Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Robert Dovžan

# Napovedovanje razmerja med prikazi in kliki oglasov s faktorizacijskimi metodami

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Lovro Šubelj
Somentor: Davorin Kopič

Ljubljana, 2019

*Besedilo je oblikovano z urejevalnikom besedil LaTeX.*

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomski nalogi preučite domeno programatičnega spletnega oglaševanja v realnem času. Posebej se osredotočite na napovedovanje razmerja med prikazi in kliki oglasov. Predlagajte lastno rešitev za napovedovanje verjetnosti klika z uporabo faktorizacijskih metod na podlagi podatkov o oglasu, uporabniku in spletnem mestu. Rešitev implementirajte in preizkusite v produkcijskem okolju podjetja Zemanta d.o.o. ter jo primerjajte z obstoječo rešitvijo, ki uporablja logistično regresijo. Rezultate testiranja kritično ovrednotite in predlagajte možnosti za nadaljnje delo.

*Na tem mestu se zahvaljujem mentorju doc. dr. Lovru Šublju za odlično odzivnost in strokovno pomoč pri izdelavi diplomske naloge. Enaka zahvala gre somentorju Davorinu Kopiču in podjetju Zemanta d.o.o., ki mi je omogočilo študentsko prakso in s tem izdelavo te diplomske naloge. Posebna zahvala pa gre mojemu dekletu in družini. Hvala za neizmerno podporo, vzpodbudo in pomoč, ki ste mi jo nudili na poti do cilja.*

# Kazalo

# Seznam uporabljenih kratic

| kratica | angleško | slovensko |
|---|---|---|
| **CTR** | click-through rate | razmerje med prikazi in kliki |
| **CPM** | cost-per-mille | cena za tisoč prikazov |
| **CPC** | cost-per-click | cena za klik |
| **CPA** | cost-per-action | cena za opravljeno akcijo |
| **RTB** | real-time bidding | izbira oglasov v realnem času |
| **GDPR** | general data privacy regulation | splošna ureditev zasebnosti podatkov |
| **DSP** | demand-side platform | okolje za povpraševanje |
| **SSP** | supply-side platform | okolje za prodajo |
| **DMP** | data-management platform | okolje za upravljanje podatkov |
| **FM** | factorization machines | faktorizacijske metode |
| **LR** | logistic regression | logistična regresija |

# Povzetek

**Naslov:** Napovedovanje razmerja med prikazi in kliki oglasov s faktorizacijskimi metodami

**Avtor:** Robert Dovžan

V panogi programatičnega spletnega oglaševanja, ki temelji na ekosistemu izbire oglasov v realnem času (angl. *Real-Time Bidding*) je pomembno napovedati kako uspešen bo prikaz oglasa uporabniku. Napovedovanje razmerja med prikazi in kliki (angl. *Click-Through Rate prediction*) oziroma verjetnosti klika je eden večjih izzivov v spletnem oglaševanju. V diplomski nalogi se lotimo napovedovanja verjetnosti z uporabo faktorizacijskih metod na podlagi podatkov, ki jih poznamo o oglasu, spletni strani, uporabniku ipd. Opišemo celoten proces obdelave podatkov, izbire značilk, implementacije in testiranja. Cilj naloge je v podjetju Zemanta d.o.o. izboljšati obstoječo rešitev, ki temelji na logistični regresiji. Z lokalnim testiranjem in testiranjem v produkcijskem okolju v obliki A/B testa naš cilj dosežemo in s tem prispevamo k izboljšanju storitve in večjemu finančnemu izkupičku podjetja.

**Ključne besede:** faktorizacijske metode, oglaševanje, oglasi, napovedovanje, strojno učenje, podatkovno rudarjenje.

# Abstract

**Title:** Predicting the click-through rate of ads using factorization machines

**Author:** Robert Dovžan

In the field of programmatic advetising based on the ecosystem called real-time bidding, it is important to know, how successful an ad impression will be. Click-through rate prediction is one of the biggest challenges in online advertising. In this thesis we use factorization machines to predict the click-through rate based on data about the ad, website, user etc. We describe the process of data preparation, feature selection, implementation and testing. The goal is to improve the current solution in company Zemanta d.o.o. which is based on logistic regression. With local testing and online A/B testing we reach our goal and contribute to improving the service and financial performance of the company.

**Keywords:** factorization machines, advertising, ads, prediction, machine learning, data mining.

# Poglavje 1

# Uvod

Podjetij, ki v današnjem času ne nastopajo na spletu skoraj ni več. Mnoga izmed njih preko spleta poslujejo in s tem ustvarjajo večino svojega prihodka. Tam se odvijajo prava tekmovanja v tem, čigava sporočila bodo bolj izpostavljena med oglasi, na spletnih iskalnikih, socialnih omrežjih in drugih spletnih straneh. Vse to zato, da bi na svojo stran privabili kar se da veliko obiskovalcev in prodali čim več. Oglaševanje na spletu je zanimivo področje, ki je v zadnjih letih postalo glavni medij za širjenje oglasnih sporočil [11]. Ciljanje konkretnih uporabnikov in enostavno povečevanje obsega sta dva izmed razlogov, zakaj je spletno oglaševanje za podjetja danes privlačno povsod po svetu. Obstaja več načinov, kako se oglasi preko spleta širijo, prikazujejo in zaračunavajo. V diplomskem delu se bolj podrobno osredotočimo na področje programatičnega oglaševanja, ki deluje po principu izbire oglasov v realnem času oziroma RTB (angl. *Real-Time Bidding*). V tem poglavju predstavimo kratek uvod v spletno oglaševanje in podrobnosti o delovanju ekosistema RTB. Naš problem rešujemo s pomočjo strojnega učenja, zato pripravimo tudi kratek uvod v to področje. V ekosistemu RTB oglasi med seboj tekmujejo za oglasne prostorčke, ki so na spletnih straneh na voljo poleg dejanske vsebine. Pomembno je, da znamo oceniti kako perspektiven je določen oglas na določeni spletni strani za določenega uporabnika. Kazalec, ki nam bo to kar se da dobro predstavil je razmerje med prikazi in

kliki oziroma CTR (angl. *Click-Through Rate*). Naš cilj je, kar se da dobro napovedati CTR in s tem omogočiti serviranje oglasov, ki imajo verjetnost za klik največjo. Metode strojnega učenja nam pomagajo iz preteklih podatkov razbrati značilnosti, ki vplivajo na končni rezultat. V diplomskem delu se bolj podrobno poglobimo v logistično regresijo, ki je enostaven in učinkovit princip za reševanje klasifikacijskih problemov, in faktorizacijske metode, ki so v zadnjem času med bolj uspešnimi na tekmovanjih iz odkrivanja znanj iz podatkov [17]. V 2. poglavju opišemo postopek obdelave podatkov, teoretično ozadje obeh napovednih modelov ter opis implementacije faktorizacijskih metod. V 3. poglavju opišemo postopke testiranj ter proces prehajanja iz lokalnega v spletno okolje. Cilj naloge je izboljšati sistem napovedovanja CTR v podjetju Zemanta d.o.o., ki trenutno temelji na prirejeni različici logistične regresije. Že najmanjše izboljšanje napovedi ima velik vpliv na kvaliteto storitve in poslovne rezultate. V 4. poglavju predstavimo rezultate ter jih interpretiramo, v zaključku pa nadomestitev obstoječega sistema potrdimo oziroma zavrnemo.

## 1.1 Spletno oglaševanje

V tem podpoglavju predstavimo področje spletnega oglaševanja, nekaj pozitivnih in negativnih lastnosti ter kako se lahko primerja s tradicionalnim oglaševanjem. V zadnjem delu povemo več o spletnem oglaševanju, ki temelji na ekosistemu RTB v katerem obratuje naš sistem.

Oglas je tržno sporočilo katerega namen je spodbuditi potencialno stranko k nakupu izdelka oziroma storitve [28]. Z vse večjo uporabo spleta so uporabniki zmožni izražati svoje zahteve po informacijah. Odkar so podjetja pričela množično prehajati v spletno okolje, je splet postal velika priložnost za oglaševalce pri razširitvi svojih potencialnih strank. Rezultat tega je, da je spletno oglaševanje ena izmed najhitreje rastočih panog v industriji informacijske tehnologije. Spletno oglaševanje je postalo eden izmed glavnih virov prihodka modernih podjetij. Prodajalci ga uporabljajo, da privabijo poten-

cialne stranke na svoje spletne strani. Tam obiskovalci brskajo po njihovih izdelkih in storitvah, ki jih lahko kupijo v nekaj klikih. Oglasi lahko nastopajo v najrazličnejših oblikah, na primer slika, besedilo, obvestilo, članek ipd. Sporočila se lahko nahajajo na mestih, kot so spletne strani, mobilne aplikacije, video posnetki ipd.

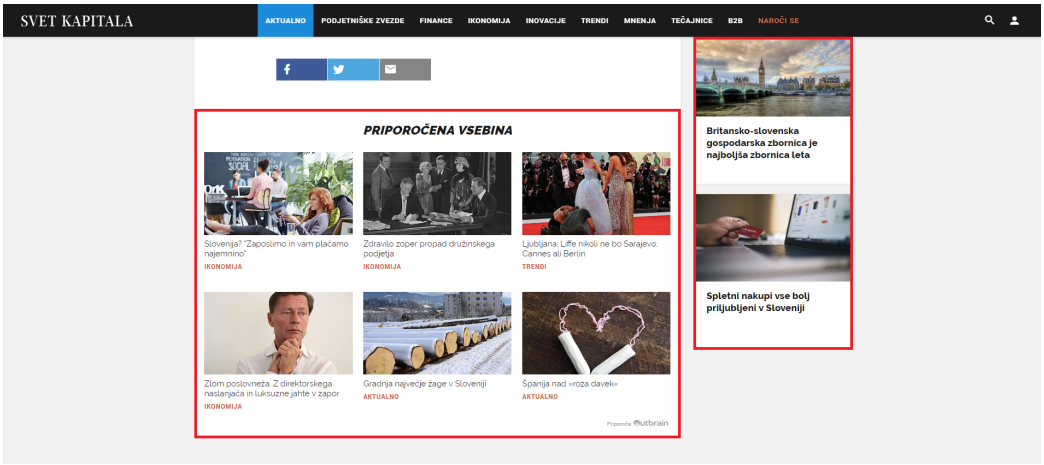
Spletno oglaševanje se razlikuje od tradicionalnega oglaševanja. Arhitektura spletnih komunikacij omogoča založnikom in oglasnim mrežam da izvedo veliko več o svojih uporabnikih, kot je bilo to mogoče v časih tiska, radia in televizije [11]. Na primer, spletni mediji ali oglasne mreže običajno vedo, če si posameznik ogleduje njihovo spletno stran v določenem času, v nasprotju z radijskimi postajami, ki imajo omejeno zmožnost prejemanja informacij o tem ali posameznik posluša njihove oglase ali ne. Spletni mediji se lahko naučijo pomembnih lastnosti o posamezniku. Vsak uporabnik spleta ima svoj IP (angl. *Internet Protocol*) naslov, ki običajno določa njihovo lokacijo. Spletni mediji in oglasne mreže lahko uporabijo ta naslov, da izvejo katere strani je ta uporabnik obiskal pred tem, in iz tega izvejo več informacij o njem. Za razliko od tradicionalnega oglaševanja, spletno omogoča ciljanje oglasov na stranke v določenih državah, regijah in mestih. Poleg IP naslova veliko spletnih oglaševalcev uporablja tako imenovane piškotke, ki jih uporabniki sprejmejo v svoje brskalnike. Ti omogočajo spletnim agencijam, da izvejo katere so bile pretekle spletne strani, ki jih je uporabnik obiskal. Dejstvo, da oglaševalci vedo, katere vsebine si uporabnik ogleduje, jim omogoča izvajati usmerjeno oglaševanje.

Ciljanje uporabnikov je izmed tehnik spletnega oglaševanja v zadnjem času požela največ pozornosti [29]. Izkorišča zgodovinsko vedenje uporabnika z namenom, da zanj izbere najprimernejši oglas. Uporaba osebnih podatkov za usmerjeno oglaševanje sproža več in več spornih vprašanj. GDPR (angl. *General Data Protection Regulation*) je ena izmed zadnjih uredb, ki določa nova pravila glede varstva osebnih podatkov na področju Evropske Unije [2]. Pravila zagotavljajo potrošnikom večjo moč nad osebnimi podatki, medtem ko so podjetja ki te podatke zbirajo dolžna upoštevati višji

nivo transparentnosti. Takšne uredbe lahko močno vplivajo na podjetja, ki hranijo kakršnekoli podatke o svojih potrošnikih.

Bistvo oglaševalske industrije je rešiti ogromen problem ujemanja: veliko število oglaševalcev, ki želi dostaviti veliko število oglasnih sporočil velikemu številu potrošnikov [11]. Spletno oglaševanje ponuja potencialno bolj učinkovit mehanizem povezovanja oglaševalcev in potrošnikov kot tradicionalni pristop. Vsaka spletna stran, ki privablja obiskovalce je potencialni dobavitelj oglaševalskega prostora.

Spletne strani se med seboj razlikujejo po načinu prikazovanja svojih oglasov. Nativno oglaševanje je uporaba oglasov na tak način, da je njihov izgled predstavljen kot del vsebine spletne strani oziroma aplikacije v kateri se prikazujejo. Največkrat jih najdemo na socialnih omrežjih in na delih spletnih strani, kjer uporabnik običajno išče članke za nadaljnje branje. Takšno oglaševanje je za uporabnika bolj prijazno od pojavnih oken ali vsiljenih video posnetkov.




Slika 1.1: Primer nativnega oglaševanja na dnu in ob robu spletne strani [7].

Podjetja, ki želijo koristiti oglaševalske storitve, največkrat to storijo preko oglaševalskih agencij [26]. Obstajajo trije glavni načini, kako agencije zaračunavajo oglaševalske storitve na spletu. Vsaka izmed njih ima prednosti in slabosti. Običajno je odvisno, kaj oglaševalec s prikazovanjem svoje

vsebine želi doseči. Tudi ko uporabnik ne klikne na oglas, obstaja verjetnost, da ga opazi in si ustvari mnenje o produktu oziroma storitvi.

- **CPM (angl. *Cost Per Mille*)** – Oglaševalci plačajo glede na to, kolikokrat se je njihov oglas prikazal uporabnikom. Običajno je cena postavljena glede na tisoč prikazov.
- **CPC (angl. *Cost Per Click*)** – Oglaševalci plačajo samo takrat, ko uporabnik klikne na njihov oglas in je uspešno preusmerjen na ciljno spletno stran. Cena je torej sorazmerna s številom klikov.
- **CPA (angl. *Cost Per Action or Acquisition*)** – Oglaševalci plačajo samo za tiste klike na oglas, kjer uporabnik na ciljni strani izvede kakšno izmed akcij na primer vpis elektronskega naslova, ustvarjanje računa ali nakup.

