UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO
FAKULTETA ZA MATEMATIKO IN FIZIKO

Juš Hladnik

Napovedovanje stopnje interakcij z oglasi

DIPLOMSKO DELO

INTERDISCIPLINARNI UNIVERZITETNI
ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN MATEMATIKA

MENTOR: prof. dr. Igor Kononenko
SOMENTOR: dr. Domen Košir

Ljubljana, 2020

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V spletnem oglaševanju je stopnja interakcij z oglasom (relativno število klikov na oglas) osnovna metrika za merjenje uspešnosti oglaševalskih kampanj. Na podlagi zgodovinskih podatkov o stopnjah interakcij oglasov na spletnih straneh z metodami strojnega učenja preizkusite različne modele za napovedovanje stopnje interakcij za nove oglase, ki v preteklosti še niso bili prikazani na nobeni spletni strani in na tak način obravnavajte problem hladnega zagona. Ocenite pomembnost posameznih karakteristik oglasov za uspešnost napovedi in tako identificirajte atribute, ki najbolj vplivajo na stopnjo interakcije oglasov.

Iskreno se zahvaljujem mentorju prof. dr. Igorju Kononenku in somentorju dr. Domnu Koširju za hitro odzivnost in napotke ter ideje pri izdelavi naloge. Hvala sošolcem za pomoč pri študijskih obveznostih. Hvala tudi družini, ki me je tekom študija podpirala in vzpodbujala.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
CTR	click-through rate	delež klikov (stopnja interakcij)
kNN	k-nearest neighbors	k-najbližjih sosedov
RF	random forest	naključni gozd
MF	matrix factorization	matrična faktorizacija
RMSE	root mean squared error	koren srednje kvadratne napake
RRMSE	relative root mean squared error	relativni koren srednje kvadratne napake
RAE	relative absolute error	relativna absolutna napaka

Povzetek

Naslov: Napovedovanje stopnje interakcij z oglasi

Avtor: Juš Hladnik

V spletnem oglaševanju je stopnja interakcij z oglasom (angl. *click-through rate* oz. CTR) ena izmed bolj pomembnih metrik o uspešnosti posameznega oglasa. V diplomskem delu se ukvarjamo z napovedjo CTR oglasov na posameznih spletnih straneh za novo kreirane oglase, ki v preteklosti še niso bili prikazani, in z ocenjevanjem atributov oglasov. Podatke o oglasih in klikih na oglase podjetja Celtra d.o.o. pripravimo na več različnih načinov in na njih preizkusimo več regresijskih metod strojnega učenja - naključni gozd, k-najbližjih sosedov in matrično faktorizacijo. Atribute ocenjujemo z RReliefF-om in razliko variance. Ugotovimo, da na CTR najbolj vpliva spletna stran in velikost oglasa. Na podlagi napovedi stopenj interakcij se lahko podjetje odloči, na katere spletne strani želi objaviti oglas in s tem povečati število klikov nanj.

Ključne besede: naključni gozd, k-najbližjih sosedov, matrična faktorizacija, RReliefF, spletno oglaševanje.

Abstract

Title: Prediction of degree of interaction with creatives

Author: Juš Hladnik

Click-through rate (CTR) is one of the most important measurements in online advertising that tells us how successful a certain ad is. In this thesis we work on CTR prediction of ads on individual websites for newly created ads that have not been published on any website in the past. We also work on evaluation of attributes that describe these ads. We use data about ads and ad clicks from Celtra d.o.o. and process it in different ways. Then we apply different machine learning methods - random forest, k-nearest neighbors, and matrix factorization. For attribute evaluation we use RReliefF and the difference of variance. We find out that the website, on which an ad is published, and the size of an ad influence CTR the most. Based on our predictions of CTR, a company can decide on which websites they should publish the ad thus enlarge the number of clicks on the ad.

Keywords: random forest, k-nearest neighbors, matrix factorization, RReliefF, online advertising.

Poglavje 1

Uvod

Oglaševanje se je skozi zgodovino razvijalo, na kar so vplivali faktorji, kot so ciljna množica, stil oglaševanja, namen, geografska regija in medij. Med različne medije oglaševanja lahko uvrstimo vse od oglasnih panojev, letakov, poslikav na stavbah, do medijev, ki so se pojavljali z razvojem novih tehnologij skozi zgodovino. To so časopisi, radio, televizija, s pojavitvijo interneta pa tudi spletno oglaševanje. Skoraj polovica sredstev za oglaševanje se porabi za oglaševanje na internetu, napovedi pa pravijo, da se bo v prihodnjih letih ta delež še povečeval [1]. Razlog za usmerjanje v digitalno oglaševanje je predvsem v možnosti zajema in analize velike količine podatkov. Podjetja tako lahko natančno ciljajo na uporabnike, ki bi jih njihove storitve ali produkti lahko zanimali. V spletnem oglaševanju je stopnja interakcij z oglasom osnovna metrika za merjenje uspešnosti oglaševalskih kampanj. Stopnja interakcij z oglasom pomeni relativno število klikov na oglas oziroma delež števila klikov na oglas glede na število prikazov oglasa. Nanjo, poleg same vsebine oglasa, najbolj vplivata vsebina spletne strani oz. aplikacije, v kateri se oglas prikazuje in primernost oglasa za izbrano ciljno skupino. Podjetja želijo doseči čim večjo stopnjo interakcij, saj s tem uporabnika preusmerijo na njihovo spletno stran in k nakupu njihovih izdelkov ali storitev.

1.1 Motivacija

Pri strategiji zakupa medijskega prostora se oglaševalci ali njihovi posredniki pogosto zanašajo na domensko znanje, ki pa ima zaradi hitrih sprememb trendov popularnosti spletnih aplikacij omejeno trajanje. Napovedovanje stopnje interakcij z oglasom je lahko podlaga za informirane odločitve pri zakupu medijev in s tem bolj optimalno trošenje oglaševalskega denarja. Za učenje na podlagi velike količine zgodovinskih podatkov je zelo primerna matrična faktorizacija. Pri tem se algoritem na podlagi interakcij na posameznih kombinacijah med oglasi in spletnimi stranmi, ki so se zgodile v preteklosti, nauči, katere spletne strani so si med seboj podobne in kateri oglasi so si med seboj podobni. To pa pomeni, da lahko za oglas, ki na določeni strani še ni bil prikazan, napovemo, kakšno stopnjo interakcij bo dosegel. Na podlagi teh napovedi pa se lahko odločimo, na katere spletne strani bomo postavili naš oglas, da bo nanj kliknilo čim več ljudi. Problem pa se pojavi pri novih oglasih, ki v preteklosti še niso bili prikazani na nobeni spletni strani ali pa so bili prikazani na zelo malo spletnih straneh. Ta problem imenujemo problem hladnega zagona (angl. *cold start*). V tem primeru se moramo poslužiti drugih metod, ki jih bomo preizkusili v tem diplomskem delu.

1.2 Cilji

Cilji naloge so priprava in obdelava danih podatkov tako, da bomo z metodami strojnega učenja dosegli čim boljše rezultate ter izgradnja in ocena točnosti modela za napovedovanje stopnje interakcij z novimi oglasi, ki v preteklosti še niso bili prikazani na nobeni spletni strani. Točnost modela bomo ocenjevali na podlagi točnosti napovedi stopnje interakcij (angl. *click-through rate* oz. CTR) posameznih oglasov na določenih spletnih straneh. Poiskali bomo značilke, ki najbolj vplivajo na CTR oglasov in jih ocenili glede na pomembnost.

V 2. poglavju naštejemo in opišemo metode strojnega učenja, ki smo jih uporabili za napovedovanje CTR in ocenjevanje atributov. V 3. poglavju opišemo obliko, v kateri smo dobili podatke, povemo, na kakšne načine smo podatke pro-

cesirali in kako smo preverjali točnost napovedi. 4. poglavje opiše izvedene eksperimente ter predstavi dosežene rezultate. V 5. poglavju povzamemo diplomsko delo in opišemo ideje za delo v prihodnje.

Poglavje 2

Pregled metod

2.1 Naključni gozdovi

Za napovedovanje lahko uporabljamo več posameznih napovednih modelov, njihove napovedi pa lahko kombiniramo na več načinov: glasovanje, uteženo glasovanje, kombiniranje po metodi naivnega Bayesa in drugo. Takemu načinu povezovanja preprostih klasifikatorjev pravimo ansambelske metode (angl. *Ensemble methods*). V ansambelske metode spada tudi metoda *bagging* (okrajšava za *bootstrap aggregating*) [11]. Pri baggingu iz učne množice, ki ima *n* elementov, *n* krat naključno izberemo primer iz učne množice z vračanjem. Tako dobimo novo učno množico z *n* primeri, ki pa ne vsebuje nujno vseh primerov iz prvotne učne množice, prav tako pa se lahko nekateri primeri večkrat ponovijo. Primerov iz prvotne učne množice je v novi generirani učni množici povprečno 36.8%. Postopek večkrat ponovimo, da dobimo več učnih množic in nato na vsaki zgradimo napovedni model. Pri napovedovanju novih vrednosti pa napovedi posameznih modelov kombiniramo med seboj.

Naključni gozdovi (angl. *random forests*) razširjajo metodo bagginga tako, da na vsaki generirani učni množici zgradijo odločitveno (ali regresijsko) drevo. Poleg tega pa pri gradnji drevesa v vsakem vozlišču izberejo majhno naključno podmnožico atributov, izmed katerih izberejo najbolj optimalnega. Breiman predlaga, da je velikost naključne podmnožice enaka $\log_2 M + 1$, kjer je M število vseh atri-

butov [3]. Odločitvena drevesa so nestabilen algoritem z visoko varianco, metoda naključnih gozdov pa je robustna in zmanjšuje varianco posameznih odločitvenih dreves. Število dreves, ki jih zgradimo v naključnem gozdu, je ponavadi 100 ali več.

