VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

Magistrinis darbas

Biudžeto optimizavimas internetinės paieškos aukcionuose

Budget optimization in internet search auctions

Laurynas Radzevičius

VILNIUS 2017

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

STATISTIKOS KATEDRA

Darbo vadovas dr. Vytautas Kazakevičius _____

Darbo recenzentas _____

Darbas apgintas _____

Darbas įvertintas _____

Registravimo NR. _____

Turinys

1 SANTRAUKA

Biudžeto optimizavimas internetinės paieškos aukcionuose

Reklama internetinės paieškos aukcionuose yra milžiniška industrija. Vien Kinijoje šiai reklamai 2011 metais buvo išleista beveik 19 milijardų dolerių. Reklamos pirkimas vyksta aukciono principu, todėl reklamos užsakovai susiduria problema - kokius statymus reikia parinkti kokretiems raktažodžiams, kad gautas pelnas būtų maksimalus.

Darbo tikslas. Sukurti algoritmą, kuris įvertintų esamą situaciją konkretaus raktažodžio aukcione ir sumodeliuotų optimalų statymą kiekvienam iš jų.

Literatūros apžvalga. Egzistuojančią literatūrą, analizuojančią internetinės paieškos aukcionus, galima suskirstyti į 3 rūšis pagal jų naudojamus metodus - naudojantys MDP's (*Markov Decision Process*), naudojantys stochastinius modelius ir naudojantys generalizuota sekretorės problemos versiją. Visi modeliai turi savų pranašumų, tačiau taip pat jie iš dalies neatitinka realios egzistuojančios situacijos, su kuria susiduria reklamos pirkėjai. Šiame darbe biudžeto optimizavimo problema sprendžiama naudojant tik tuos duomenis, kurie yra laisvai prieinami kiekvienam reklamos užsakovui.

Naudojami duomenys. Darbe buvo naudojami Google Search reklamos duomenys, rinkti nuo 2016.06.01 iki 2016.12.10. Buvo modeliuojamas 31 skirtingas raktažodis.

Rezultatai ir išvados. Naudojant turimus duomenis ir *machine learning* algoritmus buvo įvertinti esminiai kintamieji, darantys įtaką galutiniam pelnui - reklamos pozicija, reklamos kiekis, paspaudimų funkcija, vidutinis pardavimų dažnis. Sukurto modelio rezultatai indikuoja, jog analizuojamoje situacijoje buvo naudojami itin netinkami statymai - vidutinis dienos pelnas yra neigiamas. Naudojant statymų optimizavimo modelį buvo sugeneruoti du galimi scenarijai - kai statymas dienos eigoje nekinta ir kai statymas gali kisti kas valandą. Abiejų modelių pelnas yra teigiamas, ženkliai vienas nuo kito nesiskiriantis.

Raktiniai žodžiai: Internetiniai aukcionai, biudžeto optimizavimas, internetinė reklama.

2 ABSTRACT

Budget optimization in internet search auctions

Internet search advertising is a huge worldwide industry. In 2011 about 19 billion dollars were spent on it in China alone. The ad buying process is auction based therefore the main problem every advertiser encounters is how to optimize bids for certain keywords as to maximise the expected profit.

Research aim. The main goal is to create an algorithm/model which could evaluate the market situation on every bid in consideration and predict the ones which should maximise expected profit.

Literature overview. Even though the scope of the articles, analysing budget optimization problem, is quite wide, it could be categorized into three main branches based on the models, used in them. Those three branches can be called MDP's (Markov Decision Process), stochastic models and knapsack secretary. All of them have some benefits, compared to others, however at the same time they all lack correspondence towards a real life situation, which is experienced by advertisers everyday. Here the budget optimization problem is being solved by only using the data, available to an ordinary advertiser.

Data. Data from Google Search auctions was used in model creation. Data was collected from 2016.06.01 to 2016.12.10. Data consists of statistics of 31 different keywords.

Results and conclusions. After applying machine learning algorithms, a few elements were distinguished as important to a final profit estimation. Those are average ad position, number of impressions, click-through-rate and average conversion rate. A predicting models were constructed to evaluate each variable. Results of the model showed that the bids, used in reality, were extremely off - the average daily profit was negative. The model created two possible bid scenarios, where optimal bids were calculated on hourly and daily basis. The later more closely resembles a real life situation. Each of those scenarios predicted a positive daily profit with marginal differences between them.

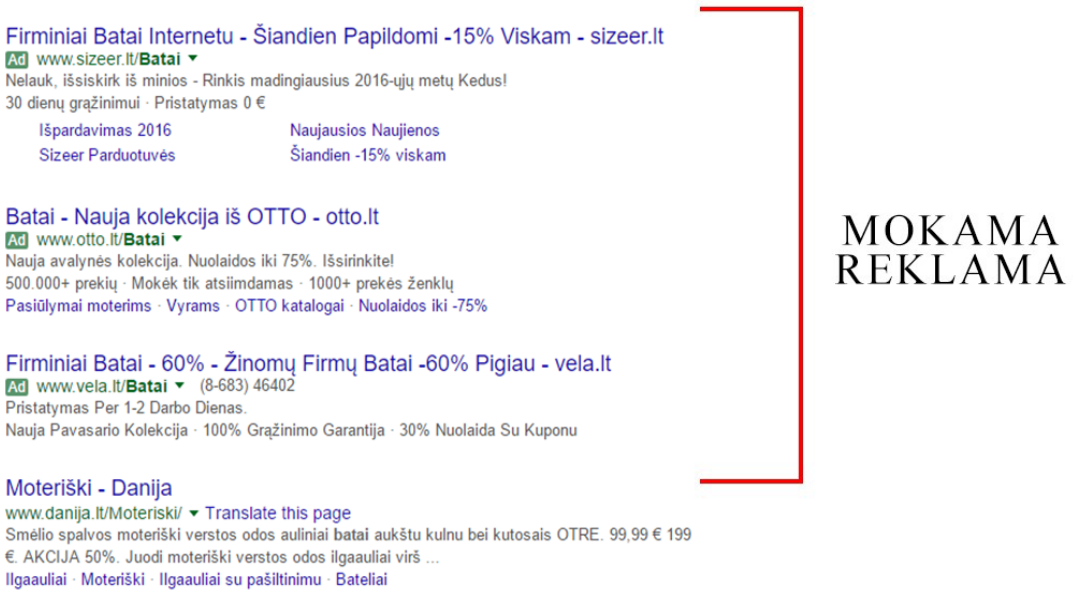
Keywords: Internet auctions, budget optimization, internet advertising.

3 ĮVADAS

Paieškos varikliai (angl. *search engines*) yra sukurti padėti interneto vartotojams ieškoti reikalingos informacijos internete. 2013 metais atliktas tyrimas parodė, jog 92% interneto vartotojų naudoja bent vieną paieškos variklį (Yang, Y. and Wang, F., 2013, p. 2) – Google, Yahoo, Bing ir t.t. Žinomiausias ir labiausiai naudojamas iš jų yra Google Search – jo pavyzdys ir bus naudojamas šiame darbe.

Didžioji dauguma internetinių puslapių, pradedant naujienų portalais ir baigiant tokiais interneto gigantais kaip Facebook ar Google, pardavinėja reklamą savo puslapiuose. Paieškos varikliai nėra išimtis. Internetinė reklama, kad ir kaip ji kartais erzintų vartotoją, yra būtina tiek patiems portalams, tiek ir tam pačiam vartotojui. Dažniausiai naudojimasis internetiniais portalais vartotojui yra nemokamas. Vis dėlto, akivaizdu, kad internetinio tinklapio palaikymas turi savo kainą – tai gali būti technikos atnaujinimas, svetainės tobulinimas ir optimizavimas, naujienų portalų atveju – žurnalistų darbas ir t.t. Todėl reklama daugeliui internetinių puslapių yra pagrindinis ir vienintelis pajamų šaltinis, be kurio svetainės apskritai negalėtų gyvuoti.

Paieškos varikliuose reklamai dedikuota vieta yra puslapio viršuje. Šiuo metu populiariausias paieškos variklis Google viename puslapyje rodo iki 5 skirtingų reklamų.



1 pav. Google paieškos variklio reklamos pozicijos

Reklamos pozicija search tipo reklamose yra itin svarbi – vartotojai yra linkę spausti

ant pirmųjų 10 paieškos rezultatų, o 99% vartotojų apskritai peržiūri ne daugiau kaip 30 paieškos rezultatų (Yang, Y. and Wang, F., 2013, p. 2).

Reklama internetinės paieškos aukcionuose yra milžiniška industrija. Vien Kinijoje šiai reklamai 2011 metais buvo išleista beveik 19 milijardų dolerių (Yang, Y. and Wang, F., 2013, p. 2). Reklamos pirkimas, plačiau aptariamas tolesniame skyriuje, vyksta aukciono principu, todėl reklamos užsakovai susiduria su itin svarbia kaštų paskirstymo problema. Biudžeto optimizavimas panaudojant statistinius modelius ir yra pagrindinis šio darbo tikslas.

4 INTERNETINIŲ AUKCIONŲ VEIKIMO PRINCIPAS

Google *Search* reklamos kampaniją sudaro 3 pagrindiniai elementai (Fieldman, et. al., 2007):

- **Raktažodžiai (angl. *keywords*)**. Reklaminės kampanijos užsakovas pasirenka raktažodžius, kurie yra susiję su jo verslu. Didelėse reklamos kampanijose tokių raktažodžių gali būti keli šimtai. Raktažodžiai gali būti itin konkretūs, pvz. *raudoni moteriški bateliai*, arba labai platūs, pvz. *batai*. Jeigu vartotojo paieškos užklausa atitinka nurodytus raktažodžius, kampanijos užsakovas dalyvauja reklamos aukcione.

- **Biudžetas**. Reklamos kampanijose turimas biudžetas yra ribotas išteklius. Biudžeto apribojimai gali būti dviejų rūšių – lankstūs ir nelankstūs (Yang, Y. and Wang, F., 2013). Lankstus biudžetas apibūdina situaciją, kai galima išlesti tam tikru kiekiu daugiau, nei yra numatyta – pvz. kampanijos išleista pinigų suma negali viršyti pradinio biudžeto daugiau nei 20%. Nelakstus biudžetas, priešingai, apibūdina situaciją, kai išleista suma negali viršyti pradinio numatyto biudžeto. Praktikoje dažniausiai yra naudojamas nelankstus biudžetas.

- **Statymas (angl. *bid*)**. Google *Search* reklamos kampanijose reikia nurodyti savo siūlomą paspaudimo kainą (angl. CPC – *cost per click*). Kaip ir paprastuose aukcionuose, didesnę paspaudimo kainą pasiūlęs reklamos užsakovas laimi aukcioną ir jo reklama yra parodoma vartotojui. Skirtingai nei įprastuose aukcionuose, vartotoją gali *„laimėti"* iki 5 reklamos užsakovų, priklausomai nuo aukcione dalyvaujančių žaidėjų skaičiaus. Pvz. jei su tam tikru raktažodžiu reklamuotis nori tik vienas užsakovas, tik jo reklama ir bus rodoma vartotojams, atlikusiems paiešką su tuo raktažodžiu, pirmoje pozicijoje. Kita vertus, jei prie tam tikro raktažodžio nori reklamuotis 5 skirtingi reklamos užsakovai, jie yra išrikiuojami pagal jų siūlomą CPC kainą – didžiausią kainą pasiūlęs užsakovas gauna pirmąją reklamos poziciją, žemiausią – penktąją. Tačiau visų 5 aukcione dalyvavusių žaidėjų reklama yra parodoma vartotojui. Jei aukcione dalyvauja daugiau nei 5 žaidėjai, tik penkių aukščiausią kainą pasiūliusių reklamos užsakovų skelbimai yra parodomi vartotojui.

Google *Search* aukcione yra mokama tik už įvykusius paspaudimus. Sakykime, kad mūsų įsivaizduojamas reklamos užsakovas pasiūlė aukščiausią kainą aukcione ir jo reklama yra parodoma vartotojui. Reklamdavys už ją turės sumokėti tada ir tik tada, jei vartotojas ant jos paspaus. Todėl į biudžeto optimizavimo problemą įtraukiamas dar vienas kintamasis – paspaudimų dažnis CTR (angl. *click through rate*). Žinoma, ne visi paspaudimai atneša reklamdaviui naudos. Paspaudę ant reklamos vartotojai patenka į reklamos užsakovo tinklapį kuriame, tikėtina, yra prekės, kurias vartotojas gali nusipirkti. Akivaizdu, kad ne visi ant reklamos paspaudę žmonės kažką nusiperka. Į tai svarbu atsižvelgti siekiant tiksliai optimizuoti turimą biudžetą, norint sugeneruoti kiek įmanoma didesnį pelną.

Biudžeto optimizavimo problemai ypač pastaruoju metu yra skiriama nemažai dėmesio moksliniuose žurnaluose. Egzistuojančios biudžeto optimizavimo strategijos ir naudojami algoritmai yra nagrinėjami tolesniame skyriuje.

5 LITERATŪROS APŽVALGA

Egzistuojanti literatūra, nagrinėjanti internetinių aukcionų biudžeto optimizavimo problemą, gali būti suskirstyta į 3 pagrindines rūšis pagal taikomus modelius – naudojami MDP (*Markov Decision Process*), stochastiniai modeliai ir galiausiai biudžeto optimizavimas analizuojamas kaip generalizuota sekretorės problemos versija, literatūroje įvardijamos kaip *knapsack secretary* ar *online multiple-choice knapsack* problemomis.

5.1 MDP's biudžeto optimizacijoje

MDP (*Markov Decision Process*) panaudojimas biudžetų optimizacijai paieškos variklių aukcionuose yra išsamiai išnagrinėtas Gummadi, R., Key, P.B.. Proutiere, A. (2011) ir Amin, K., Kearns, M., Key, P. and Schwaighofer (2012) darbuose. Gummadi su kolegomis analizuoja situaciją, kai skirtingi žaidėjai dalyvauja daug aukcionų, kiekvieno žaidėjo aukščiausi statymai yra nepriklausomi ir vienodai pasiskirstę. Tikslas yra maksimizuoti pelno funkciją išreikštą kaip

$$\sum_{i=0}^{\infty} e^{-\beta i} E[1_{u(i)>w(i)}(v(i) - w(i))]$$

$u(i)$ yra žaidėjo siūloma kaina laiko momentu i, $w(i)$ yra konkurentų siūloma kaina laiko momentu i, o v yra prekės vertė. Stengiamasi maksimizuoti gaunamą naudą atsižvelgiant į biudžeto apribojimą b. Kiekvieno periodo pradžioje biudžetas yra apribotas $b(i+1) = b(i) + a + 1_{u(i)>w(i)}w(i)$, kur a yra biudžeto padidinimas kiekvieno laikotarpio pradžioje, o $b(0) = b$. Žaidėjas negali viršyti savo turimo biudžeto $b(i)$ laikotarpiu i.

Nagrinėdami Markovo sprendimų procesą, apibrėžtą kaip

$$V_\beta(b) = sup_u(E[\sum_{i=0}^{\infty} e^{-\beta i} 1_{u(i)>w(i)} w(i)])$$

autoriai priena išvadą, jog geriausia statymo strategija apibrėžiama funkcija $\frac{1}{1+V'(B)}$.

Kareem A. ir kolegos tuo tarpu analizuoja labiau realybę atitinkančią situaciją. Jų situacijos apibrėžime žaidėjai stengiasi maksimizuoti gaunamus paspaudimus (ang. *clicks*), neatsižvelgiant į jų tolimesnę vertę, kurią prognozuoti yra itin sudėtinga. Taip pat atsi-

žvelgiama į tai, kad analizuojami duomenys yra cenzūruoti iš dešinės - tik tada, jei žaidėjas laimi paspaudimą, mes sužinome *tikrąją* rinkos kaina. Kitu atveju žinoma tik tiek, kad statymas buvo per mažas. Šiuo atveju MDP būsenos nurodo likusį laiką ir biudžetą.

Nagrinėjama problema primena realiai reklamos industrijoje egzistuojančias situacijas - kiekvieno naujo periodo pradžioje turimas biudžetas yra fikstuoas ir lygus B. Periodo metu žaidėjas dalyvauja daugybėje aukcionų. Tikslas yra sukonstruoti algoritmą, kuris padėtų maksimizuoti per laikotarpį u gaunamų paspaudimų skaičių, neviršijant biudžeto B. Svarbu yra tai, kad autoriai nagrinėja situaciją, kai parodymo (angl. *impression*) nupirkimas automatiškai garantuoja paspaudimą. Realybėje tik nedidelė parodymų dalis virsta paspaudimais. Taigi, nors šioje vietoje autoriai priima realybės neatitinkančią prielaidą, jų pasiūlytas algoritmas turėtų būti pritaikomas ir realiai situacijai jei tik paspaudimų dažnio (ang. *CTR - click through rate*) įvertinys būtų tinkamas. Galiausiai paskutinė autorių daroma prielaida yra ta, jog žaidėjas dalyvauja aukcionuose susijusiuose tik su vienu raktažodžiu. Realybėje naudojama keliasdešimt ar net keli šimtai raktažodžių. Dėl šios priežasties autorių pasiūlytas algoritmas taip pat minimaliai neatitinka realybės.