V CPC oglaševalskih sistemih so oglasi rangirani po eCPM (angl. *effective Cost Per Mille*), ki je produkt med ponujeno ceno za klik – CPC in razmerjem CTR [32]. CTR mora biti predviden s strani sistema. Zaradi tega ima izvedba napovedovanja CTR velik finančni pomen in igra pomembno vlogo v oglaševalskih sistemih. Modeliranju CTR napovedovanja je bilo posvečeno veliko pozornosti tako s strani znanstvenih raziskav kot s strani industrije.

Eno izmed ključnih vprašanj pri napovedovanju CTR je razpoložljivost in izbor primernih vhodnih značilk, ki omogočajo najbolj natančno napovedovanje za dan vtis oziroma prikaz oglasa [14]. Te značilke lahko razvrstimo v tri različne skupine:

- **Značilke oglasnega sporočila** – Ključne besede, naslov, opis, ciljna spletna stran, fotografija in njene lastnosti itd.
- **Značilke poizvedovanega niza** – Ključne besede, postopek za razširitev poizvedovanega niza itd.
- **Kontekstualne značilke** – Lastnosti pozicije prikaza, geografska lokacija, čas, podatki o uporabniku, zgodovina iskanja itd.

Seveda so to osnovne značilke, iz katerih zgradimo bolj kompleksne z modeliranjem vplivov med oglasnim sporočilom, poizvedovanim nizom in kontekstom. Bolj kompleksne značilke lahko ustvarimo na primer s kartezičnim produktom osnovnih značilk. Kot v vseh ostalih problemih strojnega učenja, sta izgradnja in izbira dobrih značilk dva izmed glavnih izzivov. Za učeči algoritem je pomembno, da dobro modelira diskretne značilke zelo različnih številnosti na primer spol lahko zasede dve različni vrednosti, medtem ko identifikacijska številka uporabnika lahko zasede čez milijardo različnih vrednosti. Medtem ko je napovedovanje CTR v bistvu problem sklepanja, se bo uspešnost sistema za izbor oglasov merila v smislu sprejetih odločitev. Še več, ker je napoved CTR uporabljena pri izboru oglasov skozi dražbo, napoved sistema določa prisotne oglase v bodočih učnih primerih. Mehanizem izbora oglasov mora zato na nekakšen način nasloviti problem raziskovanja oziroma izkoriščanja t.j. požrešen algoritem se bo prej ali slej znašel v lokalnem maksimumu, ker zanemari prednost raziskovanja celotne zaloge oglasov, kar se na dolgi rok ne obnese.

Poleg vseh prednosti, ki jih ima spletno oglaševanje pred tradicionalnim pa se mora soočati tudi z nekaj izzivi. Eden izmed večjih je zagotovo onemogočanje oglasov na spletu s strani uporabnikov. Uporabniki si lahko v svoj spletni brskalnik namestijo vtičnik, ki prepreči nalaganje večine oglasov. Ta vrsta programske opreme je brezplačna in je na voljo za vse vrste naprav, ki lahko tako ali drugače brskajo po spletu. Za potrošnika uporabljanje tovrstne programske opreme omogoča boljšo uporabniško izkušnjo in večjo varnost v smislu neželenih preusmeritev in deljenja osebnih podatkov. Lastnikom spletnih strani pa blokiranje oglasov s strani njihovih uporabnikov ne prinaša pozitivnih učinkov. Da bi uporabljanje takšnih orodij kar se da zmanjšali, nekatere spletne strani ne dovolijo ogleda vsebine, če zaznajo, da uporabnik uporablja tovrstno orodje.

### 1.1.1 Programatično spletno oglaševanje

Programatično kupovanje oglasov in dostavljanje njih s pomočjo analize velike količine podatkov je nadomestilo tradicionalni vzorec, ki je temeljil na pogajanjih in pogodbah za veliko količino vtisov [30]. Ta se je prelevil v sistem, ki je prilagodljiv in občutljiv glede na ciljno občinstvo, in temelji na spletnem kupovanju in dostavljanju oglasov. Ta inovacija omogoča popolen izkoristek vrednosti podatkov in trga spletnega oglaševanja, obenem pa občutno poveča njegovo učinkovitost.

Vodja programerjev na IAB UK pravi: "*Začelo se je kot način uporabe preostanka oglaševalskega prostora in način povečanja učinkovitosti na ostankih. Kasneje se je to razvilo v nekaj povsem prefinjenega [5]. Obstaja pogost nesporazum, da celotno področje programatičnega spletega oglaševanja spada v ekosistem izbiranja oglasov v realnem času. Oglaševanje v realnem času je samo podmnožica programatičnega spletnega oglaševanja in s tem samo način uporabe programatičnega oglaševanja za izvedbo takojšnjega nakupa oglaševalskega prostora.*"

### 1.1.2 Izbira oglasov v realnem času

V zadnjih nekaj letih je zahteva po avtomatizaciji, integraciji in optimizaciji glavno gonilo tega, da je spletno oglaševanje eno najhitrejše rastočih področij [10]. V prikaznem oglaševanju je razlog za največji razvoj pojav RTB, ki omogoča kupovanje in prodajanje prikaznih vtisov v realnem času.

Uporaba RTB je v zadnjem obdobju sunkovito narasla [30]. Poročali so, da je na mednarodnih trgih 88% severno-ameriških oglaševalcev v letu 2011 presedljalo na RTB. Pričakovana rast RTB trga je za leto 2017 znašala 8,49 milijarde dolarjev, kar je predstavljalo 29% proračuna celotnega prikaznega oglaševanja. Na Kitajskem je RTB v uporabi od leta 2011. Poročajo, da je leta 2013 količina RTB zahtevkov dosegla 5 milijard vtisov, proračun RTB pa se je povečal za 300% kar znaša 83 milijard dolarjev. RTB postaja standardni poslovni model za spletno oglaševanje.

RTB je ekosistem, ki omogoča oglaševalcem, da ponudijo znesek za posamezen prikaz oglasa določenemu uporabniku v realnem času [31]. Ta princip presega kontekstualno oglaševanje s tem, da motivira ponujanje zneska na podlagi podatkov o uporabniku. Razlikuje se od sponzorirane iskalne dražbe, kjer je znesek ponudbe odvisen od ključne besede. Največji tehnični izziv okolj za povpraševanje je avtomatizacija procesa ponujanja, ki temelji na proračunu, cilju kampanije in več drugih informacij zbranih pred in v času izvajanja. RTB je spremenil nekaj temeljnih dejavnikov v zvezi s prikaznim oglaševanjem. Olajšal je proces nakupovanja večje količine oglaševalskega prostora in omogočil dostop do podatkov v realnem času. Zaradi tega je možno ciljano oglaševanje, česar posledica je serviranje oglasov, ki so osredotočeni na podatke o uporabniku namesto na kontekstualne podatke.

V RTB ekosistemu nastopa več različnih delov programske opreme, ki med seboj komunicirajo in definirajo njegov obstoj. Na skrajnem začetku je naključen uporabnik interneta, na skrajnem koncu pa v idealnem primeru podjetje s produktom ali storitvijo, ki je v interesu tega uporabnika. Da je med njima mogoča komunikacija obstajajo naslednji sestavni deli:

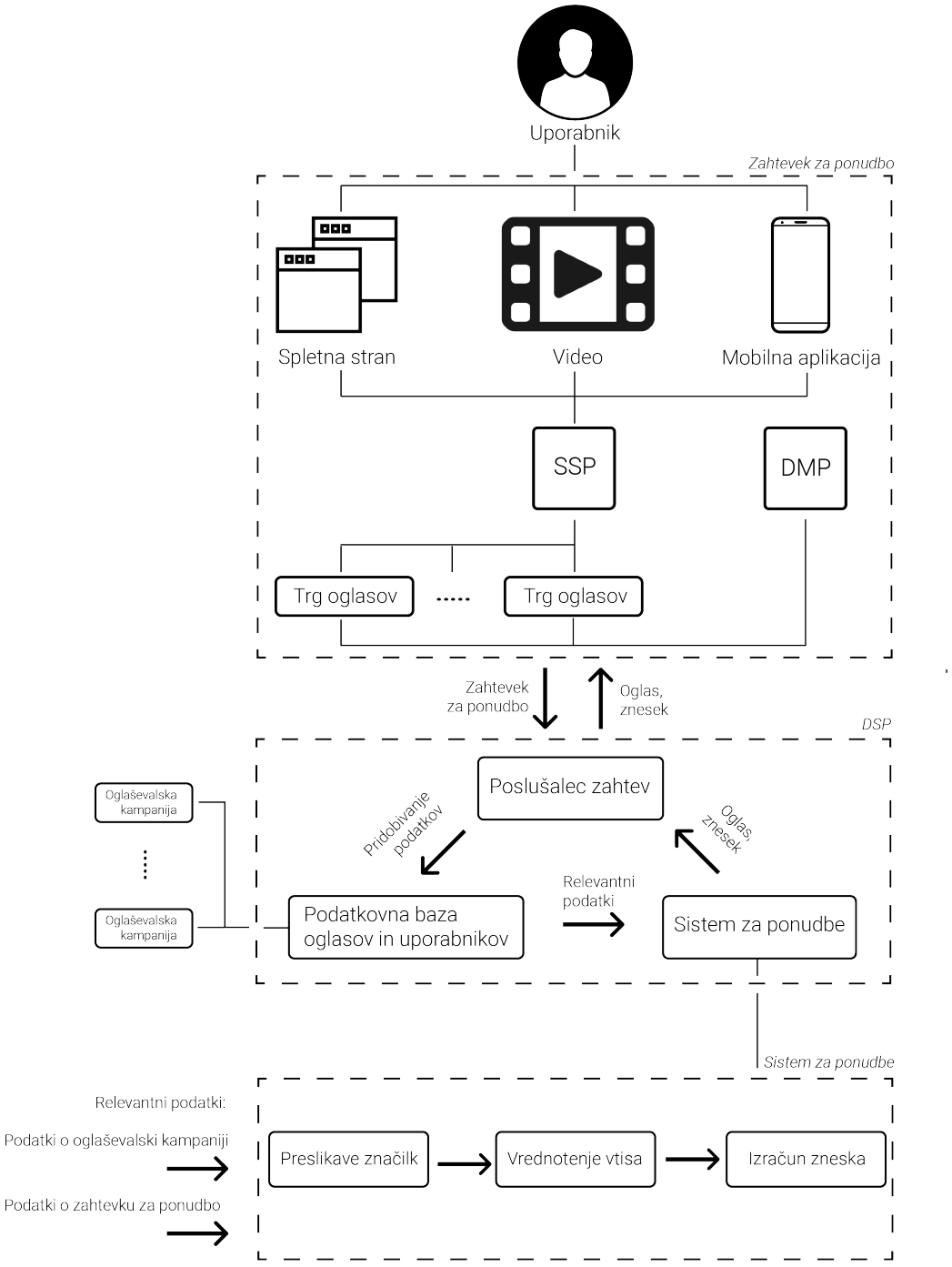
- **Okolje za povpraševanje (angl. *Demand-side platform*)** – Pomaga oglaševalcem pri upravljanju njihovih kampanj in optimiziranju RTB aktivnosti. Preko vgrajene aplikacije, ki je le del celotnega okolja, omogoča vpogled v preteklo dogajanje, dodajanje vsebine, nastavljanje proračuna, targetiranje itd. Poleg aplikacije vsebujejo tudi glavni sestavni del – udeleženca dražbe. Ta del programske opreme je zadolžen za izbiro oglasa in določitev cene, ki jo je pripravljen ponuditi na trgu oglasov za določen vtis. Celoten proces, ki se zgodi na strani DSP, je bolj podrobno opisan na naslednji strani.

- **Okolje za prodajo (angl. *Supply-side platform*)** – Založnikom omogoča upravljanje njihovega oglaševalskega prostora. Njihova naloga je, da zapolnjujejo oglasni prostor in s tem ustvarjajo prihodek.

- **Okolje za upravljanje podatkov (angl. *Data management platform*)** – Zbira in upravlja podatke o uporabnikih spleta, ki so koristni tako za oglaševalce kot ponudnike. Omogoča analizo podatkov na veliko večji skali in s tem dostavljajo SSP-jem in DSP-jem koristne informacije o njihovem prometu.

- **Trg oglasov (angl. *Ad exchange*)** – Trg oglasov deluje kot borza. Na njej se sreča ogromno število DSP-jev in SSP-jev, ki sodelujejo v dražbah. DSP-jem omogočajo, da na enem mestu kupijo vtise iz najrazličnejših spletnih strani, medtem ko SSP-jem omogočajo, da na enem mestu prodajo svoje vtise najrazličnejšim oglaševalcem. Sistem omogoča obema stranema prihranek tako na času kot denarju. Poleg javnih trgov oglasov, kjer vtise lahko kupi praktično kdorkoli, obstajajo tudi privatni, preko katerih lahko SSP-ji svoje vtise prodajajo samo točno določenim DSP-jem, s katerimi imajo najverjetneje sklenjen dogovor.

Slika 1.2 opisuje vlogo okolja za povpraševanje v RTB ekosistemu [31]. Ko uporabnik obišče spletno stran, se ustvari vtis, za katerega se sproži zahtevek za ponudbo s strani ponudnika oglaševalskega prostora navadno preko okolja za prodajo. Zahtevek se preko trga oglasov razpošlje okoljem za povpraševanje. Na strani okolja za povpraševanje se izračuna ponujen znesek za ta vtis in vrne odgovor trgu oglasov, kjer poteka dražba. Sledi obvestilo o morebitni zmagi in prikaz oglasa uporabniku. Bolj natančno, po prejetju zahtevka za ponudbo mora okolje za povpraševanje najti najustreznješe oglase iz vseh kampanj in izračunati ponujen znesek za vsako izmed njih. Okolje za povpraševanje uporablja tako kontekstualne (domena, spletna stran, ključne besede, datum in ura, geografska lokacija, vreme, jezik, operacijski sistem, brskalnik, itd.) kot značajske (zgodovina iskanja, brskanja in kupovanja, poklic, prihodek itd.) podatke za izračun zneska. Pogosto in priporočljivo je, da oglaševalci kupujejo podatke o interesih uporabnikov od tretjega ponudnika podatkov. Izračun zneska je najpomembnejši problem vsakega okolja za povpraševanje.




