Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Fakulteta za matematiko in fiziko

Jaka Demšar

Razlaga napovednih modelov in posameznih napovedi pri inkrementalnem učenju

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in matematika

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: 00007/2012
Datum: 10.04.2012

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

Kandidat: **JAKA DEMŠAR**

Naslov: **RAZLAGA NAPOVEDNIH MODELOV IN POSAMEZNIH NAPOVEDI PRI INKREMENTALNEM UČENJU**

EXPLANATION OF PREDICTIVE MODELS AND INDIVIDUAL PREDICTIONS IN INCREMENTAL LEARNING

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

Metodologija razlage napovednih modelov in posameznih napovedi, ki je razvita za učenje iz stacionarnih podatkov (Štumbelj in Kononenko, 2010), je orodje, ki omogoča vpogled v delovanje modela ter omogoča ekspertom večje razumevanje problemske domene.

Kandidat naj prilagodi obstoječo metodologijo področju učenja iz podatkovnih tokov, kjer smo omejeni s procesorskim časom in spominom, naborom inkrementalnih modelov, prisotne pa so tudi spremembe v porazdelitvah učnih podatkov. Kandidat naj evalvira uspešnost delovanja prilagojene metode na naboru inkrementalnih učnih algoritmov in predlaga naj vizualizacijo spremembe razlage modela skozi čas.

Mentor:

doc. dr. Zoran Bosnić

Dekan Fakultete za računalništvo in informatiko:

prof. dr. Nikolaj Zimic

Dekan Fakultete za matematiko in fiziko:

akad. prof. dr. Franc Forstnerič

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Jaka Demšar, z vpisno številko **63090014**, sem avtor diplomskega dela z naslovom:

Razlaga napovednih modelov in posameznih napovedi pri inkrementalnem učenju

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 14. septembra 2012 Podpis avtorja:

Najprej bi se rad zahvalil svojemu mentorju doc. dr. Zoranu Bosniću za prijazno sodelovanje, strokovno pomoč in nasvete pri pisanju. Hvala moji družini – očetu Matevžu in materi Tadeji za spodbujanje in življenjske nasvete, sestrama Urši in Luciji pa za potrpežljivost in vse pečene dobrote, s katerimi sta mi lajšali pisanje. Na koncu gre zahvala dragi Neži za obilico razumevanja, podpore in bodrenja.

Kazalo

Povzetek

Uporaba strojnega učenja za podporo pri sprejemanju odločitev je vedno pogostejša. Zaupanje v rezultate povečamo z njihovo razlago, ki jih v obliki prispevkov vrednosti atributov utemelji in nam posledično dá dodaten vpogled v razumevanje problemske domene. Posplošena metoda za razlago je bila razvita za statične množice, odprt problem pa je ostala razlaga na podatkovnih tokovih, kjer smo omejeni z viri, vezani na inkrementalne učne modele in izpostavljeni možnosti spremembe v konceptih za podatki.

V diplomskem delu predstavimo novo posplošeno metodologijo za razlago inkrementalnih napovednih modelov in posameznih napovedi. Opremo se na obstoječe metode inkrementalnega učenja in statično metodo razlage, ki temelji na teoriji iger. Razvijemo tudi vizualizacijo spremembe konceptov v podatkovnih tokovih. Rešitev testiramo na več množicah, jo primerjamo s stacionarnimi metodami, vizualiziramo rezultate in jih analiziramo z mero podobnosti. Iz rezultatov eksperimentalnega dela sklenemo, da predlagana metoda uspešno razlaga inkrementalne učne modele in primere, statična pa za njih odpove. Vizualizacija nam nudi jasen vpogled v problemsko domeno.

Ključne besede:

strojno učenje, inkrementalno učenje, razlaga, podatkovni tok, sprememba koncepta, vizualizacija, klasifikacija

Abstract

Use of machine learning for decision support is evermore prevalent. We can increase the trust in computer-generated decisions by explaining them using feature value contributions which provide us with additional insight into concepts behind the domain of the problem. While a generalised method for static data sets has already been developed, we still face the problem of explaining examples and decision models built on data streams which demand limited resources, incremental learning models and the functionality of concept drift detection.

We present a novel generalised method for explanation of incremental learning models and individual instances. We derive our solution from existing incremental machine learning techniques and the static method for explanation based on game theory. We also develop a novel concept drift visualization method. The solution is tested on several datasets and then compared to static methods. Results are visualised and analysed with similarity measure. We conclude that the proposed method successfully explains incremental learning models and individual instances while outperforming the static method. Visualization proves to be a valuable technique for presentation of concepts behind data streams.

Keywords:

machine learning, incremental learning, explanation, data stream, concept drift, visualization, classification

Poglavje 1

Uvod

Podatkovni tokovi z razvojem računalniških in komunikacijskih tehnologij v svetu dobivajo vedno večjo vlogo. Hitri in neprestani viri novih informacij, ki vedno bolj natančno odražajo resnično (dinamično) naravo sveta, zaznamujejo našo dobo.

Strojno učenje je eden od mehanizmov za spopadanje s preobiljem podatkov. Z upoštevanjem računalniških, komunikacijskih in človeških faktorjev gradimo modele, ki iščejo vzorce, povzemajo koncepte in izpeljujejo novo znanje. Klasična paradigma je učenje na statičnih množicah, na podlagi katerih model napoveduje značilnosti novih podatkov. V kontekstu podatkovnih tokov to ne zadostuje, saj se koncepti, ki ležijo za podatki, skozi čas lahko spreminjajo in s tem razvrednotijo napovedni model. Tehnike strojnega učenja je zato potrebno primerno prilagoditi na spremembe, hitrost, porazdeljenost in neomejenost tokov. S tem v mislih gradimo *inkrementalne* modele, katerih ključna lastnost je zmožnost hitrega pozabljanja nerelevantnih podatkov in prilagajanja na nove koncepte.

Čeprav umetna inteligenca pogosto pri odločitvenih/napovednih problemih prekaša človeka, se mnogi pri kritičnih odločitvah neradi zanašajo nanjo. To je razumljivo, saj v splošnem tehnike strojnega učenja delujejo kot črna škatla – po vnosu podatkov vrnejo rezultat brez utemeljitve. Ljudje z ekspertnim znanjem tako zavržejo komponento, ki bi lahko izdatno izboljšala

kvaliteto njihovega dela. V ta namen lahko uporabimo metode za razlago napovednih modelov in posameznih napovedi, ki dajo vpogled v vzroke za določen rezultat strojnega učenja in s tem nudijo utemeljitev odločitve. Razvite so bile za statične modele, zato ostajata vprašanje njihove primernosti za uporabo na podatkovnih tokovih in odprta pot za razvoj inkrementalnih prilagoditev, kar je tudi tema tega diplomskega dela.

1.1 Pregled vsebine

Vsebina je zajeta v šest poglavij. Drugo poglavje je uvod v podatkovne tokove in strojno učenje s poudarkom na inkrementalnih modelih. Definiramo pojme, ki jih bomo potrebovali za izgradnjo modela razlage, posebno pozornost namenimo obravnavi sprememb v podatkovnem toku. Predstavimo statistično obvladovanje procesov, ki bo osnova za predlagano metodo.

V naslednjem poglavju si ogledamo poobdelavo podatkov na statičnih množicah. Osredotočimo se na obstoječe metode za razlago klasifikacij primerov in razlago napovednih modelov [13][14] (E. Štrubelj, I. Kononenko in M. Robnik Šikonja). S teorijo iger modeliramo problem in predstavimo obstoječa algoritma za razlago, ki ju bomo vključili v rešitev.

Izgradnjo in delovanje metodologije razlage napovednih modelov in posameznih napovedi pri inkrementalnem učenju formalno opišemo v četrtem poglavju. Rešitev dopolnimo z tehnikami vizualizacije in obdelavo toka razlag.

V eksperimentalnem delu (poglavje 5) metodo preizkusimo na nekaj množicah, analiziramo rezultate in jih primerjamo s statičnimi metodami, ter na ta način ugotavljamo smotrnost uporabe. Velik del zajemajo vizualizacije rezultatov.

Šesto poglavje sklene delo z končnimi ugotovitvami in predlogi za nadaljnje raziskave, kjer se dotaknemo drugih idej za metodo razlage in nadaljevanja razvoja predlagane rešitve.

Poglavje 2

Podatkovni tokovi

V poglavju preučimo značilnosti podatkovnih tokov in formaliziramo osnovne pojme. Obravnavamo pojav spremembe koncepta in ga skupaj z ostalimi posebnostmi umestimo v model inkrementalnega strojnega učenja. Cilj poglavja je predstaviti koncepte strojnega učenja (razen poobdelave, ki jo obravnavamo v poglavju 3), ki jih bomo potrebovali pri uspešni izgradnji modela za razlago napovednih modelov pri inkrementalnem učenju.

2.1 Značilnosti

Podatkovni tok je urejeno neomejeno zaporedje primerov, ki nastajajo v realnem času in jih je običajno možno prebrati le enkrat. S porastom mobilnih naprav, mrežnih tehnologij (GPS, TCP/IP promet, senzorji v električnih in vodnih omrežjih, informatizacija prometne infrastrukture...) in na splošno senzorjev vseh oblik ta oblika informacije postaja vedno bolj pogosta. Naraščajoča moč računalniških sistemov in komunikacij pomeni vedno hitreje naraščajočo količino podatkov, zato je strojno učenje smiselna rešitev za pridobivanje znanja in vzorcev. Pogosta je raba v procesiranju signalov, marketingu, industriji, medicini, bioinformatiki, finančnem sektorju, nadzornih procesih itd. Narava podatkovnih tokov je dinamična, kar je v kontrastu s statičnimi tabelami, ki smo jih vajeni v strojnem učenju. Predstavimo

nekaj glavnih značilnosti [5]:

1. Podatki so na voljo v toku, ki je lahko zelo hiter in praktično neomejen. Množica primerov, ki jo tako dobimo, ni stalna in niti ni nujno popolna. Zaradi relativno omejenih virov za shranjevanje je potreben sistem pozabljanja starih in sprejemanja novih primerov (operaciji dekrement in inkrement). Na voljo imamo manj kot $O(n)$ prostora, ponavadi želimo imeti rezultat v $\epsilon - okolici$ z verjetnostjo $1 - \delta$, zato je tipično v rabi prostor $O(\frac{1}{\epsilon^2}\log(\frac{1}{\delta}))$. Ker koncept naključnega dostopa posledično za podatkovne tokove ne velja, je dostop do starejših primerov lahko omejen, podatke beremo le enkrat.

2. Koncepti in znanje, ki leži za tokom, se lahko spreminjajo (soočamo se s problemom zaznave spremembe, kako ločiti anomalijo od nove porazdelitve podatkov) – podatki niso niti neodvisni niti enako porazdeljeni.

3. Prihod novih primerov ni reden, tokovi so porazdeljeni. Prejete vrednosti lahko vsebujejo šum, so nepopolne in težko predvidljive. Viri podatkov so pogosto avtomatizirani.

4. Podatki imajo poleg prostorskih še časovno dimenzijo (urejen vrstni red).

Učenje iz podatkovnih tokov obravnavamo v razdelku 2.4.

2.2 Osnovna formalizacija

V tem razdelku definiramo osnovne pojme iz področja podatkovnih tokov in strojnega učenja ter jih matematično opišemo. Najprej si oglejmo osnovne pojme iz področja *učenja iz statičnih množic* (*batch learning*).

Statične množice

Pri strojnemu učenju preučujemo *domeno* – rešujemo problem, ki izhaja iz te domene oziroma iščemo *koncepte* (znanje), ki ležijo za njo. Nabor podatkov,

ki ga uporabljamo, je množica m *primerov* (*instanc*) iz domene, opisanih z n *atributi* $A_1, A_2, ..., A_n$. Vsak atribut A_i ima zalogo vrednosti $\mathfrak{R}_i$, kar nam dá n-dimenzionalni *prostor instanc* $\mathcal{X} = \mathfrak{R}_1 \times \mathfrak{R}_2 \times ... \times \mathfrak{R}_n$. Množico primerov lahko torej zapišemo kot množico vektorjev $\mathbf{D} = \{\mathbf{X}_i = [x_1, x_2, ..., x_n]^T \in \mathcal{X}; i \in \{1, 2, ..., m\}\}$. Vektor $\mathbf{X}_i$ včasih označimo z $\vec{x}_i$.

Pogosta naloga strojnega učenja je *klasifikacija*, t. j. pripis *razreda* y_i vsakemu primeru. Definiramo množico k razredov $C = \{C_1, C_2, ...C_k\}$ in *klasifikator* f, ki slika iz prostora instanc v normaliziran k dimenzionalen prostor: $f : \mathcal{X} \longrightarrow [0, 1]^k$. Klasifikator je vrsta *odločitvenega modela*, ki ga v splošnem označimo z Φ. Običajen klasifikator slike v vektorje, ki imajo eno komponento enako 1, ostale pa 0. Poznamo še *verjetnostni klasifikator* h, katerega slike so verjetnostne porazdelitve razredov (vsota komponent vektorja je 1). Primer s pripadajočim razredom se imenuje *označen primer*, zapišemo ga kot par $(\vec{x}_i, y_i)$.

Podatkovni tokovi

Primeri v podatkovnem toku imajo poleg prostorskih dimenzij tudi časovno – vrstni red primerov je pomemben. *Podatkovni tok* označenih primerov v *trenutku* t lahko torej zapišemo kot zaporedje $E_t = (\vec{x}_i, y_i | y_i = f(\vec{x}_i))_t$, $i \in \{1, 2, ..., t\}$. Nimamo več statične množice, marveč zaporedje, čigar dolžina se spreminja.

Ena ključnih značilnosti podatkovnih tokov je, da jih ponavadi ne generira le ena porazdelitev oziroma za njimi ne leži le en koncept. Pojav definiramo v razdelku 2.3.1, značilnosti učnih algoritmov, primernih za podatkovne tokove pa obravnavamo v razdelku 2.4.

2.3 Sprememba koncepta

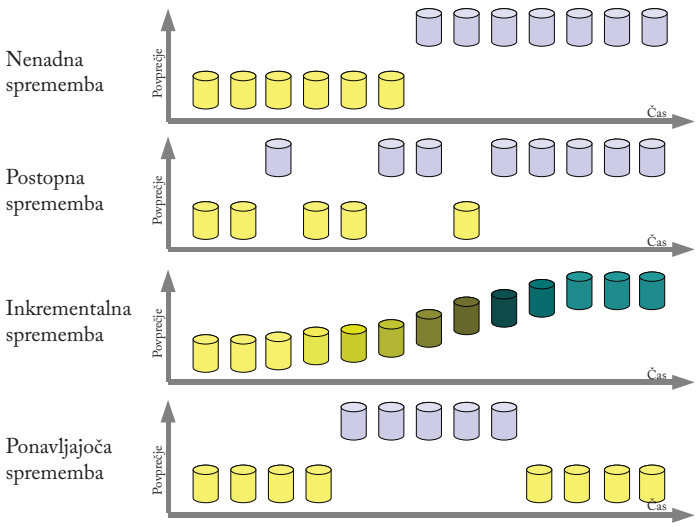
Ena ključnih lastnosti podatkovnih tokov, ki jih loči od statičnih učnih množic, je možnost, da vključujejo *spremembo koncepta* (*concept drift*) [11] – porazdelitev, ki generira primere v podatkovnem toku, se spremeni. La-

stnost je značilna za večino podatkovnih množic, zajetih čez čas, saj imajo primeri takrat dinamično naravo. Pri tem moramo ločevati prave spremembe od šuma, anomalij in izjemnih odstopanj. Z drugimi besedami, koncept podatkov, ki nas zanima, ni statičen, v danem trenutku starejši primeri v toku niso nujno več relevantni za trenutno stanje.

Pozorni moramo biti na dve dimenziji tega problema – na *vzroke* in *hitrost* sprememb (slika 2.1). Pri vzrokih [17] moramo upoštevati, da primere v toku opišemo s končno množico atributov, za katero ni nujno, da zmore dovolj natančno posnemati resnično stanje – možen je obstoj skritih spremenljivk, ki spreminjajo koncept, medtem ko zmoremo z algoritmi analizirati le opazovane vrednosti. Sprememba koncepta je lahko *nenadna* (*sudden*, tako spremembo imenujemo tudi *concept shift*) ali pa *postopna* (*gradual*), pri čemer je slednjo težje zaznati, saj se primeri iz novega koncepta dalj časa mešajo s primeri iz starega koncepta. Poznamo še *inkrementalno* (*incremental*) spremembo, kjer je prehod med koncepti skoraj zvezen (na daljšem intervalu) in *ponavljajočo* (*reoccuring*) spremembo, za katero je značilno, da se stari koncepti neperiodično ponovno pojavljajo. Pri vseh vrstah je izziv najti točke teh sprememb z določeno gotovostjo in primerno prilagoditi učni algoritem. Ta ključna lastnost vpliva ne le na same algoritme za učenje, marveč jo bomo morali upoštevati tudi pri algoritmih za razlago napovedi.

2.3.1 Formalizacija

Opazujemo zaporedje primerov $e_i = (\vec{x}_i, y_i)$ pri nadzorovanem učenju, torej s k razredi $y_i \in \{C_1, C_2, ..., C_k\}$, kjer ob vsakem času t algoritem vrne napoved y_t za primer x_t. Če predpostavimo, da so primeri neodvisni in slučajno generirani z eno samo porazdelitvijo $\mathcal{D}$, lahko slednjo ocenimo s poljubno natančnostjo, ko povečujemo število primerov proti neskončnosti. To je bistvo modela strojnega učenja *PAC* [11] (*Probably approximately correct learning – verjetnostno približno pravilno učenje*). Klasifikacijska napaka je vedno navzdol omejena z Bayesovo napako [3] (H_i je območje, ki ga klasifikator f



Slika 2.1: Vrste sprememb koncepta [17].

klasificira kot C_i):

$$\sum_{i=1}^{k} \int_{x \in H_i} P(C_i|x)p(x)dx \tag{2.1}$$

Če porazdelitev $\mathcal{D}$ ni statična, si lahko podatkovni tok predstavljamo kot zaporedje $< S_1, S_2, ..., S_l >$, pri čemer je vsak S_i zaporedje primerov, ki ga generira porazdelitev D_i in ga imenujemo *kontekst*. Taki skupini zaporedij se ne moremo približati na razdaljo Bayesove napake, lahko se pa ob zadostni količini primerov za posamezen S_i približamo D_i. Spremembo koncepta predstavimo kot spremembo v verjetnosti $P(\vec{x}, y)$:

$$P(\vec{x}, y) = P(y|\vec{x}) \times P(\vec{x})$$

Če verjetnost klasifikacije v razred y zapišemo kot aposteriorno verjetnost:

$$p(y|\vec{x}) = \frac{P(y)p(\vec{x}|y)}{p(\vec{x})},$$

lahko razberemo tri vire za možno spremembo koncepta:

1. apriorne verjetnosti razredov $P(y)$,

2. porazdelitve razredov $p(\vec{x}|y)$,

3. aposteriorne porazdelitve vsebovanosti v razredih $p(y|\vec{x})$.

Problematično pa je iskanje točk, kjer se sprememba zgodi, saj je prehod med zaporednima porazdelitvama lahko dolg (postopen) – primeri naslednje porazdelitve glede na trenutno delujejo kot šum. Algoritmi za zaznavo sprememb morajo torej biti *robustni*, da zmanjšajo vpliv šuma in *občutljivi* na resnične spremembe koncepta. Zaznava spremembe, ki se zgodi nenadno, je lažja in zahteva manj primerov, medtem ko za bolj postopne spremembe potrebujemo več primerov.

Upoštevati moramo tudi možnost, da morebitna sprememba porazdelitve ne pomeni tudi spremembe koncepta – v tem primeru govorimo o *navidezni spremembi koncepta* (*virtual concept drift*), tj. spremembi $p(\vec{x}|y)$ neodvisno od $p(y|\vec{x})$. Tak koncept je *dosleden* (*consistent*) – sprememba med koncepta med zaporednima primeroma ni prevelika, a ni *vztrajen* (*persistent*) – dosleden v več kot polovici primerov v oknu. V primeru, da izpolnjuje zadnja pogoja, govorimo o *pravi* (*permanent, real*) spremembi koncepta, tj. spremembi $p(y|\vec{x})$. V praksi moramo, ne glede na vrsto spremembe koncepta, reagirati in prilagoditi učni algoritem.

2.3.2 Posledice

Posledice za učni model, ki jih sprememba koncepta povzroči, lahko razdelimo na več komponent [5], ki obenem predstavljajo tudi razlike od statičnih modelov:

- shranjevanje podatkov,
- metode zaznave spremembe konceptov,
- prilaganje na spremembe.

Shranjevanje podatkov

Zaradi sprememb koncepta moramo primerno prilagoditi shranjevanje podatkov, torej poleg zaznave spremembe koncepta implementirati tudi *mehanizem pozabljanja*. Če želimo obdržati zadostno statistiko vseh primerov (**popoln spomin**), uporabimo utežitve primerov, ki povzročijo, da imajo starejši manjšo veljavo. Uvedemo lahko *faktor pozabljanja* (*fading factor*) $\alpha \in (0, 1)$, s katerim ob vsakem prihodu novega primera v trenutku t množimo vse stare. Možna je tudi raba *eksponente funkcije staranja* $w_\lambda(\vec{x}) = exp(-\lambda t)$. Določitev vrednosti parametrov α oziroma λ ni trivialna naloga.

Če se odločimo nekatere primere popolnoma zavreči (**delni spomin**), uporabimo drseča okna, ki delujejo na principu vrste (*fifo*: *first-in first-out*) oziroma podzaporedja podatkovnega toka. Srečamo se s problemov velikosti okna – lahko je fiksna (slepo zavržemo dovolj stare primere ali ob določenem času ali ob številu primerov) ali prilagodljiva. V slednjem modelu se okno oži ob zaznavi spremembe koncepta (s tem postane bolj občutljivo za odstopanja, potrebujemo metodo zaznave) in širi ob statični porazdelitvi, kar zagotavlja stabilne rezultate. Uporaba drsečih oken je primerna za podatkovne tokove, kjer pričakujemo nenadne spremembe koncepta.

Algoritem *ADWIN* (*adaptive sliding window*) vzdržuje prilagodljivo drseče okno maksimalne velikosti, v katerem so primeri, ki so statistično skladni s hipotezo, ki pravi, da se povprečna vrednost primerov v oknu ni spremenila (algoritem 1). Zato je uporaben za zaznavo sprememb koncepta (okno se oži le ob statistično značilni spremembi v podatkovnem toku) in hkrati rešuje problem približka povprečne vrednosti v toku. Vhodna podatka sta podatkovni tok in stopnja zaupanja $\delta \in (0, 1)$, tako da je test pravzaprav neparametričen. Koncept algoritma je primerjanje povprečnih vrednosti – kadarkoli sta dve dovolj veliki podokni okna W v povprečni vrednosti dovolj razlikujeta, starejše zavržemo. Verjetnost lažnega alarma (*napaka prvega tipa*) je največ δ, medtem ko je verjetnost *napake drugega tipa* $1 - \delta$.

Zaradi omejenih virov je uporabno še **povzemanje** podatkov, kamor štejemo tehnike vzorčenja, izgradnje histogramov in raznih vrst transforma-

Algoritem 1: ADWIN

Data: Podatkovni tok realnih števil $(x)_t, \forall i \in \{1, 2, ..., t\} : x_i \in (0, 1)$

Stopnja zaupanja $\delta \in (0, 1)$

begin

Inicializraj okno W;

foreach $(t) > 0$ **do**

$W \leftarrow W \cup \{x_t\}$

repeat

Reži stare elemente iz repa W

`// Harmonično povprečje`

$m \leftarrow \frac{2}{1/|W_0|+1/|W_1|}$

$\epsilon_{rez} \leftarrow \sqrt{\frac{1}{2m} \cdot \ln \frac{4|W|}{\delta}}$ `// Meja, ki določa pojma 'dovolj veliko podokno' in 'dovolj velika razlika', izhaja iz Hoeffdingove meje (enačba 2.2)`

until

$|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| < \epsilon_{rez}$ *za vsako delitev okna* W *na* $W = W_0 \cdot W_1$;

Vrni $\hat{\mu}_W$ `// Povprečna vrednost v oknu`

end

end

cij. V vsakem primeru je cilj dobiti čim bolj reprezentativen povzetek.