Autoriai savo darbe išvysto *Greedy Product-Limit* algoritmą, trumpai aprašomą taip:

```
Input: Budget B
Initialize distribution p uniform on [B]
Initialize K = []; Initialize O = []
for period u = 1,2,... do
    Set B_u,T := B
    for auctions remaining t=T,T-1,...,1 do
        Bid \pi_p(B_u,t, t)
        Set k_u,t \pi_p(B_u,t, t) + 1
        K [K; k_u,t]
        if Click won at price x_u,t then
            O = [O; x_u,t]
        else
            O = [O; k_u,t]
        end if
        Update p to PL(K, O)
    end for
end for
```

Vis dėlto, autoriai patys teigia, jog algoritmas yra empiriškai efektyvus, tačiau kol kas nėra griežtai įrodę, kodėl.

5.2 Stochastiniai modeliai biudžeto optimizacijoje

Stochastinių modelių panaudojimas biudžeto optimizacijai yra aptariamas Muthukrishnan, S., Pál, M. and Svitkina, Z. (2010) ir DasGupta, B. and Muthukrishnan, S. (2013) darbuose.

DasGupta, B. ir Muthukrishnan atkreipia dėmesį į svarbų paieškos reklamos aspektą - aukcionų, kuriuose dalyvauja konkretus žaidėjas, skaičius iš esmės yra stochastinis procesas. Turėdami tai omenyje autoriai suformuluoja keletą SSBO (ang. *Scenario Stochastic Budget Optimization*) algoritmų, vienas iš kurių čia bus aptariamas detaliau.

Sukurtas modelis remiasi prielaidomis:

- Yra tik viena reklamai dedikuota vieta. T.y. kitaip nei realioje Google paieškoje, kurioje yra 5 vietos, čia nagrinėjama situacija tik su viena vieta.
- Turime n dominančių raktažodžių $K_1, K_2, ..., K_n$. Raktažodžio K_j paspaudimo kainą pažymėkime d_j.
- Žaidėjo turimas biudžetas žymimas B.
- Yra m skirtingų scenarijų. Scenarijų šiuo atveju galima suprasti kaip modelio įvertinius skirtingais laiko periodais. Scenarijus aprašo šie parametrai:
  - ϵ_i tikimybė, kur $\sum_{i=1}^{m} \epsilon_i = 1$
  - Paspaudimų vektorius $(a_{i,1}, a_{i,2}, ..., a_{i,n})$. Reikšmė $a_{i,j}$ parodo raktažodžio j paspaudimų skaičių i-tąjame scenarijuje.

Algoritmo tikslas yra maksimizuoti visų scenarijų vidutinį pelną, kai scenarijaus pelnas išreiškiamas funkcija

$$E[payoff_i] = \begin{cases} \epsilon_i \sum_{j=1}^{n} a_{i,j} x_j, & \text{jeigu } \sum_{j=1}^{n} a_{i,j} d_j x_j \leqslant B \\ \frac{B}{\sum_{j=1}^{n} a_{i,j} d_j x_j} (\epsilon_i \sum_{j=1}^{n} a_{i,j} x_j), & \text{kitais atvejais} \end{cases}$$

Dėl to, jog ši nagrinėjama problema yra *NP-hard* (kaip ir kuprinės užpildymo (angl. *knapsack*) uždaviniai yra *NP-complete*) tikslus algoritmas yra nepateikimas.

5.3 "Knapsack secretary" problema

Šiame skyriuje bus apžvelgiama paskutinė literatūros kryptis - modeliai, besiremiantys sekretorės problemos generalizacija. Bus apžvelgiama klasikinė sekretorės problema, jos generalizacijos ir pritaikymas internetinės paieškos aukcionuose.

Sekretorės problema pirmą kartą buvo paminėta Martin'o Gardner'io 1960-aisias metais (Ferguson, T.S., 1989). Sutrumpintai ši problema skamba taip:

- Yra viena laisva sekretorės pozicija.
- Yra žinomas skaičius n kandidačių.
- Su kandidatėmis atliekami interviu atsitiktine tvarka. Kiekviena atsitiktinė eilės tvarka yra vienodai galima.
- Visos kandidatės yra išranguojamos eilės tvarka be lygiųjų. Kandidatės pasirinkimas turi priklausyti tik nuo jos rango lyginant su prieš tai buvusiomis kandidatėmis.
- Nepriimta kandidatė negali grįžti antrą kartą.
- Tikslas yra pasirinkti geriausią kandidatę.

Problema išsprendžiama atmetant pirmuosius $r-1$ elementus ir priimant pirmąjį elementą, kuris yra geresnis už prieš tai buvusius r-1 (Freeman, P.R., 1983). Tokiu atveju tikimybė pasirinkti geriausią įmanomą elementą kai $r > 1$ yra:

$$\mathsf{P}(r) = \sum_{i=1}^{r-1} 0 + \sum_{j=r}^{n} \frac{r-1}{j-1} \cdot \frac{1}{n} = \frac{r-1}{n} \sum_{j=r}^{n} \frac{1}{j-1}$$

Tegul $n \to \infty$, o $x = \lim_{n\to\infty} \frac{r}{n}$. Tokiu atveju:

$$\mathsf{P}(r) = \frac{r-1}{n} \sum_{j=r}^{n} \frac{n}{j-1} \cdot \frac{1}{n} \to x \cdot \int_x^1 \frac{1}{t} \mathrm{d}t$$

Taigi, kai $n \to \infty$ funkciją $\mathsf{P}(r)$ galima aproksimuoti funkcija

$$x \cdot \int_x^1 \frac{1}{t} \mathrm{d}t = x \cdot (ln(1) - ln(x)) = x \cdot ln(\frac{1}{x}) = -x \cdot ln(x)$$

Pirmoji funckijos išvestinė yra $(-x \cdot ln(x))' = -ln(x) - 1$. Prilyginę pirmąją išvestinę 0 gauname, jog $-ln(x) - 1 = 0$, kai $x = e^{-1}$. Antroji funkcijos išvestinė yra $(-ln(x) - 1)' = -\frac{1}{x}$. Taške $x = e^{-1}$ antrosios išvestinės reikšmė yra neigiama. Todėl taškas $x = e^{-1}$ yra funkcijos maksimumas. $x = e^{-1} \approx 0,3679$

Kadangi $x = \lim_{n\to\infty} \frac{r}{n}$ galima teigti, kad esant dideliam kandidatų skaičiui n, optimaliausia strategija, turinti didžiausią tikimybę, jog bus pasirinktas geriausias įmanomas kandidatas, yra nesamdyti pirmųjų 37% kandidatų ir pasirikti pirmąjį geresnį už prieš tai buvusius kandidatus. Tokiu atveju tikimybė pasirinkti geriausią įmanoma kandidatą taip pat yra apie 37%.

Klasikinė sekretorės problema sukėlė straipsnių, parašytų panašia tematika, antplūdį. Bandyta sekretorės problemą perkelti į kitus kontekstus, ją generalizuoti. Svarbi sekretorės problemos generalizacija pateikta Robert Kleinberg (2005) straipsnyje. Kleinberg nagrinėja logiškai išplaukiančią klasikinės sekretorės problemos generalizaciją, vadinamą *„Multiple choice secretary problem"* – kelių pasirinkimų sekretorės problema. Sutrumpintai ji skamba taip (Kleinberg, R., 2005):

- Pirmasis žaidėjas užrašo skaičių rinkinį S, sudarytą iš n teigiamų realių skaičių. Rinkinio S elementai antrajam žaidėjui rodomi atsitiktine tvarka vienas po kito.
- Po kiekvieno elemento atvertimo antrasis žaidėjas turi nuspręsti, ar jį pasirinkti, ar praleisti. Kaip ir klasikinėje sekretorės problemoje, grįžti atgal negalima, t.y. vieną kartą atmetus elementą, jo vėliau pasirinkti jau nebus galima.
- Žaidimo tikslas – pasirinkti k elementų taip, kad jų suma būtų didžiausia.

Kai $k = 1$, tai tampa klasikine sekretorės problema. Egzistuoja nemažai darbų panašia tematika, tačiau daugelis jų, pvz. Kleywegt, A., Papastavrou, J. (1998) daro prielaidas apie apriori žinomą elementų skirstinį. Minėtu atveju – elementai laike yra pasiskirstę pagal Puasono skirstinį. Kleinbergo pateiktoje sekretorės problemos generalizacijoje elementai pateikiami atsitiktine tvarka, nedaromos prielaidos apie jų skirstinį.

Taigi, uždavinys yra sukonstruoti algoritmą, kuris padėtų maksimizuoti pasirinktų k elementų sumą, t.y. $\sum_{i=1}^{k} x_i$

Kleinbergo pasiūlytas algoritmas yra toks:

- Jei $k = 1$, tai naudojamas klasikinės problemos algoritmas.
- Jei $k > 1$, tegul m yra pasiskirstęs pagal binominį dėsnį $B(n, \frac{1}{2})$. Rekursyviai pritaikome algoritmą pasirinkti iki $l = [k/2]$ elementų iš pirmųjų m elementų. Po $m - tojo$ elemento pasirenkame kiekvieną elementą, didesnį už y_l.

Turime atsitiktine tvarka išrikiuotų n teigiamų realių skaičių rinkinį S. Pažymėkime $T \subseteq S$ rinkinį k didžiausių S elementų. Modifikuojame visų x_i vertes pagal tokią logiką:

$$x_i = \begin{cases} 0, & \text{jeigu } x_i \notin T \\ x_i, & \text{jeigu } x_i \in T \end{cases}$$

Taip modifikavus turimą elementų rinkinį, didžiausių k elementų suma išlieka tokia pati. Šią sumą pažymėkime v.

Tarkime, kad m yra pasiskirstęs pagal binominį dėsnį $B(n, \frac{1}{2})$. Pažymėkime Y pirmųjų m elementų rinkinį $y_1 > y_2 > ... > y_m$. Y elementų skaičius yra atsitiktinis dydis iš aibės S elementų. Kadangi $T \subseteq S$ ir rinkinio T elementų skaičius yra k, vadinasi:

$$|Y \cap T| \sim B(k, \frac{1}{2})$$

Iš to išplaukia, kad jeigu $|Y \cap T| = r$, tai modifikuota Y vertė (suma visų x_i elementų) yra $\frac{r}{k} \cdot v$. Tarkime, kad l yra $\frac{k}{2}$ didžiausi rinkinio Y elementai. Jų modifikuotos reikšmės vidurkis yra:

$$\sum_{r=1}^{k}[\mathsf{P}(|Y \cap T| = r) \cdot (\frac{min(r,l)}{k} \cdot v)] = \sum_{r=1}^{k}[\frac{k!}{(k-r)! \cdot r!} \cdot \frac{1}{2^k} \cdot (\frac{min(r,l)}{k} \cdot v)] =$$

$$\sum_{r=1}^{[k/2]-1}[\frac{k!}{(k-r)! \cdot r!} \cdot \frac{1}{2^k} \cdot \frac{r)}{k} \cdot v] + \sum_{r=[k/2]}^{k}[\frac{k!}{(k-r)! \cdot r!} \cdot \frac{1}{2^k} \cdot \frac{[k/2]}{k} \cdot v] =$$

$$\frac{v}{2} \cdot [\sum_{r=1}^{[k/2]-1}[\frac{k!}{(k-r)! \cdot r!} \cdot \frac{1}{2^k} \cdot \frac{2r}{k} + \sum_{r=[k/2]}^{k}[\frac{k!}{(k-r)! \cdot r!} \cdot \frac{1}{2^k} \cdot \frac{[k/2]}{k}] =$$

$$\frac{v}{2} \cdot [\sum_{r=1}^{[k/2]-1}[\frac{k!}{(k-r)! \cdot r!} \cdot \frac{1}{2^k} \cdot \frac{2r}{k} + 1 - \sum_{r=1}^{[k/2]-1}[\frac{k!}{(k-r)! \cdot r!} \cdot \frac{1}{2^k} =$$

$$\frac{v}{2} \cdot [1 - \sum_{r=1}^{[k/2]-1} \frac{k!}{(k-r)! \cdot r!} \cdot \frac{1}{2^k} \cdot \frac{2r}{k} - \frac{k!}{(k-r)! \cdot r!} \cdot \frac{1}{2^k} =$$

$$\frac{v}{2} \cdot [1 - \sum_{r=1}^{[k/2]-1} \frac{k!}{(k-r)! \cdot r!} \cdot \frac{1}{2^k} \cdot (1 - \frac{2r}{k})] \geq \frac{v}{2} \cdot [1 - \frac{1}{2\sqrt{k}}]$$

Šio algoritmo pasirinktų elementų modifikuota vertė yra nemažesnė už $(1 - \frac{5}{\sqrt{\frac{k}{2}}}) \cdot \frac{v}{2} \cdot [1 - \frac{1}{2\sqrt{k}}]$

Pažymėkime Z likusių $n - m$ elemetų rinkinį. Anot Kleinberg'o, algoritimas parenka likusius $k - l$ elementus, kurių modifikuota vertė yra nemažesnė nei $v \cdot (\frac{1}{2} - \frac{1}{\sqrt{k}})$.

Sulyginus abi gautas reikšmes įrodoma, kad naudojant Kleinbergo pasiūlytą algoritmą pasirinktų elementų vertė yra nemažesnė nei $(1 - \frac{5}{\sqrt{k}}) \cdot v$.

Iš kelių pasirinkimų sekretorės problemos galiausiai išsivystė taip vadinama *"Knapsack secretary"* problema, nagrinėjama Babaioff, M., Immorlica, N., Kempe, D. and Kleinberg, R., (2007) darbe. Uždavinys formalizuojamas taip: turimas rinkinys $U = 1, 2, ..., n$, sudarytas iš n elementų, kiekvienas iš kurių turi savo svorį $w(i)$ ir vertę $v(i)$. Pažymėjus tam tikrą pasirinktų elementų rinkinį S galima teigti, kad $w(S) = \sum_{i \in S} w_i$ ir $v(S) = \sum_{i \in S} v_i$. Uždavinys yra maksimizuoti $\sum_{i \in S} w_i$, atsižvelgiant į apribojimą $\sum_{i \in S} v_i \leqslant W$, kur W yra tam tikra konstanta, internetinės paieškos aukcionų atveju - turimas biudžetas.

Elementai $i \in U$ atvyksta atsitiktine tvarka. Kitaip nei realybėje, n yra žinomas, tačiau svorių ir verčių pasiskirstymai nėra žinomi. Pasirodžius *i-tąjam* elementui, sužinomas jo svoris w_i ir vertė v_i. Kaip ir tradicinėje sekretorės problemoje, algoritmas turi nuspręsti ar *"priimti"* elementą i, ar ne. Pasirinkti elementai vėliau negali būti išmesti ir atvirkščiai - atmesti elementai negali būti vėliau pasirinkti. Tokiu būdu sudaromas elementų rinkinys S, kuris privalo tenkinti sąlygą $w(S) \leqslant W$.

Autoriai pasiūlo kiek lengvesnį kelių pasirinkimų sekretorės problemos algoritmą, kuris yra naudojamas toliau plėtojant *"Knapsack secretary"* algoritmą. Tai yra tiesiog *"Knapsack secretary"* problemos algoritmo atskiras atvejis kai $w(i) = 1$, o $W = k$. Algoritmas per pirmuosius t žingsnių sukonstruoja atskaitos rinkinį R, į kurį patenka k didžiausių per t žingsnių stebėtų elementų. Tarkime, kad $j_1, j_2, j_3, ..., j_k$ yra rinkinio R elementai, išrikiuoti mažėjančia tvarka pagal $v(j_i)$. Elementas $i > t$ yra pasirenkamas tada ir tik tada, kai $v(i) > v(j_k)$. Tokiu atveju j_k elementas yra išmetamas iš rinkinio R, o elementas $v(i)$ užima jo vietą. Autoriai teigia, jog tokio algoritmo efektyvumo santykis (ang. *competitive ratio*) artėja prie e kai $n \to \infty$.

Tarkime, kad $v_1^*, v_2^*, v_3^*, ..., v_k^*$ yra didžiausi rinkinio $V = v_1, v_2, v_3, ..., v_n$ elementai, i_a^*,

$a \in 1, 2, 3, ..., k$ yra elemento v_a^* pozicija sekoje V. Tokiu atveju

$$\mathsf{P}(i_a^* \in S) = \sum_{i=t+1}^{n} \frac{1}{n} \cdot \frac{t}{i-1} = \sum_{i=t+1}^{n} \frac{t}{n} \cdot \frac{1}{i-1} = \frac{t}{n} \sum_{i=t+1}^{n} \frac{1}{i-1} > \frac{t}{n} \int_t^n \frac{1}{x} \mathrm{d}x = \frac{t}{n}(ln(n) - ln(t)) = \frac{t}{n} ln(\frac{n}{t})$$

Vidutinė algoritmo pasirinkto rinkinio S vertė yra

$$E(v(S)) \geqslant \sum_{a=1} k\mathsf{P}(i_a^* \in S) \cdot v_a^* > \frac{t}{n} ln(\frac{n}{t}) \cdot v(S^*)$$

Ši funkcija maksimumą pasiekia:

$$(\frac{t}{n} ln(\frac{n}{t}) \cdot v(S^*))' = [\frac{v(S^*)}{n}(t \cdot ln(\frac{n}{t}))]' \propto [t \cdot ln(\frac{n}{t})]' = (t \cdot ln(n) - t \cdot ln(t))' = ln(n) - ln(t) - 1$$

$$ln(n) - ln(t) - 1 = 0, \text{ kai } t = \frac{n}{e}$$

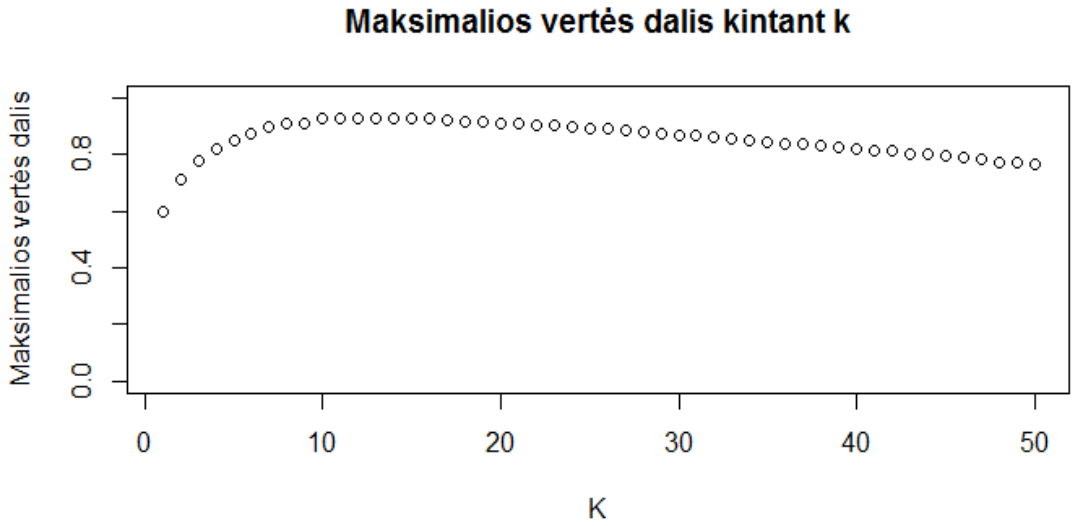
Tokiu atveju $E(v(S)) \to \frac{1}{e}$, kai $n \to \infty$, o $t = \frac{n}{e}$.