Slika 1.2: Shema arhitekture ekosistema RTB s poudarkom na DSP.

Celoten proces se ne zaključi na tej točki. V primeru, da uporabnik na servirani oglas klikne, ob tem v svoj brskalnik pridobi piškotek, s katerim okolje za povpraševanje beleži opravljene akcije na strani, kamor ga je oglas preusmeril. Te akcije so lahko na primer vpisan elektronski naslov, ustvarjen račun ali opravljen nakup. Med seboj jih ločimo po težavnosti, velikokrat pa služijo tudi kot indikator uspešnosti kampanj. Ob tem je vredno omeniti še eno pomembno količino, t.j. razmerje med opravljenimi akcijami oziroma konverzijami in vsemi kliki (angl. *Conversion rate*). Okoljem za povpraševanje ta predstavlja informacijo o tem, kako kvaliteten in relevanten promet preusmerja na ciljno spletno stran.

## 1.2 Strojno učenje

V tem podpoglavju predstavimo kratek uvod v strojno učenje, problem ki ga naslavljamo in dve metodi, ki ju lahko uporabimo za njegovo reševanje.

Strojno učenje (angl. *machine learning*) je veja raziskav umetne inteligence, ki je v zadnjih desetletjih močno napredovala, kar se odraža v številnih komercialnih sistemih za strojno učenje in njihovi uporabi v industriji, medicini, ekonomiji, naravoslovnih in tehničnih raziskavah, ekologiji, bančništvu itd. [18]. Strojno učenje se uporablja za analizo podatkov, odkrivanje zakonitosti v podatkovnih bazah imenovano podatkovno rudarjenje (angl. *data mining*), za avtomatsko generiranje baz znanja za ekspertne sisteme, za učenje načrtovanja, igranje iger, za gradnjo numeričnih in kvalitativnih modelov, za razpoznavanje naravnega jezika in prevajanje, klasifikacijo besedil in spletno rudarjenje, za avtomatsko ekstrakcijo znanja dinamične kontrole procesov, razpoznavanje govora, pisave, slik itd. Osnovni princip strojnega učenja je avtomatsko opisovanje oziroma modeliranje pojavov iz podatkov. Rezultat učenja so pravila, funkcije, relacije, sistemi enačb, verjetnostne porazdelitve ipd., ki so predstavljene z različnimi formalizmi kot na primer odločitvenimi pravili, odločitvenimi drevesi, regresijskimi drevesi, Bayesovimi mrežami, ne-

vronskimi mrežami itd. Naučeni modeli poskušajo razložijo podatke in se lahko uporabljajo za podporo pri odločanju enakega procesa (napovedovanje, diagnosticiranje, nadzor, preverjanje, simulacije itd.). Metode strojnega učenja delimo glede na način uporabe naučenega znanja na primer uvrščanje, regresija, učenje asociacij in logičnih relacij, učenje sistemov diferencialnih enačb in gručenje. Ena najpogostejših uporab strojnega učenja je klasifikacija ali uvrščanje. Naloga klasifikatorja je za objekt oziroma problem, opisan z množico lastnosti določiti, 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 ciljni razred glede na vrednosti neodvisnih spremenljivk.

Učenje in napovedovanje uporabnikovega odziva je ključno pri priporočanju vsebine, spletnem iskanju in spletnem oglaševanju [28]. Cilj učenja je predvideti verjetnost, da se bo uporabnik odzval s klikom, branjem ali konverzijo v danem kontekstu. Napovedana verjetnost je indikator uporabnikovega interesa za določen članek, spletno stran ali oglas. Ta lahko vpliva na kasnejše odločitve oziroma akcije. Če za primer vzamemo spletno oglaševanje, je razmerje med prikazi in kliki kasneje uporabljeno za izračunavanje cene ponudbe na dražbah oglasov. Naša želja je pridobiti kar se da natančno napoved, ne samo zaradi uporabniške izkušnje, temveč tudi zaradi povečanja obsega in pridobljene vrednosti oglaševalcev. Za RTB prikazno oglaševanje, ki je usmerjeno v učinkovitost, sta napovedovanje razmerja med prikazi in kliki in razmerja konverzij ključna sestavna dela, iz katerih sledi celotna strategija ponujanja, ki na koncu določa kako uspešne bodo RTB kampanije.

# Poglavje 2

# Metode in podatki

V tem poglavju predstavimo teoretično ozadje logistične regresije in faktorizacijskih metod. Opisan je proces obdelave podatkov, kar vključuje predpripravo in izbiro tistih atributov, ki najbolj pripomorejo h končni natančnosti našega napovednega modela. Sledi opis implementacije faktorizacijskih metod s podrobnejšim opisom ključnih funkcij.

## 2.1 Metode

### 2.1.1 Logistična regresija

V statistiki je logistični model široko uporabljena metoda, katere osnovna oblika uporablja t.i. logistično funkcijo za modeliranje binarne odvisne spremenljivke [8]. Obstaja več kompleksnejših nadgradenj na primer, ko odvisna spremenljivka lahko zasede več kot samo dve vrednosti. V regresijski analizi logistična regresija ocenjuje parametre logističnega modela. Logistična regresija se uporablja na več različnih področjih na primer strojno učenje, medicina, družbene vede, itd. V medicini se lahko na podlagi značilnosti pacienta uporablja pri napovedovanju tveganja za razvoj bolezni. Široko je uporabljena tudi v inženirskih praksah, posebej pri napovedovanju verjetnosti napake v danem procesu, sistemu ali produktu.

V primeru problema napovedovanja razmerja CTR želimo napovedati

verjetnost za klik, zato je logistična regresija primerno orodje tudi za področje spletega oglaševanja. Eden izmed glavnih razlogov za uporabo je enostavna interpretacija napovedi modela s pomočjo koeficientov v danem stanju. Pri učenju velike količine podatkov imajo enostavni linearni modeli kot na primer logistična regresija veliko prednosti [22]. Naj bo $\mathbf{x}$ vektor značilk. Čeprav ima $\mathbf{x}$ lahko več milijard dimenzij, bo tipična instanca imela zgolj nekaj sto vrednosti različnih od nič. To omogoča učinkovito učenje na velikih zbirkah podatkov s tokom podatkov z diska ali preko omrežja, saj mora biti vsak primer obravnavan zgolj in samo enkrat.

Če želimo naš problem modelirati z logistično regresijo, potem v $t$-ti iteraciji izvedemo napoved na podlagi vektorja značilk $\mathbf{x}_t \in \mathbb{R}^d$. Pri danih parametrih modela $\mathbf{w}_t$, izračunamo napoved z enačbo logistične regresije:

$$p_t = \sigma(\langle \mathbf{w}_t, \mathbf{x}_t \rangle) \tag{2.1}$$

pri čemer $\langle \cdot, \cdot \rangle$ predstavlja skalarni produkt dveh vektorjev dolžine $k$:

$$\langle \mathbf{w}_t, \mathbf{x}_t \rangle := \sum_{f=1}^{k} w_f \cdot x_f \tag{2.2}$$

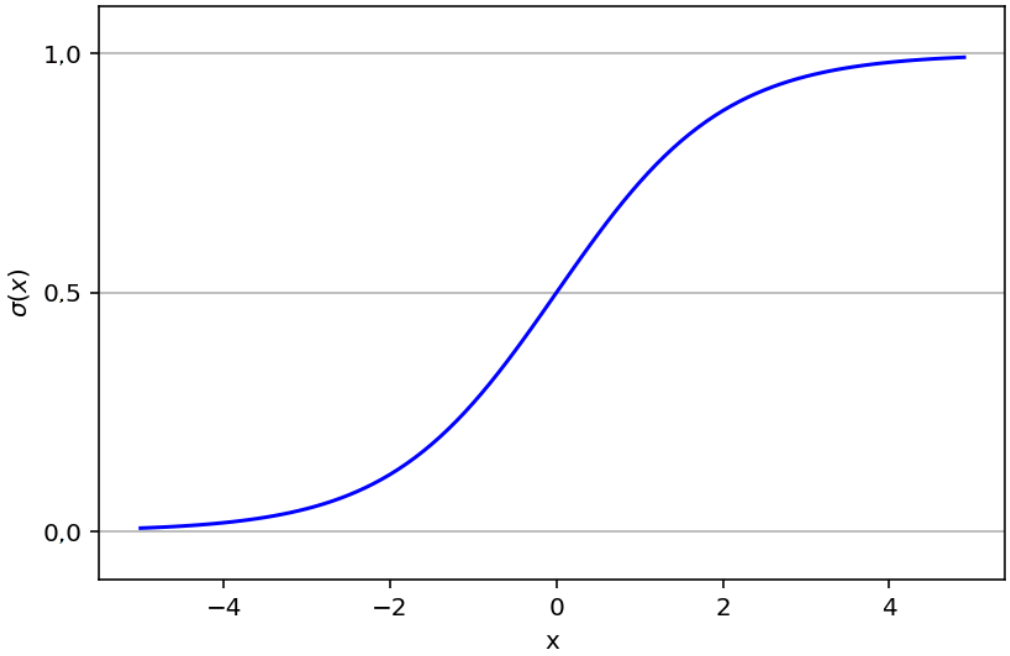
in kjer $\sigma(x) = \frac{1}{1+e^{-x}}$ predstavlja sigmoidno funkcijo. Sigmoidna funkcija je značilne oblike črke S, odvedljiva, monotona funkcija omejena med 0 in 1, kot na sliki 2.1. Po tem opazujemo oznako $y_t \in \{0, 1\}$, iz katere izračunamo logistično izgubo, ki nam predstavlja cenilno funkcijo:

$$\ell_t(\mathbf{w}_t) = -y_t \log p_t - (1 - y_t) \log(1 - p_t). \tag{2.3}$$

Iz tega sledi, da je:

$$\nabla \ell_t(\mathbf{w}) = (\sigma(\mathbf{w} \cdot \mathbf{x}_t) - y_t)\mathbf{x}_t = (p_t - y_t)\mathbf{x}_t \tag{2.4}$$

in ta gradient je vse kar potrebujemo za proces optimizacije s pomočjo gradientnega spusta. Gradientni spust je dokazano učinkovit pri reševanju problemov take vrste, saj izvaja natančne napovedi z minimalnimi računskimi viri. Vendar, v praksi obstaja stvar na katero moramo biti pozorni t.j. velikost končnega modela. Ker je model shranjen v redki obliki, je število ne ničelnih koeficientov v $\mathbf{w}$ odločilni faktor pri obremenitvi spomina.




Slika 2.1: Sigmoidna funkcija $\sigma(x)$.

### 2.1.2 Faktorizacijske metode

Faktorizacijska metoda je bila prvič predstavljena v znanstvenem članku leta 2010 in kmalu za tem uporabljena na največjih tekmovanjih iz odkrivanja znanj iz podatkov, ki so jih organizirala podjetja kot so Outbrain, Netflix in Criteo [24]. Faktorizacijske metode izkoriščajo prednosti metode podpornih vektorjev (angl. *Support Vector Machines*) ter matrične faktorizacije. Uporabljamo jih lahko tako za reševanje regresijskih kot klasifikacijskih problemov. Za razliko od metod podpornih vektorjev so faktorizacijske metode bolj učinkovite takrat, ko so naši podatki redki. Redkost podatkov je zelo značilna ravno za naš problem.

Redkost podatkov je najlažje razložiti, če podatke predstavimo z matriko. Vsaka vrstica matrike predstavlja učni primer (pri našem problemu prikaz nekega oglasa), vsak stolpec pa lastnost, ki jo določen primer ima oziroma nima (1 ali 0). Ker je večina lastnosti klasifikacijskih, nabor vrednosti za določeno značilnost pa ogromen, je končna matrika visokih dimenzij in vsebuje večino

ničel. Takšni problemi so za določene modele strojnega učenja prezahtevni (tako časovno, kot prostorsko) in zato na njih ne dosežejo dobrih rezultatov. Faktorizacijske metode se z redkostjo podatkov spopadejo tako, da hranijo interakcije med značilkami z uporabo faktorizacijskih parametrov. Da bi si lažje predstavljali, kaj so interakcije med značilkami, si poglejmo konkreten primer.

| **ZALOŽNIK** | **OGLAŠEVALEC** | **KLIK** | **NE KLIK** |
|---|---|---|---|
| ESPN | Nike | 80 | 20 |
| ESPN | Gucci | 10 | 90 |
| ESPN | Adidas | 0 | 1 |
| Vogue | Nike | 15 | 85 |
| Vogue | Gucci | 90 | 10 |
| Vogue | Adidas | 10 | 90 |
| NBC | Nike | 85 | 15 |
| NBC | Gucci | 0 | 0 |
| NBC | Adidas | 90 | 10 |

Tabela 2.1: Izmišljen nabor podatkov, kjer vsaka vrstica opisuje primer oglaševalca pri določenem založniku s številom kliknjenih in ne kliknjenih prikazov [17].

Tabela 2.1 prikazuje, kako dobro se oglaševalci izkažejo pri določenih založnikih. Oglas podjetja Gucci ima zelo visok CTR pri založniku Vogue. To informacijo je z linearnimi modeli težko zaznati, saj hranijo ločene uteži za Gucci in Vogue. Zaradi zmožnosti hranjenja interakcij med značilkami so faktorizacijske metode sposobne zaznati takšno informacijo. Takšna sposobnost napovednega modela je ključna pri dobrem napovedovanju CTR.

Enačba modela faktorizacijskih metod stopnje $d = 2$ je definirana kot:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j, \tag{2.5}$$

kjer morajo biti parametri modela ocenjeni v območjih:

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^n, \quad \mathbf{V} \in \mathbb{R}^{n \times k},$$

pri čemer $\langle \cdot, \cdot \rangle$ predstavlja skalarni produkt dveh vektorjev dolžine $k$:

$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle := \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f}. \tag{2.6}$$

Vrstica $\mathbf{v}_i \in \mathbf{V}$ opisuje $i$-to značilko s $k$ faktorji. Hiperparameter $k \in \mathbb{N}$ definira dimenzionalnost faktorizacije. Dvodimenzionalne faktorizacijske metode (stopnje $d = 2$) zajamejo vse posamezne in parne interakcije med značilkami:

- $w_0$ - člen pristranskosti (angl. *bias term*) nam pomaga pri hitrejši konvergenci modela k končni rešitvi.
- $w_i$ - utež $i$-te značilke modela.
- $\hat{w}_{i,j} := \langle \mathbf{v}_i, \mathbf{v}_j \rangle$ - modelira interakcijo med $i$-to in $j$-to značilko.