Metode zaznave spremembe koncepta

Metode lahko delujejo na principu **opazovanja indikatorjev natančnosti učnega algoritma** ali z **primerjanjem porazdelitev dveh oken** – referenčnega, ki povzema stare informacije in aktivnega, ki vsebuje najnovejše primere.

Algoritem CUSUM (Cumulative Sum) je klasičen primer iz prve skupine, saj vrne opozorilo, ko povprečje vhodnih podatkov (residuali ali napake učnega algoritma) statistično značilno odstopa od 0 (algoritem 2).

Algoritem 2: CUSUM

Data: Podatkovni tok residualov $(r)_t$
Prag opozorila λ
Občutljivost (dovoljena velikost spremembe) v
`// Manjši` v `pomeni hitrejšo zaznavo spremembe in večjo možnost lažnega alarma`
begin
  $g_0 = 0$
  $g_t = max(0, g_{t-1} + (r_t - v))$
  if $g_t > \lambda$ **then**
    ALARM
    `// Mesto, kjer moramo ustrezno reagirati na spremembo`
    $g_0 = 0$
  end
end

Delo istega avtorja je tudi inačica slednjega algoritma, ki se v praksi obnese bolje – Page-Hinkley test (algoritem 3) je inkrementalna adaptacija modela zaznave spremembe povprečja v Gaussovem šumu. V praksi ga največkrat srečamo v metodah za zaznavo spremembe (slika 2.2) pri pro-

cesiranju signalov. Algoritem deluje še bolje, če vanj vključimo faktor pozabljanja $\alpha \in (0, 1)$.

Algoritem 3: Page-Hinkley test

Data: Podatkovni tok residualov $(r)_t$

Prag opozorila λ

Občutljivost (dovoljena velikost spremembe) δ

Faktor pozabljanja α

begin

`// t je čas zadnjega prispelega podatka`

$\overline{r_t} = \frac{(t-1)\times\overline{r_{t-1}}+r_t}{t}$ `// Inkrementalen izračun povprečja`

$m_t = \alpha \times m_{t-1} + (r_i - \overline{r_t} - \delta)$

$M_t = min(m_i, i = 1, 2, ..., t)$

$PH_t = m_t - M_t$ `// Page-Hinkley statistika`

if $PH_t > \lambda$ **then**

ALARM

`// Mesto, kjer moramo ustrezno reagirati na spremembo`

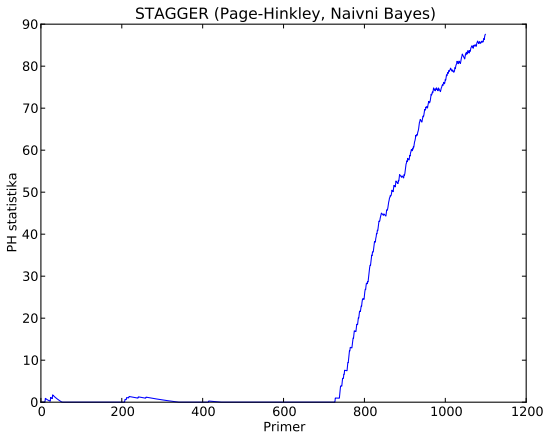
end

end

Obstaja še veliko drugih algoritmov za zaznavo spremembe koncepta [2, 11] (FLORA, FLORA2, Fixed Cumulative Window Model, Early Drift Detection Method, meta učenje Splice, uporaba podpornih vektorjev, itd.), ki jih tu ne bomo obravnavali.

Reagiranje na spremembe

Da bi ohranili točnost napovedi, ob določenih časih uporabimo primerno **prilagoditveno metodo**, ki popravi odločitveni model. Pri tem ločujemo *slepe metode*, ki model spreminjajo ob rednih intervalih (drseče okno fiksne velikosti), in *informirane metode*, ki model spremenijo le po zaznani spremembi, torej potrebujejo metodo zaznave spremembe koncepta.



Slika 2.2: Page-Hinkley statistika za uporabo klasifikatorja naivni Bayes na podatkovnem toku STAGGER (razdelek 5.2), $\alpha = 0.999$. Slika prikazuje zaznavo prve spremembe koncepta – iz strmine grafa razberemo njeno hitrost.

Upravljanje skupine odločitvenih modelov

Pogost pristop za prilagajanje na spremembo koncepta je uporaba skupine odločitvenih modelov (*classifier ensemble*), kar sovpada s predpostavko večjega števila porazdelitev in neostrih prehodov med njimi [17]. Znan primer takšnega algoritma je *Dynamic Weighted Majority* (*DWM*). DWM vzdržuje množico odločitvenih modelov (*ekspertov*), ki glasujejo o napovedi. Glasovi so uteženi glede na uspešnost napovedi – ekspertom, ki glasujejo napačno se utež zmanjša za faktor β, v primeru napačne napovedi celotne množice pa se doda novega. Tako dinamično vzdržuje množico odločitvenih modelov, ki niso ustvarjeni naenkrat in imajo zato različne učne množice. Napaka množice odločitvenih modelov se zmanjša (monotono pada z naraščajočim številom modelov) natanko tedaj, ko ima vsak model boljšo natančnost kot naključna izbira in vsi modeli delujejo neodvisno.

2.3.3 Statistično obvladovanje procesov

Inkrementalni model učenja klasificira vsak primer, takoj ko je ta na voljo, kasneje pa dobi povratno informacijo o svoji uspešnosti in se ustrezno prilagodi. Če je porazdelitev, ki generira primere v podatkovnem toku statična, se bo, pod predpostavko modela PAC, z naraščajočim številom primerov i napaka (verjetnost napačne klasifikacije) p_i manjšala, natančneje, konvergirala bo proti Bayesovi napaki (enačba 2.1). Statistično značilen porast napake si torej lahko razlagamo kot spremembo porazdelitve (spremembo koncepta).

Vsak nov primer podatkovnega toka $(\vec{x_i}, y_i)$ odločitveni model klasificira bodisi pravilno (1) $(\hat{y_i} = y_i)$ bodisi nepravilno (0) $(\hat{y_i} \neq y_i)$, kar nam da Bernoullijevo zaporedje poskusov, torej napaka za i-ti element znaša p_i s standardno deviacijo $s_i = p_i(1 - p_i)/i$. Pri zadostnem številu primerov ($n > 30$) po centralno limitnem izreku vsota tega zaporedja (vsota indikatorjev) konvergira proti normalni porazdelitvi. Z stopnjo zaupanja $1 - \alpha/2$ lahko zato ocenjujemo p z intervalom $p_i+ = z * s_i$, pri čemer je z ustrezen kvantil normalne porazdelitve.

Za zaznavo spremembe koncepta (algoritem 4) vzdržujemo statistiki p_{min} in s_{min} ter za vsak primer j definiramo tri stanja sistema:

1. *Pod nadzorom* (*In-Control*): $p_j + s_j < p_{min} + \beta * s_{min}$
2. *Izpod nadzora* (*Out-of-Control*): $p_j + s_j \geq p_{min} + \alpha * s_{min}$
3. *Opozorilo* (*Warning*): sistem med stanji 1. in 2.

Če sistem prekorači 1. mejo, je v stanju *opozorila*, a še ne moremo zanesljivo trditi, da se je porazdelitev, ki generira predmete spremenila – napaka se lahko zmanjša in smo spet v stanju *pod nadzorom*. Če pa sistem prekorači 2. mejo, so napake statistično značilne – s stopnjo zaupanja $1 - \alpha/2$ lahko trdimo, da se je porazdelitev spremenila. Stopnja *opozorilo* nam na ta način določa velikost drsečega okna in posledično tudi nakazuje hitrost sprememb. α in β določimo tako, da rešimo enačbo $n = \sqrt{2}erf^{-1}(P)$, če intervale zaupanja izražamo s standardnimi deviacijami – meritve padejo v

interval $[\mu - x_n, \mu + x_n]$, $x_n = n\sigma$. Ponavadi uporabimo stopnji 0.95 in 0.99, kar da približno $\alpha = 2$ in $\beta = 3$. SPC je zanimiv tudi zato, ker ga lahko uporabimo kot *ovojnico* (angl. *wrapper*), s katero prevedemo statični odločitveni model na inkrementalnega.

2.4 Strojno učenje na podatkovnih tokovih

Pri snovanju algoritmov za strojno učenje na podatkovnih tokovih pogosto izhajamo iz problem in tehnik, s katerimi se srečujemo pri statičnih množicah. Te metode je potrebno prilagoditi značilnostim podatkovnih tokov (razdelek 2.1) ali pa na podlagi slednjih razviti nove modele. Inkrementalno učenje je potreben, a ne zadosten pogoj za uspešen algoritem na podatkovnih tokovih. Nekateri algoritmi strojnega učenja so že v osnovi inkrementalni (k-najbližjih sosedov, naivni Bayes), drugi očitno ne (odločitvena drevesa). Upoštevati je potrebno še spremembo koncepta, pozabljanje nerelevantnih primerov in prilaganje na tekoče stanje. Pomemben koncept pri izbiri osnovnega algoritma je *granularnost* (*granularity*) – spremembe v podatkovnih tokovih ne zadevajo celotne množice primerov, marveč le njen del. Zaželen je torej granularen algoritem, ki je zmožen prilagoditi le dele odločitvenega modela brez celotne rekonstrukcije.

Med pogoste naloge strojnega učenja na podatkovnih tokovih spadajo:

- **odkrivanje skupin** (uvrščanje primerov v enote s skupnimi značilnostmi),
- **napovedna analitika** (napovedovanje prihodnjih vrednosti in izjemnih odstopanj za različne intervale v prihodnosti, iskanje periodičnosti in vzorcev),
- **analitika sprememb** (zaznava sprememb koncepta, odpovedi strojne opreme, odrivanje kritičnih točk in odstopanj, splošno odkrivanje sprememb in njih klasificiranje).

Algoritem 4: SPC za zaznavo spremembe koncepta

Data: Podatkovni tok $(\vec{x}_i, y_i)_t$

Odločitveni model Φ

begin

$(\vec{x}_j, y_j) \leftarrow$ trenutni primer

$\hat{y}_j \leftarrow \Phi(\vec{x}_j)$

$napaka_j \leftarrow L(\hat{y}_j, y_j)$

$p_j \leftarrow$ *povprecna vrednost napak*

$s_j \leftarrow$ *standardna deviacija napak*

if $p_j + s_j < p_{min} + s_{min}$ **then**

`// Posodobitev potrebnih statistik`

$p_{min} \leftarrow p_j$

$s_{min} \leftarrow s_j$

end

if $p_j + s_j < p_{min} + \beta * s_{min}$ **then**

`// Pod nadzorom`

$Opozorilo \leftarrow FALSE$

Posodobitev TRENUTNEGA odločitvenega modela z $(\vec{x_j}, y_j)$

else

if $p_j + s_j < p_{min} + \alpha * s_{min}$ **then**

`// Opozorilo`

if *NOT Opozorilo* **then**

$buffer \leftarrow (\vec{x_j}, y_j)$ `// buffer - medpomnilno okno`

$Opozorilo \leftarrow TRUE$

else

$buffer \leftarrow buffer \cup (\vec{x_j}, y_j)$

end

else

`// Izpod nadzora`

Izdelaj NOV odločitveni model $\Phi(buffer)$

$Opozorilo \leftarrow FALSE$

$p_{min} = \inf$

$s_{min} = \inf$

end

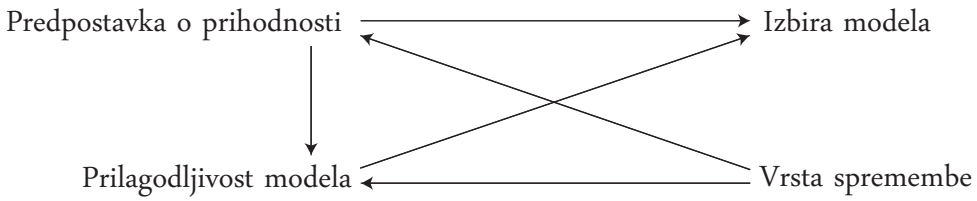
end

end

2.4.1 Inkrementalni model učenja

Pri inkrementalnem učenju (klasifikaciji primerov) na podlagi podatkovnega toka (zgodovinskih podatkov) $E_t = (\vec{x}_i, y_i)_t$ ali njegovega podzaporedja, določamo (napovedujemo) razred $\hat{y}_{t+1}$ za novoprispeli primer $\vec{x}_{t+1}$. Ko po času l izvemo razred, označen primer vključimo v zgodovinske podatke in izračunamo *napako napovedi* (*izguba točnosti, loss of prediction*) $L(\hat{y}_{t+l}, y_{t+l})$. Učna množica se torej stalno spreminja (lahko z veliko hitrostjo), zato od učnega modela zahtevamo inkrementalnost, takojšnjo odzivnost, konstanten čas za obravnavo posameznega primera ($O(1)$), le en prehod čez učno množico, ustrezno reagiranje na spremembe koncept in učinkovito metodo aproksimacije rezultatov. Izzivi ležijo v upravljanju podatkov v modelu in v spremembah koncepta. Opravka imamo z neomejenimi tokovi in omejenimi viri (pomnilnik, pasovna širina, procesorska moč, dostop do podatkov, odzivni časi) ter problemom porazdeljenih podatkovnih tokov.

Inkrementalen učni model torej iz zaporedja $\{..., E_{j-1}, E_j, ...\}$ zgradi zaporedje *hipotez* $\{..., H_{j-1}, H_j, ...\}$, pri čemer je hipoteza H_i odvisna le od H_{i-1} in E_i. Potrebujemo dva operatorja – *inkrement* (primer E_k vključimo v model) in *dekrement, operator pozabljanja* (primer E_k izključimo iz modela, ga pozabimo). Ne dovolimo *blokadnih* (*blocking*) operatorjev, kot je npr. transponiranje, saj ti ne dovoljujejo algoritmu nadaljnjega poteka, dokler se sami ne končajo.



Slika 2.3: Komponente inkrementalnga modela učenja

Zasnovo inkrementalnega modela učenja [17] lahko strnemo v 4 naloge (slika 2.3):

1. **Predpostavka o prihodnosti** – koncept za primerom $\vec{x}_{t+1}$ ni znan, imamo pa več možnih predpostavk:
   (a) nespremenjen koncept $\mathcal{D}_{t+1} = \mathcal{D}_t$ (najpogostejša predpostavka),
   (b) določitev koncepta na podlagi primera $\vec{x}$ (ponavadi z merjenjem razdalje do primerov v zgodovinskih podatkih),
   (c) predvidevanje spremembe.
2. **Vrsta spremembe** – upoštevamo vzroke in hitrost (obravnava v razdelku 2.3)
3. **Prilagodljivost modela** – odvisna od predpostavke o prihodnosti in vrste spremembe, razdelimo jo lahko na štiri področja:
   (a) prilagodljivost osnovnih algoritmov (npr. dodajanje primera pri naivnem Bayesu),
   (b) parametrizacija osnovnih algoritmov (npr. dodajanje faktorjev pozabljanja),
   (c) upravljanje z učno množico (npr. dodajanje, odvzemanje primerov in šuma, izbor atributov, drseča okna),
   (d) skupinske odločitve (npr. uporaba DWM).
4. **Izbira modela** – izbira parametrizacije ob vsakem novoprispelem primeru za čim manjšo možnost napake. Pristop je odvisen predvsem od predpostavke o prihodnosti – lahko poizkušamo izračunati teoretično napako ali pa njen približek z uporabo *navzkrižnega preverjanja* (*cross validation*).

Glavni vrsti prilaganja na spremembe sta upravljanje učne množice in parametrizacija učnega modela. Za razvoj inkrementalnega modela učenja pa sta ključni predvsem vrsta spremembe in izbira modela. Na podlagi teh postavk dveh lahko ločimo kategoriji:

1. Modeli, ki **reagirajo** na indikator so odvisni od vrste spremembe. Indikator (ponavadi metoda zaznave spremembe koncepta, več v razdelku 2.3.2) neposredno vpliva na strukturo učnega modela.

2. Modeli, ki **se razvijajo** ob rednih intervalih neodvisno od indikatorjev in nimajo neposredne povezave med podatkovnim tokom in izgradnjo, ne zaznavajo sprememb. Da bi vseeno ohranjali natančnost napovedi, se uporabljajo skupine klasifikatorjev, mehanizmi za izgradnjo prototipov, uteževanje primerov, izbor atributov in upravljanje s parametri, ki so specifični za osnovni učni model.

2.4.2 Učni algoritmi za klasifikacijo primerov v podatkovnem toku

Večino učnih algoritmov, ki jih uporabljamo za klasifikacijo primerov v statičnih množicah, je možno prirediti za podatkovne tokove [5]. Pri tem se osredotočamo predvsem na vpeljavo inkrementalnosti oziroma granularnosti v obstoječ algoritem. V tem razdelku predstavimo nekaj najpogostejših naravno inkrementalnih in prilagojenih klasifikacijskih učnih algoritmov (*klasifikatorjev*).

Naivni Bayes

Naivni Bayes je naravno inkrementalen verjetnostni algoritem, ki predpostavlja medsebojno neodvisnost atributov. Osnova metode je Bayesova formula:

$$p(C_j|x_1, x_2, ..., x_n) = \frac{p(C_j)p(x_1, x_2, ..., x_n|C)}{p(x_1, x_2, ..., x_n)}$$

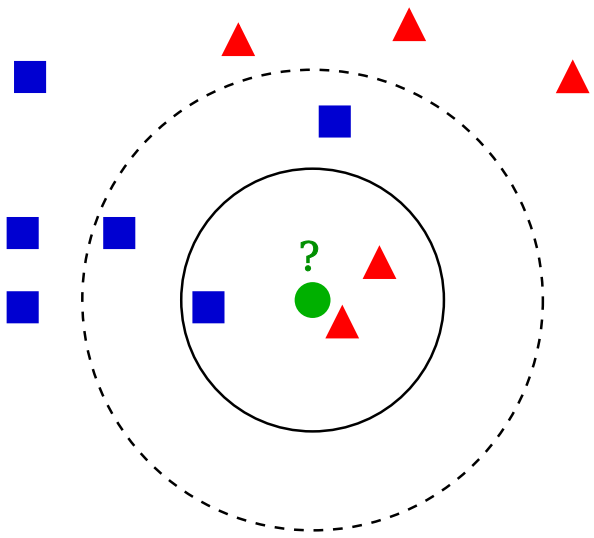
pri čemer so $x_1, x_2, ..., x_n$ vrednosti atributov pri primeru $\mathbf{X}$ za posamezen razred $C_j, j \in \{1, 2, ..., k\}$. Upoštevaje predpostavke neodvisnosti z uporabo verižnega pravila za pogojne verjetnosti dobimo verjetnost klasifikacije v razred C_j:

$$p(C_j|\mathbf{X}) = p(C_j) \prod_{i=1}^{n} p(x_i|C_j)$$

Učni algoritem izračunava verjetnosti na desni strani enačbe na podlagi učne množice. Primeru priredimo razred, pri katerem je dosežena največja aposteriorna verjetnost $p(C_j|\mathbf{X})$.

k-najbližjih sosedov

k-najbližjih sosedov (*k-NN – k-nearest neighbours*) je algoritem, ki uporablja *leno učenje*[1] (*lazy learning*) in je naravno inkrementalen. Za dani primer $\mathbf{X}$ v učni množici poišče k najbližjih sosedov glede na določeno mero razdalje in mu priredi njihov večinski (modus) razred (slika 2.4). Pogosto sosede obtežimo z razdaljami do primera in s tem povečamo klasifikacijsko točnost.



Slika 2.4: k-najbližjih sosedov: klasificiramo primer, označen z zelenim krogom. Pri $k = 3$ (polna črta), ga uvrstimo v razred rdečih trikotnikov, pri $k = 5$ (črtkana črta) pa v razred modrih kvadratov [18].

[1]Znanje iz podatkov dobimo ob zahtevani poizvedbi, sicer metoda le vzdržuje učno množico v spominu.

Hoeffdingova drevesa

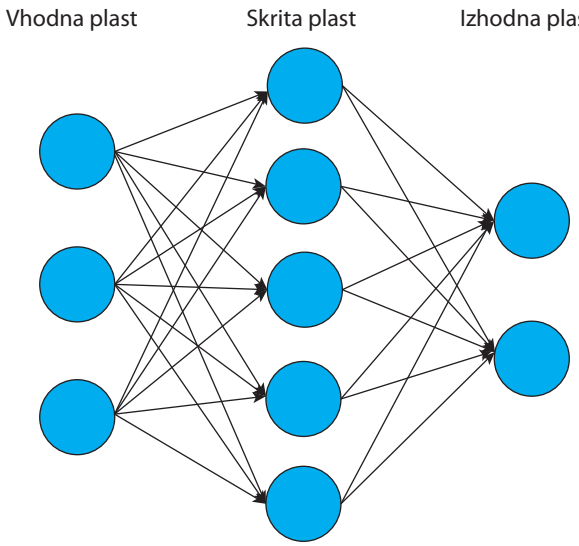
Ker običajna odločitvena drevesa niso inkrementalna, za uporabo na podatkovnih tokovih le-ta niso primerna. Hoeffdingova drevesa izkoriščajo dejstvo, da je relativno majhno število primerov dovolj za statistično utemeljeno odločitev o deljenju lista (širitvi drevesa z odločitvenim vozliščem). Ime so dobili po Hoeffdingovi meji, ki je kriterij za delitev:

$$\varepsilon = \sqrt{\frac{R^2 ln(\frac{2}{\delta})}{2n}} \tag{2.2}$$

Torej, če je razlika kriterijskih funkcij G (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. Vozlišča v drevesu so hiperpravokotniki, ki razdelijo prostor instanc na vgnezdena območja (koren pokriva celoten prostor). To zagotovi granularnost, potrebno za odziv na nenadne in postopne spremembe v toku – prilagodimo lahko le del drevesa. Izboljšava algoritma so funkcionalni listi, ki vsebujejo naivni Bayesov klasifikator, naučen na primerih v posameznem listu. Uporabimo ga za določitev razreda novega primera (namesto modusa razredov primerov v listu). Znan algoritem za gradnjo Hoeffdingovih dreves je VFDT (*Very Fast Decision Tree Algorithm*).

Umetne nevronske mreže

Umetna nevronska mreža je sestavljena iz med seboj povezanih nevronov, ki so organizirani v plasti (acikličen graf). Poznamo več vrst glede na število plasti, usmerjenost povezav in namen uporabe. Troplastna mreža (slika 2.5) vsebuje vhodno, skrito in izhodno plast. Učenje na njej poteka s spreminjanjem uteži na povezavah glede na vhodne podatke, katerim se poizkuša prilagoditi. Nevronske mreže so znane kot robustna (glede na manjkajoče podatke in napake), prilagodljiva in natančna metoda, ki je uporabna predvsem pri prepoznavanju vzorcev. Avtomatsko pozablja nerelevantne podatke in se inkrementalno prilagaja okolju. Z njo lahko dobimo poljubno natančen približek katerekoli zvezne funkcije.



Slika 2.5: Troplastna umetna nevronska mreža

2.4.3 Mere uspešnosti klasifikatorjev

Na področju strojnega učenja je pravilno ocenjevanje kvalitete napovedi pomembna komponenta celotnega sistema. To pride posebno do izraza pri podatkovnih tokovih, kjer lahko z opazovanjem indikatorjev kvalitete napovedi skozi čas določamo mesta sprememb koncepta[2].

Za izpeljavo mere moramo poznati vrste napak[3], ki jih predstavimo v tabeli 2.1.

Od tod izpeljemo mere uspešnosti:

- *senzitivnost* (*priklic*) – delež pravilno klasificiranih pozitivnih primerov (stopnja pravilne potrditve): $\frac{TP}{P} = \frac{TP}{TP+FN}$,
- *specifičnost* – delež pravilno klasificiranih negativnih primerov (stopnja pravilne zavrnitve): $\frac{TN}{N} = \frac{TN}{TN+FP}$,

[2]Več v razdelku 2.3.2.

[3]Predpostavljamo binarne razrede.

