Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

David Šenica

Zasnova dinamičnega dušilnika v interoperabilnem sistemu

diplomsko delo

interdisciplinarni univerzitetni
študijski program prve stopnje
računalništvo in matematika

Mentor: izr. prof. dr. Matjaž Kukar

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Zasnujte in izvedite dinamični dušilnik poizvedb na podlagi strojnega učenja iz zgodovinskih podatkov. Dušilnik naj bo sposoben napovedati možnost zasičenja vira in ustrezno ukrepati. Definirajte kriterije uspešnosti in z njimi preverite delovanje dušilnika v praktičnem, vsaj simuliranem, okolju.

Kazalo

Povzetek

Naslov: Zasnova dinamičnega dušilnika v interoperabilnem sistemu

Avtor: David Šenica

Danes poznamo veliko informacijskih sistemov, ki hranijo in vračajo različne informacije. Da bi odjemalcem omogočili lažje pridobivanje podatkov iz tovrstnih sistemov, jih povežemo s pomočjo interoperabilnega informacijskega sistema. Ta iskanje razširi na vse povezane sisteme in vrne sestavljen odgovor. Odjemalcu tako ni potrebno skrbeti za posebnosti velikega števila sistemov, da bi pridobil kombinirano informacijo več sistemov. Interoperabilnemu sistemu izziv predstavlja dosegljivost in odzivnost informacijskih sistemov, na katere se povezuje. Ti so lahko le deloma odzivni ali celo nedostopni. Da bi interoperabilnemu sistemu preprečili preobremenitev oddaljenega informacijskega sistema, bi bilo potrebno dušiti povpraševanja za ta sistem. V diplomski nalogi smo zgradili dušilnik, ki zna prepoznati nedelujoč oziroma preobremenjen sistem in v takem primeru zmanjšati število povpraševanj na ta vir. V ta namen smo zbirali podatke, jih analizirali in pripravili za učenje napovednih modelov. Rezultati kažejo, da dušilnik zna prepoznati nedelujoč vir in mu zmanjšati število povpraševanj.

Ključne besede: dušilnik, interoperabilni sistem, strojno učenje.

Abstract

Title: Dynamic throttling in an interoperabile system

Author: David Šenica

Nowdays we know a lot of information systems that store and provide different information. To make it easier for remote clients to get information from those systems, we connect them with the help of an interoperable information system, which expands the search to every connected system and returns a composed response. That way remote clients don't have to worry about specific information systems to get combined information from them. Interoperable informative system is challenged by the avaliability and resposiveness of information system to which it connects. They can sometimes only be partly responsive or even unaccesible. To prevent interoperability system to overload remote information system, we would like to throttle requests to that system. We built a throttling subsystem that is capable of recognizing an inoperative or overloaded system and in that case lower its number of demands. That's why we collected production data, analysed it and prepared it for learning prediction models. The results show that throttle can reliably recognise or predict pending inoperative source and lower its demands.

Keywords: throttle, interoperable system, machine learning.

Poglavje 1

Uvod

1.1 Opis interoperabilnega sistema

Interoperabilni informacijski sistem [1] omogoča povezovanje med informacijskimi sistemi brez večjega napora za posameznika. Uporabniku poenostavi pridobivanje podatkov iz različnih virov, saj se mu ni potrebno ukvarjati s kompleksnostjo pridobivanja podatkov in tehničnimi značilnostmi posameznega vira (lokacija, transportni protokol,...) in se tako lahko osredotoči na interpretacijo vsebine podatkov.

Primer takšnega sistema je AGRIS [2]. AGRIS (angl. International System for Agricultural Science and Technology) je sistem, ki skrbi za pridobivanje večjezičnih bibliografskih zapisov o kmetijstvu. Vanj je vključenih preko 350 institucij iz 140 različnih držav. Od maja 2018 ima na razpolago več kot 9211000 bibliografskih zapisov. Uporabniku omogoča, da poizvedbo napiše v enem izmed podprtih jezikov, iskanje pa se potem izvrši tudi nad zapisi v drugem jeziku. AGRIS ne shranjuje nobenega zapisa, ampak jih pridobi od drugih ponudnikov, ki so vanj vključeni.

Primer interoperabilnega sistema v Sloveniji je sistem Ministrstva za javno upravo, ki predstavlja centralni informacijski sistem za zbiranje poizvedb, imenovan Pladenj [3].

1.2 Pomanjkljivost interoperabilnega sistema

Interoperabilni sistemi uporabnikom lajšajo dostop do podatkov, zbranih v različnih virih, z oddajo le ene zahteve. Viri v večini primerov predstavljajo oddaljen sistem, s katerim upravlja druga organizacija kot z interoperabilnim sistemom. V takem sistemu viri predstavljajo kritično točko uspeha, saj v primeru, da en vir ne deluje pravilno, uporabnik pridobi le delni odgovor. Poleg interoperabilnega sistema se lahko na iste vire povezujejo tudi druge aplikacije, ki lahko vir še dodatno upočasnijo.

Viri, na katere se sistem povezuje, so lahko različno zmogljivi. Delovanje lahko pohitrimo z vzporednimi klici na vire, vendar pa je celotna odzivnost še vedno odvisna od najšibkejšega vira. V primeru nedelovanja vira je potrebno poskrbeti za kasnejšo pridobitev podatkov. Eden izmed možnih pristopov k reševanju tovrstnih težav je izdelava samo-ponovitvene vrste (ang. autoresume queue). Ta vrsta zbira neuspešne poizvedbe in poskrbi, da se te ponovno izvedejo, ko se vir ponovno vzpostavi. Dodatno kompleksnost taki zasnovi prinese komponenta za časovno zakasnitev poizvedb. Samo-ponovitvena vrsta, v primeru velike količine poizvedb v vrsti, oddaljenega vira ne obremeni do take mere, da bi ga lahko ponovno onesposobila. Pri zasnovi dušilnih komponent bi si želeli izdelati sistem, ki bi znal omejiti število poizvedb na posamezni vir glede na njegovo trenutno odzivnost. To pomeni, da v primeru povečevanja časa obdelave ene poizvedbe, zmanjšamo frekvenco izvajanja na viru in mu tako omogočimo, da zopet deluje normalno.

1.3 Namen

Namen diplomske naloge je zasnovati komponento, ki bo na podlagi preteklih podatkov omogočala napoved ali bodo nadaljnje poizvedbe preobremenile oddaljen vir. Zasnovo bomo v praksi preizkusili na podatkih Slovenskega interoperabilnega sistema Pladenj [3]. Z izdelavo take komponente želimo zmanjšati število neuspešnih poizvedb, ki končajo v samo-ponovitveni vrsti

in s tem zmanjšati čas čakanja na odgovor celotnega postopka.

Zasnova komponente je bila že predstavljena na konferenci Dnevi slovenske informatike 2018 [4]. Avtorji v prispevku predstavijo zasnovo komponente na IS Pladenj na podlagi strojnega učenja, ki napoveduje prepustnost postopkov, sama implementacija komponente pa je bila narejena v okviru diplomske naloge. Tu smo metričnim podatkom dodali novo tabelo, ki olajša izračun novih atributov-značilk.

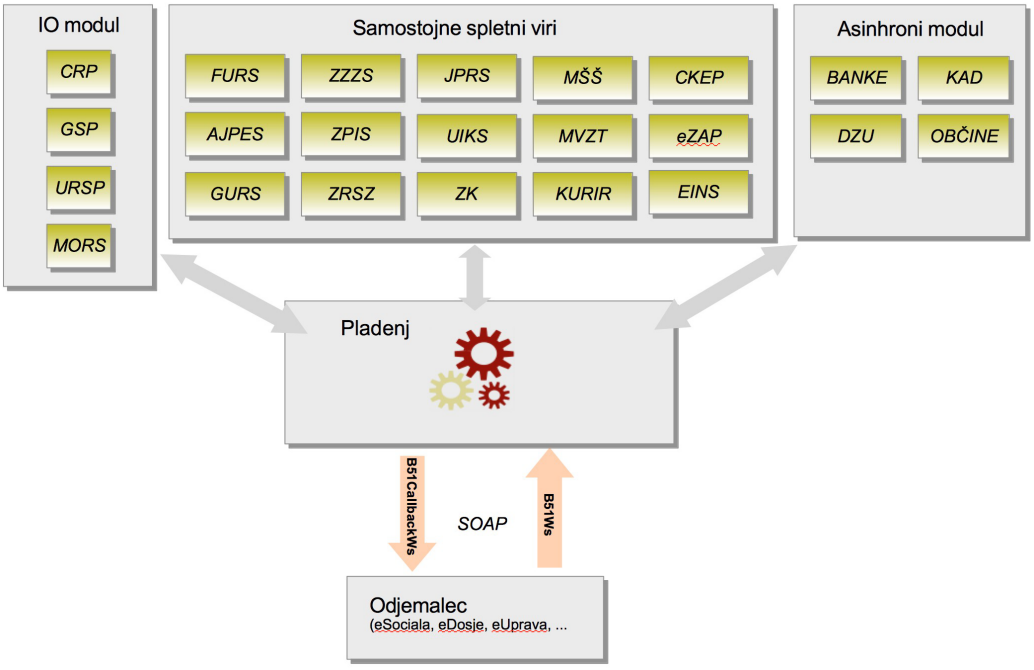
V diplomi zastavljeni problem rešujemo z gradnjo komponente, ki bi v samem bistvu omogočila dinamično napoved prepustnosti posameznih poizvedb na oddaljene vire. V praksi to predstavlja izdelavo komponente za dušenje poizvedb (ang. throttling). Na primeru IS Pladenj lahko tako komponento razvijemo s pomočjo statistične analize preteklih podatkov, ali s pomočjo algoritmov strojnega učenja izdelamo napovedni model, ki bo znal napovedati, ali bi poizvedbe novega postopka preobremenile vir.

V diplomski nalogi želimo s pomočjo produkcijskih podatkov IS Pladenj, zbranih od 8.3.2018 do 30.7.2018, preveriti, ali lahko s pomočjo strojnega učenja bolje prepuščamo oziroma dušimo postopke kot z uporabo napovedi na podlagi statistične analize preteklih podatkov. V naslednjih poglavjih bodo tako podrobneje predstavljeni delovanje IS Pladenj in njegova problematika (poglavje 2), izboljšava in merjenje uspešnosti (poglavje 3), potek zajema podatkov in njihova predstavitev (poglavje 4), zasnova dušilnika (poglavje 6), zajem in priprava podatkov za učenje (poglavje 5) in uporaba modela, ter potek izgradnje modela (poglavje 6) in primerjava s statističnim odločanjem (poglavje 7).

Poglavje 2

IS Pladenj

V naslednjih odstavkih bomo predstavili delovanje in problematiko IS Pladenj. IS Pladenj je centralni, horizontalni informacijski sistem za izvajanje elektronskih poizvedb, ki temelji na osnovi najsodobnejših konceptov storitveno usmerjene arhitekture (ang. service-oriented architecture). Omogoča dinamično gradnjo postopkov za varno in zanesljivo hranjenje pridobljenih podatkov iz različnih podatkovnih virov v realnem času [3]. Zagotavlja enotno vhodno točko na osnovi SOAP (Simple Object Access Protocol) spletne storitve in enotno vhodno podatkovno XSD (XML Schema Definition) shemo za vse odjemalce. Je izredno fleksibilen, saj preko administrativnega modula omogoča enostavno dodajanje novih podatkovnih virov in njihovo vključevanje v postopke. Administracija omogoča tudi gradnjo medsebojno odvisnih poizvedb, kar pomeni da je lahko odgovor ene poizvedbe vhodni podatek v drugo poizvedbo. Na ta način lahko ustvarimo zelo kompleksne postopke za pridobivanje podatkov.



Slika 2.1: Osnovno delovanje Pladnja [3]

2.1 Delovanje

IS Pladenj se povezuje na več kot 50 podatkovnih virov in odjemalcem omogoča pridobivanje podatkov iz virov preko postopkov, ki se oddajo preko spletnega vmesnika. Vsak postopek je lahko sestavljen iz ene ali več poizvedb na vir. Postopki se v realnem času preslikajo v BPM (ang. Business process management) [5] procese. Pri tem se upošteva vrstni red izvajanja poizvedb, način izvajanja (vzporedno, zaporedno), kombiniranje poizvedb iz različnih postopkov, itd. BPM poskrbi tudi, da se bo poizvedba, v primeru neuspešnega odgovora (ko je prišlo do napake pri dobivanju podatka z vira) samodejno ponovno izvedla.

