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

Miha Štravs

Napovedovanje uspešnosti aplikacij v trgovini Google Play

diplomsko delo

interdisciplinarni univerzitetni
študijski program prve stopnje
računalništvo in matematika

Mentor: prof. dr. Zoran Bosnić
Somentor: Luka Kacil

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednje delo:

Tematika dela:

Kandidat naj v diplomski analogi analizira podatke o popularnosti aplikacij v spletni trgovini Google Play. Iz obstoječih zapisov naj oblikuje ustrezne atribute za nadzorovano učenje. Ker inkrementi podatkov prihajajo v dnevnih časovnih intervalih, naj uporabi in primerja uspešnost različnih metod za inkrementalno učenje. Diplomsko delo naj zaključi z ugotovitvami, katere metode in atributi dosegajo najboljše rezultate za uporabo v praksi.

Zahvaljujem se mentorju prof. dr. Zoranu Bosniću in somentorju Luki Kacilu za številne nasvete pri izdelavi diplomske naloge.
Zahvaljujem se tudi podjetju AppMonsta, ki je bilo pripravljeno deliti podatke o aplikacijah trgovine Google Play.
Nazadnje se zahvaljujem še svoji družini za nenehno podporo v času študija in pisanju diplomskega dela.

Kazalo

Povzetek

Naslov: Napovedovanje uspešnosti aplikacij v trgovini Google Play

Avtor: Miha Štravs

V času, ko ima skoraj vsaka oseba pametni telefon, je naraslo povpraševanje po mobilnih aplikacijah. Zaradi velikega povpraševanja se je povečalo tudi število podjetij in posameznikov, ki aplikacije razvijajo. To je povzročilo izdelavo številnih aplikacij, ki jih lahko posameznik prenese iz spletnih trgovin. Zaradi velikega števila različnih aplikacij le majhen delež teh uspe. V tem diplomskem delu pokažemo, kako dobro lahko s pomočjo metod strojnega učenja na podlagi podatkov, ki jih pridobimo iz trgovine Google Play, napovemo, katere aplikacije bodo v prihodnosti uspele. Najprej se seznanimo s področjem učenja iz podatkovnih tokov. Nato pregledamo, kateri podatki o aplikacijah so nam na voljo in iz njih izpeljemo nove atribute, ki se bodo uporabili pri napovedovanju. Za napovedovanje uporabimo že implementirane metode paketa MOA. Uporabijo se metode naivni Bayes, Hoeffdingova drevesa in drevesa IADEM. Testi so se izvedli na različnih metodah, na različni velikosti učnih podatkov in na različni časovni dolžini napovedi Uspešnost napovedi metode smo ocenili s klasifikacijsko točnostjo, srednjo vrednostjo absolutne napake, relativno srednjo vrednostjo absolutne napake in oceno F1 za vsak razred. Od vseh metod se je najbolje izkazala metoda dreves IADEM.

Ključne besede: strojno učenje, podatkovni tokovi, MOA, klasifikacija, mobilne aplikacije.

Abstract

Title: Predicting success of applications in Google Play store

Author: Miha Štravs

In times when almost everybody has a smartphone, a demand for mobile apps has risen. Because of the high demand, a lot of businesses and individuals have started to develop mobile apps. A big number of new mobile apps is being made available in the Google Play every day. Because of the number of different apps, only a few become popular and successful. In this thesis, we evaluate how well can we predict the success of mobile applications using the machine learning algorithms on data from Google Play. First, we overview the machine learning algorithms used for data streams. Then we describe the available data and derive the appropriate attributes. For prediction, we use already implemented methods from the MOA package: Naive Bayes, Hoeffding trees and IADEM trees are used. Methods are then tested by using a different amount of data and different prediction time lengths. The success is then measured using the classification accuracy, mean absolute error, relative mean absolute error and the F1 score for each class. The IADEM trees had the best scores from all the methods used.

Keywords: machine learning, data streams, MOA, classification, mobile apps.

Poglavje 1

Uvod

Prenosni telefoni imajo v našem vsakdanjem življenju velik pomen. Razlog temu je prihod pametnih telefonov, ki poleg storitev, kot so klicanje in pošiljanje sporočil, prinašajo tudi možnost iskanja po internetu in uporabo številnih aplikacij, ki jih najdemo v virtualnih trgovinah. Aplikacij je veliko in se jih uporablja za komunikacijo, učenje, kuhanje, dostop do družabnih omrežij, nakupovanje, bančništvo itd. Zaradi velikega povpraševanja po novih in boljših aplikacijah je bila opažena velika rast števila podjetij in posameznikov, ki razvijajo nove aplikacije. Zato ni presenetljivo, da so mobilne aplikacije v zadnjem času postale eden izmed najbolj dobičkonosnih poslov. Lastniki aplikacij služijo s prodajo aplikacij v spletnih trgovinah in z oglaševanjem v njih. V prvi polovici leta 2018 sta imeli trgovini Google Play in Apple App Store skupaj kar 34,4 milijard dolarjev bruto prihodka od prodaje po vsem svetu [6].

Kljub velikemu dobičku, ki jih nekatere uspešne aplikacije prinesejo, jih veliko ne povrne časovnega in finančnega vložka. Vsakodnevno se v spletnih trgovinah, kot Google Play in Apple App Store, pojavi več tisoč novih aplikacij. Leta 2015 je šestdeset odstotkov razvijalcev imelo prihodek manjši od 500 dolarjev na mesec [8], kar je pod mejo minimalnih plač v razvitem svetu. Vsako leto se priljubljenost in prodaja aplikacij povečuje in s tem tudi skupni bruto prihodek njihovih razvijalcev. To pa ne pomeni, da se povečuje delež

razvijalcev, ki zaslužijo dovolj za dostojno življenje le z razvijanjem aplikacij.

V tem diplomskem delu se bomo posvetili napovedovanju uspešnosti trženja aplikacij (števila novih prenosov aplikacij) trgovine Google Play s pomočjo metod za učenje iz podatkovnih tokov. Uspešnost metod bomo ocenili s klasifikacijsko točnostjo ter z ocenami F1 za vsak razred napovedi. Uporabnost atributov pri napovedovanju bomo ocenili z Gini indeksom.

Delo je razdeljeno na pet poglavij. Poglavje 2 vsebuje teoretično ozadje učenja iz podatkovnih tokov, poglavje 3 opisuje podatke, ki so bili uporabljeni v eksperimentu, poglavje 4 prikaže izvedene teste in pojasnjene rezultate. V poglavju 5 povzamemo ugotovitve diplomskega dela.