Pateikiamas algoritmo kodo pavyzdys R programoje.

```
results=data.frame(Maximum=integer(), Algorithm=integer())
n=200
k=20
loop=0
repeat {
loop=loop+1
S=0
t=floor(n/exp(1))
rand=floor(runif(n, min = 0, max = 1000))
offline=sort(rand, decreasing = TRUE)
offline=offline[1:k]
Maximum=sum(offline)
R=rand[1:t]
R=sort(R, decreasing = TRUE)
R=R[1:k]
for (i in ((t+1):n)){
  min=min(R)
  if (rand[i] > min){
    if (length(S)<k+1){
      S=c(S,rand[i])
      R=c(R[1:k-1], rand[i])
    }
  }
```

```
}
Algorithm=sum(S)
temp=data.frame(Maximum,Algorithm)
results=rbind(results,temp)
if (loop == 1000){
  break
  }
}
results<- cbind(results, results$Algorithm/results$Maximum)
names(results) <- c("Maximum", "Algorithm", "Fraction")
efficiency <- mean(results$Fraction)
efficiency
```

Šiuo atveju iš $n = 200$ reikia išsirinkti $k = 20$ elementų taip, kad jų suma būtų didžiausia. Algoritmas kartojamas 1000 kartų su naujomis sekomis. Algoritmo pasirinkti elementai tokioje situacijoje sudaro apie 91% galimos maksimalios k elementų vertės. Svarbu yra tai, kad didėjant k, o n tuo tarpu nesikeičiant, algoritmo efektyvumas ženkliai krenta. Pvz, kai $n = 200$, o $k = 50$, algoritmas pasirenka elementus, kurie sudaro tik apie 77% maksimalios galimos vertės. Taigi, algoritmas geriausiai veikia, kai k yra ženkliai mažesnis už n. Algoritmo efektyvumas, kai k kinta nuo 1 iki 50, o n išlieka 200, pateikiamas grafike.



Toliau nagrinėjamas atvejis, kai $w(i) \neq 1$. Elemento i tankis apibrėžiamas kaip $\rho(i) = \frac{v(i)}{w(i)}$. Šį kartą t yra atsitiktinis dydis, pasiskirstęs pagal binominį dėsnį $B(n, \frac{1}{2})$. Pažymėkime $X = 1, 2, 3, ..., t$ ir $Y = t+1, t+2, t+3, ..., n$. Algoritmas nepasirenka nei vieno elemento iš X, tačiau stebi kiekvieno rinkinio elemento vertę $v(i)$ ir svorį $w(i)$. Vi-

dutinį rinkinio X tankį pažymėkime ρ_X. Algoritmas tada pasirenka kiekvieną elementą iš Y kuris tenkina sąlygas:

- $w(i) \leq 3^{-4}$. Nagrinėjamas atvejis, kai bendras svoris W negali viršyti 1. Todėl visų elementų svoriai $w(i) \leq 1$.
- $\rho(i) \geq \sqrt{\rho_X}$
- $w(S_{<i} \cup i \leq 1)$

Taigi, kyla klausimas, kaip ši sekretorės problemos generalizacija siejasi su internetinės paieškos aukcionais, analizuojamais šiame darbe. Paskutinis šiame skyriuje nagrinėjamas algoritmas iš atsitiktine tvarka pasirodančių, skirtingus svorius ir vertes turinčių elementų stengiasi parinkti tokius elementus, kad jų bendra vertė būtų didžiausia, o svoris neviršytų tam tikros konstantos. Su panašia situacija susiduria ir reklamos užsakovai - paieškos užklausos (elementai) ateina atsitiktine tvarka, jų kaina ir svoriai iki jiems pasirodant nėra žinomi. Reklamos užsakovo tikslas yra pasirinkti tokias užklausas, kurios sugeneruotų maksimalų pelną (maksimizuoti bendrą elementų vertę) neviršijant biudžeto (svorio apribojimo). Vis dėlto, skirtumas nuo realios situacijos yra toks, kad realybėje raktažodžio vertė yra sužinoma tik tada, kai jis yra nuperkamas - t.y. tik nupirkus paspaudimą galima sužinoti, ar tas paspaudimas iš tikro atneš tam tikrą pelną.

Apibendrinant, kiekvienas iš šiame skyriuje aptartų modelių vienaip ar kitaip neatitinka realios situacijos, į kurią patenka reklamos užsakovai, norėdami reklamuotis internetinės paieškos rezultatuose:

- MDP's prieigoje daroma prielaida, kad kiekvienas reklamos parodymas garantuoja paspaudimą, ignoruojami tolimesni vertotojo veiksmai.
- Stochastinius modelius nagrinėjančioje literatūroje nurodomi algoritmai yra *NP-hard*.
- Modeliuose, kurie remiasi sekretorės problemos generalizacija, daroma prielaida, kad užklausos vertė w_i yra žinoma jai pasirodžius. Realybėje, elemento vertė nėra žinoma tol, kol jis nepasirenkamas.

6 BIUDŽETO OPTIMIZAVIMAS

Kaip jau buvo aptarta literatūros apžvalgoje, sunku sukonstruoti algoritmą, kuris būtų pritaikomas bendruoju atveju ir pilnai atitiktų realybėje egzistuojančią situaciją. Todėl šiame skyriuje bus analizuojamas konkretus pavyzdys remiantis Kitts, B. ir Leblanc, B., (2004) pasiūlyta idėja atskirai įvertinti pagrindinius parametrus ir pagal juos modeliuoti bendrą aukciono modelį. Bus stengiamasi sukurti biudžeto optimizavimo modelį, pritaikomą vienam, šiame darbe analizuojamam atvejui. Problema formalizuojama taip:

$$b_{k,t} = argmax \sum_{k} \sum_{t} [(r_k - b_{k,t}) \cdot CTR_{k,t}(p_{k,t}(b_{k,t}, t), t) \cdot imp_{k,t}(b_{k,t})] \tag{6.1}$$

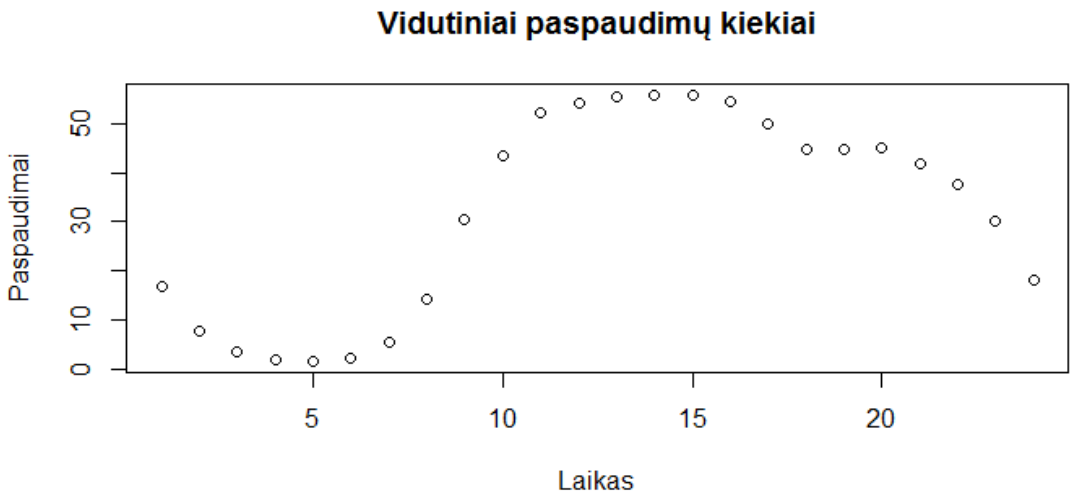
Čia:

- $b_{k,t}$ yra statymas ant $k - tojo$ raktažodžio laiko momentu t.
- r_k yra pelnas, gaunamas iš paspaudimo ant raktažodžio k.
- $p_{k,t}(b_{k,t}, t)$ yra reklamos pozicija raktažodyje k laiko momentu t, kai statymas yra $b_{k,t}$.
- $CTR_{k,t}$ yra paspaudimų ant $k - tojo$ raktažodžio, esančio pozicijoje p, funkcija.
- $imp_{k,t}$ yra $k - tojo$ raktažodžio reklamos parodymų skaičius.

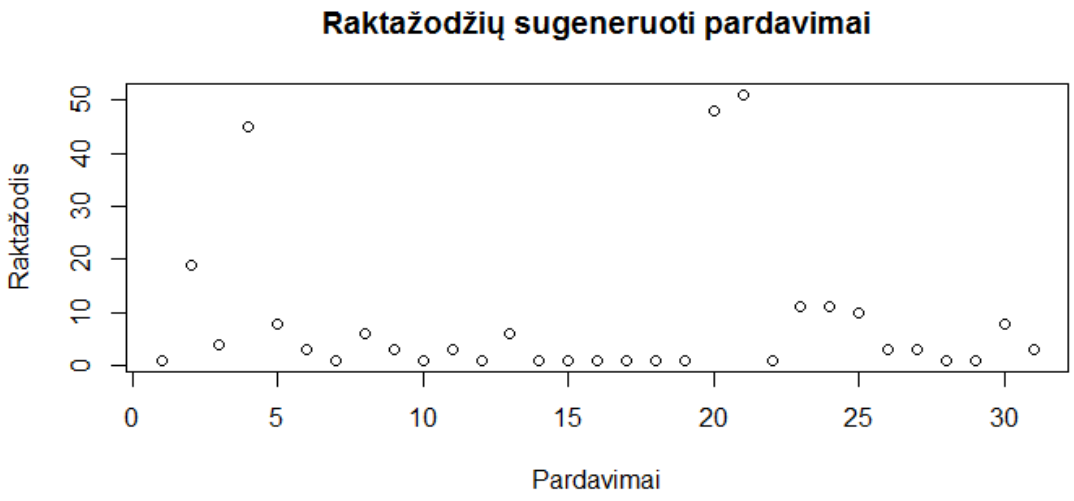
Nagrinėjamame pavyzdyje bus analizuojami 98 skirtingi naudojami raktažodžiai. Duomenys buvo renkami nuo 2015.06.01 iki 2016.12.10. Iš tolimesnės analizės buvo išmesti raktažodžiai, kurie nėra sugeneravę jokių pardavimų. Buvo paliktas 31 skirtingas raktažodis, sugeneravęs pardavimus per analizuojamą laikotarpį. Kiekvienam raktažodžiui stebėtas laikotarpis buvo išskaidytas į valandų intervalus. Tokiu būdu kiekvienas raktažodis turėjo po 13416 skirtingus stebėjimus. Buvo stebėti tokie kintamieji:

- *Keyword* - užklausą atitinkantis raktažodis.
- *Time* - dienos valanda žymima nuo 0 iki 23.
- *Bid* - siūlytas statymas raktažodžui K momentu t.

- *Clicks* - paspaudimų skaičius. Fiksuotas paspaudimų skaičius kiekvienam raktažodžiui laiko momentu t. Vidutinis visų raktažodžių paspaudimų skaičius, skirtingais laiko momentais, pateikiamas grafike.



- *Impressions* - reklamos parodymų skaičius.
- *Position* - vidutinė reklamos pozicija laiko momentu t.
- *Conversions* - raktažodžio sugeneruotų pardavimų skaičius stebėtą laiko momentą. Raktažodžiai nevienodai generuoja pardavimus - vieni raktažodžiai, labiau susiję su reklamuojamomis prekėmis, generuoja daugiau pardavimų už kitus. Kiekvieno raktažodžio sugeneruotų pardavimų kiekis per stebėtą laikotarpį pateikiamas grafike.



- *Conv. Value* - raktažodžio sugeneruotų pardavimų vertė stebėtą laiko momentą.
- *Share* - užklausų dalis, kuriai buvo parodyta reklama.

6.1 CTR prognozavimas

Šiame skyriuje bus modeliuojama paspaudimų funkcija - CTR (angl. *click through rate*). Kadangi paspaudimai priklauso nuo reklamos parodymų kiekio, tiesiogiai jie nėra prognozuojami. Vietoje to, bus prognozuojamas paspaudimų dažnis, pagal kurį bus galima apskaičiuoti ir realius paspaudimus.

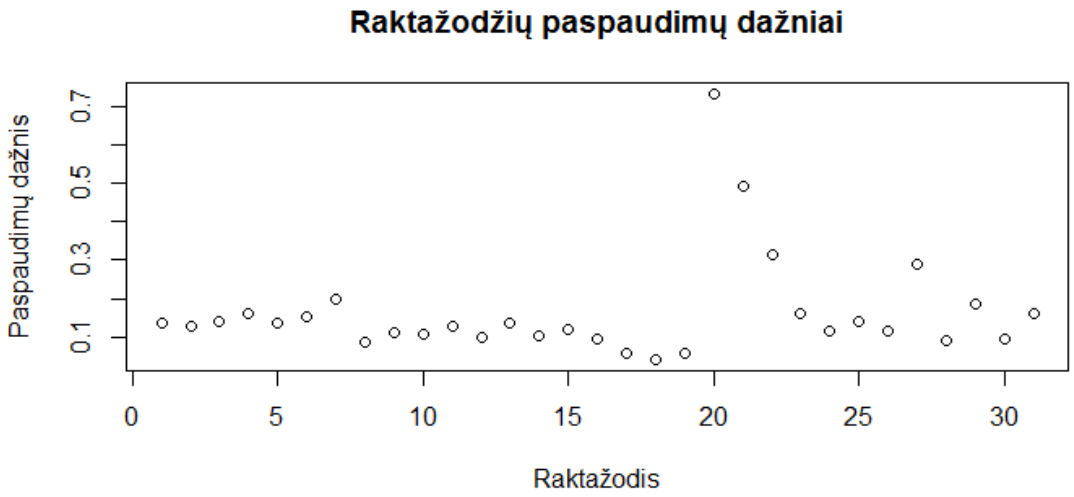
Logiška, kad CTR turėtų priklausyti nuo reklamos pozicijos ir (galbūt) nuo laiko momento t. CTR prognozavimui bus naudojami skirtingi *machine learning* metodai, tokie kaip tiesinė regresija, *ridge* regresija, *boosting* ir *random forest*.

Analizuojamus duomenis sudaro kiekvienos stebėtos dienos pozicija, CTR ir laiko momentas t. CTR paprasta išskaičiuojamas iš turimų duomenų pagal formulę

$$CTR_{k,t}(b) = \frac{clicks_{k,t}(b)}{imp_{k,t}(b)} \tag{6.2}$$

Visų raktažodžių vidutiniai CTR pateikiami grafike. Visi jie gana ženkliai skiriasi, todėl tikėtina, kad ir juos aproksimuojančios funkcijos skirsis. Dėl to *machine learning* algoritmai kiekvienam raktažodžiui buvo pritaikyti atskirai. Analizė iliustruojama $K02$ kintamuoju. Visas analizės R kodas pateikiamas prieduose.

Pirmiausia, visas analizuojamas duomenų masyvas atsitiktinai buvo padalintas į *training* ir *testing* duomenų rinkinius. Mokymuisi skirti duomenys sudaro 75% visų duomenų, testavimui palikta 25% visų stebėjimų.



Pritaikomas tiesinės regresijos modelis

$$CTR_{k,t,p} = \beta_0 + \beta_1 \cdot t_k + \beta_2 \cdot p_k + e_{k,t,p}$$

Pateikiami raktažodžio $K02$ tiesinės regresijos rezultatai.

