Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Davorin Strehar

Optimizacija števila prikazov spletnih oglasov na uporabnika

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Zoran Bosnić

Ljubljana, 2014

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: 00546 / 2013
Datum: 15.9.2013

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

Kandidat: **DAVORIN STREHAR**

Naslov: **OPTIMIZACIJA ŠTEVILA PRIKAZOV SPLETNIH OGLASOV NA UPORABNIKA**
OPTIMIZING NUMBER OF ONLINE AD IMPRESSIONS PER USER

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V nalogi naj kandidat preuči problem optimizacije števila prikazov spletnih oglasov za vsakega posameznega uporabnika. Motivacijo naj črpa s področja spletnega oglaševanja, na osnovi katerega naj ovrednoti število prikazov glede na prihodke (uspeh oglaševanja) in odhodke (stroški oglasa). V nalogi naj uporabi metode strojnega učenja za napovedno modeliranje, rezultate naj ovrednoti in primerja s smiselnimi statičnimi modeli.

Mentor:

doc. dr. Zoran Bosnić

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Davorin Strehar, z vpisno številko **63990140**, sem avtor diplomskega dela z naslovom:

Optimizacija števila prikazov spletnih oglasov na uporabnika

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. marca 2014 Podpis avtorja:

Kazalo

Povzetek

Diplomska naloga se ukvarja z optimizacijo prikazovanja spletnih oglasov, kjer postreženi oglasi predstavljajo strošek, kliki na oglas pa prihodek oglaševanja. Oglas bi radi prikazali uporabniku natanko tolikokrat, kolikokrat je potrebno, da je verjetnost uporabnikovega klika nanj največja.

V te namene smo podatke o postreženih oglasih pretvorili v primerno obliko, jih analizirali in s postopki strojnega učenja na njih naredili napovedne modele s ciljem napovedovanja optimalnega števila potrjenih prikazov na uporabnika. Nato smo napovedne modele uporabili na simulaciji streženja oglasov in merili uspešnost posameznega postopka.

Rezultat naloge je izbira najboljšega postopka za ocenitev optimalnega števila potrjenih prikazov oglasov na uporabnika.

Ključne besede

spletno oglaševanje, napovedovanje, oglasi, strojno učenje

Abstract

This thesis deals with the optimization of displaying online ads (frequency capping), where ad serving represents a cost, and clicks on the ads represent advertising revenue. In this case we would like to display the ad to the user so many times to have the biggest probability of the ad click.

For this purpose, we converted the ad serving data to the appropriate format, analyzed it and used machine learning methods to make predictive models with the aim of predicting the optimal number of viewable impressions per user on a specific ad campaign. Afterwards, we used these predictive models in the ad serving simulation where we measured the performance of each procedure.

The result of the thesis is the selection of the best procedure to assess the optimal number of viewable ad impressions per user.

Keywords

online advertising, prediction, ads, machine learning

Poglavje 1

Uvod

Oglaševanje nas v digitalnem in analognem svetu spremlja na vsakem koraku. Kot gonilo sodobne digitalne komunikacije pa je v digitalnem okolju prisotno še toliko bolj, saj je s pomočjo oglaševalskih prihodkov možno ponujati vsebine oz. storitve uporabnikom brezplačno. Uporabniki so za to brezplačnost primorani z vsebino nalagati na svoje naprave tudi oglase. Po drugi strani pa oglaševalčev osnovni interes ni, da uporabniki gledajo oglase, temveč da ti uporabniki kupijo njihov izdelek ali storitev in da ustvarjajo čim večjo razliko med stroškom oglaševanja in prihodki od prodaje. Na oglaševalčevo srečo se da pot od prikazanega oglasa do nakupa optimizirati, ker lahko zelo natančno merimo, kaj se s prikazanimi oglasi dogaja.

Spletno oglaševanje lahko razdelimo na štiri korake uporabniške interakcije z oglasom, od postreženega oglasa do opravljene akcije. Prvi korak je, da je bil oglas postrežen na uporabniško napravo (prikaz), drugi, da je bil uspešno prikazan v vidnem polju naprave (potrjeni prikaz), tretji, da je uporabnik kliknil na oglas, in četrti, ko uporabnik izvede želeno akcijo (npr. nakup).

Tehnologija sicer omogoča oglaševalcu zakupovanje katerekoli uporabniške interakcije, tudi končno akcijo, ampak to vseeno ni vedno možno. Lahko gre za oglaševanje produktov, ki se jih ne prodaja na spletu, ali pa so ti produkti

zelo dragi (npr. avto) in je takih akcij zelo malo. Po drugi strani tak način zakupa oglasov predstavlja največje tveganje za založnika, saj ne more vedeti, kako uspešno se bo prodajal določen produkt, ki se nahaja v oglasu. Najbolj priljubljena plačilna modela sta zakupovanje na prikaze oglasov in klike na oglase.

V diplomski nalogi se bomo osredotočili na razmerje med prikazanimi oglasi in kliki. V tem primeru oglaševalec plačuje storitev za prikazane oglase, a si hkrati želi čim več interakcij z oglasi. Strošek prikazov lahko za oglaševalca pomeni tudi, da mora plačevati strošek tehnologije in infrastrukture za strežnje teh oglasov, klike na oglas pa plačuje še dodatno, kot strošek oglaševanja na založnikovi spletni strani. Plačevanje na prikazane oglase je še posebej pogosto pri oglasnih borzah, ki delujejo v stvarnem času, kjer se po sistemu najboljše ponudbe kupuje vsak prikaz oglasa posebej.

Cilj naloge je tako poiskati pravo razmerje med postreženimi oglasi glede na potencialne klike uporabnikov. V diplomski nalogi se opisanega problema lotevamo z zbiranjem zgodovinskih podatkov o streženju oglasov in z izvedbo statistične analize kot osnovnega merila uspešnosti. S postopki strojnega učenja izdelamo napovedni model in primerjamo uspešnost posameznega postopka z uporabo znanja na simulatorju streženja oglasov.

Poglavje 2

Pregled področja

2.1 Spletno oglaševanje

2.1.1 Poslovni model

Ekosistem spletnega oglaševanje je začel nastajati v letu 1993, ko je bil prodan prvi spletni oglas, in od takrat naprej hitro raste. V letu 2012 so prihodki spletnega oglaševanje v ZDA znašali 36,57 milijard dolarjev, kar je za 15,2% več kot v letu 2011 [17]. Po širši definiciji spletno ali internetno oglaševanje sicer obsega vse od pošiljanja e-pošte, oglaševanja v spletnih iskalnikih in družbenih omrežij do oglaševanja na spletnih straneh. V tej diplomski smo se osredotočili samo na oglaševanje na spletnih straneh, ker imamo za to oglaševanje podatke in ker se principi optimizacije na drugih področjih razlikujejo.

V spletnem oglaševanju sodelujejo trije akterji. Prvi je **uporabnik**, ki obiskuje spletne strani, bere vsebine ali pa uporablja raznorazne druge storitve. Drugi je **založnik**. Založnik je lastnik spletnih strani ali storitev, ki jim rečemo tudi spletni inventar. Tretji je **oglaševalec**, ki je pripravljen za del prostora na spletnem inventarju založniku plačati. Ker založnik dobi plačilo

od oglaševalca za spletni inventar, lahko, če le ima dovolj uporabnikov, ponudi vsebine brezplačno. To je tudi običajni poslovni model interneta. Vsebine oz. storitve je potrebno za vse uporabnike ustvariti le enkrat, ostanejo samo še stroški strojne opreme. To pomeni, da je strošek založnika na uporabnika lahko zelo majhen in da je za uspeh poslovnega modela pomembna samo številčnost uporabnikov.

Z leti so uporabniki razvili posebno "slepoto" za oglase, kar pomeni, da oglase hitro prepoznajo in jih načrtno prezrejo. Proti tej slepoti se oglaševalci borijo z vedno novimi oblikami oglasov. Najbolj klasična oblika je pasično oglaševanje, kjer se uporabnikom prikazujejo bolj ali manj animirani grafični elementi različnih velikosti. Običajne velikosti so 728x90px, 160x600px in 300x250px. Ime so dobili po obliki, ki je v večini primerov pasaste (podolgovate) oblike. Takoj za temi so se pojavili tekstovni oglasi, ki se bolj zlijejo z vsebino, pa tudi bolj enostavno jih je izdelati. Oboji se ponavadi nahajajo ob vsebini in je ne prekrivajo. Obstajajo pa še veliko bolj vsiljivi oglasi, ki prekrivajo vsebino in s tem prekinejo uporabniško izkušnjo. So veliko bolj opazni, a tudi precej nadležni. Problem nadležnosti je, da lahko škoduje oglaševalcu, saj lahko uporabniki razvijejo odpor proti oglaševanemu izdelku. Z razvojem internetnih omrežij, in s tem boljših linij do gospodinjstev, so postali pogosti video oglasi, kjer se uporabnikom na spletu prikazujejo isti oglasi, kot jih lahko vidijo na televiziji. Tako se tudi na področju oglaševanja TV in internet združujeta.

Za čim uspešnejše oglaševanje mora oglaševalec tudi uspešno ciljati skupine uporabnikov. Uporabnikov je veliko in stroški oglaševanja bi velikokrat presegli prihodke od končne prodaje, če bi oglase kazali vsem.

Najbolj osnovno je ciljanje po fizični lokaciji uporabnika (geolokaciji). Oglaševalec iz Slovenije bo ponavadi želel prikazovati oglase Slovencem, oglaševalec iz Avstrije pa Avstrijcem. Za ta način ciljanja se uporabljajo baze podatkov, kjer so zbrani IP naslovi v povezavi s fizično lokacijo [18]. Tako je to v bistvu ciljanje na gruče IP naslovov uporabnika. Lokacija je lahko država,

regija oz. mesto, bolj natančnih podatkov pa ponudniki IP podatkovnih baz načeloma nimajo. Bolj podrobno kot želi oglaševalec ciljati, slabša je natančnost. Drugače je pri mobilnem oglaševanju, ker imajo mobilni telefoni vgrajene sisteme za pozicioniranje – GPS. Mobilne aplikacije lahko zato sporočijo uporabnikovo lokacijo tudi na meter natančno.

Naslednje ciljanje je vsebinsko ciljanje, kjer se oglase prilagaja vsebini strani. Ideja je, da če uporabnik bere o maratonu, se mu lahko zraven članka ponudi tekaško opremo. Da to deluje, mora oglasni sistem pretočiti vsebino, kjer se oglasi prikazujejo, jo poindeksirati in tako predpripraviti za morebitno ciljanje oglaševalca. Ena izmed težav takšnega ciljanja je t.i. ocenitev razpoloženja vsebine, saj nočemo prodajati najboljših pnevmatik ob članku o prometni nesreči.

Trenutno zelo priljubljeno je vedenjsko ciljanje oglasov. V tem primeru oglasni sistemi izdelajo profile uporabnikov glede na to, katere spletne strani obiskujejo, kaj vpisujejo v spletne iskalnike (npr. Google) ali pa kaj so napisali v svoj profil na družbenih omrežjih (npr. Facebook). Na ta način omogočijo oglaševalcem, da izdelane profile uporabnikov uporabljajo pri ciljanju. Dostikrat se te podatke združi tudi z demografskimi podatki, ker se na ta način lahko cilja še spol in starost uporabnika. Ta tip ciljanja je še posebej priljubljen zaradi porasti realno-časnih oglasnih tržnic, kjer se lahko glede na vsak profil uporabnika odločamo, kolikšno ceno smo pripravljeni plačati za prikaz oglasa takemu uporabniku.

Drugi zelo pogost način ciljanja je ponovno trženje (angl. retargeting). Gre za to, da se uporabniku ob ogledovanju izdelkov v spletni trgovini shrani piškotek in potem se ta piškotek uporabi za ciljanje. V praksi to deluje tako, da potem, ko si recimo ogledamo pralni stroj v spletni trgovini, lahko enak pralni stroj opazimo v oglasu med prebiranjem spletnih novic na drugi spletni strani. Ker je taka očitna sledljivost nekoliko strašljiva za ljudi, je ta način oglaševanja tudi vzrok za novo zakonodajo o piškotkih in posledično tudi za vsa opozorila o piškotkih na spletnih straneh.

Ne glede na to, v kakšnem formatu je oglas in kako se ga cilja, pa se ga lahko še vedno optimizira glede na število prikazov uporabnikom z razlogom zmanjševanja direktnih stroškov ali pa indirektnih s preprečevanjem uporabnikove nasičenosti.