Tabela 2.1: Tabela napak (*kontingenčna tabela, confusion matrix*)

		Dejanska vrednost		
		+	-	
Napovedana vrednost	+	**TP** (*true positive*) zadetek	**FP** (*false positive*) lažni alarm, napaka I reda	P'
	-	**FN** (*false negative*) pogrešek, napaka II. reda	**TN** (*true negative*) pravilna zavrnitev	N'
		P (dejansko pozitivni)	N (dejansko negativni)	

- *preciznost* – delež pravilno klasificiranih, ki so bili klasificirani kot pozitivni: $\frac{TP}{P'} = \frac{TP}{TP+FP}$,
- *delež pravilnih negativnih*: $\frac{TN}{N'} = \frac{TN}{TN+FN}$,
- *klasifikacijska točnost*[4] *(CA)* – delež pravilno klasificiranih primerov: $\frac{TP+TN}{P+N}$,
- *mera F1 (F1 score)* – lahko jo uporabimo kot eno mero, ki združuje ostale in s tem odpravimo potrebo po večih izračunih. Združuje preciznost in senzitivnost: $2 \cdot \frac{preciznost \times senzitivnost}{preciznost+senzitivnost} = 2\frac{TP^2}{P+P'}$,
- *Matthewsov korelacijski koeficient* – ena najboljših mer, ki dobro opiše celotno kontingenčno tabelo (popoln opis ni možen), saj upošteva pravilne in nepravilne klasifikacije pozitivnih ter negativnih primerov: $\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$.

[4]Mera je nezanesljiva v primeru, da imata razreda zelo različno število primerov.

Poglavje 3

Razlaga napovedi

V strojnem učenju podatke pred vnosom v učnih algoritem najprej pošljemo čez proces *predobdelave* (*preprocessing*), ki zajema urejanje in čiščenje podatkov, izbor informativnih atributov, diskretizacijo, normalizacijo, razrešitev problema manjkajočih ter konfliktnih vrednosti, itd. Rezultat predobdelave je *učna množica* (*training set*), ki jo uporabimo kot vhodni podatek za algoritem. S temi postopki se trudimo izboljšati kvaliteto (natančnost) napovedi[1]. Učni model vrne napoved, ki jo pogosto želimo *razložiti* – za končnega uporabnika je proces strojnega učenja črna škatla, zato je razlaga/utemeljitev napovedi želena komponenta sistemov za podporo odločanju. Posledično ima uporabnik večje zaupanje v odločitev (kombinirano z ekspertnim znanjem) in pridobi dodatno znanje o napovedi ali o celotnem modelu. Razlaga pridobljenega znanja je ena izmed vrst *poobdelave* (*postprocessing*) podatkov – poznamo še integracijo, filtriranje in evalvacijo.

3.1 Obstoječe metode razlage za statične množice

Za večino modelov strojnega učenja že obstaja način razlage. V nekaterih primerih je že narava modela transparentna in je odločitev možno neposredno razložiti, npr. z vozlišči v odločitvenih drevesih in v Bayesovih verjetnostnih

[1]Spomnimo se načela *garbage in – garbage out.*

mrežah. Razlago lahko ustvarimo tudi z vizualizacijami (za metodo podpornih vektorjev, nomogrami za metodo naivnega Bayesa...), izpeljavo pravil in s postopki, prirejenimi za specifične modele. V našem primeru razvijamo posplošeno metodo razlage za podatkovne tokove, ki ni odvisna od modela učenja in zato pojasnjuje konsistentno (kar je prikladno za primerjave učnih modelov in primerjave klasificiranih primerov) ter je fleksibilna (ob menjavi učnega modela ni potrebna menjava modela razlage). Izhajali bomo iz del E. Štrumblja, I. Kononenka in M. Robnik-Šikonje, v katerih spoznamo posplošen model razlage klasifikacij primerov v statičnih množicah z interakcijami podmnožic atributov [13] in sorodno učinkovito metodo razlage, ki se opira na pojme iz teorije iger [14]. Obstajajo še druge posplošene metodologije, ki so osnovane na perturbacijah posameznih atributov in opazovanju sprememb v klasifikacijah. Problem nastane pri redundantnih atributih in disjunktnih konceptih, kjer večina metod odpove, ker obravnavajo vsak atribut posebej.

3.2 Razlaga klasifikacije primerov

Klasificirane primere z metodo *IME* (*Interactions-based Method for Explanation*) [13] razlagamo s pomočjo *prispevkov* (*contributions*) atributov. Za razlago potrebujemo klasifikator na učni množici, nov primer in razred. Klasifikator na podlagi vrednosti atributov primera napove razred z določeno verjetnostjo, ki se ponavadi razlikuje od apriorne verjetnosti razreda (klasifikacije primera, v katerem ne poznamo nobene vrednosti atributa). IME vsakemu atributu pripiše realno število – prispevek. Pozitiven predznak prispevka pomeni, da določena vrednost atributa veča verjetnost ciljne klasifikacije (kaže v prid razredu), negativen predznak jo manjša. Absolutna vrednost prispevka je sorazmerna z vplivom (pomembnostjo) vrednosti atributa, vsota vseh prispevkov je pa ravno razlika med verjetnostjo klasificiranega razreda in apriorno verjetnostjo razreda. Da bi zajeli vse možne interakcije in rešili problem disjunkcije, opazujemo potenčno množico vrednosti atributov. Posledica je eksponentna časovna zahtevnost, ki jo bomo pozneje omili z

aproksimacijo, vendar je potrebno poudariti, da je generirana razlaga točna.

3.2.1 Osnovna ideja in formalizacija

Razlagamo razred $c \in C = \{C_1, C_2, ..., C_k\}$. Definiramo *podprostor instanc* $\mathcal{X}_S = \mathfrak{R}_{j_1} \times \mathfrak{R}_{j_2} \times ... \times \mathfrak{R}_{j_p}$ za atribute A_j; $j \in S \subseteq \{1, 2, ..., n\}$. Vse možne podmnožice zajamemo v $\hat{\mathcal{X}} = \bigcup_{W \subseteq \{1,2,...,n\}} \mathcal{X}_W$ in verjetnosti klasifikator $h : \hat{\mathcal{X}} \to [0, 1]$. Za h ni potrebno, da slika v k-dimenzionalni prostor, saj nas zanima razlaga le enega razreda za primer $\vec{x}_S \in \mathcal{X}_S$, zato je $h(\vec{x}_S)$ približek za $P(C = c|\vec{x}_S)$. Problem, ki nastane s klasifikacijo primerov z manjkajočimi in zveznimi atributi, naslovimo kasneje, v razdelku 3.2.2. Vpliv atributov se kaže v *razliki napovedi* (*prediction difference*):

$$\Delta(\{1, 2, ..., n\}) = h(\vec{x}_{\{1,2,...,n\}}) - h(\vec{x}_\emptyset)$$

oziroma za poljubno podmnožico atributov:

$$\Delta(S) = h(\vec{x}_S) - h(\vec{x}_\emptyset) \tag{3.1}$$

V tej vrednosti so zajete tudi interakcije med atributi, ki jih sicer (brez potenčne množice ali z opazovanjem posameznih atributov) ne bi zaznali. Ti prispevki so *interakcijski prispevki* (*interaction contributions*) $\mathcal{I}(\mathcal{W})$. Velja:

$$\Delta(S) = \sum_{W \subseteq S} \mathcal{I}(\mathcal{W})$$

Ker je $\mathcal{I}(\emptyset) = 0$, dobimo rekurzivno definicijo:

$$\mathcal{I}(S) = \Delta(S) - \sum_{W \subset S} \mathcal{I}(\mathcal{W})$$

Da bi dobili končne prispevke, interakcijske prispevke razdelimo na toliko enako velikih delov, kolikor je atributov v posamezni interakciji in nato seštejemo dobljene vrednosti za vsak atribut posebej. Delitev na enake dele pomeni, da ne upoštevamo apriornih verjetnosti in morebitnih odvisnosti, kar pomeni, da je razlaga popolnoma neodvisna od morebitne pristranskosti uporabnika. *Prispevek* vrednosti atributa A_i je torej:

$$\varphi_i(\Delta) = \sum_{W \subseteq \{1,2,...,n\} \setminus \{i\}} \frac{\mathcal{I}(W \cup \{i\})}{|W \cup \{i\}|}$$

3.2.2 Razlaga in teorija iger

S teorijo iger in aproksimacijo rešimo problem manjkajočih vrednosti, zveznih zalog vrednosti atributov ter eksponentne časovne zahtevnosti [14].

Teorija iger

Uporabili bomo koncept *kooperativne igre v koalicijski obliki s prenosljivi sredstvi*. Formalno jo definiramo kot par (N, v). $N = \{1, 2, ..., n\}$ je množica *igralcev*, $v : 2^{[n]} \to \mathbb{R}$ pa *karakteristična funkcija*, ki mora zadostovati pogojema:

1. $v(\emptyset) = 0$
2. $\forall S, T \in 2^{[n]}, S \cap T = \emptyset : \ v(S) + v(T) \leq v(S \cup T)$ (*superaditivnost*)

Množico igralcev $N = \{1, 2, ..., n\}$ imenujemo *polna koalicija*, vrednost $v(N)$ pa *vrednost igre*. Igralci se lahko povezujejo v manjše koalicije $S \subseteq 2^{[n]}$, pri čemer je vrednost posamezne koalicije $v(S)$, vseh možnih koalicij je torej 2^n. Glavna predpostavka pri igrah v koalicijski obliki je, da bo nastala polna koalicija.

Posledično je ključni problem iskanje "pravične" delitve vrednosti igre $v(N)$ oziroma iskanje *vektorja izplačil* $(x_1, x_2, ..., x_n) \in \mathbb{R}^n$, kjer igralec i dobi delež x_i. Pri iskanju tega vektorja lahko upoštevamo razne kriterije (učinkovitost, individualna racionalnost, obstoj, unikatnost, računska zahtevnost ...), vendar večina konceptov rešitve izhaja iz *množice imputacij*. Ta vsebuje vektorje izplačil, ki so *skupno racionalni* (vektor izplačil brez ostanka razdeli vrednost igre) in *individualno racionalni* (vsak igralec dobi vsaj toliko, kot bi dobil, če ne bi bil vključen v koalicijo):

$$\{(x_1, x_2, ..., x_n) \in \mathbb{R}^n | \sum_{i=1}^{n} x_i = v(N), \ \forall i \in N : \ x_i \geq v(\{i\})\}$$

Pogost koncept rešitve, ki zajema ključne kriterije "poštenosti", je *Shapleyeva vrednost*. Iščemo funkcije $\varphi : v(N) \to \mathbb{R}^n$, $\varphi(v) = (\varphi_1(v), \varphi_2(v), ...\varphi_n(v))$, ki zadoščajo lastnostnim:

1. *skupna racionalnost*: $\forall v : \sum_{i=1}^{n} \varphi_i(v) = v(N)$,
2. *simetričnost*: $\forall i, j \in N$, ki zadošča $v(S \cup \{i\}) = v(S \cup \{i\}) \forall S,\ S \not\ni i, j$ velja $\varphi_i(v) = \varphi_j(v)$,
3. *dummy player*: $\forall v : (\forall i \in N : \forall S \ni i : v(S \cup \{i\}) = v(S)) \Rightarrow \varphi_i(v) = 0$,
4. *adititvnost*: $\forall u, v :\ \varphi_i(u) + \varphi_i(v) = \varphi_i(u + v)$.

Shapleyev izrek pravi, da obstaja natanko ena funkcija ϕ, ki zadošča zgornjim pogojem in zanjo velja:

$$\varphi_i(v) = \sum_{\substack{S \subseteq N \\ i \in S}} \frac{(|S| - 1)!(n - |S|)!}{n!} (v(S) - v(S \backslash \{i\})$$

Povezava z metodo razlage

Razložene koncepte lahko apliciramo na problem razlage napovedi, ki ga zapišemo kot igro v koalicijski obliki (N, Δ). Igralci so v tem primeru atributi, karakteristična funkcija pa je razlika napovedi, zato je vrednost igre razlika napovedi, če upoštevamo celotno množico atributov (polna koalicija) $\varphi(\Delta) = (\varphi_1, \varphi_2, ..., \varphi_n)$. Iskanje vektorja izplačil je torej iskanje delitve razlike napovedi, ki bo vsebovala prispevke posameznih atributov. Da bi delitev dobro odsevala resnične vzroke za napovedjo, se lahko poslužimo Shapleyeve vrednosti, ki upošteva določene principe pravičnosti in zato vrne vrednosti ustrezne velikosti in predznaka.

Za uspešno formulacijo metode razlage kot igre je potrebno redefinirati razliko napovedi (3.1):

$$\Delta(S) = \frac{1}{|\mathcal{X}_{N \backslash S}|} \sum_{\vec{y} \in \mathcal{X}_{N \backslash S}} h(\tau(\vec{x}, \vec{y}, S)) - \frac{1}{|\mathcal{X}_N|} \sum_{\vec{y} \in \mathcal{X}_N} h(\vec{y}) \tag{3.2}$$

$$\tau(\vec{x}, \vec{y}, S) = (z_1, z_2, ..., z_n),\ z_i = \begin{cases} x_i\ ;\ i \in S \\ y_i\ ;\ i \notin S \end{cases}$$

Naj bo $\pi(N)$ množica permutacij N, $Pre^i(N)$ pa množica igralcev, ki so predhodniki igralca i v permutaciji $\mathcal{O} \in \pi(N)$. Tako izpeljemo ekvivalentno

izražavo Shapleyeve vrednosti:

$$\varphi_i(\Delta) = \frac{1}{n!} \sum_{\mathcal{O} \in \pi(N)} (\Delta(Pre^i(\mathcal{O}) \cup \{i\}) - \Delta(Pre^i(\mathcal{O})))$$

Aproskimacija

V izogib eksponentni časovni zahtevnosti (skrita v novi definiciji razlike napovedi Δ 3.2) in odvisnosti od metode učenja (skrita v stari definicije razlike napovedi Δ 3.1) za namene dobre aproksimacije z vzorčenjem uporabimo ekvivalentno definicijo razlike napovedi:

$$\Delta(S) = \frac{1}{|\mathcal{X}|} \sum_{\vec{y} \in \mathcal{X}} (h(\tau(\vec{x}, \vec{y}, S)) - h(\vec{y}))$$

Tedaj je Shapleyeva vrednost:

$$\varphi_i(\Delta) = \frac{1}{n! \cdot |\mathcal{X}|} \sum_{\mathcal{O} \in \pi(N)} \sum_{\vec{y} \in \mathcal{X}} (h(\tau(\vec{x}, \vec{y}, Pre^i(\mathcal{O}) \cup \{i\})) - h(\tau(\vec{x}, \vec{y}, Pre^i(\mathcal{O}))))$$

Populacija, ki jo vzorčimo so primeri z vsemi možnimi permutacijami atributov $\pi(N) \times \mathcal{X}$. En vzorec je definiran kot $X_{\mathcal{O}, \vec{y} \in \mathcal{X}} = h(\tau(\vec{x}, \vec{y}, Pre^i(\mathcal{O}) \cup \{i\})) - h(\tau(\vec{x}, \vec{y}, Pre^i(\mathcal{O})))$, pri čemer zveznost atributov ne vpliva na postopek vzorčenja. Definiramo cenilko $\hat{\varphi}_i = \frac{1}{m} \sum_{j=1}^{m} X_j$ za m vzorcev, ki jih izvlečemo z vračanjem. Ker so vzorci enako porazdeljeni (vsakega izvlečemo z verjetnostjo $\frac{1}{n! \cdot |\mathcal{X}|}$) in neodvisni, je po centralno limitnem izreku $\hat{\varphi}_i \dot{\sim} N(\varphi_i, \frac{\sigma_i^2}{m})$. Ker je $E(\hat{\varphi}_i) \to \varphi_i$ in $D(\hat{\varphi}_i) \to 0$, ko $m \to \infty$, je $\hat{\varphi}_i$ nepristranska in dosledna cenilka za φ_i.

Število primerov v vzorcu m je odvisno od disperzije posameznega atributa σ_i^2, ki ima zgornjo mejo $\sigma_i^2 \leq 1$. Ocenimo ga lahko intervalsko z intervalom zaupanja $1 - \alpha$ in maksimalnim odstopanjem e, kjer velja $P(|\hat{\varphi}_i - \varphi_i| < e) = 1 - \alpha$. Omejitev je $m_i = \frac{Z_{1-\alpha}^2 \cdot \sigma_i^2}{e^2}$, kjer je $Z_{1-\alpha}^2$ kvantil normalne porazdelitve. Če predpostavimo najslabši primer ($\sigma_i^2 = 1$), moramo za 99% interval za odstopanje 0.01 vzeti približno $m = 65000$. Da bi to število čim bolj zmanjšali, moramo čim bolje oceniti (lahko že med vzorčenjem) disperzijo z oceno $\overline{\sigma^2} = \frac{1}{n} \sum_{i=1}^{n} \sigma_i^2$, ki skupaj s številom atributov n opiše

kompleksnost modela napovedi in s tem določi minimalno število primerov v vzorcu $m_{min} = n \cdot \frac{Z^2_{1-\alpha} \cdot \overline{\sigma^2}}{e^2}$. Časovna zahtevnost razlage primera $O(n \cdot T(\mathcal{X}))$ je odvisna od števila atributov n in časa, potrebnega za klasifikacijo primera $T(\mathcal{X})$.

Natančnost in kvaliteta razlage

Metode razlage razložijo učni model in njegove odločitve. Ni pa nujno, da s tem dobro razložijo tudi koncepte, ki ležijo za njim, kar je naš cilj. Opisana metoda vrne razlago, katere kvaliteta je pozitivno korelirana z natančnostjo modela. Natančnost modela (*kvaliteta napovedi*) meri odstopanja klasifikacij od dejanskih razredov, *kvaliteta napovedi* pa meri odstopanja generiranih prispevkov od prispevkov pri klasifikaciji z optimalnim Bayesovim klasifikatorjem. Boljši model torej pomeni boljšo razlago.

3.2.3 Algoritma

Postopek razlage posameznega primera povzamemo v algoritmu 5. Uporabili ga bomo tudi pri inkrementalni prilagoditvi. Razlago posameznega primera lahko z uporabo vzorčenja za vse možne vrednosti atributov razširimo na razlago celotnega modela (algoritem 6).

Algoritem 5: Razlaga klasifikacije primera

Data: Klasifikator h

Učni podatki $\mathcal{X}$

Primer $\vec{x} \in \mathcal{X}$

Število primerov v vzorcu m `// Lahko obravnavamo ločeno`

begin

`// Poiščemo prispevke vseh atributov za določen primer`

for $i = 1$ *to* n **do**

$\varphi_i \leftarrow 0$

for $j = 1$ *to* m **do**

izberi permutacijo $\mathcal{O} \in \pi(N)$

naključno izberi primer $\vec{y} \in \mathcal{X}$

$v_1 \leftarrow h(\tau(\vec{x}, \vec{y}, Pre^i(\mathcal{O}) \cup \{i\}))$

$v_2 \leftarrow h(\tau(\vec{x}, \vec{y}, Pre^i(\mathcal{O}))$

$\varphi_i \leftarrow \varphi_i + (v_1 - v_2)$

end

$\varphi_i \leftarrow \frac{\varphi_i}{m}$

end

end

Algoritem 6: Razlaga celotnega modela

Data: Klasifikator h
Učni podatki $\mathcal{X}$
Število primerov m v vzorcu `// Lahko obravnavamo ločeno`
begin
  `// Obravnavamo vse atribute`
  for $i = 1$ *to* n **do**
    `// Pri posameznem atributu iteriramo čez njegovo zalogo vrednosti. Za zvezne atribute izberemo resolucijo in jih tako diskretiziramo na ekvidistančne intervale.`
    $D_i \leftarrow \mathfrak{R}_i$
    if $zvezen(A_i)$ **then**
      $D_i \leftarrow diskretiziraj(\mathfrak{R}_i)$
    end
    for $a \in D_i$ **do**
      `// Razlaga vrednosti` a `atributa` A_i `za cel model`
      $\psi \leftarrow tabela[m]$
      for $k = 1$ *to* m **do**
        `//` $\vec{x}$ `je sestavljen iz naključno izbranih vrednosti atributov, razen za` i`-tega, ki je enak` a`.` $\vec{y}$ `je enak` $\vec{x}$`, razen na` i`-tem mestu, kjer je naključen.`
        $\vec{x} \leftarrow nakljucnoIzbraniAtributi(\mathcal{X})$
        $\vec{y} \leftarrow \vec{x}$
        $\vec{x}[i] = a$
        $\psi[k] \leftarrow h(\vec{x}) - h(\vec{y})$
      end
      `// Povprečen prispevek in standardna deviacija vrednosti` a `atributa` A_i`.`
      $\varphi_{ia} = mean(\psi)$
      $\sigma_{ia} = stdev(\psi)$
    end
  end
end

Poglavje 4

Metoda za razlago inkrementalnih modelov in posameznih napovedi

Upoštevanje značilnosti podatkovnih tokov (razdelek 2.1) je potreben pogoj za uspešen algoritem za njihovo razlago. Na voljo imamo omejen pomnilnik in procesorski čas. Ključna razlika od razlage statičnih množic je možnost, da v toku pride do spremembe koncepta, zaradi katerega običajne metode odpovedo. Iz tega izhaja ideja, da bi bila dobra razlaga podatkovnega toka v osnovi tudi sama podatkovni tok. Pri implementaciji ideje se bomo oprli na obstoječo metodo razlage za statične množice (razdelek 3.2) in metode strojnega učenja za podatkovne tokove (razdelek 2.4).

4.1 Razlaga modela

Razvijamo posplošeno metodo razlage, ki je neodvisna od tipa klasifikatorja in zmožna zaznave spremembe koncepta v toku. Robusten, hiter in splošen algoritem, ki lahko deluje kot ovojnica za katerikoli klasifikator in poleg mesta

spremembe koncepta meri tudi hitrost, je SPC[1] (algoritem 4). Osnova za njegovo delovanje je spremljanje statistik na toku napak (izgub točnosti), zato lahko vanj lahko vključimo poljuben klasifikator [2]. Njegova slabost je lahko zakasnjena zaznava postopne spremembe koncepta.

4.1.1 Modifikacije SPC

Osnovno inačico SPC bomo za naše potrebe nekoliko predrugačili. Vpeljali bomo drseče okno, učinkovito posodabljanje statistk in v algoritem vključili razlago modela.

Drseče okno

Okno v algoritmu ni gladko drseče, marveč se ob zaznavi spremembe model na novo vzpostavi iz medpomnilnega okna, v katerega shranjujemo primere iz opozorilne stopnje. V primeru, da dolgo ne pride do spremembe koncepta, lahko okno (ali cel inkrementalni model) postane preveliko za pomnilnik oziroma močno poveča klasifikacijske čase. Na podlagi domenskega znanja uvedemo parameter *max_okno*, s katerim omejimo število primerov v trenutnem modelu. Če je meja prekoračena, ga krajšamo po principu vrste (FIFO), kar pomeni, da smo v algoritem vgradili drseče okno.

Ob tem je potrebno poudariti, da ob zaznani spremembi koncepta (izgradnji novega modela) ne izgubljamo tistih starih podatkov, ki so še relevantni. V opozorilni stopnji jih namreč shranjujemo v ločeno medpomnilno okno (prav tako omejeno z *max_okno*), ki ga ob stanju izpod nadzora uporabimo za učenje novega modela. Na ta način zajamemo le tiste zgodovinske

[1] Za zaznavo sprememb nismo vezani na algoritem SPC, lahko uporabimo npr. tudi PageHinkleyev test (algoritem 3). Smo pa omejeni na metode učenja iz podatkovnih tokov, ki uporabljajo okna (delni spomin). Utežitve (faktorje pozabljanja) lahko uporabljamo na toku generiranih razlag.

[2] Inkrementalnost klasifikatorja oziroma visoka granularnost ni potreben pogoj, marveč zelo zaželen. Metoda deluje tudi s statičnimi učnimi algoritmi, vendar jo to znatno upočasni (ni več primerna za praktično uporabo), saj mora ob vsakem novoprispelem primeru ponovno zgraditi, ne le posodobiti odločitveni model na oknu.

podatke, ki so statistično značilni za nov koncept, in zavržemo ostale.