2.2 Najbližji sosedi

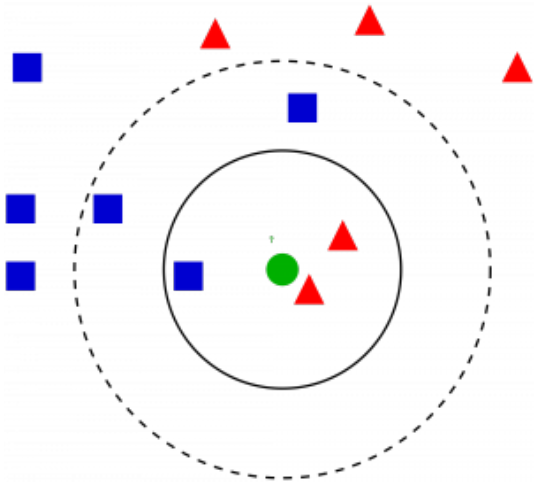
Pri algoritmu *k-najbližjih sosedov*, na kratko kNN (angl. *k-nearest neighbors*) učenje predstavlja le shranitev učne množice. Taki vrsti učenja pravimo *leno učenje*. Pri klasifikaciji ali napovedi vrednosti novega primera ga moramo primerjati z vsemi primeri iz učne množice, kar pa pomeni, da je tu uporabljena večina procesiranja in s tem večja časovna zahtevnost, kot pri drugih metodah strojnega učenja. Ko želimo napovedati vrednost r_x novega primera u_x, poiščemo k najbližjih primerov $u_1, ..., u_k$ iz učne množice, ki so mu najbolj podobni po določeni meri razdalje, in pri regresiji napovemo povprečno vrednost odvisne spremenljivke za *k* najbližjih sosedov [11]

$$r_x = \frac{1}{k} \sum_{i=1}^{k} r^{(i)} \tag{2.1}$$

k je parameter, ki vpliva na napovedno točnost modela in ga običajno določimo eksperimentalno. Majhen *k* pomeni veliko prilagajanje učni množici, v kolikor pa je v njej veliko napak, bomo dosegli slabšo napovedno točnost, ker napačni učni primeri pridejo do večjega izraza. S povečevanjem parametra *k* povprečimo vrednosti najbližjih sosedov, s čimer se zmanjša verjetnost, da so vsi sosedi napačni. Vendar pri uporabi prevelikega parametra *k* upoštevamo tudi tiste učne primere, ki novemu primeru niso več dovolj podobni, kot je prikazano na sliki 2.1.

Poleg parametra *k* vpliva na napoved tudi metrika za merjenje razdalj med novim in učnimi primeri. Najpogosteje uporabljeni metriki sta manhattanska in evklidska razdalja. Manhattanska razdalja med dvema primeroma u_k in u_l, z a zveznimi atributi, je podana z:

$$d_m(u_k, u_l) = \sum_{i=1}^{a} \left| u_k^{(i)} - u_l^{(i)} \right| \tag{2.2}$$



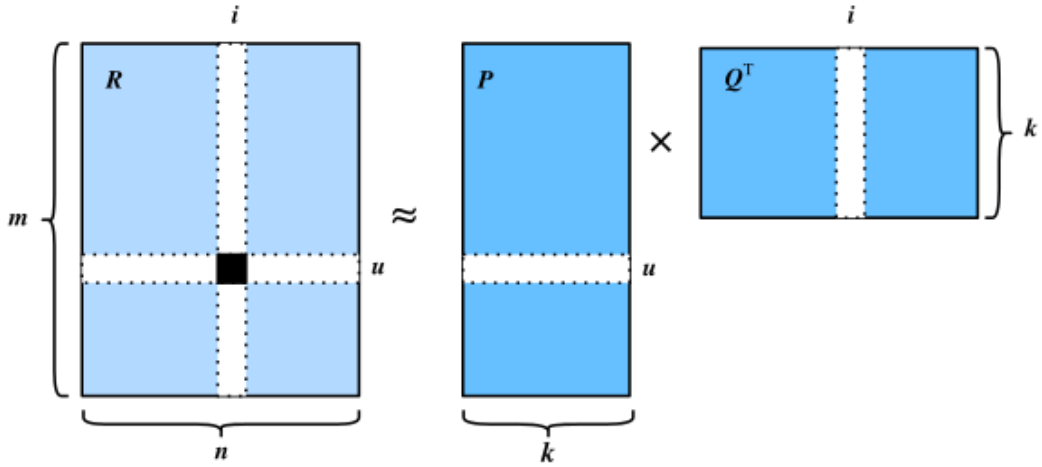
Slika 2.1: Povečenje parametra k iz k=3 na k=5 (črtkana črta). Vir: [9]

evklidska razdalja pa z:

$$d_e(u_k, u_l) = \sqrt{\sum_{i=1}^{a} (u_k^{(i)} - u_l^{(i)})^2} \tag{2.3}$$

2.3 Matrična faktorizacija

Učne podatke lahko predstavimo kot matriko R, katere elementi so stopnje interakcij oziroma *click-through rate* r_{ui} oglasa *u* na strani *i*. Ker je oglasov *N*, strani pa *M*, je matrika R oblike $M \times N$. Tipično sta *M* in *N* zelo velika. Vsak posamezen oglas u pa se oglašuje le na nekaj spletnih straneh in ne na vseh M straneh, zato je v posameznem stolpcu zelo malo CTR-jev. Prav tako pa se na posamezni strani i oglašuje le nekaj, veliko manj kot N, oglasov in posamezna vrstica v matriki R vsebuje le nekaj vrednosti CTR-jev, vse to pa prispeva k temu, da je matrika R redka (angl. *sparse matrix*). Matriko $R_{M\times N}$ želimo predstaviti z matrikama $P_{M\times K}$ in $Q_{K\times N}$ tako, da velja $R \approx PQ$ in da je $K \ll M, N$. Želimo, da je produkt matrik P in Q čim bližje matriki R, oziroma tistim elementom, ki se nahajajo v matriki R. Poleg tega pa dobimo še ocene za elemente, ki se ne nahajajo v matriki R. Matriki P in Q predstavljata latentna prostora oglasov in spletnih



Slika 2.2: Produkt latentnih profilov. Vir: [14]

strani, K pa predstavlja število latentnih faktorjev, ki korelirajo z nam znanimi lastnostmi strani in oglasov, kar pride prav pri razlagi matrične faktorizacije [8].

Latentni profil oglasa u označimo z vektorjem $p_u \in \mathbb{R}^k$, ki je element matrike P. Latentni profil strani i pa zapišimo z vektorjem $q_i \in \mathbb{R}^k$, ki je element matrike Q. Skalarni produkt latentnih profilov $q_i^T p_u$ (kot je prikazano na sliki 2.2) nam da približek za stopnjo interakcije oglasa u na strani i, ki je označena z r_{ui} [12]. Za oceno stopnje interakcije $\hat{r}_{ui}$ torej velja

$$\hat{r}_{ui} = q_i^T p_u. \tag{2.4}$$

Želimo, da je ta približek čim boljši, oziroma da se čim manj razlikuje od dejanske vrednosti. Definiramo kvadrat napake za stopnjo interakcije oglasa u na strani i:

$$e_{ui}^2 = (\hat{r}_{ui} - r_{ui})^2. \tag{2.5}$$

Za iskanje vrednosti elementov matrik P in Q uporabimo stohastični gradientni sestop. Matriki P in Q na začetku nastavimo na naključne majhne vrednosti in nato za vsak element r_{ui} v učni množici popravimo latentne profile oglasa u in strani i v matrikah P in Q. To naredimo tako, da za element r_{ui} izračunamo

gradient napake (enačbo 2.5 smo pomnožili z $\frac{1}{2}$, ki se je pri odvajanju okrajšala):

$$\begin{aligned} \frac{\partial e_{ui}^2}{\partial p_{uk}} &= -e_{ui}q_{ki} \\ \frac{\partial e_{ui}^2}{\partial q_{ki}} &= -e_{ui}p_{uk}. \end{aligned} \tag{2.6}$$

Nato ustrezno popravimo elemente P in Q v nasprotni smeri gradienta, kjer je α stopnja učenja:

$$\begin{aligned} p_{uk} &\leftarrow p_{uk} + \alpha e_{ui}q_{ki} \\ q_{ki} &\leftarrow q_{ki} + \alpha e_{ui}p_{uk}. \end{aligned} \tag{2.7}$$

Da bi preprečili preveliko prileganje učnim podatkom, lahko tudi tu uporabimo regularizacijo in v našo kriterijsko funkcijo vključimo kaznovanje, ki izhaja iz velikosti elementov matrik P in Q:

$$\tilde{e}_{ui}^2 = \frac{1}{2}(e_{ui}^2 + \eta p_u^T p_u + \eta q_i^T q_i). \tag{2.8}$$

Po upoštevanju na novo izračunanih gradientov zapišemo popravke vrednosti elementov matrik P in Q vekotrsko [15]:

$$\begin{aligned} \mathbf{p}_u &\leftarrow \mathbf{p}_u + \alpha(e_{ui}\mathbf{q}_i - \eta\mathbf{p}_u) \\ \mathbf{q}_i &\leftarrow \mathbf{q}_i + \alpha(e_{ui}\mathbf{p}_u - \eta\mathbf{q}_i). \end{aligned} \tag{2.9}$$

Postopek popravljanja elementov v matrikah P in Q ponavljamo, dokler nismo zadovoljni z majhnostjo napake v kriterijski funkciji.

2.3.1 Dodajanje novih oglasov v matrični razcep

Ko smo enkrat izračunali matrični razcep in nas zanima, kakšno stopnjo interakcij bi oglas u dosegel na strani i, na kateri se v preteklosti še ni oglaševal, lahko preprosto izračunamo oceno kot skalarni produkt $p_u^T q_i$. Problem pa se pojavi, ko se v naših podatkih pojavijo novi oglasi, ki jih še nismo upoštevali v prejšnjem matričnem razcepu. Rešitev, na katero najprej pomislimo, bi bila, da ponovimo algoritem na novi učni množici oziroma matriki R' in ponovno izračunamo P in Q, vendar je taka možnost pogosto časovno prezahtevna. Bolj primeren način je,

da obstoječi matriki P in Q le popravimo tako, da bosta upoštevali tudi nove učne primere. V matriki R vrstice predstavljajo posamezne oglase in ko želimo dodati nov oglas r v učno množico, le dodamo novo vrstico r matriki R in tako dobimo novo matriko R'. Kriterijsko funkcijo iz 2.8 spremenimo in definiramo novo kriterijsko funkcijo $L(R - PQ)$ tako, da seštejemo vse kvadrate napak iz naše učne množice, seštevek kvadratov napak, oziroma $L(R-PQ)$, pa želimo minimizirati. Po dodajanju novih oglasov v učno množico postane naš cilj minimizirati funkcijo $L(R' - P'Q)$. Ker je naša kriterijska funkcija seštevek posameznih kvadratov napak, jo lahko zapišemo kot:

$$L(R' - P'Q) = L(R - PQ) + L(r - r_pQ), \tag{2.10}$$

kjer je r na novo dodana vrstica v R in predstavlja nov oglas, r_p pa je nova vrstica v P in predstavlja latentni profil novega oglasa.

V prvotnem matričnem razcepu smo že izračunali matriki P in Q tako, da minimizirata $L(R - PQ)$, torej ju po dodajanju novih oglasov spreminjamo tako, da minimiziramo $L(r-r_pQ)$ [4], to pa počnemo kot v 2.9. V matriki P spreminjamo le nove vrstice, matriko Q pa spreminjamo celotno. Lahko pa se odločimo tudi, da latentega prostora strani, oziroma matrike Q, ne bomo spreminjali, torej drugo enačbo v 2.9 ignoriramo. S tem se obdrži globalna slika strani, ki je bila pridobljena v prvotnem matričnem razcepu in se ne prilagaja na nove oglase. V našem primeru se odločimo za to opcijo, saj vrednosti CTR-jev za oglase, ki jih dodamo v matrični razcep, niso resnične vrednosti ampak napovedi, ki jih dobimo z nekim drugim modelom. S tem ko bi upoštevali drugo enačbo v 2.9, bi se latentni prostor strani prilegal na napovedi, ki niso točne in se zato temu izognemo.