2.1.2 Mere za obračunavanje v spletnem oglaševanju

Oglaševalec in založnik se lahko dogovorita za različne načine sodelovanja, vendar gre v osnovi vedno za dogodek, vezan na oglas v povezavi z uporabnikom. Prvi je, da se oglas prikaže uporabniku, kar dejansko pomeni, da je bil oglas naložen na napravo uporabnika. Naslednji dogodek je potrjeni prikaz, to je, ko se oglas pojavi na ekranu v vidnem polju uporabnika. Oba se zaradi pogostosti dogodka in s tem majhnosti cene obračunavata na tisoč enot (angl. CPM – Cost Per Mille).

Prikaz oz. potrjeni prikaz še ne pomeni nobene direktne interakcije uporabnika z oglasom, a oglaševalec lahko kljub temu izračuna, koliko jih potrebuje za en pridobljen klik. Razmerje med prikazi in kliki se imenuje stopnja klikov (angl. CTR – Click Through Rate) in je ena pomembnejših metrik spletnega oglaševanja, saj se z njo ocenjuje uspešnost grafične podobe oglasa in primerja oglasne prostore na različnih spletnih straneh. Klik sam po sebi pomeni, da je uporabnik interaktiral z oglasom in da je bil usmerjen na oglaševalčevo spletno stran. Ta dogodek se obračunava posamezno s ceno na klik (angl. CPC – Cost Per Click).

Ko oglaševalec pridobi uporabnika na svojo spletno stran, lahko sledi njegovim premikom in pričakuje določeno akcijo. Akcija bi lahko bila npr. nakup izdelka, izpolnitev nagradne igre ali pa kakšnega obrazca. Seveda ni nujno, da se akcija zgodi takoj po kliku na oglas, ampak je možna tudi odložena akcija. Pri odloženi akciji je lahko zamik med klikom in dejansko akcijo tudi 90 dni. Da bi to delovalo, se ob kliku na oglas shrani na računalnik uporabnika manjša datoteka (piškotek), ki vsebuje informacije o uporabnikovem kliku

na oglas. Piškotek se prebere, ko uporabnik napravi nakup, in se tako zaključi transakcija. Cena na akcijo (angl. CPA – Cost Per Action) je običajno obračunana kot delež vrednosti nakupa.

2.1.3 Osnovni pojmi spletnega oglaševanja

V Tabeli 2.1 je povzeta do sedaj uporabljena terminologija.

pojem	opis
`uporabnik`	Obiskovalec spletne strani ali storitve.
`založnik`	Lastnik spletne strani ali storitve.
`oglaševalec`	Oseba oz. podjetje, ki bi rada prikazala določeno sporočilo uporabnikom.
`oglas`	Oglasno sporočilo.
`zahteva`	Zahteva za oglas je prišla na oglasni strežnik.
`prikaz`	Brskalnik je vstavil oglas v spletni dokument in s tem je bil oglas naložen na napravo uporabnika.
`potrjeni prikaz`	Oglas je bil v vidnem polju brskalnika vsaj 2 sekundi.
`klik`	Uporabnik je kliknil na oglas.
`akcija`	Uporabnik je izvedel želeno akcijo na strani oglaševalca.
`CTR`	Razmerje klikov glede na prikaze. Imenovano tudi stopnja klikov (angl. Cost Per Action).
`CPM`	Cena na tisoč prikazov oglas (angl. Cost Per Mille).
`CPC`	Cena na klik (angl. Cost Per Click).
`CPA`	Cena na akcijo (angl. Cost Per Action).

Tabela 2.1: Terminologija spletnega oglaševanja

2.1.4 Sorodna dela

Spletno oglaševanje je zelo konkurenčna panoga, zato obstaja precej člankov na temo optimizacije prikazovanja oglasov na uporabnika. Področje se stro-

kovno imenuje tudi *omejitev frekvence* (angl. frequency capping).

Zgodovinsko smo imeli najprej oglasne strežnike, nato oglasne mreže in sedaj imamo oglasne tržnice. Pri oglasnih strežnikih gre za optimizacijo založnikove strani. Na voljo je več oglasov pri omejenem oglasnem prostoru in zato se pojavi problem, kateri oglas prikazati. Tako avtorja v delu Omejitev frekvence v spletnem oglaševanju (angl. Frequency Capping in Online Advertising) [14] ugotavljata, kako najboljše v realnem času izračunati dobičkonosnost prikazovanja oglasov, in s tem optimizirati dodeljevanje oglasnega prostora oglaševalcem (angl. ad allocation). Pri večjih založnikih z več oglaševalci se pojavlja tudi kombinacija različnih poslovnih modelov in ciljanja. Nekateri oglaševalci zakupijo točno določeno (garantirano) število prikazov in uporabnikov, medtem ko drugi vzamejo, kar ostane. Take primere se optimizira s pomočjo uteži na prikazih oglasov na uporabnika [13].

Na strani oglasne mreže smo v položaju, ko je spletnega inventarja veliko in prav tako je veliko tudi oglasov. V tem primeru se poskuša optimizirati več oglaševalcev in založnikov hkrati, kjer se maksimira dobičkonosnost celotne oglasne mreže [7].

Novejši model se je pojavil v zadnjem času, s pojavom oglasnih tržnic. V oglasne tržnice so povezani tako posamezni oglaševalci kot tudi celotne oglasne mreže. V tržnicah vedno nekdo kupuje oglasni prostor – oglaševalec in nekdo ga prodaja – založnik. Gre za enostranske sisteme, ki so vezani samo na založnika, ko gre za optimizacijo prihodkov različnih oglasnih mrež, in samo na oglaševalca, ko gre za optimizacijo prikazovanja oglasov. Ker gre za sistem ponudb, je možno optimizirati velikost ponudbe glede na podatke, ki jih imamo, in glede na oglaševalske cilje [11].

V diplomski nalogi smo se postavili samo na stran oglaševalca, kjer se bomo osredotočili na to ali oglas prikažemo določenemu uporabniku ali ne. V primeru, ko bi prišli do omejitve prikazov na uporabnika, ne bomo dali ponudbe, oglas se tako ne bo prikazal, oglasna mreža pa bo morala zapolniti prostor drugače.

2.2 Strojno učenje

Za optimizacijo prikazov na uporabnika bomo uporabili postopke strojnega učenja. To pomeni, da bomo z različnimi algoritmi izdelali napovedni model, ki ga bomo nato uporabili pri napovedovanju maksimalnega števila potrjenih prikazov na uporabnika in oglasno akcijo.

Strojno učenje je področje umetne inteligence [5], ki se ukvarja z razvojem tehnik, ki omogočajo računalnikom oz. strojem, da se lahko učijo. Strojno učenje je v bistvu metoda za kreiranje računalniških programov na podlagi podatkov (vzorcev). Strojno učenje se s podatki ukvarja v smislu iskanja pravil v učnih podatkih – z algoritmi in računskimi operacijami.

V vseh razen v najbolj preprostih primerih vpogled ali znanje, ki se nahaja v neobdelanih podatkih, ne bo vidno, če si samo prostoročno ogledamo podatke. Na primer, odkrivanje neželene e-pošte (angl. spam), ki ocenjuje vsebino glede na prisotnost samo ene besede, ne bo zelo koristno. Če pa preučimo pojav besed, ki se pogosto uporabljajo skupaj v kombinaciji z dolžino e-pošte in drugimi dejavniki, pa bomo dobili jasnejšo sliko o tem, ali gre za neželeno e-pošto ali ne. Lahko bi rekli, da strojno učenje spreminja podatke v informacije [15].

Nekateri sistemi strojnega učenja poskušajo izločiti potrebo po človeški intuiciji za analizo podatkov, medtem ko drugi sistemi temeljijo na sodelovanju med človekom in strojem.

Algoritme strojnega učenja delimo na več vrst, glede na to, kaj je njihov cilj oz. rezultat učenja. Podrobneje opišimo področje nadzorovanega učenja (angl. supervised learning), ki ga bomo uporabili za modeliranje števila klikov na oglas.

2.2.1 Nadzorovano učenje

Nadzorovano učenje ali učenje z učiteljem je princip strojnega učenja za kreiranje (modeliranje) funkcije na podlagi učne množice vzorcev. Odziv je podan vnaprej, učenje mora nadgraditi model tako, da se bo rezultat ujemal z želenim odzivom oz. izhodnimi vrednostmi, ki jih določa človek oz. učitelj, nadzornik. Rečemo, da okolje deluje kot učitelj.

Učna množica vzorcev je par množice vzorcev in množice njihovih oznak razredov oz. par množice vhodnih podatkov sistema in množice želenih izhodov sistema. Izhod funkcije-sistema je lahko zvezno področje vrednosti – regresija, ali pa enolična oznaka razreda, ki mu pripada dani vzorec oz. vhodni podatki – klasifikacija. Naloga učečega sistema je, da generalizira znanje, ki ga dobi iz učne množice, na pravilen način.

Pri modeliranju z metodami nadzorovanega učenja običajno postopamo na naslednji način:

1. Določi se področje uporabe in s tem vrsto podatkov, ki jih sistem obdeluje.
2. Zbiranje podatkov. Podatki morajo biti značilni za področje, v katerem deluje sistem.
3. Izbira atributov (angl. feature selection): Atributi so lastnosti vhodnih podatkov, ki poudarjajo posebnosti posameznih razredov vzorcev. Boljše množice atributov zagotovijo natančnejšo predstavitev vzorcev in posledično privedejo do boljših rezultatov. Izločeni atributi so uporabljeni za izgradnjo vektorja atributov, ki je osnova za učenje. Pravilnost modelirane funkcije je precej odvisna od izbire atributov.
4. Definiranje strukture modela oz. predstavitve modelirane funkcije (algoritem). Primeri pogosto uporabljenih algoritmov:
   - drevesa,
   - nevronske mreže,

- metoda k najbližjih sosedov,
- metoda podpornih vektorjev,
- naivni Bayesov klasifikator,
- linearna regresija,
- ...

5. Izvedba metode učenja na učni množici vzorcev. Parametre algoritma se nastavi optimalno glede na testno množico vzorcev. Testna množica je lahko tudi podmnožica učne množice.

Regresija

Naloga regresijskega modela je, da za vzorec, ki je tako kot pri klasifikaciji opisan z množico atributov, določi vrednost odvisne (regresijske) spremenljivke, ki je, za razliko od klasifikacijskega razreda, zvezna (numerična).

Podobno kot pri klasifikaciji mora tudi regresijski predikator imeti predstavljeno zvezno funkcijo, ki preslika prostor atributov v napovedano vrednost. Ta funkcija je lahko podana vnaprej ali pa je naučena iz podatkov – primerov rešenih problemov v preteklosti. Naloga učnega algoritma je torej iz množice vzorcev z znanimi vrednostmi odvisne spremenljivke izračunati zvezno funkcijo, ki jo lahko uporabimo za določanje vrednosti regresijske spremenljivke novih primerov.

Najpogostejše uporabljeni regresorji so regresijska drevesa, linearna regresija, lokalno utežena regresija, regresija po metodi podpornih vektorjev ter usmerjene (večnivojske) umetne nevronske mreže. V nadaljevanju opišemo algoritme za napovedno modeliranje, ki jih bomo uporabili v eksperimentalnem delu diplome:

- Regresijska drevesa: Algoritmi za gradnjo regresijskih dreves, podobno kot pri odločitvenih drevesih, glede na oceno posameznih atributov v vozliščih izbirajo atribute in ustrezne podmnožice njihovih vrednosti za gradnjo regresijskega drevesa. Vozlišča v drevesu predstavljajo zvezne

in diskretne atribute, veje vrednosti atributov (konjunktivne pogoje). Listi predstavljajo sklepni del pravila, ki je poljubna zvezna funkcija, ki preslika vrednosti ostalih atributov v vrednost regresijske spremenljivke. Zvezna funkcija v listih predpostavlja, da so vsi preostali atributi zvezni. Najpogosteje se uporabljata točkovna funkcija (povprečna vrednost odvisne spremenljivke pri ustreznih učnih primerih) ter linearna regresijska funkcija.

- Linearna regresija: Linearna regresijska funkcija predpostavlja, da so vsi atributi zvezni (in ustrezno normalizirani) ter da obstaja linearna relacija med odvisno regresijsko spremenljivko in atributi. Treba je le določiti koeficiente linearne funkcije tako, da se minimizira vsota kvadratov napak regresijske spremenljivke preko vseh učnih primerov.
- Lokalno utežena regresija: Lokalno utežena regresija je metoda, ki je sorodna metodi k najbližjih sosedov in je prilagojena tako, da linearno regresijsko funkcijo izračuna samo za vrednosti regresijske spremenljivke najbližjih učnih vzorcev. Tako je linearna regresija prilagojena lokalnim lastnostim prostora učnih vzorcev, ki so najbolj podobni novemu vzorcu, katerega regresijsko vrednost iščemo.
- Metoda podpornih vektorjev (Support vector machines ali SVM) v regresiji: Naloga učnega algoritma je izračunati vrednosti koeficientov v naprej podane diskriminantne funkcije, ki predstavlja hiperploskev med dvema razredoma v prostoru atributov. Metoda podpornih vektorjev se prilagodi za reševanje regresijskih problemov tako, da definiramo rob okoli regresijske hiperravnine, znotraj katerega menimo, da je odvisna spremenljivka za vse vzorce eksaktno napovedana, primeri na robu pa so podporni vektorji, ki dejansko določajo potek hiperravnine. Seveda želimo tukaj rob okoli hiperravnine minimizirati, hkrati pa želimo minimizirati tudi napako na učnih primerih. Problem iskanja regresijske optimalne hiperravnine je definiran tudi kot optimizacijski problem kriterijske funkcije, ki upošteva napake napovedi regresijske spremenljivke na učnih primerih.