Vsak vir, ki se želi priključiti v IS Pladenj, mora ustrezati specifikaciji v dokumentu GTZ (Generične Tehnološke Zahteve) [6], kjer so opisane podrobnosti za priključitev. V dokumentu ni točno določeno število poizvedb, ki naj bi jih bil vir zmožen obdelati v eni sekundi. Ta podatek zato, glede na

pričakovano število klicev vira, določi naročnik ob razpisu. Ker so viri med seboj različnokrat klicani, ni potrebe po tem, da bi bili vsi isto zmožni.

Vsak vir, na katerega se IS Pladenj povezuje, ima svojo čakalno vrsto, ki omogoča tudi hkratno pošiljanje več poizvedb. Vsaki vrsti lahko določimo maksimalno število vzporednih klicev na posamezen vir. Število klicev se določi preko administracije ob prijavi novega vira v sistem, glede na to, kako zmogljiv je vir in koliko drugih sistemov se nanj še povezuje. Določanje poteka glede na specifikacijo in testiranje vira, preden je ta dostopen v produkcijskem okolju. Zaradi večje kontrole nad spremembami vira se parameter čakalnih vrst popravlja ročno.

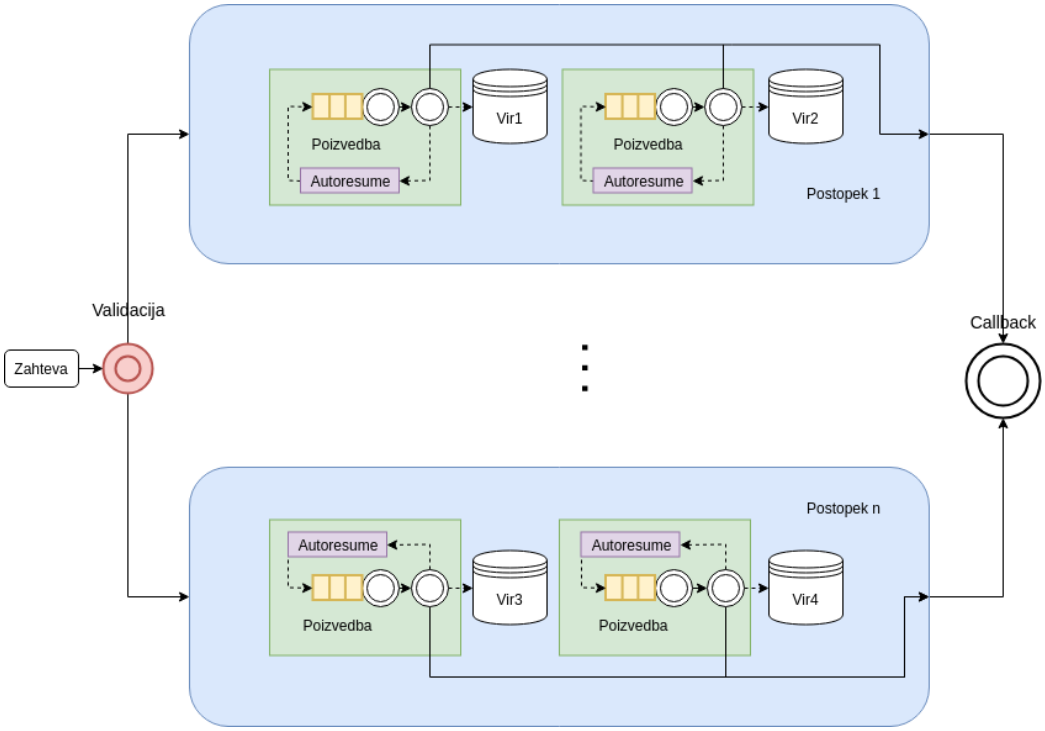
V primeru neuspešnega odgovora vira gre poizvedba v tako imenovano samo-ponovitveno vrsto. Ta poskrbi, da se po določenem času poizvedba ponovno izvede. Vsakič ko ista poizvedba pride v samo-ponovitveno vrsto, se ji poveča čas čakanja pred ponovnim klicem. Ker bi bilo nesmiselno, da bi neka poizvedba v nedogled prihajala v vrsto, imamo omejeno število ponovitev iste poizvedbe.

Sekvenčni potek potovanja zahtevkov v sistemu Pladenj lahko opišemo na primeru poteka enega zahtevka:

- v sistem pride zahtevek v obliki XML(Extensible Markup Language), preveri se mu pravilnost vhodnih podatkov in avtorizacija,
- iz njega se izluščijo relevantni podatki iz katerih se oblikuje postopek,
- shranijo se podatki, ki ji lahko merimo na postopku,
- iz postopka se oblikujejo poizvedbe na vire, katerih odgovore Pladenj zakodira in shrani v začasno transakcijsko zbirko,
- poizvedba kliče enega ali več virov po informaciji, ki jo odjemalca zahteva,
- za vsak odgovor poizvedbe IS Pladenj obvesti odjemalca o uspešnosti, ta pa ima možnost, da prenese in dekodira podatke ali da, še preden se celotni postopek zaključi, zaustavi nadaljnje poizvedbe,

- shranijo se podatki, ki jih lahko merimo za posamezno poizvedbo,
- ko odjemalec prejme vse želene podatke in teh ne potrebuje več, se njihova vsebina izbriše iz Pladnja, meta podatki se arhivirajo.

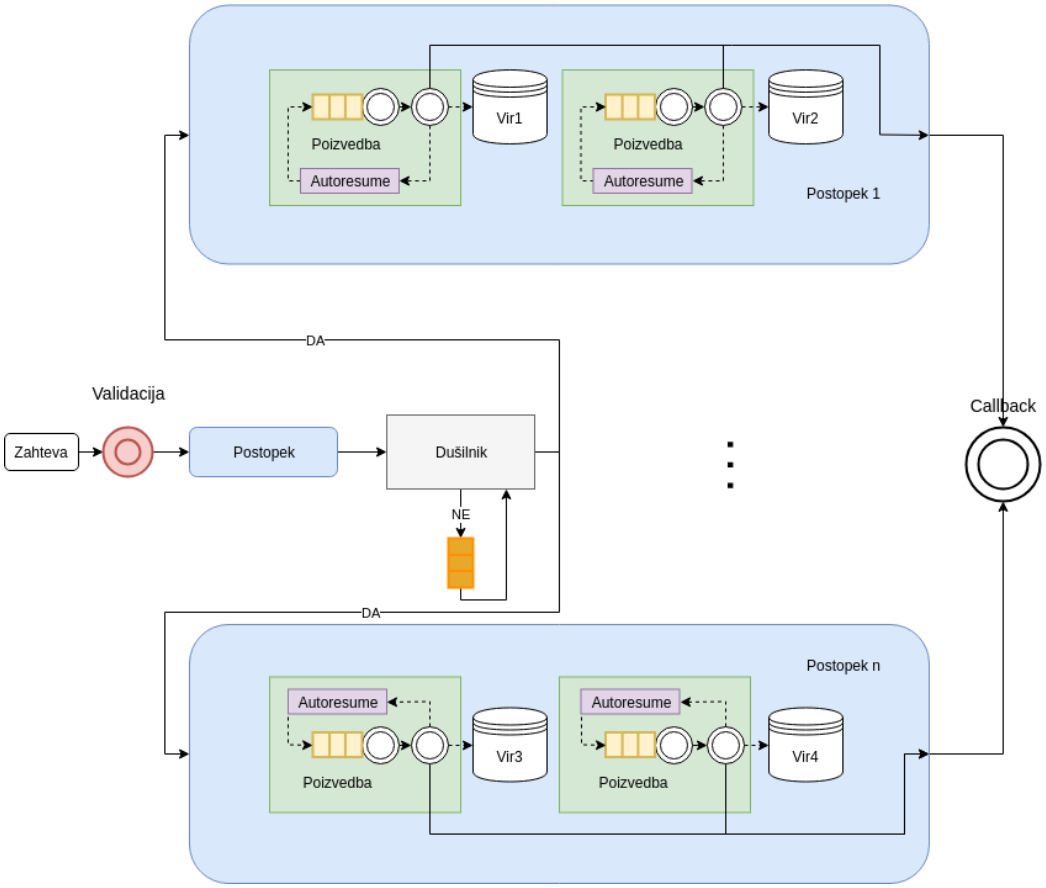
Opisano delovanje prikazuje slika 2.2. Z modro barvo sta označena postopka. Vsak postopek se razdeli na eno ali več poizvedb, ki so označene z zeleno barvo. Vsaka poizvedba, pred klicem vira, vstopi v vrsto (označeno z rumeno barvo). V primeru, da ni napake, pošljemo odgovor poizvedbe naprej do povratnega klica (ang. callback), ki združi odgovore za vsak postopek in jih vrne uporabniku. V primeru napake poizvedba konča v samo-ponovitveni vrsti, ki je označena z vijolično barvo.



Slika 2.2: Potek postopka

Slika 2.3 prikazuje potek postopka z dušilnikom. Razlika v primerjavi s potekom postopka, prikazanim na sliki 2.2, je v tem, da mora vsak postopek,

preden pride v IS Pladenj še skozi dušilnik, ki pove ali lahko gre naprej ali pa začasno konča v čakalni vrsti (označena z oranžno barvno). Ne dušimo torej le posamezne poizvedbe, temveč celotne postopke.



Slika 2.3: Potek postopka z dušilnikom

2.2 Pomankljivosti

IS Pladenj ni edini sistem, ki se povezuje na podatkovne vire, zato se jim stalno spreminjata zmogljivost in odzivnost. Zaradi tega lahko drugi sistemi (ali IS Pladenj sam) preobremenijo vir in s tem povečajo čas vrnitve odgovora. Za optimalno delovanje bi tako morali večkrat ročno usklajevati parametre čakalnih vrst. Pri selitvi vira na močnejše strežnike je ponovno

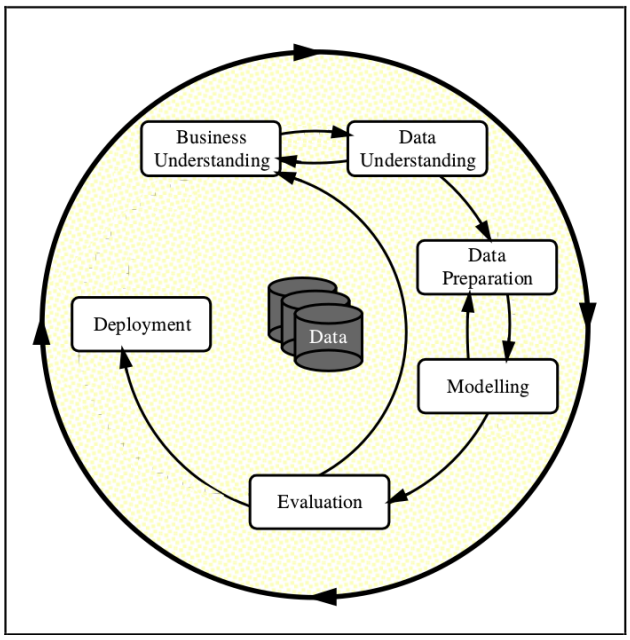
potrebno ročno spreminjati parametre (na primer število vzporednih klicev), da omogočimo večjo prepustnost. Pri tem opazimo, da sta dostopnost in odzivnost virov ključna za uspešno delovanje sistema.

Pri neuspešnem odzivu vira IS Pladenj s samo-ponovitveno vrsto poskrbi, da kasneje ponovno pridobi odziv vira. Lahko pa se zgodi, da je ob ponovnem klicu poizvedb iz samo-ponovitvene vrste, ta tako polna, da vir preobremenimo in mu zmanjšamo odzivnost, oziroma ga popolnoma onemogočimo. To se zgodi takrat, ko v samo-ponovitveni vrsti čaka več poizvedb na vir, ki so se tam nakopičile zaradi nedelovanje vira.

Poglavje 3

Zasnova dušilnika

S pomočjo metodologije CRISP-DM (Cross-industry standard process for data mining) [7] smo izdelali načrt in izvedbo zasnove dušilnika prepustnosti. CRISP-DM zagotavlja strukturiran pristop k izdelavi projekta podatkovnega rudarjenja. V tem poglavju je opisan prvi korak CRISP-DM-a, tj. poslovni vidik, izvedba ostalih pa je predstavljena v naslednjih poglavjih.