2.4 Ocenjevanje atributov

Poleg čim bolj točnega napovednega modela nas velikokrat zanima tudi pomembnost atributa, oziroma koliko posamezni atributi in koliko določene podmnožice atributov skupaj vplivajo na napoved. Z zmanjšanjem množice atributov dosežemo hitrejše učenje napovednega modela in izboljšamo natančnost napovedi z odstranitvijo nepomembnih atributov. Z rangiranjem atributov po pomembnosti pa se

poveča naše razumevanje napovednega modela. V našem primeru to pomeni, da lahko oglaševalske ekipe upoštevajo pomembnost atributov pri snovanju novih oglaševalskih kampanj in s prilagajanjem novih oglasov dosežejo večjo stopnjo interakcij.

2.4.1 Ocenjevanje atributov z razliko variance

Pri regresijskih problemih mero nečistoče nadomesti *varianca* odvisne spremenljivke [11]. Ocenjevanje z razliko variance je kratkovidna mera za ocenjevanje kvalitete atributov, saj predpostavlja apriorno in pogojno neodvisnost atributov. *Varianca* je definirana kot povprečni kvadrat napake, kjer je $\overline{r}$ povprečna vrednost zveznega razreda z *n* učnimi primeri:

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}(r^{(i)} - \overline{r})^2 \tag{2.11}$$

Pri ocenjevanju atributa A_i uporabljamo *pričakovano razliko variance*, kar pomeni, da za vsako vrednost atributa A_i izračunamo varianco učnih primerov in nato izračunamo uteženo povprečje varianc vrednosti atributa A_i. *Pričakovana razlika variance* je razlika med *varianco* odvisne spremenljivke s^2 in uteženim povprečjem varianc atributa A_i:

$$ds^2(A_i) = s^2 - \sum_{j=1}^{n_i}\left(p_j\frac{1}{n_j}\sum_{k=1}^{n_j}(r_j^{(k)} - \overline{r}_j)^2\right) \tag{2.12}$$

$r_j^{(k)}$ predstavlja vrednost *k*-tega primera, ki ima *j*-to vrednost atributa A_i, $\overline{r}_j$ pa je povprečna vrednost primerov z *j*-to vrednostjo atributa A_i. Večja kot je *pričakovana razlika variance*, bolj pomemben je atribut. Ocena predpostavlja, da je atribut diskreten, oziroma je treba zvezne atribute diskretizirati (ponavadi se ocenjujejo različne binarne variante zveznega atributa in se izbere najboljša binarna varianta). Poleg binarnih atributov smo v našem primeru imeli še dva nebinarna atributa, ki sta bila razdeljena, prvi atribut na tri vrednosti in drugi atribut na 9 različnih vrednosti, ki smo jih tako tudi upoštevali pri ocenjevanju variance, saj smo jih v taki obliki uporabili tudi pri napovednih modelih.

2.4.2 Ocenjevanje atributov z algoritmom RReliefF

Za razliko od kratkovidnih mer za ocenjevanje kvalitete atributov algoritem Relief ne predpostavlja neodvisnosti atributov, temveč upošteva kontekst in zanesljivo ocenjuje atribute v problemih z močno medsebojno odvisnostjo. Osnovna ideja algoritma Relief je, da ocenjuje kvaliteto atributov glede na to, kako dobro vrednost atributa ločuje med primeri, ki so si med seboj podobni [13]. Algoritem za naključni primer R_i poišče dva najbližja soseda, en primer *H* iz istega razreda in en primer *M* iz nasprotnega razreda. Glede na vrednosti posameznih atributov iz primerov R_i, *H*, *M* nato posodobi ocene kvalitete atributov. Postopek *m*-krat ponovimo, kjer je *m* parameter algoritma.

ReliefF je razširitev algoritma Relief, ki odpravlja problem neznanih vrednosti podatkov, algoritem razširi na reševanje večrazrednih problemov in poveča zanesljivost iskanja zadetkov *H* in pogreškov *M* tako, da za dani učni primer poišče *k* najbližjih zadetkov in pogreškov.

Regresijski ReliefF oz. RReliefF je razširitev ReliefF-a na probleme z zveznimi napovednimi vrednostmi. Zaradi zveznosti napovedne spremenljivke ne more uporabiti najbližjih zadetkov in pogreškov, namesto tega pa uporablja *"verjetnost, da dva primera pripadata različnima razredoma"*, ta verjetnost pa je modelirana kot relativna razdalja med vrednostima odvisne spremenljivke dveh primerov [11].

Poglavje 3

Metodologija

3.1 Podatki

Podatke je priskrbelo podjetje Celtra d.o.o. Uporabljeni podatki vsebujejo anonimizirane podatke o 1315 oglasih na 2407 straneh med 1.1.2019 in 30.9.2019. Podatki so razdeljeni po mesecih, torej imamo za vsak posamezen mesec od januarja do septembra svojo datoteko, v kateri je vsaka vrstica podana v obliki: creative, externalSite, numShown, numInteractions. Dodatno imamo datoteko, v kateri so shranjene lastnosti vseh oglasov, ki so bili uporabljeni. V tej datoteki je vsaka vrstica podana v obliki: creative, hasVisualEffect, hasCreativity, hasLocation, hasPersonalization, hasDynamicContent, creativeComplexityRank, isCreatedFromCopy, adExperience, effectiveIndustry, industry.

creative je ID oglasa in je zapisana kot zgoščena vrednost (angl. *hash value)*.

externalSite je ID spletne strani, na kateri je oglas prikazan in je zapisana kot zgoščena vrednost.

numShown je podatek o številu prikazov oglasa na spletni strani.

numInteractions je podatek o številu klikov uporabnikov na oglas, ki je bil prikazan na tej strani.

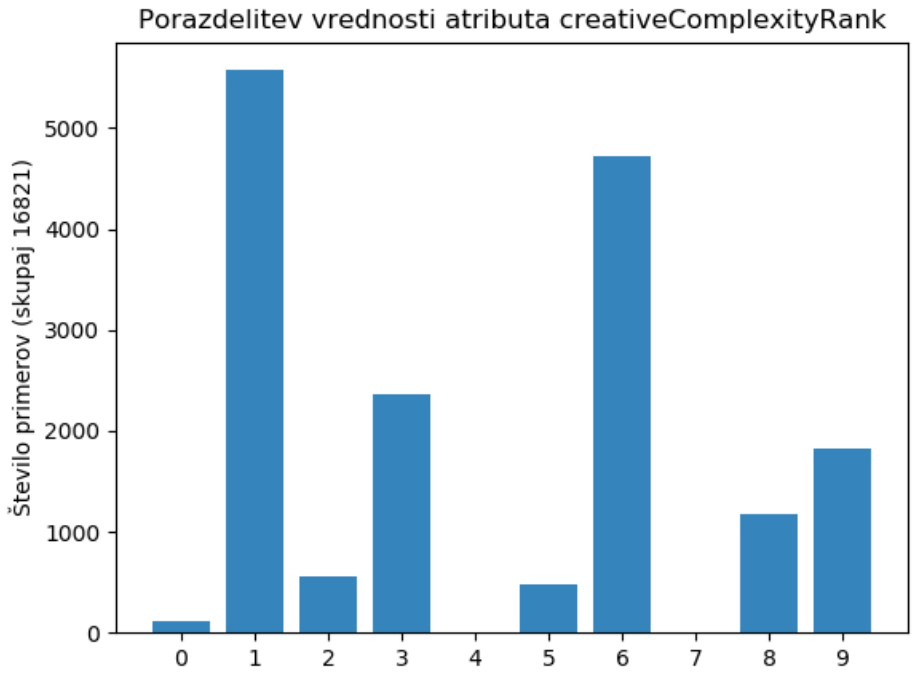
Slika 3.1: Porazdelitev vrednosti atributa hasVisualEffect. Tako kot ostali binarni atributi v množici ima velika večina primerov enako vrednost.

hasVisualEffect je binaren atribut, ki pove, ali so v oglasu uporabljeni vizualni effekti, npr. gladki prehodi, preproste animacije snega/dima in podobno. Porazdelitev vrednosti je prikazana na sliki 3.1.

hasCreativity je binaren atribut, ki pove, ali so v oglasu uporabljene komponente, ki 'spodbujajo' kreativnost/interaktivnost uporabnikov (npr. drag'n'drop, flip, swipe, puzzle).

hasLocation je binaren atribut, ki pove, ali so v oglasu uporabljene komponente za prikazovanje zemljevidov, lokacij trgovin ipd.

hasPersonalization je binaren atribut, ki pove, ali se vsebina (besedila, slika ali druge komponente) prilagaja trenutnemu uporabnikovemu kontekstu (nje-



Slika 3.2: Porazdelitev vrednosti atributa creativeComplexityRank.

gova lokacija, trenutno vreme na njegovi na lokaciji, čas v dnevu, tip uporabnikove naprave...).

hasDynamicContent je binaren atribut, ki pove enako kot hasPersonalization, vsebuje pa še oglase z a/b testi, split testi ipd (tega je zelo malo). Pri teh testih so uporabniki dodeljeni control/treatment skupinam, od česar je odvisna tudi vsebina oglasa, npr. različne slike za vsako treatment skupino, kar nam omogoča primerjavo performančne analize.

creativeComplexityRank je atribut, ki zavzema vrednosti med 0 in 9 in ponazarja kompleksnost oglasov: preproste statične pasice $<$ ima animacije $<$ ima video ali napredne komponente $<$ ima dinamične/personalizirane komponente ipd. Porazdelitev vrednosti je prikazana na sliki 3.2

isCreatedFromCopy je binaren atribut, ki pove, ali je bil oglas ustvarjen s kopiranjem drugega oglasa in kasneje modificiran.

adExperience je podatek o velikosti oglasa, ki združuje oglase podobnih velikosti in zavzema vrednosti 'bigBanner', 'mediumBanner', 'smallBanner', 'irregularSizeBanner', 'unknown'.

industry je podatek o tem, v kakšno industrijo spada oglas, oziroma kateri industriji pripada podjetje, ki je izdalo ta oglas. industry zavzema 6 različnih vrednosti in nam pove bolj splošno industrijo. Kategorije, katerim lahko oglas pripada, so: automotive, retail, consumerPackagedGoods, foodBeverage , entertainmentMedia, other

effectiveIndustry je podatek, ki nam, podobno kot prejšnji atribut, pove, v katero industrijo spada oglas, le da je to bolj specifično, saj je na voljo 17 razlicnih kategorij: healthSports, beautyPersonalCare, retail, energyUtilities, education, pharmaceutical, quickServiceRestaurants, unknown, finance, insuranceRealEstate, travelHospitality, artsDesignFashion, telecommunications, government, technology, shippingLogistics, automotive, foodBeverage, entertainmentMedia, other, consumerPackagedGoods

3.2 Priprava podatkov

Pri strojnem učenju je pogosto bolj kot izbira najboljšega modela pomembna primerna obdelava podatkov. Potrebno je dobro razumevanje podatkov, da jih lahko pravilno oblikujemo, in s tem preprečimo gradnjo modela na nepopolnih podatkih. V pripravo podatkov spada obravnava manjkajočih vrednosti, takih primerov pa najpogosteje ne upoštevamo pri gradnji modela. Prav tako lahko iz podatkov odstranimo tiste atribute, ki vsebujejo šumne podatke, ali na kakšen drugi način ne prispevajo h kakovosti zgrajenega napovednega modela.