- Umetne nevronske mreže: Za regresijo se najpogosteje uporablja usmerjene večnivojske umetne nevronske mreže, ki kaskadno povezujejo vhodne nevrone (ki ustrezajo atributom), enega ali več nivojev skritih nevronov in en sam izhodni nevron, ki ustreza regresijski spremenljivki. Naloga učnga algoritma je nastaviti uteži na povezavah med nevroni (iz katerih vsak nevron izračunava uteženo vsoto) tako, da bo regresijska napaka čim manjša.

Klasifikacija

Klasifikacija ali uvrščanje je eno izmed najpogosteje uporabljenih področij strojnega učenja.

Naloga klasifikatorja je, da za objekt, opisan z množico atributov (vektor atributov), določi, kateremu izmed možnih razredov pripada. Atributi so neodvisne zvezne ali diskretne spremenljivke, s katerimi opisujemo objekte, razred pa je odvisna diskretna spremenljivka, ki ji določimo vrednost glede na vrednosti neodvisnih spremenljivk.

Zato, da lahko klasifikator določi razred, mora imeti na nek način predstavljeno diskretno funkcijo, ki preslika prostor atributov v razred. Ta funkcija je lahko podana vnaprej ali pa je naučena iz podatkov – primerov rešenih problemov v preteklosti.

Klasifikatorje ločimo glede na način predstavitve klasifikatorjeve funkcije.

Najpogosteje uporabljeni klasifikatorji so naivni Bayes-ov klasifikator, Bayesove verjetnostne mreže, odločitvena drevesa, odločitvena pravila, klasifikator s k najbližjih sosedov, linearna diskriminantna funkcija, logistična regresija, klasifikator po metodi podpornih vektorjev (SVM) ter usmerjene (večnivojske) nevronske mreže.

- Naivni Bayes: Naloga Bayes-ovega klasifikatorja je izračunati pogojne verjetnosti za vsak razred pri danih vrednostih (vseh) atributov za dani

novi primer (problem), ki ga želimo klasificirati. Bayes-ov klasifikator, ki natančno izračuna pogojne verjetnosti razredov, je optimalen, saj minimizira pričakovano napako. Ker Bayes-ovega klasifikatorja, ki bi natančno izračunal pogojne verjetnosti razredov, ne poznamo (razen v primerih, ko učna množica pokriva celoten prostor vrednosti vseh atributov), je potrebno izračunati približke verjetnosti z vpeljavo določenih predpostavk [5].

2.2.2 Učenje ansamblov modelov

Učenje ansamblov modelov (angl. ensemble learning) uporablja več napovednih modelov s ciljem, da se doseže boljše rezultate oz. boljše napovedovanje. Ansambel modelov je tako skupina klasifikatorjev oz. regresorjev z izbranim postopkom za njihovo kombiniranje ali izbiranje. Najpreprostejša sta glasovanje in izbor najboljšega člana ansambla modelov s prečnim preverjanjem. Glasovanje, ki ne zahteva nobenega učenja, poteka tako, da vsak član ansambla modelov prispeva svoj (utežen) glas za neko napoved. Napoved z največ glasovi zmaga. Ločimo dve različici: večinsko glasovanje, kjer ima vsak glas enako težo, ter uteženo glasovanje, kjer se utež glasov lahko spreminja (npr. glas točnejšega klasifikatorja ima večjo utež). Pri izboru najboljšega člana ansambla modelov s prečnim preverjanjem na učni množici ocenimo klasifikacijsko točnost vsakega člana. Za klasifikacijo novega primera uporabljamo le člana ansambla modelov z najvišjo natančnostjo.

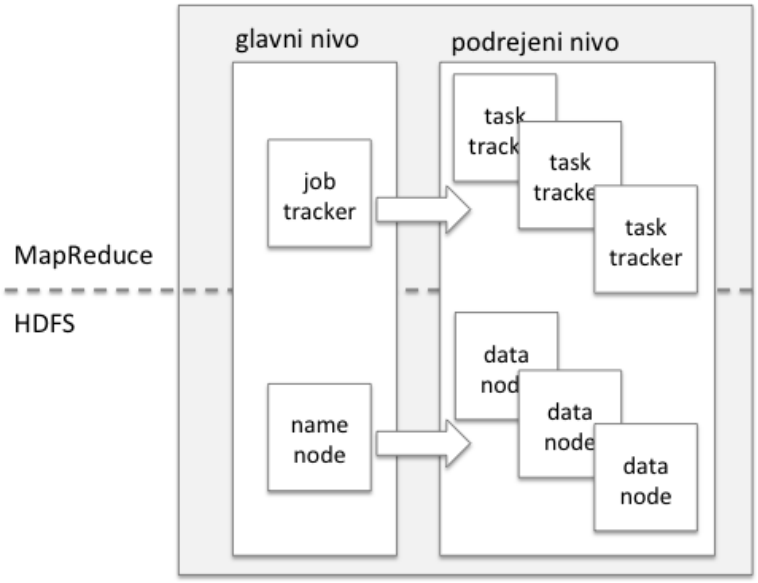
Ansambli modelov se delijo glede na to, kako so sestavljeni. Ločimo ansamble modelov, kjer se za gradnjo ansambla modelov uporablja en sam učni algoritem, raznolikost v ansamblu modelov pa se doseže npr. s spreminjanjem učne množice. Potem pa poznamo še ansamble modelov, ki vsebujejo napovedne modele z različnimi postopki strojnega učenja.

Najbolj znani metodi iz prve skupine sta *boosting* in *bagging* ter nadgradnja metode *bagging*, metoda *naključnih gozdov* (angl. random forest):

- *Bagging* [1] je metoda za kombiniranje regresorjev, ki izkorišča nestabilnost učnih algoritmov. Iz osnovne učne množice sestavimo več novih učnih množic, ki se od prvotne (in med seboj) nekoliko razlikujejo. Uporabimo postopek vzorčenja s ponavljanjem (angl. sampling with replacement): iz osnovne učne množice naključno izbiramo primere ter z njimi sestavljamo novo učno množico, pri čemer je pomembno, da je določen primer lahko izbran večkrat. Na ta način sestavimo učno množico z enakim številom primerov, kot jih ima prvotna učna množica. Opisani postopek imenujemo tudi *bootstrap*, ali natančneje *bootstrap* 0,632, ker nova učna množica v povprečju vsebuje le 63,2% primerov iz osnovne množice – preostali primeri so kopije. Na osnovi tako dobljenih učnih množic lahko s primernim učnim algoritmom zgradimo množico regresorjev, katerih napovedi združimo z računanjem povprečja [3].
- Podobno kot pri metodi *bagging*, tudi metoda *boosting* [2] kombinira regresorje, zgrajene z enim učnim algoritmom. Pri metodi *boosting* z iterativnim postopkom načrtno gradimo regresorje, ki se med seboj čim bolj dopolnjujejo. Postopek se začne s prvim, ki je zgrajen nad celotno množico z enako uteženimi primeri. Za naslednje regresorje v ansamblu modelov želimo, da bi pravilno zajeli primere, ki jih do sedaj zgrajeni regresorji niso. Zato povečamo težo primerov, ki so bili do sedaj napačno vključeni, in zgradimo nov regresor. S povečevanjem regresorjev se regresijska funkcija čedalje bolj prilega.
- Metoda *naključnih gozdov* [12] je nadgradnja metode *bagging*, kjer zgradimo več odločitvenih dreves in napovedi združimo z večinskim glasovanjem. Posamezna odločitvena drevesa zgradimo z vzorčenjem z zamenjavo in naključnim izborom atributov, uporabljenih v notranjih vozliščih. Gozd sestavlja veliko število odločitvenih dreves, ki so posamezno slabi klasifikatorji, a z njihovo kombinacijo dobimo dober klasifikator.

2.3 Obdelovanje podatkov

2.3.1 Hadoop



Slika 2.1: Arhitektura Hadooop platforme [16]

Količina podatkov se povečuje, narašča z neverjetno hitrostjo. Danes proizvedemo več podatkov v enem dnevu, kot smo jih od začetka človeštva do leta 2003 [8]. Ta eksplozija podatkov nas je pripeljala tudi do popolnoma novih postopkov obdelave podatkov, ki omogočajo paralelno obdelavo podatkov na gruči strežnikov, saj jih je preveč, da bi jih obdelal en sam strežnik. Ti novi postopki so dobro zajeti v odprtokodni platformi Hadoop, ki omogoča zanesljivo shranjevanje podatkov in njihovo obdelavo preko več strežnikov. Shranjevanje podatkov je omogočeno preko datotečnega sistema HDFS in njihova obdelava preko MapReduce opravil kot je vidno v Sliki 2.1.

HDFS

HDFS je kratica za Hadoop Distributed Filesystem (Hadoop porazdeljeni datotečni sistem), ki omogoča shranjevanje zelo velikih količin podatkov na omrežje standardnih strežnikov [10]. Ko količina podatkov preseže kapaciteto enega strežnika, je nujno, da se podatki porazdelijo preko več različnih strežnikov. Datotečni sistemi, ki omogočajo shranjevanje podatkov preko omrežja strežnikov, se imenujejo porazdeljeni datotečni sistemi. Ker gre za omrežje strežnikov in s tem tudi za vse mrežne težave (npr. prekinitev povezave), so porazdeljeni omrežni sistemi veliko kompleksnejši kot navadni datotečni sistemi. Največji izziv je narediti robusten sistem, kjer se podatki ne bodo izgubili ob izpadu enega strežnika iz omrežja.

MapReduce

Najbolj priljubljen postopek paralelne obdelave zadnjih let je MapReduce, ki so ga razvili pri podjetju Google [9]. MapReduce je relativno preprost model za procesiranje podatkov, ki pa že v svoji osnovi omogoča vzporednost in tako omogoča obdelavo masivnih količin podatkov vsakemu, ki ima na razpolago dovolj strežnikov.

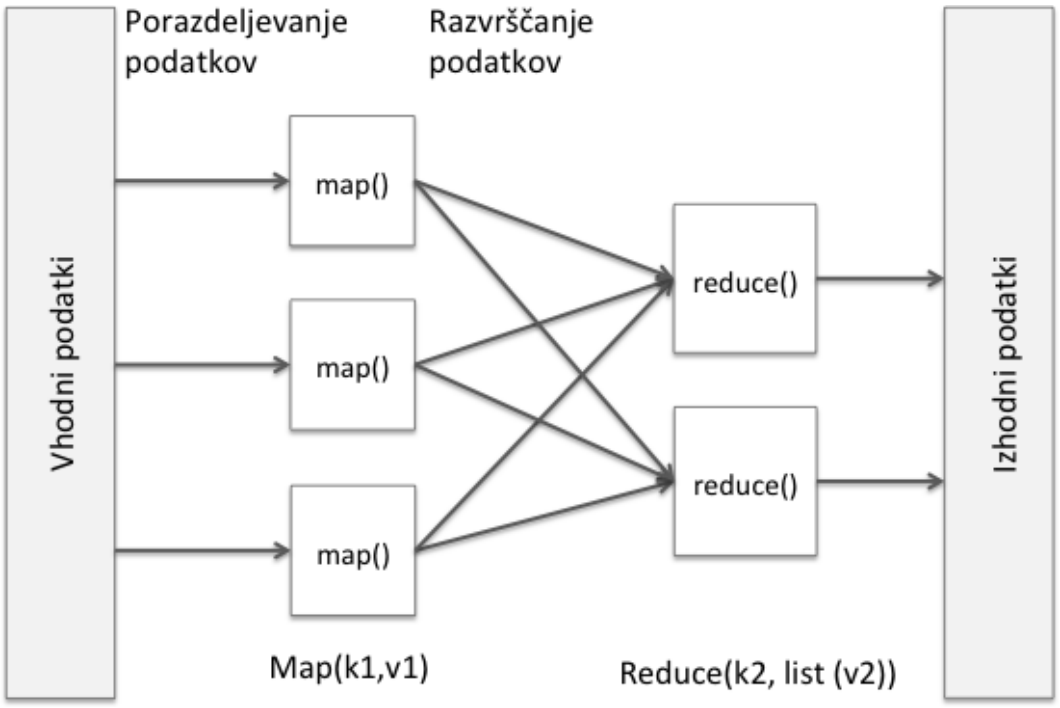
Deluje tako, da porazdeli obdelavo podatkov na dve fazi. Prva je *Map*, ki za vhod vzame ključ-vrednost pare ene domene in jih lahko spremeni v drugo:

`Map(k1,v1)` $\rightarrow$ `list(k2,v2)`

Map se lahko izvaja paralelno na vsakem paru v vhodni množici (vhodnih podatkih), ki se tako tudi porazdeli po strežnikih. Druga faza je *Reduce*, ki za vhod vzame razvrščene izhodne podatke iz *Map* in jih spremeni v množico vrednosti:

`Reduce(k2, list (v2))` $\rightarrow$ `list(v3)`

Shema MapReduce opravila je vidna v Sliki 2.2.