Poglavje 2

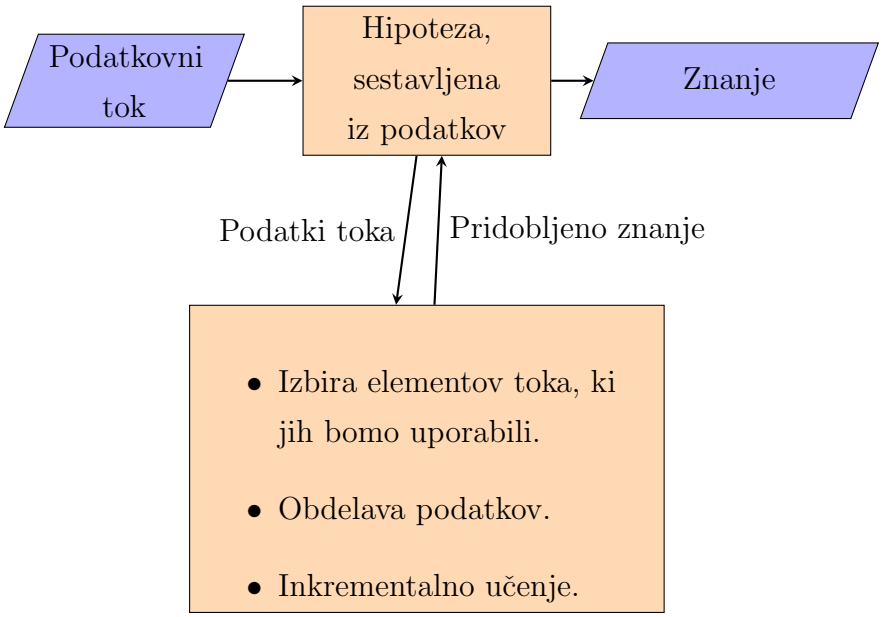
Učenje iz podatkovnih tokov

S pojavom številnih sistemov, ki generirajo velike tokove podatkov, je nastala potreba po metodah za obdelavo teh podatkov. Primeri takih sistemov so sistemi za nadzor in varnost, telekomunikacijski sistemi ter senzorji. Iz podatkovnih tokov pridobimo informacije o strukturi in vzorcih z uporabo metod za strojno učenje.

Strojno učenje oziroma odkrivanje znanja iz podatkov je proces, kjer iz podatkov pridobimo informacije o strukturi, vzorcih in pravilih v teh podatkih. Metode za strojno učenje so bile na začetku ustvarjene za uporabo na relacijskih podatkovnih bazah, katerih velikost se skozi čas ne povečuje. To pri tokovih ne drži, ker skozi čas pridobivamo nove podatke. Preko podatkovnega toka nenehno pridobivamo podatke, ki jih potem shranimo ali obdelamo. Veliko metod za strojno učenje zahteva večkratno iteracijo čez podatke, česar si v neskončno velikih podatkovnih tokovih žal ne moremo privoščiti. V zelo velikih tokovih je že samo ena iteracija preveč in moramo izbrati majhen del podatkov, ki dobro predstavljajo ostale. Tako dobimo prva dva pogoja metod za podatkovne tokove. Velikokrat tudi nimamo možnosti vpliva na vrstni red samih podatkov, ki jih pridobimo preko toka. V toku se lahko pojavi tudi sprememba porazdelitve podatkov (angl. concept drift). Dobra metoda mora torej imeti tudi detektor za zaznavo spremembe porazdelitve podatkov in se mora nanjo primerno odzvati.

Ogrodje metode za izvajanje učenja iz podatkovnih tokov prikazano na sliki 2.1, ustreza naslednjim zahtevam:

- Podatki iz tokov prihajajo neprestano. Model mora imeti možnost klasifikacije v katerem koli trenutku.
- Sistem nima nadzora nad vrstnim redom podatkov.
- Tok je lahko neskončen.
- Ko se element enkrat obdela, se ga zavrže oziroma začasno shrani v spomin, ki je veliko manjši od velikosti podatkovnega toka.
- Metoda mora zaznati spremembo porazdelitve podatkov in se nanjo primerno odzvati.



Slika 2.1: Ogrodje metode za učenje iz tokov

Tok ima lahko več podatkov, kot jih lahko na enkrat hranimo v spominu. Ena rešitev je uporaba inkrementalnih algoritmov, kjer lahko vsak primer zavržemo po enkratni obdelavi. V primerih, ko uporabimo drugačne metode

oziroma, ko je podatkov preveč tudi za inkrementalne metode, pride do problema prekoračitve prostorske ali časovne omejitve. Za omenjeni problem je bilo predlaganih več rešitev, ki jih lahko razdelimo v dve skupini: med rešitve, kjer podatke povzamemo, da so dovolj majhni za metodo, in rešitve, kjer se metoda za strojno učenje prilagodi veliki količini podatkov. V nadaljevanju podrobneje opisujemo obe skupini metod.

1. Metode za obdelavo podatkov:

   - Vzorčenje (angl. sampling) podatkov je metoda za izbiro elementov na podlagi verjetnosti. Preprost način vzorčenja je, da izberemo periodo, po kateri bomo izbirali elemente. Problem pri vzorčenju na podatkovnem toku je to, da ne poznamo njegove velikosti. Če je tok neskončno velik, je tudi množica izbranih elementov neskončna ne glede na periodo. Temu problemu se izognemo z uporabo rezervoarjev [9]. Rezervoar hrani k elementov. Na začetku ga napolnimo s prvimi k elementi toka. Nato iteriramo čez preostali del toka. n-ti element bo z verjetnostjo $\frac{k}{n}$ zamenjal naključni element v rezervoarju.
   - Nekateri tokovi imajo lahko v določenih časovnih obdobjih povečano količino podatkov. Na primer tok, ki hrani informacije klicev klicnega centra, ima popoldne največjo količino podatkov. Če metoda ne zmore preprocesirati vseh podatkov pri povečavah, se uporabi razbremenitev (angl. load shedding) [2], ki del podatkov zavrže.

2. Prilagajanje metod za obdelavo velike količine podatkov:

   - Drseča okna dajejo prednost novejšim podatkom. Na podatkih, ki so trenutno v oknu, se izvede podrobnejša analiza, medtem ko se za starejše podatke uporabi le shranjen povzetek.
   - Nekatere metode lahko prevedemo v inkrementalne s pomočjo mej, izpeljanih iz statističnih zakonitosti. Primer take zakonitosti je

Chebyshejev teorem. Chebyshejev teorem pravi, da pri veliki količini naključno izbranih med seboj neodvisnih primerov drži, da je verjetnost, da je vrednost naključno izbranega primera X oddaljena od povprečne vrednosti μ manj kot za k-krat standardnega odklona σ, enaka $\frac{1}{k^2}$.

$$P(|X - \mu| \leq k\sigma) \leq \frac{1}{k^2}$$

Iz teorema sta izpeljani Chernoffova in Hoeffdingova meja [5], ki sta uporabljeni za gradnjo Hoeffdingovih dreves in dreves IADEM, opisanih v razdelku 2.1.

2.1 Klasifikacija

Metode za klasifikacijo ugotavljajo, kateremu izmed končno mnogih razredov primer pripada. Klasifikacija je primer nadzorovanega učenja. To pomeni, da se za učenje uporabi množica s primeri, ki imajo že znan razred. Primeri metod za klasifikacijo so nevronske mreže, metoda podpornih vektorjev, metoda k-najbližjih sosedov, naivni Bayes in odločitvena drevesa.

2.1.1 Večinski razred

Metoda poišče razred, ki vsebuje največ primerov v učni množici. Vse nove primere klasificira s tem razredom. Uporablja se za določanje spodnje pričakovane meje uspešnosti ostalih metod. Če katera od metod deluje slabše od metode večinskega razreda, je neuporabna.

2.1.2 Naivni Bayes

Bayesov klasifikator [7] je izpeljan iz Bayesovega teorema in predpostavlja pogojno neodvisnost vrednosti atributov pri danem razredu.