Splošno znano je, da za vsako pozitivno definitno matriko $\mathbf{W}$ obstaja matrika $\mathbf{V}$ za katero velja $\mathbf{W} = \mathbf{V} \cdot \mathbf{V}^t$. Iz tega sledi, da faktorizacijske metode lahko izrazijo poljubno interakcijsko matriko $\mathbf{W}$, če je izbran $k$ dovolj velik. Kljub temu pri redkih podatkih tipično izberemo manjši $k$, ker zaradi pomanjkanja podatkov stežka ocenimo kompleksne interakcije $\mathbf{W}$. Omejevanje $k$-ja in s tem tudi izraznosti vodi k večji splošnosti modela.

Rendle v svojem članku obrazloži, zakaj se faktorizacijske metode na redkih podatkih obnesejo bolje od polinomske regresije druge stopnje [17]. Za bolj nazorno predstavo se spomnimo tabele 2.1 s prejšnje strani. Na primer, vse kar vemo o paru (ESPN, Adidas) je zgolj en negativen učni primer. Utež $w_{ESPN \cdot Adidas}$ bo posledično pri polinomski regresiji nizka. Ker je napoved za (ESPN, Adidas) pri faktorizacijskih metodah določena na podlagi

$w_{ESPN} \cdot w_{Adidas}$ in ker sta $w_{Adidas}$ in $w_{ESPN}$ naučeni že iz preteklih parov na primer ((ESPN, Nike), (NBC, Adidas)) bo lahko napoved bolj natančna. Pri polinomski regresiji prav tako nastane problem, ko v testni množici naletimo na primer, ki ga v učni množici ni bilo na primer (NBC, Gucci). Faktorizacijske metode lahko zaradi $w_{NBC}$ in $w_{Gucci}$, ki sta bili naučeni iz preteklih primerov, vrnejo smiselno predikcijo.

Faktorizacijske metode so uporabne predvsem zaradi ugodne časovne zahtevnosti. Zaradi poenostavitve enačbe modela (zadnje dvojne vsote) je lahko enačba (2.5) izračunana v linearnem času in sicer $O(kn)$. Celotna izpeljava in dokaz sta na voljo v objavljenem članku. Če uporabimo omenjeno poenostavitev, dobimo enačbo našega modela:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i x_i + \frac{1}{2} \sum_{f=1}^{k} \left( \left( \sum_{i=1}^{n} v_{i,f} x_i \right)^2 - \sum_{i=1}^{n} v_{i,f}^2 x_i^2 \right). \tag{2.7}$$

Omenjena lastnost je razlog, da je ta metoda vse bolj priljubljena med raziskovalci. Tudi za nas je časovna zahtevnost algoritma pomembna, saj je potrebno v 100 milisekundah odgovoriti na več 100.000 zahtev.

Če omenimo še proces učenja faktorizacijskih metod, lahko za to nalogo izbiramo med več različnimi optimizacijskimi metodami. Za nas bo najbolj zanimiv stohastični gradientni spust (angl. *stochastic gradient descent*). Je ena izmed bolj uporabljenih metod optimizacije v strojnem učenju. Pri stohastičnem gradientnem spustu izvajamo posodobitve uteži po vsakem učnem primeru. Beseda 'stohastični' pomeni, da je gradient, ki je izračunan na podlagi enega primera, 'stohastični' približek resničnega. Vsaka posodobitev uteži se izračuna tako, da se za določen korak pomaknemo v negativni smeri odvoda cenilne funkcije (2.8) in se s tem korak za korakom bližamo lokalnemu minimumu cenilne funkcije.

$$\frac{\partial}{\partial \theta} \hat{y}(\mathbf{x}) = \begin{cases} 1, & \text{če je } \theta \text{ enak } w_0. \\ x_i, & \text{če je } \theta \text{ enak } w_i. \\ x_i \sum_{j=1}^{n} v_{j,f} x_j - v_{i,f} x_i^2 & \text{če je } \theta \text{ enak } v_{i,f}. \end{cases} \tag{2.8}$$

Velikost posodobitve uteži je določena s parametrom $\alpha$ katerega imenujemo stopnja učenja in je eden izmed parametrov našega modela, ki ima velik vpliv na kvaliteto napovedi. Tudi stopnja regularizacije je eden izmed parametrov modela, ki ga označimo z $\lambda$. Namenjen je 'glajenju' modela, pravilna uporaba pa omogoča povečati splošnost in s tem večjo natančnost na novih primerih.

## 2.2 Obdelava podatkov

### 2.2.1 Predpriprava

V strojnem učenju poznamo veliko faktorjev, ki vplivajo na kvaliteto izvedbe [19]. Dva izmed večjih sta predstavitev in kvaliteta podatkov. Če v podatkih obstaja veliko nepomembnih oziroma nepotrebnih informacij ali šumnih in nezanesljivih podatkov, potem bo odkrivanje znanja v času učenja težje. Splošno znano je, da predpriprava in filtriranje vzameta veliko časa pri reševanju problemov strojnega učenja. Predpriprava podatkov običajno vključuje čiščenje podatkov, normalizacijo, transformacije in izbiro značilk. Za nas bodo najbolj zanimivi čiščenje, vzorčenje in izbira atributov. Produkt predpriprave podatkov je končna učna množica.

Izključitev šumnih podatkov je eden izmed težjih problemov induktivnega strojnega učenja. Eden izmed problemov s katerimi se soočamo v domeni spletnega oglaševanja je nečloveški promet. Učenje na primerih in kupovanje vtisov za katerimi se ne skriva človeški um predstavlja lažno informacijo za nas, ki te podatke proučujemo, naše stranke in podjetje. Problem nečloveškega prometa je v spletnem oglaševanju velik, zato za njegovo reševanje obstajajo najrazličnejše metode, predvsem v smislu razpoznavanja vzorcev v prometu. Primere za katere domnevamo, da niso plod človekovega ravnanja, je smiselno odstraniti iz učne množice naših podatkov in se s tem delno izogniti nadaljnjemu kupovanju takšnih vtisov.

Podatki, ki bodo podani algoritmu strojnega učenja so lahko nepopolni (manjkajoče vrednosti atributov), šumni ali neskladni. Razlogi za nastanek

so lahko različni na primer atribut, ki smo ga zahtevali ni bil na voljo (na primer piškotki v določenem brskalniku), napaka v sistemu ali napaka med prenosom. Proces, ki takšne anomalije odpravlja imenujemo čiščenje podatkov. Ta poteka tako, da manjkajoče vrednosti primerno obravnavamo, na primer:

- **Učni primer ignoriramo** – Velikokrat se poslužujemo te obravnave še posebej, če je atribut za nas zelo pomemben (na primer na kateri spletni strani se je zgodil prikaz oglasa).
- **Manjkajoče vrednosti vnesemo ročno** – V splošnem zelo neučinkovit in nepriporočljiv način obravnave.
- **Uporabimo globalno konstanto s katero nadomestimo manjkajočo vrednost atributa** – Vrednost lahko na primer nastavimo na 'Neznano'. Ta metoda ni priporočljiva, saj lahko algoritem strojnega učenja to vrednost interpretira kot skupno značilnost vseh primerov, ki jo vsebujejo.
- **Uporabimo povprečno vrednost atributa** – Uporabno pri zveznih atributih vendar običajno nepriporočljiv način, sploh kadar je manjkajočih vrednosti veliko.
- **Uporabimo povprečno vrednost atributa primerov, ki pripadajo istemu razredu** – Uporabno pri zveznih atributih vendar običajno nepriporočljiv način, sploh kadar je manjkajočih vrednosti veliko.
- **Uporabimo največkrat zastopano vrednost atributa** – Uporabno pri kategoričnih atributih.

Zadnje štiri metode podatkom povečajo pristranskost. Vnešene vrednosti najverjetneje niso pravilne, kar lahko negativno vpliva na izvedbo.

Za celoten proces priprave podatkov namenjenih eksperimentom pripravimo program, ki iz standardnega vhoda bere (nečiste) podatke vrstico za vrstico, jih preveri ter nato pretvori v enega izmed izbranih formatov, na

koncu pa izpiše statistiko. Preden program zaženemo si v ustrezni datoteki nastavimo vse potrebne spremenljivke:

- **Pot do ciljnega direktorija**
- **Format** – Zemanta, libFM, Vowpal Wabbit, CSV.
- **Pot do datoteke, ki vsebuje seznam značilk, ki jih želimo vključiti v učno množico**
- **Stopnja negativnega vzorčenja** – Privzeta vrednost je enaka 1, kar pomeni da vzamemo vsak negativen primer.
- **Delež podatkov namenjen učni množici** – Na podlagi željene razdelitve se v statistiki izračuna vrednost, ki predstavlja število vrstic, ki spadajo v učno množico.

Preverjanje vrstic poteka tako, da najprej ugotovimo, če je vhoden format podatkov ustrezen, nato če lahko vrstico razdelimo po parih (atribut, vrednost) in če vsebuje vse potrebne vrednosti atributov, na primer vrednost ciljne spremenljivke. Ločimo tri vrste atributov: *osnovne* (vrednosti podane že v grobih podatkih), *generirane* (njihova vrednost je določena sproti, na podlagi osnovnih, na primer končnica pri naslovu spletne strani) in *eksplodirane* (množica vrednosti je določena sproti, na podlagi osnovnih, na primer naslov spletne strani ločen po poševnicah). Za tem sledi pretvorba v ustrezen format, ki ga zna na vhodu sprejeti naš model strojnega učenja. Ker v naših ekperimentih ne uporabljamo zgolj naše implementacije algoritma temveč tudi uradno implementacijo faktorizacijskih metod *libFM* [4], tudi zanjo podpremo način pretvorbe. Ob koncu izvajanja program poleg prevedenih podatkov vrne statistiko, ki vsebuje: število vseh primerov na vhodu/izhodu, število negativnih/pozitivnih primerov ter število vrstic, ki sodijo v učno množico.

### 2.2.2 Izbor značilk

V statistiki in strojnem učenju je izbor značilk proces iskanja podmnožice relevantnih značilk, ki jih kasneje uporabimo za gradnjo našega napovednega modela [16]. Poleg čiščenja podatkov je to eden izmed pomembnejših korakov v izvedbi našega ekperimenta in ima velik vpliv na kvaliteto napovedi. Celoten proces je običajno avtomatiziran s pomočjo preiskovalnega algoritma, a je človeški faktor še vedno ključen, predvsem pri končni interpretaciji. Razlogi za uporabo metod za izbiro značilk so: poenostavitev modela (lažja interpretacija), krajši čas učenja (z dodajanjem značilk se čas izvedbe algoritma povečuje), obvarujemo se pred prekletstvom dimenzionalnosti, naš model naredimo bolj splošen in s tem zmanjšamo verjetnost prevelikega prileganja učnim podatkom.

Metode za izbor značilk lahko grobo razdelimo v tri kategorije glede na to kdaj in kako se izračuna koristnost značilk [20]:

- **Filtri** – Slonijo na splošnih karakteristikah podatkov, značilke so izbrane v odsotnosti učnega algoritma.
- **Ovojnice** – Zahtevajo vnaprej določen učni algoritem, opazujejo rezultate pri različnih podanih značilkah.
- **Vgrajene metode** – Vključijo izbiro značilk kot del procesa prileganja oziroma učenja, uporabnost značilk je določena na podlagi optimizacije cenilne funkcije.

Prednost postopkov, ki jih uvrščamo med *filtre* je, da so neodvisni od učnega algoritma. Struktura metod je enostavna in izvedba običajno zelo hitra. Raziskovalci so ugotovili, da metode ki jih uvrščamo med *ovojnice* in *vgrajene metode* običajno izberejo značilke, ki producirajo boljše rezultate na specifičnem učnem algoritmu, ki je bil uporabljen med procesom izbora značilk.

Naš pristop k reševanju tega problema vključuje rešitev tipa *ovojnice*. Za namen odkritja najboljšega seznama značilnosti za naš model prilagodimo program, ki po principu požrešnega iskanja z različnimi strategijami preiskuje prostor možnih seznamov značilk in se pomika v smeri največje izboljšave.

## 2.3 Implementacija faktorizacijskih metod

*Go* je programski jezik v katerem implementiramo naš napovedni model. Izdalo ga je podjetje Google leta 2003 in je tipiziran, preveden programski jezik (zanj ne potrebujemo interpreterja) [3]. Je izrazen, jedrnat, čist in učinkovit. Njegov mehanizem vzporednosti omogoča enostaven razvoj programov, ki dobro izkoristijo večjedrne in omrežne naprave, poleg tega pa omogoča fleksibilno in modularno izdelavo programov. *Go* ima vgrajeno funkcionalnost in podporo za pisanje vzporednih programov. Vzporednost ne samo v smislu paralelnega procesiranja, temveč tudi asihronosti, t.j. omogoča počasnejšim operacijam (kot na primer branje iz podatkovne baze), da se izvajajo, medtem ko program počne preostale stvari. Po nekaj lastnostih spominja na programski jezik C (tipiziran in učinkovit), obenem pa vsebuje več varnostnih mehanizmov in s tem omogoča enostavnejši razvoj. Zaradi svoje hitrosti in enostavnosti se nam zdi primeren programski jezik za razvoj algoritmov strojnega učenja.

Logika napovednega modela predstavlja del programske opreme imenovane ponudnik (angl. *bidder*), ki je glavni sestavni del DSP okolja. Napovedovanje CTR je eden izmed korakov, ko se odločamo, katerega izmed oglasov naših strank bomo prikazali uporabniku, ter kakšno ceno smo za ta prikaz pripravljeni plačati. Ob tej priložnosti kličemo funkcijo *predict*, ki nam vrne napoved za trenutni primer. Ko za našo ponudbo prejmemo odgovor, lahko kličemo funkcijo *AddExample*, ki glede na napoved in končnen izid prilagodi uteži. Poleg implementacije algoritma v produkcijski sistem pa želimo v lokalnem okolju z modelom eksperimentirati na preteklih podatkih in se s tem prepričati, če je model sploh lahko konkurenčen obstoječemu. Zato si pripravimo ločen kos programske kode, ki nam omogoča izvajanje učenja in napovedovanja na podatkih iz preteklih dni.

Model faktorizacijskih metod je v programski kodi predstavljen kot struktura z naslednjimi sedmimi atributi:

```
type FactorizationMachine struct {
    w                   Weights
    v                   []Weights
    learningRate        float32
    regularization      float32
    factorNumber        int
    initWeight          float32
    negSubSample        int
}
```

pri tem je *Weights* struktura, s katero predstavimo pare (značilka, utež) in je definirana kot:

```
type Weights struct {
    map[string]float32
}
```

To pomeni, da ima vsaka instanca našega algoritma naslednje lastnosti:

- **w** – Predstavlja seznam parov (značilka, utež).
- **v** – Predstavlja seznam seznamov parov (značilka, utež).
- **learningRate** – Parameter modela, ki predstavlja stopnjo učenja.
- **regularization** – Parameter modela, ki predstavlja stopnjo glajenja.
- **factorNumber** – Parameter modela, ki predstavlja število dimenzij in določa velikost $v$-ja.
- **initWeight** – Parameter modela, ki določa vrednosti uteži za atribute, ki jih vidimo prvič.
- **negSubSample** – Parameter modela, ki znižuje napovedi v primeru, ko je naš nabor učnih podatkov negativno vzorčen.