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.211053   0.010104  20.888  < 2e-16 ***
Position    -0.047892   0.003526 -13.581  < 2e-16 ***
Time1       -0.030062   0.011629  -2.585  0.00976 **
Time2       -0.024897   0.012340  -2.018  0.04368 *
Time3       -0.042732   0.013418  -3.185  0.00146 **
...
...
...
Time23      -0.025620   0.016329  -1.569  0.11671
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1521 on 6195 degrees of freedom
Multiple R-squared:  0.03579,Adjusted R-squared:  0.03206
```

Rezultatai indikuoja, kad CTR priklauso tiek nuo pozicijos, tiek ir nuo laiko momento t - kintamojo *Position* p reikšmė yra $< 2e - 16$. Kaip ir tikėtasi, koeficientas prie pozicijos kintamojo yra neigiamas - kuo didesnė (žemiau esanti) pozicija, tuo mažesnis CTR. Nepaisant to, jog abu kintamieji yra statistiškai reikšmingi, modelio determinacijos koeficientas R yra itin žemas - 0.03579.

Sugeneruotas modelis išbandomas su testavimo duomenų rinkiniu ir apskaičiuojama vidutinė kvadratinė paklaida MSE. Pagal MSE bus lyginamas skirtingų metodų efektyvumas. Tiesinės regresijos modelio testinių duomenų **MSE yra 0.02522814**.

Toliau analizuojama sąveika tarp kintamųjų *Position* ir *Time*. Tam papildomas tiesinės regresijos modelis. Gauti rezultatai indikuoja, kad egzistuoja statistiškai reikšminga sąveika tarp kintamųjų:

```
Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      0.229623   0.029291   7.839  5.3e-15 ***
Position        -0.058871   0.016637  -3.538 0.000405 ***
Time1           -0.086114   0.036903  -2.333 0.019654 *
Time2           -0.045967   0.037290  -1.233 0.217735
Time3           -0.112254   0.037671  -2.980 0.002895 **
Time4           -0.087008   0.038630  -2.252 0.024337 *
Time5           -0.099601   0.038959  -2.557 0.010595 *
...
...
...
Position:Time3   0.042118   0.021078   1.998 0.045738 *
Position:Time4   0.035214   0.021215   1.660 0.096989 .
Position:Time5   0.064190   0.021434   2.995 0.002757 **
Position:Time6   0.016637   0.020175   0.825 0.409611
...
...
...
```

Naujo tiesinės regresijos modelio determinacijos koeficientas yra 0.04554, t.y. didesnis, už anksčiau naudoto modelio. Pritaikius jį testiniams duomenims, gauta **MSE yra 0.02540487**. Gauta MSE - didesnė už paprastos tiesinės regresijos.

Kitas duomenims pritaikomas *machine learning* metodas yra *ridge* regresija. *Ridge* regresija nuo tiesinės regresijos skiriasi koeficientų apskaičiavimo būdu. Tiesinėje regresijoje koeficientai β_k yra parenkami taip, kad jie minimizuotų funkciją

$$\sum_{i=1}^{n}(y_i - \beta_0 + \sum_{j=1}^{k}\beta_j \cdot x_{i,j})^2$$

Ridge regresijoje koeficientai turi minimizuoti šiek tiek kitokią funkciją (James, G., Witten, D. and Hastie, T., 2014):

$$\sum_{i=1}^{n}(y_i - \beta_0 + \sum_{j=1}^{k}\beta_j \cdot x_{i,j})^2 + \lambda\sum_{j=1}^{k}\beta_j^2$$

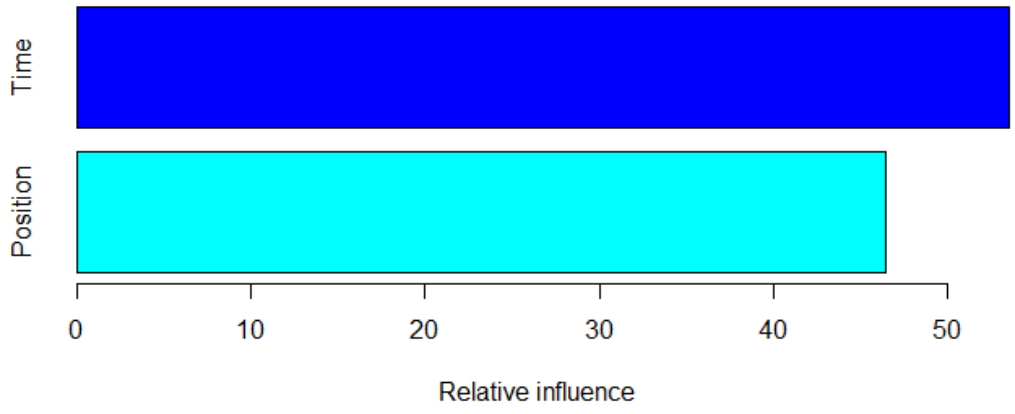
λ vadinamas sumažėjimo (angl. *shrinkage*) parametru. R pakete *glmnet*, nenurodžius λ reikšmės, išbandomos įvairios reikšmės ir parenkama λ reikšmė labiausiai tinkama duomenims. *Ridge* regresija sugeneruoja modelį, kurio **MSE yra 0.02576125**, t.y. šiek tiek didesnė, už tiesinės regresijos.

Kitas išbandomas prognozavimo metodas yra *boosting*. *Boosting* yra *tree based* metodas, kurį galima apibrėžti algoritmu (James, G., Witten, D. and Hastie, T., 2014):

1. $\hat{f}(x) = 0$ ir $r_i = y_i$.
2. Kiekvienam $b = 1, 2, ..., B$ atlikti:
   (a) Pritaikyti $\hat{f}^b$ duomenims (X, r).
   (b) Atnaujinti $\hat{f}$ pridedant pasvertą naujo regresijos medžio $\hat{f}^b$ versiją $\hat{f}(x) = \hat{f}(x) + \lambda \hat{f}^b(x)$.
   (c) Atnaujinti liekamąsias paklaidas $r_i = r_i - \lambda \hat{f}^b(x)$.
3. Gaunamas modelis $\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$

Boosting metodas pritaikomas raktažodžiui $K02$:

```
var  rel.inf
Time 53.57292
Position 46.42708
```



Rezultatai indikuoja, kad reklamos pozicija turi daugiau įtakos CTR, nei laiko momentas t.

Pritaikius *boosting* testiniams duomenimias, apskaičiuota **MSE yra 0.02519145**, t.y. šiek tiek mažesnė už tiesinės regresijos modelio MSE.

Galiausiai paskutinis pritaikomas modelis yra *random forest*. Šis metodas veikia *bootstrap* metodo principu - iš analizuojamo duomenų rinkino pakartotinai išrenkami atsitiktiniai poaibiai B, kuriems yra pritaikomas *decision tree* metodas. Tai atlikus visi modeliai sujungiami į vieną.

Random forest, pritaikius jį testiniams duomenimis, **MSE yra 0.02549085**.

Visų keturių naudotų modelių MSE įvertiniai pateikiami lentelėje.

	Linear	Ridge	Boosting	Random forest
MSE	0.02522814	0.02576125	0.02519145	0.02549085

Taigi, žemiausią MSE turi *boosting* metodas, kuris toliau bus naudojamas prognozuojant $K02$ CTR. Tokia pat procedūra atlikta su 31 analizuojamu raktažodžiu. Rezultatų santrauka pateikiama lentelėje.

Raktažodis	Efektyviausias modelis	MSE
K01	Linear	0.03972882
K02	Boosting	0.02519145
K04	Boosting	0,02537863
K07	Random Forest	0.01553107
K08	Linear	0,02444477
K10	Linear	0,03788226
K12	Linear	0,03025086
K14	Linear	0,01290083
K15	Boosting	0,01809969
K24	Boosting	0,08931335
K32	Linear	0,03001721
K36	Linear	0,05674077

Raktažodis	Efektyviausias modelis	MSE
K39	Linear	0,02323451
K44	Linear	0,03272535
K54	Linear	0,04280989
K57	Linear	0,04453403
K60	Linear	0,01202982
K65	Linear	0,01448793
K68	Random Forest	0,01141138
K71	Random Forest	0,09858233
K72	Boosting	0,09987203
K73	Boosting	0,1618511
K83	Linear	0,04614861
K85	Ridge	0,04296663
K88	Boosting	0,04227786
K91	Linear	0,06611905
K92	Linear	0,1160078
K93	Linear	0,09519855
K95	Random Forest	0,105658
K97	Linear	0.04081024
K98	Linear	0,07796385

6.2 Reklamos pozicijos prognozavimas

Google *Search* yra uždaras aukcionas, todėl nėra žinoma, kokio dydžio turi būti statymas, norint būti pirmoje, atroje ir t.t. pozicijoje. Tačiau analizuojmuose duomenyse turimi kintamieji *position* ir *bid*, kuriuos krostabuliuojant galima sužinoti, kokia buvo vidutinė reklamos pozicija momentu t, kai statymas ant raktažodžio k buvo $b_{k,t}$. Pagal šiuos duomenis galima sukonstruoti reklamos pozicijos prognozavimo modelį.

Vis dėlto, turimi duomenys apriboja analizės galimybes - kiekviename raktažodyje statymai nebuvo koreguojami nuo pat stebėjimo pradžios. Vadinasi, turimas tik vienas statymo įvertinys ir nėra aišku, kaip pakistų pozicijų pasiskirstymas jį šiek tiek sumažinus

ar padidinus. Tačiau situacija, kai statymas yra toks pats, o vidutinė reklamos pozicija tam tikru momentu pasikeičia, indikuoja pačios rinkos pokyčius - naujų žaidėjų atsiradimą, jų veiksmus ir t.t. Taigi, pradžioje bus įvertinta vidutinės reklamos pozicijos kaita, kai statymas išlieka pastovus.

Analizei bus naudojami visų raktažodžių vidutinės pozicijos. Kiekviena stebėta diena, kurių buvo apie 200, segmentuota pagal valandas ir surasta kiekvienos valandos vidutinė reklamos pozicija. Tokio kiekio duomenų turėtų būti pankamai, kad pagrįstai būtų galima įvertinti pozicijos kaitą.

Iš pradžių atliktas ANOVA testas, parodantis, ar egzistuoja statistiškai reikšmingi skirtumai tarp vidutinės pozicijos skirtingais dienos momentais. Visos atliktos procedūros bus iliustruotos tik su raktažodžiu $K01$, tačiau analogiškai atliktos ir su likusiais raktažodžiais.

Pradžioje įvertinama duomenų normalumo hipotezė, t.y. ar vidutinė raktažodžio pozicija normaliai pasiskirsčiusi skirtingais laiko momentais t. Analizė iliustruojama remiantis raktažodžio $K01$ duomenimis momentu $t = 12$.

```
Shapiro-Wilk normality test

data:  k1_12
W = 0.92783, p-value = 2.171e-13
```

Shapiro-Wilk testas rodo, kad duomenys nėra normaliai pasiskirstę - $p < 0.05$. Tačiau tikėtina, jog tokie rezultatai gauti dėl didelio analizuojamų duomenų kiekio.

Patikrinama duomenų homoskedastiškumo hipotezė naudojant du testus - Bartlett'o ir Flinger-Killeen:

```
Bartlett's K-squared = 405.84, df = 23, p-value < 2.2e-16
Fligner-Killeen:med chi-squared = 274.84, df = 23, p-value <
2.2e-16
```

Tiek Bartlett'o, tiek Flinger-Killeen testai rodo, kad dispersijos tarp skirtingų grupių, kurios šiuo atveju yra laiko momentai t, yra nevienodos - $p < 0.05$. Vis dėlto, dėl itin didelio duomenų kiekio net ir nedidelį dispersijos nukrypimą šie testai gali fiksuoti kaip reikšmingą.

Toliau atliekama ANOVA analizė, lyginant vidutinę reklamos poziciją skirtingais laiko momentais.

```
> fit <- aov(positions$K01~positions$Time)
> summary(fit)
                 Df Sum Sq Mean Sq F value Pr(>F)
positions$Time   23    161   6.986   10.97 <2e-16 ***
Residuals      6893   4391   0.637
```

ANOVA pirmojo raktažodžio analizė rodo statistiškai reikšmingus skirtumus tarp vidutinės reklamos pozicijos skirtingais laiko momentais t. Pateikiami ir antrojo raktažodžio ANOVA rezultatai:

```
> fit <- aov(positions$K02~positions$Time)
> summary(fit)
                 Df Sum Sq Mean Sq F value   Pr(>F)
positions$Time    1    9.3   9.297   29.43 5.95e-08 ***
Residuals      8234 2600.9   0.316
```

Tiek TukeyHSD, tiek porinis t testas indikuoja, kad labiausiai vidutinė pozicija skiriasi tarp nakties ir dienos laikų. Pritaikius ANOVA analizę visiems raktažodžiams paaiškėjo, kad $K54$, $K73$ ir $K91$ raktažodžiuose vidutinės reklamos pozicijos skirtinguose laiko momentuose reikšmingai nesiskiria.

Svarbu suprasti, kad čia analizuotos vidutinės pozicijos yra tokios tik su duomenyse buvusiais statymais. Vis dėlto, vidutinė pozicija yra funkcija nuo siūlomo statymo. Šią funkciją įvardinkime kaip $p_{t,k}(b)$. Be to, kiekvienam naudojamam raktažodžiui ši funkcija yra skirtinga. Funkcijos savybės:

- Funkcija turi būti mažėjanti, t.y. kuo didesnis statymas, tuo žemesnė (geresnė) pozicija.
- $p(0) = 0$.
- $MAXp(b_{t,k}) = 5$, nes egzistuoja tik 5 reklamai skirtos pozicijos Google Search aukcione.

Iš pradžių bus mėginama šią funckiją įvertinti iš turimų duomenų. Literatūroje, taip pat ir Kitts, B. ir Leblanc, B., (2004) straipsnyje, tokios funkcijos dažniausiai išreiškiamos

eksponentėmis. Funkcija galėtų atrodyti maždaug taip:

$$p_{k,t}(b) = 1 + \alpha \cdot e^{-\beta \cdot b}$$

Logaritmavus abi puses gaunama išraiška $ln(p_{k,t}(b) - 1) = -\beta \cdot b \cdot ln(\alpha) - \beta \cdot b$. Taigi, pagal turimus kintamuosius būtų galima modeliuoti $ln(p_{k,t}(b) - 1)$. Kaip jau buvo minėta, kiekvieno raktažodžio statymai nebuvo koreguoti nuo stebėjimo pradžios. Vis dėlto, statymai tarp skirtingų raktažodžių skiriasi - pagal tai būtų galima modeliuoti ir pozicijų kitimą. Žinoma, negalima tiesiogiai pritaikyti vieno raktažodžio statymo ir vidutinės pozicijos kitam raktažodžiui - skiriasi raktažodžių paklausa, nuo kurios priklauso ir vidutinė pozicija, gauta naudojant atitinkamą statymą.

Raktažodžių paklausą būtų galima įvertinti naudojant *Share* kintamąjį. Kintamasis *Share* parodo, kokiai daliai visų egzistuojančių užklausų buvo parodyta analizuojama reklama. Tačiau šis kintamasis nėra nepriklausomas - jį lemia statymas *Bid*. Dėl šios priežasties susiduriame su ta pačia problema - negalima patikimai įvertinti, kaip keistųsi kintamasis *Share* keičiantis statymui.

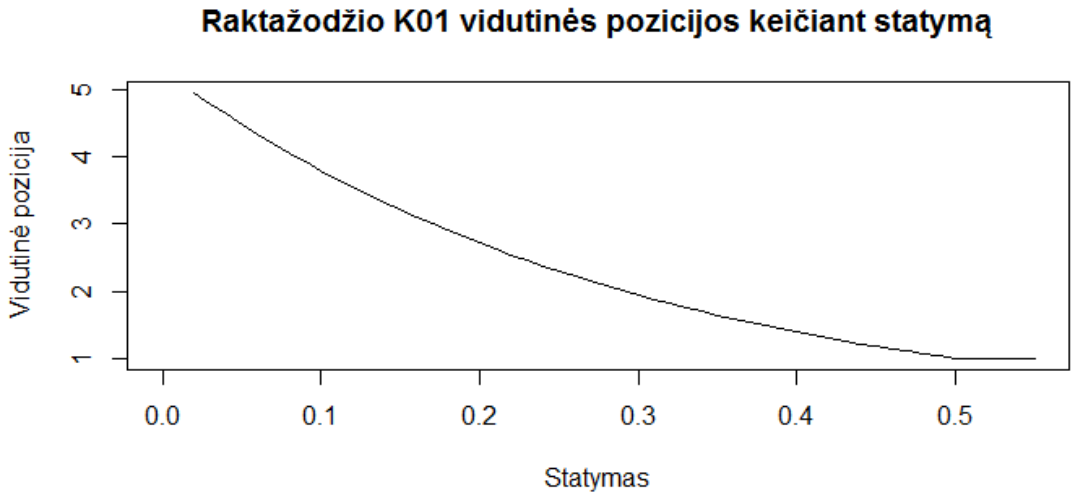
Taigi, duomenimis pagrįsti šios funkcijos konstravimo nėra galimybės, todėl šiame darbe bus tiesiog priimta, kad

$$p_{k,t}(b) = p^*_{k,t} \cdot e^{(1-\frac{bid_{k,t}}{bid^*_{k,t}})} \tag{6.3}$$

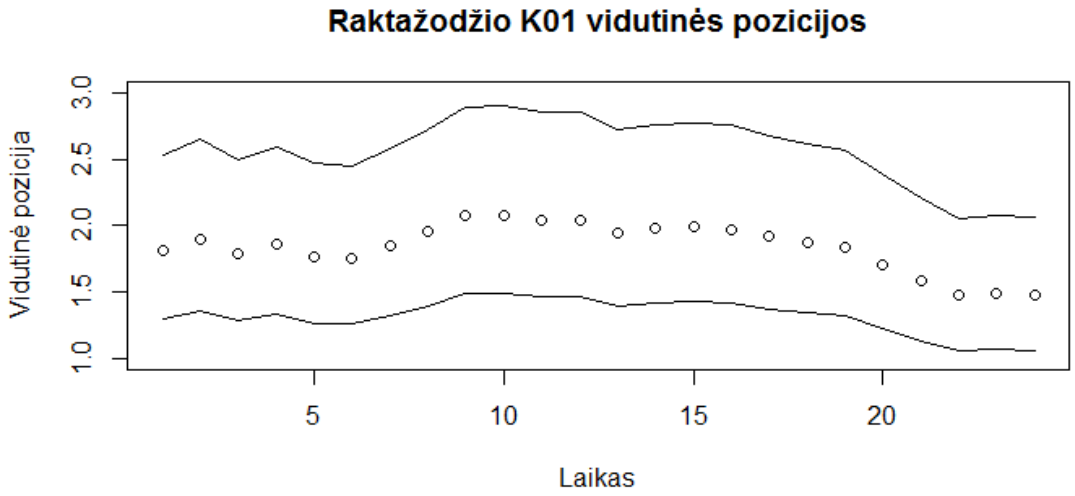
$p^*_{k,t}$ yra šiame skyriuje iškaičiuota reklamos pozicija, o $bid^*_{k,t}$ yra statymas, su kuriuo ta pozicija buvo gauta. Pritaikius šią formulę duomenys yra transformuojami remiantis šiomis taisyklėmis:

- Jei statymas yra lygus 0, vidutinė pozicija turi būti lygi 0.
- Jei prognozuojama vidutinė pozicija yra daugiau nei 5, jos reikšmė yra pakeičiama į 0, nes kaip jau buvo minėta, egzistuoja tik 5 reklamai skirtos pozicijos. Jei išskaičiuota vidutinė reklamos pozicija yra didesnė, nei 5, tai reiškia, kad statymas yra per mažas, kad vidutinė pozicija būtų tarp 5 egzistuojačių galimybių.
- Jei prognozuojama vidutinė pozicija yra žemesnė, nei vienas, ji pakeičiama į 1.

Pateikiamas pirmojo raktažodžio numatomų pozicijų grafikas momentu $t = 12$, kai statymas svyruoja nuo 0 iki 0,55 Eur.



Pateikiamas grafikas, kaip keičiasi raktažodžio $K01$ vidutinė pozicija momentais t, kai statymas aukštesnis arba žemesnis už stebėtą 0.1 Eur.



2 pav. Viršutinė linija - vidutinė reklamos pozicija, kai statymas yra $b_{1,t} - 0,1$. Viduryje esantys taškai - stebėtos vidutinės reklamos pozicijos. Apatinė linija nurodo vidutinę reklamos pozicija, kai statymas yra $b_{1,t} + 0,1$.

Šiame skyriuje naudotas R kodas pateikiamas prieduose.

6.3 Reklamos kiekio prognozavimas

Šiame skyriuje bus įvertintas raktažodžių K_i užklausų skaičius laike t. Kitts, B. ir Leblanc, B. savo straipsnyje į tai nekreipia dėmesio, nes, matyt, analizuoja rinką, kurioje užklausų skaičius kiekvieną valandą yra itin didelis. Šiame darbe analizuojama Lietuvos rinka, todėl akivaizdu, kad paieškos užklausos kiekvieną valandą turi tam tikras *lubas*. Jos daro didelę įtaką ir reklamai - reklama negali būti parodoma daugiau kartų, nei yra užklausų. Na, o analizuojami pardavimai yra funckija nuo paspaudimų, kurie savo ruožtu yra funkcija nuo reklamos parodymų.

Tiesiogiai negalima sužinoti, kiek užklausų egzistuoja konkrečiam raktažodžiui, tačiau iš turimų duomenų galima patikimai tai įvertinti naudojant formulę

$$\text{max.imp}(k_j, t) = \frac{\text{imp}}{\text{share}}$$

T.y. nors nėra žinoma, kiek užklausų būna tam tikru laiko momentu, tarp analizuojamų duomenų turimi kintamieji *impressions* ir *share*, atitinkamai parodantys, kiek kartų momentu t buvo parodyta reklama ir kokią dalį nuo visų užklausų tai sudarė. Taigi, analizuojamus duomenis sudaro išskaičiuoti maksimalūs raktažodžio parodymai kiekvienos stebėtos dienos laiko momentu t.

Atliekama analizė, panaši į jau naudotą reklamos pozicijos įvertinimui. Pradžioje ištiriama, ar duomenys turi normalų pasiskirstymą. Analizė iliustruojama remiantis raktažodžio $K01$ duomenimis momentu $t = 12$.

Kaip ir vidutinės pozicijos atveju, Shapiro-Wilk testas rodo, kad duomenys nėra pasiskirstę pagal normalųjį skirstinį.

Atliekami dispersijų vienodumo testai:

```
Bartlett's K-squared = 4577.5, df = 23, p-value < 2.2e-16
Fligner-Killeen:med chi-squared = 1667.3, df = 23, p-value <
2.2e-16
```

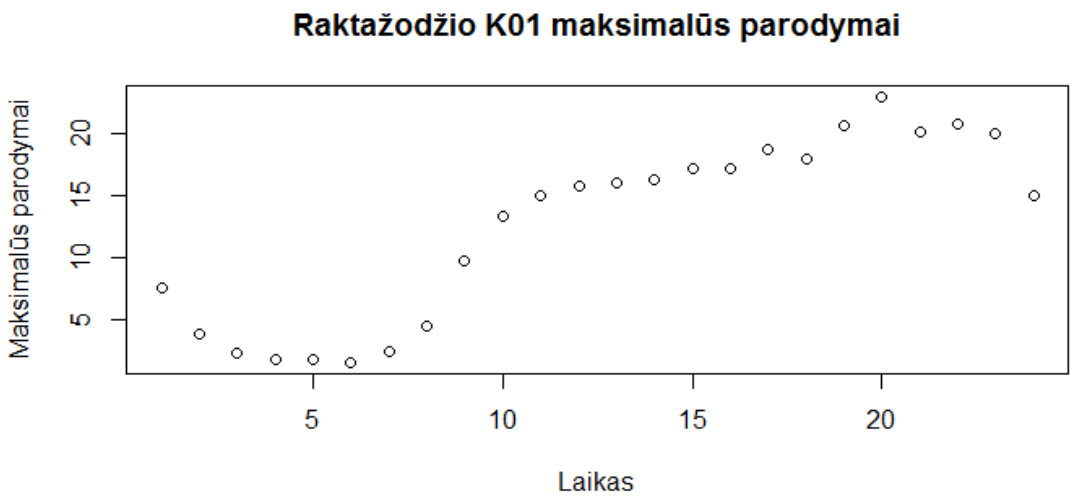
Atliekama ANOVA analizė, iliustruojama su $K01$ kintamuoju.

```
> fit <- aov(max.imp$K01~max.imp$Time)
> summary(fit)
```

```
               Df  Sum Sq Mean Sq F value Pr(>F)
max.imp$Time   23  275456   11976   81.67 <2e-16 ***
Residuals    6893 1010842     147
```

Anova rodo statistiškai reikšmingus skirtumus tarp maksimalių parodymų kiekio skirtingais laiko momentais t praktiškai visuose raktodžiuose. Vienintelis K93 raktažodis neturi statistiškai reikšmingų skirtumų tarp dienos valandų.

Toliau analizėje bus naudojamas vidutinis maksimalių parodymų kitimas dienos eigoje. Raktažodžio $K01$ maksimalių parodymų kitimas pateikiamas grafike.



Kaip ir pozicijos prognozavime, reklamos parodymų kiekis priklauso nuo statymo. Todėl ir čia bus laikoma, kad maksimalių parodymų dalis, kuriai bus parodyta analizuojama reklama, kinta tokiu principu:

$$sh_{k,t}(b) = sh^*_{k,t} \cdot \frac{1}{e^{1-\frac{bid_{k,t}}{bid^*_{k,t}}}} \tag{6.4}$$

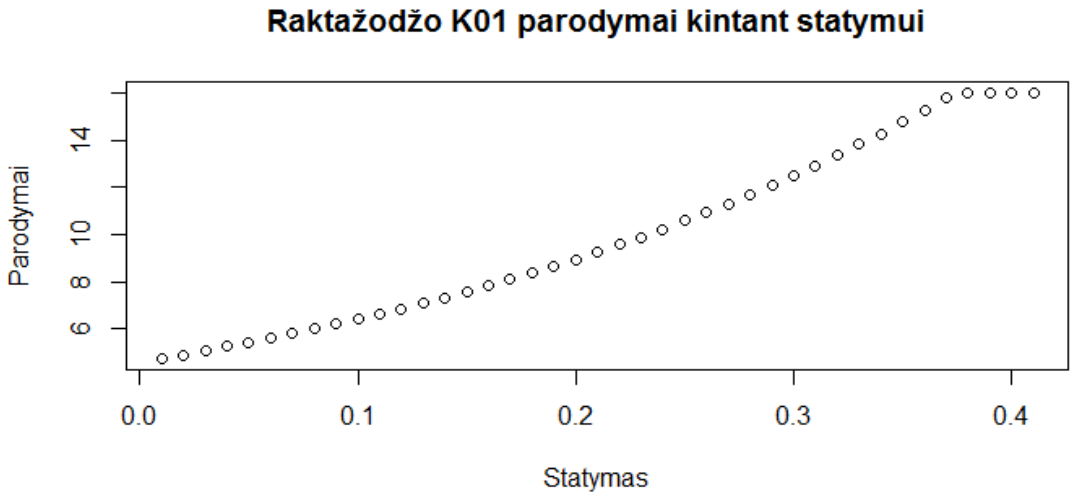
Čia:

- $sh_{k,t}$ - prognozuojama maksimalių parodymų dalis.
- $sh^*_{k,t}$ - stebėta makslimalių parodymų dalis.
- $bid_{k,t}$ - siūlomas statymas.
- $bid^*_{k,t}$ - statymas analizuojamuose duomenyse.

Kaip ir reklamos pozicijos atveju, atliekamos funkcijos modifikacijos, kad gautos reikšmės atitiktų realybę:

- $MAX(sh_{k,t}) = 1$, t.y. užklausų dalis, kuriai bus parodyta reklama, negali būti didesnė už 100%.
- Jei $bid_{k,t} = 0$, tai $sh_{k,t} = 0$.

Naudojant tokį patį principą, kaip ir pozicijų prognozavime, apskaičiuojama, kokia turėtų būti maksimalių parodymų dalis, kuriai bus parodyta reklama. Paprastos daugybos būdu iš to apskaičiuojama, kiek kartų bus parodyta reklama. Raktažodio $K01$ parodymų kitimas momentu $t = 12$ iliustruotas grafike.



Šiame skyriuje naudojamas R kodas pateikiamas prieduose.

6.4 Pardavimų prognozavimas

Paskutinis parametras aukciono modelyje, kurį reikia įvertinti, yra pardavimai. Pardavimai vertinami pagal jų dažnį ir vertę. Vidutinė pardavimo vertė yra skirtinga tarp raktažodžių - jie gali reklamuoti skirtingas prekes ar paslaugas. Tačiau bus laikoma, kad to paties raktažodžio pardavimo vertė nepriklauso nuo laiko momento t - nėra priežasčių manyti, kad pvz. naktį perkamos didesnės vertės prekės, nei dieną. Vidutinės kiekvieno raktažodžio pardavimų kainos pateikiamos lentelėje.

K01	K02	K04	K07	K08	K10	K12	K14	K15	K24	K32	K36	K39	K44
894	417	323	420	315	346	49	355	1	379	285	381	100	269

K54	K57	K60	K65	K68	K71	K72	K73	K83	K85	K88	K91	K92	K93
688	294	378	55	770	379	269	791	646	131	14	17	33	149

K95	K97	K98
235	109	1

Toliau bus įvertinamas pardavimų dažnis, t.y. kaip dažnai po paspaudimo vartotojai kažką iš tikro nusiperka. Yra priežasčių manyti, kad pardavimų dažnis priklauso nuo laiko momento t, todėl pradžioje bus įvertinamas pardavimų pasiskirstymas laike. Kaip ir ankstesnių parametrų įvertinime, bus naudojama ANOVA procedūra.

Nors Bartlett testas rodo, kad $K01$ raktažodžio dispersija nėra homogeniška skirtingais laiko momentais, Fligner-Killeen indikuoja atvirkščiai:

```
Bartlett's K-squared = Inf, df = 23, p-value < 2.2e-16
Fligner-Killeen:med chi-squared = 16.249, df = 23, p-value =
0.8444
```

Tęsiama ANOVA analizė. $K01$ raktažodyje statistiškai reikšmingų skirtumų tarp pardavimų kiekio skirtingais liko momentais nėra:

```
> #Anova
> fit <- aov(conv$K01_conv~conv$Time)
> summary(fit)
              Df Sum Sq   Mean Sq F value Pr(>F)
conv$Time      1 0.0000 3.561e-05   0.246   0.62
Residuals   6915 0.9998 1.446e-04
```

Iš tikro nei viename raktažodyje nebuvo pastebėta statistiškai reikšmingų skirtumų tarp pardavimų kiekio skirtingais laiko momentais.

Toliau mėginama pardavimų dažnį prognozuoti remiantis vidutine reklamos pozicija. Kaip jau buvo aptarta, vidutinė reklamos pozicija daro reikšmingą įtaką paspaudimams, tačiau galbūt gali lemti ir pardavimų dažnį. Vis dėlto, paprasta tiesinė regresija indikuoja, kad taip nėra:

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  9.021e-04  9.746e-04   0.926    0.355
Position    -4.768e-05  5.377e-04  -0.089    0.929
```

Tik trijuose iš visų analizuotų raktažodžių buvo pastebėta bent minimaliai reikšminga priklausomybė tarp pardavimų dažnio ir reklamos pozicijos. Tačiau net ir juose determinacijos koeficientas buvo itin žemas, pvz. K91 raktažodžio determinacijos koeficientas buvo 0.01637.

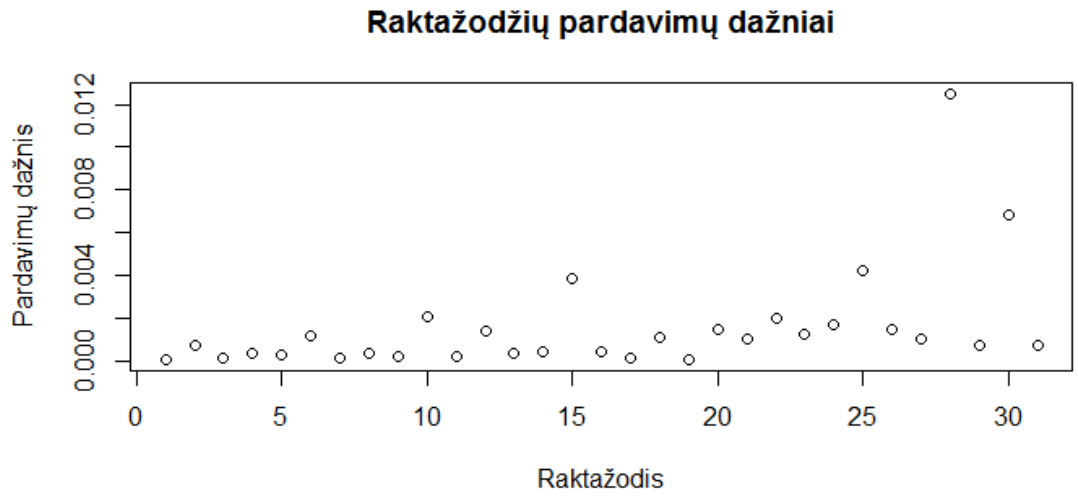
Taigi, galima daryti išvadą, kad pardavimų dažnis nepriklauso nei nuo reklamos pozicijos, nei nuo laiko. Todėl užtenka įvertinti vieną bendrą raktažodžio pardavimų dažnį. Jie pateikiami lentelėje.

K01	K02	K04	K07	K08	K10	K12	K14	K15
6,3E-05	7,2E-04	1,1E-04	3,8E-04	2,5E-04	1,2E-03	1,0E-04	3,3E-04	2,2E-04

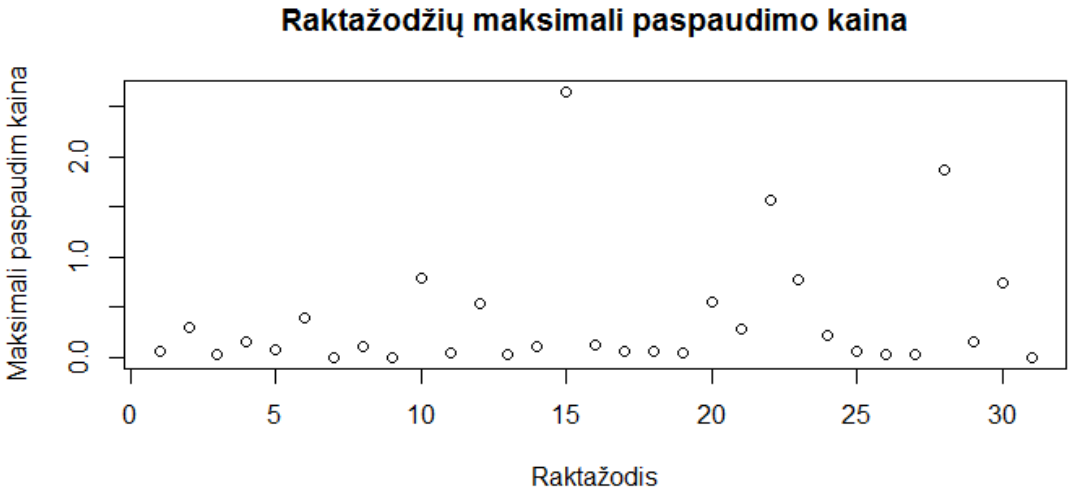
K24	K32	K36	K39	K44	K54	K57	K60	K65
2,1E-03	1,6E-04	1,4E-03	3,5E-04	3,9E-04	3,8E-03	4,2E-04	1,5E-04	1,1E-03

K68	K71	K72	K73	K83	K85	K88	K91	K92
5,4E-05	1,5E-03	1,0E-03	2,0E-03	1,2E-03	1,7E-03	4,2E-03	1,5E-03	1,0E-03

K93	K95	K97	K98
1,3E-02	6,9E-04	6,8E-03	7,1E-04



Iš pardavimo dažnio ir vidutinės pardavimo kainos galima išskaičiuoti vidutinį pelną nuo vieno paspaudimo. Kartu tai yra maksimali paspaudimo kaina, norint kad raktažodis būtų pelningas - jeigu paspaudimo kaina yra didesnė nei vidutinis pelnas, gaunamas iš vieno paspaudimo, raktažodio pelnas yra neigiamas.