Slika 2.2: Shema MapReduce [16]

Programer specificira metodi *Map* in *Reduce* in s tem definira, kako se bodo podatki obdelali. Hadoop omogoča, da so MapReduce programi lahko napisani v programskih jezikih vse od Jave, Python ali C++. Za potrebe te diplomske naloge smo uporabili programski jezik Python.

Apache Hive

Hive [20] je sistem za upravljanje podatkovnih skladišč, s katerim preprosto upravljamo s podatki na HDFS, saj nam omogoča, da za običajne besedilne datoteke definiramo tabele in tipe atributov. Seveda morajo biti podatki v vseh vrsticah datotek, ki jih označimo kot tabelo, tipsko enaki. Datoteke so ponavadi v formatu CSV, ki vsebuje z vejico ločene vrednosti v tekstovni datoteki, zato predstavlja vsaka z vejico ločena vrednost nov atribut.

Ko imamo nastavljene Hive tabele, lahko na njih izvajamo SQL poizvedbe,

ki se v ozadju samodejno pretvorijo v MapReduce opravila. To nam zelo olajša delo z MapReduce, saj lahko delamo s podatki zelo podobno, kot bi to počeli v relacijski bazi, le da so ti shranjeni v tekstovnih datotekah in da izvajanje ukaza traja precej dlje.

2.3.2 Weka

Weka (Waikato Environment for Knowledge Analysis) [24] je zbirka algoritmov strojnega učenja za opravila podatkovnega rudarjenja. Algoritme se lahko bodisi neposredno uporabi na naboru podatkov ali pa se jih pokliče iz kode. Weka vsebuje orodja za predpripravo podatkov, klasifikacijo, regresijo, rojenje, asociativna pravila in vizualizacije. Prav tako je zelo primerna za razvoj novih shem strojnega učenja. Weka je napisana v programskem jeziku Java, kar je zelo koristno, če uporabljamo programski jezik Clojure.

Poglavje 3

Eksperimentalno okolje

V tem poglavju se bomo osredotočili na predstavitev priprave uporabljenih podatkov, razvojno in eksperimentalno okolje ter algoritme strojnega učenja.

3.1 Struktura podatkov

Za diplomsko nalogo smo uporabili podatke iz spletnih oglasnih strežnikov, ki so del oglasne mreže v Srbiji. Oglasni strežniki zapisujejo dnevnike dogodkov ob prikazovanju oglasov spletnim uporabnikom v tekstovno obliko CSV, kjer so različni atributi v datoteki ločeni z vejico. Uporabili smo podatke od junija do novembra 2013. Izvorna velikost podatkov je nekaj več kot 6 TB.

V podatkih so zajeti štirje različni dogodki: Zahteva, Prikaz, Potrjeni prikaz in Klik. Podatki, ki se shranijo ob vsaki uporabnikovi zahtevi, so prikazani v Tabeli 3.1. Strukturo podatkov ob prikazu, potrjenem prikazu ali kliku pa lahko vidimo Tabeli 3.2.

atribut	opis
ad_hash	identifikator dogodka
viewer_tag	identifikator uporabnika
ad_id	identifikator oglasa
ad_format	velikost oglasnega bloka
event_time	čas dogodka
...	podatki vsebujejo še druge atribute, ki za to obdelavo niso pomembni

Tabela 3.1: Struktura podatkov ob zahtevi

atribut	opis
ad_hash	identifikator dogodka
event_time	čas dogodka
...	podatki vsebujejo še druge atribute, ki za nas niso pomembni

Tabela 3.2: Struktura podatkov ob prikazu, potrjenem prikazu ali kliku

Podatki o dogodkih nam povedo, kateri uporabnik je kdaj kliknil na oglas. Te podatke pa lahko v našem primeru oplemenitimo še z atributi, ki predstavljajo lastnosti oglasov in so prikazani v Tabeli 3.3.

atribut	opis
ad_id	identifikator oglasa
group_id	identifikator oglasne skupine
campaign_id	identifikator oglasne akcije
targeting	način ciljanja oglasov
ad_product	oglasni produkt
cat_primary	primarna kategorija oglasa
cat_secondary	sekundarna kategorija oglasa
ctr	zgodovinska stopnja klikov glede na prikaze oglasa

Tabela 3.3: Lastnosti oglasov

3.2 Predpriprava podatkov

Da bi eksperiment približali realnosti, smo podatke razdelili na dva večja dela. Prvi del podatkov sestavljajo učni primeri. Na teh podatkih bomo gradili napovedne modele. Drugi del so podatki, ki jih bomo uporabili za simulacijo streženja oglasov in ocenitev uspešnosti. Dostikrat se za ocenjevanje uspešnosti uporablja prečno preverjanje učnih primerov, kjer se del učnih primerov uporablja za učenje in del za testiranje, vendar to pri nas ne bo primarni način ocenjevanja uspešnosti, ker bi radi testirali uspešnost postopkov v kar se da realnem okolju.

Za pripravo učnih primerov smo potrebovali tudi zgodovinske podatke o uporabnikovih interakcijah z oglasi. Za zgodovinske podatke uporabljamo dnevnike dogodkov iz junija, julija in avgusta 2013. Učni primeri bodo izhajali iz dnevnikov dogodkov iz septembra in oktobra 2013, ki jih bomo oplemenitili z zgodovinskimi podatki. Streženje podatkov bomo simulirali z novembrskimi podatki, kjer bomo ocenjevali uspešnost postopkov.

Uporabljamo podatke oglasnega sistema, kjer oglaševalec upravlja z oglasi skozi trinivojsko hierarhijo. Če pogledamo od spodaj navzgor, so oglasi združeni v oglasne skupine, ki so potem združene v oglasne akcije. Oglasna akcija lahko vsebuje več oglasnih skupin in oglasna skupina lahko vsebuje več oglasov. Da bi dobili čim boljše rezultate, bomo podatke o oglasih (ad_id) združili v oglasne akcije (campaign_id). To lahko naredimo, ker so oglasi v oglasnih akcijah zelo sorodni. Tematika oglasa je skoraj vedno enaka, razlikuje se le grafična podoba.

3.2.1 Ciljna struktura podatkov

Zgodovinski podatki

Kot napisano, bodo zgodovinski podatki vsebovali zgodovinske podatke o uporabnikovih interakcijah z oglasi. V Tabeli 3.4 je razvidna relativno preprosta struktura teh podatkov – gre samo za seznam klikov na oglasne akcije posameznega uporabnika. Te podatke bomo potem združili v učnih primerih za izdelavo napovednega modela.

atribut	opis
`viewer_tag`	identifikator uporabnika
`campaign_id`	identifikator oglasne akcije
`click_time`	čas dogodka

Tabela 3.4: Lastnosti oglasov

Učni primeri

Za napovedovanje bomo združili zgodovinske podatke o uporabnikih in oglasnih akcijah in prešteli, koliko potrjenih prikazov je uporabnik potreboval, preden je kliknil na oglas.

ad_format Format oglasnega bloka. Diskretni atribut: 0x0, 120x240, 120x600, 125x125, 160x240, 160x600, 180x150, 234x60, 240x350, 240x400, 250x250, 300x250, 300x600, 336x280, 468x60, 500x0, 510x0, 608x0, 620x90, 728x90

product Oglasni produkt. Diskretni atribut: bigshop, engage, rich, text, shop

targeting Način ciljanja oglasne akcije. Diskretni atribut: contextual, performance

ctr Zgodovinsko razmerje med potrjenimi prikazi in kliki oglasne skupine. Zvezni atribut.

cat_primary Primerna kategorija oglasne skupine. Diskretni atribut: Business, FarmAnimals, Games, Internet, Politics, ConsumerElectronics, Motorcycles, Religion&Spirituality&Astrology, Cooking&Recipes, Books&Literature, ComputerHardware, DentalCare, Movies, Fragrances, E-Commerce, Accounting&Tax, Cosmetics, Hygiene&Toiletries, Cookware, Music, FoodRetailers, Sex, Teenagers, MobilePhones, Telecommunications, Software, Construction&Maintenance, Society, Parenting&Family, Beauty&PersonalCare, Entertainment, Education, Clothing, Co-

mics&Cartoons, Arts&Hobies, News&CurrentEvents, GraphicDesign&Publishing, Kids&Teens, RealEstate, Dating&Relationships, AlcoholicBeverages, Gossip&Tabloids, Footwear, SummerSports, Advertising&Marketing, InternetISPs, Agriculture&Forestry, VisualArts, TravelLocal, Beauty&PersonalCare, WebDesign&Development, Watches&Accessories, SocialNetworks, WebHosting&DomainRegistration, Architecture, Office&PrintingServices, VideoEquipment, TravelSerbia, Computers&Electronics, SearchEngines&SEO&Marketing, HealthConditions, Alternative&NaturalMedicine, Sport, Lifestyle, Clubs&Nightlife, Garden, CasinoGames&Gambling, Toys, Automotive, Energy&Utilities, Restaurants, LuxuryGoods&Jewelry, VideoGames&OnlineGames, Animals, TravelWorld, Finance&Insurance, AudioEquipment, Nutrition, HumanResources, Legal, Transportation&Logistics&Aviation, Insurance, NonalcoholicBeverages, Travel, Food&Drink, WeightLoss, InsideSports, Banking&Credit&Leasing&Investing, Theater&PerformingArts, TravelEurope, Health, BloggingResources&Services&Forums, Industries, Attractions&Activities, Fashion&Modeling, WomensClothing, ComputerLaptops, Location&Tracking, Home, Cars

cat_secondary Sekundarna kategorija oglasne skupine. Diskretni atribut z enakimi vrednostmi kot *cat_primary*

clicked_campaign_before Uporabnik je že kliknil na trenutno oglasno akcijo. Binarni atribut: T, F

views Število potrjenih prikazov pred klikom. Se pravi kolikokrat je uporabnik videl oglase oglasne skupine, preden je kliknil nanj. Zvezen atribut, ki ga napovedujemo.

clicked_before Uporabnik je že kdaj kliknil na oglas. Binarni atribut: T, F

Ker je napovedni atribut zvezen, se bomo osredotočili na regresijske metode strojnega učenja.

Podatki za simulacijo

Učne primere že imamo, sedaj pripravimo še podatke za simulacijo streženja oglasov. To so praktično direktni podatki iz dnevnikov dogodkov, ki delujejo kot filmski posnetek postreženih oglasov. Le vnaprej jih oplemenitimo z zgodovinskimi podatki uporabnikov in oglasnih akcij, da bo simulacija hitreje delovala.

Vrstica v dnevniku dogodkov za simulacijo bo vsebovala naslednje atribute:

`viewer_tag, campaign_id, ad_format, event_time, product, targeting, ctr, cat_primary, cat_secondary, clicked_before, clicked_campaign_before, view, click`

view je 'F', kadar je bil oglas samo postrežen, potrjeni prikaz pa se ni zgodil, drugače je 'T'.

click je 'T', ko je uporabnik kliknil na oglas, oz. 'F', kadar se to ni zgodilo.

Te podatke bomo uredili po času, uporabniku in oglasni akciji ter jih uporabili kot vhod v sistem, ki bo simuliral obnašanje oglasnega strežnika, ki za omejevanje prikazov oglasov uporablja napovedni model.

3.2.2 Uporabljena orodja

Pri reševanju problema smo uporabili cel spekter različnih orodji. Za predpripravo in analizo podatkov smo uporabili platformo za obdelavo masivnih podatkov (angl. big data) Hadoop [19], kjer smo uporabili Hive [20] za enostavno izvajanje SQL poizvedb na podatkih in programski jezik Python [23] za pisanje bolj naprednih MapReduce [9] opravil na platformi. Za analizo, strojno učenje in simulacijo streženja oglasov smo uporabili funkcijski programski jezik Clojure [22] in knjižnico algoritmov strojnega učenja Weka [24]. Vsa uporabljana orodja so odprtokodna.