Slika 3.1: Potek korakov CRISP-DM [7]

3.1 Razumevanje poslovnega vidika

Naš namen je izboljšati uporabo informacijskega sistema Pladenj. IS Pladenj je interoperabilni sistem Ministrstva za javno upravo. Omogoča dinamično gradnjo postopkov za varno in zanesljivo hranjene podatkov, pridobljenih iz različnih podatkovnih virov v realnem času. Postopki se oddajo preko spletnega vmesnika. Vsak postopek je zgrajen iz ene ali več poizvedb, ki kličejo vire po informacijah. Celotno delovanje in pomanjkljivosti so podrobneje opisani v poglavju 2.

Radi bi zmanjšali povprečen čas izvajanja enega postopka, ki ga odjemalec proži in število poizvedb, ki končajo v samo-ponovitveni vrsti (v primeru neuspešno pridobljenega odgovora ta poskrbi za kasnejšo pridobitev podatkov z vira). To želimo doseči tako, da bi IS Pladenj zadržal postopek, čigar poizvedba bi preobremenila vir in bi s tem zmanjšala čas odziva oziroma naredila vir popolnoma nedostopen. Postopka ne bi v celoti zavrnili, ampak bi ga dali v čakalno vrsto in ga sprožili, ko bodo viri zopet delovali. S tem bi zmanjšali tudi število poizvedb, ki končajo v samo-ponovitveni vrsti, kamor pridejo zaradi nepravilnega delovanja vira.

V ta namen bi zgradili dušilnik, ki bo glede na stanje IS Pladenj prepuščal ali zadrževal postopke. Zgradili ga bomo na dva načina. Prvi bo na podlagi statističnega odločanja, drugega pa bomo zgradili s pomočjo algoritmov za strojno učenje. Ker je IS Pladenj zasnovan v programskem jeziku Java smo se odločili, da bomo za učenje modela uporabili MOA (Massive online analysis) [8], ki je odprto-kodno ogrodje za rudarjenje podatkovnih tokov, napisano v Javi. Implementiranih ima že nekaj inkrementalnih algoritmov strojnega učenja, ki bodo prišli prav pri nadgrajevanju modela, saj se ga ne želimo vedno znova učiti od začetka.

Uspešnost projekta bomo merili po dveh kriterijih:

- zmanjšanem številu poizvedb v samo-ponovitveni vrsti,
- zmanjšanju povprečnega časa obdelave enega postopka.

Preverili jo bomo tako, da bomo še naprej zbirali metrične podatke in iz njih

izluščili čas trajanja enega postopka ter število v samo-ponovitveni vrsti in primerjali s podatki pred implementacijo dušilnika.

Poglavje 4

Podatkovni vidik

Za realizacijo bolj natančnega odločanja ali napovedi prepustnosti oddaljenih virov moramo razpolagati s podatki, ki omogočajo statistično analizo delovanja sistema in hkrati opisujejo ne samo stanje sistema IS Pladenj, pač pa tudi stanje virov. Ker ne moremo direktno pridobiti metrike za posamezen vir, bomo metriko merili glede na njihovo odzivnost. Pri tem si ne želimo dodatno obremenjevati virov, zato bomo za stanje uporabili realne klice poizvedb in obremenjenost IS Pladenj. Poleg odzivnosti virov bomo spremljali še postopke in BPM procese.

Zajeli smo 4 različne sklope podatkov. Prvi sklop zajema podatke na začetku, ko postopek pride v sistem, v drugi sklop spadajo podatki v procesnem stanju, ki opisuje delovanje atomičnih stanj procesa, tretji sklop obsega podatke, ki jih pridobimo s klici spletnih storitev, zadnji pa so podatki nad sinhronimi klici, saj ti nimajo zapisa v procesnem stanju, ker se poizvedba pošlje direktno na vir.

Ime	Opis
id_trans_postopek	Enolična številka transakcije in postopka
start_time	Začetni čas sprejetega postopka
start_tip	Način poizvedovanja za začetni postopek
sif_odjemalca	Identifikacijska številka odjemalca
tip_transakcije	Zaznamek tipa transakcije
bpm	Zaznamek ali se pri asinhronem poizvedovanju uporabi BPM proces
id_postopek	Identifikacijska številka administriranega postopka

Tabela 4.1: Metrika postopka pri vstopu v sistem

Tabela 4.1 prikazuje podatke, ki se zajamejo ob nastanku postopka. Tu se ustvari unikatna označba "id_trans_postopek", s katero povežemo ostale podatke. Prav tako se zabeleži čas vstopa, za kakšen način poizvedovanja gre (sinhroni, asinhroni) in ali se zgradi BPM (Business process management) proces. BPM proces poskrbi, da neuspešna poizvedba pride v samoponovitveno vrsto in jo čez čas ponovno sproži na vir. Kot neuspešno poizvedbo štejemo tisto, ki se pošlje na neaktiven vir, ali če je prišlo do kakšne druge napake kot je na primer iztečeni čas branja (ang. Read Timeout), ki se proži, če vir ne vrne nobenega odgovora po 30 sekundah.

Ime	Opis
id_trans_postopek	Enolična številka transakcije
id_endpoint	Identifikacijska številka podatkovnega vira
start_time	Začetni čas procesnega stanja
end_time	Končni čas procesnega stanja
sporocilo	Napaka pri procesiranju postopka
endpoint_location	Naslov končne točke spletne storitve

Tabela 4.2: Metrika klica na vir

Tabela 4.2 prikazuje podatke o klicih na vir. V primeru da je prišlo do napake se v polje "sporocilo" zapiše sporočilo napake, v nasprotnem primeru pa ima polje vrednost "null", kar pomeni, da vir deluje pravilno. Prav tako zabeležimo začetni čas klica ter čas vrnitve odgovora in podatek o tem, na kateri vir je bila poizvedba poslana.

Ime	**Opis**
id_trans_postopek	Edinstvena identifikacija številka transakcije
process_id	Identifikacijska številka procesa
autoresume_retry	Število kolikokrat se je postopek nahajal v samo-ponovitvi
valid_error	Zaznamek posebne predvidene napake
start_time	Začetni čas procesnega stanja
end_time	Končni čas procesnega stanja
id_postopek_poizvedba	Identifikacijska številka postopka poizvedbe
id_poizvedba	Identifikacijska številka poizvedbe
sporocilo	Napaka pri procesiranju postopka
state	Procesno stanje
max_autoresume_retry	Največje dovoljeno število za posamezno poizvedbo, da konča v samo-ponovitvenem postopku. Konstanta, ki se jo lahko spreminja preko administracije za vsak vir posebej.
am	Zaznamek ali gre za asinhroni modul

Tabela 4.3: Metrika v procesnem stanju

Tabela 4.3 prikazuje procesne podatke o poizvedbah. Vrednost sporočila nosi informacijo o napaki poizvedbe. V primeru, da vir ne vrne napake to v metričnih podatkih zabeležimo kot novo stanje in ga poimenujemo "null". Prav tako tu zabeležimo kolikokrat je poizvedba prišla v samo-ponovitveno vrsto in ali gre poizvedba na asinhroni modul. Poizvedbe, katerih naloga je komunikacija z asinhronim modulom, imajo to posebnost, da na njihov

odgovor čakamo tudi do 3 delovne dni. Tovrstni informacijski sistemi (viri) ne dovoljujejo direktnega dostopa, zato imajo posrednika, ki ročno vrača odgovore.

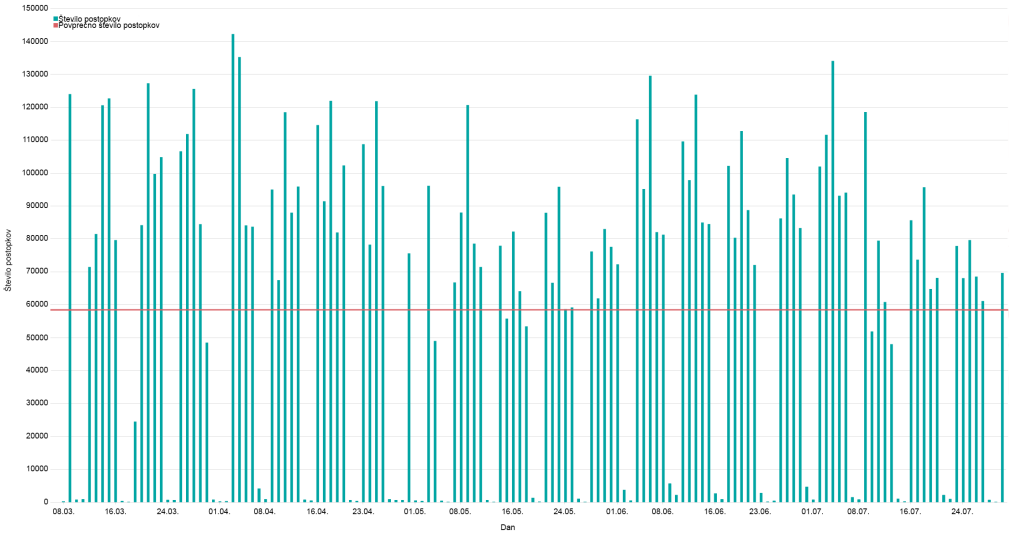
Ime	**Opis**
id_trans_postopek	Enolična številka transakcije in postopka
end_time	Končni čas procesnega stanja
sporocilo	Napaka pri procesiranju postopka
strat_tip	Način poizvedovanja za začetni postopek

Tabela 4.4: Metrika na sinhronih poizvedbah

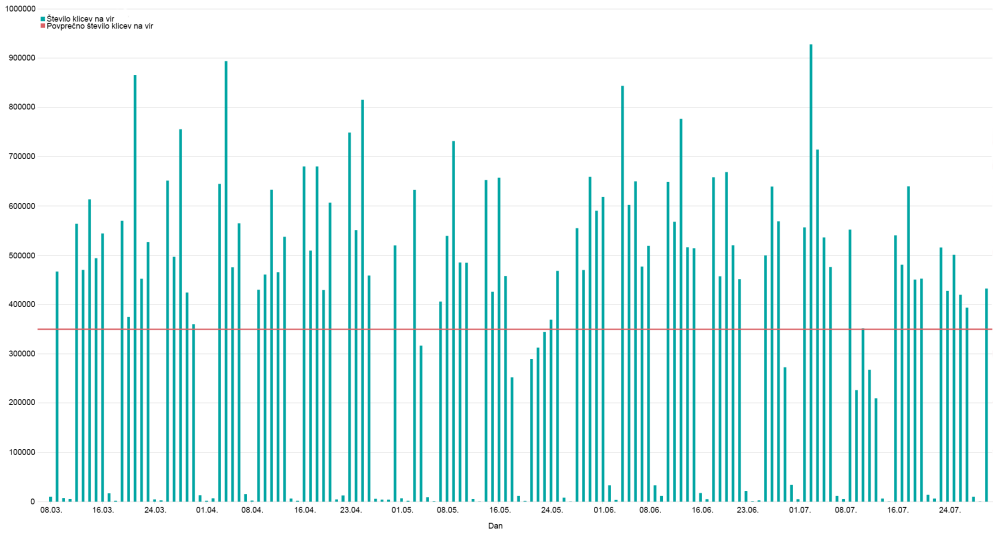
Tabela 4.4 ponazarja informacijo o uspešnem izvajanju sinhrone poizvedbe, saj te nimajo zapisa v procesni tabeli.

4.1 Predstavitev podatkov

Vse pridobljene podatke smo iz IS Pladenj prenesli v datoteke, te pa smo potem razčlenili in shranili v primerne tabele v lokalno bazo za kasnejšo obdelavo in predstavitev. Vsi podatki, ki so predstavljeni v tem delu poglavja, so bili zbrani med 8.3.2018 in 30.7.2018.