Računanje statistik

Pri večjih parametrih *max_okno* lahko težave povzročata izračuna povprečne napake p_j in standardne deviacije napak s_j, ki v naivni verziji staneta $O(n)$ časa. Problem rešimo z uporabo inkrementalnega posodabljanja obeh statistik [16] (L_j je izguba točnosti pri trenutnem primeru):

$$p_0 = 0, \;\; p_j = \frac{(j-1) \times p_{j-1} + L_j}{j}$$

$$S_1 = 0, \;\; S_j = S_{j-1} + (L_j - p_{j-1})(L_j - p_j)$$

$$s_j = \sqrt{\frac{S_j}{j-1}}; \; k \geq 2$$

Vpeljava razlage modela

Spremenjen SPC uporabimo kot običajno metodo za klasifikacijo primerov v podatkovnem toku, vendar fazi *opozorilo* in *izpod nadzora* delujeta tudi kot indikatorja, ki sprožita razlago modela (algoritem 6) za trenutno okno. Dodaten indikator je še perioda razlage ω, ki pomeni, da na vsakih ω primerov naredimo razlago trenutnega modela, ne glede na spremembe v toku. Razlog za tem je, da sprememba koncepta zelo verjetno pomeni spremembo razlage, ni pa nujno. Periodično razlago uvedemo, da zajamemo manjša nihanja prispevkov in imamo oporne točke, neodvisne od zaznanih sprememb. S tem zajamemo mogoče skrite spremembe v toku razlag, ki jih sicer ne bi opazili, in dobimo enako velika okna, neodvisno od kvalitete klasifikacije. S konstanto M navzdol omejimo velikost okna za razlago, saj bi v nasprotnem primeru dobili nereprezentativno razlago (to se ponavadi zgodi v začetnih fazah učenja koncepta). Ob indikaciji spremembe koncepta shranimo trojico $(\phi, l, tip)_t$, ki predstavlja razlago ϕ modela na oknu velikost l ob času t zaradi indikatorja tip[3].

[3] $tip \in \{''opozorilo'', '' izpod\_nadzora'', '' periodicna''\}$

4.1.2 Algoritem

Rezultat je metoda (algoritem 7), ki lokalno (na oknih) izvaja običajno razlago modela. V dobljenem zaporedju razlag skušamo zajeti koncepte, ki ležijo za tokom, obenem pa dobimo še razlago prehodnih stanj (okno med opozorilom in spremembo koncepta) in periodičen vpogled v stanje modela. Podatkovni tok razlag je torej zrnata predstavitev konceptov, ki ležijo za problemsko domeno vhodnega toka. Uporabimo ga lahko za analizo z običajnimi metodami za strojno učenje iz podatkovnih tokov (da dobimo bolj zgoščeno razlago oziroma pridobivamo dodatna znanja) ali za vizualizacijo. S tem prevedemo problem razlage klasifikacije podatkovnih tokov na znano strukturo (podatkovni tok), ki je primerna za splošno analizo.

Ob uporabi inkrementalnega algoritma vsak primer obdelamo (posodobimo model in statistike, klasificiramo) v konstantnem času. Časovno bolj zahtevna je faza razlage modela (okna), ki je enaka kot pri statični metodi – $O(p \cdot T(\mathcal{X}))$, $p = n \cdot |A_i|, \forall i \in \{1, 2, ..., n\}$. Ta se zgodi ob vsakem opozorilu, stanju izpod nadzora in periodičnem intervalu. Na časovno zahtevnost zato najbolj vpliva pogostost sprememb v toku, število vzorcev m in perioda redne razlage ω.

4.1.3 Ocenjevanje podobnosti razlag

Podatkovni tok razlag $(\phi, l, tip)_t$ lahko analiziramo z ocenjevanjem podobnosti. Zanimajo nas spremembe v toku razlag oziroma razlike med zaporednimi razlagami. Merimo lahko tudi podobnost s preteklimi razlagami in na ta način odkrivamo ponavljajoče se koncepte v toku. Iz posamezne trojice vzamemo razlago modela ϕ_t in jo predstavimo kot vektor prispevkov vrednosti atributov $\phi_t = [\varphi_1, \varphi_2, ..., \varphi_p] \in [-1, 1]^p, p = n \cdot |A_i|, \forall i \in \{1, 2, ..., n\}$. Ker imajo podatkovni tokovi časovno komponento, bomo primerjali zaporedne člene φ_{t-1}, φ_t. Razdaljo med vektorjema lahko izmerimo z eno od številnih mer razdalj oziroma iz njih izpeljanih mer podobnosti (razdalje Minkowskega, razdalja Mahalanobisa, Hammingova razdalja, kosinusna podobnost, podob-

Algoritem 7: Osnovna metoda za razlago napovednih modelov za podatkovne tokove

Data: Podatkovni tok $(\vec{x}_i, y_i)_t$

Klasifikator h

Razred za razlago r

Število primerov v vzorcu m

Perioda razlage ω

Meja za razlago M

Največja velikost okna max_okno

begin

Izvajaj SPC

$t \leftarrow timestamp$

if $|okno| > max\_okno$ **then**

Odstrani najstarejši primer iz *okno*

end

`// V spominu ohranjamo model, naučen na trenutnem oknu, okno in trenutni primer`

if *(Opozorilo* $\vee$ *Izpod nadzora* $\vee$ *Periodična razlaga)* $\wedge$ $|okno| > M$ **then**

$(\phi, l, tip)_t \leftarrow (razlaga(model(okno), okno, m, r), |okno|, tip)$

`// Shranjujemo trojice oblike (razlaga modela, velikost okna, razlog za razlago)`

end

end

nost po Jaccardu, ...). Uporabili bomo kosinusno mero podobnosti, ki jo z uporabo skalarnih produktov in norm izračunamo kot kosinus kóta med vektorjema[4].

$$cos(\varphi_{t-1}, \varphi_t) = \frac{\varphi_{t-1} \cdot \varphi_t}{|\varphi_{t-1}||\varphi_t|} = \frac{\varphi_{t-1} \cdot \varphi_t}{\sqrt{\varphi_{t-1} \cdot \varphi_{t-1}}\sqrt{\varphi_t \cdot \varphi_t}}$$

Mera je primerna, ker nas bolj zanima smer vektorjev, ki ustreza profilu razlage, kot pa njihova velikost. Vrednosti $cos(\varphi_{t-1}, \varphi_t)$ ležijo na intervalu $[-1, 1]$, pri čemer -1 pomeni popolno različnost (vektorja kažeta v nasprotni smeri), 1 pa enakost vektorjev.

Poleg primerjanja zaporednih razlag lahko merimo tudi podobnost s preteklimi razlagami, kolikor jih še hranimo v spominu, in na ta način odkrivamo ponavljajoče se koncepte v toku. Za φ_t takrat iščemo razlago, ki bo maksimizirala podobnost med njima:

$$\varphi_{najblizja} = \varphi \in (\varphi)_t : cos(\varphi, \varphi_t) = \max_{i \in \{1,2,\ldots,t-1\}} cos(\varphi_i, \varphi_t)$$

4.2 Razlaga klasifikacije posameznih primerov

Posamezen primer razložimo kot pri statičnih množicah, s tem da za učno množico in naučen model uporabimo trenutno okno in model na njem. To je pravilen način, saj učni algoritem v naši metodi klasificira le na podlagi trenutnega okna. Velikost in položaj slednjega se samodejno prilagajata glede na opozorila s ciljem, da bi dobro zajeli različne koncepte v toku. Razlaga torej odraža model. Če je slednji kvaliteten, njegova razlaga pravilno odseva koncepte za trenutnim oknom (razdelek 3.2), kar nam da dobro razlago klasifikacije primera. Četudi v resnici starejši primeri (izven okna) vplivajo na razrede trenutnih, jih ne smemo upoštevati, saj je model naučen le na tre-

[4]Pri tem izvzamemo ničelne vektorje ($\vec{0}$), ki nimajo določene smeri in zato z njimi ne moremo računati kotov. V programski implementaciji vračamo `NaN`.

nutnem oknu. Razlaga primera pri inkrementalnih učnih modelih je zato le klic metode (algoritem 5): $razlagaPrimera(model(okno), okno, primer, m)$.

4.3 Vizualizacije

Razlaga je sestavljena iz prispevkov na intervalu $[-1, 1]$. V primeru obsežnih prostorov instanc, pogostih sprememb ali kratkih period njihovo število lahko precej naraste. Pri takšni količini medsebojno podobnih podatkov ljudje težko najdemo pomenljive vzorce iz gole številske predstavitve, zato se poslužimo vizualizacij[5]. Te so tudi v skladu s ciljem razlage, proces odločanja narediti transparenten in jasno utemeljiti rezultate. Tehnike tu le na kratko predstavimo – veliko primerov uporabe je med rezultati testiranj v razdelku 5.3.

Razlaga modela in primerov

Prispevki vrednosti atributov so primerni za vizualizacijo z horizontalnimi stolpičnimi diagrami. Ponazorimo jih z modrimi stolpci, za vsak atribut pa dodamo še zelen stolpec, ki predstavlja povprečen pozitiven in povprečen negativen prispevek posameznega atributa kot celote. Zaporedje razlag dopolnimo z oznakami za velikost oken, čas in vzrok za razlago.

Posamezen primer razlagamo na podoben način, le da v sliko vključimo samo prispevke vrednosti atributov, s katerimi je opisan. Povprečen prispevek atributa je zato enak prispevku vrednosti, ki jo zavzame.

Trodimenzionalna vizualizacija razlag podatkovnega toka

Izhajamo iz osnovne vizualizacije, ki je problematična, če tok opazujemo dolgo časa ali če pogosto prihaja do sprememb. Sosledje stolpičnih diagra-

[5]Zaradi potencialno velikega števila podrobnosti in majhnih nians med barvami so vizualizacije primarno namenjene za prikaz na zaslonih z visoko resolucijo in ne za tisk na papir.

mov takrat ni primerno. Pri vizualizaciji se nasploh izogibamo zaporedjem podobnih grafik, saj ljudje težko primerjamo več kot dva elementa naenkrat. Čeprav smo pri algoritmu, ki generira razlage vezani na drseča okna, lahko na toku razlag, ki ga dobimo, uporabljamo utežitve in faktorje pozabljanja, kakor tudi druge tehnike strojnega učenja. To dejstvo izkoristimo za izboljšavo vizualizacije sosledja razlag kot poskus razlage konceptov na celotnem toku v eni sliki z uporabo barve kot tretje dimenzije.

V spomin shranimo tok razlag $(\phi, l, tip)_t, |(\phi, l, tip)_t| = k$. Upoštevamo ali tiste, ki smo jih dobili ob indikaciji spremembe ali pa tiste, ki smo jih dobili s periodičnim zajemanjem enako velikih oken. S tem preprečimo prekrivanje oken. Utežitve β glede na velikost okna l in faktorje pozabljanja[6] γ skrijemo v faktorje prosojnosti α (za vsako okno po en faktor):

$$\alpha((\phi, l, tip)_i) = \gamma^{(k-i)} + \frac{l}{|vsi\ primeri|}$$

in s tem zmanjšamo pomembnost starih in majhnih oken. Tabelo faktorjev prosojnosti normaliziramo in dodatno z zeleno obrobo označimo zadnjo razlago. Okna, ki so stara in/ali majhna zato na vizualizaciji zbledijo, njihove prekrivajoče sledi pa nam dajo občutek o gibanju trendov v konceptih.

Razdalje med razlagami

Razdalje med zaporednimi razlagami ponazorimo s pokončnimi stolpičnimi diagrami. Velikost stolpca je sorazmerna s podobnostjo oziroma različnostjo. Dodatno informacijo dobimo s postavitvijo stolpcev na mesta primerov, kjer so se razlage generirale in s tem zajamemo še časovno komponento.

[6] Ponavadi $\gamma = 0.9$

Poglavje 5

Eksperimentalna evalvacija

V tem poglavju preučimo in komentiramo rezultate rabe predlaganih metod na nekaj testnih množicah. Vsi algoritmi, potrebni za izvedbo testiranja, so bili implementirani v jeziku `Python` s knjižnicami `scipy`, `numpy`, `Orange`, `matplotlib` in `pandas`.

5.1 Testiranje

Osnovna metoda, ki jo uporabljamo, je predlagan algoritem za razlago napovednih modelov pri podatkovnih tokovih, ki smo ga predstavili v poglavju 4. Ovili ga bomo okrog dveh naravno inkrementalnih klasifikacijskih algoritmov:

- naivni Bayesov klasifikator,
- k-najbližjih sosedov.

Testiranje poteka na treh testnih množicah, na katerih študiramo uporabnost predlaganih metod. Potek testiranja lahko ločimo na komponente:

1. **Analiza učenja algoritma** – preučujemo zaznavo spremembe koncepta, njen vpliv na razlago in dobljeni tok, ki ga vizualiziramo in naprej analiziramo. Prikažemo velikosti oken in grafe, ki predstavljajo napake in spremembo koncepta pri učenju.

2. **Obravnava toka razlag in klasifikacij posameznih primerov** – razlage vizualiziramo in komentiramo. Prikažemo tudi nekaj razlag posameznih primerov, jih umestimo v kontekst in jih analiziramo. Tok razlag in našo razlago toka, primerjamo s tisto, ki bi jo dobili, če bi vse primere v testnih množicah imeli na voljo naenkrat. Na ta način študiramo razlike med razlago podatkovnega toka in razlago statične množice oziroma ugotavljamo, če z uporabo naše metode pridobimo boljši vpogled v koncepte za podatkovnim tokom. Z zadnjo množico obravnavamo tudi obraten problem – z metodo za razlago inkrementalnih modelov in primerov obravnavamo prvotno statično množico.

3. **Podobnost med razlagami** – uporabljamo kosinusno mero podobnosti za odkrivanje vzorcev v toku razlag in primerjanje rezultatov različnih klasifikatorjev.

4. **Vizualizacije** – kjer je možno, rezultate predstavimo grafično in preučimo njihovo uporabnost.

Če ni navedeno drugače, pojasnjujemo pozitiven razred, število primerov[1] v vzorcu m pa nastavimo na 65000. Periodo razlage ω, minimalno velikost za razlago M in maksimalno velikost drsečega okna max_okno nastavimo na podlagi domenskega znanja za posamezno testno množico.

5.2 Testne množice

Predstavimo testne podatkovne tokove – dva sta sintetična, kar nam omogoča bolj natančno ocenjevanje kvalitete rezultatov[2] in lažjo izgradnjo nadaljnjih testnih množic.

[1] Za utemeljitev izbora tega števila glej razdelek 3.2.2.

[2] Testne podatkovne tokove iz realnega sveta, primerne za klasifikacijo in z zanesljivim mestom spremembe koncepta, je zelo težko dobiti [7].

STAGGER

Klasična množica za testiranje spremembe koncepta, katere primeri so opisani s tremi diskretnimi atributi atributi, ki predstavljajo karakteristike likov v toku *size* (*small*, *medium*, *large*), *color* (*red*, *green*) in *shapre* (*circle*, *square*, *triangle*). Razred je binaren, pozitivno vrednost pa določajo trije koncepti:

1. $size = small \wedge color = red$
2. $color = green \vee shape = square$
3. $size = medium \vee size = large$

Prehod med koncepti je postopen (na vmesnem intervalu se primeri dveh konceptov mešajo), množica vsebuje disjunktne koncepte in nereleavnatne atribute.

Uporabili bomo podatkovni tok, generiran z orodjem MOA [1]. V tabeli 5.1 ga predstavimo z intervali konceptov in dolžino prehodov med njimi.

Tabela 5.1: Primer podatkovnega toka STAGGER

Interval	**Koncept**	**Širina intervala prehoda**
$[0, 749]$	1	50
$[750, 1799]$	2	50
$[1800, 3599]$	3	150
$[3600, 4500]$	1	/

SEA concepts

Podatki zajemajo štiri koncepte, ki so popolnoma določeni z dvema od treh zveznih atributov x_1, x_2, x_3, $x_i \in [0, 10]$ Koncepti so oblike $x_1 + x_2 \leq \beta$, $\beta \in \{7, 8, 9, 9.5\}$. V toku je 60000 primerov, prehod med koncepti je nenaden na vsakih 15000, razredna spremenljivka pa vsebuje 10% šuma.

Titanic

Znana statična množica obsega 2201 primer. Primeri predstavljajo potnike na ladji, opisane z diskretnimi atributi: *status* (*first*, *second*, *third*, *crew*), *age* (*adult*, *child*) in *sex* (*male*, *female*). Binarna razredna spremenljivka *survived* določa preživetje potnika. Primere v množici permutiramo in jih uporabimo za preizkus uspešnosti delovanja predlagane metode razlage inkrementalnih modelov na statični množici.

5.3 Rezultati

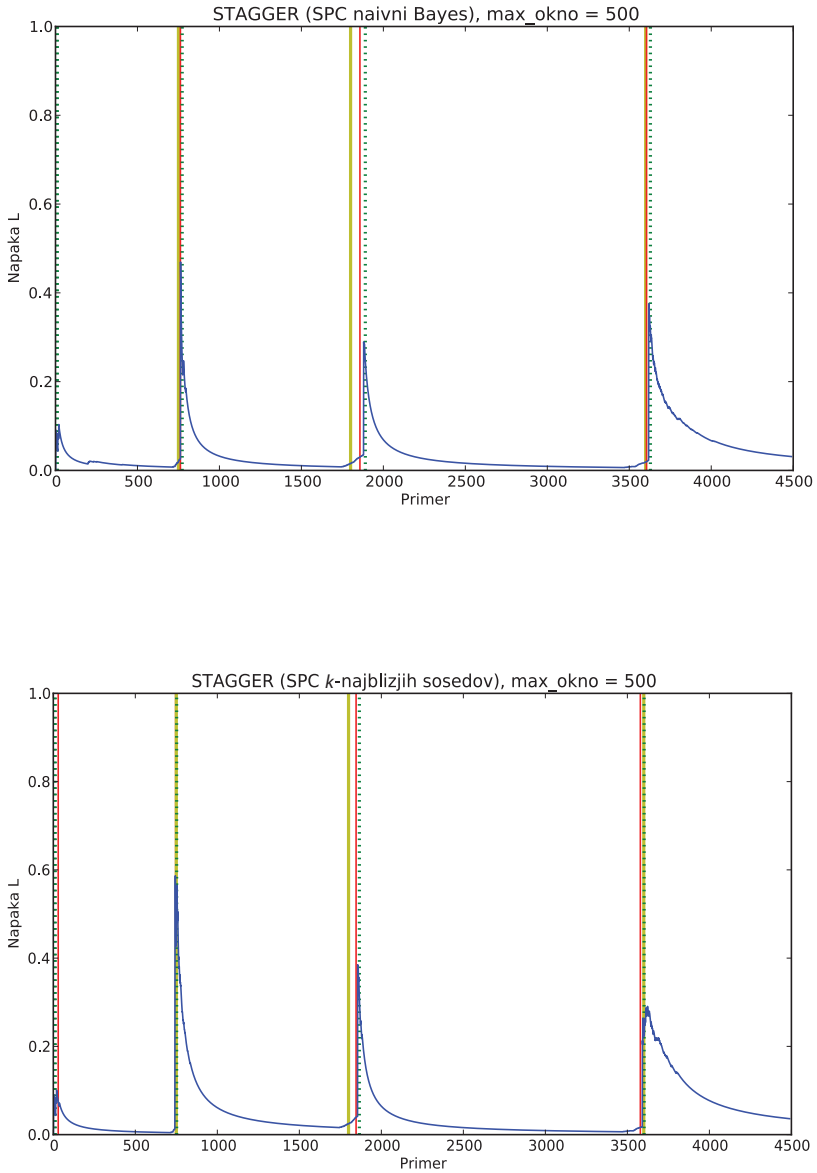
5.3.1 Podatkovni tok Stagger

Na podatkovnem toku STAGGER (slika 5.1) opazimo pravilno zaznavo spremembe koncepta, ki sovpada z intervali, ki smo jih generirali. Iz teh indikatorjev dobimo podatkovni tok razlag, pri čemer na vsakih 500 primerov opravimo še periodično razlago, velikost okna pa omejimo z $max\_okno = 500$.

Razlaga toka

Vizualizacija (sliki 5.2 in 5.3) predstavlja razlage oken, ki jih dobimo ob zaznavi spremembe koncepta. Oba klasifikatorja pravilno razporedita prispevke, pri čemer imata v razlagi malo medsebojnih razlik (k-NN v splošnem pripisuje po absolutni vrednosti večje prispevke). Zlahka prepoznamo menjajoče se koncepte, ki nastopajo v obliki parov razlag. Prvi element v paru je razlaga, ki jo sproži opozorilo, drugi pa okno, s katerim se algoritem prilagodi na spremembo (izpod nadzora) [3]. Slednja razlaga je zaradi kratkega okna lahko nenatančna in odraža vmesno stanje pri prehodu med koncepti. Ko vizualizacijo beremo po parih, vidimo, da sovpada s pravili, ki ležijo za podatki (tabela 5.1), vključno z disjunktnimi koncepti (2 in 3). Pri konceptu

[3] k-NN pri prvem konceptu nima razložene stopnje izpod nadzora, ker se je algoritem prehitro prilagodil – velikost okna ni bila dovoljšnja za razlago.



Slika 5.1: Graf povprečnih napak pri klasifikaciji podatkovnega toka STAGGER z uporabo algoritma SPC kot ovojnice za metodi naivnega Bayesa in k-najbližjih sosedov. Debela rumena črta predstavlja resnično spremembo koncepta, tanka rdeča opozorilno stopnjo, pikčasta zelena pa stanje izpod nadzora.

3 je zanimiv velik negativen prispevek vrednosti *small*, ki pomeni pravilno razlago disjunkcije $size = medium \vee size = large$.

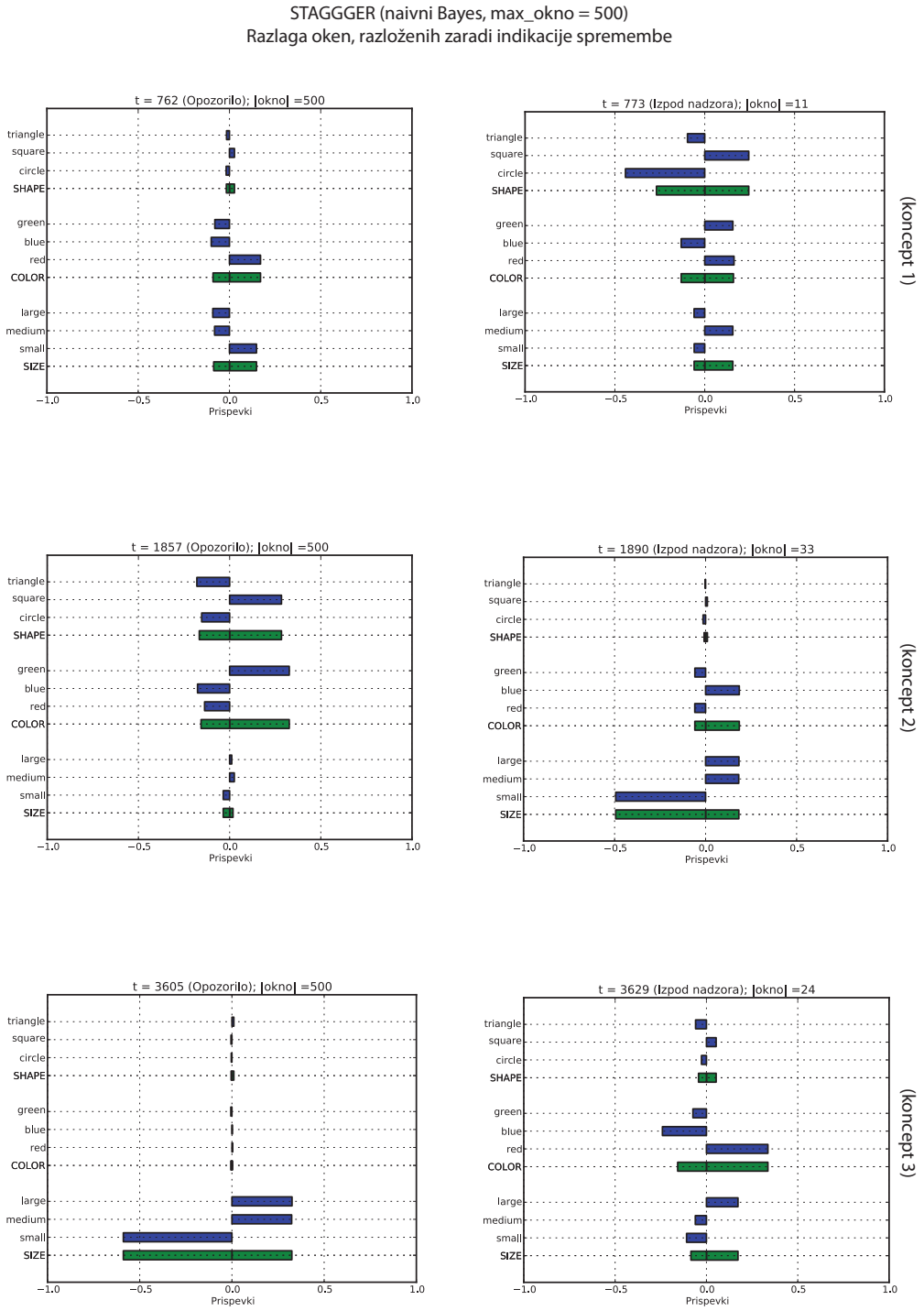
Bistven pomen razlage podatkovnega toka s posebno metodo pa vidimo ob primerjavi z razlago, ki jo dobimo, če množico naložimo v spomin in jo obravnavamo kot statično (sliki 5.5 in 5.6). Po prvi spremembi koncepta ($t = 750$) napaka sunkovito naraste, ker se algoritem ustrezno ne prilagodi. Posledično je tudi razlaga (slika 5.4) nesmiselna – rezultat je nekakšno povprečje vseh treh konceptov, ki ni informativno. Iz nje ne moremo razbrati, koliko konceptov smo sploh prešli, kako se spreminjajo in kakšne so njihove karakteristike. Uporaba metode razlage, prilagojene za podatkovne tokove, je torej bila v tem primeru nujna za dober rezultat.