3.2.3 Postopek predpriprave

V Hive smo za vsako skupino dogodkov, zajetih v množico datotek, definirali tabelo. Primer definicije tabele za prikaze je na Sliki 3.1.

```
CREATE TABLE impression (
        ad_hash string,
        viewer_ip string,
        event_time timestamp,
        proxy_ip string,
        viewer_tag string
)
ROW FORMAT serde 'com.bizo.hive.serde.csv.CSVSerde';
```

Slika 3.1: Tabela prikazov

Nalaganje podatkov v Hive pomeni, da se podatki premaknejo brez kakršnekoli spremembe, zato se ukaz vedno izvede hitro. Za razliko od običajnih relacijskih baz se struktura preveri ob izvajanju SQL poizvedbe, in če podatki niso v pravem formatu, poizvedba ne bo delovala. Tak arhitekturni kompromis pomeni, da se v zameno za hiter uvoz podatkov poizvedbe počasneje izvajajo.

Pri relacijskih podatkovnih bazah smo navajeni na posodabljanje podatkov, uporabo indeksov in transakcije. Tega Hive trenutno ne omogoča, je samo plast, ki omogoča zelo enostavno brskanje po podatkih na HDFS.

Ko smo definirali podatkovne tabele, smo napisali SQL poizvedbo, ki je združila podatke glede na identifikator dogodka in jih zapisala v nove tabele. Za začetek smo združili *zahteve*, *prikaze* in *potrjene prikaze* v skupno tabelo. Nato pa smo dodali še dodatne podatke o oglasih in zgodovinske podatke o uporabnikih.

Podatki so sedaj združeni, a so še vedno v formatu dnevnika – vsak dogodek je v svoji vrstici. Ta oblika je sicer že primerna za podatke, ki jih potrebujemo za simulacijo, a za učne primere jih moramo združiti zaporedno po dogodkih. Tukaj bomo šteli, kolikokrat je uporabnik videl določeno oglasno akcijo, preden je kliknil nanjo. Tu ne gre enostavno s SQL poizvedbo, zato je bilo potrebno spisati opravilo MapReduce.

V fazi *Map* združimo identifikator uporabnika in oglasne akcije v skupni ključ,

da bomo imeli v fazi *Reduce* pod enim ključem celoten dnevnik dogodkov kombinacije uporabnika in oglasne akcije.

V fazi *Reduce* se sprehodimo skozi dnevnik dogodkov enega uporabnika v kombinaciji z eno oglasno akcijo, kot smo to nastavili v fazi *Map*. Za vsako kombinacijo preštejemo število potrjenih prikazov za klik. V primeru, da je uporabnik kliknil večkrat, začnemo ob vsakem kliku štetje znova.

Končni rezultat opravila MapReduce je učna množica, ki smo jo uporabili za izdelavo napovednega modela.

Za testne podatke smo vzeli pet milijonov vrstic novembrskih podatkov, ki smo jih uredili po uporabniku in oglasni akciji, pogledali, kje se zaključi dnevnik dogodkov predzadnjega uporabnika in oglasne akcije ter pobrisali ostalo.

3.3 Analiza podatkov

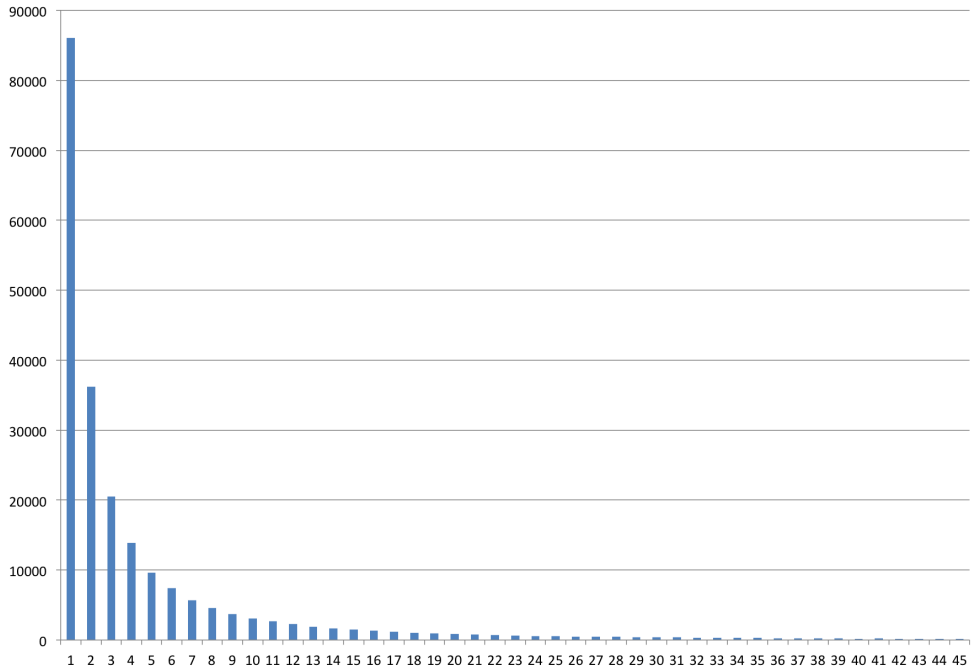
3.3.1 Učni primeri

Iz podatkov smo izluščili dvesto tisoč učnih primerov, na katerih bomo gradili napovedne modele. V učnih primerih so zajeti dnevniki dogodkov, kjer se je dejansko zgodil klik na oglas.

Statistični vpogled v razmerja med potrjenimi prikazi in kliki uporabnika:

Učnih primerov: 200.000
Minimum: 1
Maksimum: 10.538
Povprečna vrednost: 7,931
Standardna deviacija: 49,242

Povprečno vrednost potrjenih prikazov smo kasneje uporabili za izračun primerjalne ocene uspešnosti postopkov.



Slika 3.2: Porazdelitev števila potrjenih prikazov pred klikom

Iz grafa porazdelitve števila potrjenih prikazov (Slika 3.2) pred klikom lahko vidimo, da se kliki ponavadi zgodijo že zelo zgodaj v življenjskem ciklu oglasa. Skoraj 40% uporabnikov klikne na oglas, ko ga vidi prvič.

3.3.2 Testni podatki

Ker je naš cilj realna ocena uspešnosti, smo za testno množico vzeli del podatkov iz novembrskega dnevnika dogodkov oglasnega strežnika. S pomočjo teh podatkov bomo lahko naredili dobro oceno, ali je določen postopek ekonomsko sprejemljiv ali ne.

Številčnost posameznih dogodkov v testni množici:

Prikazov: 4.999.999
Potrjenih prikazov: 1.880.987
Klikov: 684

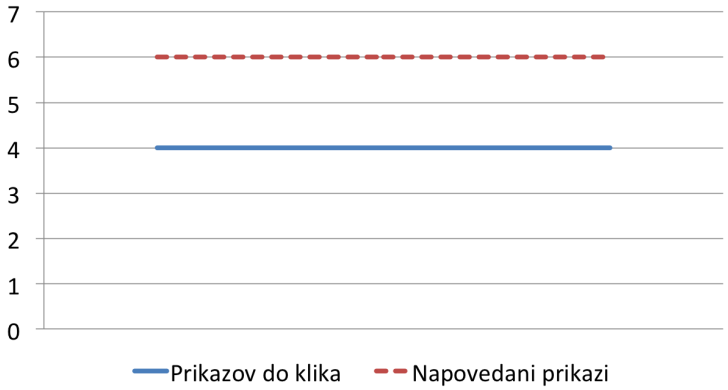
Stopnja klikov (CTR): 0,0137%

Za razliko od učnih primerov imamo v testnih podatkih zajete podatke s kliki in brez klikov. Kot lahko vidimo, se kliki zgodijo redko – nekaj več kot eden na deset tisoč.

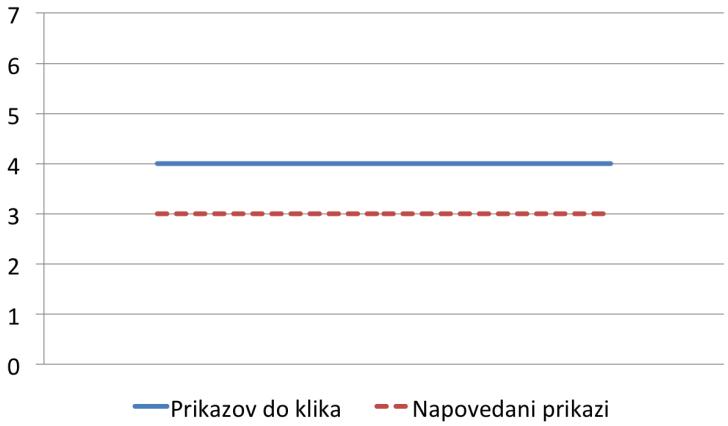
Poglavje 4

Ocenjevanje uspešnosti modeliranja

Kot smo napisali v uvodu, bo naše merilo uspešnosti razlika med številom prikazov in številom klikov na oglas. Zaradi narave problema smo se za ocenjevanje odločili izdelati lasten prilagojen model ocenjevanja, ki je zmožen oceniti ekonomičnost določenega postopka. Klikov se zgodi malo, a imajo veliko vrednost, na drugi strani pa imamo tudi veliko prikazov, ki so vredni malo. Za nas je torej pomembnejše, da naredimo kakšen prikaz več kot premalo in s tem ne izgubimo klika na oglas.



Slika 4.1: Napovedano število potrjenih prikazov je preseglo klike



Slika 4.2: Napovedano število potrjenih prikazov je pod kliki

Če pogledamo Sliki 4.1 in 4.2 lahko vidimo, da je pri Sliki 4.1 relativna napaka napovedi res manjša, a smo pri tem izgubili vse klike. Večina generičnih ocenitvenih modelov tega ne upošteva, zato smo se odločili za bolj plastično ocenitev uspešnosti, ki vključuje simuliranje streženja oglasov.

Prikazov nikakor ne more biti manj kot 0, tudi če je napovedano negativno število. Ker oglaševalec omejuje množico uporabnikov, ki jih bodo oglasi dosegli s ciljanjem, mu jih mi ne bomo dodatno omejevali – oglas bomo prikazali vsaj enkrat vsakemu uporabniku.

4.1 Mere za uspešnost učenja

Da bi najlažje realno prikazali uspešnost postopka, bomo vsak prikaz pomnožili s stroškovno oceno in klik s prihodkovno oceno. Ob vsakem prikazu oglaševalčev strošek znaša 0,00002 €, ob kliku ima prihodek 0,40 €. V realnosti bi se številke razlikovale glede na oglasno akcijo in oglaševalčeve cilje, a za našo oceno uspešnosti bo to zadostovalo. Razlika med prikazi in kliki je velika, ker je tudi v povprečju potrebno vsaj tisoč prikazov oglasa na en klik. Ker pa so stroški prikaza izredno majhni, a tudi pogosti, se običajno upora-

blja cena na tisoč prikazov, ki se ji reče tudi CPM (Cost Per Mille).

Po končani simulaciji bomo vse prikaze in klike pomnožili z ocenami, nato odšteli skupni strošek od skupnega prihodka in dobili dobiček. Dobiček je absolutna razlika, ki bo pri večjem številu klikov in prikazov ter pri istem razmerju večja.

Drugi pomemben parameter je CTR, ki zagotavlja, da smo dobili čim več klikov s čim manj prikazi. Večji kot je CTR, hitreje bo oglaševalec prišel do ciljnega števila klikov in boljše bomo izkoristili prikaze oglasov.

```
CTR = kliki / prikazi * 100
```

Koeficient uspešnosti Da bi maksimirali oba parametra, dobiček in CTR, ju bomo pomnožili med seboj in glede na to ocenili uspešnost določenega postopka. V našem primeru bo ta številka zelo majhna, zato jo pomnožimo še z 10.000.

```
K = CTR * Dobiček * 10000
```

V Tabeli 4.1 smo izračunali K za našo testno množico.

Omejitev	Prikazi	Kliki	Strošek	Promet	Dobiček	CTR	**K**
Brez	4999999	684	100,00€	136,80€	36,80€	0,0137%	**50,34**

Tabela 4.1: Prikaz izračuna uspešnosti na testnih podatkih

Za izračun uspešnosti *potrjeni prikazi* niso pomembni, uporabljamo jih za napovedovanje in s tem tudi za omejevanje števila prikazov uporabnikov, saj so to prikazi, ki so bili dejansko vidni uporabniku in lahko vplivajo na to, ali bo uporabnik izvedel klik ali ne.