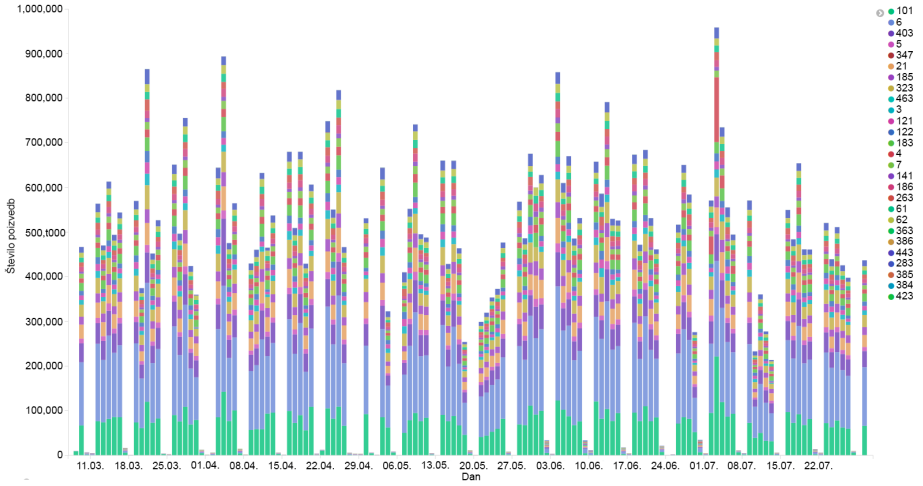


Slika 4.1: Število postopkov na dan

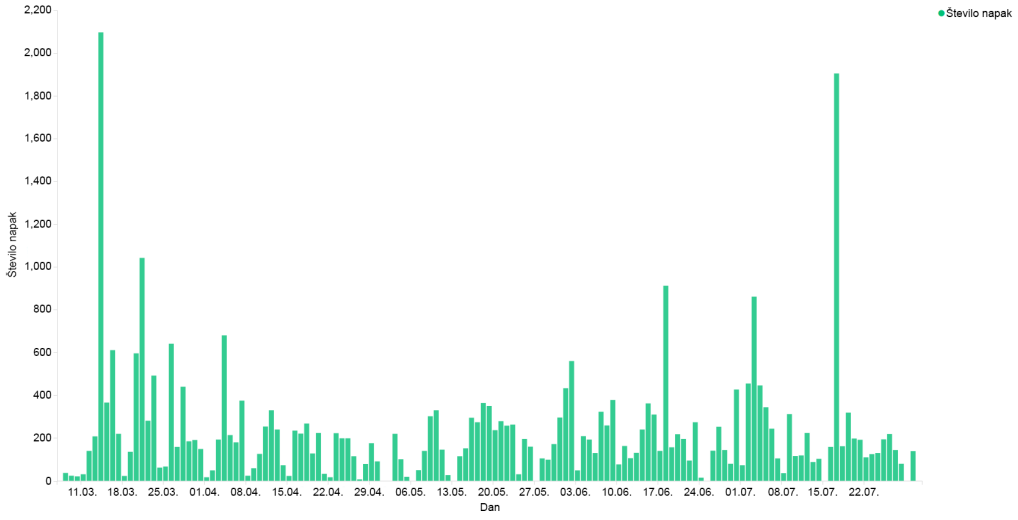


Slika 4.2: Število poizvedb na dan

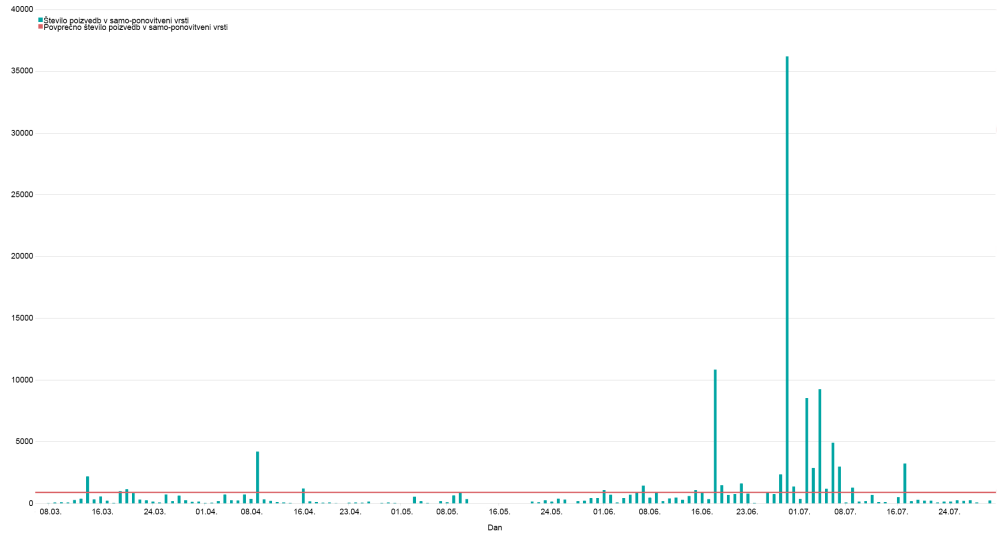
Grafa 4.1 in 4.2 sta konsistentna glede na to, da med vikendi in prazniki skoraj ni klicev. Opazi se tudi upad v klicih, in sicer v času od 10.7. do 13.7, saj je bilo takrat večje število odjemalcev na dopustu.



Slika 4.3: Porazdelitev poizvedb po viru (legenda na desni strani predstavlja ID vira, ki pripada posamezni barvi)

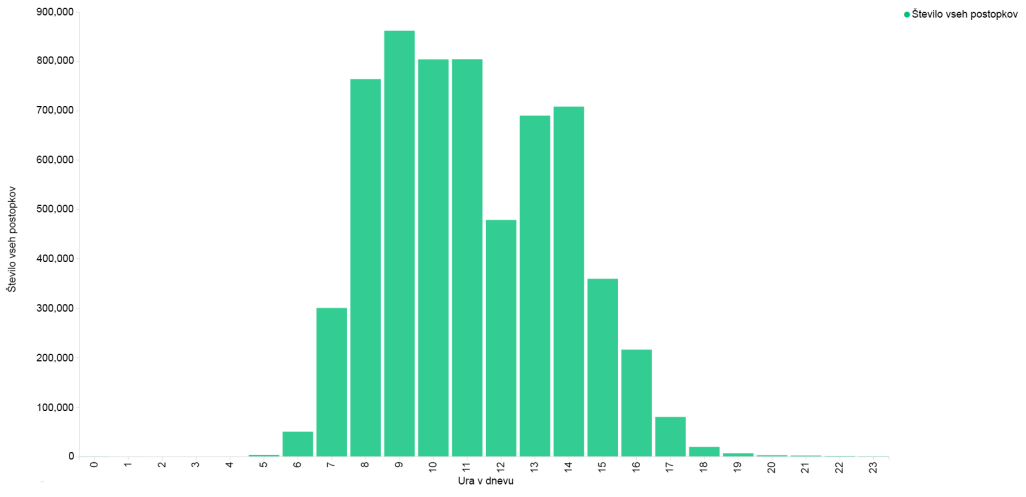


Slika 4.4: Skupno število napak na dan (kot napaka se šteje neodzivnost vira oziroma kakšen drug vzrok, kot je recimo HTTP koda 500, ki pomeni notranjo napako strežnika, ali iztečeni čas branja (ang. Read Timeout))



Slika 4.5: Število poizvedb, ki so končale v samo-ponovitveni vrsti

Velika količina poizvedb v samo-ponovitveni vrsti (špice v grafu na sliki 4.5) pomeni, da je bil en ali več virov nedosegljivih dalj časa. Zaradi tega so poizvedbe, ki jih je samo-ponovitvena vrsta ponovno sprožila, zopet prišle v samo-ponovitveno vrsto, notri pa so prišle tudi vse nove poizvedbe.



Slika 4.6: Število vseh postopkov, razdeljenih na uro v dnevu

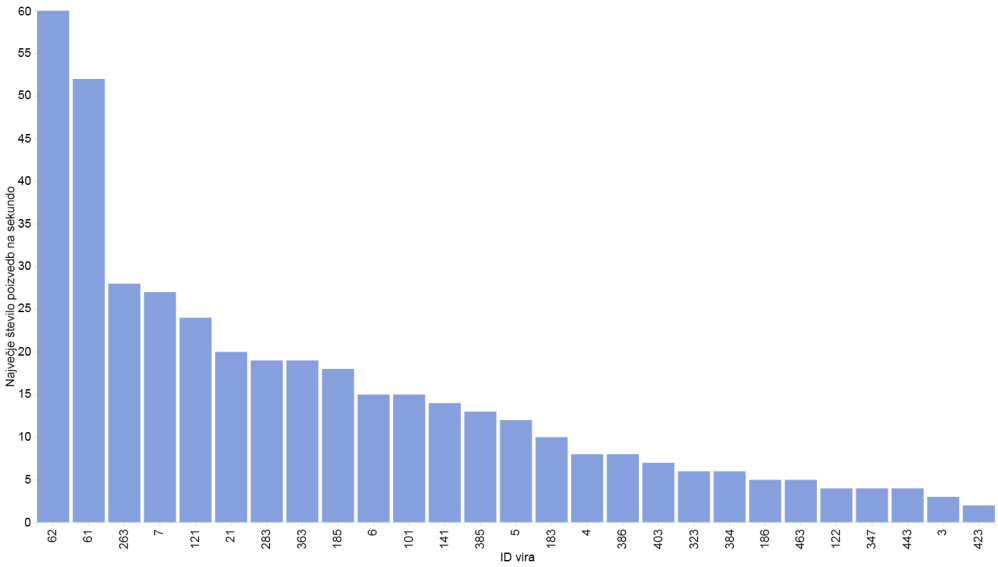
Iz grafov 4.1, 4.2 in 4.6 je razvidno, da se večina postopkov in klicev na

vir zgodi med delovnikom, natančneje med 8. in 17. uro, zato je v tem času obremenitev IS Pladenj največja in je večja verjetnost, da pride do preobremenitve vira. Tudi obremenjenost med viri je različna, na primer viri z identifikacijskimi številkami 6, 101 in 403 predstavljajo polovico vseh dnevnih poizvedb, kar je razvidno iz grafa 4.3. Na dan se proži povprečno 58500 postopkov in 360000 poizvedb, maksimum pa seže tudi do 142000 postopkov in 932000 poizvedb v enem dnevu. Podrobnejši statistični podatki o samo-ponovitveni vrsti so predstavljeni v tabeli 4.5. Spremembe v teh podatkih odražajo uspešnost projekta.

Ime	**Število poizvedb v samo-ponovitveni vrsti**
Minimum	1
Maksimum	36690
Povprečje	1248
Mediana	283
Standardni odklon	4297.65
1. kvartil	123
3. kvartil	735
Interkvartilni razpon	612

Tabela 4.5: Osnovni statistični podatki o samo-ponovitveni vrsti glede na dan

Da smo lahko določili diskretna območja za napovedovanje prepustnosti, smo iz podatkov za vsak vir posebej izluščili, koliko v povprečju porabi posamezen vir za obdelavo uspešne poizvedbe, kolikšen je najdaljši čas obdelave ene poizvedbe in koliko poizvedb lahko sprejme v eni sekundi. Predstavljeni so v tabeli 4.6, za vsak vir, za katerega imamo zbran podatek o vsaj eni poizvedi nanj.



Slika 4.7: Največje število poizvedb na posamezen vir v eni sekundi

ID vir	**Največji čas obdelave poizvedbe (v sekundah)**	**Povprečen čas obdelave poizvedbe (v sekundah)**	**Največje število poizvedb v sekundi (slika 4.7)**
62	6	0.009	60
61	11	0.016	52
263	21	0.02	28
7	24	0.03	27
121	19	0.024	24
21	21	0.018	20
283	24	0.021	19
363	7	0.02	19
185	19	0.021	18
101	16	0.023	15
6	24	0.002	15
141	24	0.02	14
385	15	0.054	13
5	15	0.064	12
183	28	0.027	10
386	22	0.032	8
4	21	0.03	8
403	22	0.045	7
384	11	0.067	6
323	28	0.045	6
463	17	0.036	5
186	29	0.015	5
347	12	0.087	4
443	6	1.184	4
122	24	0.586	4
3	29	2.222	3
423	28	2.424	2