Klasifikator naivnega Bayesa napove tisti razred, ki maksimizira verjetnost $p(r|a_1, a_2, ..., a_n) = p(r) \times \prod_{i=1}^{n} \frac{p(r|a_i)}{p(r)}$, kjer je $p(A|B)$ pogojna verjetnost A pri podanem B, r je razred in a_1, a_2, ..., a_n so vrednosti atributov.

2.1.3 Hoeffdingova drevesa

Klasifikacija z odločitvenim drevesom poteka s potovanjem primera od korena drevesa do lista. Odločitveno drevo ima v vsakem vozlišču test glede na atribut primera, ki loči prostor glede na vrednost atributa. List, v katerem konča, določi razred primera. Da so ločitve prostora dovolj dobre, se uporabljajo hevristike, kot sta Gini indeks in informacijski prispevek. Te mere potrebujejo dostop do vseh učnih primerov hkrati, torej ne ustrezajo pogojem za klasifikacijo podatkovnih tokov.

Domingos in Hulten [7] sta predlagala algoritem za gradnjo odločitvenih dreves, ki se uporablja na podatkovnih tokovih. Ta algoritem inkrementalno gradi odločitveno drevo. Ko obdela primer, ga takoj zavrže in hrani v spominu le drevo. Pri gradnji uporablja t.i. Hoeffdingovo mejo, ki nadomesti prej naštete mere nečistosti. Hoeffdingova meja je dokazana z enačbo.

Vzemimo n naključno izbranih med seboj neodvisnih realnih spremenljivk z intervala $[0, R]$. Naj bo $\overline{r}$ povprečje teh spremenljivk. Hoeffdingova meja pravi, da je z verjetnostjo $1 - \delta$ prava srednja vrednost vsaj $\overline{r} - \epsilon$, kjer je $\epsilon = \sqrt{\frac{R^2 \times ln(\frac{2}{\delta})}{2n}}$.

V algoritmu za grajenje Hoeffdingovih dreves naključna spremenljivka predstavlja razliko količine pridobljene informacije med dvema najboljšima atributoma. Če je najboljši atribut X_a in drugi najbolši atribut X_b, je po n primerih razlika median $\Delta\overline{G} = \overline{G}(X_a) - \overline{G}(X_b) \geq \epsilon^2 \geq 0$. Potem Hoeffdingova meja pravi, da je X_a boljša izbira za delitev z verjetnostjo $1 - \delta$ po n-tih primerih. Če drži $\Delta\overline{G} > \epsilon$, Hoeffdingova meja zagotavlja, da za pravo razliko ΔG drži $\Delta G \geq \Delta\overline{G} - \epsilon > 0$. Tako algoritem na vsakih n vzorcev določi najboljši atribut in po njem razdeli vozlišče.

2.1.4 Drevesa IADEM

IADEM (Incremental algorithm driven by error margin) [4] slovensko: algoritem, zgrajen glede na toleranco napake, je tudi inkrementalni algoritem za grajenje odločitvenih dreves. Na začetku uporabnik nastavi dva parametra:

največjo možno napako ϵ in verjetnost, da se bo algoritem držal te napake $1 - \delta$. Algoritem v vsakem trenutku hrani zgornjo in spodnjo mejo napake odločitvenega drevesa.

- Če je zgornja meja pod ϵ, pomeni, da je napaka dovolj majhna in se v primeru, da je algoritem dosegel minimalno število prejetih podatkov, gradnja drevesa ustavi.
- Če je ϵ med zgornjo in spodnjo mejo, algoritem sprejema nove podatke, a ne razširja drevesa.
- Če je ϵ pod spodnjo mejo, algoritem sprejema podatke in po možnosti naprej gradi drevo.

Algoritem uporablja dva tipa vozlišč na robu dreves – virtualna in prava. Prava vozlišča so del drevesa. Vsako pravo vozlišče ima virtualno vozlišče za vsako možno razširitev. Preko Chernoffove in Hoeffdingove meje s podatki iz vozlišč nato izračunamo prej omenjene vrednosti, ki vplivajo na gradnjo drevesa.

2.2 Vrednotenje

Da lahko ocenimo uspešnost metod in kakovost uporabljenih atributov, potrebujemo mere za njihovo vrednotenje.

2.2.1 Klasifikacijska točnost

Klasifikacijska točnost (angl. classification accuracy) je osnovna mera za vrednotenje metod. Klasifikacijska točnost je enaka $CA = \frac{\text{pravilno klasificirani primeri}}{\text{vsi primeri}}$.

2.2.2 Priklic, preciznost in F1 ocena

Te tri mere se uporabljajo pri vrednotenju sistemov za pridobivanje informacij (angl. information retrival). Priklic (angl. recall) pove, kolikšen delež

iskanih dokumentov smo pridobili. Priklic = $\frac{|\text{najdeni pravilni dokumenti}|}{|\text{vsi pravilni dokumenti}|}$. Preciznost (angl. precision) pove delež pridobljenih dokumentov, ki so pravilni. Preciznost = $\frac{|\text{najdeni pravilni dokumenti}|}{|\text{vsi najdeni dokumenti}|}$. Ocena F1 je sestavljena iz prejšnjih dveh in se uporablja kot merilo za uspešnost metode.

$$F1 = 2 \times \frac{\text{preciznost} \times \text{priklic}}{\text{preciznost} + \text{priklic}}$$

Pri vrednotenju klasifikacije izračunamo vrednost F1 za vsako vrednost razreda, kjer je:

$$\text{priklic} = \frac{|\text{pravilno klasificirani primeri}|}{|\text{vsi primeri z iskano vrednostjo}|}$$

$$\text{preciznost} = \frac{|\text{pravilno klasificirani primeri}|}{|\text{vsi primeri klasificirani z iskano vrednostjo}|}$$

2.2.3 Srednja absolutna napaka in relativna srednja absolutna napaka

Srednja absolutna napaka (angl. mean absolute error) se uporablja za izračun povprečnega odstopanja napovedane vrednosti od prave vrednosti. Izračunamo jo kot $E = \frac{1}{N}\sum_{i=1}^{N}|f(i) - \hat{f}(i)|$, kjer je $f(i)$ napovedana vrednost primera i in $\hat{f(i)}$ pravilna vrednost primera i.

Relativna srednja absolutna napaka se uporablja za preverjanje kvalitete metode napovedovanja. Če je relativna srednja absolutna napaka večja kot 1, pomeni, da je napovedovanje s povprečno vrednostjo bolj natančno. Relativno napako izračunamo $RE = \frac{N \times E}{\sum_{i=1}^{N}|f(i) - \overline{f}|}$, kjer je $\overline{f}$ povprečna vrednost napovedi.

2.2.4 Vrednotenje kakovosti atributov

Za vrednotenje kakovosti atributov se uporablja mere nečistosti. Te mere nam povejo količino pridobljenih informacij, ki jih atribut ima. V odločitvenih drevesih se uporabljajo za ocenjevanje deljenja podatkov v listih po vrednosti

atributa. Primer mere nečistosti je Gini indeks.

$$\text{Gini}(A) = \sum_j p_j \sum_k p_{k|j}^2 - \sum_k p_k^2$$

Kjer je p_j delež primerov, kjer ima atribut A vrednost j, $p_{k|j}$ je delež primerov, kjer ima atribut A vrednost j in ciljna spremenljivka vrednost k, deljen s p_j, in p_k je delež primerov, kjer ima ciljna spremenljivka vrednost k.