4.2 Simulacija streženja oglasov

Za simulacijo smo spisali program, ki za vhodne podatke vzame napovedni model in dnevnik združenih dogodkov, urejenih po uporabnikih (*viewer_tag*), oglasnih akcijah (*campaign_id*) in času dogodka, in bo kot rezultat vrnil omejen dnevnik dogodkov glede na napovedni model.

Program je bil napisan v programskem jeziku Clojure, ki spada v družino programskih jezikov Lisp. Je funkcijski, operira predvsem z nespremenljivimi objekti (angl. immutable objects) in ni tipiziran, tipov spremenljivk ni potrebno definirati vnaprej, pozna refleksijo, makre in funkcijske ovojnice (angl. closure). Ker teče na javanskem navideznem stroju (angl. Java Virtual Machine) ima dostop do vseh programskih knjižnic napisanih v Javi, kar je priročno. Najbolj znan je zaradi dobre podpore nadzora sočasnosti v programih s tehniko programskega transakcijskega pomnilnika (angl. Software Transactional Memory ali STM).

Priročnost jezika Clojure se pokaže že takoj na začetku. Ker bomo delali z velikimi tekstovnimi datotekami, moramo obdelovati podatke vrstico po vrstico, blok po blok. Da bi to lahko učinkovito naredili, moramo odpreti datoteko, jo leno brati (angl. lazy read) po vnaprej določenih blokih in na vsakem bloku izvesti operacijo. Blok je v našem primeru dnevnik dogodkov uporabnika z oglasno akcijo. Kot lahko vidimo na Sliki 4.3, smo z nekaj programskimi ukazi dosegli ravno to. Procedura *processor-partitioned* odpre datoteko, vsako vrstico spremeni v atribute glede na CSV strukturo, jo nato porazdeli s ‘partition-by‘ proceduro, ki je del Clojure knjižnice za porazdeljevanje seznama glede na podan parameter – pri nas je to prvi atribut, in jo pošlje v našo obdelovalno proceduro.

```
(defn processor-partitioned
  "Procesiranje podatkov.
   Vhodni parametri:
    * `filename` datoteka, ki jo prebiramo
    * `func` procedura s katero obdelujemo podatke
   Opomba: v prvi vrstici pricakuje seznam atributov, zato jo
    preskoci"
  [filename func]
  (with-open [rdr (clojure.java.io/reader filename)]
    (func (partition-by first (rest (csv/parse-csv rdr))))))
```

Slika 4.3: Procedura za izvajanje funkcijske ovojnice na dnevniku dogodkov

Kot funkcijsko ovojnico bomo podali proceduro, ki bo za vsako kombinacijo uporabnika in oglasne akcije vrnila samo toliko prikazov, kot ji bomo zapovedali. V primeru, da se bo v zapovedanih prikazih zgodil klik, se bodo števci ponastavili in s tem bomo uporabniku "omogočili", da klikne na oglasno akcijo več kot enkrat. V vsakem primeru se bo oglas uporabniku prikazal vsaj enkrat, tudi če bo napoved enaka ali manjša ničli.

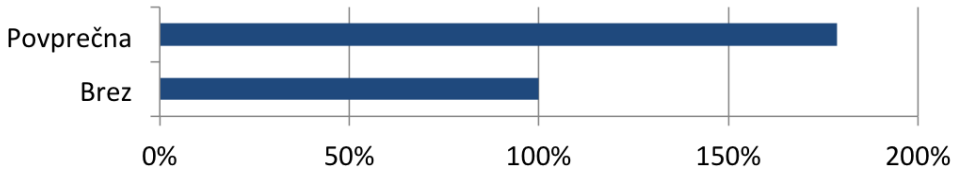
Rezultat obdelave bo skrajšan dnevnik dogodkov postreženih oglasov, v katerem je za končni rezultat potrebno samo še prešteti preostale dogodke v množici in izpisati rezultate.

Sedaj lahko uporabimo povprečno vrednost potrjenih prikazov pred klikom, ki smo jo izračunali pri statistični obdelavi učnih primerov, in jo primerjamo z rezultati, ko prikazov ne omejujemo. Kot lahko vidimo v Tabeli 4.2, imamo pri povprečni vrednosti omejitve potrjenih prikazov za skoraj dvakrat boljši rezultat K, kot kadar prikazov ne omejujemo. Ta ocena nam bo služila kot osnova za primerjavo uspešnosti algoritmov strojnega učenja, saj iščemo boljši K, kot smo ga dobili s povprečjem.

Omejitev	Prikazi	Kliki	Strošek	Promet	Dobiček	CTR	**K**
Brez	4999999	684	100,00€	136,80€	36,80€	0,0137%	**50,34**
Povprečna	3496234	608	69,92€	121,60€	51,68€	0,0174%	**89,86**

Tabela 4.2: Primerjalna tabela rezultatov med povprečno vrednostjo omejitve prikazov in brez omejitve

Slika 4.4 nazorno prikazuje razmerje velikosti parametra K pri simulaciji brez omejitve in s povprečno omejitvijo prikazov na uporabnika.



Slika 4.4: Razmerje parametra K pri omejeni in neomejeni simulaciji

4.3 Rezultati

V tem koraku bomo naredili napovedne modele z algoritmi strojnega učenja in potem glede na napoved ustavili prikaze na uporabnika v simulatorju. Za izdelavo napovednih modelov bomo uporabili odprtokodno programsko knjižnico Weka, ki vsebuje vse pomembnejše algoritme za regresijo v strojnem učenju.

Da bo vse skupaj delovalo kot mora, bomo naš simulator razširili s proceduro, ki bo za vhod vzela učno množico, na njej zgradila napovedni model in ga nato uporabila na istih testnih podatkih, kot smo jih uporabili za izračun K pri povprečni vrednosti potrjenih prikazov.

Iz knjižnice Weka smo za naše potrebe izbrali naslednje algoritme strojnega učenja: odločitveno drevo (M5P), linearna regresija, boosted regresija, *bagging*, metoda k najbližjih sosedov, nevronska mreža, metoda podpornih

vektorjev (SVM) [6]. Pri metodi *bagging* smo zgradili ansambel modelov iz hitro se učečega odločitvenega drevesa *REPTree*, ki za izgradnjo uporablja informacijski prispevek (angl. information gain). Pri boosted regresiji smo uporabili *DecisionStump*, ki je preprosto enonivojsko odločitveno drevo z zmožnostjo odločanja samo na podlagi enega atributa. Vsi algoritmi se lahko uporabljajo kot regresorji.

Algoritme bomo uporabili s privzetimi nastavitvami. Pri metodi k najbližjih sosedov to pomeni, da je k enak 1, medtem ko je pri nevronski mreži privzeto število skritih plasti pol manjše kot število atributov.

4.3.1 Ocenjevanje kakovosti atributov

Preden smo začeli z izdelavo napovednih modelov, smo ocenili, kako dobri so naši atributi. Za ocenjevanje kakovosti atributov smo izbrali metodo ReliefF [4], ker se običajno izkaže za najboljšo.

Ocena	Št. atributa	Ime atributa
0.0002566	2	product
0.00008934	8	clicked_campaign_before
-0.00000449	3	targeting
-0.00008286	4	ctr
-0.00055293	7	clicked_before
-0.00177665	1	ad_format
-0.01271635	5	category_primary
-0.01537212	6	category_secondary

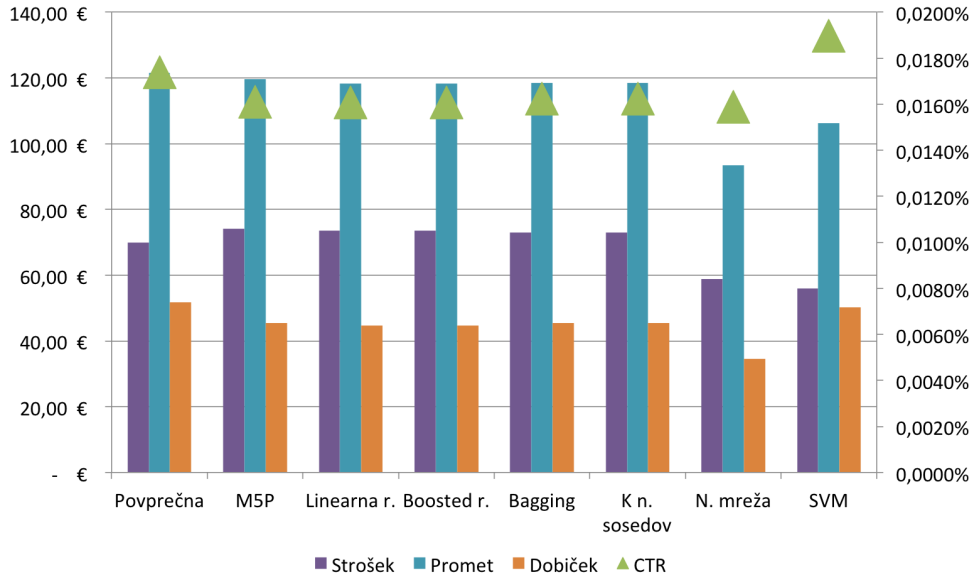
Tabela 4.3: ReliefF ocenitev atributov

Kot je vidno iz Tabele 4.3, izstopata atribut, ki označuje produkt oglasa, *product*, in atribut, ki določa, ali je uporabnik že kdaj v preteklosti kliknil na trenutno oglasno akcijo, *clicked_campaign_before*. Tako smo za napovedni model in seveda tudi za simulacijo upoštevali samo ta dva atributa.

Omejitev	Prikazi	Kliki	Strošek	Promet	Dobiček	CTR	**K**
Povprečna	3496234	608	69,92€	121,60€	51,68€	0,0174%	**89,86**
M5P	3710927	598	74,22€	119,60€	45,38€	0,0161%	**73,13**
Linearna r.	3676787	591	73,54€	118,20€	44,66€	0,0161%	**71,79**
Boosted r.	3676787	591	73,54€	118,20€	44,66€	0,0161%	**71,79**
Bagging	3644605	592	72,89€	118,40€	45,51€	0,0162%	**73,92**
K n. sosedov	3645140	592	72,90€	118,40€	45,50€	0,0162%	**73,89**
N. mreža	2938329	467	58,77€	93,40€	34,63€	0,0159%	**55,04**
SVM	2799375	531	55,99€	106,20€	50,21€	0,0190%	**95,25**

Tabela 4.4: Primerjalna tabela rezultatov algoritmov pri dveh najboljših atributih

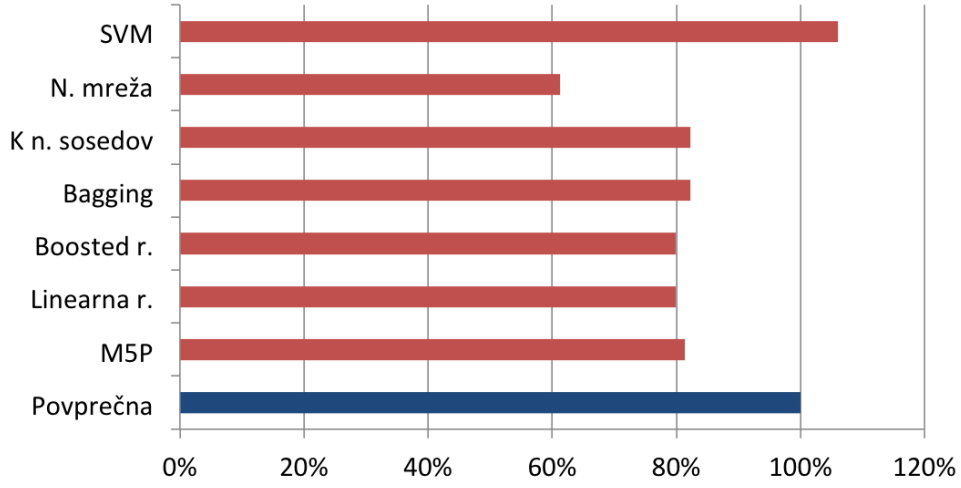
V seznamu rezultatov v Tabeli 4.4 je na vrhu rezultat povprečne vrednosti prikazov oglasov glede na klike. Algoritme strojnega učenja bomo tako primerjali z oceno K in videli, kateri algoritem deluje boljše od povprečja.



Slika 4.5: Razmerja med metrikami pri dveh najboljših atributih

Slika 4.5 predstavlja grafični pregled Tabele 4.4. Tukaj lepo vidimo, da je

CTR pri SVM presegel povprečje.



Slika 4.6: Razmerja K pri uporabi dveh najboljših atributov

V Sliki 4.6 lahko jasno vidimo velikosti ocene K in tudi, da je K pri algoritmu SVM za malenkost boljši od povprečne vrednosti števila potrjenih prikazov na uporabnika. Ostali algoritmi so precej slabši.