Tabela 4.6: Obdelava poizvedb

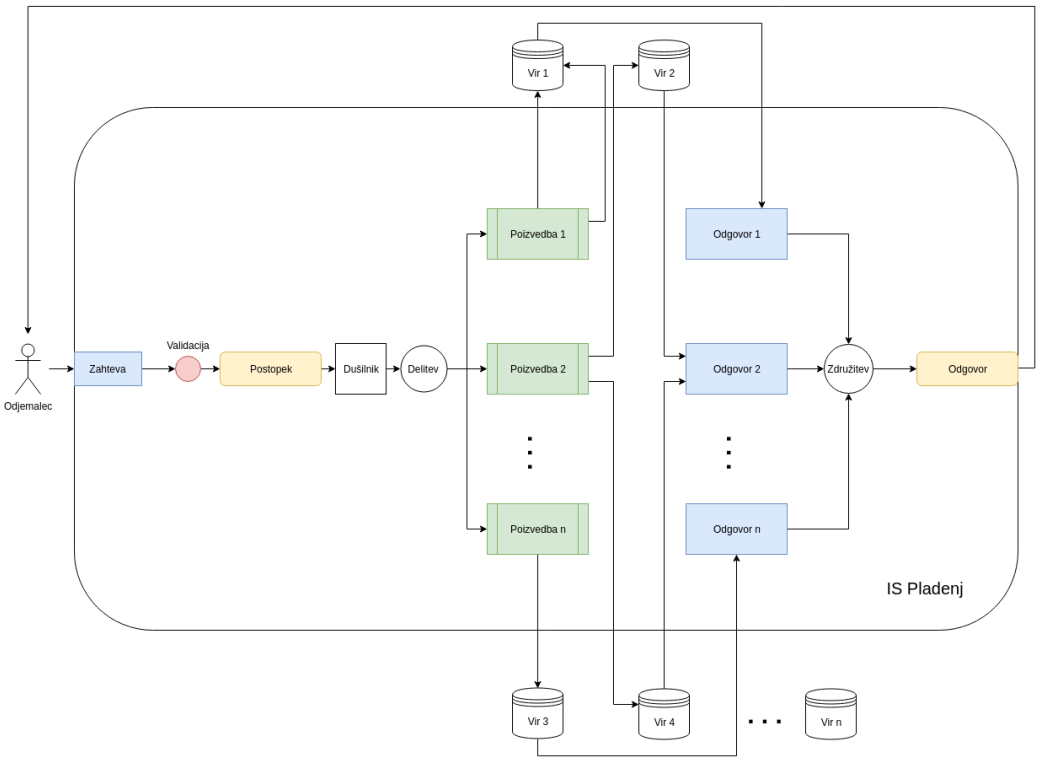
S pomočjo podatkov iz tabele 4.6 smo določili kolikšen je še sprejemljiv čas obdelave in število poizvedb na posameznem viru, preden jih zmanjšamo. Kot sprejemljivo število poizvedb na posamezen vir v eni sekundi smo vzeli 50% največjega števila poizvedb v sekundi. V primeru, da se število poizvedb dvigne nad to vrednost, zmanjšamo prepustnost postopkov, ki vsebujejo poizvedbe na ta vir. Če število postopkov v eni sekundi preseže 90% največjega števila, upoštevamo vir kot obremenjen in zmanjšamo prepustnost postopkov, ki vsebujejo poizvedbe na ta vir, na minimalno vrednost. Prav tako zmanjšamo prepustnost, kadar je čas obdelave daljši od povprečja. Tudi v tem primeru upoštevamo toleranco 10%, kar pomeni da mora biti čas obdelave nekoliko daljši od povprečja. Če čas obdelave preide nad 90% največje vrednosti, pa vir zopet štejemo kot preobremenjen.

Poglavje 5

Priprava podatkov

V tem poglavju bomo predstavili značilke, ki smo jih izpeljali iz zajetih podatkov. Uporabili jih bomo za učenje napovednih modelov.

Na podlagi natančnega pregleda delovanja in produkcije izvorne kode IS Pladenj smo najprej podrobno preučili celoten proces in robne pogoje pri pretoku postopkov, poizvedb in zahtevkov na oddaljene vire. Povezavo med njimi predstavlja slika 5.1. Odjemalec v sistem pošlje zahtevo, ki se ji preveri pravilnost vhodnih podatkov in dovoljenje za dostop na vire, nato pa se iz nje oblikuje postopek. Ta predstavlja skupek poizvedb. Vsak postopek je sestavljen iz ene ali več poizvedb, ki pridobivajo podatke iz virov. Pridobljeni podatki se pretvorijo v odgovore poizvedbe, ki se na koncu združijo v skupni odgovor postopka, ki ga vrnemo odjemalcu.



Slika 5.1: Povezava med zahtevkom, postopkom in poizvedbo

Z nadaljnjo analizo metričnih podatkov smo določili naslednje značilne atribute, značilke IS pladenj, ki opisujejo natančno stanje sistema in oddaljenih virov.

Ime	**Opis**
delovnik	Ali je delovni čas ali ne
st_novih_postopkov	Število postopkov
st_novih_poizvedb	Število poizvedb
st_novih_autoresume_poizvedb	Število poizvedb, ki so bile v samo-ponovitveni vrsti
st_sync_postopkov	Število sinhronih postopkov
st_async_no_bpm_postopkov	Število asinhronih postopkov brez BPM

Ime	**Opis**
st_async_no_am_postopkov	Število asinhronih postopkov brez asinhronega modula
st_async_am_postopkov	Število asinhronih postopkov z asinhronim modulom
med_uspesni-_no_bpm_postopkov	Srednja vrednost časa odzivov uspešno zaključenih postopkov brez BPM (čas v sekundah)
med_uspesni_bpm_postopki	Srednja vrednost časa odzivov uspešno zaključenih postopkov z BPM (čas v sekundah)
med_neuspesni-_autoresume_no_am_postopki	Srednja vrednost časa odzivov postopkov, ki so bili neuspešni v samo-ponovitveni vrsti brez asinhronega modula (čas v sekundah)
med_neuspesni-_autoresume_am_postopki	Srednja vrednost časa odzivov postopkov, ki so bili neuspešni v samo-ponovitveni vrsti z asinhronim modulom (čas v sekundah)
med_autoresume_postopki	Srednja vrednost časa odzivov postopkov, ki so bili del samo-ponovitvene vrste (čas v sekundah)
med_sync_error_postopki	Srednja vrednost časa odzivov postopkov, ki so pri sinhronem klicu vrnili napako (čas v sekundah)
st_uspesni_no_bpm	Število odzivov vseh uspešno zaključenih postopkov brez BPM
st_uspesni_bpm	Število odzivov vseh uspešno zaključenih postopkov z BPM

Ime	Opis
st_autoresume_error_no_am	Število odzivov vseh postopkov, ki so bili neuspešni v samo-ponovitveni vrsti brez asinhronega modula
st_autoresume_error_am	Število odzivov vseh postopkov, ki so bili neuspešni v samo-ponovitveni vrsti z asinhronim modulom
st_autoresume_no_am	Število odzivov vseh postopkov, ki so bili v samo-ponovitveni vrsti brez asinhronega modula
st_autoresume_am	Število odzivov vseh postopkov, ki so bili v samo-ponovitveni vrsti z asinhronim modulom
st_sync_error	Število odzivov sinhronega postopka, ki je imel napako
st_start_record_error	Število postopkov, ki niso imeli zapisa v procesni matriki ali matriki, ki spremlja vire
hitrost_uspesni-_no_bpm_postopki	Hitrost uspešno zaključenih postopkov brez BPM (število postopkov na sekundo)
hitrost_uspesni_bpm_postopki	Hitrost uspešno zaključenih postopkov z BPM (število postopkov na sekundo)
hitrost_neuspesni-_autoresume_no_am_postopki	Hitrost vseh postopkov, ki so bili neuspešni v samo-ponovitveni vrsti brez asinhronega modula (število postopkov na sekundo)
hitrost_neuspesni-_autoresume_am_postopki	Hitrost vseh postopkov, ki so bili neuspešni v samo-ponovitveni vrsti z asinhronim modulom (število postopkov na sekundo)

Ime	**Opis**
hitrost_autoresume_postopki	Hitrost vseh postopkov, ki so bili v samoponovitveni vrsti (število postopkov na sekundo)
hitrost_sync_error_postopki	Hitrost sinhronega postopka, ki je imel napako (število postopkov v eni sekundi)
hitrost_uspesni_endpoint	Hitrost praznjenja uspešno zaključenih poizvedb v čakalni vrsti za vir (število poizvedb na sekundo)
hitrost_neuspesni_endpoint	Hitrost praznjenja neuspešno zaključenih poizvedb v čakalni vrsti za vir (število poizvedb na sekundo)
hitrost_autoresume_endpoint	Hitrost praznjenja poizvedb iz samoponovitvene vrste za vir (število poizvedb na sekundo)
hitrost_err_endpoint	Hitrost detekcije napake od vstopa postopka do poizvedbe na viru, za vir posebej (število poizvedb na sekundo)
st_uspesnih_na_endpoint	Število uspešnih poizvedb na viru
st_neuspesnih_na_endpoint	Število neuspešnih poizvedb na viru

Tabela 5.1: Značilke

Vse značilke so bile izračunane na podlagi 5 minutnega okna in za vsak zabeležen vir posebej. Ker nismo želeli računati oken za vsako sekundo, smo kot izhodišče za računanje oken vzeli vse edinstvene "start_time" vrednosti iz tabele metrike klica na vir. S tem smo se izognili računanju istih značilk oziroma takih, ki bi imele same vrednosti 0, kar bi se zgodilo predvsem v

nočnem času. Nato smo vzeli vse metrične podatke za 5 minut nazaj od izhodišča in iz njih izračunali značilko. Postopek smo ponavljali dokler nismo izračunali vseh značilk. 5 minutno okno smo določili na podlagi zmogljivosti hranjena metričnih podatkov v spominu IS Pladenj, da ga pri tem dodatno ne oviramo pri delovanju. Želeli smo zajeti kar se da veliko število informacij o sistemu, zato okna nismo zmanjšali na manjši interval.

Značilke smo izpeljali tako, da smiselno povzamejo podatke iz metrike. Tako smo povzeli podatke kot so število postopkov in poizvedb, ki se nahajajo v sistemu, podatek o tem ali smo v delovnem času, števila napak, koliko je poizvedb v samo-ponovitveni vrsti, hitrost praznjena vrste in odziva vira ter druge, predstavljene v tabeli 5.1. Imeli smo tudi značilko "start_time", ki je beležila čas izračuna značilke, vendar pa ta ni imela večjega pomena, saj je imela za vsako značilko drugo vrednost, ki je naraščala in ni o stanju IS Pladenj povedala ničesar. Pretvorili smo jo v značilko "delovnik".

Poglavje 6

Modeliranje

V naslednjih odstavkih bomo podrobneje predstavili zasnovo dušilnika in učenje napovednih modelov.

V sistem želimo vpeljati dušilnik, ki bo skrbel za optimalno obremenitev virov in s tem skrajšal povprečni čas čakanja odgovorov postopkov. V smislu odjemalca lahko izvedeno optimizacijo opišemo kot hitrejše delovanje IS Pladenj. Želimo si, da bi se parametri dušnilnika dovolj hitro prilagajali virom, da do preobremenjenosti vira in morebitne odpovedi ne bi nikoli prišlo. Z implementacijo dušilnika si torej želimo:

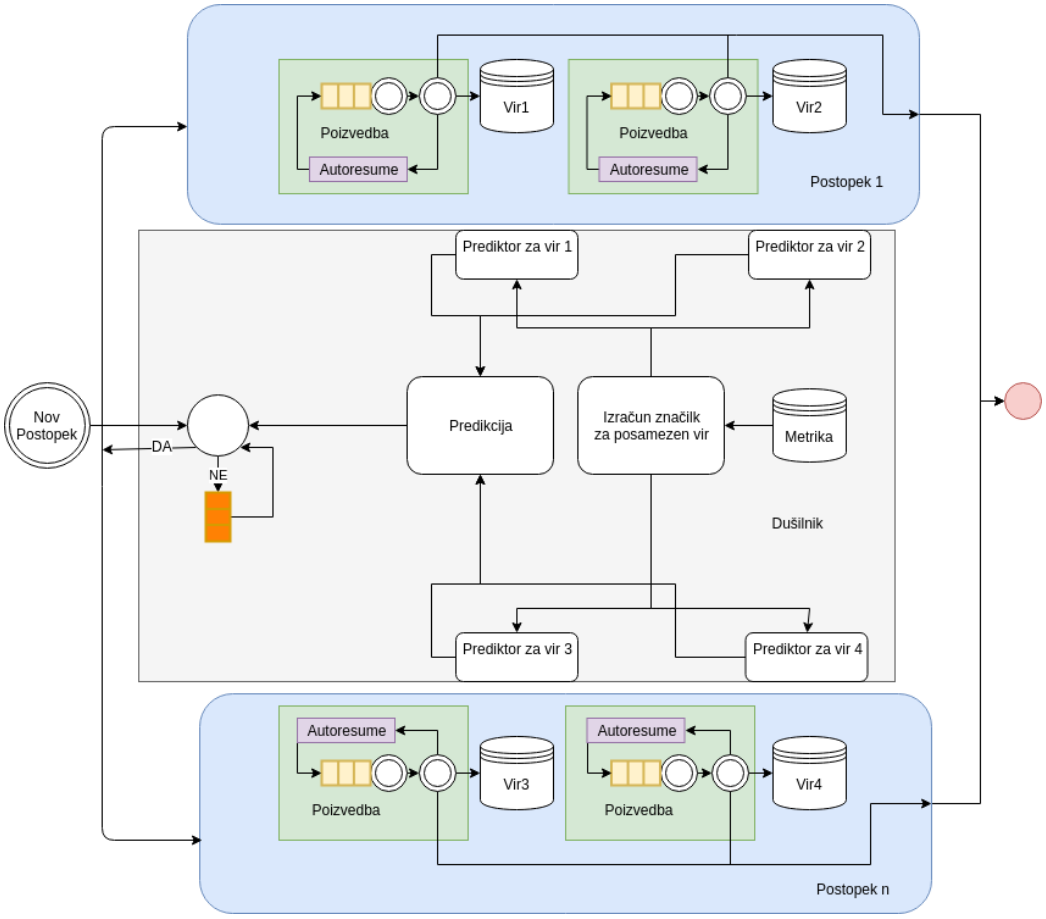
- zmanjšati povprečen čas čakanja na odgovor,
- zmanjšati število poizvedb, ki čakajo v samo-ponovitveni vrsti,
- preprečiti odpoved vira zaradi sočasnega ali zaporednega previsokega števila klicev virov s strani IS Pladenj.