Šiame skyriuje naudojamas R kodas pateikiamas prieduose.

6.5 Aukciono modelis

Praėjusiuose skyriuose buvo vertinami atskiri aukciono elementai, kurie čia bus sujungti į vieną prognozės modelį:

$$b_{k,t} = argmax \sum_{k} \sum_{t} [(r_k - b_{k,t}) \cdot CTR_{k,t}(p_{k,t}(b_{k,t}, t), t) \cdot imp_{k,t}(b_{k,t})]$$

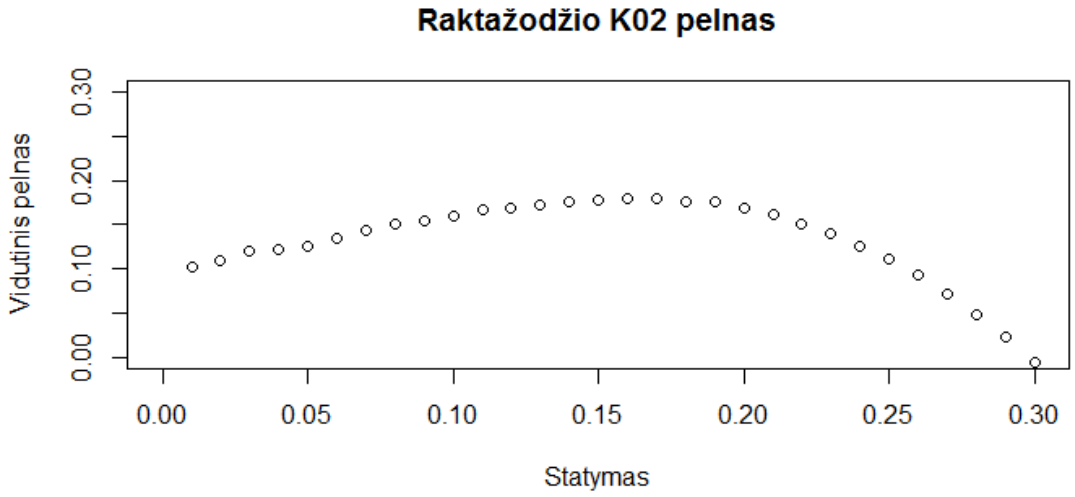
Turimi kintamieji yra:

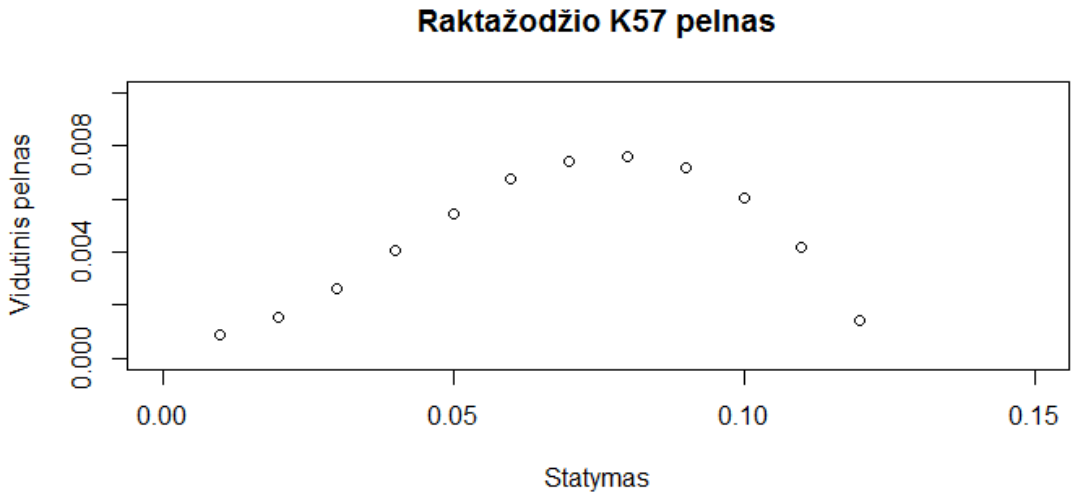
- Vidutinė reklamos pozicija $p_{k,t}$, apibrėžta kaip funkcija nuo statymų.
- Reklamos parodymai $imp_{k,t}$, apibrėžti kaip funkcija nuo statymų.
- CTR, apibrėžtas kaip funckija nuo laiko ir pozicijos.
- Pardavimai r_k, nepriklausantys nuo kitų kintamųjų.

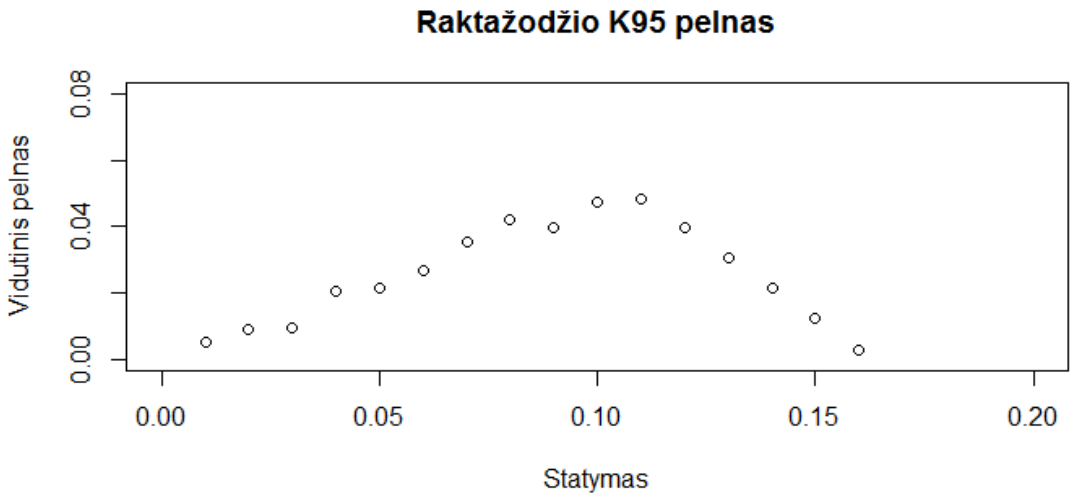
Su turimais kintamaisiais galima prognozuoti paspaudimus ant reklamos, o nuo jų - tikėtiną pelną ar nuostolius. Šiame darbe siekiama surasti tam tikrą statymą $b_{k,t}$, tai-

komą raktažodžiui K momentu t, tokį, kad raktažodžio pelnas būtų didžiausias. Taigi, maksimizavus kiekvieno raktažodžio atnešamą pelną, bus maksimizuotas ir bendras visų raktažodžių pelnas. Pirmiausia bus analizuojamas labiau realią situaciją atitinkantis atvejis, kai nustatomas vienas, visais laiko momentais raktažodžiui galiojantis statymas.

Visos jau aptartos kintamuosius įvertinančios funkcijos buvo sujungtos į vieną R kodą, kuris pateikiamas prieduose. Paprastu *for* ciklu kiekvienam raktažodžiui buvo paskaičiuotas vidutinis pelnas nuo vieno paspaudimo naudojant statymus nuo 0.01 Eur. iki 1 Eur. Kelių raktažodžių vidutinio pelno kitimas pateikiamas grafikuose.





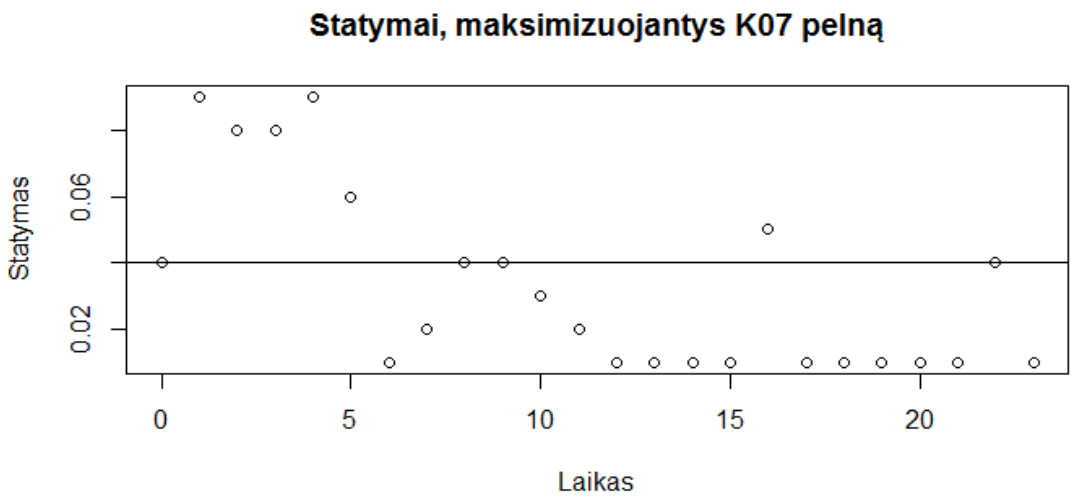


Taigi, iš grafikų galima matyti, kaip keičiasi raktažodžio vidutinis dienos pelnas kintant statymams. Tam tikras statymas maksimizuoja vidutinį dienos pelną - jis bus laikomas optimaliu dienos statymu. Pateikiama lentelė su apskaičiuotais optimaliais ir realiai naudotais statymais. Jei optimalus statymas lygus 0 - raktažodis ir esmės yra nepelningas, nepaisant naudojamų statymų.

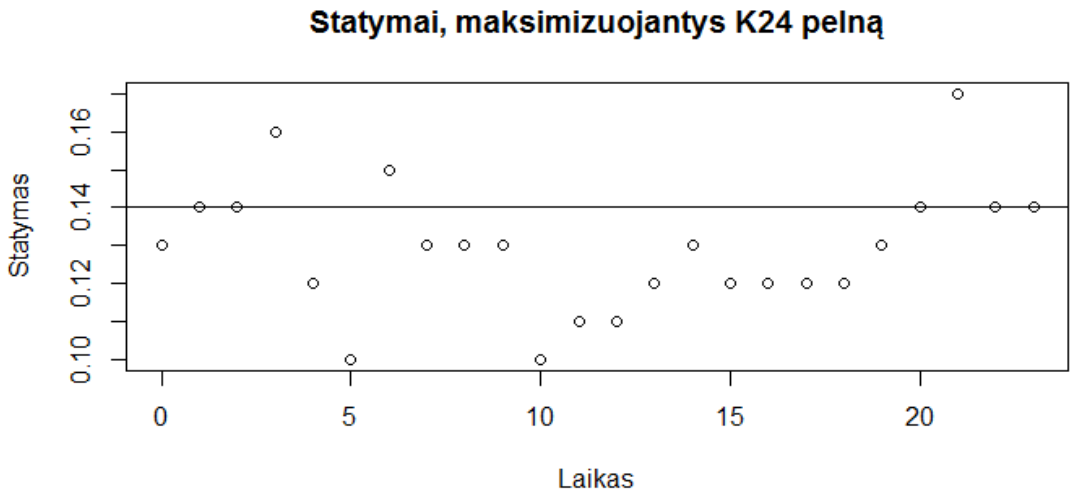
Raktažodis	Naudotas statymas	Optimalus statymas
K01	0,3	0,02
K02	0,3	0,16
K04	0,3	0,01
K07	0,3	0,04
K08	0,3	0,06
K10	0,1	0,12
K12	0,2	0
K14	0,12	0,08
K15	0,08	0
K24	0,09	0,14
K32	0,12	0,03
K36	0,09	0,17
K39	0,76	0,01
K44	0,15	0,06

Raktažodis	Naudotas statymas	Optimalus statymas
K54	0,23	0,52
K57	0,12	0,08
K60	0,15	0,03
K65	0,12	0,02
K68	0,41	0,01
K71	0,1	0,1
K72	0,1	0,1
K73	0,2	0,26
K83	0,04	0,12
K85	0,04	0,06
K88	0,08	0,03
K91	0,04	0,01
K92	0,14	0,01
K93	0,04	0,16
K95	0,07	0,11
K97	0,08	0,16
K98	0,08	0

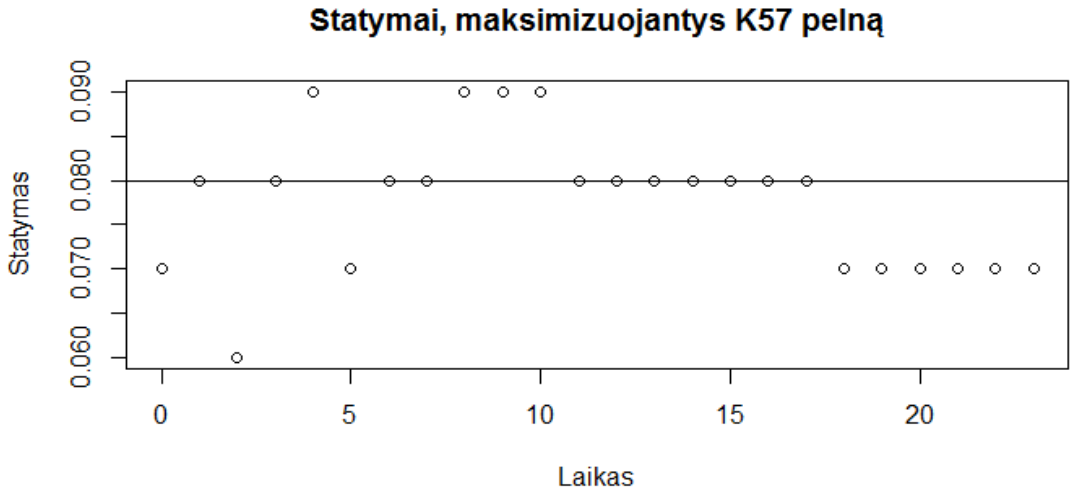
Kitas analiztuojas atvejis modeliuoja situaciją, kai statymas $b_{k,t}$ gali keistis momentais t. Tokiu atveju gaunami skirtingi statymai kiekvienu momentu t, maksimizuojantys pelną būtent tuo momentu. Toks modelis gaunamas šiek tiek modifikuojant tą patį naudotą aukciono modelio kodą. Grafikuose pateikiama, kokie yra optimalūs kai kurių raktažodžių statymai laiko momentu t.



3 pav. Statymai, maksimizuojantys raktažodžio pelną. Linija indikuoja optimalų statymą.



4 pav. Statymai, maksimizuojantys raktažodžio pelną. Linija indikuoja optimalų statymą.



5 pav. Statymai, maksimizuojantys raktažodžio pelną. Linija indikuoja optimalų statymą.

Galiausiai palyginus vidutinį dienos pelną, naudojant skirtingas optimizacijos strategijas, tampa akivaizdu, kad net ir nekoreguojant statymo kiekvieną valandą, sukurtas prognozavimo modelis generuoja ženkliai geresnius rezultatus.

Stebėti statymai	-2.32
Optimalūs statymai	10,94
Optimalūs statymai, keičiami kas valandą	11,11

7 IŠVADOS

Šiame darbe buvo stengtasi sukurti modelį, padedantį optimizuoti internetinės paieškos aukcionų reklamos biudžetą, surandant tinkamiausius statymus konkretiems raktažodžiams. Išanalizuotos dvi galimos situacijos - kai statymai yra nekintantys visos dienos eigoje ir kai statymus galima keisti kas valandą. Kaip jau buvo minėta, pirmasis variantas yra labiau atitinkantis realią situaciją, nes kiekvienas pakeitimas reikalauja tam tikrų žmogiškųjų išteklių. Rezultatai parodė, kad bendras dienos pelnas, naudojant šias dvi strategijas, ženkliai nesiskiria. Taip pat rezultatai atskleidė, kad buvo naudojami netinkami statymai, dėl kurių vidutinis dienos pelnas buvo neigiamas. Taip pat sukurtas modelis nurodė, kurių raktažodžių apskritai nevertėtų naudoti, nes jų pardavimų dažnis tiesiog yra per žemas, norint turėti net minimalų pelną.