### 2.3.1 Funkcija za napovedovanje

Funkcija *predict* je ena izmed dveh glavnih funkcij modela faktorizacijskih metod. Uporablja se pri izbiri oglasov ter pri oceni, kolikšno ceno smo pripravljeni plačati za njegov prikaz. Kot parameter sprejme trenuten primer t.j. vektor značilk (angl. *feature vector*), ki vključuje vse navedene značilke in njihove vrednosti. Na podlagi trenutnega stanja uteži funkcija vrne napoved kot tip *float64*.

Naša funkcija za napovedovanje uporablja poenostavljeno različico prvotne formule (2.7). Za njen izračun potrebujemo naslednje sestavne dele:

- $w_0$ – Člen pristranskosti.
- $\sum_{i=1}^{n} w_i x_i$ - Skalarni produkt parov (utež atributa, vrednost atributa).
- $\left( \sum_{i=1}^{n} v_{i,f} x_i \right)^2$ – Kvadriran skalarni produkt parov faktorjev, ki opisujejo interakcije in vrednosti atributov.
- $\sum_{i=1}^{n} v_{i,f}^2 x_i^2$ – Skalarni produkt kvadriranih faktorjev interakcij in kvadriranih vrednosti atributov.

Zadnja dva člena seštejemo po vseh dimenzijah, dobljeno vrednost pomnožimo z $\frac{1}{2}$ in dobimo končno vrednost. Ker napovedujemo verjetnost za klik, želimo, da so naše napovedi omejene med 0 in 1. To dosežemo tako, da uporabimo funkcijo, ki smo jo spoznali v 2. poglavju . Zadnji korak je, da preverimo če je parameter *negativnega vzorčenja* enak 1, sicer popravimo napoved s formulo $q = \frac{p}{p+(1-p)/w}$ [15], pri čimer $q$ predstavlja popravljeno napoved, $p$ napoved, ki jo je vrnila logistična funkcija, in $w$ stopnjo negativnega vzorčenja. Pozorni moramo biti pri tem, da popravljeno napoved uporabljamo le za izpis, ne pa tudi za popravljanje uteži.

### 2.3.2 Funkcija za učenje

Druga ključna funkcija faktorizacijskih metod je funkcija za učenje *AddExample*. Tu imamo na voljo kanček več svobode kot pri funkciji za napovedovanje, saj lahko izbiramo med različnimi optimizacijskimi metodami. Mi se osredotočimo na implementacijo metode stohastičnega gradientnega spusta. Za razliko od funkcije za napovedovanje ta metoda ne vrača vrednosti, torej je tipa *void*. Kot parametera sprejme oznako primera in vektor značilk. Naloga funkcije je na podlagi njenih parametrov popraviti trenutne uteži v smeri zmanjšane napake. Za izračun gradienta potrebujemo enačbo (2.8).

```
func AddExample(example FeatureVector, success bool) {
    y := 0.0
    if success {
        y := 1.0
    }
    pred := predict(example)

    // izračun napake
    err := y - pred

    for i := range example {
        // izračun gradienta in posodobitev uteži za w_i

        for f := range v {
            // izračun gradienta in posodobitev uteži za v_{i,f}
        }
    }
}
```

# Poglavje 3

# Testiranje

V tem poglavju opišemo postopke testiranja implementiranega sistema v različnih okoljih. Pri tem si pomagamo s testiranjem posameznih enot znotraj programskega jezika *Go*, z uradno implementacijo faktorizacijskih metod *libFM*, okoljem za postopno izvajanje programske kode *Jupyter Notebook* in okoljem za analizo in nadzor *Grafana*.

Testiranje programske opreme je proces preučevanja obnašanja sistema z namenom, da odkrijemo morebitne napake. [9]. Preizkušanje programske opreme lahko zagotovi objektiven, neodvisen pogled, ki podjetjem omogoča, da so vešči in razumejo tveganja pri izvajanju programske opreme. Različne tehnike vsebujejo postopke izvajanja programov ali aplikacij z namenom iskanja napak v programski kodi in preverjanja, ali je program primeren za uporabo. S testiranjem želimo programu dodati vrednost [23]. Dodati vrednost v takšnem smislu, kot je povečanje kvalitete ali izboljšanje berljivosti. Boljša berljivost pomaga pri odkrivanju in odstranjevanju napak. Zaradi tega, ne testiramo svojih programov tako, da pokažemo da delujejo temveč začnemo s predpostavko, da program vsebuje napake in nato testiramo program tako, da jih odkrijemo čim več.

"*Testiranje je proces izvajanja programa z namenom, da odkrije napake.*"

## 3.1 Testiranje posameznih enot

Testiranje enot (angl. *unit testing*) je proces testiranja individualnih podprogramov, funkcij in procedur znotraj programa [23]. Namesto prvotnega testiranja programa kot celote, je testiranje enot osredotočeno na manjše gradnike programa. Razlogov za izvajanje takšnega testiranja je več. Prvič, testiranje modulov je način upravljanja kombiniranih elementov testiranja, saj je na začetku pozornost usmerjena na manjše enote programa. Drugič, testiranje modulov olajšuje odkrivanje napak, saj je, ko se napaka zgodi znano, v katerem modulu se nahaja. Nazadnje, testiranje modulov uvaja paralelizem v proces testiranja programov.

V našem primeru modul vsebuje implementacijo faktorizacijskih metod. Med procesom razvoja algoritma se želimo čim hitreje prepričati, ali naš program deluje v pravi smeri oziroma ali je uspešen pri klasifikaciji primerov, ki so na prvi pogled intuitivni. Prav tako, pa si želimo zaznati slabo učinkovitost algoritma v primeru, da pride do sprememb v implementaciji, ki bi slabo vplivale na izvedbo. Zato pripravimo uvodni test in si pri tem pomagamo z izmišljenim naborom podatkov v tabeli 3.1. Na njem izvedemo večkratno testiranje s pomočjo metode prečne validacije. Prečna validacija je množica metod, s katerimi lahko preverimo kako dober je model na še ne videnih podatkih [27]. Za naš primer uporabimo različico prečne validacije (angl. *leave-one-out cross-validation*), ki ob vsakem izmed $n$ obhodov izbere enega izmed primerov, ki ga ne prikaže v procesu učenja. To je edini izmed primerov na katerem izvedemo napoved. Preostalih $n-1$ primerov tvori učno množico. Postopek ponovimo $n$-krat in na koncu rezultat ovrednotimo.

| COLOR | SIZE | ACT | AGE | INFLATED |
|---|---|---|---|---|
| Yellow | Small | Stretch | Adult | **T** |
| Yellow | Small | Stretch | Child | **T** |
| Yellow | Small | Dip | Adult | **T** |
| Yellow | Small | Dip | Child | **F** |
| Yellow | Large | Stretch | Adult | **T** |
| Yellow | Large | Stretch | Child | **T** |
| Yellow | Large | Dip | Adult | **T** |
| Yellow | Large | Dip | Child | **F** |
| Purple | Small | Stretch | Adult | **T** |
| Purple | Small | Stretch | Child | **T** |
| Purple | Small | Dip | Adult | **T** |
| Purple | Small | Dip | Child | **F** |
| Purple | Large | Stretch | Adult | **T** |
| Purple | Large | Stretch | Child | **T** |
| Purple | Large | Dip | Adult | **T** |
| Purple | Large | Dip | Child | **F** |

Tabela 3.1: Nabor podatkov za postopek prečne validacije [6].

Iz danih primerov lahko razberemo, da atributa *COLOR* in *SIZE* slabo oz. ne določata ciljne spremenljivke *INFLATED*, medtem ko atributa *ACT* in *AGE* imata vlogo pri tem, kakšna je vrednost ciljne spremenljivke. Če vrstico v tabeli označimo z $x$ potem velja, ko je $x_{ACT=Stretch}$ je ciljna spremenljivka vedno enaka $T$, v nasprotnem primeru pa preverimo še vrednost spremenljivke *AGE*, in če je vrednost te enaka $x_{AGE=Adult}$, je ciljna spremenljivka enaka $T$. V vseh ostalih primerih je vrednost ciljne spremenljivke enaka $F$. Ker je nabor podatkov izmišljen in za klasifikacijo precej intuitiven bo napovedni model v primeru pravilnega delovanja dosegel vsaj 90% natančnost.

## 3.2 Testiranje v lokalnem okolju

Po uspešnem testiranju posameznih enot se želimo prepričati o pravilni implementaciji matematično zahtevnejšega algoritma. Za ta namen se odločimo pripraviti manjši nabor podatkov ter na njem preizkusiti kako se naša implementacija obnese v primerjavi z uradno. Program, ki vključuje uradno implementacijo, lahko pridobimo s spletne strani, ki je navedena v avtorjevem uradnem *GitHub* repozitoriju [4]. Pri našem testiranju uporabimo trenutno najnovejšo verzijo programa (1.4.2). Namestitev je enostavna, saj si je program potrebno le prenesti, ga razpakirati ter prevesti z ukazom `make all`.

Po uspešni namestitvi se lotimo priprave vhodnih podatkov. *LibFM* lahko na vhodu sprejme dve različni obliki datotek v katerih se nahajajo vhodni podatki [25]. Ker knjižnjico uporabljamo prvič, se odločimo za tekstovni format. Vsaka vrstica vhodnih podatkov vsebuje učni primer $(x, y)$, kjer $x$ predstavlja vektor značilk, $y$ pa ciljno spremenljivko. Vrstica se prične z vrednostjo $y$, sledijo ji pa od nič različne vrednosti za $x$. Pri binarni klasifikaciji so $y > 0$ obravnavani kot pozitivni primeri, tisti z $y \leq 0$ pa kot negativni.

**Primer podatkov v libFM formatu:**

```
4    0:1.5    3:-7.9
2    1:1e-5   3:2
-1   6:1
```

Zgoraj so opisani trije učni primeri. Prvi stolpec vsebuje vrednosti ciljne spremenljivke. Za tem sledijo od nič različne vrednosti $x$-a, kjer se vnos $0 : 1.5$ bere kot $x_0 = 1.5$ in $3 : -7.9$ pomeni $x_3 = -7.9$ itd. To pomeni, da leva stran izraza *INDEKS:VREDNOST* pomeni indeks znoraj $x$-a, medtem ko desna stran pomeni vrednost $x$-a pri tem indeksu, torej: $x_{INDEKS} = VREDNOST$. Celoten zgornji nabor lahko predstavimo tudi z matriko:

$$X = \begin{bmatrix} 1.5 & 0.0 & 0.0 & -7.9 & 0.0 & 0.0 & 0.0 \\ 0.0 & 10^{-5} & 0.0 & 2.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}, y = \begin{bmatrix} 4 \\ 2 \\ -1 \end{bmatrix}.$$

Sedaj, ko je program pravilno nameščen ter vhodni podatki pripravljeni, ga lahko z enim samim ukazom zaženemo. Za uspešen zagon je potrebno določiti vsaj tri vhodne parametre. Prvi izmed njih je `-task` s katerim povemo katere vrste problem rešujemo klasifikacijski `-task c` oz. regresijski `-task r`. Drugi obvezni parameter je pot do datotek z učnimi podatki `-train` in testnimi podatki `-test`. Kot že rečeno so podatki lahko v tekstovni oziroma binarni obliki. Zadnji je dimenzionalnost faktorizacijskih metod, ki ga določimo s parametrom `-dim`. Slednji je sestavljen iz treh delov.

- $k_0 \in 0, 1$ – Določa, če želimo uporabiti člen pristranskosti – $w_0$.
- $k_1 \in 0, 1$ – Določa, če želimo uporabiti člene z enosmernimi interakcijami – $w_i$.
- $k_2 \in \mathbb{N}_0$ – Določa število faktorjev, ki so uporabljeni za beleženje parnih interakcij, tj. $k$ pri $V \in \mathbb{R}^{p \times k}$.

**Primer ukaza za zagon libFM:**

```
./libFM -task r -train ml1m-train.libfm
        -test ml1m-test.libfm -dim '1,1,8'
```

Poleg obveznih parametrov nam je na voljo še množica neobveznih. Eden izmed uporabnejših je `-out` za katerim navedemo pot do datoteke v katero želimo zapisati napovedi na množici testnih podatkov. Tako je ustvarjena tekstovna datoteka, ki vsebuje natanko toliko vrstic kot testni podatki, v $i$-ti vrstici pa najdemo napoved za $i$-ti primer. Pri klasifikaciji so napovedi enake verjetnosti da testni primer pripada pozitivnemu razredu.

Sedaj, ko sta pripravljeni tako naša implementacija algoritma kot tudi uradna, lahko na manjšem naboru podatkov opazujemo kako se pri različnih vrednostih parametrov (stopnje učenja, regularizacije, število faktorjev) spreminja natančnost napovedi. Pričakujemo podobno obnašanje obeh različic. To bi občutno povečalo verjetnost, da je naša implementacija algoritma pravilna. Rezultate primerjanj bolj podrobno opišemo v poglavju 4.2.

Po potrditvi, da je naša implementacija algoritma pravilna se lotimo primerjanja z logistično regresijo. Pred tem se spomnimo še dveh stvari. Trenutno v produkcijskem okolju obratuje model logistične regresije in lokalno testiranje poteka na podatkih iz preteklih zmaganih dražb. Pri RTB dražbah se znesek ponudi zgolj enkrat, pri čemer obvestilo o zmagi prejme le tisti udeleženec, ki plača največ. Posledično je zmagovalec dražbe edini, ki izve izid prikaza. Kaj to pomeni? Ker je cena, ki jo je naš udeleženec dražbe pripravljen plačati, določena na podlagi algoritma, ki takrat obratuje v produkcijskem okolju, so s tem kupljeni zgolj tisti prikazi, ki jih ta algoritem obravnava kot perspektivne. To pomeni, da so podatki ki jih uporabljamo za testiranje v lokalnem okolju pristranski. Kljub temu pa nam lokalno testiranje razmeroma dobro pokaže ali je nov model vsaj konkurečen obstoječemu.