Ciljna vrednost, ki nas zanima in jo želimo napovedovati, je stopnja interakcij, oziroma *click-through rate* (CTR). Ta vrednost pomeni delež števila klikov od

števila prikazov in jo dobimo tako, da delimo vrednost atributa numInteractions z vrednostjo atributa numShown.

Binarni atributi, opisani v prejšnjem razdelku, imajo vrednosti 0 in 1 in jih nismo spreminjali. Atribut creativeComplexityRank smo normalizirali in je nato zavzemal vrednosti med 0 in 1. Vrednosti atributa adExperience, ki nam pove velikost oglasa, smo nadomestili s številkami, 'smallBanner' z 0, 'mediumBanner' z 0.5 in 'bigBanner' z 1. Vrednost 'irregularSizeBanner' pomeni netipično obliko oglasa, na primer zelo ozek in podolgovat oglas. Ker se je kasneje izkazalo, da je adExperience zelo pomemben atribut, smo oglase z vrednostjo 'irregularSizeBanner' izključili iz učne množice, prav tako pa smo izključili oglase z neznano vrednostjo. Atribut industry smo binarizirali, kar pomeni, da smo ga razbili na 6 posameznih atributov. Za vsako izmed vrednosti, ki jo lahko atribut zaseda, smo naredili svoj atribut. Torej, če je imel atribut industry vrednost 'automotive', smo nov atribut automotive nastavili na 1 in ostalih 5 na 0. Enako kot industry smo binarizirali atribut effectiveIndustry.

3.2.1 Spletna stran kot atribut

Najprej smo spletno stran, na kateri se oglas nahaja, vzeli kot atribut tako, da je bil en primer v učni množici sestavljen iz (prej opisanih) lastnosti oglasa, strani, na kateri je bil oglas prikazan, in CTR, ki je ciljna vrednost. Za testno množico smo vzeli le tiste primere, pri katerih se je stran že pojavila v učni množici. Problem se pojavi pri atributih z nominalnimi vrednostmi, torej pri atributih o industriji in atributu o strani. Atributa o industriji smo binarizirali in s tem ustvarili nekaj atributov več, atribut strani pa smo obdelali na več načinov:

Pri prvem načinu smo atribut strani prav tako binarizirali (kasneje *binarizacija* oz. *bin*). Vendar se ob tem pojavi problem, ker je vseh različnih strani v 9 mesecih več kot 2400 in se z binarizacijo, ko vsaka stran dobi svoj atribut, število atributov poveča iz cca. 35 na skoraj 2500. Tako veliko število atributov pa slabo vpliva na gradnjo napovednega modela, saj se čas učenja bistveno poveča.

Pri drugem načinu smo strani kategorizirali (kasneje *kategorizacija* oz. *kat*) po velikosti tako, da smo izračunali povprečni CTR vseh oglasov, ki so bili v

učni množici, prikazani na tej strani. Strani smo po velikosti razdelili v 50 enako velikih skupin in jih označili z zaporedno številko. Strani v prvi skupini so bile tiste, ki so dosegle največji CTR in so za nov atribut strani dobile vrednost 1. Strani v drugi skupini so bile tiste, ki so se po velikosti povprečnega CTR uvrstile za tistimi stranmi v prvi skupini, in so za nov atribut strani dobile vrednost 2. Tako smo naredili za vseh 50 skupin in nato atribut normalizirali.

V 3. načinu smo združili prva dva načina in prvih 50 strani, na katerih so oglasi dosegali največji povprečni CTR, binarizirali, ostale strani pa smo kategorizirali kot v 2. načinu (kasneje *kombinacija* oz. *komb*).

3.2.2 Izdelava modela za vsako stran

Kasneje se je izkazalo, da je stran zelo pomemben atribut, najboljše rezultate pa smo dosegli, če je bila spletna stran binarizirana. Kot smo omenili, se z binarizacijo strani močno poveča čas učenja, zato smo se odločili, da bomo stran upoštevali drugače. Problema smo se lotili tako, da nismo vzeli strani kot atribut, temveč za vsako stran iz učne množice zgradili svoj napovedni model. Učna množica za posamezen model pa sestoji iz lastnosti oglasov, ki so oglaševali na tej spletni strani. V testni množici za model na določeni spletni strani so bili le oglasi, ki so se oglaševali na tej spletni strani in se v preteklosti niso oglaševali še na nobeni spletni strani.

3.3 Mere uspešnosti

3.3.1 RMSE

Prva metrika, ki smo jo uporabili za ocenjevanje uspešnosti napovedovanja vrednosti CTR za nove oglase, je koren srednje kvadratne napake (angl. *root mean squared error*) in se pogosto uporablja pri regresijskih problemih in bolj kaznuje

večja odstopanja kot MSE [7]:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - \hat{r}_i)^2} \tag{3.1}$$

kjer je r_i prava vrednost ciljne spremenljivke CTR, $\hat{r}_i$ pa napoved za CTR, ki jo dobimo z napovednim modelom.

3.3.2 RRMSE

Druga metrika je koren relativne srednje kvadratne napake (angl. *root relative mean squared error*) in nam omogoča boljši vpogled v uporabnost napovednega modela, saj napake primerja z napakami, ki bi se zgodile, če bi napovedovali s povprečno vrednostjo učne množice $\bar{r}$:

$$RRMSE = \sqrt{\frac{\sum_{i=1}^{n}(r_i - \hat{r}_i)^2}{\sum_{i=1}^{n}(r_i - \bar{r})^2}} \tag{3.2}$$

3.3.3 RAE

Uspešnost smo merili tudi z relativno absolutno napako (angl. *relative absolute error*) [11], ki prav tako kot $RRMSE$ omogoča vpogled v uporabnost modela, vendar pa namesto kvadrata napake uporablja absolutno napako:

$$RAE = \frac{\sum_{i=1}^{n} \mid r_i - \hat{r}_i \mid}{\sum_{i=1}^{n} \mid r_i - \bar{r} \mid} \tag{3.3}$$

3.4 Prečno preverjanje

Prečno preverjanje je tehnika, ki je pogosto uporabljena za ocenjevanje in izgradnjo napovednih modelov v strojnem učenju. Metoda prečnega preverjanja ima le en parameter k, zato metodi pravimo tudi k-kratno prečno preverjanje (angl. *k-fold cross validation*), ki nam pove, na koliko delov razdelimo učno množico [6]. Metoda deluje tako, da učne primere razdelimo v k naključnih enako velikih množic. Za vsako izmed k nastalih množic, vzamemo to množico za novo

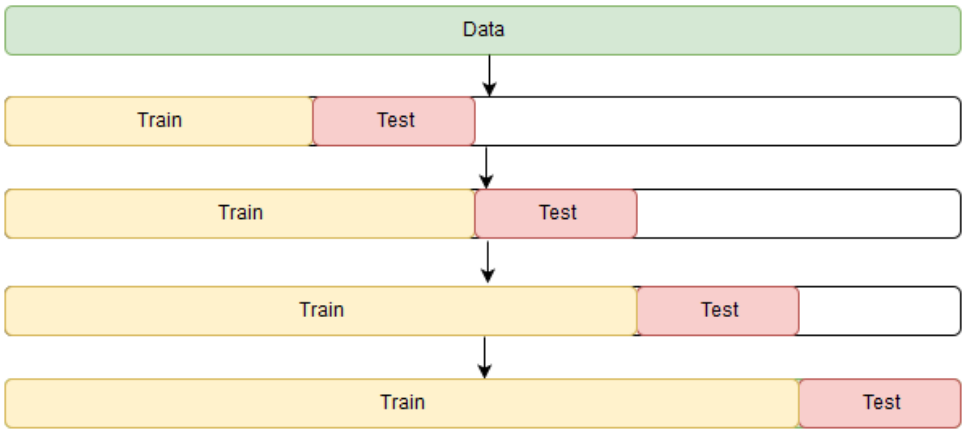
testno množico in preostalih *k-1* množic, kot novo učno množico. Na novo nastali učni množici natreniramo naš napovedni model in ga preizkusimo na novi testni množici. Shranimo si napovedno točnost za vsako testno množico in na koncu vzamemo povprečje napovednih točnosti. Na tak način se vsak primer v testni množici pojavi enkrat in v učni množici (*k-1*)-krat.

3.4.1 Prečno preverjanje za časovno odvisne podatke

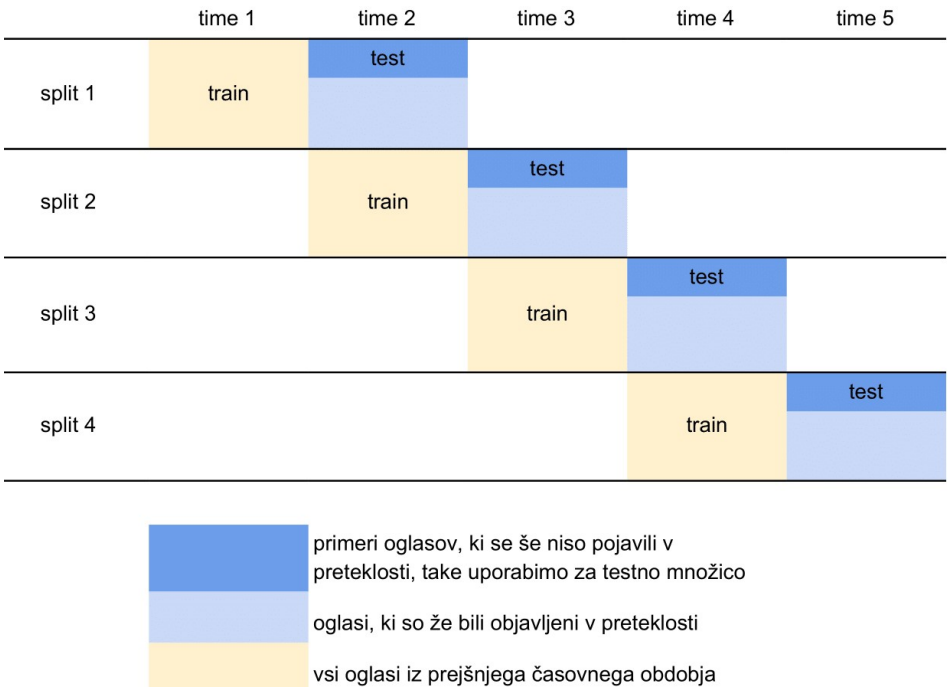
Pri časovno urejenih podatkih se pojavi problem, saj s prečnim preverjanjem in naključnim izbiranjem podatkov za testne množice dosežemo, da so podatki iz istega časovnega obdobja, ki so med seboj močno korelirani, pomešani tako v testni kot v učni množici. Na ta način napovedni model 'dobi vpogled' v testno množico in s tem dobimo optimistično oceno za napovedno točnost modela.