4.3.2 Posamezni atributi

Atributov ni veliko, zato smo se odločili preveriti, kako deluje vsak atribut posamezno. Tako bomo izvedeli, ali lahko algoritmi poiščejo uporaben vzorec na katerem od atributov. Za začetek smo vzeli atributa, ki sta se izkazala kot najboljša pri ocenitvi kakovosti, in jih primerjali s povprečjem.

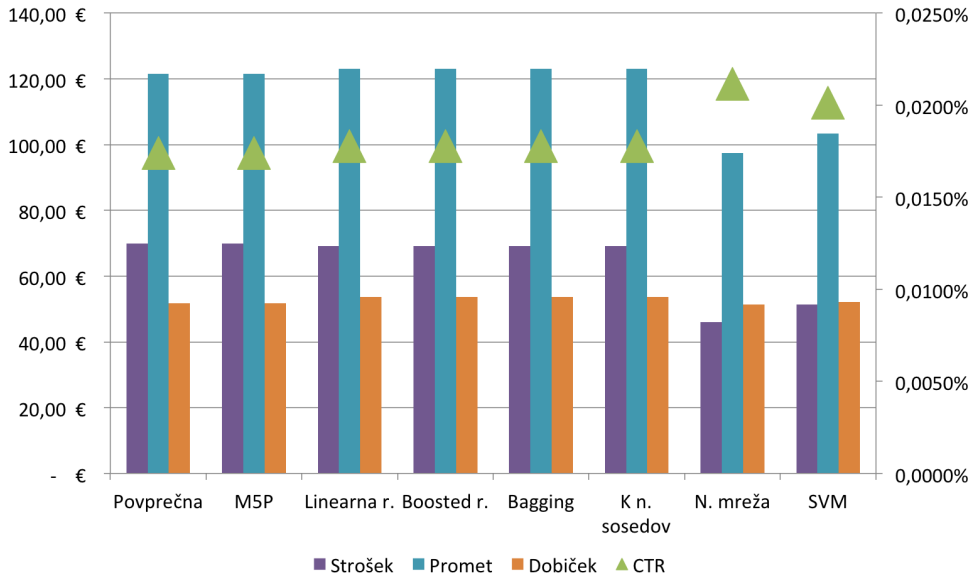
Atribut *product* se ni izkazal za pretirano uporabnega, kljub temu, da je bil glede na oceno *ReliefF* ocenjen kot najboljši (Tabela 4.3). Naslednji atribut *clicked_campaign_before* pa je bil že boljši kot povprečna vrednost števila prikazov oglasa. Po še nekaj neuspelih poskusih z različnimi atributi smo na koncu izluščili najboljšega. To je atribut *clicked_before*, v katerem je

zapisano, ali je uporabnik že kdaj v preteklosti kliknil na oglas. Rezultati poskusa so zapisani v Tabeli 4.5.

Omejitev	Prikazi	Kliki	Strošek	Promet	Dobiček	CTR	**K**
Povprečna	3496234	608	69,92€	121,60€	51,68€	0,0174%	**89,86**
M5P	3496234	608	69,92€	121,60€	51,68€	0,0174%	**89,86**
Linearna r.	3461713	615	69,23€	123,00€	53,77€	0,0178%	**95,52**
Boosted r.	3461713	615	69,23€	123,00€	53,77€	0,0178%	**95,52**
Bagging	3461713	615	69,23€	123,00€	53,77€	0,0178%	**95,52**
K n. sosedov	3461713	615	69,23€	123,00€	53,77€	0,0178%	**95,52**
N. mreža	2300233	487	46,00€	97,40€	51,40€	0,0212%	**108,81**
SVM	2566812	517	51,34€	103,40€	52,06€	0,0201%	**104,87**

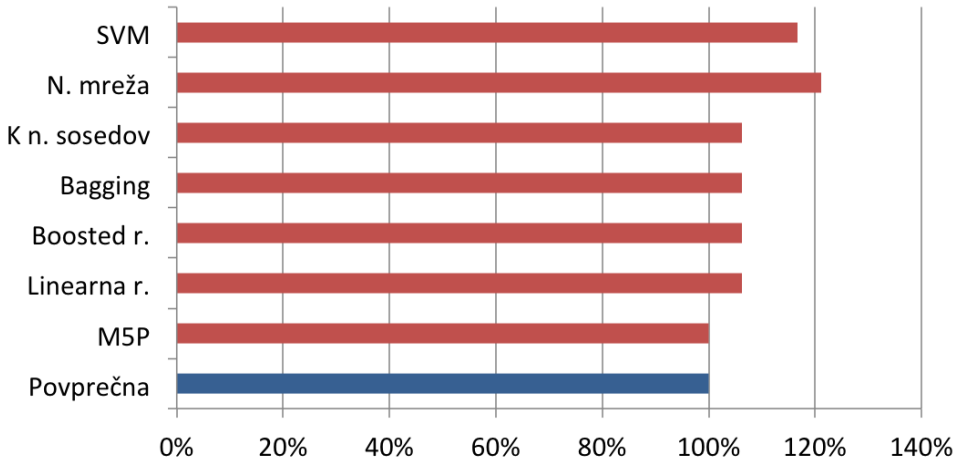
Tabela 4.5: Primerjalna tabela rezultatov algoritmov pri atributu clicked_before

Razlika je še bolj razvidna na Sliki 4.5, kjer lahko vidimo višje postavljen trikotnik, ki predstavlja metriko CTR in je tokrat postavljen nad 0,02%.



Slika 4.7: Razmerja med metrikami pri atributu clicked_before

Ker je CTR neprimerno večji pri malce manjšem dobičku, je K precej boljši od povprečja (Slika 4.8).



Slika 4.8: Razmerje K pri atributu clicked_before

Zanimivo je rezultat pri *nevronski mreži* za več kot za četrtino boljši od povprečja. To pomeni, da nam je uspelo premagati povprečno vrednost, in s tem smo prišli do uporabnih rezultatov.

Za atribut *clicked_before* smo preverili tudi relativno točnost v učni množici. Narejeno je bilo desetkratno prečno preverjanje z namenom izračuna korenjene relativne srednje kvadratne napake (RRMSE). Rezultat poskusa je bila RRMSE 1,000315. Čeprav višina napake napeljuje na to, da je regresijski model neuporaben, mera RRMSE ne upošteva, da:

- se mora oglas prikazati vsaj enkrat, kljub temu, da je lahko napoved manjša,
- je napoved ob kliku lahko premajhna, ker se potem klik ne bo zgodil,
- imamo lahko tudi prikaze, ki se ne končajo s klikom, kjer je dobro, če je napoved majhna.

Teh pogojev korenjena relativna kvadratna napaka ne upošteva. RRMSE upošteva samo razdaljo napovedanega od povprečja, ki pa je za nas lahko

kvečjemu delni pokazatelj uspeha.

Ker imamo atribut *clicked_before*, ki ima samo dve dimenziji, lahko pogledamo v napovedi nevronske mreže in preverimo, kaj se dejansko dogaja in kakšen je vzorec, ki smo ga našli. Algoritem po naslednjem ključu napoveduje potrebno število prikazov:

Uporabnik še nikoli ni kliknil na oglas: -1,202
Uporabnik je že kliknil na oglas: 7,746

Na tem mestu je treba poudariti, da naš simulator vedno streže oglas vsaj enkrat, kljub temu da je napoved -1. To pomeni, da ko za uporabnika vemo, da je nekoč že kliknil na oglas, mu pokažemo oglas osemkrat (povprečno število), ko tega ne vemo, mu pokažemo oglas samo enkrat. To je tudi smiselno, saj po ComScore raziskavi iz leta 2009 8% ljudi na svetu naredi 85% klikov [21].

4.3.3 Razdelitev problema na dva razreda

Rezultati zadnjega postopka so nas napeljali na idejo, da bi napovedovanje števila prikazov lahko razdelili v dve stopnji. Prva stopnja bi bila napoved možnega klika, druga pa napovedovanje števila prikazov pred klikom. V namen izdelave napovednega modela za napovedovanje možnega klika smo iz učnih primerov vzeli sto tisoč primerov klikov, ki smo jim dodali še sto tisoč primerov, ko se klik ni zgodil. Tako smo dobili uravnoteženo množico dvesto tisoč učnih primerov, na katerih bomo naredili napovedni model za prvo stopnjo napovedovanja.

Ker gre za napovedovanje razreda, kjer se klik *je* ali *ni* zgodil, smo za napovedovanje morali uporabiti klasifikacijske algoritme strojnega učenja. Za napovedovanje smo preizkusili vse algoritme strojnega učenja, ki smo jih uporabili že pri regresiji in ki znajo reševati klasifikacijske probleme. Namesto M5P, linearne regresije in metode *boosted regression*, ki so izključno regre-

sijski smo dodali naivni Bayes-ov klasifikator in metodo naključnih gozdov (angl. random forest) s sto odločitvenimi drevesi.

Atribute smo ponovno ocenili s postopkom *ReliefF*, a tokrat so bili vsi atributi spoznani kot koristni kot lahko vidimo v Tabeli 4.6. Vidno pa je tudi, da so atributi, vezani na uporabnikovo zgodovino, ocenjeni zelo dobro (*clicked_campaign_before* in *clicked_before*).

Ocena	Št. atributa	Ime atributa
0.027373	1	ad_format
0.020137	8	clicked_campaign_before
0.005545	7	clicked_before
0.003775	2	product
0.002461	6	category_secondary
0.001744	5	category_primary
0.000381	4	ctr
0.000284	3	targeting

Tabela 4.6: ReliefF ocenitev atributov za napovedovanje klikov

Rezultati klasifikacijske točnosti izbranih algoritmov pri 10-kratnem prečnem preverjanju so zbrani v Tabeli 4.7.

Algoritem	Klasifikacijska točnost	Senzitivnost	Specifičnost
Naivni Bayes	68,84%	0,696	0,682
Naključni gozdovi	73,94%	0,722	0,759
Bagging	73,90%	0,722	0,759
K najbližjih sosedov	73.94%	0,723	0,758
SVM	73,77%	0,726	0,751

Tabela 4.7: Klasifikacijska točnost algoritmov

Iz klasifikacijske točnosti lahko razberemo, da je najboljša metoda naključnih gozdov in k najbližjih sosedov, ki imata klasifikacijsko točnost 73,94%.

Senzitivnost je definirana kot odstotek pravilno klasificiranih pozitivnih primerov (klik se je zgodil), medtem ko je specifičnost odstotek pravilno klasificiranih negativnih primerov (klik se ni zgodil).

```
Senzitivnost = RP / (RP + LN)
Specifičnost = RN / (RN + LP)
```

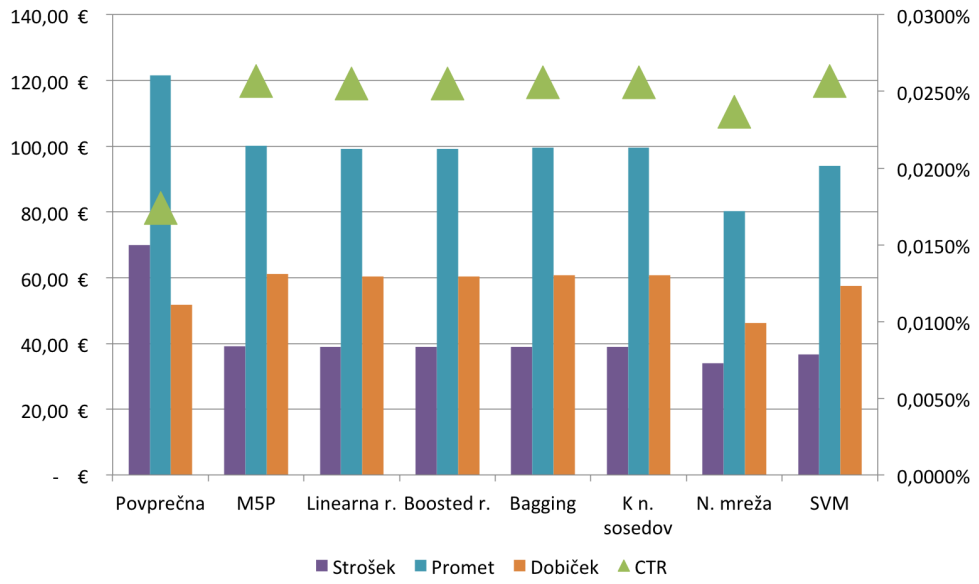
Resnično pozitivni (RP): točna napoved, da bo klik
Lažno pozitivni (LP): napačna napoved, da bo klik (klik se ni zgodil)
Resnično negativni (RN): točna napoved, da ne bo klika
Lažno negativni (LN): napačna napoved, da ne bo klika (klik se je zgodil)

Za nas je pomembna visoka senzitivnost, saj bi imeli radi čim manj napačno napovedanih negativnih napovedi, ker bi radi ujeli čim več klikov, a imamo lahko več napačnih pozitivnih napovedi (zmerna specifičnost). Po tem izračunu je za nas najboljša metoda SVM.