2.3 MOA

MOA (**M**assive **O**nline **A**nalysis) [3] je orodje, implementirano v Javi, z metodami za klasifikacijo, gručenje in regresijo podatkovnih tokov. Je eksperimentalno orodje, izdelano v univerzi Waikato s številnimi algoritmi za strojno učenje, generacijo podatkovnih tokov in evalvacijo uspešnosti algoritmov na podatkovnih tokovih.

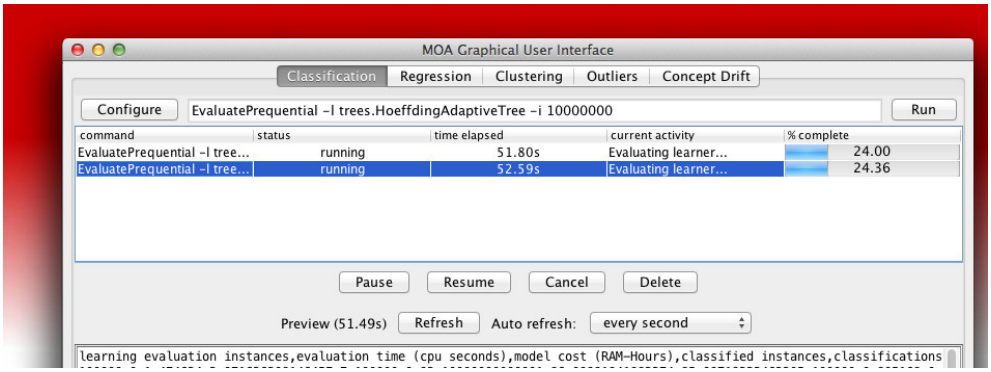
Uporaba MOA je sestavljena iz treh glavnih delov:

1. izbira podatkovnega toka,

2. izbira algoritma za strojno učenje in nastavitve parametrov,

3. uporaba metode za evalvacijo algoritma na izbranem podatkovnem toku.

Delo z MOA olajša grafični uporabniški vmesnik, prikazan na sliki 2.2. Med delovanjem se rezultati in podatki o delovanju sproti izpisujejo v poročilo, ki se ga lahko shrani po končanem procesu. Pridobljene podatke in rezultate lahko tudi vizualizira v obliki grafa.



Slika 2.2: Orodje MOA [1]

Poglavje 3

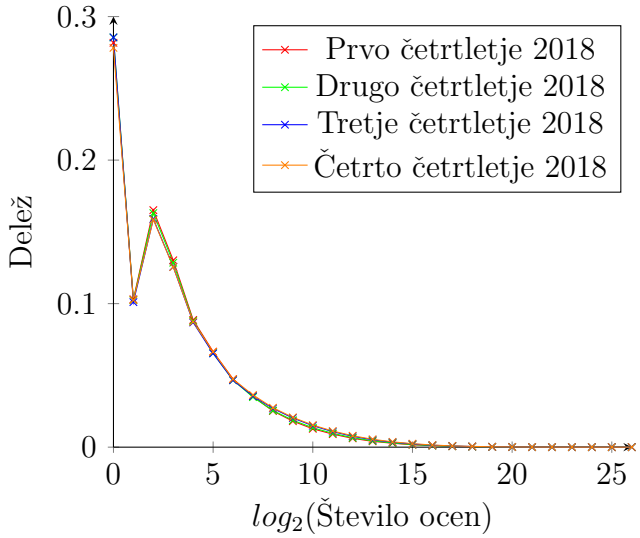
Obdelava podatkov

Podatke, na katerih se napoveduje uspešnost aplikacij, smo pridobili iz arhiva, ki ga hrani podjetje AppMonsta. AppMonsta zbira podatke o aplikacijah iz 155 držav. Novi podatki o aplikacijah prihajajo vsaki dan. Do njihovih podatkov se dostopa preko vmesnika Rest API. Pridobljeni podatki so v obliki dokumenta JSON za vsako aplikacijo posebej.

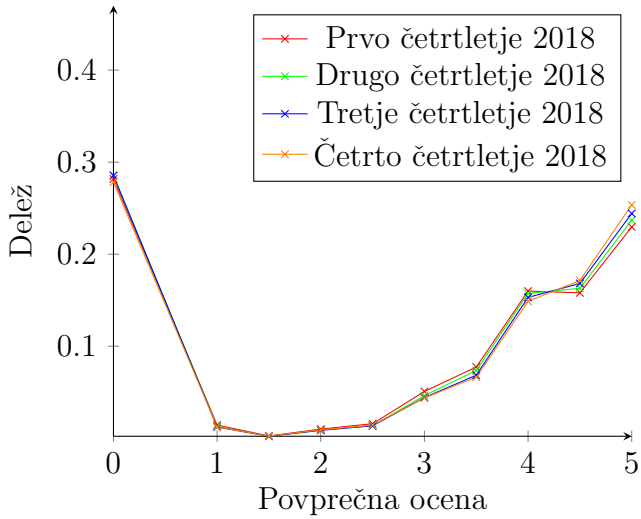
3.1 Porazdeljenost vrednosti atributov v letu 2018

Za prikaz porazdeljenosti vrednosti atributov smo iz podatkov, pridobljenih iz leta 2018 na osmi in triindvajseti dan vsakega meseca, izdelali grafe. Vsak graf vsebuje štiri krivulje. Krivulje kažejo delež primerov za vsako vrednost atributa za podatke zbrane iz vsakega četrtletja.

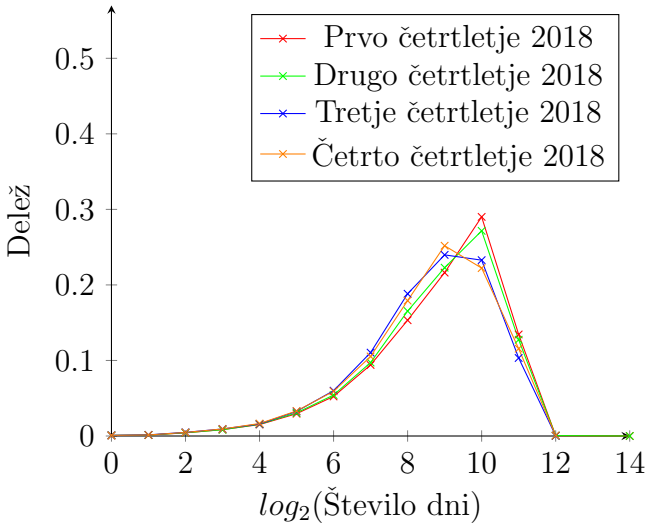
Iz grafikonov na slikah 3.1 in 3.4 vidimo, da je aplikacij z velikim številom prenosov relativno malo. Če bo ciljni atribut za napovedovanje število prenosov aplikacije, bodo aplikacije, ki imajo večje število prenosov, manj zastopane. Zato smo kot ciljni atribut izbrali spremembo v številu prenosov. Opazili smo, da se skozi leto 2018 porazdelitev atributov, prikazanih na slikah 3.1 - 3.4, skoraj ne spreminja.