Za časovno urejene podatke lahko uporabljamo prečno preverjanje za časovne vrste (angl. *cross validation for time series*) [10]. Tudi tukaj učno množico razdelimo na n podmnožic (če le že ni tako urejena) in nato zgradimo model na podmnožicah y_1, ..., y_t. Za podmnožico y_{t+1} napovemo vrednosti in jih primerjamo z dejanskimi ter izračunamo napako. Postopek ponovimo za $t = m, ..., n-1$ (kot je prikazano na sliki 3.3) in izračunamo povprečno napako.

V našem primeru učne množice nismo povečevali, vendar smo jo premikali in smo za učno množico vzeli določeno časovno obdobje $time_i$ (1 mesec oz. 2 meseca) pred testno množico, saj prevelika učna množica, ki zajema več mesecev, bistveno poveča čas učenja in celo poslabša napovedno točnost, saj se povprečni *click-through rate* spreminja glede na časovno obdobje v letu, v decembru je na primer le ta višji zaradi prazničnih nakupov, kar povzroči večje število klikov na oglase. Za testno množico smo vzeli le tiste primere iz časovnega obdobja $time_{i+1}$ (kot je prikazano na sliki 3.4), kjer so oglasi, ki se v preteklosti še niso oglaševali.



Slika 3.3: Prečno preverjanje za časovno odvisne podatke. Vir: [5]



Slika 3.4: Prečno preverjanje v našem primeru.

Poglavje 4

Eksperimenti in rezultati

Različne napovedne modele z omenjenimi obdelavami podatkov smo testirali na dveh testnih množicah. Prva testna množica je bila *test1*, ki vsebuje tiste primere, kjer je oglas na spletni strani, na kateri je bil v preteklosti vsaj 1 oglas, druga testna množica *test15* pa vsebuje primere, kjer je oglas na spletni strani, na kateri je bilo v preteklosti vsaj 15 različnih oglasov. S primerjavo rezultatov na dveh testnih množicah smo želeli ugotoviti, ali s tem, ko dobimo podatke o stopnji interakcij z več oglasi na isti spletni strani, dobimo nek vpogled, kakšni oglasi so bolj primerni za to spletno stran in ali se s tem izboljša napovedna točnost.

Poleg tega smo za vsak napovedni model, uporabili množico podatkov, ki so bili pridobljeni en mesec prej in pa še eno množico podatkov, kjer so bili podatki iz prejšnjih 2 mesecev. Želeli smo ugotoviti, ali povečanje obdobja pomeni večjo točnost, saj se skozi leto trend stopnje interakcij z oglasi spreminja, poleg tega pa povečanje učne množice pomeni večjo časovno zahtevnost, ki je morda nismo pripravljeni sprejeti za majhno izboljšanje napovedne točnosti.

Vse modele smo testirali s prečnim preverjanjem, kjer smo postopek ponovili 7-krat (glej sliko 3.4) tako, da so bili v testni množici podatki za mesece od marca do septembra. Rezultati pa so uteženo povprečje, glede na število testnih primerov v posameznem mesecu, preko 7 mesecev (glej sliko 3.4). V tabelah od 4.1 do 4.8 ’1 mesec’ pomeni rezultate, ki smo jih dobili na učni množici, ki je vsebovala primere iz prejšnjega meseca, ’2 meseca’ pa pomeni rezultate, ki smo jih dobili

		1 mesec			2 meseca		
		RMSE	RRMSE	RAE	RMSE	RRMSE	RAE
RF	bin	**0.004693**	**0.6420**	**0.3847**	**0.004653**	**0.6399**	**0.3749**
	kat	0.004949	0.6800	0.3990	0.004802	0.6614	0.3786
	komb	0.004913	0.6751	0.3967	0.004839	0.6646	0.3833
kNN, k = 3	bin	0.005638	0.7652	0.4837	0.005406	0.7388	0.4581
	kat	0.005772	0.7782	0.5062	0.005253	0.7062	0.4622
	komb	0.006172	0.8369	0.4388	0.006292	0.8633	0.5381
kNN, k = 5	bin	0.005561	0.7590	0.4951	0.005153	0.7103	0.4588
	kat	0.005437	0.7388	0.4841	0.005041	0.6847	0.4470
	komb	0.006385	0.8629	0.4446	0.005784	0.7847	0.5236

Tabela 4.1: Rezultati testiranja modelov na množici *test1* s spletno stranjo kot atributom. '1 mesec' pomeni rezultate, ki smo jih dobili na učni množici, ki je vsebovala primere iz prejšnjega meseca, '2 meseca' pa pomeni rezultate, ki smo jih dobili na učni množici, ki je vsebovala primere iz prejšnjih dveh mesecev. Rezultati so povprečje prečnega preverjanja preko 7 mesecev (glej sliko 3.4).

na učni množici, ki je vsebovala primere iz prejšnjih dveh mesecev.

4.1 Spletna stran kot atribut

V tabelah 4.1 in 4.2 so rezultati testiranja modela naključnega gozda in k-najbližjih sosedov na množicah *test1* in *test15* z različnimi obdelavami podatkov, kot so opisani v razdelku 3.2.1.

Na testni množici *test1* se je najbolje obnesel model naključnega gozda z binarizacijo strani, ne glede na metriko. Opazili smo tudi, da smo pri vseh treh različicah obdelave spletne strani z naključnim gozdom dobili bistveno boljše rezultate kot s kNN. Pri večini modelov smo s povečanjem učne množice dobili tudi boljše rezultate.

Na testni množici *test15* se je prav tako v vseh primerih najbolje obnesel model

		1 mesec			2 meseca		
		RMSE	RRMSE	RAE	RMSE	RRMSE	RAE
RF	bin	0.004095	0.6297	**0.4086**	**0.003712**	**0.5981**	**0.3523**
	kat	**0.003907**	**0.5975**	0.4247	0.003746	0.6003	0.3722
	komb	0.004299	0.6517	0.4393	0.003930	0.6261	0.3879
kNN, k = 3	bin	0.005001	0.7618	0.5225	0.004205	0.6766	0.4222
	kat	0.006127	0.9179	0.6607	0.005198	0.8030	0.5380
	komb	0.007041	1.0167	0.5596	0.005718	0.9033	0.5654
kNN, k = 5	bin	0.005204	0.7943	0.5385	0.004451	0.7125	0.4367
	kat	0.005623	0.8544	0.6100	0.004982	0.7810	0.5115
	komb	0.006969	1.0063	0.5525	0.005297	0.8273	0.5390

Tabela 4.2: Rezultati testiranja modelov na množici *test15* s spletno stranjo kot atributom. Rezultati so povprečje prečnega preverjanja preko 7 mesecev (glej sliko 3.4).

naključnega gozda, pri enomesečni učni množici, je glede na RMSE in RRMSE najboljši rezultat dosegla kategorizacija strani, v ostalih primerih pa binarizacija strani. Tudi na tej testni množici so se, s povečanjem učne množice, rezultati praviloma izboljšali.

Na testni množici *test1* smo preizkusili tudi naključni gozd z binarizacijo strani, ki smo ga natrenirali na tromesečni učni množici in ugotovili, da res s konstantnim povečevanjem ne dobimo boljših rezultatov, saj je stopnja interakcije odvisna tudi od obdobja v letu. Dobili smo RMSE = 0.004679, RRMSE = 0.6663 in RAE = 0.3756.

4.2 Izdelava modela za vsako stran

Kot smo ugotovili z ocenjevanjem atributov, spletna stran najbolj vpliva na stopnjo interakcij z oglasom in ker se med posameznimi stranmi CTR zelo razlikujejo, smo se odločili, da za vsako spletno stran sestavimo svoj napovedni model in za

	1 mesec			2 meseca		
	RMSE	RRMSE	RAE	RMSE	RRMSE	RAE
RF	0.005003	0.6945	0.3884	0.004754	0.6601	0.3656
kNN, k = 3	0.005121	0.7124	0.3995	0.004936	0.6920	0.3808
kNN, k = 5	0.005152	0.7163	0.4111	0.005022	0.7041	0.3841
kNN, k = 3, utežen	**0.004919**	**0.6849**	**0.3785**	**0.004671**	**0.6544**	**0.3556**
kNN, k = 5, utežen	0.005028	0.7008	0.3994	0.004837	0.6814	0.3668

Tabela 4.3: Rezultati testiranja modelov, zgrajenih na posameznih straneh, na množici *test1*. Rezultati so povprečje prečnega preverjanja preko 7 mesecev (glej sliko 3.4).

učno množico uporabimo oglase iz tiste spletne strani.

V tabelah 4.3 in 4.4 so rezultati testiranja modela naključnega gozda, k-najbližjih sosedov in uteženih k-najbližjih sosedov na množicah *test1* in *test15* z različnimi obdelavami podatkov, kot so opisani v razdelku 3.2.2. Za uporabo uteženega kNN smo se odločili, ker smo želeli na podlagi ocen atributov nekatere atribute bolj upoštevati kot druge in s tem zmanjšati napovedno napako. Tako smo pri izračunu razdalje med dvema primeroma za atribut adExperience eksperimentalno določili in uporabili utež $w_{adExperience} = 3$ in za binarne atribute, ki so predstavljali posamezne industrije (tako splošne, kot efektivne), uporabili uteži $w_i = 0.1$.

Na testni množici *test1* z enomesečnimi učnimi podatki je najboljše rezultate dosegel utežen kNN s parametrom k = 3, sledil pa mu je model naključnega gozda. Prav tako sta ta dva modela zasedla prvi mesti pri dvomesečnih podatkih, in sta izboljšala svoj rezultat, tako kot ostali modeli.

Z omejitvijo na vsaj 15 oglasov na vsaki strani, smo dosegli, da ima posamezen model kNN več oglasov, med katerimi lahko izbira in tako najde najbolj podobnega tistemu iz testne množice. To vidimo v tabeli 4.4, kjer oba modela utežen kNN dosežeta boljše rezultate od naključnega gozda, in tako z uteženim modelom kNN k=3 dosežemo najnižji RMSE in RAE tako na enomesečnih kot na dvomesečnih učnih množicah na testni množici *test15*.