Dušimo lahko na dva načina. Prvi način je, da dušimo postopke preden ti pridejo v sistem in se iz njih generirajo poizvedbe, drugi pa je dušenje poizvedb preden se izvedejo na vsakem viru posebej. Dušilnik bi se na podlagi trenutnega stanja IS Pladenj odločil, ali neki postopek oziroma poizvedbo spusti naprej, ali pa z njim počaka. Trenutno stanje sistema se opiše s pomočjo metrike, ki je opisana v poglavju 4.

Dušilnik na osnovi dinamičnega parametra prepustnosti prepušča postopke v sistem. Ta parameter se določi glede na stanje virov. Stanje vira lahko opišemo kot tristopenjski semafor, kar pomeni, da delovanje vira predstavimo s tremi stanji. Stanje virov lahko določimo na podlagi statistike ali s pomočjo algoritmov strojnega učenja. Vsako stanje ima vnaprej določeno prepustnost, ki smo jo določili na podlagi statistične analize metričnih podatkov. Dokler vir deluje optimalno ("zelena" na semaforju) ima prepustnost, ki je enaka 90% največjega števila poizvedb na sekundo (tabela 4.6). Če mu zmogljivost upade ("rumena" na semaforju), mu zmanjšamo prepustnost na 50% največjega števila poizvedb na sekundo. V primeru, da je vir nedosegljiv oziroma preobremenjen ("rdeča" na semaforju), zmanjšamo prepustnost na eno poizvedbo na minuto.

Zasnovali bomo dušilnik, ki duši postopke preden se razdelijo na posamezne poizvedbe, ker v tem primeru potrebujemo zgolj eno čakalno vrsto, ki jo je lažje implementirati. Dušiti si želimo zgolj postopke, katerih poizvedbe bi preobremenile vir. Prediktor [9] lahko za vsak postopek posebej napove ali gre lahko naprej v sistem, kjer se bo razdelil na poizvedbe, ali pa vsake toliko časa določi stanje za posamezen vir. Dušilnik se nato na podlagi števila poizvedb na virih, njihovega stanja in zmožnosti odloči, ali spusti postopek ali ne.

V obeh primerih zasnove hranimo v sistemu vse podatke za 5 minut nazaj. Vsako sekundo se iz nje izračuna stanje posameznih virov in stanje IS Pladenj kot celote.



Slika 6.1: Delovanje IS Pladenj z dušilnikom

Slika 6.1 prikazuje delovanje IS Pladnja z dušilnikom, ki prepustnost določi s pomočjo napovednih modelov. Komponenta "predikcija", ki se nahaja na sredini slike, vsako sekundo prejme od posameznega napovednega modela stanje vira glede na njegovo obremenjenost. Na podlagi teh stanj se odločimo, ali neki postopek spustimo v nadaljnjo obdelavo, ali ga zadržimo. Podobno izgleda tudi dušilnik na podlagi statistične analize. Od dušilnika z napovednimi modeli se razlikuje zgolj v tem, da napovednih modelov za posamezen vir nima.

6.1 Zasnova dušilnika na podlagi statistične analize

Na podlagi časa izvajanja in števila uspešnih ter neuspešnih poizvedb v obliki semaforja določimo delovanje virov. To pomeni, da bomo kot odgovor vrnili informacijo o tem, ali vir trenutno deluje optimalno, ali se je njegova odzivnost zmanjšala, ali je vir neodziven oziroma preobremenjen. Vsakemu viru smo na podlagi podatkov, ki smo jih zbrali, določili meje njegovega delovanja, ki smo jih podrobneje opisali v razdelku 4.1. Pomembno vlogo pri določanju stanja vira pri dušilniku na podlagi statistične analize imata tudi zadnji dve poizvedbi, ki sta prišli na ta vir v 5 minutnem časovnem oknu. V primeru, da sta obe poizvedbi uspešni, se stanje določi glede na čas obdelave in število poizvedb v eni sekundi na tem viru. Kadar sta obe neuspešni, vir označimo kot nedelujoč, če pa je ena poizvedba uspešna in druga neuspešna, vir upoštevamo kot delno delujoč. V primeru, da v 5 minutnem oknu za določen vir ni nobene poizvedbe ali pa je samo ena, je stanje vira enako optimalnemu. Na podlagi celotnega stanja IS Pladenj in glede na to, katere vire bo klical postopek, ter v kakšnem stanju so, se odločimo ali postopke spustimo naprej, ali počakamo.

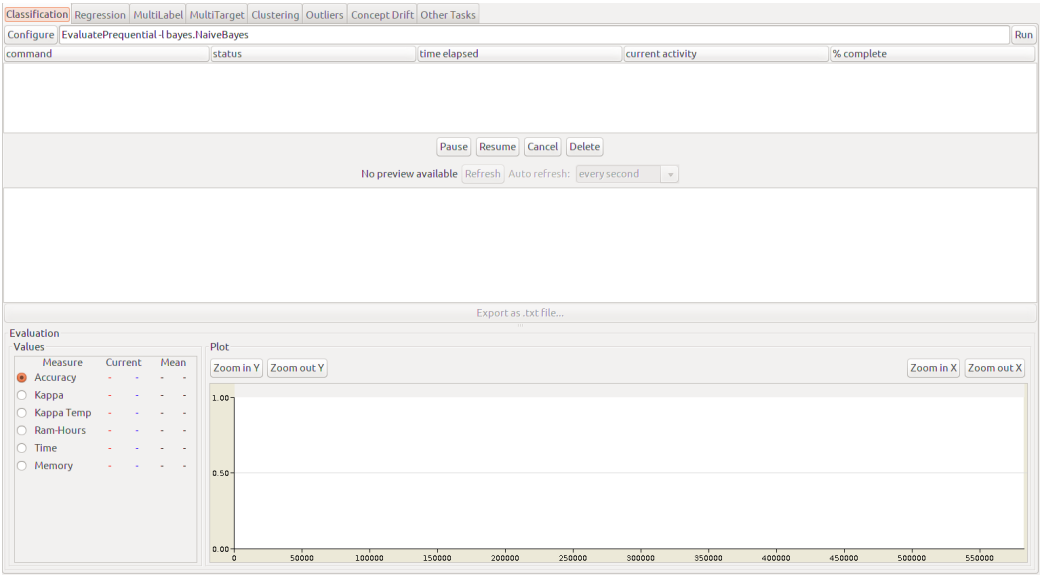
6.2 Zasnova dušilnika z algoritmi strojnega učenja

Podobno, kot deluje dušilnik na podlagi statističnega odločanja, deluje tudi dušilnik z napovednimi modeli. Bistvena razlika v primerjavi z zasnovo s statistiko je, da tu za stanje virov uporabljamo napovedne modele, ki ravno tako vračajo odgovore v obliki semaforja. Razlikuje se tudi v tem, da tu potrebujemo še dodatno zunanjo virtualko, ki skrbi za dodatno učenje napovednih modelov. Sem spada potek pridobivanja novih podatkov za potrebe učenja in nadgrajevanje napovednih modelov, kar bo podrobneje predstavljeno v razdelku 7.2. Napovedni modeli za posamezen vir napovejo s kakšno

optimalnostjo delujejo. Vsako sekundo povprašamo modele po delovanju virov in se, glede na to, katere vire bo klical postopek, v kakšnem stanju so in koliko poizvedb še lahko sprejmejo, odločimo ali postopek spustimo v sistem ali pa ga pošljemo v čakalno vrsto.

6.3 Učenje modela

Za učenje napovednih modelov smo uporabili MOA [8]. MOA je odprtokodno ogrodje za rudarjenje podatkovnih tokov. Vsebuje kar nekaj algoritmov za strojno učenje, kot sta Naive Bayes [10] in Hoeffding drevo [11]. Poleg klasifikacije omogoča tudi regresijo in algoritme za določanje rojev ter detekcijo izjem. Celotno ogrodje je napisano v Javi. Interakcijo lahko delamo na več načinov. Prvi način uporablja MOA API direktno v kodi, drugi deluje preko terminala in tretji z uporabo grafičnega vmesnika. Napovedne modele smo naučili s pomočjo terminala, samo napoved pa smo izvedli s pomočjo API-ja.



Slika 6.2: MOA grafični vmesnik

Eden izmed razlogov, zakaj smo se odločili za MOA je, da je napisana v Javi, kar nam zelo poenostavi integracijo v IS Pladenj. Drugi razlog je ta, da napovednih modelov ne želimo vedno znova učiti od začetka vsakič, ko zberemo nove podatke, ampak jih želimo zgolj posodobiti.
Za učenje napovednih modelov smo uporabili inkrementalne algoritme, saj jih želimo posodabljati. Model bo deloval v obliki semaforja. Napovedal bo enega izmed 3 razredov in sicer 0, če vir deluje brezhibno, 1, če se mu je odzivnost zmanjšala, in 2, če je vir preobremenjen oziroma neaktiven. Slabši primer je, če napovedni model vrne nižji razred za stanje vira, kot je njegovo dejansko stanje (namesto 1 vrne 0), medtem ko je primer boljši, ko vrne višji razred, kot je dejansko stanje (namesto 0 vrne 1). Slednji je boljši zato, ker le upočasni pretok postopkov (ki se bo sčasoma dvignil), medtem ko lahko prvi privede do preobremenitve vira.

	Razred 0	**Razred 1**	**Razred 2**
Razred 0	0	1	1
Razred 1	2	0	1
Razred 2	2	2	0

Tabela 6.1: Tabela cen napačne klasifikacije. Vrstice predstavljajo dejanski razred, stolpci pa napovedan razred

Ocene, prikazane v tabeli 6.1, so bile pridobljene na podlagi mnenja strokovnjaka. Pridobljenega ekspertnega znanja iz tabele 6.1 zaenkrat še nismo uporabili pri strojnem učenju in bo uporabljeno pri nadaljnjem delu.

ID Vira	**Razred 0 (v %)**	**Razred 1 (v %)**	**Razred 2 (v %)**	**Manjkajoči razredi (v %)**
385	39.51	28.45	32.03	0
403	87.44	0.06	12.50	0
122	56.75	39.68	3.57	0
21	75.91	13.75	10.33	0
347	98.47	0.11	1.42	0
263	87.41	4.79	7.81	0
5	74.90	23.26	1.85	0
3	96.86	0.16	2.98	0
186	91.59	4.37	4.04	0
7	52.64	42.81	4.54	0
384	98.01	0.00	1.99	0
185	53.20	46.50	0.30	0
141	99.82	0.03	0.15	0
423	100.00	0.00	0.00	0
121	79.32	1.14	19.54	0
363	73.62	18.57	7.81	0
4	93.14	3.17	3.69	0
61	48.80	45.01	6.19	0
6	99.24	0.32	0.44	0
443	94.38	0.35	5.26	0
463	51.41	18.93	29.66	0
101	53.65	38.85	7.50	0
386	86.63	7.29	6.07	0
62	87.72	0.00	12.28	0
323	97.49	0.44	2.08	0
183	82.91	8.45	8.64	0
283	96.20	0.15	3.65	0

Tabela 6.2: Tabela porazdelitve razredov in manjkajočih razredov za posamezen vir

Poglavje 7

Ovrednotenje in uporaba rezultatov

V tem poglavju bomo predstavili primerjavo med tremi algoritmi strojnega učenja ter primerjavo med delovanjem dušilnika, ki deluje na podlagi statistične analize, in dušilnika z napovednimi modeli. Opisali bomo tudi arhitekturo prenosa izračunanih modelov v IS pladenj in njihovo avtomatično nadgrajevanje.