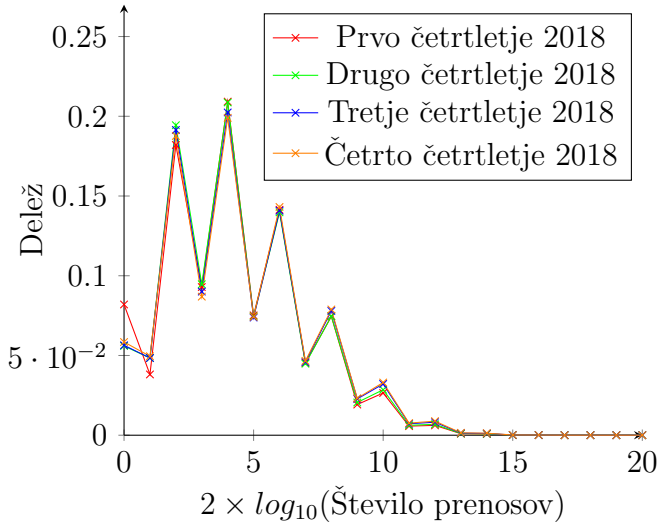
Slika 3.1: Število ocen aplikacije



Slika 3.2: Povprečna ocena aplikacije



Slika 3.3: Čas obstoja aplikacije na trgu



Slika 3.4: Število prenosov aplikacije

3.2 Učni podatki

Podatki, ki so uporabljeni v testih, so bili zbrani od junija 2017 do februarja 2019 na osmi dan vsakega meseca. Testi so razdeljeni v pet skupin glede na

časovno dolžino napovedi uspešnosti: en mesec, dva meseca, tri mesece, šest mesecev in devet mesecev. Vsaka dolžina napovedi ima še tri različne učne množice in testno množico. Testna množica vsebuje aplikacije iz maja 2018 in ciljni atribut, izračunan iz meseca, za katerega napovedujemo. Učne množice vsebujejo podatke za: pretekli mesec, tri mesece oziroma šest mesecev z napovedmi za mesece pred in vključno z majem 2018. Izjema je napoved za devet mesecev, za katero ni učne množice s šestimi meseci podatkov.

3.2.1 Uporabljeni atributi

- **Starost**: Atribut je izpeljan iz **release_date**, ki vsebuje datum prihoda aplikacije v trgovino. Diskretni atribut razdeli aplikacije glede na logaritem časa obstoja, prikazanega na sliki 3.3. Atribut lahko zavzame diskretne vrednosti: $0-2$, $2-5$, $5-8$, $8-11$ in $11+$.

- **Starost zadnje posodobitve**: Atribut je izpeljan iz **status_date**, ki vsebuje datum zadnje posodobitve aplikacije. Diskretni atribut razdeli aplikacije glede na logaritem časa od zadnje posodobitve. Atribut lahko zavzame diskretne vrednosti: $0-2$, $2-5$, $5-8$, $8-11$ in $11+$.

- **Povprečna ocena**: Atribut je izpeljan iz **all_rating**, ki vsebuje povprečno oceno aplikacije. Povprečna ocena aplikacije je zaokrožena na celo število in zavzame vrednosti 0, 1, 2, 3, 4 in 5.

- **Število ocen**: Atribut je izpeljan iz **all_rating_count**, ki vsebuje število ocen aplikacije. Diskretni atribut razdeli aplikacije glede na logaritem števila ocen aplikacije. Atribut zavzame vrednosti: $0-2$, $2-5$, $5-8$, $8-12$, $12-16$, $16-21$ in $21+$.

- **Ciljna publika**: Atribut je izpeljan iz **content_rating**, ki vsebuje informacijo o ciljni publiki aplikacije. Atribut razdeli aplikacije glede na ciljno publiko. Možne ciljne publike so prikazane v tabeli 3.2.

- **Plačljivo**: Atribut je izpeljan iz **price**, ki vsebuje ceno aplikacije. Atribut vsebuje vrednost 1, če je aplikacija plačljiva, in vrednost 0, če aplikacija ni plačljiva.

- **Zvrst**: Atribut je izpeljan iz **genre**, ki vsebuje informacijo o zvrsti aplikacije. Atribut razdeli aplikacije glede na zvrsti. Možne zvrsti so prikazane v tabeli 3.3.

- **Lestvica**: Atribut vsebuje ime lestvice najboljših aplikacji, če aplikacija nastopa v njej. Vse možne lestvice so prikazane v tabeli 3.1.

- **Število prenosov**: Atribut je izpeljan iz **downloads**, ki vsebuje informacijo o številu prenosov aplikacije zaokroženo na potenco števila deset oziroma petkratnik potence števila deset. Atribut vsebuje diskretna števila od 0 naprej, ki predstavljajo vse po velikosti razvrščene vrednosti **downloads**.

- **Povečanje števila prenosov aplikacije**: Povečanje atributa **Število prenosov** aplikacije za obdobje: enega meseca, dveh mesecev, treh mesecev, šestih mesecev in devetih mesecev.

apps_topselling_paid	apps_topselling_free
apps_topselling_new_paid	apps_topselling_new_free
apps_topgrossing	apps_movers_shakers
apps_featured	apps_daydream_moreapps
apps_daydream_moregames	

Tabela 3.1: Lestvice aplikacij

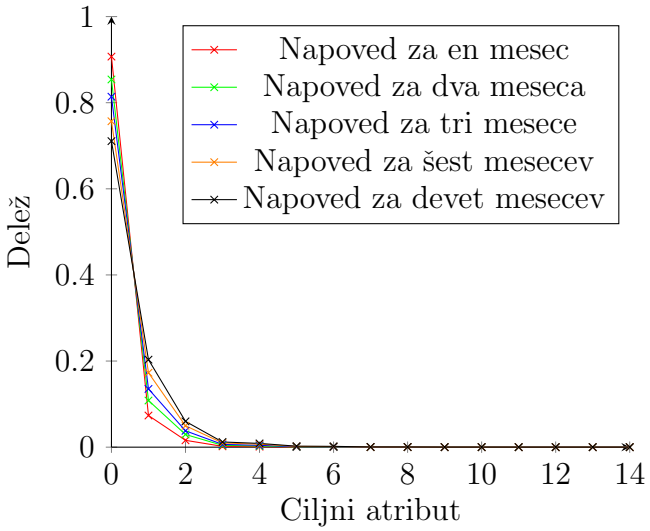
Unrated	Everyone	Everyone 10+
Teen	Mature 17+	Adults only 18+