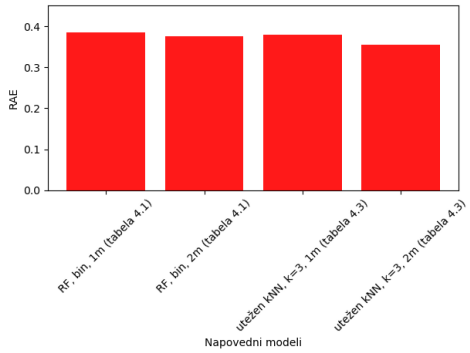
	1 mesec			2 meseca		
	RMSE	RRMSE	RAE	RMSE	RRMSE	RAE
RF	0.003664	0.5670	0.3787	0.003282	0.5333	0.3185
kNN, k = 3	0.003923	0.5961	0.4081	0.003539	0.5680	0.3503
kNN, k = 5	0.003915	0.5914	0.4150	0.003541	0.5661	0.3498
kNN, k = 3, utežen	**0.003454**	0.5354	**0.3645**	**0.003027**	**0.4961**	**0.3058**
kNN, k = 5, utežen	0.003463	**0.5349**	0.3763	0.003042	0.5003	0.3070

Tabela 4.4: Rezultati testiranja modelov, zgrajenih na posameznih straneh, na množici *test15*. Rezultati so povprečje prečnega preverjanja preko 7 mesecev (glej sliko 3.4).

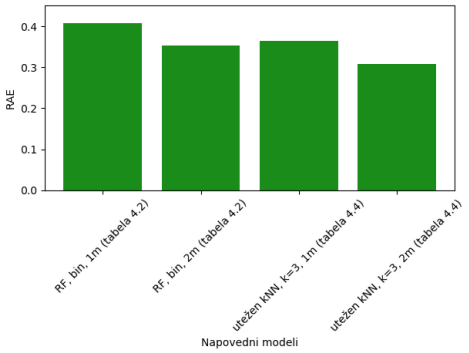
Model uteženih k-najbližjih sosedov s parametrom k=3, ki je dosegal najboljše rezultate na obeh testnih množicah, smo natrenirali tudi na tromesečni učni množici in ugotovili, da tudi najboljšemu modelu večinoma poslabšamo napovedno točnost s takšnim povečanjem časovnega obdobja. Na testni množici *test1* je dosegel rezultate RMSE = 0.004749, RRMSE = 0.6815, RAE = 0.3498. Na testni množici *test15* pa je dosegel rezultate RMSE = 0.003524, RRMSE = 0.5615 in RAE = 0.3161. Grafični prikaz rezultatov najboljših modelov vidimo na sliki 4.1.

4.3 Eksperiment z matrično faktorizacijo

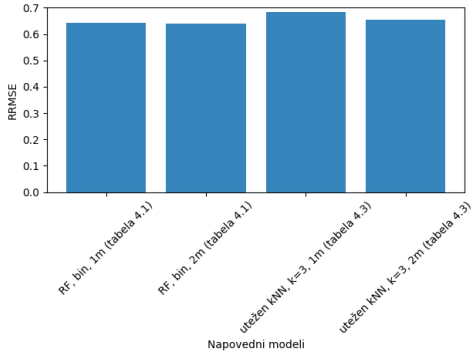
Kot smo opisali v razdelku 2.3, lahko matrično faktorizacijo uporabimo za napoved CTR oglasov na spletnih straneh, ki še niso bili prikazani, so bili pa v preteklosti prikazani na nekih drugih spletnih straneh. V našem primeru pa oglasi, za katere želimo napovedati CTR na spletnih straneh, v preteklosti še niso bili prikazani na nobeni spletni strani in zato na tak način ne moremo rešiti problema. Ideja, ki smo jo želeli preizkusiti, je, da obstoječe napovedi CTR oglasov na spletnih straneh, ki smo jih dobili z nekim drugim napovednim modelom (v našem primeru naključni gozd in k-najbližjih sosedov), poizkušamo izboljšati z matrično faktorizacijo tako, da matrično faktorizacijo izvedemo na učni množici in množici



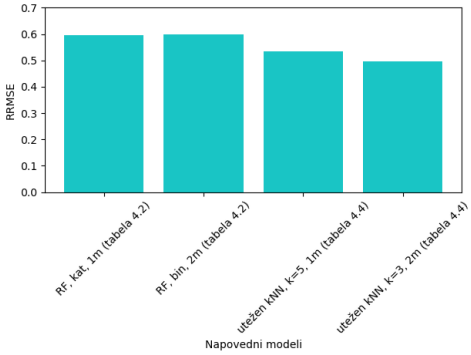
(a) RAE, *test1*



(b) RAE, *test15*



(c) RRMSE, *test1*



(d) RRMSE, *test15*

Slika 4.1: Grafična primerjava RRMSE in RAE najboljših modelov na učnih množicah *test1* in *test15*.

napovedi skupaj. Želeli smo doseči, da bi se napovedi malo spremenile tako, da bi se prilagodile glede na globalno sliko, ki jo ustvarjajo učni podatki. V tabelah 4.5, 4.6 so prvi trije modeli, ki so razdeljeni vsak na 3 obdelave podatkov, zgrajeni na množici, kjer je stran upoštevana kot atribut, zadnjih 5 modelov pa na podatkih, kjer smo na vsaki strani zgradili posamezen model. V tabelah 4.7 in 4.8 so rezultati podani v eni vrstici saj so za vse modele enaki.

4.3.1 Matrična faktorizacija

V tabeli 4.5 lahko vidimo, da je na testni množici test1 po izvedeni matrični faktorizaciji, tako na enomesečni kot na dvomesečni učni množici, najboljši rezultat glede na RMSE in RRMSE dosegel model kNN, k = 5, zgrajen na vseh podatkih, s kategorizacijo strani. Glede na RAE pa je najboljši rezultat dosegel naključni gozd z binarizacijo strani.

Ob primerjavi rezultatov matrične faktorizacije z rezultati posameznih prejšnjih modelov (tabeli 4.1 in 4.3), smo ugotovili, da je izvedba matrične faktorizacije kar nekaj modelom izboljšala napovedno točnost, vendar so bili to modeli, ki so prej dosegli slabše rezultate. Modelom, ki so do sedaj dosegli najboljše rezultate, se napaka ni zmanjšala. Opazimo tudi, da so se najbolj zmanjšale metrike RMSE in RRMSE. To nam pove, da matrična faktorizacija res prepozna tiste primere, ki zelo odstopajo od globalne slike in jim prilagodi napoved.

Na testni množici *test15* (v tabeli 4.6) je na enomesečnih učnih podatkih najboljši rezultat dosegel model utežen kNN, k=3, zgrajen na vsaki strani posebej. Na dvomesečnih učnih podatkih pa je najboljše rezultate dosegel utežen kNN, k=5, zgrajen na posameznih spletnih straneh. Skoraj v vseh primerih pa so se najboljše napovedne točnosti po matrični faktorizaciji bistveno poslabšale glede na tiste pred matrično faktorizacijo (tabela 4.4). Modelom, ki so prej dosegali najslabše rezultate, pa je matrična faktorizacija izboljšala napovedno točnost glede na RMSE in RRMSE. RAE je tukaj praviloma slabši kot tisti pred matrično faktorizacijo.

		1 mesec			2 meseca		
		RMSE	RRMSE	RAE	RMSE	RRMSE	RAE
RF	bin	0.004811	0.6617	**0.4351**	0.004726	0.6548	**0.4212**
	kat	0.005183	0.7140	0.4530	0.005041	0.6991	0.4371
	komb	0.005088	0.7019	0.4497	0.005049	0.7002	0.4407
kNN, k=3	bin	0.004924	0.6743	0.4595	0.004849	0.6695	0.4512
	kat	0.004819	0.6634	0.4500	0.004698	0.6497	0.4382
	komb	0.005425	0.7464	0.4633	0.004978	0.6875	0.4677
kNN, k=5	bin	0.004846	0.6658	0.4620	0.004712	0.6532	0.4483
	kat	**0.004798**	**0.6608**	0.4478	**0.004692**	**0.6494**	0.4359
	komb	0.005630	0.7732	0.4702	0.004920	0.6775	0.4673
RF		0.005249	0.7260	0.4552	0.005135	0.7133	0.4414
kNN, k=3		0.005248	0.7267	0.4519	0.005138	0.7170	0.4365
kNN, k=5		0.005301	0.7346	0.4583	0.005236	0.7310	0.4420
kNN, k=3, utežen		0.005222	0.7236	0.4524	0.005106	0.7126	0.4354
kNN, k=5, utežen		0.005281	0.7322	0.4574	0.005209	0.7279	0.4407

Tabela 4.5: Rezultati poskusa matrične faktorizacije na testni mnozici *test1*. Rezultati so povprečje prečnega preverjanja preko 7 mesecev (glej sliko 3.4).

		1 mesec			2 meseca		
		RMSE	RRMSE	RAE	RMSE	RRMSE	RAE
RF	bin	0.004637	0.7052	0.5703	0.004308	0.6830	0.5034
	kat	0.004682	0.7124	0.5713	0.004355	0.6897	0.5184
	komb	0.004681	0.7109	0.5682	0.004360	0.6901	0.5129
kNN, k=3	bin	0.004661	0.7078	0.5789	0.004333	0.6868	0.5165
	kat	0.004791	0.7287	0.5981	0.004459	0.7048	0.5364
	komb	0.005671	0.8428	0.6258	0.004577	0.7223	0.5598
kNN, k=5	bin	0.004643	0.7044	0.5743	0.004376	0.6930	0.5247
	kat	0.004802	0.7297	0.5921	0.004457	0.7048	0.5313
	komb	0.005524	0.8244	0.6154	0.004502	0.7110	0.5559
RF		0.004646	0.7086	0.5609	0.004305	0.6827	0.5032
kNN, k=3		0.004597	0.6988	0.5557	0.004304	0.6822	0.5040
kNN, k=5		0.004602	0.6995	0.5570	0.004290	0.6796	0.5036
kNN, k=3, utežen		**0.004552**	**0.6927**	**0.5568**	0.004268	0.6774	0.5035
kNN, k=5, utežen		0.004565	0.6950	0.5569	**0.004260**	**0.6764**	**0.5007**

Tabela 4.6: Rezultati poskusa matrične faktorizacije na testni mnozici *test15*. Rezultati so povprečje prečnega preverjanja preko 7 mesecev (glej sliko 3.4).

	1 mesec			2 meseca		
	RMSE	RRMSE	RAE	RMSE	RRMSE	RAE
vsi modeli	0.00544	0.755	0.480	0.00540	0.755	0.464

Tabela 4.7: Rezultati poskusa matrične faktorizacije z dodajanjem na testni množici *test1*. Rezultati so povprečje preko prečnega preverjanja 7 mesecev (glej sliko 3.4). Za vse modele dobimo enake rezultate na 5 decimalk pri RMSE in na 3 decimalke pri RRMSE ter RAE.

	1 mesec			2 meseca		
	RMSE	RRMSE	RAE	RMSE	RRMSE	RAE
vsi modeli	0.00502	0.767	0.612	0.00466	0.740	0.548

Tabela 4.8: Rezultati poskusa matrične faktorizacije z dodajanjem na testni množici *test15*. Rezultati so povprečje prečnega preverjanja preko 7 mesecev (glej sliko 3.4). Za vse modele dobimo enake rezultate na 5 decimalk pri RMSE in na 3 decimalke pri RRMSE ter RAE.