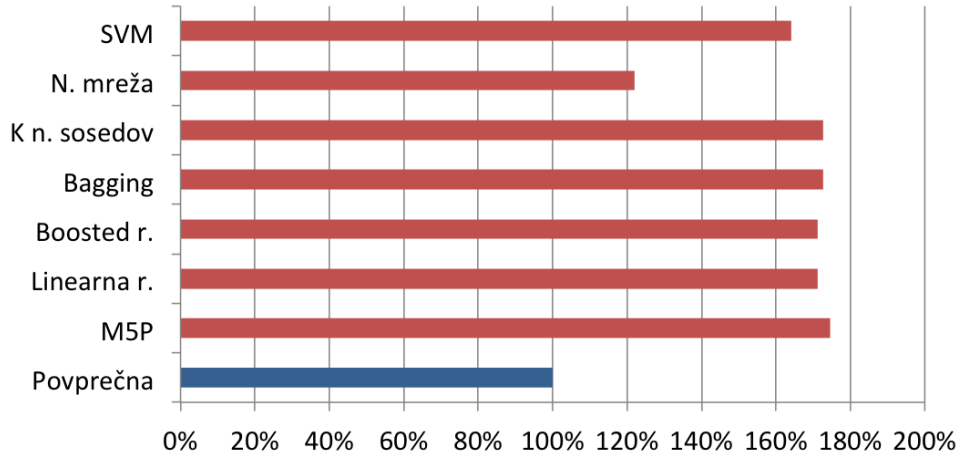
Napoved klasifikacijskih algoritmov smo uporabili kot prvi del v simulaciji, kjer če:

- je bil napovedan klik, smo uporabili napoved za število prikazov z dvema najboljšima atributoma po *ReliefF*,
- ni bil napovedan klik, potem smo uporabniku pokazali oglas samo enkrat.

Skozi simulator streženja oglasov smo spustili vse postopkovne kombinacije in za vsako izračunali K. Kot najboljša kombinacija sta se izkazala naivni Bayes kot prvostopenjski algoritem in odločitveno drevo M5P kot drugostopenjski.



Slika 4.9: Razmerja med metrikami pri kombiniranem pristopu z naivnim Bayes-om



Slika 4.10: Primerjava koeficienta uspešnosti pri kombiniranem pristopu z naivnim Bayes-om

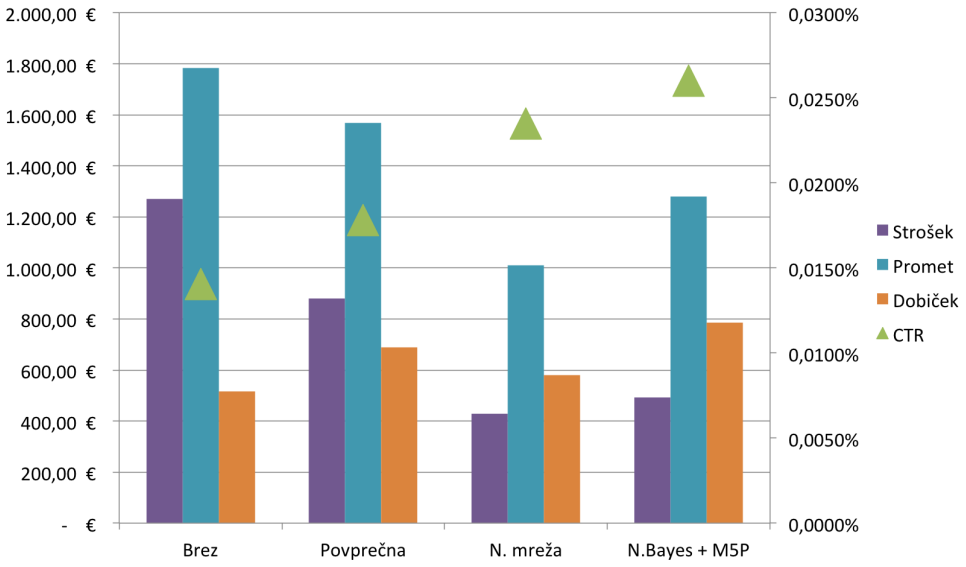
V rezultatih na Sliki 4.9 in 4.10 so prikazani rezultati posameznih drugostopenjskih postopkov pri istem prvostopenjskem postopku (naivni Bayes).

Kombinirani pristop naivnega Bayes-a in M5P je boljši kot prejšnji najboljši rezultat z nevronsko mrežo.

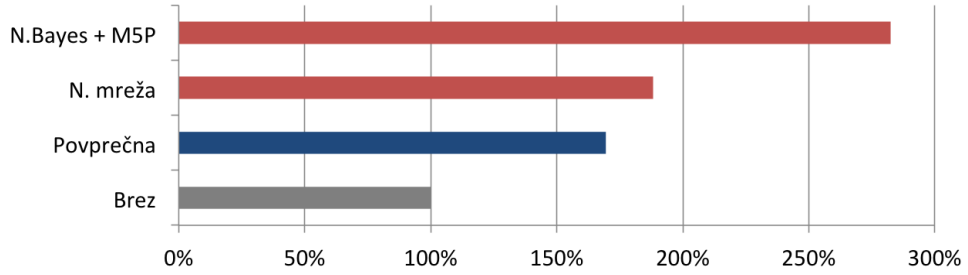
Da pa bi še dodatno preverili uspešnost postopka, smo najboljšo kombinacijo preizkusili še na podmnožici novembrskih podatkov, ki je zajemal tristo tisoč naključno izbranih uporabnikov. To je povečalo testno množico za več kot desetkrat. Dobljeni rezultati so zajeti v Tabeli 4.8.

Omejitev	Prikazi	Kliki	Dobiček	CTR	**K**
Brez	63476600	8923	515,07€	0,0141%	**724,04**
Povprečna	44030827	7846	688,58€	0,0178%	**1.227,01**
N. mreža	21489603	5050	580,21€	0,0235%	**1.363,47**
N.Bayes + M5P	24597800	6394	786,84€	0,0260%	**2.045,34**

Tabela 4.8: Primerjalna tabela rezultatov pri večji testni množici



Slika 4.11: Razmerja med metrikami pri desetkrat večji testni množici



Slika 4.12: Razmerja K pri desetkrat večji testni množici

Kot lahko vidimo na Slikah 4.11 in 4.12, so dobljeni rezultati tudi v tem primeru boljši od povprečja. Tako lahko trdimo, da je za naše podatke to najboljši postopek optimizacije.

Poglavje 5

Sklepne ugotovitve

V okviru diplomske naloge smo preizkusili algoritme strojnega učenja na podatkih srbske oglasne mreže. Izkazalo se je, da se je treba lotiti problema dvostopenjsko, kjer najprej napovedujemo verjetnost klika, nato pa optimiziramo prikaze na delu podatkov, kjer je verjetnost klika večja. Prav tako se je izkazalo, da so za uspešno optimiziranje števila prikazov oglasov pomembni atributi, ki so vezani na uporabnikove zgodovinske akcije. Atributi, ki so vezani na oglase same, pa niso tako pomembni pri iskanju vzorcev, kar je zelo uporabna ugotovitev. Intuitivno bi mislili, da če ima, generalno gledano, oglas večji CTR, potem lahko bolje napovemo, koliko prikazov bo potrebnih za klik (uporabniki kliknejo prej?), vendar temu ni tako. To tudi pomeni, da postopek ne bo deloval dobro na čisto novih podatkih, kjer nimamo nič poznanih uporabnikov.

Implementacija celotnega procesa je bila uspešna, ker smo prišli do zelo zadovoljivih rezultatov. Vseeno pa je bil proces zelo dolgotrajen. Ker je bilo podatkov veliko, je postopek predpriprave vzel veliko časa. Na začetku smo imeli za tri mesece zgodovinskih podatkov in en mesec podatkov, ki so bili uporabljeni navzkrižno za učenje in preverjanje. Kasneje v procesu smo zamenjali način ocenjevanja uspešnosti in s tem se je povečala tudi količina podatkov, za kar je bilo treba določen del podatkov še enkrat predpripra-

viti.

Za nadaljnje raziskovanje bi se lahko še bolj osredotočili na podatke o uporabniku. Upoštevali bi, na katere tipe oglasov je že kliknil, prav tako pa bi lahko izdelali njegov profil glede na spletne strani, ki jih je obiskal v preteklosti. S podobnimi optimizacijami bi verjetno prišli do še boljših rezultatov. Za dodatno potrditev rezultatov bi lahko postopek preizkusili še s podatki kakšne druge oglasne mreže.

Literatura

[1] Breiman, L. Bagging predictors. "Machine Learning", 24(2):123–140, 1996.

[2] Schapire, R. E. "A brief introduction to boosting. In IJCAI", pages 1401–1406, 1999.

[3] Bernard Ženko "Izboljšave metode skladanja klasifikatorjev" Dostopno na: `http://kt.ijs.si/bernard/papers/03-zen-msc.pdf`, 13.2.2014.

[4] Igor Kononenko "Estimating Attributes: Analysis and Extensions of RELIEF" Dostopno na: `http://sci2s.ugr.es/keel/pdf/algorithm/congreso/FS-Relief-F.pdf`, 5.2.2014.

[5] Igor Kononenko, Marko Robnik Šikonja "Inteligentni sistemi". Ljubljana, Založba FE in FRI, 2010.

[6] Ian H. Witten, Eibe Frank, Mark A. Hall "Data Mining: Practical Machine Learning Tools and Techniques, Third Edition". Morgan Kaufmann, 2011.

[7] Flavio Sales Truzzi, Valdinei Freire da Silva, Anna Helena Reali Costa, Fabio Gagliardi Cozman "Markov Decision Processes for Ad Network Optimization" Dostopno na: `http://sites.poli.usp.br/p/fabio.cozman/publications/Article/truzzi-silva-costa-cozman-enia2012F.pdf`, 9.3.2014.

[8] Jason Kolb, Jeremy Kolb "The Big Data Revolution". Applied Data Labs Inc., 2013, stran 13.

[9] Jeffrey Dean, Sanjay Ghemawat "MapReduce: Simplied Data Processing on Large Clusters" Dostopno na: http://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf, 10.2.2014.

[10] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler "The Hadoop Distributed File System" Dostopno na: http://storageconference.org/2010/Papers/MSST/Shvachko.pdf, 15.2.2014.

[11] Kuang-Chih Lee, Ali Jalali, Ali Dasdan "Real Time Bid Optimization with Smooth Budget Delivery in Online Advertising" Dostopno na: http://arxiv.org/pdf/1305.3011.pdf, 9.3.2014.

[12] Leo Breiman "Random forests". Machine Learning, 45:5-32, 2001.

[13] Martin Zinkevich "Optimal Online Frequency Capping Allocation using the Weight Approach" Dostopno na: http://martin.zinkevich.org/publications/weights.pdf, 9.3.2014.

[14] Niv Buchbinder, Moran Feldman, Arpita Ghosh, Joseph (Seffi) Naor "Frequency Capping in Online Advertising" Dostopno na: http://theory.epfl.ch/moranfe/Publications/WADS2011.pdf, 15.2.2014.

[15] Peter Harrington, "Machine Learning in Action". Manning Publications, 2012, pogl. 1.

[16] Tom White. "Hadoop: The Definitive Guide". Yahoo Press, 2012.

[17] "Online Advertising" Dostopno na: http://en.wikipedia.org/wiki/Online_advertising, 15.2.2014.

[18] "Geolocation Software" Dostopno na: http://en.wikipedia.org/wiki/Geolocation_software, 15.2.2014.

[19] "Apache Hadoop" Dostopno na: `http://hadoop.apache.org/`, 15.2.2014.

[20] "Apache Hive" Dostopno na: `http://hive.apache.org/`, 15.2.2014.

[21] "comScore and Starcom USA Release Updated "Natural Born Clickers" Study Showing 50 Percent Drop in Number of U.S. Internet Users Who Click on Display Ads" Dostopno na: `http://www.comscore.com/Insights/Press_Releases/2009/10/comScore_and_Starcom_USA_Release_Updated_Natural_Born_Clickers_Study_Showing_50_Percent_Drop_in_Number_of_U.S._Internet_Users_Who_Click_on_Display_Ads`, 10.3.2014.

[22] "Clojure" Dostopno na: `http://clojure.org/`, 15.2.2014.

[23] "Python" Dostopno na: `http://www.python.org/`, 15.2.2014.

[24] "Weka", Dostopno na: `http://www.cs.waikato.ac.nz/ml/weka/`, 15.2.2014.