Velik vpliv na kvaliteto končne izvedbe imajo vhodni parametri faktorizacijskih metod. Za to testiranje si pripravimo večji nabor podatkov (preko 10

milijonov primerov) pri čimer 70% predstavlja učne podatke, 30% pa testne. Za začetek parametre nastavimo na privzete vrednosti (enake kot pri *libFM*). Nato z mrežnim preiskovanjem iščemo najbolj optimalne vrednosti na primer vrednosti regularizacije preiščemo med $10^{-4}$ in $10^{-8}$ z razmikom za faktor 10. Ko ocenjujemo najboljšo vrednost za enega izmed parametrov na primer stopnjo učenja, preostale parametre fiksiramo, stopnjo učenja pa spreminjamo ter na koncu izberemo tisto z najboljšim rezultatom. Pri tem se je potrebno zavedati, da vrstni red preiskovanja parametrov vpliva na končen rezultat in da ta rezultat ni optimalen, zadošča pa za našo začetno primerjavo.

Pri prejšnjem testiranju je bilo očitno uporabiti enak seznam značilk tako pri naši kot tudi pri uradni implementaciji. Tokrat temu ni več tako, saj vemo, da enaki seznami značilk različno dobro delujejo na različnih algoritmih strojnega učenja. Seznam značilk logistične regresije, ki trenutno deluje v obstoječem sistemu, je bil zgrajen s preiskovalnim algoritmom o katerem smo pisali v poglavju 2. Vsebuje preko deset kombinacij, ki štejejo tudi do štiri značilke v eni sami kombinaciji (na primer kako se določen oglas obnese na določenem mestu, na določeni spletni strani, na določeni napravi, v določenem brskalniku). Podpore za preiskovanje značilk s faktorizacijskimi metodami v fazi testiranja v lokalnem okolju še ni. Za potrebe našega testiranja izberemo dva različna seznama značilk:

- **Seznam značilk iz obstoječega sistema** – kompleksen, kombinacije do 4 značilk, prilagojen glede na delovanje logistične regresije.

- **Seznam značilk iz obstoječega sistema brez kombinacij** – enostavnejši, samo unikatne samostojne značilke, bolj splošen.

Takšno nastavitev uporabimo v prvem delu primerjav med faktorizacijskimi metodami in logistično regresijo. Rezultati so na voljo v poglavju 4.

## 3.3 Testiranje v produkcijskem okolju

Ko uporabimo izraz produkcijsko okolje, si moramo znati predstavljati kaj to pomeni in kakšno vlogo ima naš sistem v njem. Kot že opisano v 1. poglavju, naš sistem igra pomembno vlogo pri tem, kolikšen znesek smo pripravljeni plačati za prikaz določenega oglasa. Slednje vpliva tudi na to, kateri oglas je izbran izmed vseh v podatkovni bazi našega sistema. Z drugimi besedami, ima velik vpliv na to, kako udeleženec dražb ravna z razpoložljivimi sredstvi, kakšno vsebino širi po spletu ter kako dobre rezultate dosega.

Če želimo dva različna modela med seboj primerjati, moramo pred tem opraviti še nekaj stvari. Najprej si pripravimo množico najnovejših učnih primerov, ki bo naš model pripravila na produkcijsko okolje. Poleg faktorizacijskih metod moramo na istih primerih naučiti tudi instanco trenutnega modela logistične regresije. Enako predznanje je pomemben faktor, ko želimo med seboj pravično primerjati dva različna algoritma. To pomeni, da bodo v času testiranja v produkcijskem okolju obratovali trije algoritmi in sicer: trenutni produkcijski model logistične regresije, ki bo še vedno obratoval na večini prometa, naš model faktorizacijskih metod in kontrolni model logistične regresije. Zadnja dva bosta delovala na manjšem in enakem deležu naključno razdeljenega prometa. S tem se v začetni fazi želimo izogniti večji nastali škodi v primeru nepredvidljivih napak ter slabše izvedbe (predvsem zaradi manjše količine predznanja). Test mora trajati dovolj dolgo, saj s tem zmanjšamo verjetnost, da eden izmed algoritmov prejme boljši del prometa kot drug.

Ker se stanje na trgih iz trenutka v trenutek spreminja, ni smiselno primerjati rezultatov iz različnih časovnih obdobij. Zato uporabimo statistično tehniko naključnega eksperimenta z dvema variantama t.j. A/B test. Enak princip se prav tako uporablja pri razvoju spletnih strani, ko želimo preizkusiti sveže funkcionalnosti na dejanskih uporabnikih. Ko zahteva za ponudbo pride v sistem, ta s pomočjo naključne vrednosti izbere model, kateremu preda odločitev.

Tudi če bi opravljali A/B test med dvema enakima modeloma (torej A/A

test), bi prišlo do rahlih odstopanj v rezultatih. Ta odstopanja bi se z neko verjetnostjo nahajala znotraj določenega intervala. Tu v poštev pride znanje statistike t.j. intervali zaupanja in testiranje hipotez. S pomočjo statističnih metod lahko opišemo, kako verjetno je, da je na podlagi končnih rezultatov eden izmed modelov zares boljši od drugega.

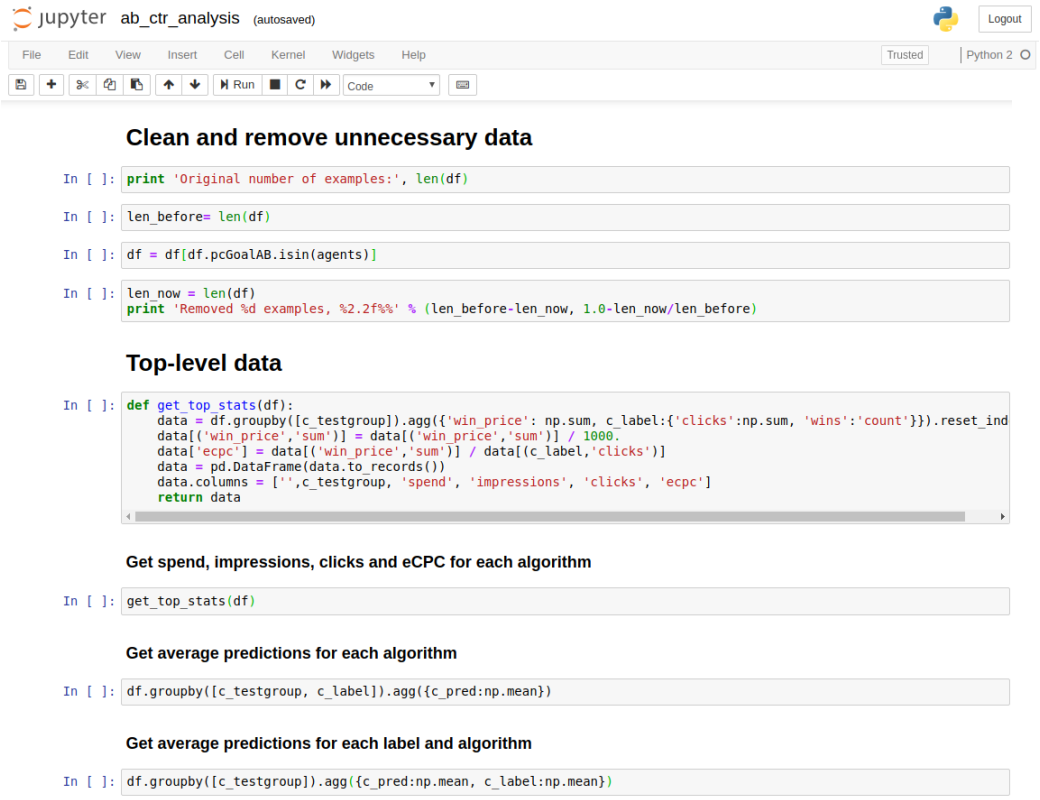
Pri kompleksnem produkcijskem okolju kot je naše, je koristno imeti sistem s katerim lahko v živo spremljamo dogajanje. Za to je zadolžena ena izmed podatkovnih baz, ki v realnem času hrani vse podatke v zvezi z akcijami, ki jih izvršuje naš sistem kot na primer:

- **Število zahtevkov za ponudbo, ki jih naš sistem prejme v določenem časovnem obdobju**
- **Število izjem ter napak, ki se zgodijo v določenem časovnem obdobju**
- **Povprečne napovedi naših algoritmov strojnega učenja**
- **Največje in najmanjše vrednosti uteži v napovednih modelih**
- **Čas iteracije generiranja učnih podatkov**

Podatki so na voljo preko preglednega uporabniškega vmesnika večinoma v obliki grafov. Namenjeni so predvsem inženirjem, da lahko opazujejo, kaj se s sistemom dogaja, ko nadgrajujejo sistem z novo funkcionalnostjo. Skoraj vsaka izmed količin se mora nahajati znotraj določenega intervala, v nasprotnem primeru so o tem obveščeni. Takšen sistem koristi, ko v produkcijsko okolje namestimo vse tri omenjene algoritme in pričnemo s produkcijskim tesiranjem.

Po nekaj dnevih delovanja so nam v enaki obliki kot učni podatki na voljo rezultati naših napovedi. Poleg ciljne spremenljivke ter vseh lastnosti o zahtevku za ponudbo, nam je na voljo tudi atribut, ki opisuje, kateri izmed modelov je zahtevek za ponudbo obravnaval. To nam pomaga pri ločevanju podatkov v dve ločeni množici. Na vsaki izmed njih izvedemo najrazličnejše

meritve. Pomagamo si s prej pripravljenimi kratkimi funkcijami, ki jih hranimo v *Jupyter Notebook*-u. To orodje nam omogoča na pregleden način izvesti delčke programske kode, jih jasno označiti ter nemudoma videti njihov učinek. Te kratke funkcije uporabljajo najbolj znane odprtokodne knjižnice za manipuliranje in analiziranje podatkov v programskem jeziku *Python*. To so na primer *numpy*, *pandas*, *matplotlib*, *scikit-learn* itd. S pomočjo teh orodij lahko temeljito in hitro analiziramo rezultate vsakega izmed modelov ter jih na koncu primerjamo.




Slika 3.1: Primer pripravljenih funkcij za analizo v *Jupyter Notebook*-u.

# Poglavje 4

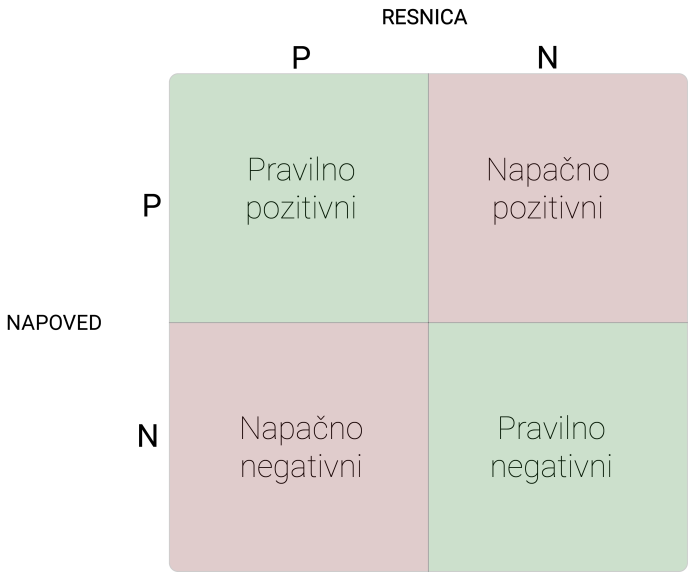
# Rezultati in interpretacija

V tem poglavju predstavimo končne rezultate primerjav in jih interpretiramo. Opišemo načine za merjenje uspešnosti modelov, primerjavo rezultatov naše implementacije z uradno implementacijo faktorizacijskih metod *libFM* ter primerjavo z logistično regresijo v lokalnem in kasneje v produkcijskem okolju.

## 4.1 Načini merjenja uspešnosti

Načini merjenja uspešnosti modelov so pomemben dejavnik vsakega eksperimenta v strojnem učenju. Na našem področju načini ne bodo zgolj statistični, temveč tudi takšni, ki nam povejo nekaj o poslovnem učinku modela na celoten sistem. Če si zamislimo naključen testni primer in naš model, ki izvede napoved za njegovo ciljno spremenljivko (v našem primeru klik ali ne klik), potem obstajajo štirje možni izidi [12]. Če je testni primer pozitiven in ga tudi naš model oceni kot pozitivnega, se to šteje kot *pravilno pozitiven* (angl. *true positive*) primer. Če je primer ocenjen kot negativen, se to šteje kot *napačno negativen* (angl. *false negative*) primer. Če je testni primer negativen in ga tudi naš model oceni kot negativnega, se to šteje kot *pravilno negativen* (angl. *true negative*) primer. Če je primer ocenjen kot pozitiven, se to šteje kot *napačno pozitiven* (angl. *false positive*) primer.

Če te štiri izide dva po dva razvrstimo v matriko (kot na sliki 4.1) dobimo matriko napak (angl. *confusion matrix*).



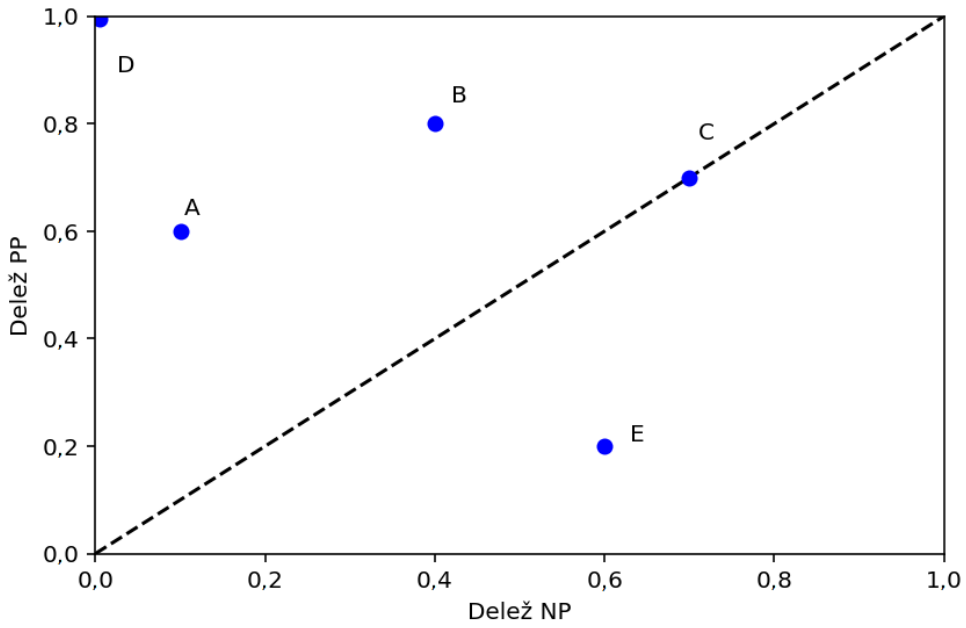

Slika 4.1: Matrika napak je temelj za veliko načinov merjenja uspešnosti napovednih modelov.