Rezultatų patikimumas priklauso tinkamų parametrų įvertinimo. Skirtingų *machine learning* metodų panaudojimas leido tai tinkamai atlikti kiekvieno raktažodžio atveju. Vis dėlto, būtina buvo padaryti prielaidas apie reklamos kiekio ir pozicijos dinamiką keičiantis statymams. Todėl tolimesnių darbų kontekste būtina įvertinti šių prielaidų pagrįstumą – to nebuvo galima padaryti naudojant šiame darbe analizuojamus duomenis. Taip pat tolesni modelio tobulinimai turėtų įtraukti biudžeto lubas – modelis turėtų atsižvelgti, jog egzistuoja tam tikras dienos ar savaitės biudžetas, kurio viršyti negalima. Galiausiai verta paminėti, kad sukurtas modelis analizuoja biudžeto optimizavimą vidutinės dienos eigoje. Yra logiška manyti, kad rinkoje egzistuoja pirkimo bangos – pvz. Kalėdiniai apsipirkimai, *Black Friday*, etc. Todėl ilgalaikėje perspektyvoje būtų galima sukonstruoti modelį, kuris atsižvelgtų į pirkimų sezoniškumą ir pagal tai optimizuotų naudojamus statymus.

8 LITERATŪROS SĄRAŠAS

- Amin, K., Kearns, M., Key, P. and Schwaighofer, A., 2012. Budget optimization for sponsored search: Censored learning in mdps. arXiv preprint arXiv:1210.4847.

- Babaioff, M., Immorlica, N., Kempe, D. and Kleinberg, R., 2007. A knapsack secretary problem with applications. In *Approximation, randomization, and combinatorial optimization. Algorithms and techniques* (pp. 16-28). Springer Berlin Heidelberg.

- DasGupta, B. and Muthukrishnan, S., 2013. Stochastic budget optimization in internet advertising. *Algorithmica*, 65(3), pp.634-661.

- Feldman, J., Muthukrishnan, S., Pal, M. and Stein, C., 2007, June. Budget optimization in search-based advertising auctions. In *Proceedings of the 8th ACM conference on Electronic commerce* (pp. 40-49). ACM.

- Ferguson, T.S., 1989. Who solved the secretary problem?. *Statistical science*, pp.282-289.

- Freeman, P.R., 1983. The secretary problem and its extensions: A review. *International Statistical Review/Revue Internationale de Statistique*, pp.189-206.

- Gummadi, R., Key, P.B. and Proutiere, A., 2011, September. Optimal bidding strategies in dynamic auctions with budget constraints. In *Communication, Control, and Computing (Allerton)*, 2011 49th Annual Allerton Conference on (pp. 588-588). IEEE.

- James, G., Witten, D. and Hastie, T., 2014. An Introduction to Statistical Learning: With Applications in R.

- Yang, Y. and Wang, F., 2013. *Budget constraints and optimization in sponsored search auctions*. Elsevier.

- Kitts, B. and Leblanc, B., 2004. Optimal bidding on keyword auctions. *Electronic Markets*, 14(3), pp.186-201.

- Kleinberg, R., 2005, January. A multiple-choice secretary algorithm with applications to online auctions. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 630-631). Society for Industrial and Applied Mathematics.

- Kleywegt, Anton J., and Jason D. Papastavrou. "The dynamic and stochastic knapsack problem." *Operations Research* 46.1 (1998): 17-35.

- Muthukrishnan, S., Pál, M. and Svitkina, Z., 2010. Stochastic models for budget optimization in search-based advertising. *Algorithmica*, 58(4), pp.1022-1044.

9 PRIEDAI

9.1 CTR prognozacimas - R kodas

```
library("stats", lib.loc="C:/Program Files/R/R-3.3.1/library")
library("foreach", lib.loc="~/R/win-library/3.3")
library("ISLR", lib.loc="~/R/win-library/3.3")
library("tree", lib.loc="~/R/win-library/3.3")
library("randomForest", lib.loc="~/R/win-library/3.3")
library("gbm", lib.loc="~/R/win-library/3.3")
library("caret", lib.loc="~/R/win-library/3.3")
library("knitr", lib.loc="~/R/win-library/3.3")
library("MASS", lib.loc="C:/Program Files/R/R-3.3.1/library")
library("glmnet", lib.loc="~/R/win-library/3.3")

ctr <- read.table("ctr.txt", dec = ",", header = TRUE)
ctr$Time <- as.factor(ctr$Time)

#Sampling
smp_size <- floor(0.75 * nrow(ctr))
set.seed(123)
train_ind <- sample(seq_len(nrow(ctr)), size = smp_size)
train <- ctr[train_ind, ]
test <- ctr[-train_ind, ]

K02{
  train.K02 <- train[,c('Position','Time','K02')]
  train.K02 <- na.omit(train.K02)
  test.K02 <- test[,c('Position','Time','K02')]
  test.K02 <- na.omit(test.K02)

  #Linear regression K02
  linear.model <- lm(K02 ~ Position*Time, data = train.K02)
  summary(linear.model)
  K02_linear_pred <- predict(linear.model, newdata = test.K02)
  linear.MSE <- mean((test.K02$K02-K02_linear_pred)^2)
  linear.MSE

  linear.model <- lm(K02 ~ Position+Time, data = train.K02)
  summary(linear.model)
  K02_linear_pred <- predict(linear.model, newdata = test.K02)
  linear.MSE <- mean((test.K02$K02-K02_linear_pred)^2)
  linear.MSE

  #Ridge regression K02
  mat <- train.K02
  mat$Time <- as.integer(mat$Time)
  mat <- as.matrix(mat)
  mat2 <- test.K02
  mat2$Time <- as.integer(mat2$Time)
  mat2 <- as.matrix(mat2)
```

```
  x <- mat[,1:2]
  y <- mat[,3]
  new.y <- mat2[,3]
  new.x <- mat2[,1:2]
  ridge <- glmnet(x,y, alpha = 0)
  K02_ridge_pred <- predict(ridge, newx = new.x)
  ridge.MSE <- mean((new.y-K02_ridge_pred)^2)
  ridge.MSE

  #Boosting K02
  boost <- gbm(K02 ~ Position+Time, data = train.K02,
               distribution = 'gaussian',
               n.trees = 5000,
               interaction.depth = 4)
  summary(boost)
  K02_boost_pred <- predict(boost,test.K02, n.trees = 5000)
  boost.MSE <- mean((test.K02$K02-K02_boost_pred)^2)
  boost.MSE

  #Random Forests
  rf <- randomForest(K02 ~ Position+Time, data = train.K02,
    importance = TRUE)
  K02_rf_pred <- predict(rf, test.K02)
  rf.MSE <- mean((test.K02$K02-K02_rf_pred)^2)
  rf.MSE

  #Best model
  errors <- data.frame(linear.MSE,ridge.MSE,boost.MSE,rf.MSE)
  errors
  which.min(errors)
}
```

9.2 Reklamos pozicijos prognozavimas - R kodas

```
positions <- read.table("positions.txt", dec = ",", header = TRUE)
positions$Time <- as.factor(positions$Time)
library("psych")
library("stats")

#Normalumo testas
k1_12 <- positions[which(positions$Time==12),2]
shapiro.test(k1_12)

#Dispersijų vienodumo testai
bartlett.test(positions$K01, positions$Time)
fligner.test(positions$K01, positions$Time)

#Anova
fit <- aov(positions$K01~positions$Time)
summary(fit)
pairwise.t.test(positions$K01, positions$Time, p.adjust="bonferroni")
```

```
TukeyHSD(fit, conf.level = 0.95)

#Means
pos.means <- aggregate(positions[, 2:32], list(positions$Time),
    na.rm=TRUE, mean)

#Graph
plot(pos.means$K01, type ="p", ylim = c(1,3), ylab = "Vidutinė pozicija",
    xlab = "Laikas", main = "Raktažodžio K01 vidutinės pozicijos")

#Adding bids
bids <- read.table("bids.txt", dec = ",", header = TRUE)
pos.bids <- cbind(pos.means,bids)
pos.bids <- pos.bids[,2:63]

write.table(pos.bids, file = "pos.bids.txt", col.names = TRUE)

pos.bids <- read.table("pos.bids.txt", header = TRUE)
pos.bids <- as.matrix(pos.bids)
new.bids <- read.table("bids_new.txt", dec = ",")
new.bids <- as.matrix(new.bids)
est.pos <- matrix(nrow = 24, ncol = 31)
for (i in (1:31)){
  est.pos[,(i)] <- (pos.bids[,i]*(exp(1-new.bids[,i]/pos.bids[,i+31])))
}

est.pos[is.na(est.pos)]=0
new.bids[is.na(new.bids)]=0


#Jei daugiau nei 5 - bus 0 pozicija
est.pos.bids <- cbind(est.pos, new.bids)
for (i in (1:31)){
  for (j in (1:24)){
    if (est.pos.bids[j,i] > 5){
      est.pos.bids[j,i]=0
    }
  }
}

#Jei <1, bus 1 pozicija
for (i in (1:31)){
  for (j in (1:24)){
    if (est.pos.bids[j,i] < 1){
      est.pos.bids[j,i]=1
    }
  }
}

#Jei statymas 0, pozicija yra 0
for (i in (1:31)){
  for (j in (1:24)){
    if (est.pos.bids[j,31+i]==0){
```

```
      est.pos.bids[j,i]=0
    }
  }
}
```

9.3 Reklamos kiekio prognozavimas - R kodas

```
max.imp <- read.table("max.imp.txt", dec = ",", header = TRUE)
max.imp$Time <- as.factor(max.imp$Time)

#Normalumo testas
k1_12 <- max.imp[which(max.imp$Time==12),2]
shapiro.test(k1_12)

#Dispersijų vienodumo testai
bartlett.test(max.imp$K01, max.imp$Time)
fligner.test(max.imp$K01, max.imp$Time)

#Anova
fit <- aov(max.imp$K01~max.imp$Time)
summary(fit)
pairwise.t.test(max.imp$K01, max.imp$Time, p.adjust="bonferroni")
TukeyHSD(fit, conf.level = 0.95)

#Means
imp.means <- aggregate(max.imp[, 2:32], list(max.imp$Time),
    na.rm=TRUE, mean)
write.table(imp.means, file = "imp.means.txt")
#Graph
plot(imp.means$K01, type ="p", ylab = "Maksimalūs parodymai",
    xlab = "Laikas", main = "Raktažodžio K01 maksimalūs parodymai")

#Adding bids
imp.means <- read.table("imp.means.txt")
share <- read.table("share.txt", dec = ",", header = TRUE)
share <- share[,2:32]
bids <- read.table("bids.txt", dec = ",", header = TRUE)
share.bids <- cbind(share,bids)
share.bids <- as.matrix(share.bids)
new.bids <- read.table("bids_new.txt", dec = ",")
new.bids <- as.matrix(new.bids)
est.share <- matrix(nrow = 24, ncol = 31)
for (i in (1:31)){
  est.share[,i] <- share.bids[,i]/(exp(1-new.bids[,i]/share.bids[,31+i]))
}
est.share[is.na(est.share)]=0
new.bids[is.na(new.bids)]=0
est.share.bids <- cbind(est.share,new.bids)

for (i in(1:31)){
  for (j in (1:24)){
```

```
    if (est.share.bids[j,i]>1){
      est.share.bids[j,i]=1
    }
  }
}
for (i in(1:31)){
  for (j in (1:24)){
    if (est.share.bids[j,i]<0){
      est.share.bids[j,i]=0
    }
  }
}
for (i in(1:31)){
  for (j in (1:24)){
    if (est.share.bids[j,31+i]==0){
      est.share.bids[j,i]=0
    }
  }
}
est.share <-est.share.bids[,1:31]
est.share <- as.matrix(est.share)
imp.means <- as.matrix(imp.means)
imp.means <- imp.means[,2:32]
imp <- matrix(nrow = 24, ncol = 31)
for (i in(1:31)){
  imp[,i]<- est.share[,i]*imp.means[,i]
}
```

9.4 Pardavimų prognozavimas - R kodas

```
library("stats")
conv <- read.table("conv.txt", dec = ",", header = TRUE)
conv$Time <- as.factor(conv$Time)

#Dispersijų vienodumo testai
bartlett.test(conv$K01_conv, conv$Time)
fligner.test(conv$K01_conv, conv$Time)

#Anova
fit <- aov(conv$K01_conv~conv$Time)
summary(fit)
conv.means <- aggregate(conv[, 2:63], list(conv$Time), na.rm=TRUE, mean)

rated <- read.table("rates.txt", dec = ",", header = TRUE)
rated$Time <- as.factor(rated$Time)

lm <- lm(K02~Position, data = rated)
summary(lm)

#Rate
rate <- matrix(nrow = 1, ncol = 31)
```

```
conv.means <- as.matrix(conv.means)
for (i in (1:31)){
  rate[i] <- mean(conv.means[,i+1]/conv.means[,i+32],na.rm = TRUE)
}
plot(rate[1:31], type ="p", ylab = "Pardavimų dažnis",
    xlab = "Raktažodis", main = "Raktažodžių pardavimų dažniai")

rate <- data.frame(rate)
colnames(rate) <- c("K01","K02","K04","K07","K08","K10",
    "K12","K14","K15","K24","K32","K36","K39","K44",
    "K54","K57","K60","K65","K68","K71","K72","K73",
    "K83","K85","K88","K91","K92","K93","K95","K97","K98")
write.table(rate, file = "rate.txt")



#Conv value
avg.value <- conv[, 64:94]
avg.value[avg.value==0] <- NA
avg.value <- as.matrix(avg.value)
value <- matrix (nrow = 1, ncol = 31)
for (i in (1:31)){
  value[i] <- mean(avg.value[,i], na.rm = TRUE)
}
value <- data.frame(value)
colnames(value) <- c("K01","K02","K04","K07","K08",
    "K10","K12","K14","K15","K24","K32","K36","K39",
    "K44","K54","K57","K60","K65","K68","K71","K72",
    "K73","K83","K85","K88","K91","K92","K93","K95",
    "K97","K98")
write.table(value, file = "values.txt")

value <- as.matrix(value)
rate <- as.matrix(rate)
max.bid <- matrix(nrow = 1, ncol = 31)
for (i in (1:31)){
  max.bid[i] <- value[i]*rate[i]
}
plot(max.bid[1:31], type ="p", ylab = "Maksimali paspaudim kaina",
    xlab = "Raktažodis", main = "Raktažodžių maksimali paspaudimo kaina")
```