Podobnost med zaporednimi razlagami predstavimo na slikah 5.7 in 5.8. Povprečna podobnost med razlagami pri klasifikatorju naivni Bayes in razlagami pri k-najbližjih sosedov je 0.6605 (indikacije spremembe koncepta) oziroma 0.9970 (redni intervali). Razliko med njima lahko pripišemo vmesnim stanjem oken in razlikam pri času zaznave spremembe. Ob rednih intervalih se to povprečje zgladi, ker je razlag več in se izvajajo na dobro naučenih modelih.

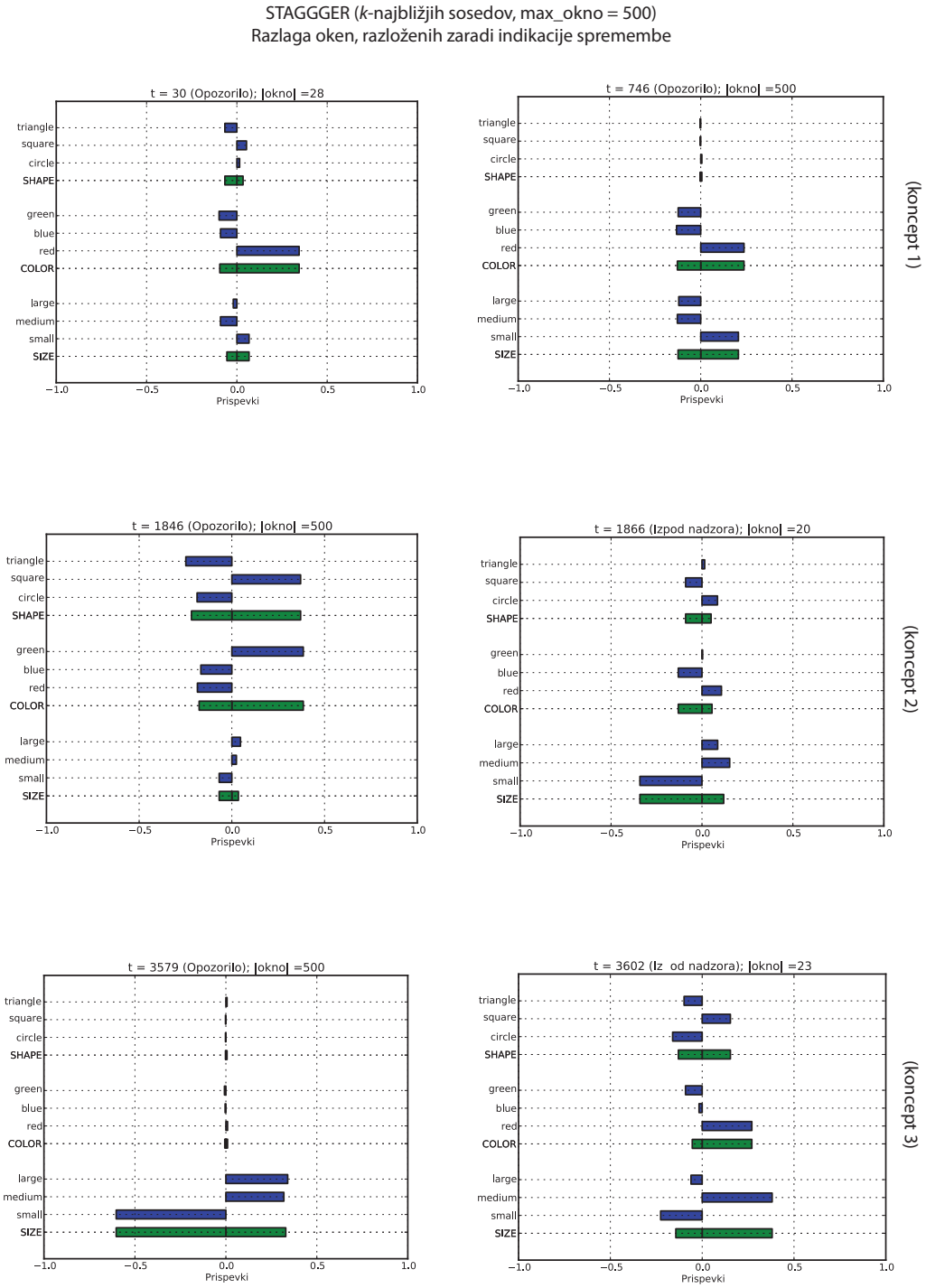
Vizualizacija toka razlag

Navkljub težavam, na katere smo naleteli pri obravnavi množice kot statične, si želimo predstavitve toka v eni sliki. Uporabimo predlagano metodo vizualizacije[4] (razdelek 4.3), s katero generiramo sliki 5.9 in 5.10. Iz njih dobimo dober občutek o gibanju trendov v toku. Predvsem so tu uporabne razlage, ki smo jih dobili ob rednih intervalih (desno), saj z njimi dobro zajamemo pomembnost trajnejših konceptov (prekrivajoči se stolpci) in dobimo bolj fino reprezentacijo kot pri indikatorjih sprememb (levo).

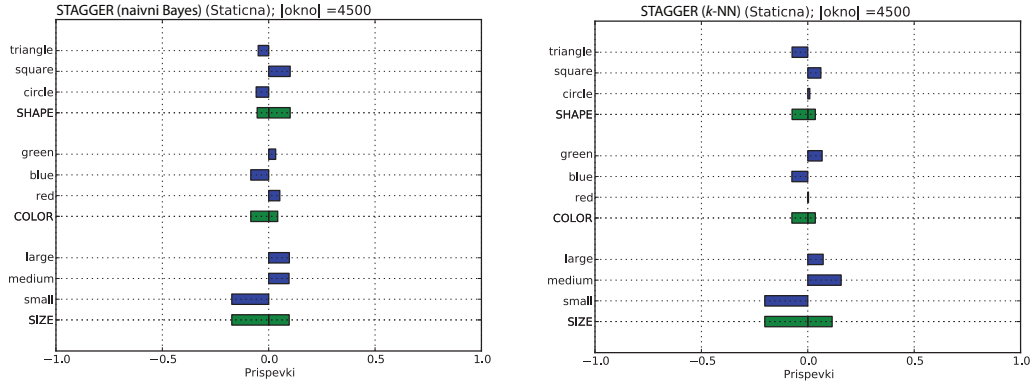
[4]Uporabili smo faktor pozabljanja $\gamma = 0.9$.



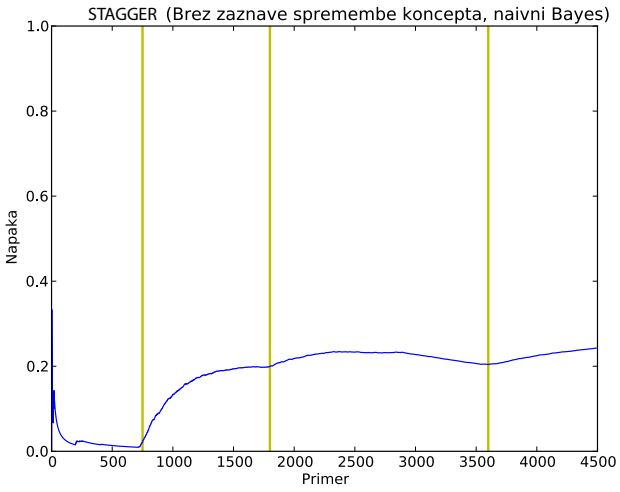
Slika 5.2: Razlage oken algoritma SPC (naivni Bayes) za podatkovni tok STAGGER glede na indikatorje sprememb koncepta (slika 5.1).



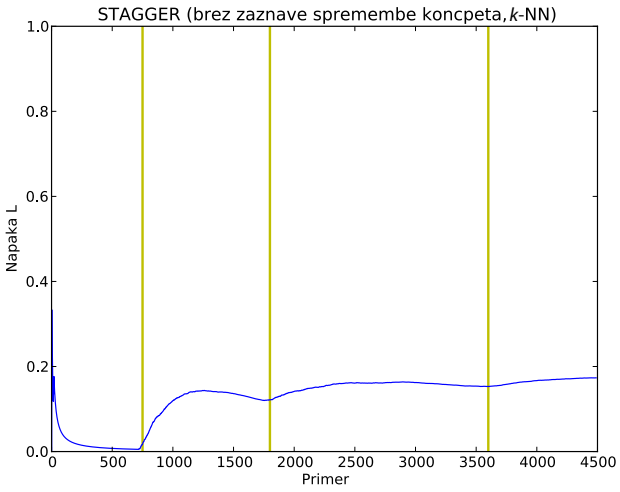
Slika 5.3: Razlage oken algoritma SPC (k-najbližjih sosedov) za podatkovni tok STAGGER glede na indikatorje sprememb koncepta (slika 5.1).



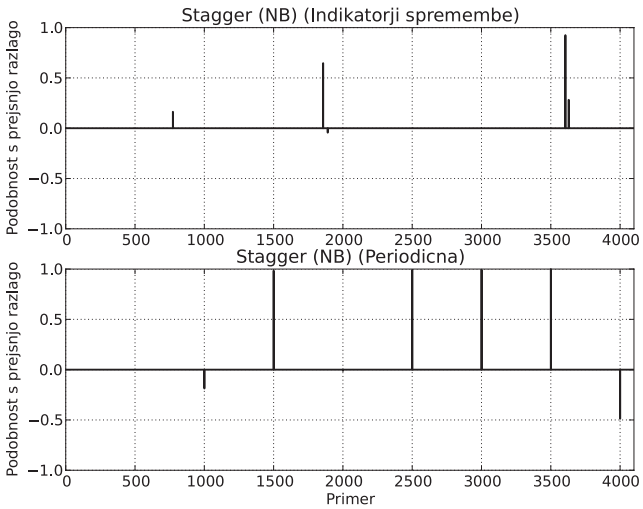
Slika 5.4: Razlaga podatkovnega toka STAGGER, če ga obravnavamo kot statično množico s klasifikatorjema naivni Bayes in k-najbližjih sosedov brez zaznave spremembe koncepta.



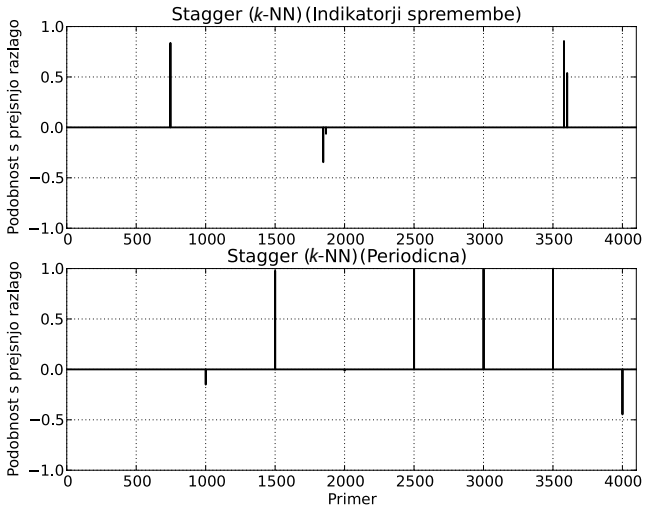
Slika 5.5: Napaka pri klasifikaciji v podatkovnem toku STAGGER brez zaznave spremembe koncepta (naivni Bayes).



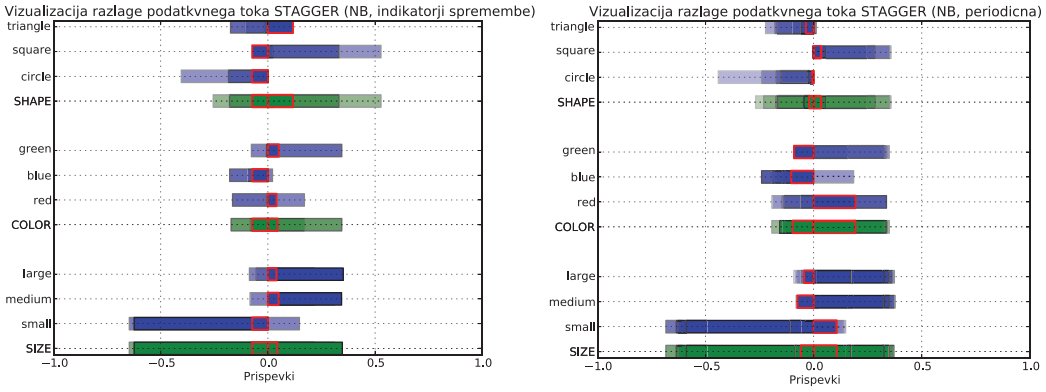
Slika 5.6: Napaka pri klasifikaciji v podatkovnemu toku STAGGER brez zaznave spremembe koncepta (k-najbližjih sosedov).



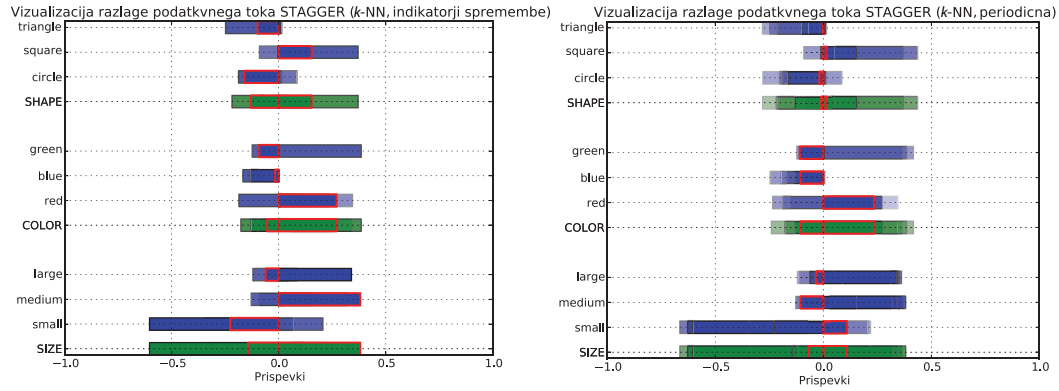
Slika 5.7: Medsebojna (kosinusna) podobnost zaporednih razlag podatkovnega toka STAGGER (klasifikator: naivni Bayes).



Slika 5.8: Medsebojna (kosinusna) podobnost zaporednih razlag podatkovnega toka STAGGER (klasifikator: k-najbližjih sosedov).



Slika 5.9: Vizualizacija razlage celotnega toka STAGGER z uporabo predlagane metode (naivni Bayes).



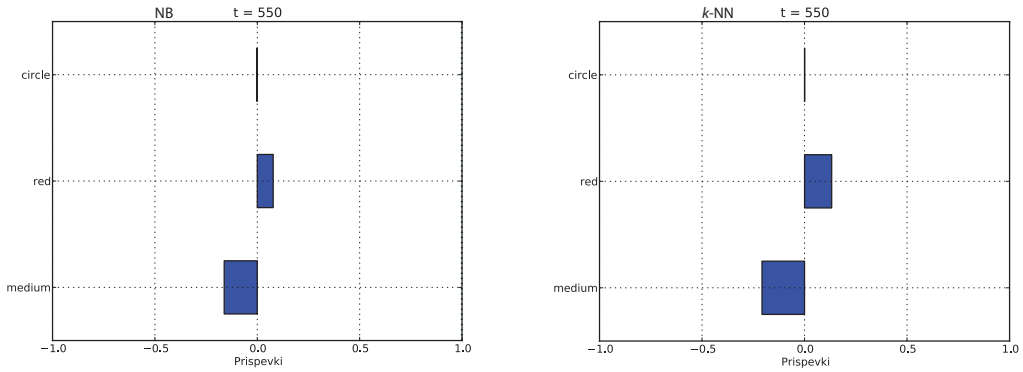
Slika 5.10: Vizualizacija razlage celotnega toka STAGGER z uporabo predlagane metode (k-najbližjih sosedov).

Razlage klasifikacij posameznih primerov

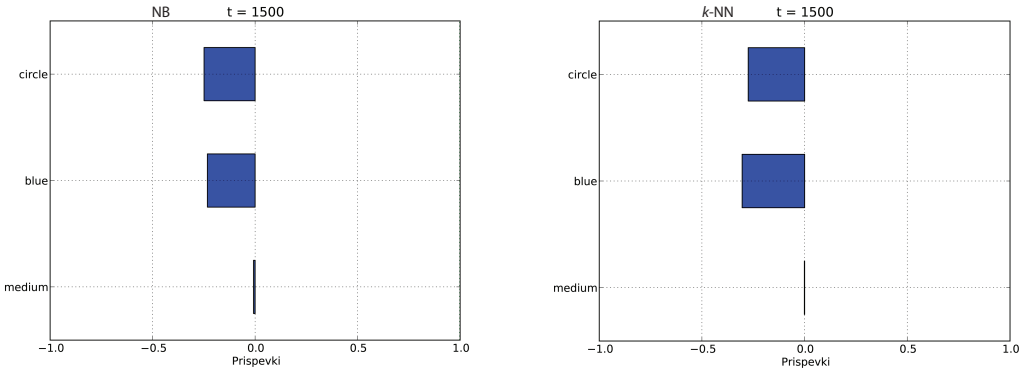
Razlage posameznih primerov naredimo kot pri statičnih množicah, le da za učno množico vzamemo trenutno okno. Opazimo, kako enak primer (*shape = circle, color = blue, size = medium*) v različnih konceptih razlagamo povsem drugače (sliki 5.12 in 5.13), kar je bistvena razlika od razlage pri statičnih množicah. Pri primeru iz koncepta 1 (slika 5.11) je razvidno, da prispevki pri razlagi klasifikacije posameznega primera niso nujno sorazmerno veliki s prispevki pri razlagi celotnega modela (glej vrednost atributov *medium* in *red*), vpliv imajo interakcije (več v razdelku 3.2).

Ugotovitve

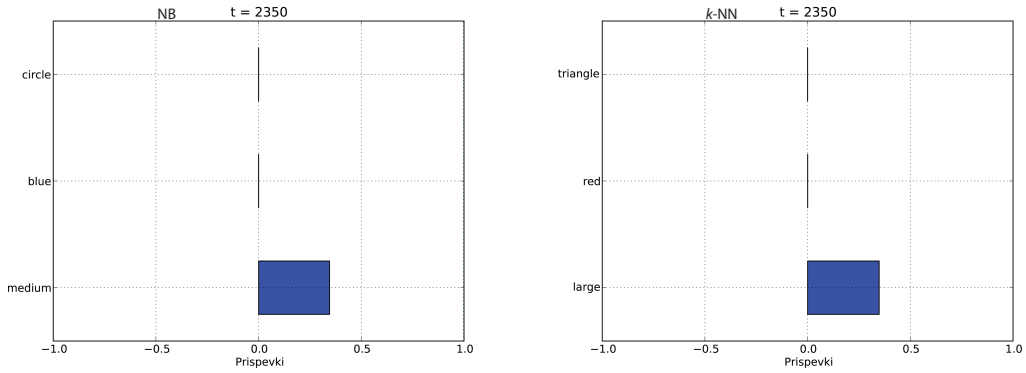
Z upoštevanjem spremembe koncepta in ustrezno prilagoditvijo razlage smo torej pridobili mnogo boljši vpogled v podatkovni tok, kot če bi uporabili statično metodo, ki v tem primeru vrne napačne rezultate. Poudariti pa je potrebno, da so bili tu podatki umetni in brez šuma, predvsem pa je pomembno dejstvo, da so na njih klasifikatorji dobro delovali. Spomnimo se, da je kvaliteta razlage korelirana s kvaliteto klasifikatorja (razdelek 3.2.2). V primeru podatkovnih tokov je dodaten velik vir napak lahko tudi neustrezno



Slika 5.11: Razlaga klasifikacije primera (*shape = circle, color = red, size = medium*) iz podatkovnega toka STAGGER (koncept 1).



Slika 5.12: Razlaga klasifikacije primera (*shape = circle, color = blue, size = medium*) iz podatkovnega toka STAGGER (koncept 2).



Slika 5.13: Razlaga klasifikacije primera (*shape = circle, color = blue, size = medium*) iz podatkovnega toka STAGGER (koncept 3).

reagiranje na spremembe. SPC na podatkovnem toku STAGGER deluje zelo dobro, upoštevati pa je potrebno, da bi ob zgrešeni spremembi padla kvaliteta razlage; skrajen primer tega vidimo v obravnavi množice kot statične (sliki 5.5 in 5.6).

5.3.2 Podatkovni tok SEA concepts

Za klasifikacijo imamo v toku SEA concepts na razpolago veliko število primerov z zveznimi atributi, zato so tudi okna večja ($max\_okno = 4000$, $\omega = 5000$) in posledično zaznava spremembe koncepta nekoliko bolj zakasnjena (slika 5.14), vendar so vse spremembe zaznane. Za potrebe razlage zalogo vrednosti $\mathfrak{R}_i$ posameznega atributa A_i diskretiziramo na 10 ekvidistančnih intervalov.

Razlaga toka

Na slikah 5.15, 5.16, 5.17 in 5.18 je prikazana razlaga oken, ki jih dobimo ob indikaciji spremembe koncepta. Če primerjamo prispevke pri uporabi obeh klasifikatorjev, vidimo, da se ujemajo po predznakih, po absolutni vrednosti pa prihaja do razlik. k-najbližjih sosedov daje manj točne razlage – spomnimo se na koncepte za tokom SEA concepts: $x_1 + x_2 \leq \beta$, $\beta \in \{7, 8, 9, 9.5\}$.

Pri idealnem klasifikatorju bi zato prispevki vrednosti posameznega atributa strogo padali z vrednostmi. Z izjemo vmesnih stanj, to dosežemo z naivnim Bayesom. V obeh primerih pa absolutne vrednosti vseh prispevkov padajo s časom, kar je smiselno, ker se meja za pozitivno klasifikacijo β viša – visokim vrednostim se tedaj manjša negativen vpliv, majhnim pa pozitiven. Atribut x_3, ki ne vpliva na razlago, je pravilno razložen – prispevki so majhni in naključno predznačeni, kar lahko pripišemo šumu v podatkih in rabi drsečega okna.