Pomembnejši izpeljani količini sta delež pravilno pozitivnih primerov (angl. *true positive rate*) in delež napačno pozitivnih primerov (angl. *false positive rate*), ki jih zapišemo kot:

$$Delež\ PP\ =\ \frac{Pravilno\ uvrščeni\ pozitivni\ primeri}{Vsi\ pozitivni\ primeri} \tag{4.1}$$

$$Delež\ NP\ =\ \frac{Napačno\ uvrščeni\ negativni\ primeri}{Vsi\ negativni\ primeri} \tag{4.2}$$

Ta dva deleža igrata ključno vlogo pri t.i. krivuljah ROC (angl. *Reciever Operating Characteristic*), ki so dvo-dimenzionalna predstavitev izvedbe klasifikatorja.




Slika 4.2: Osnoven primer grafa ROC s petimi diskretnimi klasifikatorji.

Na sliki 4.2 je predstavljenih nekaj pomembnih točk na ROC grafu. Koordinatno izhodišče $(0, 0)$ predstavlja strategijo uvrščanja vseh primerov v negativni razred. Takšen klasifikator ne stori nobene napačno pozitivne napake, ampak obenem tudi ne pridobi nobene pravilno pozitivne uvrstitve. Nasprotna strategija, ki vse primere uvršča v pozitivni razred se nahaja v točki $(1, 1)$. Točka D predstavlja optimalen klasifikator. Neformalno, točka v ROC prostoru je boljša od druge če se nahaja bližje točki $(0, 1)$. Klasifikatorje na levi polovici grafa lahko obravnavamo kot konzervativne, kar pomeni, da primer uvrstijo v pozitivni razred samo takrat, ko imajo zanj zelo močne dokaze, in s tem naredijo zelo malo napačno pozitivnih napak. Takšni klasifikatorji imajo ponavadi nizek delež pravilno pozitivnih uvrstitev. Klasifikatorje na desni polovici grafa označimo za bolj liberalne, kar pomeni, da primer uvrstijo v pozitivni razred že z zelo nizko stopnjo dokazov in s tem pravilno uvrstijo večino pozitivnih primerov. Takšni klasifikatorji imajo običajno visok napačno pozitiven delež. Na sliki 4.2 je točka A bolj konzervativna od točke B. Diagonalna črta $y = x$ predstavlja strategijo naključnega

uvrščanja. Če klasifikator polovico primerov naključno uvrsti v pozitiven razred, potem pričakujemo, da bo pravilno uvrstil polovico pozitivnih in polovico negativnih primerov. Takšna izvedba bi bila v prostoru ROC izražena s točko $(\frac{1}{2}, \frac{1}{2})$. Če klasifikator ugiba pozitivni razred v 90% primerih, potem lahko pričakujemo, da bomo 90% pozitivnih primerov uvrstili pravilno, ampak bo s tem narasel tudi delež napačno pozitivnih primerov, posledično končamo v točki $(\frac{9}{10}, \frac{9}{10})$, To pomeni, da se bo točka naključnega klasifikatorja nahajala na diagonalni črti, kje na njej pa je odvisno od frekvence s katero ugiba pozitivni razred. Če se želimo pomakniti v prostor od $y = x$ levo navzgor, mora klasifikator izkoristiti nekaj informacije iz podatkov. Na sliki 4.2 je izvedba klasifikatorja C v točki $(\frac{7}{10}, \frac{7}{10})$ naključna, pozitiven razred ugiba v 70% primerov.

Izvedba vsakega klasifikatorja na spodnji desni strani $y = x$ je slabša od naključnega ugibanja. Tam običajno v ROC grafih ne najdemo točk. Če klasifikator negiramo, potem vse pravilno pozitivne klasifikacije postanejo napačno negativne in vse napačno pozitivne postanejo pravilno negativne. To pomeni, da vsak klasifikator katerega točka se nahaja v spodnjem desnem delu grafa lahko negiramo in posledično bo njegova točka v zgornjem levem delu grafa. Na sliki 4.2 je izvedba klasifikatorja E občutno slabša od naključja, obenem pa je ta točka negacija točke B. Za vsak klasifikator katerega točka se nahaja na $y = x$ pravimo, da nima informacije o razredu. Klasifikator katerega točka se nahaja pod $y = x$ pravimo, da ima uporabne informacije, ampak jih izkorišča na napačen način. Za možnost primerjave dveh klasifikatorjev želimo poenostaviti ROC graf na samostojno skalarno vrednost, ki predstavlja pričakovano kvaliteto izvedbe. To naredimo z metodo za izračun ploščine pod krivuljo ROC, krajše AUC (angl. *Area Under Curve*).

Da lahko pridobimo krivuljo ROC in izračunamo ploščino pod njo, moramo najprej definirati prag $t \in \mathbb{R}$ (angl. *threshold*), ki je število med 0 in 1, njegova naloga pa je, da loči primere na pozitivne in negativne glede na napovedane verjetnosti. Nato pri vseh vrednostih $t$-ja izračunamo deleža PP

in NP iz česar pridobimo točke na krivulji. AUC nam zagotavlja agregiran način merjenja uspešnosti za vse možne vrednosti $t$-ja [1]. Ker je AUC enak ploščini pod krivuljo znotraj enotskega kvadrata, bo njegova vrednost vedno med 0 in 1. Naključno ugibanje nas pripelje do diagonalne črte med (0,0) in (1,1), ki ima ploščino $\frac{1}{2}$. Noben smiseln klasifikator nima vrednosti AUC manjše od $\frac{1}{2}$. AUC ima pomembno statistično lastnost: AUC klasifikatorja je ekvivalenten verjetnosti, da klasifikator naključno izbran pozitiven primer uvrsti višje od naključno izbranega negativnega primera.

Seveda pa AUC ni brezhiben [21]. Mnogi uporabo AUC za način merjenja kvalitete izvedbe odsvetujejo zaradi številnih razlogov, med drugimi ker:

- **Ne upošteva kako dobro se napovedane verjetnosti ujemajo s pravo verjetnostjo razreda.**
- **Povzema rezultat testiranja na območjih prostora ROC, v katerem bo klasifikator redko deloval.**
- **Enako obravnava napačno pozitivne in napačno negativne primere.**

Kljub temu AUC uporabimo za način merjenja uspešnosti. Z njim želimo izvedeti, kako dobro naš model loči med prikazi oglasov, ki so bolj potencialni od drugih. Poleg AUC, ki je statistična metrika, pa predvsem v produkcijskem okolju potrebujemo še kakšno količino, ki nam pove nekaj o tem, kako dobro naš klasifikator vpliva na finančni izkupiček podjetja. Količine, ki nam opišejo poslovni učinek so:

- **Število zmaganih dražb**
- **Število pridobljenih klikov**
- **Količina porabljenih sredstev**

Iz njih je možno izpeljati še množico ostalih količin kot na primer *eCPC* (angl. *effective cost per click*), ki jo izračunamo kot količnik med količino

porabljenih sredstev in številom pridobljenih klikov. Pove nam, koliko nas povprečno stane en klik, če na dražbah kupujemo s tem klasifikatorjem. Ker želimo za stranko pridobiti kar se da veliko klikov, porabiti pa kar se da malo denarja, je cilj *eCPC* minimizirati.

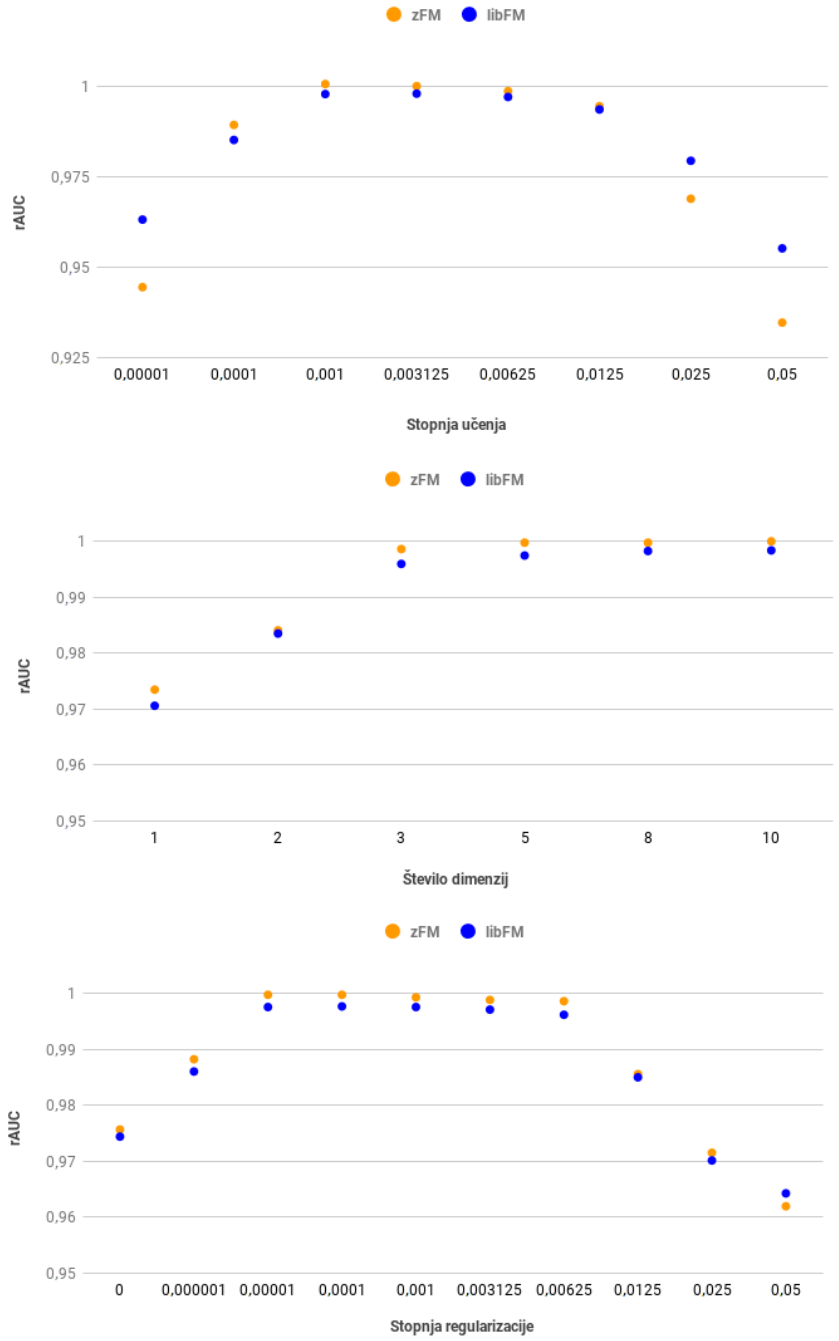
## 4.2 Rezultati lokalnega testiranja

Najprej predstavimo rezultate primerjave naše implementacije faktorizacijskih metod *zFM* z uradno implementacijo *libFM*. Motiv in celoten postopek namestitve, priprave podatkov, nastavljanja parametrov in izvedbe smo opisali v poglavju 3.2. Cilj testiranja ni doseči najboljšo možno izvedbo, temveč se prepričati o pravilnem delovanju naše implementacije. Za primerjavo uporabimo nabor podatkov s 3 milijoni primerov med katerimi je približno 0,5% pozitivnih. Zadnjih 30% podatkov namenimo testiranju. Ker nam knjižnjična implementacija ne omogoča istočasnega učenja in napovedovanja, takšno namestitev uporabimo tudi pri naši implementaciji. Prav tako nam ne omogoča sprotnega kombiniranja značilk, zato za to primerjavo uporabimo le seznam značilk brez kombinacij.

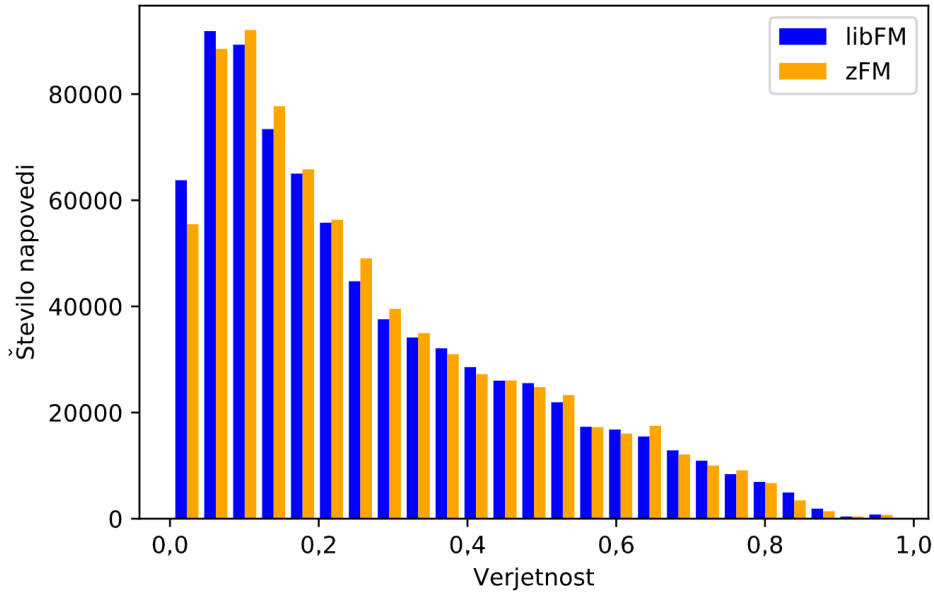
Ker so bili eksperimenti opravljeni na nekriptiranih realnih preteklih podatkih, rezultati razkrivajo poslovno skrivnost. Rezultate primerjav vrednosti AUC predstavimo v obliki (4.3).

$$rAUC(x) = \frac{AUC(x)}{max(AUC)} \tag{4.3}$$

Iz slike 4.3 razberemo podobno obnašanje kvalitet izvedb obeh različic modelov. Grafi so predstavljeni v posebni skali. Na ordinatni osi beležimo vrednost (4.3), na abscisni pa vse preizkušene vrednosti parametrov. Za vsak model in za vsako vrednost parametera izračunamo uspešnost in ju med seboj primerjamo. Za vse izmed štirih vrst parametrov velja, da je odstopanje v najvišji točki manjše od polovice odstotka. Skupaj s potrditvijo, da naša implementacija deluje v pravi smeri, smo pridobili tudi informacije o tem, kje približno se nahajajo bolj optimalne vrednosti parametrov.