Tabela 3.2: Ciljne publike aplikacij

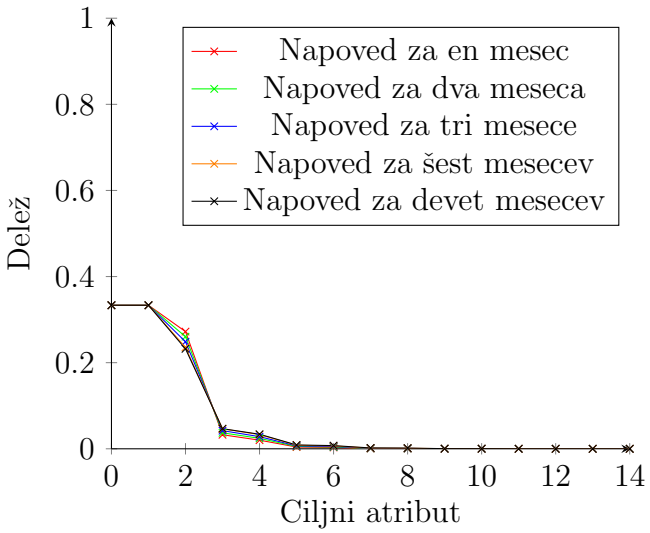
Action	Adventure	Arcade
Art & Design	Auto & Vehicles	Beauty
Board	Books & Reference	Business
Card	Casino	Casual
Comics	Communication	Dating
Education	Educational	Entertainment
Events	Finance	Food & Drink
Health & Fitness	House & Home	Libraries & Demo
Lifestyle	Maps & Navigation	Medical
Music	Music & Audio	News & Magazines
Parenting	Personalization	Photography
Productivity	Puzzle	Racing
Role Playing	Shopping	Simulation
Social	Sports	Strategy
Tools	Travel & Local	Trivia
Video Players & Editors	Weather	Word

Tabela 3.3: Kategorije aplikacij

Primere, kjer ni podatkov, ki jih potrebujemo za izpeljavo atributov, smo zavrgli. Zaradi velikega deleža primerov, kjer ciljni atribut zavzame vrednosti 0 ali 1, kot je prikazano na sliki 3.5, smo se odločili, da bomo uravnotežili učne množice s podvzorčenjem primerov iz prevladujočih razredov. Iz učnih množic smo zato odstranili delež primerov s ciljnima atributoma 0 in 1. Uravnotežene učne množice vsebujejo: tretjino primerov z vrednostjo ciljnega atributa enakega 0, tretjino primerov z vrednostjo ciljnega atributa enakega 1 in tretjino preostalih primerov. Nove porazdelitve razredov so prikazane na grafu slike 3.6. Število vseh primerov pred in po uravnoteževanju je prikazano v tabelah 3.4 ter 3.5.



Slika 3.5: Porazdeljenost ciljnega atributa v učnih množicah iz podatkov treh mesecev.



Slika 3.6: Porazdeljenost ciljnega atributa v uravnoteženih učnih množicah iz podatkov treh mesecev.

en mesec	dva meseca	trije meseci	šest mesecev	devet mesecev
11421454	10980673	10429436	9428063	8397471

Tabela 3.4: Število primerov v učni množici iz podatkov treh mesecev glede na dolžino napovedi

en mesec	dva meseca	trije meseci	šest mesecev	devet mesecev
667203	1242853	3881895	1987515	2167416

Tabela 3.5: Število primerov v uravnoteženi učni množici iz podatkov treh mesecev glede na dolžino napovedi

Poglavje 4

Rezultati

V tem poglavju bomo na podatkih, predstavljenih v poglavju 3, klasificirali uspešnost, ki se kaže v številu novih prenosov aplikacij iz trgovine Google Play.

4.1 Uporaba MOA

Za napovedovanje bomo uporabili že izdelane metode v paketu MOA. Kot je bilo omenjeno v razdelku 2.3, se pri uporabi MOA določi: podatkovni tok, metodo za klasifikacijo in metodo za evalvacijo. Tokove imamo shranjene v datotekah, s podatki, opisanimi v razdelku 3.2. Metode, uporabljene za klasifikacijo, so omenjene v razdelku 2.1. Evalvacijo prožimo z metodo EvaluateModel, kjer se bo najprej zgradil model na podlagi učnih podatkov in bo kasneje testiran na testni množici. Vsak test bo predstavljen s klasifikacijsko točnostjo metode in ocenami F1 za vse razrede. Ciljni atribut ima petnajst različnih razredov od 0 do 14, ki predstavljajo velikost spremembe v povečavi števila prenosov. Razredi so urejeni po velikosti. To pomeni, da razred z višjim številom pomeni večjo povečavo v prenosih. Primerom z razredom 0 se število prenosov ni opazno spremenilo in primerom razreda n se je število prenosov povečalo za približno $10^{\frac{n}{2}}$ krat.

4.2 Rezultati različnih metod za klasifikacijo

V tem razdelku predstavljamo rezultate klasifikacijske točnosti za uporabljene klasifikatorje. Ker imamo pri klasifikaciji opravka z urejenimi razredi (ciljna spremenljivka je ordinalna), bomo v nadaljevanju rezultate vrednotili tudi z merama za ocenjevanje uspešnosti regresije – s srednjo absolutno napako in z relativno srednjo absolutno napako.

4.2.1 Naivni Bayes

Prva testirana metoda je naivni Bayes, opisana v razdelku 2.1.2. Iz tabel 4.1, 4.2 in 4.3 je vidno, da je imela metoda pri testih z učnimi podatki iz največ treh mesecev premalo učnih primerov. Pri napovedih za en mesec in šest mesecev je zato veliko manjša klasifikacijska točnost in višja vrednost srednje absolutne napake. Metoda doseže na testih, kjer so učne množice iz šestih mesecev, boljše rezultate, ki so še zmeraj slabši od rezultatov ostalih metod. Relativne srednje absolutne napake, prikazane v tabeli 4.2, ki so veliko večje od 1, nam povejo, da metoda napoveduje slabše od metode, ki bi vsakemu primeru napovedala povprečni razred. Grafikon na sliki 4.1 prikazuje, da se metoda sicer prilagodi pogostejšim razredom, a še vedno napoveduje tudi razrede, ki predstavljajo velike spremembe števila prenosov.

	En mesec	Dva meseca	Trije meseci	Šest mesecev	Devet mesecev
En mesec	8.359	2.439	3.453	12.076	7.930
Trije meseci	8.231	3.097	1.823	12.059	7.913
Šest mesecev	2.069	1.589	1.675	2.854	/

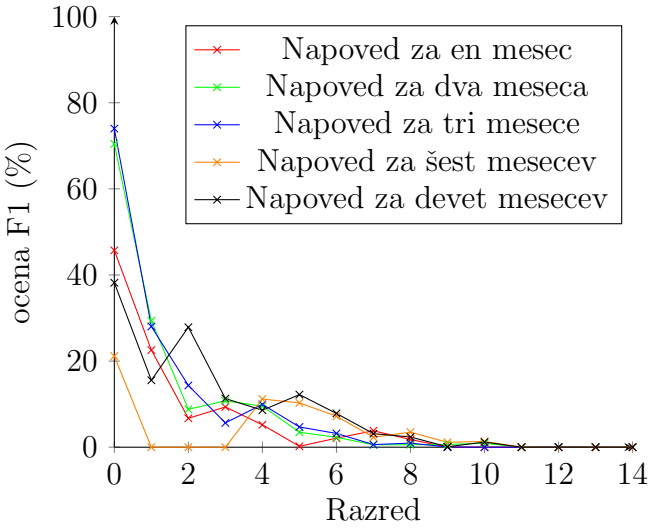
Tabela 4.1: Srednja absolutna napaka metode naivni Bayes (dolžina učnih podatkov \dolžina napovedi)