7.1 Ovrednotenje

Na podlagi metode Test-Then-Train, ki je prav tako že implementirana v MOA, smo testirali pravilnost klasifikacije. Ta metoda deluje tako, da za vsako značilko najprej napove razred, nato se preveri pravilnost napovedi, zatem pa se model na njej nauči. Pred testiranjem pravilnosti klasifikacije smo vsak model naučili na 1000 primerih zato, da v začetku nimamo prevelike napake. Točnost klasifikacije smo testirali na algoritmih Naive Bayes, Hoeffding Tree in Adaptive Random Forest.

Iz tabele 6.2 lahko razberemo, da prevladuje razred 0, ki predstavlja optimalno delovanje vira. Razlog za tako neenakomerno porazdelitev je premalo podatkov, kjer so viri neodzivni oziroma delujejo počasneje. Zaradi tega tudi

algoritmi, ki smo jih testirali, vračajo tako visoko klasifikacijsko točnost.

V tabelah 7.1–7.9 so prikazani podatki za preciznost, priklic ter F1 mero za posamezen razred in posamezen algoritem. "?" v teh tabelah pomeni, da za vir nismo imeli nobenega primera iz danega razreda, zato tam ne moremo izračunati preciznosti, priklica in F1 mere. Prav tako so za nekatere razrede vrednosti veliko manjše od ostalih, kar je posledica majhnega števila primerov s tem razredom na tem viru. Na podlagi podatkov, prikazanih v tabelah 7.10, 7.11, 7.12, smo se odločili, da bomo uporabili Hoeffding tree. Izbrali smo ga kljub temu, da je imel Adaptive Random Forest malenkost boljše rezultate, vendar pa sta bila pri testiranju oba podobno uspešna, zato smo izbrali manj kompleksen algoritem.

ID vira	**Preciznost za razred 0**	**Preciznost za razred 1**	**Preciznost za razred 2**
385	89.68	71.56	89.09
403	97.81	10.46	58.17
122	66.38	83.73	32.74
21	84.44	88.45	43.04
347	99.16	29.35	14.04
263	91.66	16.23	88.72
5	96.63	93.24	11.16
3	99.23	35.81	39.14
186	98.84	4.48	14.15
7	56.17	78.77	29.12
384	99.87	?	42.01
185	94.97	62.55	9.49
141	99.99	12.92	4.69
121	99.72	2.1	98.85
363	78.11	51.7	44.62
4	99.33	5.05	6.22
61	81.24	46.55	66.67
6	99.61	17.9	15.13
443	99.55	15.87	100
463	92.86	87.72	45.78
101	57.24	48.83	46.9
386	93.05	9.92	8.77
62	96.68	?	53.64
323	99.71	32.64	51.47
183	99.37	59.52	48.4
283	98.41	71.54	76.74

Tabela 7.1: Tabela preciznosti algoritma Naive Bayes za posamezen razred

ID vira	**Preciznost za razred 0**	**Preciznost za razred 1**	**Preciznost za razred 2**
385	99.95	99.94	99.99
403	99.99	91.27	99.92
122	99.99	99.99	99.95
21	99.96	99.95	99.79
347	99.98	83.04	99.61
263	99.99	99.75	99.86
5	99.99	99.97	99.98
3	99.99	99.13	99.95
186	99.99	99.93	99.96
7	99.99	99.99	99.95
384	99.99	?	99.97
185	99.99	99.99	99.72
141	99.99	98.51	99.61
121	99.99	99.89	99.98
363	99.97	99.94	99.97
4	99.99	99.94	99.95
61	99.99	99.99	99.98
6	99.99	98.81	99.4
443	99.76	55.08	99.76
463	99.96	99.91	99.98
101	99.99	99.96	99.96
386	99.99	99.94	99.96
62	99.99	?	99.99
323	99.99	99.63	99.82
183	99.99	99.98	99.99
283	99.99	99.71	99.94

Tabela 7.2: Tabela preciznosti algoritma Hoeffding tree za posamezen razred

ID vira	**Preciznost za razred 0**	**Preciznost za razred 1**	**Preciznost za razred 2**
385	99.94	99.85	99.91
403	99.97	96.48	99.89
122	99.88	99.94	99.55
21	99.98	99.97	99.96
347	99.99	99.21	99.87
263	99.99	99.81	99.95
5	99.98	99.94	99.77
3	99.99	99.83	99.89
186	99.98	99.55	99.79
7	99.97	99.99	99.93
384	99.99	?	99.96
185	99.99	99.98	99.54
141	99.99	99.71	99.82
121	99.99	99.52	99.97
363	99.94	99.94	99.83
4	99.99	99.69	99.85
61	99.99	99.97	99.91
6	99.99	99.92	99.85
443	99.95	89.69	99.18
463	99.91	99.93	99.92
101	99.99	99.97	99.93
386	99.98	99.82	99.89
62	99.99	?	99.99
323	99.99	99.88	99.94
183	99.99	99.85	99.94
283	99.99	99.8	99.94

Tabela 7.3: Tabela preciznosti algoritma Adaptive Random Forest za posamezen razred

ID vira	**Priklic za razred 0**	**Priklic za razred 1**	**Priklic za razred 2**
385	94.25	78.51	75.81
403	90.79	93.04	85.36
122	93.06	35.75	32.01
21	82.53	19.44	94.88
347	95.89	100	43.66
263	97.23	13.99	35.84
5	95.85	95.35	11.63
3	96.18	97.55	74.59
186	8.32	74.21	69.86
7	88.27	29.1	85.14
384	97.37	?	93.96
185	48.31	94.23	91.77
141	96.9	99.46	95.88
121	41.26	90.31	92.09
363	86.96	12.29	77.94
4	7.75	87.19	64.27
61	5.04	96.29	41.6
6	97.92	18.06	73.74
443	98.95	100	92.19
463	94.86	34.61	93.34
101	96.4	3.93	40.74
386	13.04	37.66	86.82
62	90.6	?	77.77
323	97.59	94.26	82.34
183	90.92	73.17	77.08
283	99.27	99.94	57.76

Tabela 7.4: Tabela priklica algoritma Naive Bayes za posamezen razred

ID vira	**Priklic za razred 0**	**Priklic za razred 1**	**Priklic za razred 2**
385	99.98	99.92	99.96
403	99.98	95.51	99.92
122	99.99	99.99	99.94
21	99.96	99.98	99.69
347	99.97	93.2	99.05
263	99.99	99.7	99.85
5	99.99	99.98	99.93
3	99.99	99.25	99.91
186	99.99	99.87	99.96
7	99.99	99.99	99.99
384	99.99	?	99.84
185	99.99	99.99	99.55
141	99.99	98.4	98.02
121	99.99	99.62	99.99
363	99.98	99.92	99.9
4	99.99	99.94	99.94
61	99.99	99.99	99.99
6	99.99	99.41	99.72
443	99.85	98.92	95.51
463	99.97	99.99	99.92
101	99.99	99.98	99.81
386	99.99	99.89	99.91
62	99.99	?	99.99
323	99.99	98.85	99.81
183	99.99	99.97	99.99
283	99.99	99.27	99.95

Tabela 7.5: Tabela priklica algoritma Hoeffding tree za posamezen razred

ID vira	**Priklic za razred 0**	**Priklic za razred 1**	**Priklic za razred 2**
385	99.91	99.87	99.93
403	99.99	73.86	99.87
122	99.93	99.9	99.27
21	99.99	99.93	99.94
347	99.99	98.26	99.69
263	99.99	99.84	99.94
5	99.98	99.96	99.71
3	99.99	99.71	99.87
186	99.98	99.61	99.72
7	99.98	99.98	99.88
384	99.99	?	99.92
185	99.99	99.99	98.6
141	99.99	99.82	99.55
121	99.99	99.35	99.98
363	99.97	99.86	99.78
4	99.99	99.71	99.86
61	99.98	99.98	99.89
6	99.99	99.46	99.52
443	99.93	93.54	99.47
463	99.94	99.84	99.93
101	99.98	99.97	99.93
386	99.98	99.81	99.86
62	99.99	?	99.99
323	99.99	99.9	99.92
183	99.99	99.9	99.93
283	99.99	99.87	99.94

Tabela 7.6: Tabela priklica algoritma Adaptive Random Forest za posamezen razred

ID vira	**F1 mera razreda 0**	**F1 mera razreda 1**	**F1 mera razreda 2**
385	91.91	74.87	81.92
403	94.17	18.8	69.19
122	77.49	50.11	32.37
21	83.48	31.87	59.22
347	97.5	45.38	21.25
263	94.36	15.03	51.06
5	96.24	94.28	11.39
3	97.68	52.39	51.34
186	15.36	8.46	23.54
7	68.65	42.5	43.4
384	98.61	?	58.06
185	64.04	75.19	17.2
141	98.42	22.88	8.95
121	58.37	4.11	95.35
363	82.3	19.86	56.75
4	14.38	9.55	11.34
61	9.49	62.76	51.23
6	98.76	17.98	25.11
443	99.25	27.39	95.93
463	93.85	49.63	61.43
101	71.83	7.27	43.6
386	22.88	15.7	15.92
62	93.54	?	63.49
323	98.64	48.49	63.35
183	94.95	65.64	59.46
283	98.84	83.39	65.91

Tabela 7.7: Tabela F1 mere algoritma Naive Bayes za posamezen razred

ID vira	**F1 mera razreda 0**	**F1 mera razreda 1**	**F1 mera razreda 2**
385	99.96	99.93	99.97
403	99.98	93.34	99.92
122	99.99	99.99	99.95
21	99.96	99.96	99.74
347	99.98	87.83	99.33
263	99.99	99.72	99.86
5	99.99	99.98	99.96
3	99.99	99.19	99.93
186	99.99	99.9	99.96
7	99.99	99.99	99.97
384	99.99	?	99.9
185	99.99	99.99	99.64
141	99.99	98.46	98.81
121	99.99	99.75	99.98
363	99.98	99.93	99.94
4	99.99	99.94	99.94
61	99.99	99.99	99.98
6	99.99	99.1	99.56
443	99.81	70.76	97.59
463	99.97	99.95	99.95
101	99.99	99.97	99.89
386	99.99	99.92	99.93
62	99.99	?	99.99
323	99.99	99.24	99.82
183	99.99	99.97	99.99
283	99.99	99.49	99.95

Tabela 7.8: Tabela F1 mere algoritma Hoeffding tree za posamezen razred

ID vira	**F1 mera razreda 0**	**F1 mera razreda 1**	**F1 mera razreda 2**
385	99.93	99.86	99.92
403	99.98	83.67	99.88
122	99.91	99.92	99.41
21	99.98	99.95	99.95
347	99.99	98.73	99.78
263	99.99	99.83	99.94
5	99.98	99.95	99.74
3	99.99	99.77	99.88
186	99.98	99.58	99.75
7	99.98	99.98	99.9
384	99.99	?	99.94
185	99.99	99.98	99.07
141	99.99	99.77	99.68
121	99.99	99.44	99.97
363	99.95	99.9	99.81
4	99.99	99.7	99.86
61	99.98	99.97	99.9
6	99.99	99.69	99.68
443	99.94	91.57	99.32
463	99.92	99.89	99.92
101	99.98	99.97	99.93
386	99.98	99.82	99.87
62	99.99	?	99.99
323	99.99	99.89	99.93
183	99.99	99.87	99.94
283	99.99	99.84	99.94

Tabela 7.9: Tabela F1 mere algoritma Adaptive Random Forest za posamezen razred

ID vira	Večinski razred (v %)	Točnost	Preci-znost	Priklic	F1 mera	Število prime-rov
385	39.51	83.87	83.44	82.86	82.9	4424181
403	87.44	90.11	55.48	89.73	60.72	10745333
122	56.75	68.14	60.95	53.61	53.32	10527924
21	75.91	75.13	71.98	65.62	58.19	10702887
347	98.47	95.15	47.52	79.85	54.71	2080100
263	87.41	88.45	65.54	49.02	53.48	10545566
5	74.9	94.18	67.01	67.61	67.3	10726501
3	96.86	95.54	58.06	89.44	67.14	10582379
186	91.59	13.69	39.16	50.8	15.78	10689972
7	52.64	56.98	54.69	67.51	51.52	10690698
384	98.01	97.3	70.94	95.66	78.33	4605950
185	53.2	69.79	55.67	78.1	52.14	10737380
141	99.82	96.9	39.2	97.41	43.41	10618120
121	79.32	51.75	66.89	74.55	52.61	10545806
363	73.62	72.39	58.14	59.07	52.97	4779106
4	93.14	12.35	36.87	53.07	11.76	10703393
61	48.8	48.37	64.82	47.64	41.16	10694456
6	99.24	97.55	44.21	63.24	47.28	10816242
443	94.38	98.59	71.81	97.05	74.19	50899
463	51.41	63.4	75.45	74.27	68.3	1074404
101	53.65	56.3	50.99	47.02	40.9	11067939
386	86.63	19.32	37.25	45.84	18.17	7782335
62	87.72	89.03	75.16	84.18	78.51	10689597
323	97.49	97.26	61.27	91.4	70.16	10792250
183	82.91	88.22	69.1	80.39	73.35	10729038
283	96.2	97.76	82.23	85.66	82.71	10545503