Slika 4.3: rAUC pri različnih vrednostih parametrov *zFM* in *libFM*.




Slika 4.4: Histogram porazdelitve napovedanih verjetnosti

Ker za vrednotenje uporabljamo rAUC oziroma AUC, smo zaradi v prejšnjem poglavju navednih pomanjkljivosti, prikrajšani informacije o tem, ali se napovedane verjetnosti ujemajo z dejansko porazdelitvijo razreda. Zato se prepričamo o njihovem skladanju in napovedi enega izmed testiranj predstavimo s histogramom. Iz slike 4.4 razberemo zelo podobni porazdelitvi napovedanih verjetnosti. Poleg grafične predstavitve smo izračunali tudi kalibracijo, ki je enaka razmerju med povprečnim napovedanim CTR in empiričnim CTR [15]. Z drugimi besedami, pove nam kolikšno je razmerje med pričakovanim številom klikov in dejanskim številom klikov. Kalibracija je zelo pomembna mera uspešnosti, saj so natančne in dobro umerjene napovedi pogoj za uspeh v okolju RTB. Manj kot kalibracija odstopa od 1, boljši je model. Rezultati primerjave v tabeli 4.1 potrjujejo pravilnost delovanja našega programa, s tem pa smo korak bližje našemu cilju.

Naslednji del rezultatov pripada primerjavi *zFM* z logistično regresijo. Tudi za ta del testiranja smo motiv, opis podatkov, določitev parametrov in načina izbora značilk opisali v poglavju 3.2. Ker želimo kar se da dobro posnemati produkcijsko okolje, učenje izvajamo tudi na delu podatkov ki

| Algoritem | Kalibracija |
|---|---|
| zFM | 1,02113 |
| libFM | 0,96365 |

Tabela 4.1: Izid kalibracije napovedi *zFM* in *libFM*.

so namenjeni testiranju. Uporabimo oba omenjena seznama značilk t.j. s kombinacijami in brez njih na obeh napovednih modelih.

| Algoritem | Seznam značilk | rAUC |
|---|---|---|
| LR | s kombinacijami | 1,0 |
| zFM | brez kombinacij | 0,98825 |
| LR | brez kombinacij | 0,98344 |
| zFM | s kombinacijami | 0,90267 |

Tabela 4.2: Izidi prve primerjave z logistično regresijo

Iz rezultatov v tabeli 4.2 razberemo, da še nismo na cilju. Rezultat trenutnega produkcijskega modela logistične regresije je boljši od rezultata faktorizacijskih metod. Hkrati zanimivo ter pričakovano je, da se enak seznam značilk produkcijskega modela ni izkazal tudi pri faktorizacijskih metodah. Spodbudno je, da smo se že samo s pomočjo unikatnih značilk in skromnega preiskovanja parametrov tako približali rezultatu produkcijskega modela. Ker po analizi rezultatov prepoznamo perspektivnost algoritma za naš problem, se odločimo opraviti intenzivno preiskovanje značilk tudi za model faktorizacijskih metod. Za to je potrebno dopolniti obstoječe orodje opisano v poglavju 2.2.2. Po uspešni dopolnitvi in daljšem preiskovanju smo pripravljeni na novo primerjavo. Seznam značilk prirejen za faktorizacijske metode še vedno ne vsebuje tako kompleksnih kombinacij kot obstoječi produkcijski model, saj vsebuje kombinacije z največ dvema značilkama. Kljub temu te veliko prispevajo k končni uspešnosti.

| Algoritem | rAUC |
|---|---|
| zFM | 1,0 |
| LR | 0,99085 |

Tabela 4.3: Izid primerjave logistične regresije s faktorizacijskimi metodami po preiskovanju seznama značilk.

Z novim seznamom značilk nam uspe premagati produkcijski model, kar je odlična popotnica za nadaljevanje. Zaradi uspešnih testiranj v lokalnem okolju se odločimo test preseliti v produkcijsko okolje.

## 4.3 Rezultati produkcijskega testiranja

Opis produkcijskega okolja, priprave modelov in orodij za analizo najdemo v poglavju 3.3. Rezultate produkcijskega testiranja ločimo na dva dela t.j. statistične in finančne. Pri statističnih rezultatih smo poleg glavnega načina merjenja uspešnosti rAUC izmerili tudi kalibracijo. A/B test je trajal 7 dni na izbranih oglaševalskih kampanijah, med potekom pa smo redno spremljali podatke o dogajanju.

| Algoritem | rAUC | Kalibracija |
|---|---|---|
| zFM | 1,0 | 0,99255 |
| LR | 0,98527 | 1,12351 |

Tabela 4.4: Statistični rezultati produkcijskega A/B testa.

Iz statističnih rezultatov iz tabele 4.4 ugotovimo, da je AUC modela *zFM* za slab odstotek in pol boljši od AUC modela LR. Tudi kalibracija kaže v korist modelu faktorizacijskih metod. Preverili smo tudi povprečne napovedi za vsakega izmed razredov. Ugotovili smo, da so napovedi faktorizacijskih metod pri negativnih primerih v povprečju manjše od napovedi logistične regresije. Tudi za pozitivne primere je bil rezultat v korist *zFM*, saj so bile napovedi v povprečju za pol odstotka višje. Iz tega sklepamo, da faktorizacij-

ske metode bolje ločijo med pozitivnimi in negativnimi primeri. Kvalitetnejša izvedba se je dobro prenesla iz lokalnega v produkcijsko okolje.

Pri finančnih rezultatih kot že omenjeno v poglavju 4.1, preverimo vrednosti povezane s poslovnim učinkom našega napovednega modela na celoten sistem. Finančni rezultati so največkrat tisti, ki na koncu odločijo o tem, ali produkcijski model nadomestimo z novejšim ali ne.

| **Algoritem** | **Sredstva** | **Prikazi** | **Kliki** | **eCPC** |
|---|---|---|---|---|
| zFM | $23.054 | 45.786.471 | 141.393 | 0,16305 |
| LR | $24.329 | 47.438.645 | 133.660 | 0,18202 |

Tabela 4.5: Rezultati finančnega učinka v produkcijskem okolju.

Iz finančnih rezultatov v tabeli 4.5 razberemo, da je model *zFM* v primerjavi z LR porabil 5% manj sredstev ter dosegel dobre 3% manj prikazov. Dobra novica je, da je *zFM* uspel pridobiti dobrih 5% klikov več. Posledično je povprečna cena s katero je *zFM* pridobil klik za 10% nižja. Z manj sredstvi pridobimo večji poslovni učinek. To nam je uspelo storiti z napovednim modelom, ki zna bolje ločiti med prikazi, ki so bolj potencialni od drugih. Takšna izboljšava občutno vpliva na storitev, ki jo ponuja podjetje, prav tako pa tudi na njegovo poslovanje. S tem je cilj našega eksperimenta dosežen.

# Poglavje 5

# Zaključek

V uvodnih poglavjih diplomske naloge predstavimo spletno oglaševanje ter se seznanimo s ključnimi pojmi in koncepti, ki jih srečujemo na tem področju. Bolj podrobno se osredotočimo na spletno oglaševanje, ki temelji na ekosistemu RTB. Predstavimo problem napovedovanja razmerja med prikazi in kliki ter teoretično opišemo dve izmed možnih metod za spopadanje s tem problemom. Opišemo postopek predpriprave podatkov ter izbire značilk, kateremu sledi opis implementacije faktorizacijskih metod. Zatem se lotimo opisa testiranja ter končnih rezultatov.

Skozi celoten proces pisanja diplomske naloge se soočimo s kar nekaj izzivi. Večji izmed njih je zagotovo seznanitev z obstoječim sistemom in domeno ter dopolnitev orodja za odkrivanje značilk. Pri soočanju z izzivi spoznamo področje spletnega oglaševanja, programski jezik *Go*, orodja za analizo in spremljanje dogajanja, orodja za celovit razvoj in integracijo programske opreme, pristope v strojnem učenju in testiranju programske opreme itd.

Na koncu nam uspe doseči zastavljen cilj ter izboljšati obstoječi sistem, ki je temeljil na logistični regresiji. Izboljšati rešitev za tako pomemben problem v podjetju, ki se ukvarja s spletnim oglaševanjem, kot je napovedovanje CTR, je lep dosežek. Pri tem je vredno omeniti, da delo na tem problemu še zdaleč ni končano. S pridobivanjem in odkrivanjem novih značilk ter pristopov je sistem potrebno vzdrževati ter izboljševati. Ena izmed smeri, ki bi

lahko prinesla še boljše rezultate, je nadgradnja implementiranega algoritma v t.i. FFM (angl. *Field-aware Factorization Machine*) [17]. Ta nadgradnja za ceno večje časovne zahtevnosti algoritmu doda pojem polja, ki omogoča še bolj podrobno povezovanje značilk in s tem boljše rezultate na redkih podatkih. Druga izmed smeri, ki bi lahko prinesla izboljšavo sistema je prilagojena stopnja učenja glede na vrednost značilke. Poznamo več različnih metod optimizacije gradientnega spusta. Ena izmed enostavnejših in za nas primernih je metoda *ADAGRAD*, ki nam omogoča hitrejšo konvergenco in s tem boljše rezultate na redkih podatkih. Tretja in tudi zadnja smer, v katero bi se lahko zatekli pa je reševanje problema raziskovanja in izkoriščanja (angl. *explore and exploit*). Naš algoritem po svoji definiciji strmi k optimalnosti, saj vedno izbere tisti oglas, za katerega je verjetnost klika največja. Posledično bomo izbirali oglase za katere vemo, da delujejo dobro. V sistem dnevno prihajajo novi oglasi, za katere nimamo informacij o tem, kako dobri so. Problem trenutno rešujemo tako, da manjši delež prometa kupujemo naključno z oglasi za katere nimamo podatkov. Obstajajo bolj optimalne rešitve, kot na primer Bayesovske Faktorizacijske Metode (angl. *Bayesian Factorization Machines*), ki klasičnemu algoritmu dodajo pojem Bayesovskega sklepanja [13].

# Literatura


[1] Classification: ROC curve and AUC — Machine Learning Crash Course — Google Developers. Dosegljivo: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc/. Dostopano: 10.12.2018.

[2] The EU General Data Protection Regulation (GDPR). Dosegljivo: https://eugdpr.org/. Dostopano: 02.10.2018.

[3] The Go Programming Language. Dosegljivo: https://golang.org/. Dostopano: 04.11.2018.

[4] libFM: Factorization Machine Library. Dosegljivo: http://www.libfm.org/. Dostopano: 27.10.2018.

[5] Marketing week: What is programmatic advertising? Dosegljivo: https://www.marketingweek.com/2017/03/27/programmatic-advertising/. Dostopano: 30.07.2018.

[6] ML Data - Machine Learning Datasets - Balloons. Dosegljivo: https://www.mldata.io/dataset-details/balloons/. Dostopano: 30.11.2018.

[7] Poslovni tednik - Svet Kapitala. Dosegljivo: https://svetkapitala.delo.si/aktualno/deset-najboljsih-obcin-za-investicije-6907/. Dostopano: 18.11.2018.

[8] Wikipedia - Logistic regression. Dosegljivo: `https://en.wikipedia.org/wiki/Logistic_regression/`. Dostopano: 12.12.2018.

[9] Earl T Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. The oracle problem in software testing. *IEEE transactions on software engineering*, 41(5):507–525, 2015.

[10] Bowei Chen, Shuai Yuan, and Jun Wang. A dynamic pricing model for unifying programmatic guarantee and real-time bidding in display advertising. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, pages 1–9. ACM, 2014.

[11] David S Evans. The online advertising industry: Economics, evolution, and privacy. *Journal of Economic Perspectives*, 23(3):37–60, 2009.

[12] Tom Fawcett. An introduction to ROC analysis. *Pattern Recogn. Lett.*, 27(8), June 2006.

[13] Christoph Freudenthaler, Lars Schmidt-Thieme, and Steffen Rendle. Bayesian factorization machines. 2011.

[14] Thore Graepel, Joaquin Quinonero Candela, Thomas Borchert, and Ralf Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. Omnipress, 2010.

[15] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, pages 1–9. ACM, 2014.

[16] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[17] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for ctr prediction. In *Proceedings of the*

*10th ACM Conference on Recommender Systems*, pages 43–50. ACM, 2016.

[18] Igor Kononenko and Marko Robnik Šikonja. *Inteligentni sistemi*. Založba FE in FRI, 2010.

[19] SB Kotsiantis, Dimitris Kanellopoulos, and PE Pintelas. Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2):111–117, 2006.

[20] Huan Liu, Hiroshi Motoda, Rudy Setiono, and Zheng Zhao. Feature selection: An ever evolving frontier in data mining. In *Feature Selection in Data Mining*, pages 4–13, 2010.

[21] Jorge M Lobo, Alberto Jiménez-Valverde, and Raimundo Real. Auc: a misleading measure of the performance of predictive distribution models. *Global ecology and Biogeography*, 17(2):145–151, 2008.

[22] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2013.

[23] Glenford J Myers, Corey Sandler, and Tom Badgett. *The art of software testing*. John Wiley & Sons, 2011.

[24] Steffen Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, 2010.

[25] Steffen Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May 2012.

[26] Ashish Kumar Singh and Vidyasagar Potdar. Blocking online advertising-a state of the art. In *Industrial Technology, 2009. ICIT 2009. IEEE International Conference on*, pages 1–10. IEEE, 2009.

[27] Lorenzo Trippa, Levi Waldron, Curtis Huttenhower, Giovanni Parmigiani, et al. Bayesian nonparametric cross-study validation of prediction methods. *The Annals of Applied Statistics*, 9(1):402–428, 2015.

[28] Jun Wang, Weinan Zhang, and Shuai Yuan. Display advertising with real-time bidding (rtb) and behavioural targeting. *arXiv preprint arXiv:1610.03013*, 2016.

[29] Jun Yan, Ning Liu, Gang Wang, Wen Zhang, Yun Jiang, and Zheng Chen. How much can behavioral targeting help online advertising? In *Proceedings of the 18th international conference on World wide web*, pages 261–270. ACM, 2009.

[30] Yong Yuan, Feiyue Wang, Juanjuan Li, and Rui Qin. A survey on real time bidding advertising. In *Service Operations and Logistics, and Informatics (SOLI), 2014 IEEE International Conference on*, pages 418–423. IEEE, 2014.

[31] Weinan Zhang, Shuai Yuan, and Jun Wang. Optimal real-time bidding for display advertising. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1077–1086. ACM, 2014.

[32] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1059–1068. ACM, 2018.