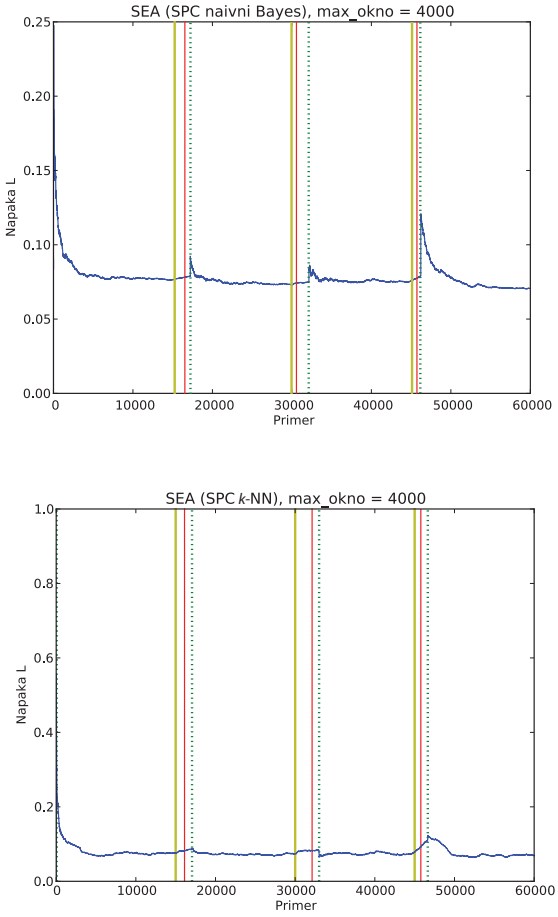
Na dejstvo, da so razlike med koncepti majhne, kažejo mere podobnosti med razlagami (sliki 5.19 in 5.20). Za razliko od podatkovnega toka STAGGER, tu nimamo negativnih vrednosti, kar pomeni, da nimamo različnih[5] razlag. Povprečna podobnost razlag obeh klasifikatorjev je 0.9308 pri indikatorjih sprememb in 0.9826 pri periodičnih razlagah. Visoka prva vrednost nakazuje, da oba klasifikatorja na podoben način zaznavata in reagirata spremembe. V primerjavi z podatkovnim tokom STAGGER je druga vrednost manjša, kar smo pripisali slabši kvaliteti razlag pri k-najbližjih sosedov.

Če podatkovni tok obravnavamo kot statično množico (slika 5.21), so izgube na kvaliteti razlage mnogo manjše kot pri podatkovnem toku STAGGER. Okvirna razlaga je pravilna – vrednosti x_1, x_2, manjše od 5.0 prispevajo v prid pozitivni klasifikaciji, večje obratno. Atribut x_3 pa je razložen celo bolje kot z metodo z zaznavo spremembe koncepta, saj razlagamo na množici, ki je mnogo večja (60000) od posameznega okna (4000), zato se prispevki bolj približajo 0.

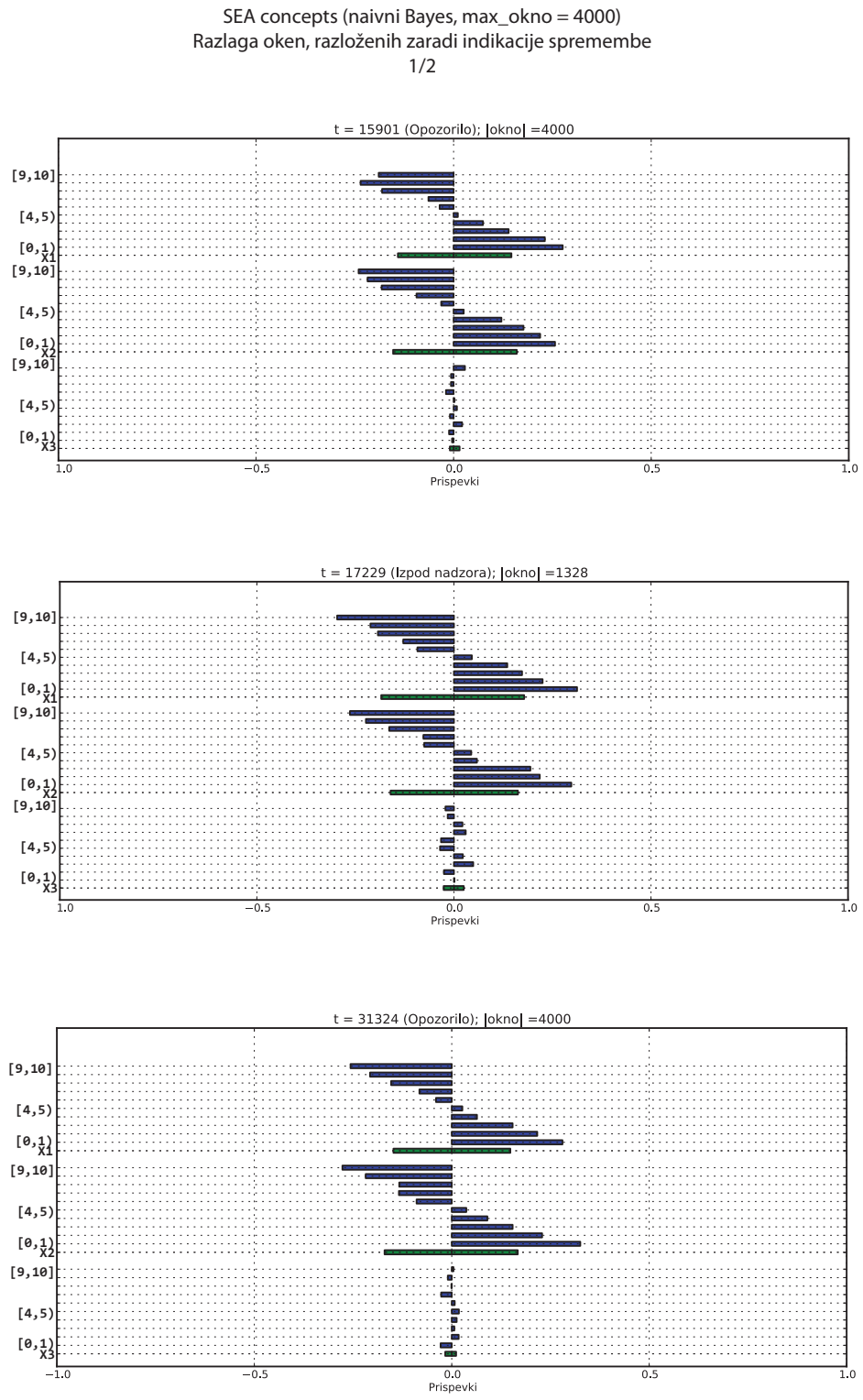
Vizualizacija toka razlag

Trodimenzionalna vizualizacija (sliki 5.22 in 5.23) je za podatkovni tok SEA concepts zelo primerna. Zopet se bolje obnesejo razlage oken, zajetih ob rednih intervalih, kjer se vidijo sledi starih prispevkov, ki s časom po absolutni vrednosti padajo in s tem pravilno odražajo spremembo koncepta. Poleg tega je izboljšana razlaga atributa x_3, kjer so nihanja prispevkov vrednosti zgla-

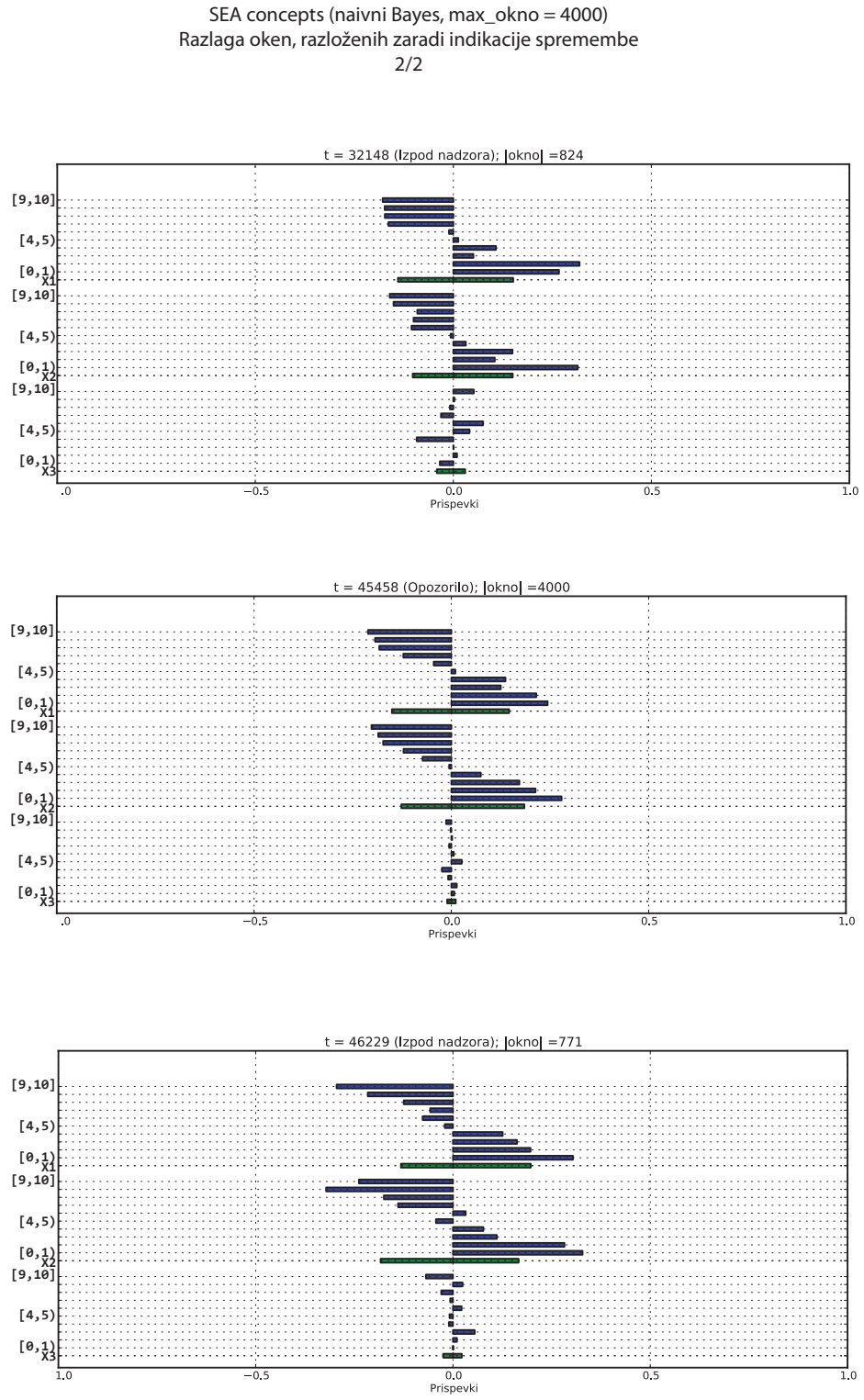
[5]Kóti med vektoriziranimi zaporednimi razlagami ne presegajo $\frac{\pi}{2}$.



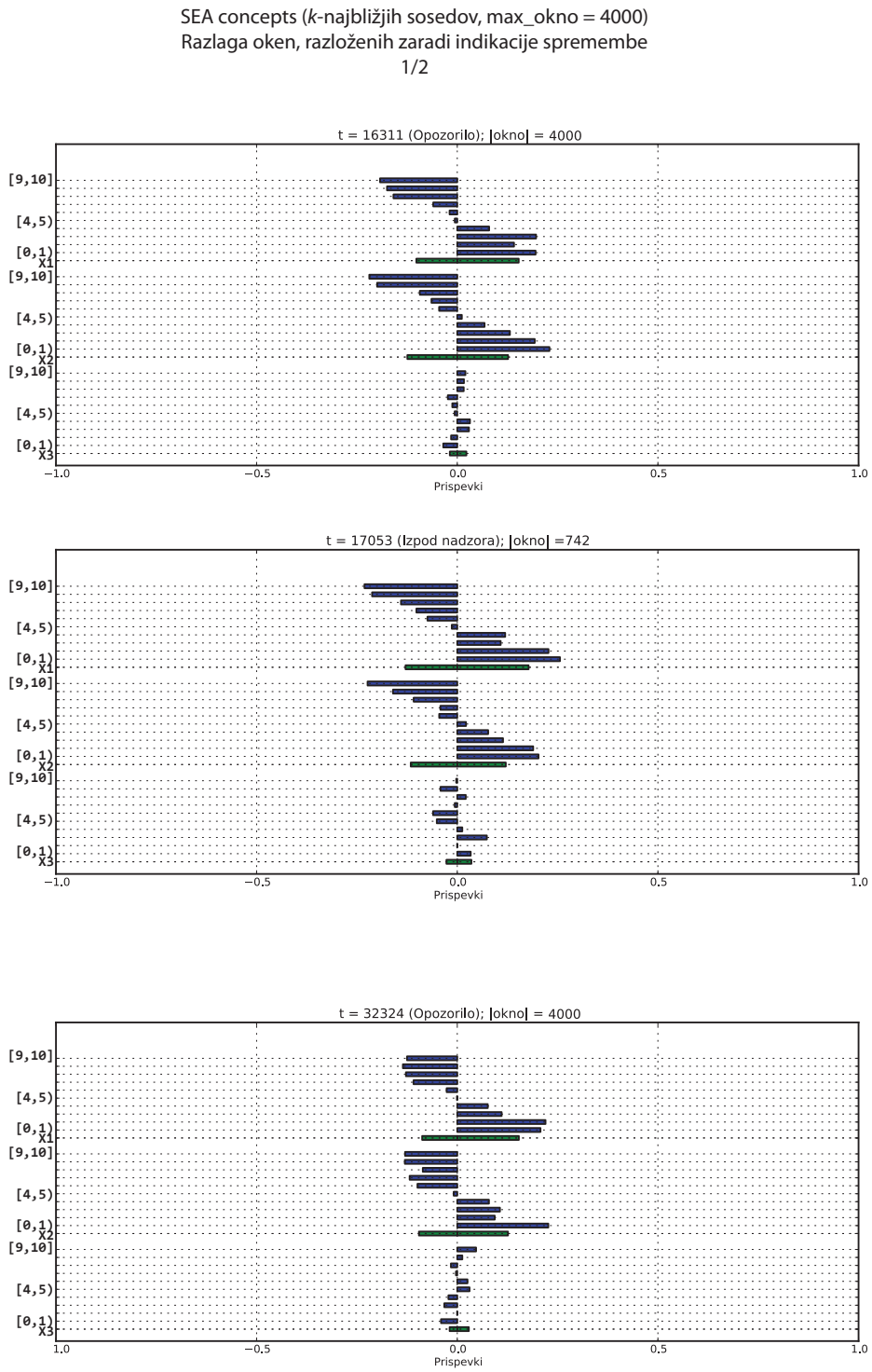
Slika 5.14: Graf povprečnih napak pri klasifikaciji podatkovnega toka SEA concepts z uporabo algoritma SPC kot ovojnice za metodi naivnega Bayesa in k-najbližjih sosedov. Debela rumena črta predstavlja resnično spremembo koncepta, tanka rdeča opozorilno stopnjo, pikčasta zelena pa stanje izpod nadzora.



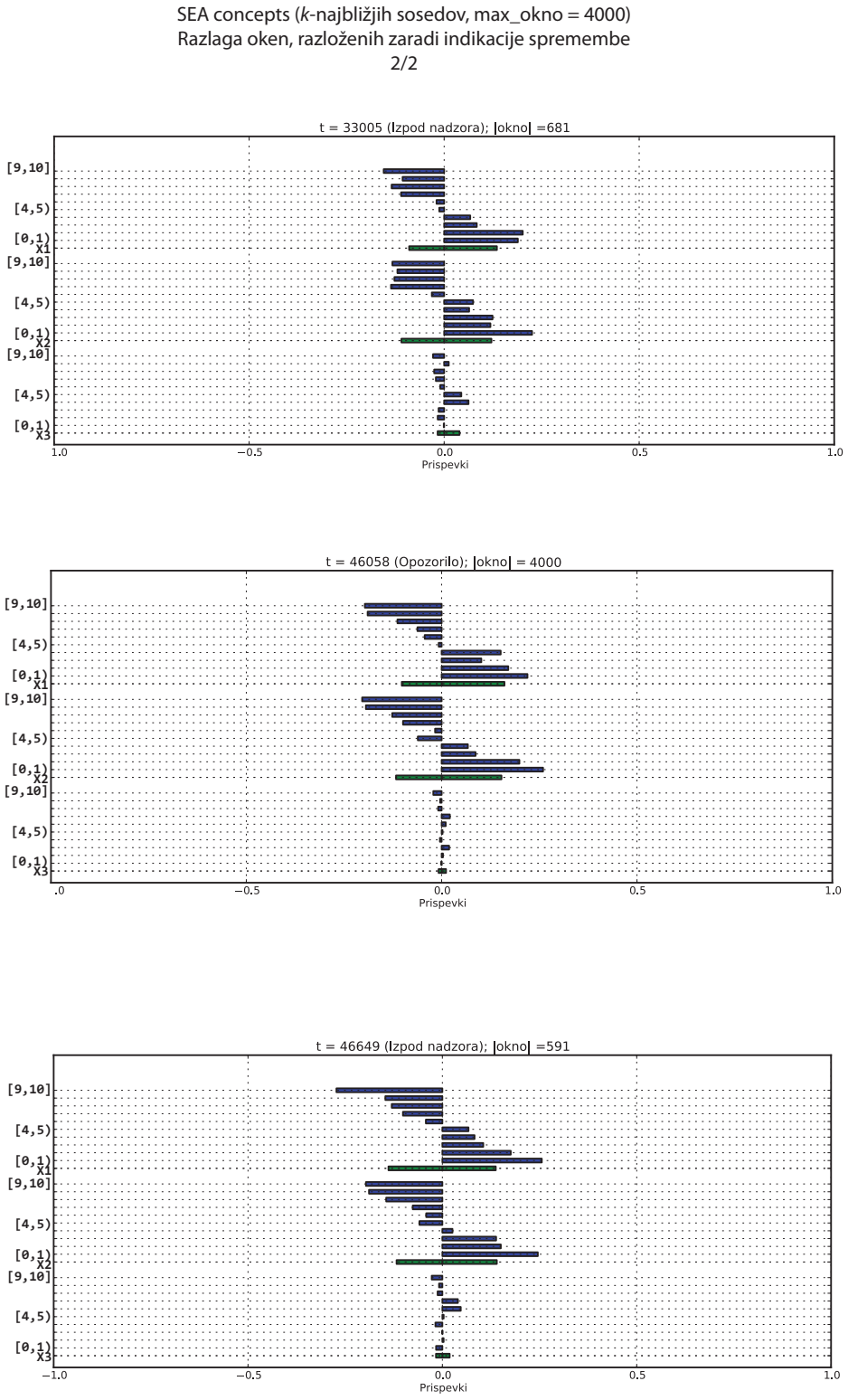
Slika 5.15: Razlage oken algoritma SPC (naivni Bayes) za podatkovni tok SEA concepts glede na indikatorje sprememb koncepta (1/2).



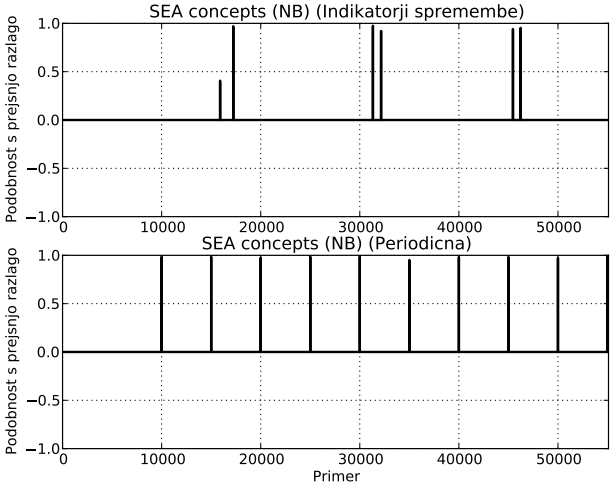
Slika 5.16: Razlage oken algoritma SPC (naivni Bayes) za podatkovni tok SEA concepts glede na indikatorje sprememb koncepta (2/2).



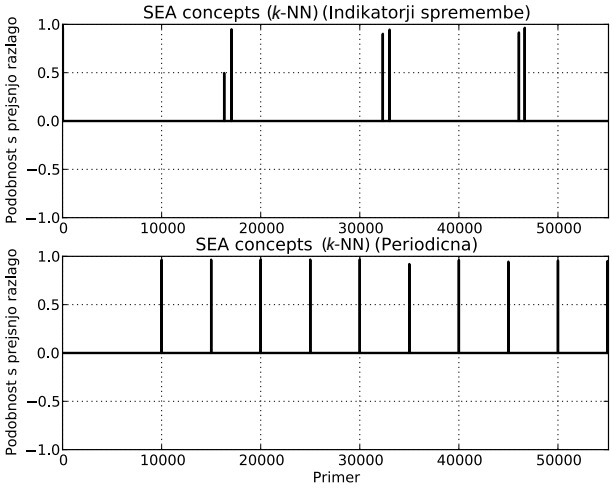
Slika 5.17: Razlage oken algoritma SPC (k-najbližjih sosedov) za podatkovni tok SEA concepts glede na indikatorje sprememb koncepta (1/2).



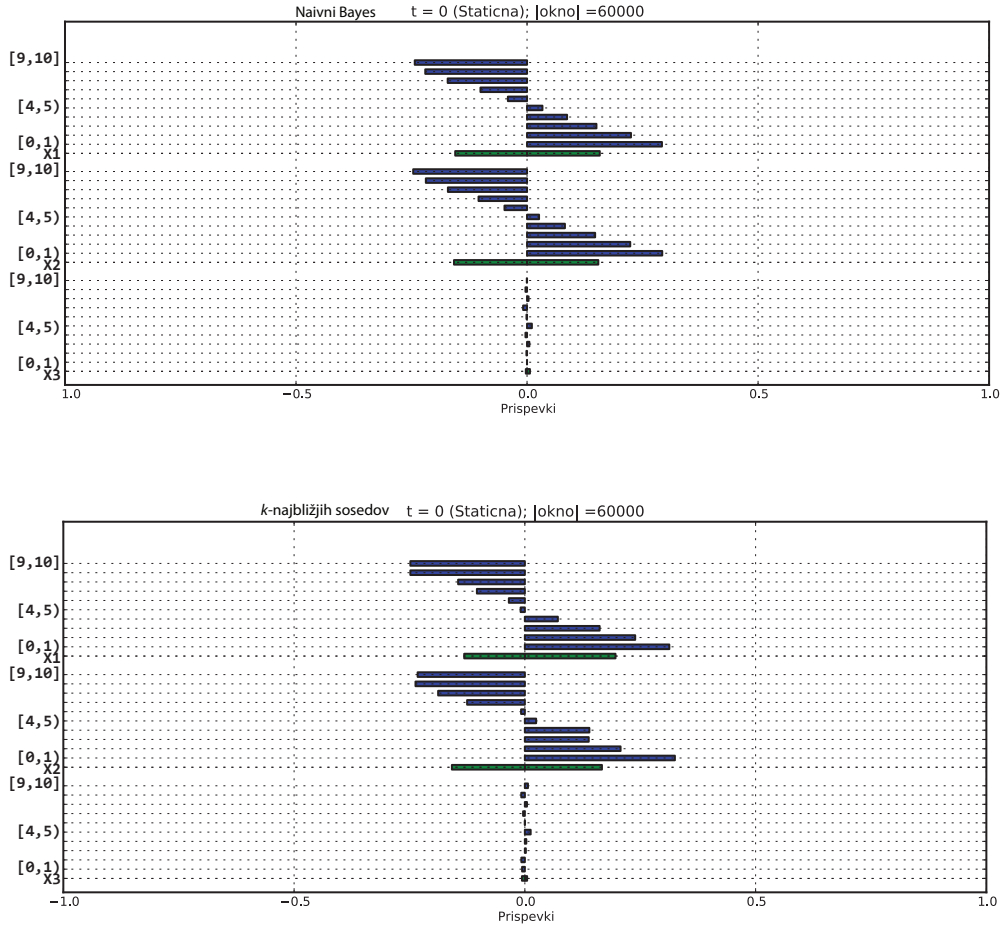
Slika 5.18: Razlage oken algoritma SPC (k-najbližjih sosedov) za podatkovni tok SEA concepts glede na indikatorje sprememb koncepta (2/2).