	En mesec	Dva meseca	Trije meseci	Šest mesecev	Devet mesecev
En mesec	37.335	7.091	7.817	19.936	11.74
Trije meseci	36.765	9.005	4.127	19.908	11.715
Šest mesecev	9.241	4.621	3.792	4.712	/

Tabela 4.2: Relativna srednja absolutna napaka metode naivni Bayes (dolžina učnih podatkov \dolžina napovedi)

	En mesec	Dva meseca	Trije meseci	Šest mesecev	Devet mesecev
En mesec	28.455	55.310	48.101	8.498	21.78
Trije meseci	29.259	52.432	55.162	8.575	21.75
Šest mesecev	60.178	60.440	57.284	46.585	/

Tabela 4.3: Klasifikacijska točnost (%) metode naivni Bayes (dolžina učnih podatkov \dolžina napovedi)



Slika 4.1: Ocene F1 napovedi za metodo navni Bayes iz podatkov zadnjih treh mesecev

4.2.2 Hoeffdingova drevesa

Naslednja testirana metoda so Hoeffdingova drevesa, opisana v razdelku 2.1.3. Iz tabel 4.3 in 4.6 vidimo, da so Hoeffdingova drevesa imela boljšo klasifikacijsko točnost kot naivni Bayes. Razlog za to je v boljšem prilagajanju primerom pogostejših razredov. To se vidi v boljših ocenah F1 pri nižjih razredih na sliki 4.2. Če primerjamo grafikona na slikah 4.1 in 4.2, se vidi, da so krivulje ocen F1 pred razredom 3 višje pri metodi Hoeffdingovih dreves kot pri metodi naivnega Bayesa. Od razreda 3 naprej prevzame prednost naivni Bayes. Višje klasifikacijske točnosti pomenijo tudi manjše srednje absolutne napake, ki so prikazane na tabeli 4.4. V vseh testih se je metoda v povprečju zmotila za manj kot za en razred. Pri napovedih za tri mesece ali več se relativne srednje absolutne napake približajo 1, a je z izjemo enega testa ne presežejo.

	En mesec	Dva meseca	Trije meseci	Šest mesecev	Devet mesecev
En mesec	0.337	0.533	0.517	0.638	0.750
Trije meseci	0.387	0.456	0.426	0.706	0.703
Šest mesecev	0.424	0.499	0.625	0.675	/

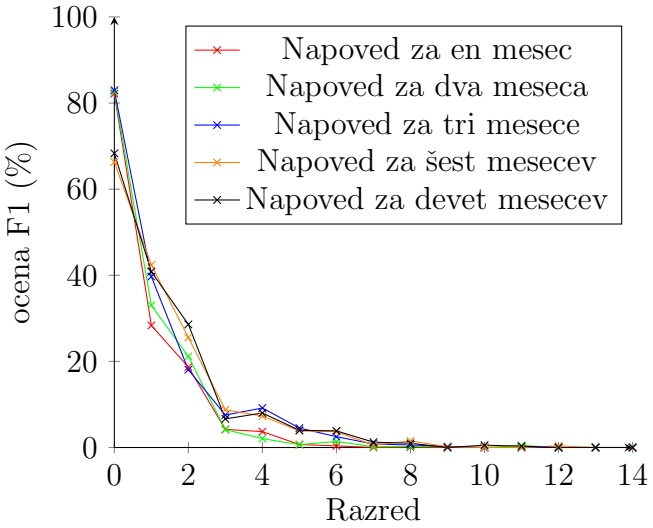
Tabela 4.4: Srednja absolutna napaka metode Hoeffdingovih dreves (dolžina učnih podatkov \dolžina napovedi)

	En mesec	Dva meseca	Trije meseci	Šest mesecev	Devet mesecev
En mesec	1.504	1.551	1.171	1.053	1.110
Trije meseci	1.729	1.327	0.965	1.166	1.041
Šest mesecev	1.893	1.451	1.416	1.114	/

Tabela 4.5: Relativna srednja absolutna napaka metode Hoeffdingovih dreves (dolžina učnih podatkov \dolžina napovedi)

	En mesec	Dva meseca	Trije meseci	Šest mesecev	Devet mesecev
En mesec	74.798	64.893	66.691	58.465	58.161
Trije meseci	69.857	69.238	69.852	52.512	52.803
Šest mesecev	68.304	66.841	57.323	55.281	/

Tabela 4.6: Klasifikacijska točnost (%) metode Hoeffdingovih dreves (dolžina učnih podatkov \dolžina napovedi)



Slika 4.2: Ocene F1 napovedi za metodo Hoeffdingova drevesa iz podatkov zadnjih treh mesecev

4.2.3 Drevesa IADEM

Zadnja testirana metoda so drevesa IADEM, opisana v razdelku 2.1.4, s statistiko v listih, sestavljeno iz utežene vsote naivnega Bayesa in večinskega razreda. Na sliki 4.3 je prikazano, da so se drevesa IADEM izmed vseh metod najbolj prilagodila primerom s pogostejšimi razredi in da metoda ne napoveduje razredov, višjih od 2. To je razlog za najvišjo klasifikacijsko točnost, prikazano na tabeli 4.9, izmed vseh metod. Najvišja klasifikacijska točnost je razlog za najnižje srednje absolutne napake, prikazanih v tabeli

4.7. Metoda dreves IADEM je edina metoda, ki relativno srednjo absolutno napako pod 1 (glej tabelo 4.8) pri vseh časovnih dolžinah napovedi.

	En mesec	Dva meseca	Trije meseci	Šest mesecev	Devet mesecev
En mesec	0.124	0.203	0.277	0.434	0.524
Trije meseci	0.124	0.262	0.269	0.495	0.526
Šest mesecev	0.271	0.203	0.279	0.401	/

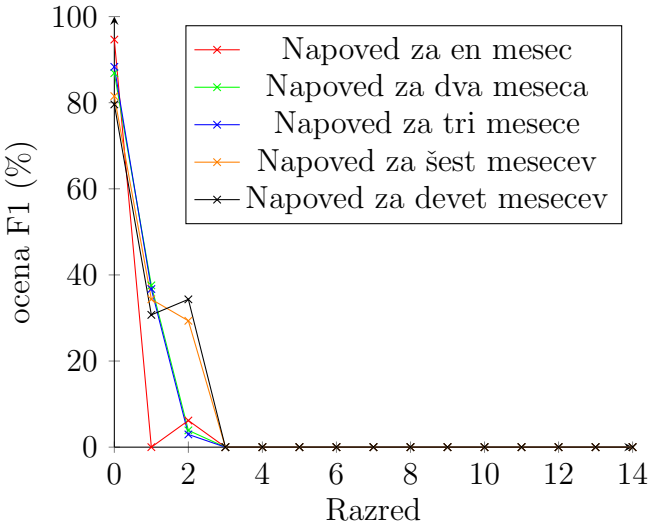
Tabela 4.7: Srednja absolutna napaka metode dreves IADEM (dolžina učnih podatkov \dolžina napovedi)

	En mesec	Dva meseca	Trije meseci	Šest mesecev	Devet mesecev
En mesec	0.555	0.590	0.627	0.716	0.776
Trije meseci	0.555	0.760	0.610	0.818	0.779
Šest mesecev	1.212	0.591	0.633	0.661	/