Tabela 7.10: Tabela uspešnosti algoritma Naive Bayes

ID vira	Večinski razred (v %)	Točnost	Preciznost	Priklic	F1 mera	Število primerov
385	39.51	99.96	99.96	99.96	99.96	4424181
403	87.44	99.97	97.06	98.47	97.75	10745333
122	56.75	99.99	99.98	99.98	99.98	10527924
21	75.91	99.94	99.9	99.88	99.89	10702887
347	98.47	99.95	94.21	97.41	95.71	2080100
263	87.41	99.97	99.87	99.85	99.86	10545566
5	74.9	99.98	99.98	99.97	99.97	10726501
3	96.86	99.99	99.69	99.72	99.7	10582379
186	91.59	99.99	99.96	99.94	99.95	10689972
7	52.64	99.99	99.98	99.99	99.99	10690698
384	98.01	99.99	99.98	99.92	99.95	4605950
185	53.2	99.99	99.9	99.84	99.87	10737380
141	99.82	99.99	99.37	98.81	99.09	10618120
121	79.32	99.99	99.95	99.87	99.91	10545806
363	73.62	99.96	99.96	99.94	99.95	4779106
4	93.14	99.99	99.96	99.96	99.96	10703393
61	48.8	99.99	99.99	99.99	99.99	10694456
6	99.24	99.99	99.4	99.7	99.55	10816242
443	94.38	99.61	84.87	98.09	89.39	50899
463	51.41	99.96	99.95	99.96	99.96	1074404
101	53.65	99.97	99.97	99.93	99.95	11067939
386	86.63	99.98	99.96	99.93	99.95	7782335
62	87.72	99.99	99.99	99.99	99.99	10689597
323	97.49	99.98	99.81	99.55	99.68	10792250
183	82.91	99.99	99.99	99.98	99.99	10729038
283	96.2	99.99	99.88	99.74	99.81	10545503

Tabela 7.11: Tabela uspešnosti algoritma Hoeffding tree

ID vira	Večinski razred (v %)	Točnost	Preciznost	Priklic	F1 mera	Število primerov
385	39.51	99.91	99.9	99.9	99.9	4424181
403	87.44	99.96	98.78	91.24	94.51	10745333
122	56.75	99.89	99.79	99.7	99.74	10527924
21	75.91	99.97	99.97	99.95	99.96	10702887
347	98.47	99.99	99.69	99.31	99.5	2080100
263	87.41	99.98	99.92	99.92	99.92	10545566
5	74.9	99.97	99.9	99.88	99.89	10726501
3	96.86	99.99	99.9	99.86	99.88	10582379
186	91.59	99.96	99.77	99.77	99.77	10689972
7	52.64	99.98	99.96	99.95	99.95	10690698
384	98.01	99.99	99.98	99.96	99.97	4605950
185	53.2	99.98	99.84	99.52	99.68	10737380
141	99.82	99.99	99.84	99.79	99.81	10618120
121	79.32	99.98	99.83	99.77	99.8	10545806
363	73.62	99.93	99.9	99.87	99.89	4779106
4	93.14	99.97	99.84	99.85	99.85	10703393
61	48.8	99.97	99.96	99.95	99.95	10694456
6	99.24	99.99	99.92	99.66	99.79	10816242
443	94.38	99.89	96.27	97.65	96.95	50899
463	51.41	99.92	99.92	99.9	99.91	1074404
101	53.65	99.98	99.96	99.96	99.96	11067939
386	86.63	99.96	99.9	99.88	99.89	7782335
62	87.72	99.99	99.99	99.99	99.99	10689597
323	97.49	99.99	99.94	99.94	99.94	10792250
183	82.91	99.97	99.93	99.94	99.93	10729038
283	96.2	99.99	99.91	99.93	99.92	10545503

Tabela 7.12: Tabela uspešnosti algoritma Adaptive Random Forest

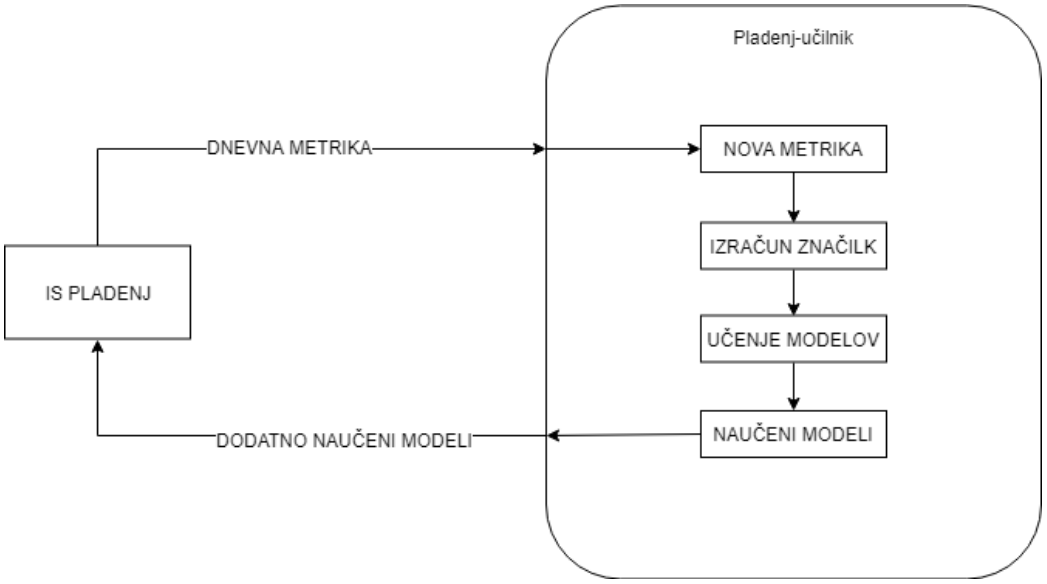
Za potrebe primerjanja obeh zasnov dušilnika smo lokalno postavili simulacijo IS Pladenj s petimi simuliranimi viri.

Na simuliranem Pladnju smo ustvarili metriko z različnimi situacijami, kot so odpovedi virov, 50% uspešnost itd., in z njo naučili napovedne modele. Ko smo zaključili z učenjem, smo najprej preverili delovanje brez dušilnika. Nato smo dodali dušilnik z napovednimi modeli in iste klice, ki smo jih prožili na sistem brez dušilnika, prožili še enkrat. Dušilnik je znal prepoznati neaktiven vir in v skladu s tem znižal prepustnost postopkov. Prav tako se je opazil upad števila poizvedb, ki končajo v samo-ponovitveni vrsti.

Na istem simuliranem okolju smo testirali še dušilnik s statistično napovedjo. Tudi ta je zaznal nedostopnost oziroma obremenjenost vira vendar pa je v primerjavi z dušilnikom z napovednimi modeli počasneje dvignil prepustnost, ko je vir začel ponovno delovati.

7.2 Postavitev

Da ne bi dodatno obremenjevali IS Pladenj s sprotnim učenjem in zaradi želje naročnika po večji kontroli nad napovednimi modeli, bo dodatno učenje potekalo na zunanjem virtualnem okolju. Za ta namen smo zasnovali tako imenovano komponento Pladenj-učilnik. Ta komponenta skrbi za dodatno učenje napovednih modelov. Vsako noč ob 1:00, ko je najmanjša uporaba IS Pladenj, se iz sistema prenese metrika za tisti dan. Iz nje se poračunajo značilke za posamezen vir, nato se posamezen model dodatno nauči in na koncu se novi napovedni modeli pošljejo nazaj v IS Pladenj.



Slika 7.1: Potek dodatnega učenja s komponento Pladenj-učilnik

Poglavje 8

Zaključek

Za zasnovo dušilnika prepustnosti v IS Pladenj smo predlagali dva načina. Prvi je bil zasnova s pomočjo statistike glede na zbrane podatke, drugi pa s pomočjo napovednih modelov, ki smo jih naučili z algoritmi strojnega učenja.

Za zasnovo dušilnika smo najprej zbrali podatke o stanju IS Pladenj in virih, na katere se povezuje. Iz njih smo nato pridobili značilke, s pomočjo katerih smo naučili napovedne modele.

Za potrebe primerjave obeh načinov smo postavili simulirano okolje Pladnja. Na tem okolju smo ustvarili podatke, na katerih smo učili napovedne modele, uporabljeni pa so bili tudi pri dušilniku na podlagi statistične analize. Oba dušilnika smo testirali na istih, v naprej določenih situacijah odzivov virov in spremljali njuno delovanje. Oba sta hitro prepoznala zmanjšano delovanje vira oziroma njegovo odpoved, vendar pa je dušilnik, ki deluje na podlagi statistike, potreboval več časa za dvig prepustnosti, ko je vir zopet začel normalno delovati. Zato smo se odločili v IS Pladenj implementirati dušilnik z napovednimi modeli.

Za določitev algoritma za učenje modelov smo na produkcijskih podatkih primerjali tri algoritme: Naive Bayes, Adaptive Random Forest in Hoeffding Tree. Odločili smo se za slednjega, saj je vrnil primerljive rezultate z Adaptive Random Forest, vendar pa je manj zapleten. Imel je zelo visoko klasifikacijsko točnost, kar je lahko posledica neenakomerne porazdelitve razredov,

saj je razred, kjer vir deluje optimalno, precej bolj zastopan kot ostala dva.

Poleg dušilnika bomo dodali komponento, ki bo skrbela za nadgrajevanje napovednih modelov. Poleg nadgrajevanja bo skrbela še za prenos novih modelov, ki jih dušilnik uporablja.

8.1 Nadaljnje delo

V nadaljevanju bomo pri učenju upoštevali neenakomerno porazdeljene razrede in cene napačne napovedi. Dodelali bomo še simulirano okolje, ki bo shranjevalo vse, kar se na njem dogaja, v svojo bazo. To bo pripomoglo k boljši primerjavi delovanja pred in po uvedbi novih sprememb. Izboljšali bomo tudi simulirane vire tako, da se bodo vklapljali in izklapljali sami, namesto da to storimo ročno. Še pred iztekom leta 2018 bomo dinamični dušilnik, v obliki ločene komponente, dodali v produkcijsko verzijo IS Pladenj.

Literatura

[1] Joerg Ziemann. Architecture of interoperable information systems. *An enterprise model-based approach for describing and enacting collaborative business processes, wirtschaftsinformatik-theorie und anwendung, Germany*, 2010.

[2] Imma Subirats. *AGRIS: Providing access to bibliographic information on agricultural science and technology*, 2018.

[3] Petar Brajak. Pladenj B51 - interoperabilnostni sistem za izvajanje elektronskih poizvedb. Tehnično poročilo, Ministrstvo za javno upravo, 2017.

[4] Tadej Justin, Petar Brajak, in Andraž Polanec. Zasnova dinamičnega dušilnika prepustnosti interoperabilnih sistemov z uporabo algoritmov strojenega učenja. V *25. konferenca Dnevi slovenske informatike*. Slovensko društvo Informatika, 2018.

[5] John Koenig. JBoss jBPM. *White Paper, JBoss Labs, Atlanta, GA*, 2004.

[6] Ministrstvo za javno upravo. Generične tehnološke zahteve (GTZ) za razvoj informacijskih rešitev v2.2.7. Tehnično poročilo, Ministrstvo za javno upravo, 2018.

[7] Rüdiger Wirth in Jochen Hipp. CRISP-DM: Towards a standard process model for data mining. V *Proceedings of the 4th international conference*

on the practical applications of knowledge discovery and data mining, strani 29–39. Citeseer, 2000.

[8] Albert Bifet, Ricard Gavaldà, Geoff Holmes, in Bernhard Pfahringer. *Machine learning for data streams with practical examples in MOA*. MIT press, 2018. `https://moa.cms.waikato.ac.nz/book/`.

[9] Ian H. Witten, Eibe Frank, Mark A. Hall, in Christopher J. Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[10] Kevin P. Murphy. Naive Bayes classifiers. *University of British Columbia*, 18, 2006.

[11] Pedro Domingos in Geoff Hulten. Mining high-speed data streams. V *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, strani 71–80. ACM, 2000.