Slika 5.19: Medsebojna (kosinusna) podobnost zaporednih razlag podatkovnega toka SEA concepts (klasifikator: naivni Bayes)



Slika 5.20: Medsebojna (kosinusna) podobnost zaporednih razlag podatkovnega toka SEA concepts (klasifikator: k-najbližjih sosedov).



Slika 5.21: Razlaga podatkovnega toka SEA concepts, če ga obravnavamo kot statično množico s klasifikatorjema naivni Bayes in k-najbližjih sosedov brez zaznave spremembe koncepta.

jena. Vizualizacijo oken ob spremembi koncepta tudi tu pokvarijo vmesna stanja, ki so vzrok prevelikim odstopanjem.

Razlaga posameznih primerov

Razlage primerov (slike 5.24, 5.25, 5.26 in 5.27) se ujemajo z razlago modela (večja odstopanja vrednosti od 5.0 pomenijo po absolutni vrednosti večji prispevek). Z uporabno naivnega Bayesa bolje zajamemo nepomembnost tretjega atributa.

Ugotovitve

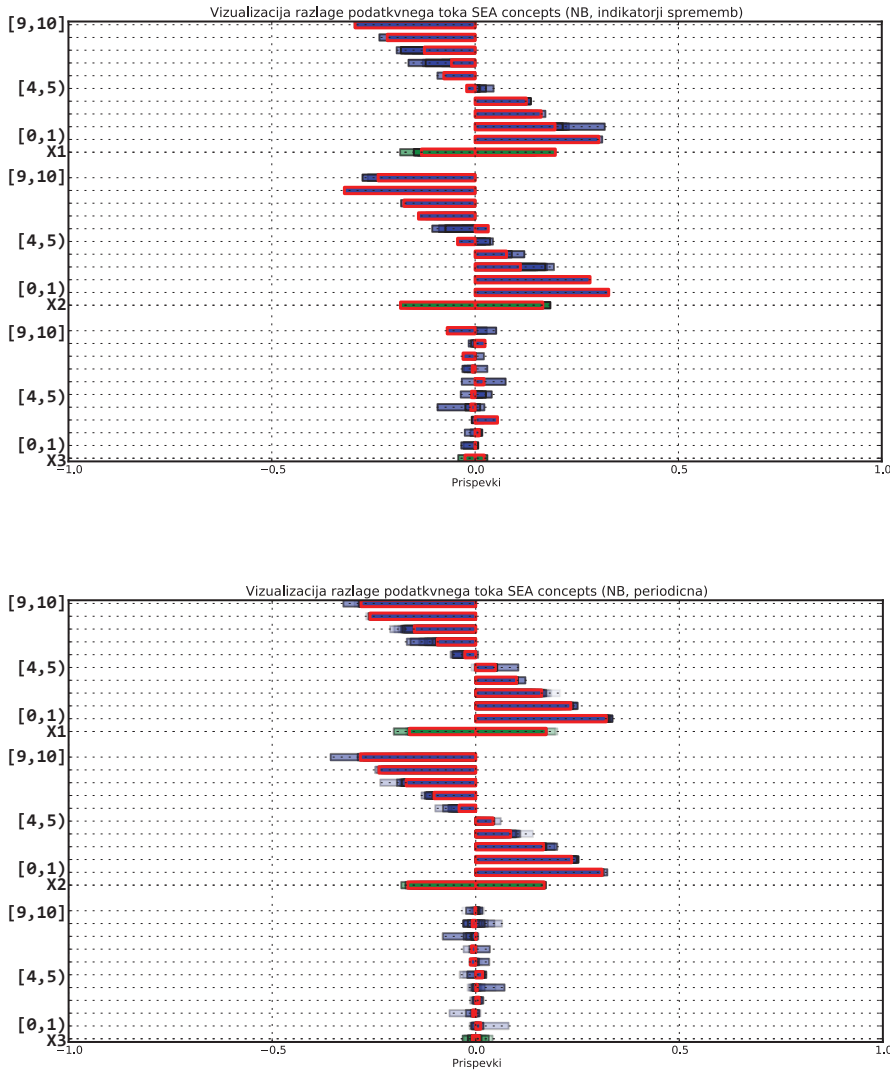
Predlagana metoda je na toku dobro razložila zvezne atribute in pravilno določila nerelevantnost atributa x_3. Razlage oken so si zelo podobne, navkljub spremembam koncepta. SEA concepts je primer podatkovnega toka, kjer tudi z uporabo metode za razlago statičnih množic dobimo zadovoljiv rezultat. Ovira, ki jo premostimo z prilagoditvijo na podatkovne tokove, je časovna zahtevnost klasifikacije in predvsem razlage velike učne množice, ki nastane pri statični obravnavi. Poleg tega nam dobljena vizualizacija daje vpogled v (minimalne) spremembe, ki so se zgodile.

5.3.3 Podatkovna množica Titanic

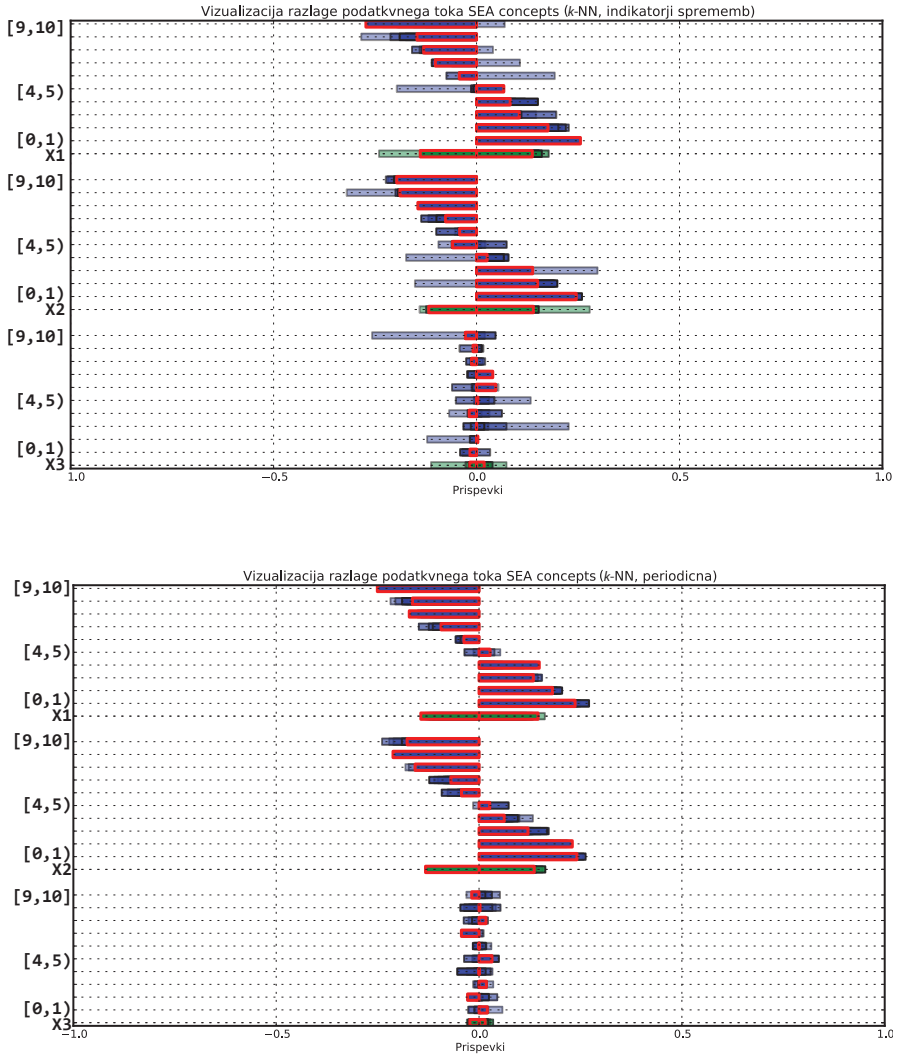
V prejšnjih poskusih smo pokazali, kako statične metode razlage delujejo na podatkovnih tokovih. Na množici Titanic pa metodo testiramo, da bi ugotovili, ali zmore pravilno razložiti tudi statične učne modele. Zanima nas torej, ali je predlagana rešitev posplošitev statične metode za razlago.

Primere permutiramo in jih pošiljamo algoritmu kot podatkovni tok. Nastavimo parametre $max\_okno = 300$ in $\omega = 400$. Napake pri klasifikaciji (slika 5.28) se po začetni fazi učenja ustalijo – ne pride do zaznave spremembe koncepta, kar je pravilno. Začetne faze izpod nadzora ne razlagamo, ker so okna premajhna.

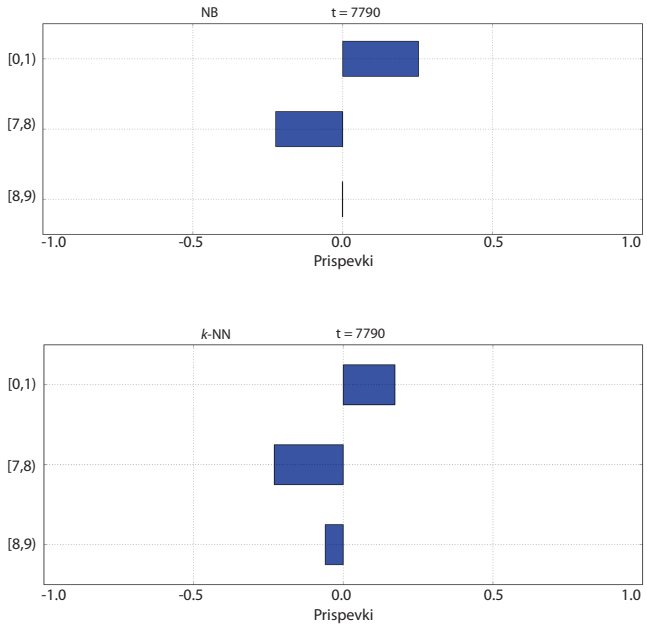
Uvedemo pa periodično razlago (sliki 5.29 in 5.30). Tu pride do izraza



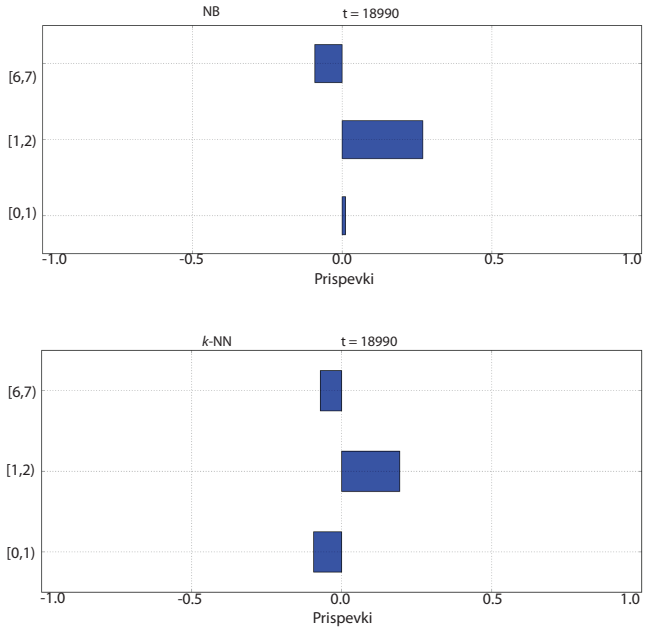
Slika 5.22: Vizualizacija razlage celotnega toka SEA concepts z uporabo predlagane metode (naivni Bayes).



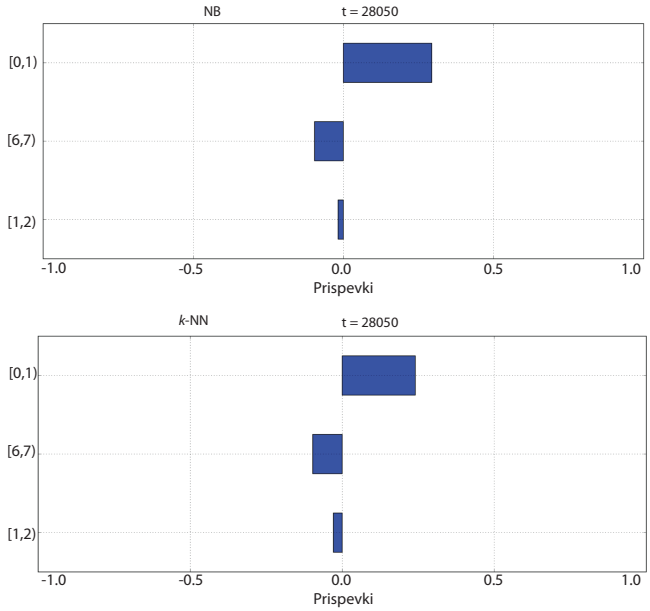
Slika 5.23: Vizualizacija razlage celotnega toka SEA concepts z uporabo predlagane metode (k-najbližjih sosedov).



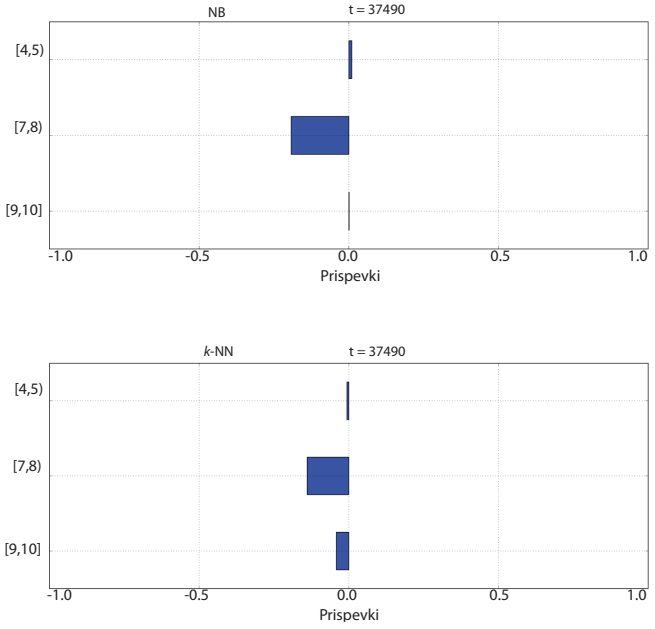
Slika 5.24: Razlaga primera ($x_1 = 0.908375$, $x_2 = 7.065032$, $x_3 = 8.070617$).



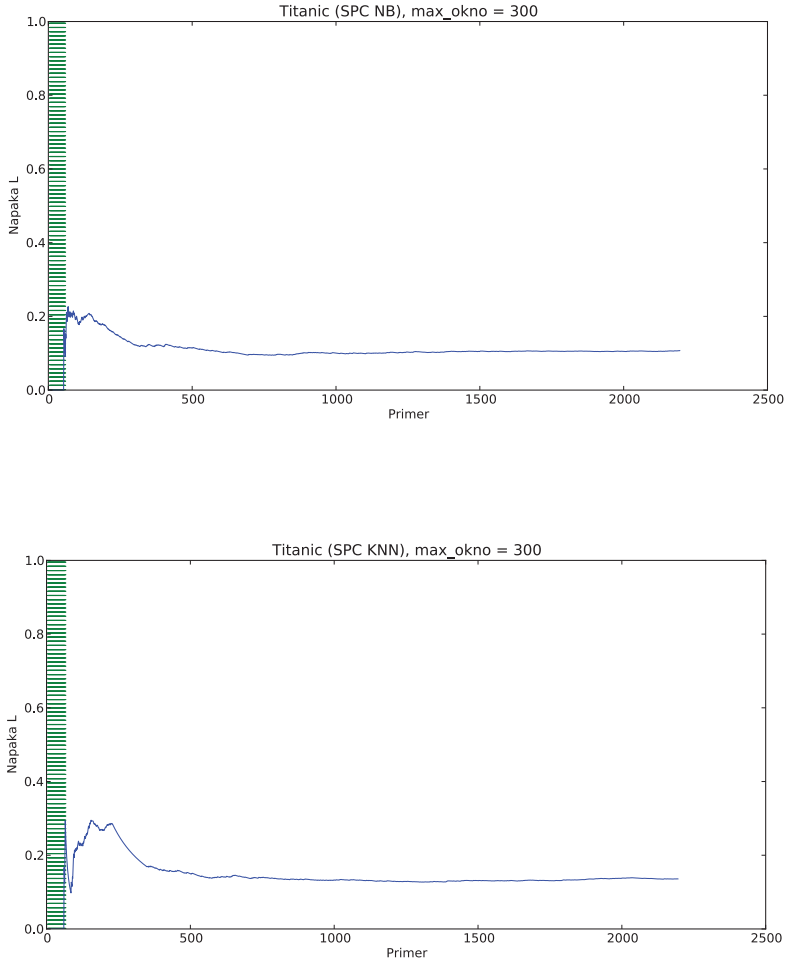
Slika 5.25: Razlaga primera ($x_1 = 6.516275$, $x_2 = 1.760698$, $x_3 = 0.192733$).



Slika 5.26: Razlaga primera ($x_1 = 0.480118$, $x_2 = 6.842060$, $x_3 = 1.865420$).



Slika 5.27: Razlaga primera ($x_1 = 4.801811$, $x_2 = 7.411255$, $x_3 = 9.922116$).



Slika 5.28: Graf povprečnih napak pri klasifikaciji permutiranih primerov statične množice Titanic, ki jih podamo algoritmu SPC kot podatkovni tok. Ovijemo klasifikatorja naivni Bayes in k-najbližjih sosedov.

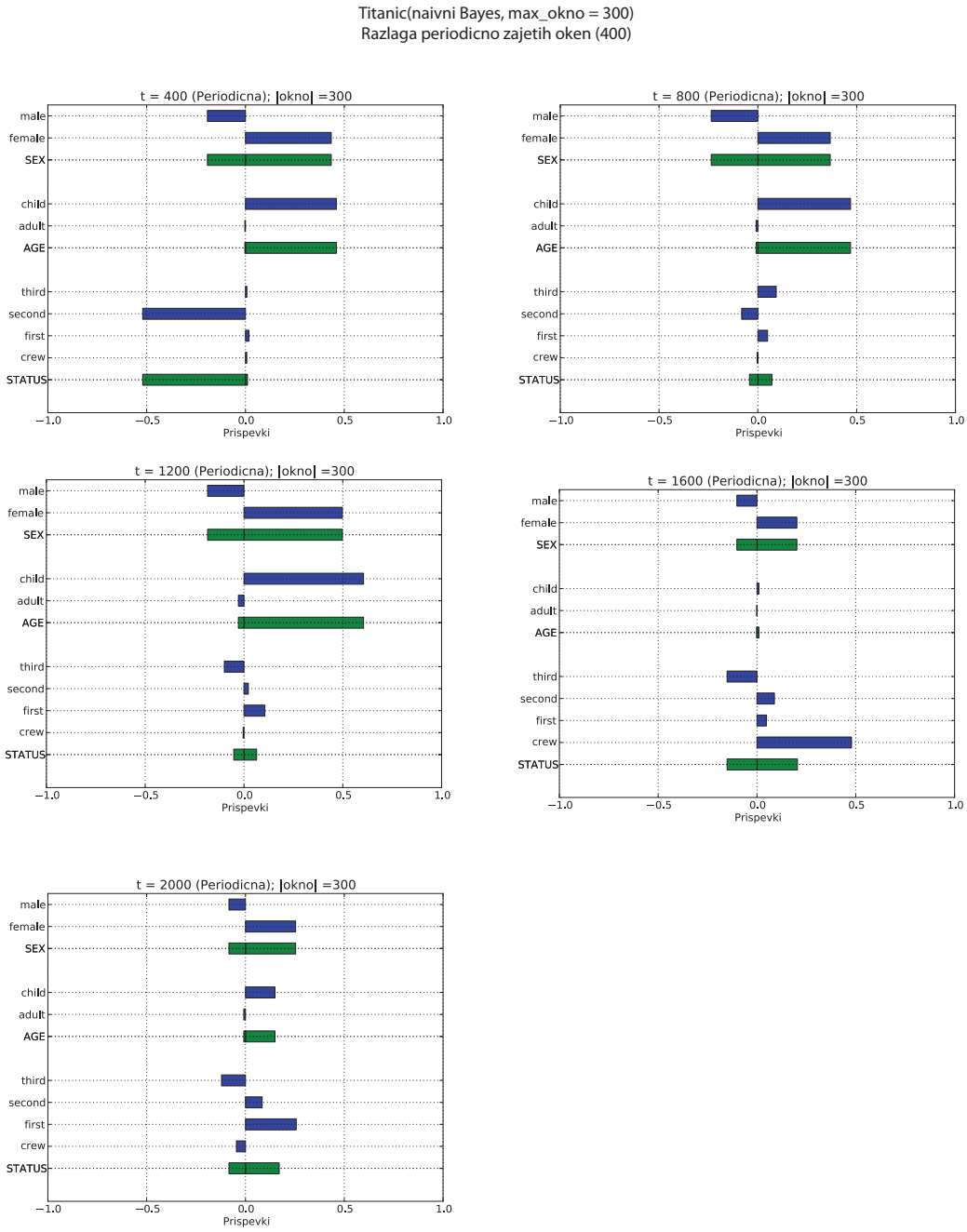
naključni vrstni red primerov, ki jih obravnavamo. Razložena okno v večini primerov kažejo pravilno predznačene prispevke, a z neustreznimi velikostmi. Prihaja do večih nihanj, pri k-NN pa celo do popolnoma napačnih razlag (zadnje okno). V tem primeru zato sklenemo, da razlaga vmesnih oken ni primerna za statične množice. Omeniti pa moramo tudi relativno majhno velikost okna, saj le-to v tej obravnavi nastopa kot vzorec celotne množice. Če bi ga povečali, bi s tem dobili bolj točne rezultate.

Podobnost med zaporednimi razlagami (slika 5.31) je zato manjša, kot smo jo izmerili pri testih s podatkovnimi tokovi. Večje so tudi razlike, če primerjamo razlage klasifikatorjev, saj mera podobnosti med njima znaša le 0.6708.

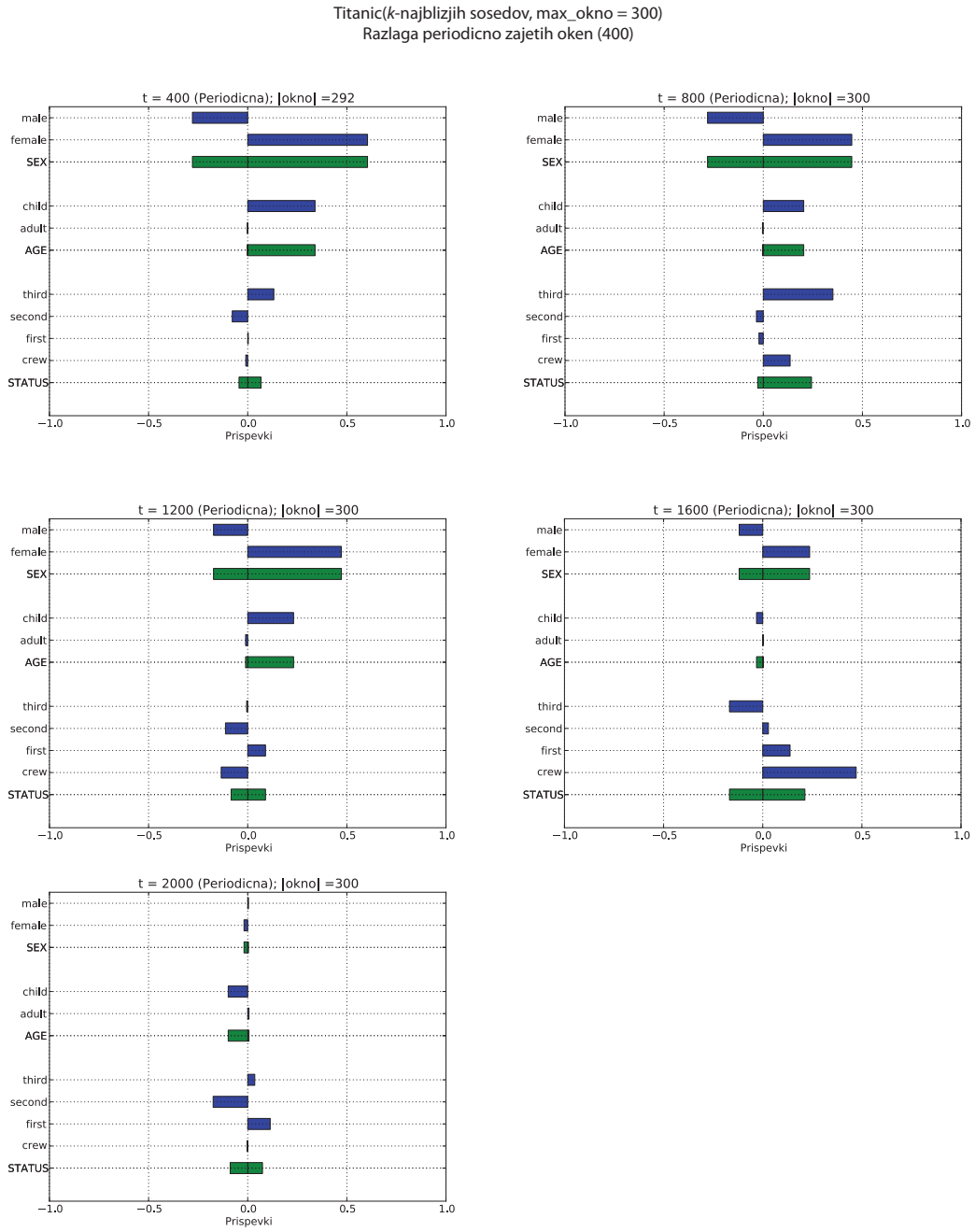
Vse te neskladnosti se prenesejo na razlage posameznih primerov (slika 5.32), kjer zopet opazimo pravilno predznačene, a nenatančno določene prispevke.

