnih časovnih obdobjih širi ali oža.

2. Kombiniranje različnih ocen zanesljivosti pri modeliranju napake in popravljanju napovedi. Tako bi lahko iz ocen SAbias in CNK ocenili smer popravka (obe oceni sta predznačeni), pri modeliranju napake pa bi nato uporabljali tudi nekatere druge ocene zanesljivosti kot na primer BAGV, LCV, DENS [6] in druge.

3. Napovedovanje s pomočjo ansamblov (angl. ensemble forecasting), kjer bi kombinirali napovedi obstoječih napovednih modelov za doseganje bolj točnih

napovedi. Vendar se je potrebno zavedati, da bi v tem primeru dodajali dodatno kompleksnost, kar bi lahko imelo posledice pri omejitvah napovedovanja iz podatkovnih tokov (hitrost, računska moč itn.).

4. Vključitev vremenskih podatkov med atribute. S tem bi pridobili dodatno informacijo, na podlagi katere bi napovedni modeli lahko bolje delovali, saj so imeli v preteklosti veliko uspehov pri uporabi vremenskih podatkov za napovedovanje električne porabe.

5. Namesto algoritmov za izbiranje optimalne podmnožice atributov za posamezni algoritem pri nekem podatkovnem toku bi lahko uporabili *metodo poglavitnih komponent* (angl. *principal component analysis*), s katero bi omejili število atributov, ki bi še vedno pokrivali dovolj velik odstotek variance atributnega prostora.

Slike

Tabele

Literatura

[1] I. Kononenko, M. Kukar, *Machine Learning and Data Mining: Introduction to Principles and Algorithms*, Horwood Publishing, 2007.

[2] D. W. Bunn, E. D. Farmer, *Comparative models for electrical load forecasting*, Wiley, 1985.

[3] L. Kyriakides, M. Polycarpou, "Short Term Electric Load Forecasting: A Tutorial",*Studies in Computational Intelligence*, št. 35, str. 391–418, 2007.

[4] P. Rodrigues, J. Gama, R. Sebastiao, "Memoryless fading windows in ubiquitous settings", *Proceedings of Ubiquitous Data Mining (UDM) Workshop in conjunction with the 19th ECAI*, str. 27–32, 2010.

[5] Z. Bosnić, P. R. Rodrigues, I. Kononenko, J. Gama, "Correcting Streaming Predictions of an Electricity Load Forecast System Using a Prediction Reliability Estimate",*Advances in Intelligent and Soft Computing*, št. 103, str. 343–350, 2011.

[6] Z. Bosnić, I. Kononenko, "Comparison of approaches for estimating reliability of individual regression predictions",*Data and Knowledge Engineering*, št. 67, zv. 3, str. 504–516, 2008.

[7] Z. Bosnić, I. Kononenko, "Estimation of individual prediction reliability using the local sensitivity analysis",*Applied Intelligence*, št. 29, zv. 3, str. 187–203, 2008.

[8] Z. Bosnić, I. Kononenko, "Correction of regression predictions using the secondary learner on the sensitivity analysis outputs",*Computing and Informatics*, št. 29, zv. 6, str. 929–946, 2010.

[9] A. J. Smola, B. Schölkopf, "A tutorial on support vector regression",*Statistics and Computing*, št. 14, zv. 3, str. 199–222, 2004.

[10] J. Gama, R. Sebastiao, P. R. Rodrigues, "Issues in evaluation of stream learning algorithms", *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, str. 329–338, 2009.

[11] J. Gama, P. P. Rodrigues, "Stream-Based Electricity Load Forecast", v zborniku *Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases*, Berlin, 2007, str. 446–453.

[12] V. Svetnik, A. Liaw, C. Tong, T. Wang, "Application of Breiman's Random Forest to Modeling Structure-Activity Relationships of Pharmaceutical Molecules',*Multiple Classifier Systems*, št. 5, str. 334–343, 2004.

[13] A. Gammerman, V. Vovk, V. Vapnik, "Learning by Transduction", v zborniku *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, Madison, Wisconsin, 1998, str. 148–155.

[14] J. Gama. *Knowledge Discovery from Data Streams*, Chapman and Hall/CRC, 2010.

[15] E. Ikonomovska, J. Gama, S. Džeroski"Learning model trees from evolving data streams",*Data Mining and Knowledge Discovery*, št. 23, zv. 1, str. 128–168, 2011.

[16] J. Fox. *An R and S-Plus Companion to Applied Regression*, Sage Publications, 2002.

[17] M. Kuhn, Variable Selection Using The caret Package, 2012. Dostopno na: `http://cran.r-project.org/web/packages/caret/vignettes/caretSelection.pdf`

[18] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, Data Stream Mining: A Practical Approach, 2011. Dostopno na: `http://www.cs.waikato.ac.nz/~abifet/MOA/StreamMining.pdf`

[19] A. Ng, Supervised Learning - CS229 Lecture notes, Stanford University, 2012. Dostopno na: `http://cs229.stanford.edu/notes/cs229-notes1.pdf`

[20] (2012) New York Independent System Operator, Load Data. Dostopno na: `http://www.nyiso.com/public/markets_operations/market_data/load_data/index.jsp`