4.3.2 Matrična faktorizacija z dodajanjem novih oglasov

V tabelah 4.7 in 4.8 so rezultati, ki smo jih pridobili, ko smo na učni množici naredili matrični razcep in nato v matrični razcep dodali napovedi za testno množico (kot je opisano v razdelku 2.3.1), ki smo jih pridobili z omenjenimi napovednimi modeli. Vidimo lahko, da učni podatki tako močno vplivajo na začetni razcep, da se po dodajanju novih elementov matrična faktorizacija vedno ujame v isti lokalni minimum in tako vedno dobimo enake rezultate, zato smo v tabeli 4.7 in 4.8 vključili le eno vrstico z rezultati, ki veljajo za vse modele. S tem, ko smo dobili enake rezultate za vse napovedne modele, smo, razen v nekaj primerih, poslabšali napovedno točnost.

4.4 Ocene atributov

Atribute smo ocenjevali na množici, sestavljeni iz vseh 9 mesecev. Ocenjevali smo z metodama RReliefF in z razliko variance.

4.4.1 RReliefF

V tabeli 4.9 so rezultati ocenjevanja atributov na množici, ki je imela spletno stran kot binariziran atribut, in so tako spletne strani predstavljale večino atributov. Med 20 najbolj pomembnih atributov se je uvrstil le en atribut, ki ni bil spletna stran, to je bil adExperience. Poleg tega se je med prvih 100 najpomembnejših atributov uvrstil le še en atribut, ki ni predstavljal strani. To je bil atribut isCreatedFromCopy, ki se je uvrstil na 26. mesto. Glede na RReliefF so spletne strani torej najpomembnejši atributi. Enako nam pove tabela 4.10, v kateri smo spletno stran predstavili kot en posamezen atribut 'site', vrednost atributa pa je pomenila kategorija strani, kot opisano v razdelku 3.2.1. Za ocenjevanje smo uporabili 50 kategorij strani. Tudi tu je spletna stran najpomembnejši atribut. Drugi najpomembnejši atribut, oziroma najpomembnejši atribut, ki ni stran, pa je adExperience.

4.4.2 Razlika variance

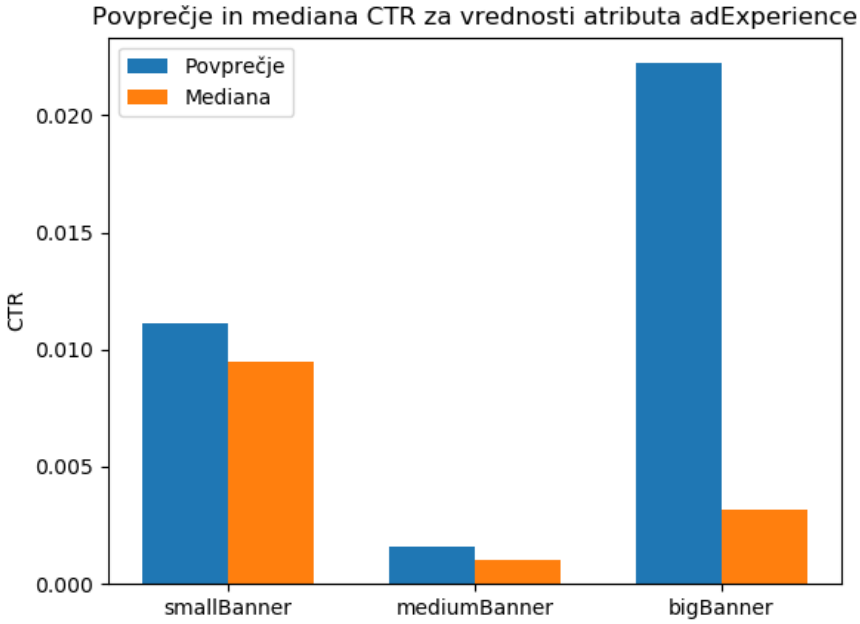
Tako kot pri ocenjevanju z RReliefF-om so se spletne strani, oziroma site in adExperience, izkazali za najpomembnejša atributa. Pri kategorizaciji strani sta site in adExperience dosegla 1. in 2. mesto (glej tabelo 4.12). Pri binarizaciji (glej tabelo 4.11) pa se je 10 strani uvrstilo med prvih 20 mest. Da se posamezne strani niso uvrstile prav na najvišja mesta, lahko pojasnimo s tem, da je primerov s točno določeno stranjo med množico primerov zelo malo ($<1\%$). Zato se težko ustvari večja razlika z varianco celotne množice. adExperience pa se tudi tu uvrsti na prvo mesto. Na sliki 4.2 lahko vidimo povprečni CTR in mediano CTR za posamezne vrednosti atributa adExperience. Vidimo, da je mediana CTR za oglase z vrednostjo bigBanner veliko manjša od povprečja. To se zgodi, ker je primerov oglasov z vrednostjo bigBanner zelo malo in je povprečje bolj občutljivo na

številka	RReliefF	
	Atribut	ocena
1	x0_2b3f012ca20a607a06f01177f7b0450b	0.1234
2	adExperience	0.04208
3	x0_38ea33e30ac13901f70acd131ea9c38f	0.03093
4	x0_8f0a5a99af1b07b165feaac0fa187869	0.02905
5	x0_ba88e95e23f980795c479fffbb31512a	0.02631
6	x0_cb28edd7446054e0f15d3906482d48c3	0.02356
7	x0_e022508e09ca7ce3befbbbd3ff7fc0e0	0.02088
8	x0_5bb959f684c08dc529c83d4c3f7c11bd	0.01964
9	x0_6f0d9275778e06cb8485934c238e25b8	0.01879
10	x0_526a8ff700825666d291932c379e6e44	0.01871
11	x0_8f42b59fbed1212ab31151a931649a25	0.01849
12	x0_446cec01fb12e5a395079dcb48337112	0.01721
13	x0_cbe767b73228c9dedbad2c45cebb4d45	0.01612
14	x0_4c6a32dc14c7efe763ea5a298493b660	0.01544
15	x0_5c0353036213bc1110d95a0bce68f3d8	0.01529
16	x0_15e970b260f45ac2b11aba911e234602	0.01317
17	x0_f64beb177b9d156d01b60d58f23aeb90	0.01240
18	x0_79b52d31fed4774e9930b8213eea5631	0.01182
19	x0_5747a3154641ed3355afa61d02dd4060	0.01181
20	x0_13f663aec99541c6a32938f40abdb964	0.01165

Tabela 4.9: Ocenjevanje atributov z RReliefF-om, spletne strani so predstavljene kot binarni atributi s predpono x0_ in nato zgoščena vrednost strani.

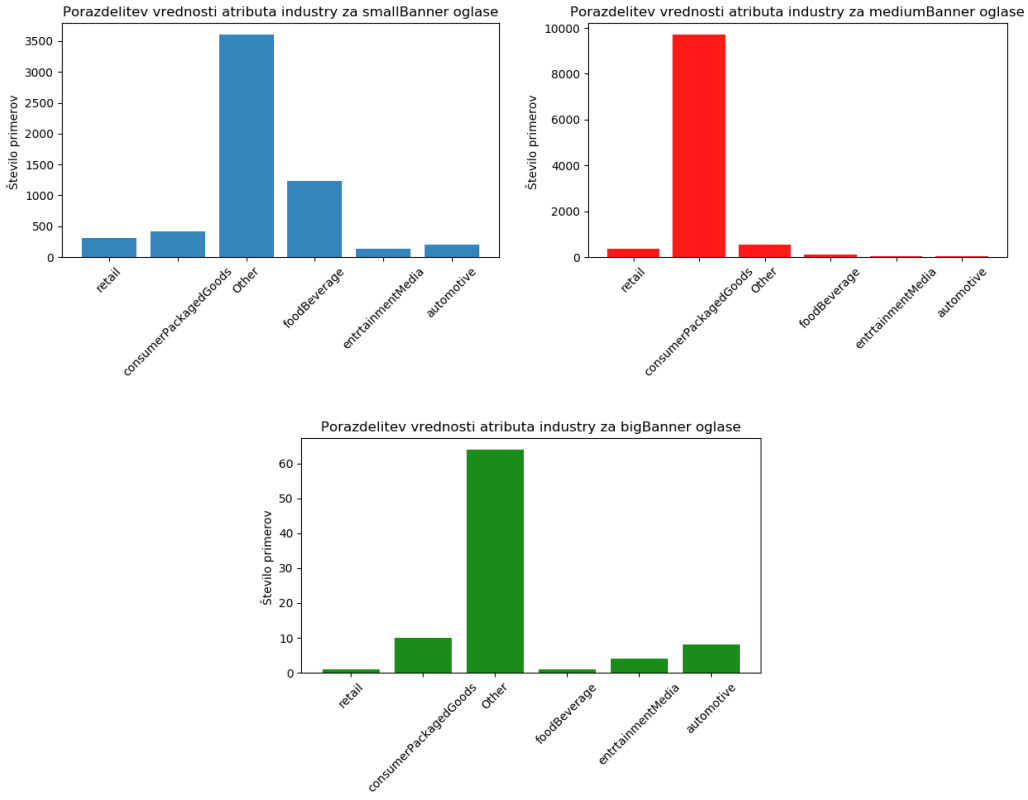
številka	RReliefF	
	Atribut	ocena
1	site	14.714e-02
2	adExperience	6.0680e-02
3	creativeComplexityRank	3.7965e-02
4	isCreatedFromCopy	1.3368e-02
5	hasVisualEffect	0.4965e-02
6	x2_pharmaceutical	0.3212e-02
7	x2_consumerPackagedGoods	0.3025e-02
8	x2_quickServiceRestaurants	0.2498e-02
9	x2_unknown	0.2162e-02
10	hasDynamicContent	0.1704e-02
11	hasLocation	0.1668e-02
12	hasCreativity	0.1509e-02
13	x2_finance, insuranceRealEstate	0.0593e-02
14	x2_education	0.0234e-02
15	x2_travelHospitality	0.0119e-02
16	x2_retail	0.0041e-02
17	x2_entertainmentMedia	0.0003e-02
18	x1_foodBeverage	0.000e+00
19	x2_other	0.000e+00
20	x1_retail	-0.0012e-02

Tabela 4.10: Ocenjevanje atributov z RReliefF-om, kjer je site atribut, ki predstavlja kategorizacijo strani, atributi s predponama x1_ in x2_ predstavljajo binarizirane vrednosti atributa industry in effectiveIndustry.