Pravilna (statična) obravnava je prikazana na sliki 5.33. Za referenco lahko prispevke pri klasifikaciji z naivnem Bayesom primerjamo z nomogramom za njegovo vizualizacijo [10] (slika 5.34). Z izjemo vrednosti (*status* = *second*) opazimo ujemanje prispevkov z vrednostmi na nomogramu.

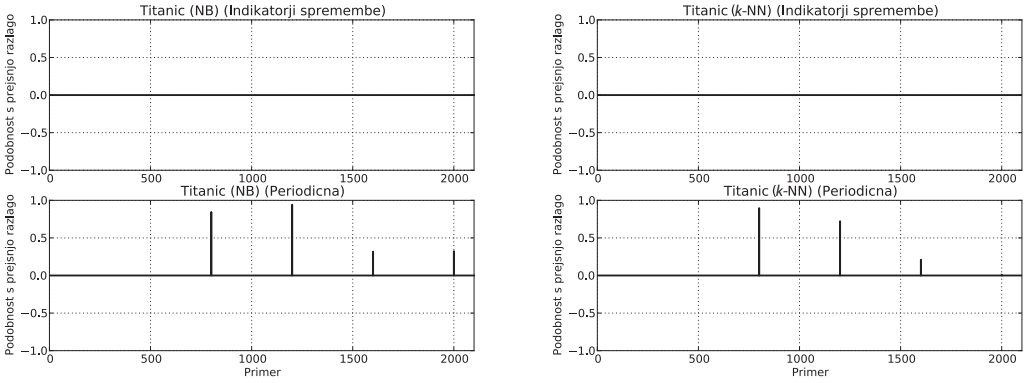
Pri uporabi trodimenzionalne vizualizacije (slika 5.35) se odstopanja, ki smo jih dobili na periodično zajetnih oknih, zgladijo in nam dajo pravilno razlago konceptov za problemsko domeno. Naivni Bayes prednjači po pravilnosti rešitve pred k-najbližjimi sosedi. Za boljši rezultat bi bila potrebna optimizacija faktorja pozabljanja in velikosti drsečega okna. Uporaba predlagane metode se je torej za statično množico Titanic izkazala za primerno, v kolikor se omejimo na končno vizualizirano razlago. Natančnost vmesnih razlag in primerov je slabša kot pri statični obravnavi in predvsem odvisna od velikosti drsečega okna.



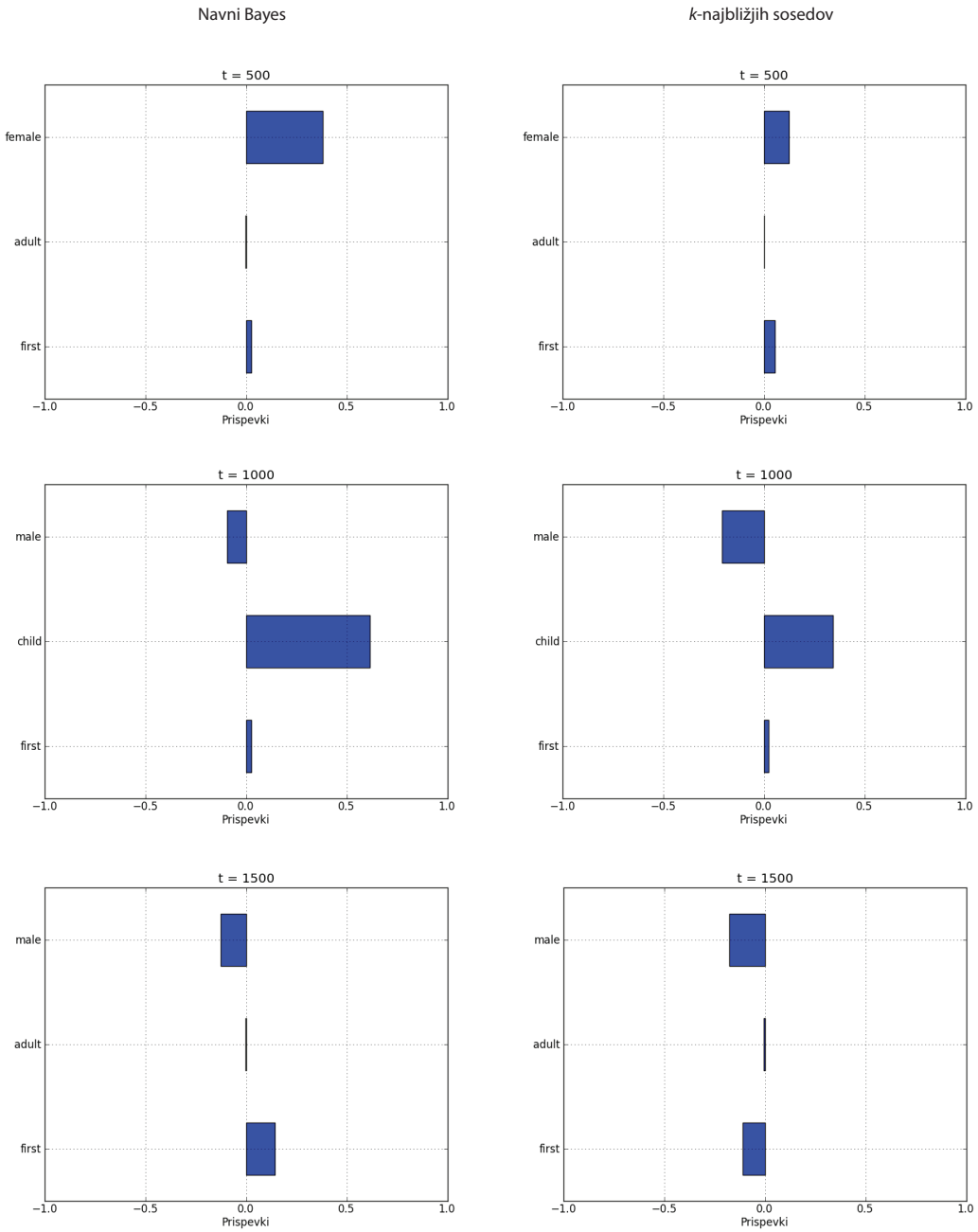
Slika 5.29: Razlage periodično zajetih oken ($\omega = 400$) algoritma SPC (naivni Bayes) za množico Titanic.



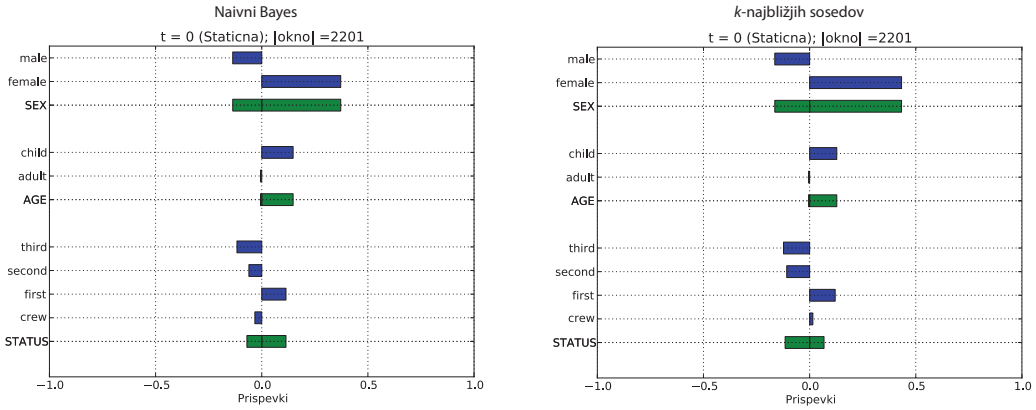
Slika 5.30: Razlage periodično zajetih oken ($\omega = 400$) algoritma SPC (k-najbližjih sosedov) za množico Titanic.



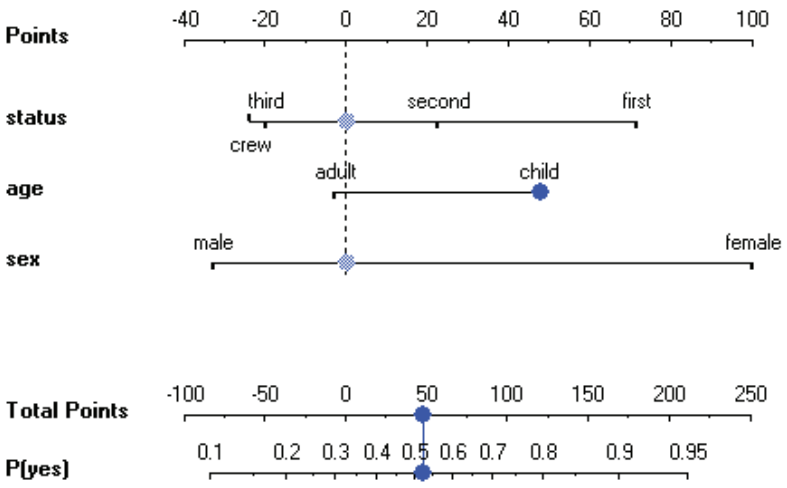
Slika 5.31: Podobnosti med zaporednimi razlagmi učenega modela pri klasifikaciji množice Titanic.



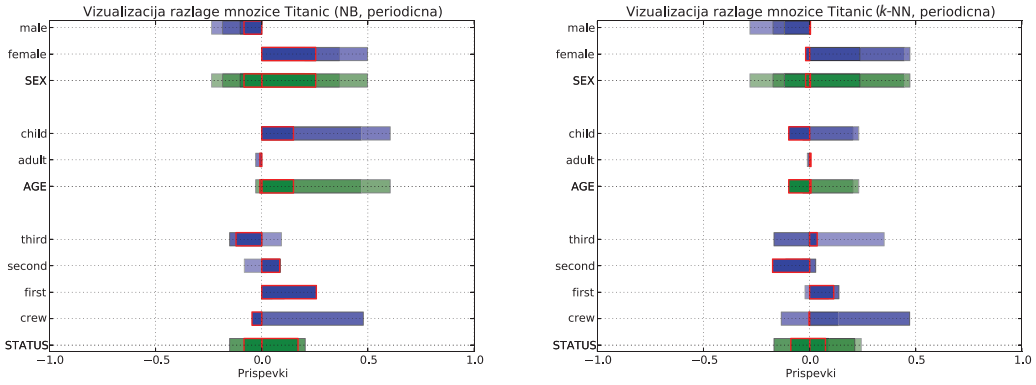
Slika 5.32: Razlaga klasifikacije primerov (*sex = female, age = adult status = first*), (*sex = male, age = child status = first*) in (*sex = male, age = adult status = first*).



Slika 5.33: Razlagi napovednih modelov (naivni Bayes in k-najbližjih sosedov) za statično množico Titanic.



Slika 5.34: Nomogram klasifikacije množice Titanic z naivnim Bayesom [10].



Slika 5.35: Vizualizacija celotnega toka razlag množice Titanic z uporabo predlagane metode (faktor pozabljanja $\gamma = 0.7$).

Poglavje 6

Sklep

Izhodiščne točke diplomskega dela so obsegale preučevanje metod za razlago napovedi, obravnavo inkrementalnih tehnik strojnega učenja in karakterizacijo podatkovnih tokov. Upoštevaje značilnosti slednjih smo razvili posplošeno metodo za razlago inkrementalnih napovednih modelov in klasifikacijo posameznih primerov. Temelj rešitve je bilo statistično nadzorovanje procesov – algoritem za zaznavo sprememb v toku, ki smo ga modificirali in vanj vključili obstoječe metode za razlago. Razlaga podatkovnega toka je tako tudi sama podatkovni tok, v katerem smo računali medsebojno podobnost elementov in ga vizualizirali na več načinov. Z eksperimentalnim delom na naboru inkrementalnih algoritmov in podatkovnih tokov smo pokazali, da metoda uspešno razlaga inkrementalne modele in posamezne primere ter ugotovili, da statične metode razlage niso ustrezne. Skupaj z vizualizacijo, ki se je izkazala za pomembno komponento rešitve, nastalo orodje omogoča vpogled v domeno problema tudi na podatkovnih tokovih, ki postajajo vedno bolj pogosta oblika informacije. Med razvojem metode smo našli prostor za izboljšave, ki predstavljajo predvsem premik od spremljanja vhodnega toka do inkrementalnega dopolnjevanja same razlage in zaznavanja sprememb na njej.

6.1 Končne ugotovitve

Možnost **spremembe koncepta** je značilnost podatkovnih tokov, ki je bila izhodišče ideje za metodo razlage inkrementalnih modelov. Predstavlja namreč glavni vzrok za odpoved statičnih metod razlage na podatkovnih tokovih, kar se je pokazalo tudi v eksperimentalnem delu. Za zaznavo sprememb smo uporabili statistično nadzorovanje procesov, ki je hiter, robusten in posplošen algoritem (statistično nadzorujemo tok klasifikacijskih napak). Ta posplošenost nam je koristila, saj smo ga lahko uporabili kot ovojnico za katerikoli klasifikator. Ob indikacijah sprememb in ob rednih intervalih smo razložili tok na trenutnem oknu in s tem zajeli spremembe v razlagi. Razlaga posameznega primera poteka na enak način kot pri statičnih množicah, le da za učno množico in model vzamemo trenutno okno ter model, naučen na njem. Na ta način smo dobili metodo razlage, ki deluje na poljubnem klasifikatorju in upošteva karakteristike podatkovnih tokov (hitrost, spremembe, omejenost virov).

Razlaga inkrementalnega modela je podatkovni tok, kar pomeni, da smo problem prevedli na znano strukturo, za katero že obstajajo tehnike analize. V diplomskem delu smo uporabili **kosinusno mero podobnosti**, s katero smo ovrednotili razlike med zaporednimi razlagami in iskali ponavljajoče se koncepte (periodičnost). Izmerili smo tudi podobnost razlag različnih klasifikatorjev in ugotovili, da so razlike med njimi majhne ($podobnost > 0.9$), kar dodatno govori v prid posplošenosti metode. Druga tehnika je bila **vizualizacija**, s katero smo rešili problem jasnega prikaza toka razlag, ki ga sestavlja zaporedje velikih vektorjev številskih prispevkov. Predstavlja pomemben del naloge, saj je jasen prikaz vzrokov za odločitev prvotni namen katerekoli razlage. Z rabo tretje dimenzije (barve) smo zajeli časovno komponento in prikazali trende celotnega toka v eni sliki.

Pri **eksperimentalnem delu** se je izkazalo, da je uporaba posebne metode razlage za podatkovne tokove smiselna. Statična obravnava lahko vrne popolnoma napačne rezultate, v najboljšem primeru pa le približno pravilne. Obratno smo kot vhod inkrementalni metodi podali statično množico – te-

daj vmesne razlage oken niso bile zadovoljive, končna vizualizacija pa je uspešno odražala koncepte za problemsko domeno. Odsotnost spremembe koncepta torej ne vpliva na končni rezultat. Moramo pa poudariti, da zaradi majhnosti nabora algoritmov in testnih množic nismo pokrili vseh scenarijev. Manjka nam predvsem nesintetični podatkovni tok z dovolj natančno določenimi točkami spremembe koncepta, da bi lahko sklepali o kvaliteti razlag, ki jih metoda vrne ob splošni rabi. Razlage so bile z manjšimi odstopanji pravilne pri obeh klasifikatorjih. Izbira klasifikatorja direktno vpliva na kvaliteto razlage. Predmet razlage je namreč učni model; če je slab, se ne ujema z koncepti za problemsko domeno, poleg tega pa je pri inkrementalnem učenju lahko dodaten vir napak zaradi lažnih alarmov za spremembo.

6.2 Nadaljnje delo (predlogi za izboljšave)

Očitna lastnost, ki manjka predlaganemu algoritmu je **inkrementalno dopolnjevanje razlage**, ki je najpomembnejša naloga za nadaljnje raziskave. Z njo bi občutno zmanjšali časovno zahtevnost in bolje prilagodili metodo naravi podatkovnih tokov. Trenutna verzija metode deluje na oknih in je zrnat približek idealnemu algoritmu. Z inkrementalnostjo bi dobili tudi možnost učinkovite razlage ob naključnem času (*anytime razlaga*), kar pomeni, da bi ob vsakem novoprispelem primeru lahko hitro vrnili razlago ter jo dodali v podatkovni tok/vizualizacijo.

Drug kompromis, ki smo ga sklenili v predlagani verziji, je reagiranje le na spremembe v vhodnem podatkovnem toku, kar smo kompenzirali z uvedbo periodične razlage. Zanašamo se torej na dejstvo, da sprememba koncepta v klasificiranem podatkovnem toku pomeni verjetno spremembo v razlagi oziroma skušamo uloviti spremembe v periodično postavljena okna. To je dokaj nenatančno in v določenih primerih odpove, saj lahko zaradi manka statistično značilne spremembe v toku klasifikacijskih napak prezremo spremembe v razlagah. Želimo si **zaznavo sprememb koncepta direktno iz toka razlag**, ne iz vhodnega podatkovnega toka. Ta zmogljivost

bo mogoča ob implementaciji inkrementalnega dopolnjevanja razlage, saj bo lahko le takrat možno učinkovito delovanje metode za zaznavo spremembe koncepta, ki leži za razlagami.

Prostor za izboljšavo je tudi v **metodi zaznave spremembe koncepta in prilagoditve nanjo**. SPC je robusten in hiter kot ovojnica za inkrementalne učne algoritme oziroma algoritme z visoko stopnjo granularnosti. Slaba stran te robustnosti pa je včasih neobčutljivost na postopne spremembe. Še ena slabost je, da smo pri shranjevanju podatkov vezani na drseča okna in zato ne moremo uporabljati popolnega spomina (faktorjev pozabljanja). Za inkrementalno dopolnjevanje razlage bi bilo smotrno razviti posebno metodo zaznave pravih sprememb v prispevkih.

Porazdeljeni podatkovni tokovi niso bili predmet naše obravnave in ostajajo odprto področje za raziskave v povezavi z razlago modelov. Veliko neraziskanih možnosti obstaja tudi za **nadaljnjo obravnavo toka razlag**, ki ga predlagana metoda generira. Pozornost bi bilo smiselno posvetiti iskanju periodičnih pojavov in vzorcev v razlagah.

Seznam algoritmov

Literatura

[1] A. Bifet, R. Kirkby. *Data Stream Mining: A Practical Approach*. COSI, 2009. Dostopno na `www.cs.waikato.ac.nz/~abifet/MOA/StreamMining.pdf`

[2] T. Dasu, S. Krishnan1, G. M. Pomann. "Robustness of Change Detection Algorithms", *IDA'11 Proceedings of the 10th international conference on Advances in intelligent data analysis*, št. X, str. 125–137, 2011.

[3] R. O. Duda, P. E. Hart, D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2009.

[4] G. Folino, C. Pizzuti, G. Spezzano, "Mining Distributed Evolving Data Streams Using Fractal GP Ensembles" *Proceedings of the 10th European Conference on Genetic Programming*, str. 160–169, Springer, 2007.

[5] J. Gama. *Knowledge Discovery From Data Streams*. Chapman & Hall/CRC, 2010.

[6] J. Gama, M. M. Gaber. *Learning from Data Streams: Processing Techniques in Sensor Networks*, Springer Verlag, 2007.

[7] J. Gama, P. Medas, G. Castillo P. Rodrigues, "Learning with drift detection", *SBIA Brazilian Symposium on Artificial Intelligence. Springer Verlag 2004*, str. 286–295, 2004.

[8] D. Kifer, S. Ben-David, J. Gehrke, "Detecting Change in Data Streams", *30. Very Large Databases Toronto*, str. 180–191, 2004.

[9] I. Kononenko. *Strojno učenje*. Založba FE in FRI, 2005.

[10] M. Možina, J. Demšar, M. Kattan, B. Zupan, "Nomograms for Visualization of Naive Bayesian Classifier", PKDD '04 Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, str. 337–348, 2004. Dostopno na: `http://eprints.fri.uni-lj.si/154/1/PKDD_camera_mozina.pdf`

[11] R. Sebastiao, J. Gama, "A Study on Change Detection Methods", *New Trends in Artificial Intelligence. 14th Portuguese Conference on Artificial Intelligence. EPIA 2009. Aveiro, October 12-15, 2009. Proceedings*, str. 353–264, 2009. Dostopno na `http://epia2009.web.ua.pt/onlineEdition/NewTrendsInArtificialIntelligence.pdf`

[12] T. Segaran. *Programming Collective Intelligence*. O'Reilly Media, 2007.

[13] E. Štrumbelj, I. Kononenko, M. Robnik Šikonja, "Explaining instance classifications with interactions of subsets of feature values", *Data & Knowledge Engineering*, št. 68, zv. 10, str. 886–904, 2009.

[14] E. Štrumbelj, I. Kononenko, "An Efficient Explanation of Individual Classifications using Game Theory", *Journal of Machine Learning Research*, št. 11, str. 1–18, 2010.

[15] V. Vovk, A. Gammerman, G. Shafer. *Algorithmic Learning in a Random World*. Springer, 2005.

[16] B. P. Welford, "Note on a method for calculating corrected sums of squares and products", *Technometrics* št. 4, str. 419–420, 1962.

[17] I. Zliobaite. *Learning under Concept Drift: an Overview*. Faculty of Mathematics and Informatics, Vilnius University, 2009.

[18] A. Ajanki, *Wikimedia Commons*. Dostopno na `https://en.wikipedia.org/wiki/File:KnnClassification.svg`

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: 00007/2012
Datum: 10.04.2012

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

Kandidat: **JAKA DEMŠAR**

Naslov: **RAZLAGA NAPOVEDNIH MODELOV IN POSAMEZNIH NAPOVEDI PRI INKREMENTALNEM UČENJU**

EXPLANATION OF PREDICTIVE MODELS AND INDIVIDUAL PREDICTIONS IN INCREMENTAL LEARNING

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

Metodologija razlage napovednih modelov in posameznih napovedi, ki je razvita za učenje iz stacionarnih podatkov (Štumbelj in Kononenko, 2010), je orodje, ki omogoča vpogled v delovanje modela ter omogoča ekspertom večje razumevanje problemske domene.

Kandidat naj prilagodi obstoječo metodologijo področju učenja iz podatkovnih tokov, kjer smo omejeni s procesorskim časom in spominom, naborom inkrementalnih modelov, prisotne pa so tudi spremembe v porazdelitvah učnih podatkov. Kandidat naj evalvira uspešnost delovanja prilagojene metode na naboru inkrementalnih učnih algoritmov in predlaga naj vizualizacijo spremembe razlage modela skozi čas.

Mentor:

doc. dr. Zoran Bosnić

Dekan Fakultete za računalništvo in informatiko:

prof. dr. Nikolaj Zimic

Dekan Fakultete za matematiko in fiziko:

akad. prof. dr. Franc Forstnerič