Tabela 4.8: Relativna srednja absolutna napaka metode dreves IADEM (dolžina učnih podatkov \dolžina napovedi)

	En mesec	Dva meseca	Trije meseci	Šest mesecev	Devet mesecev
En mesec	89.796	84.241	79.280	69.845	64.486
Trije meseci	89.796	77.376	79.666	67.014	64.437
Šest mesecev	76.814	84.031	79.532	72.331	/

Tabela 4.9: Klasifikacijska točnost (%) metode dreves IADEM (dolžina učnih podatkov \dolžina napovedi)



Slika 4.3: Ocene F1 napovedi za metodo dreves IADEM

4.3 Analiza kakovosti uporabljenih atributov

Za analizo kakovosti uporabljenih atributov smo uporabili Gini indeks kot mero nečistoče. Testirali smo učne množice za napoved dolžine: enega meseca, treh mesecev ter devetih mesecev iz podatkov preteklih treh mesecev (tabela 4.10). Atribut **Starost** je v vseh primerih dosegel najvišji Gini indeks. To si lahko poskusimo razložiti s tem, da so aplikacije, ki so preživele veliko časa v trgovini so verjetno že dosegle svojo rast v številu prenosov. Če je niso, jo z veliko verjetnostjo tudi ne bodo. Atribut z drugim najvišjim Gini indeksom je **Starost zadnje posodobitve**, ki nam pove, koliko časa je minilo od zadnje posodobitve aplikacije. Uspeh atributa lahko razlagamo tako, da pogoste spremembe v aplikaciji verjetno pritegnejo pozornost novih uporabnikov in se zato poveča prenos aplikacije. Zadnji atribut z višjim Gini indeksom je **Število prenosov**. Aplikacije, ki nimajo veliko prenosov, bodo z veliko verjetnostjo ostale neopažene. Aplikacije, ki imajo veliko število prenosov, so že dosegle svoj vrh in bodo imele relativno na njihovo trenutno stanje le majhne spremembe števila prenosov. **Povprečna ocena** in **Število ocen** imata Gini indeks manjši kot prej omenjeni trije atributi a

še vedno večjega od ostalih atributov. Atributa predstavljata mnenje uporabnikov. Atributi **Ciljna publika**, **Plačljivo** in **Zvrst** hranijo informacije o tipu aplikacije. Njihov nizek Gini indeks nam pove, da vrsta aplikacije nima velikega vpliva na njeno uspešnost. Najnižji Gini indeks je imel atribut **lestvica**. Atribut ima nizek Gini indeks, ker večina aplikacij ne nastopa v lestvicah in ima atribut zato enako vrednost.

Atributi	En mesec	Trije meseci	Devet mesesev
Starost	0.1478	0.0989	0.1016
Starost zadnje posodobitve	0.1112	0.0713	0.0711
Povprečna ocena	0.0454	0.0241	0.0238
Število ocen	0.0393	0.0215	0.0202
Ciljna publika	0.0120	0.0160	0.0198
Plačljivo	0.0087	0.0031	0.0025
Lestvica	0.0082	0.0023	0.0023
Zvrst	0.0095	0.0086	0.0066
Število prenosov	0.1127	0.0659	0.0688

Tabela 4.10: Gini indeks atributov v učni množici za napoved dolžine enega meseca, treh mesecev in devetih mesecev iz podatkov treh mesecev.

Poglavje 5

Zaključek

V diplomskem delu je bilo predstavljeno napovedovanje novih prenosov aplikacij iz spletne trgovine Google Play. Podatki so bili pridobljeni iz arhiva, ki ga hrani podjetje AppMonsta. Uporabni podatki so bili nato izbrani in pretvorjeni v atribute, ki so bili uporabljeni za napovedovanje. Pridobljene podatke smo nato razdelili v učno in testno množico. Učna množica ni vsebovala podatkov o aplikacijah za čas napovedovanja. Zaradi velikega deleža primerov, ki so imeli enako vrednost ciljnega atributa, smo uporabili uravnotežene učne množice. Za napoved so se uporabile metode za klasifikacijo podatkovnih tokov, implementirane v orodju MOA (Massive Online Analysis). Uporabljene so bile metode: naivni Bayes, Hoeffdingova drevesa ter drevesa IADEM. Metode smo nato ocenili s: klasifikacijsko točnostjo in oceno F1 za vsak razred, ter srednjo vrednostjo absolutne napake in relativno srednjo vrednostjo absolutne napake.

Metoda naivnega Bayesa je imela najnižjo klasifikacijsko točnost in najvišjo srednjo vrednost absolutnih napak vendar se je najbolje prilagodila primerom, manj pogostih razredov. Metoda dreves IADEM se je najbolje prilagodila primerom s pogosto vrednostjo ciljnega atributa in je zato imela najvišjo klasifikacijsko točnost in nizko srednjo vrednosti absolutnih napak. Razlog za dobre ocene evalvacij je napovedovanje le za prve tri najpogostejše razrede. Metoda Hoeffdingovih dreves je po ocenah evalvacij in prilagajanju primerom

manj pogostih razredov dosegla srednjo raven treh testiranih metod.

Napoved bi se lahko izboljšala z uporabo dodatnih podatkov. Primer takega podatka je količina denarja, porabljenega za oglaševanje aplikacije. Dobra aplikacija, pri kateri se razvijalec trudi pri posodabljanju, a je nihče ne opazi, ni uspešna. Še ena možnost izboljšave bi bila, da bi o vsaki aplikaciji hranili zgodovino sprememb. S tem bi imeli podatke o pretekli rasti aplikacije in bi lahko bolj natančno napovedovali njeno prihodnost. Napovedi metod bi lahko predstavili z matriko, ki vsebuje število primerov za vsak par pravilne in napovedane vrednosti ciljnega atributa.

Literatura

[1] Slika MOA uporabniškega vmesnika. Dosegljivo: `https://moa.cms.waikato.ac.nz/`. [Dostopano 5. 1. 2019].

[2] Brian Babcock, Mayur Datar, Rajeev Motwani, et al. Load shedding techniques for data stream systems. In *Proc. Workshop on Management and Processing of Data Streams*. Citeseer, 2003.

[3] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *Journal of Machine Learning Research*, 11(May):1601–1604, 2010.

[4] José del Campo-Avila, Gonzalo Ramos-Jiménez, João Gama, and Rafael Morales-Bueno. Improving the performance of an incremental algorithm driven by error margins. *Intelligent Data Analysis*, 12(3):305–318, 2008.

[5] Joao Gama. *Knowledge discovery from data streams*. Chapman and Hall/CRC, 2010.

[6] Randy Nelson. Global App Revenue Reached 34 Billion in the First Half of 2018, Up 28% Year-Over-Year . Dosegljivo: `https://sensortower.com/blog/app-revenue-and-downloads-1h-2018`, 2018. [Dostopano 5. 1. 2019].

[7] Hai-Long Nguyen, Yew-Kwong Woon, and Wee Keong Ng. A survey on data stream clustering and classification. *Knowledge and Information Systems*, 45, 12 2014.

[8] Shane Schick. Vision Mobile: 'App poverty line' represents 60. Dosegljivo: https://www.fiercewireless.com/developer/vision-mobile-app-poverty-line-represents-60-all-developers, 2015. [Dostopano 5. 1. 2019].

[9] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.