Slika 4.2: Primerjava mediane in povprečja za vrednosti atributa adExperience.

ekstremne vrednosti. Zanimivo je, da so srednje veliki oglasi dosegli najmanjšo povprečno vrednost in najmanjšo mediano, saj domenski strokovnjaki pravijo, da velikost oglasa pozitivno vpliva na stopnjo interakcij z njim. Na sliki 4.3 vidimo porazdelitev oglasov po industrijah glede na velikost oglasa. Opazimo, da pri srednje velikih oglasih najbolj prevladujejo oglasi za industrijo consumerPackaged-Goods, pri majhnih in velikih pa oglasi, ki nimajo določene industrije (vrednost 'other' v tabelah) . Oglasi z industrijo consumerPackagedGoods pa dosegajo najmanjše CTR vrednosti, medtem ko oglasi, ki nimajo določene industrije, dosegajo večje vrednosti CTR. Tako lahko pojasnimo zakaj v našem primeru majhni oglasi dosegajo večje vrednosti CTR kot srednje veliki oglasi. Pri ocenjevanju z razliko variance se visoko uvrsti tudi nekaj določenih industrij in pa edini preostali nebinarni atribut creativeComplexityRank, ki se je visoko uvrstil tudi pri ocenjevanju z RReliefF-om, in ga zato lahko štejemo med pomembnejše atribute.



Slika 4.3: Porazdelitve atributa industry glede na velikost oglasa.

številka	Ocenjevanje z varianco	
	Atribut	ocena
1	adExperience	2.2194e-05
2	x1_consumerPackagedGoods	1.6085e-05
3	x1_foodBeverage	0.9363e-05
4	x2_beautyPersonalCare	0.8000e-05
5	x1_other	0.6955e-05
6	x0_72580a17fce727b9abb2ed54bffab786	0.5211e-05
7	x0_ff22f7f7826c8cdb9e55a8134fbd161d	0.4370e-05
8	creativeComplexityRank	0.3919e-05
9	x2_telecommunications	0.3350e-05
10	x0_5ea3b4cffa8ee681becaba3d427220bf	0.3036e-05
11	x2_quickServiceRestaurants	0.2617e-05
12	x0_a571c0dd8aae0032a614f63305084814	0.2443e-05
13	x0_5c0353036213bc1110d95a0bce68f3d8	0.1683e-05
14	x0_2b3f012ca20a607a06f01177f7b0450b	0.1322e-05
15	hasDynamicContent	0.1199e-05
16	x2_unknown	0.1045e-05
17	x0_2b5c97efb1a5e588510e07f637933afc	0.0931e-05
18	x0_6f0d9275778e06cb8485934c238e25b8	0.0897e-05
19	x0_07fa81a807deb23c52afb79c49dd6328	0.0821e-05
20	x0_01de982cd19dafb2ca636643665060a2	0.0819e-05

Tabela 4.11: Ocenjevanje atributov z razliko variance, spletne strani so predstavljene kot binarni atributi s predpono x0_, atributi s predponama x1_ in x2_ predstavljajo binarizirane vrednosti atributa industry in effectiveIndustry. Varianca na celotni množici je 8.551e-05.

številka	Ocenjevanje z varianco	
	Atribut	ocena
1	site	5.2381e-05
2	adExperience	2.2194e-05
3	x1_consumerPackagedGoods	1.6085e-05
4	x1_foodBeverage	0.9363e-05
5	x2_beautyPersonalCare	0.7880e-05
6	x1_other	0.6955e-05
7	creativeComplexityRank	0.3919e-05
8	x2_telecommunications	0.3350e-05
9	x2_quickServiceRestaurants	0.2617e-05
10	hasDynamicContent	0.1199e-05
11	x2_unknown	0.1045e-05
12	x2_technology	0.0684e-05
13	hasPersonalization	0.0680e-05
14	x2_shippingLogistics	0.0593e-05
15	hasLocation	0.0573e-05
16	x2_entertainmentMedia	0.0475e-05
17	x1_retail	0.0465e-05
18	x1_automotive	0.0440e-05
19	x2_automotive	0.0435e-05
20	x2_government	0.0330e-05

Tabela 4.12: Ocenjevanje atributov z razliko variance, kjer atribut site predstavlja kateogrizacijo spletnih strani, atributi s predponama x1_ in x2_ predstavljajo binarizirane vrednosti atributa industry in effectiveIndustry. Varianca na celotni množici je 8.551e-05.

Poglavje 5

Zaključek

V diplomski nalogi smo želeli ugotoviti, kateri atributi najbolj vplivajo na stopnjo interakcij z oglasi (*click-through rate*) in zgraditi čim bolj točen regresijski model, s katerim bi reševali problem hladnega zagona in napovedali stopnjo interakcij z oglasom na določeni spletni strani za oglase, ki v preteklosti še nikjer niso bili prikazani. Za ocenjevanje atributov smo uporabili metodi RReliefF in razliko variance ter ugotovili, da na CTR najbolj vpliva velikost oglasa in spletna stran, na kateri je oglas prikazan.

Za gradnjo regresijskih modelov smo uporabili metodi naključnega gozda in k-najbližjih sosedov. Podatke smo pripravili na več različnih načinov in ugotovili, da metoda naključnega gozda najbolje deluje pri predstavitvi spletne strani kot atributa, metoda k-najbližjih sosedov pa pri izgradnji napovednega modela za vsako spletno stran posebej. Vse napovedne modele smo testirali s prečnim preverjanjem s sprehodom naprej. Glede na metriko RRMSE se je najbolje obnesel naključni gozd z binarizacijo strani, natreniran z dvomesečno učno množico, in je dosegel RRMSE = 0.6399. Pri upoštevanju metrike RAE pa se je najbolje obnesel model k-najbližjih sosedov, ki je bil zgrajen na vsaki posamezni strani, s parametrom k = 3 in primernimi utežmi za atribute, in je bil natreniran na dvomesečni učni množici, dosegel pa je RAE = 0.3556.

Ker stran najbolj vpliva na CTR, smo modele testirali tudi na bolj omejeni testni množici, ki vsebuje le tiste oglase, ki oglašujejo na spletni strani, na kateri je

v preteklosti bilo prikazanih vsaj 15 različnih oglasov. S tem smo se želeli izogniti problemu hladnega zagona za spletne strani. Na tej testni množici je glede na metriko RAE najboljši rezultat dosegel model k-najbližjih sosedov, ki je bil zgrajen na vsaki posamezni strani, s parametrom $k = 3$ in primernimi utežmi za atribute, in je bil natreniran na dvomesečni učni množici, dosegel je RAE = 0.3058. Ta model se je prav tako najbolje obnesel glede na metriko RRMSE, ki je bil prav tako natreniran na dvomesečni učni množici, dosegel je RRMSE = 0.4961. Metoda kNN z modeli na posameznih spletnih straneh je dosegla najboljše rezultate, saj se je za vsak posamezen model povečala učna množica oglasov, med katerimi je kNN iskal najbolj podobne. Še vedno pa so bile to majhne učne množice, premajhne, da bi naključni gozd uspel najti skrite vzorce in doseči boljše rezultate. Zanimalo nas je tudi ali s povečevanjem učne množice vedno dobimo boljše rezultate, zato smo najboljše modele naučili tudi na tromesečni učni množici in ugotovili, da tromesečna učna množica poslabša rezultate, saj je časovno obdobje predolgo in s tem model upošteva časovno odvisnost stopnje interakcij. Trend stopnje interakcij pa se skozi leto spreminja in zato preveliko obdobje negativno vpliva na točnost modelov.

Rezultate smo želeli izboljšati z izvedbo matrične faktorizacije in z izvedbo matrične faktorizacije z dodajanjem na obstoječih napovedih, ki smo jih dobili s prejšnjimi napovednimi modeli. Pri matrični faktorizaciji nam najboljših rezultatov ni uspelo izboljšati, so se pa izboljšali nekateri slabši rezultati. Pri matrični faktorizaciji z dodajanjem novih elementov pa se je postopek vedno ujel v lokalni minimum in prav tako nismo izboljšali obstoječih rezultatov.

V prihodnje bi lahko naše napovedi izboljšali s pridobitvijo dodatnih lastnosti o oglasih ali z distančno matriko med industrijami, ki bi jo podali domenski strokovnjaki. Prav tako bi lahko pomagali podatki o spletnih straneh. Dodatne izboljšave bi lahko prinesla tudi gradnja hibridnega napovednega modela, ki bi kombiniral napovedi večih osnovnih modelov in matrične faktorizacije, kot je recimo opisano v [2].

Literatura

[1] Digital ad spending 2019. Dosegljivo: https://www.emarketer.com/content/global-digital-ad-spending-2019, 2019. [Dostopano: 21.10.2019].

[2] Javad Basiri, Azadeh Shakery, Behzad Moshiri, and Morteza Hayat. Alleviating the cold-start problem of recommender systems using a new hybrid approach. *2010 5th International Symposium on Telecommunications, IST 2010*, December 2010.

[3] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[4] Jakub Bartczuk (https://stats.stackexchange.com/users/121270/jakub bartczuk). Matrix factorization in recommender systems: adding a new user. Dosegljivo: https://stats.stackexchange.com/q/321122. [Dostopano: 25.11.2019].

[5] Jatin Garg (https://stats.stackexchange.com/users/123886/jatin garg). Using k-fold cross-validation for time-series model selection. Dosegljivo: https://stats.stackexchange.com/q/268847. [Dostopano: 29.11.2019].

[6] Courtney Cochrane (https://towardsdatascience.com/@ccochrane_17940). Time series nested cross-validation. Dosegljivo: https://towardsdatascience.com/time-series-nested-cross-validation-76adba623eb9, Maj 2018. [Dostopano: 25.11.2019].

[7] James Moody (https://towardsdatascience.com/@james.moody). What does rmse really mean? Dosegljivo: https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e, September 2019. [Dostopano: 25.11.2019].

[8] Tumas Rackaitis (https://towardsdatascience.com/@trackait). Introduction to latent matrix factorization recommender systems. Dosegljivo: https://towardsdatascience.com/introduction-to-latent-matrix-factorization-recommender-systems-8dfc63b94875, May 2019. [Dostopano: 14.01.2020].

[9] TAVISH SRIVASTAVA (https://www.analyticsvidhya.com/blog/author/tavish1/). Introduction to k-nearest neighbors: A powerful machine learning algorithm. Dosegljivo: https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/, Marec 2018. [Dostopano: 25.11.2019].

[10] Rob J. Hyndman. Why every statistician should know about cross-validation. Dosegljivo: https://robjhyndman.com/hyndsight/crossvalidation/, Oktober 2010. [Dostopano: 29.11.2019].

[11] I. Kononenko and M. Robnik Šikonja. *Inteligentni sistemi*. Založba FE in FRI, 2010.

[12] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug 2009.

[13] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of ReliefF and RRelieff. *Machine Learning*, 53(1):23–69, Oct 2003.

[14] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. 2019. [Dostopano: 25.11.2019].

[15] Blaž Zupan. uozp-zapiski. Dosegljivo: https://github.com/BlazZupan/uozp-zapiski/blob/master/zapiski.pdf, 2018. [Dostopano: 28.10.2019].