9.5 Aukciono modelis - R kodas

```
library("psych")
library("stats", lib.loc="C:/Program Files/R/R-3.3.1/library")
library("foreach", lib.loc="~/R/win-library/3.3")
library("ISLR", lib.loc="~/R/win-library/3.3")
library("tree", lib.loc="~/R/win-library/3.3")
library("randomForest", lib.loc="~/R/win-library/3.3")
library("gbm", lib.loc="~/R/win-library/3.3")
```

```
library("caret", lib.loc="~/R/win-library/3.3")
library("knitr", lib.loc="~/R/win-library/3.3")
library("MASS", lib.loc="C:/Program Files/R/R-3.3.1/library")
library("glmnet", lib.loc="~/R/win-library/3.3")
results <- matrix(nrow = 100, ncol = 31)
ctr <- read.table("ctr.txt", dec = ",", header = TRUE)
ctr$Time <- as.factor(ctr$Time)
#Sampling
smp_size <- floor(0.75 * nrow(ctr))
set.seed(123)
train_ind <- sample(seq_len(nrow(ctr)), size = smp_size)
train <- ctr[train_ind, ]
#K01
train.K01 <- train[,c('Position','Time','K01')]
train.K01 <- na.omit(train.K01)
linear.model.K01 <- lm(K01 ~ Position, data = train.K01)
#K02
train.K02 <- train[,c('Position','Time','K02')]
train.K02 <- na.omit(train.K02)
boost.K02 <- gbm(K02 ~ Position+Time, data = train.K02,
                 distribution = 'gaussian',
                 n.trees = 5000,
                 interaction.depth = 4)
#K04
train.K04 <- train[,c('Position','Time','K04')]
train.K04 <- na.omit(train.K04)
boost.K04 <- gbm(K04 ~ Position+Time, data = train.K04,
                 distribution = 'gaussian',
                 n.trees = 5000,
                 interaction.depth = 4)
#K07
train.K07 <- train[,c('Position','Time','K07')]
train.K07 <- na.omit(train.K07)
rf.K07 <- randomForest(K07 ~ Position+Time, data = train.K07,
     importance = TRUE)
#K08
train.K08 <- train[,c('Position','Time','K08')]
train.K08 <- na.omit(train.K08)
linear.model.K08 <- lm(K08 ~ Position, data = train.K08)
#K10
train.K10 <- train[,c('Position','Time','K10')]
train.K10 <- na.omit(train.K10)
linear.model.K10 <- lm(K10 ~ Position, data = train.K10)
#K12
train.K12 <- train[,c('Position','Time','K12')]
train.K12 <- na.omit(train.K12)
linear.model.K12 <- lm(K12 ~ Position, data = train.K12)
#K14
train.K14 <- train[,c('Position','Time','K14')]
train.K14 <- na.omit(train.K14)
linear.model.K14 <- lm(K14 ~ Position+Time, data = train.K14)
#K15
train.K15 <- train[,c('Position','Time','K15')]
```

```
train.K15 <- na.omit(train.K15)
boost.K15 <- gbm(K15 ~ Position+Time, data = train.K15,
                 distribution = 'gaussian',
                 n.trees = 5000,
                 interaction.depth = 4)
#K24
train.K24 <- train[,c('Position','Time','K24')]
train.K24 <- na.omit(train.K24)
boost.K24 <- gbm(K24 ~ Position+Time, data = train.K24,
                 distribution = 'gaussian',
                 n.trees = 5000,
                 interaction.depth = 4)
#K32
train.K32 <- train[,c('Position','Time','K32')]
train.K32 <- na.omit(train.K32)
linear.model.K32 <- lm(K32 ~ Position, data = train.K32)
#K36
train.K36 <- train[,c('Position','Time','K36')]
train.K36 <- na.omit(train.K36)
linear.model.K36 <- lm(K36 ~ Position, data = train.K36)
#K39
train.K39 <- train[,c('Position','Time','K39')]
train.K39 <- na.omit(train.K39)
linear.model.K39 <- lm(K39 ~ Position, data = train.K39)
#K44
train.K44 <- train[,c('Position','Time','K44')]
train.K44 <- na.omit(train.K44)
linear.model.K44 <- lm(K44 ~ Position, data = train.K44)
#K54
train.K54 <- train[,c('Position','Time','K54')]
train.K54 <- na.omit(train.K54)
linear.model.K54 <- lm(K54 ~ Position, data = train.K54)
#K57
train.K57 <- train[,c('Position','Time','K57')]
train.K57 <- na.omit(train.K57)
linear.model.K57 <- lm(K57 ~ Position, data = train.K57)
#K60
train.K60 <- train[,c('Position','Time','K60')]
train.K60 <- na.omit(train.K60)
linear.model.K60 <- lm(K60 ~ Position, data = train.K60)
#K65
train.K65 <- train[,c('Position','Time','K65')]
train.K65 <- na.omit(train.K65)
linear.model.K65 <- lm(K65 ~ Position, data = train.K65)
#K68
train.K68 <- train[,c('Position','Time','K68')]
train.K68 <- na.omit(train.K68)
rf.K68 <- randomForest(K68 ~ Position+Time, data = train.K68,
     importance = TRUE)
#K71
train.K71 <- train[,c('Position','Time','K71')]
train.K71 <- na.omit(train.K71)
rf.K71 <- randomForest(K71 ~ Position+Time, data = train.K71,
```

```
     importance = TRUE)
#K72
train.K72 <- train[,c('Position','Time','K72')]
train.K72 <- na.omit(train.K72)
boost.K72 <- gbm(K72 ~ Position+Time, data = train.K72,
                 distribution = 'gaussian',
                 n.trees = 5000,
                 interaction.depth = 4)
#K73
train.K73 <- train[,c('Position','Time','K73')]
train.K73 <- na.omit(train.K73)
boost.K73 <- gbm(K73 ~ Position+Time, data = train.K73,
                 distribution = 'gaussian',
                 n.trees = 5000,
                 interaction.depth = 4)
#K83
train.K83 <- train[,c('Position','Time','K83')]
train.K83 <- na.omit(train.K83)
linear.model.K83 <- lm(K83 ~ Position, data = train.K83)
#K85
train.K85 <- train[,c('Position','Time','K85')]
train.K85 <- na.omit(train.K85)
x <- train.K85[,c('Position','Time')]
x$Time <- as.integer(x$Time)
x <- as.matrix(x)
y <- train.K85[,c('K85')]
ridge.K85 <- glmnet(x,y, alpha = 0)
#K88
train.K88 <- train[,c('Position','Time','K88')]
train.K88 <- na.omit(train.K88)
boost.K88 <- gbm(K88 ~ Position+Time, data = train.K88,
                 distribution = 'gaussian',
                 n.trees = 5000,
                 interaction.depth = 4)
#K91
train.K91 <- train[,c('Position','Time','K91')]
train.K91 <- na.omit(train.K91)
linear.model.K91 <- lm(K91 ~ Position, data = train.K91)
#K92
train.K92 <- train[,c('Position','Time','K92')]
train.K92 <- na.omit(train.K92)
linear.model.K92 <- lm(K92 ~ Position, data = train.K92)
#K93
train.K93 <- train[,c('Position','Time','K93')]
train.K93 <- na.omit(train.K93)
linear.model.K93 <- lm(K93 ~ Position, data = train.K93)
#K95
train.K95 <- train[,c('Position','Time','K95')]
train.K95 <- na.omit(train.K95)
rf.K95 <- randomForest(K95 ~ Position+Time, data = train.K95,
     importance = TRUE)
#K97
train.K97 <- train[,c('Position','Time','K97')]
```

```
train.K97 <- na.omit(train.K97)
linear.model.K97 <- lm(K97 ~ Position, data = train.K97)
#K98
train.K98 <- train[,c('Position','Time','K98')]
train.K98 <- na.omit(train.K98)
linear.model.K98 <- lm(K98 ~ Position, data = train.K98)

for (k in (1:100)){
  new.bids <- matrix(k/100,nrow = 24, ncol = 31)
  #Estimating positions
  positions <- read.table("positions.txt", dec = ",", header = TRUE)
  positions$Time <- as.integer(positions$Time)
  bids <- read.table("bids.txt", dec = ",", header = TRUE)

  pos.means <- aggregate(positions[, 2:32], list(positions$Time),
     na.rm=TRUE, mean)
  pos.bids <- cbind(pos.means,bids)
  pos.bids <- pos.bids[,2:63]
  pos.bids <- as.matrix(pos.bids)
  est.pos <- matrix(nrow = 24, ncol = 31)
  for (i in (1:31)){
    est.pos[,(i)] <- (pos.bids[,i]*(exp(1-new.bids[,i]/pos.bids[,i+31])))
  }
  new.bids[is.na(new.bids)]=0
  est.pos[is.na(est.pos)]=0
  est.pos.bids <- cbind(est.pos, new.bids)
  for (i in (1:31)){
    for (j in (1:24)){
      if (est.pos.bids[j,i] > 5){
        est.pos.bids[j,i]=0
      }
    }
  }
  for (i in (1:31)){
    for (j in (1:24)){
      if (est.pos.bids[j,i] < 1){
        if (est.pos.bids[j,i] > 1){
          est.pos.bids[j,i]=1
        }
      }
    }
  }
  for (i in (1:31)){
    for (j in (1:24)){
      if (est.pos.bids[j,31+i]==0){
        est.pos.bids[j,i]=0
      }
    }
  }
  est.pos <- est.pos.bids[,1:31]
  Time <- c(0:23)
  est.pos <- cbind(Time, est.pos)
  est.pos <- as.data.frame(est.pos)
```

```
colnames(est.pos) <- c("Time", "K01","K02","K04","K07","K08",
  "K10","K12","K14","K15","K24","K32","K36","K39",
  "K44","K54","K57","K60","K65","K68","K71","K72",
  "K73","K83","K85","K88","K91","K92","K93","K95","K97","K98")
est.pos$Time <- as.factor(est.pos$Time)
test.K01 <- est.pos[,c('Time', 'K01')]
colnames(test.K01) <- c("Time", "Position")
test.K02 <- est.pos[,c('Time', 'K02')]
colnames(test.K02) <- c("Time", "Position")
test.K04 <- est.pos[,c('Time', 'K04')]
colnames(test.K04) <- c("Time", "Position")
test.K07 <- est.pos[,c('Time', 'K07')]
colnames(test.K07) <- c("Time", "Position")
test.K08 <- est.pos[,c('Time', 'K08')]
colnames(test.K08) <- c("Time", "Position")
test.K10 <- est.pos[,c('Time', 'K10')]
colnames(test.K10) <- c("Time", "Position")
test.K12 <- est.pos[,c('Time', 'K12')]
colnames(test.K12) <- c("Time", "Position")
test.K14 <- est.pos[,c('Time', 'K14')]
colnames(test.K14) <- c("Time", "Position")
test.K15 <- est.pos[,c('Time', 'K15')]
colnames(test.K15) <- c("Time", "Position")
test.K24 <- est.pos[,c('Time', 'K24')]
colnames(test.K24) <- c("Time", "Position")
test.K32 <- est.pos[,c('Time', 'K32')]
colnames(test.K32) <- c("Time", "Position")
test.K36 <- est.pos[,c('Time', 'K36')]
colnames(test.K36) <- c("Time", "Position")
test.K39 <- est.pos[,c('Time', 'K39')]
colnames(test.K39) <- c("Time", "Position")
test.K44 <- est.pos[,c('Time', 'K44')]
colnames(test.K44) <- c("Time", "Position")
test.K54 <- est.pos[,c('Time', 'K54')]
colnames(test.K54) <- c("Time", "Position")
test.K57 <- est.pos[,c('Time', 'K57')]
colnames(test.K57) <- c("Time", "Position")
test.K60 <- est.pos[,c('Time', 'K60')]
colnames(test.K60) <- c("Time", "Position")
test.K65 <- est.pos[,c('Time', 'K65')]
colnames(test.K65) <- c("Time", "Position")
test.K68 <- est.pos[,c('Time', 'K68')]
colnames(test.K68) <- c("Time", "Position")
test.K71 <- est.pos[,c('Time', 'K71')]
colnames(test.K71) <- c("Time", "Position")
test.K72 <- est.pos[,c('Time', 'K72')]
colnames(test.K72) <- c("Time", "Position")
test.K73 <- est.pos[,c('Time', 'K73')]
colnames(test.K73) <- c("Time", "Position")
test.K83 <- est.pos[,c('Time', 'K83')]
colnames(test.K83) <- c("Time", "Position")
test.K85 <- est.pos[,c('Time', 'K85')]
colnames(test.K85) <- c("Time", "Position")
```

```
test.K88 <- est.pos[,c('Time', 'K88')]
colnames(test.K88) <- c("Time", "Position")
test.K91 <- est.pos[,c('Time', 'K91')]
colnames(test.K91) <- c("Time", "Position")
test.K92 <- est.pos[,c('Time', 'K92')]
colnames(test.K92) <- c("Time", "Position")
test.K93 <- est.pos[,c('Time', 'K93')]
colnames(test.K93) <- c("Time", "Position")
test.K95 <- est.pos[,c('Time', 'K95')]
colnames(test.K95) <- c("Time", "Position")
test.K97 <- est.pos[,c('Time', 'K97')]
colnames(test.K97) <- c("Time", "Position")
test.K98 <- est.pos[,c('Time', 'K98')]
colnames(test.K98) <- c("Time", "Position")
#Estimating CTR
K01_ctr_pred <- predict(linear.model.K01, newdata = test.K01)
K01_ctr_pred <-as.matrix(K01_ctr_pred)
K02_ctr_pred <- predict(boost.K02,test.K02, n.trees = 5000)
K02_ctr_pred <-as.matrix(K02_ctr_pred)
K04_ctr_pred <- predict(boost.K04,test.K04, n.trees = 5000)
K04_ctr_pred <-as.matrix(K04_ctr_pred)
K07_ctr_pred <- predict(rf.K07, test.K07)
K07_ctr_pred <-as.matrix(K07_ctr_pred)
K08_ctr_pred <- predict(linear.model.K08, newdata = test.K08)
K08_ctr_pred <-as.matrix(K08_ctr_pred)
K10_ctr_pred <- predict(linear.model.K10, newdata = test.K10)
K10_ctr_pred <-as.matrix(K10_ctr_pred)
K12_ctr_pred <- predict(linear.model.K12, newdata = test.K12)
K12_ctr_pred <-as.matrix(K12_ctr_pred)
K14_ctr_pred <- predict(linear.model.K14, newdata = test.K14)
K14_ctr_pred <-as.matrix(K14_ctr_pred)
K15_ctr_pred <- predict(boost.K15,test.K15, n.trees = 5000)
K15_ctr_pred <-as.matrix(K15_ctr_pred)
K24_ctr_pred <- predict(boost.K24,test.K24, n.trees = 5000)
K24_ctr_pred <-as.matrix(K24_ctr_pred)
K32_ctr_pred <- predict(linear.model.K32, newdata = test.K32)
K32_ctr_pred <-as.matrix(K32_ctr_pred)
K36_ctr_pred <- predict(linear.model.K36, newdata = test.K36)
K36_ctr_pred <-as.matrix(K36_ctr_pred)
K39_ctr_pred <- predict(linear.model.K39, newdata = test.K39)
K39_ctr_pred <-as.matrix(K39_ctr_pred)
K44_ctr_pred <- predict(linear.model.K44, newdata = test.K44)
K44_ctr_pred <-as.matrix(K44_ctr_pred)
K54_ctr_pred <- predict(linear.model.K54, newdata = test.K54)
K54_ctr_pred <-as.matrix(K54_ctr_pred)
K57_ctr_pred <- predict(linear.model.K57, newdata = test.K57)
K57_ctr_pred <-as.matrix(K57_ctr_pred)
K60_ctr_pred <- predict(linear.model.K60, newdata = test.K60)
K60_ctr_pred <-as.matrix(K60_ctr_pred)
K65_ctr_pred <- predict(linear.model.K65, newdata = test.K65)
K65_ctr_pred <-as.matrix(K65_ctr_pred)
K68_ctr_pred <- predict(rf.K68, test.K68)
K68_ctr_pred <-as.matrix(K68_ctr_pred)
```

```
K71_ctr_pred <- predict(rf.K71, test.K71)
K71_ctr_pred <-as.matrix(K71_ctr_pred)
K72_ctr_pred <- predict(boost.K72,test.K72, n.trees = 5000)
K72_ctr_pred <-as.matrix(K72_ctr_pred)
K73_ctr_pred <- predict(boost.K73,test.K73, n.trees = 5000)
K73_ctr_pred <-as.matrix(K73_ctr_pred)
K83_ctr_pred <- predict(linear.model.K83, newdata = test.K83)
K83_ctr_pred <-as.matrix(K83_ctr_pred)
test.K85$Time <- as.integer(test.K85$Time)
test.K85 <- as.matrix(test.K85)
K85_ctr_pred <- predict(ridge.K85, newx = test.K85)
K85_ctr_pred <- K85_ctr_pred[,1]
K88_ctr_pred <- predict(boost.K88,test.K88, n.trees = 5000)
K88_ctr_pred <-as.matrix(K88_ctr_pred)
K91_ctr_pred <- predict(linear.model.K91, newdata = test.K91)
K91_ctr_pred <-as.matrix(K91_ctr_pred)
K92_ctr_pred <- predict(linear.model.K92, newdata = test.K92)
K92_ctr_pred <-as.matrix(K92_ctr_pred)
K93_ctr_pred <- predict(linear.model.K93, newdata = test.K93)
K93_ctr_pred <-as.matrix(K93_ctr_pred)
K95_ctr_pred <- predict(rf.K95, test.K95)
K95_ctr_pred <-as.matrix(K95_ctr_pred)
K97_ctr_pred <- predict(linear.model.K97, newdata = test.K97)
K97_ctr_pred <-as.matrix(K97_ctr_pred)
K98_ctr_pred <- predict(linear.model.K98, newdata = test.K98)
K98_ctr_pred <-as.matrix(K98_ctr_pred)

K01_ctr_pred <-as.numeric(K01_ctr_pred)
K02_ctr_pred <-as.numeric(K02_ctr_pred)
K04_ctr_pred <-as.numeric(K04_ctr_pred)
K07_ctr_pred <-as.numeric(K07_ctr_pred)
K08_ctr_pred <-as.numeric(K08_ctr_pred)
K10_ctr_pred <-as.numeric(K10_ctr_pred)
K12_ctr_pred <-as.numeric(K12_ctr_pred)
K14_ctr_pred <-as.numeric(K14_ctr_pred)
K15_ctr_pred <-as.numeric(K15_ctr_pred)
K24_ctr_pred <-as.numeric(K24_ctr_pred)
K32_ctr_pred <-as.numeric(K32_ctr_pred)
K36_ctr_pred <-as.numeric(K36_ctr_pred)
K39_ctr_pred <-as.numeric(K39_ctr_pred)
K44_ctr_pred <-as.numeric(K44_ctr_pred)
K54_ctr_pred <-as.numeric(K54_ctr_pred)
K57_ctr_pred <-as.numeric(K57_ctr_pred)
K60_ctr_pred <-as.numeric(K60_ctr_pred)
K65_ctr_pred <-as.numeric(K65_ctr_pred)
K68_ctr_pred <-as.numeric(K68_ctr_pred)
K71_ctr_pred <-as.numeric(K71_ctr_pred)
K72_ctr_pred <-as.numeric(K72_ctr_pred)
K73_ctr_pred <-as.numeric(K73_ctr_pred)
K83_ctr_pred <-as.numeric(K83_ctr_pred)
K85_ctr_pred <-as.numeric(K85_ctr_pred)
K88_ctr_pred <-as.numeric(K88_ctr_pred)
K91_ctr_pred <-as.numeric(K91_ctr_pred)
```

```
K92_ctr_pred <-as.numeric(K92_ctr_pred)
K93_ctr_pred <-as.numeric(K93_ctr_pred)
K95_ctr_pred <-as.numeric(K95_ctr_pred)
K97_ctr_pred <-as.numeric(K97_ctr_pred)
K98_ctr_pred <-as.numeric(K98_ctr_pred)



ctr_pred <- cbind(K01_ctr_pred,K02_ctr_pred,K04_ctr_pred,
  K07_ctr_pred,K08_ctr_pred,K10_ctr_pred,K12_ctr_pred,
  K14_ctr_pred,K15_ctr_pred,K24_ctr_pred,K32_ctr_pred,
  K36_ctr_pred,K39_ctr_pred,K44_ctr_pred,K54_ctr_pred,
  K57_ctr_pred,K60_ctr_pred,K65_ctr_pred,K68_ctr_pred,
  K71_ctr_pred,K72_ctr_pred,K73_ctr_pred,K83_ctr_pred,
  K85_ctr_pred,K88_ctr_pred,K91_ctr_pred,K92_ctr_pred,
  K93_ctr_pred,K95_ctr_pred,K97_ctr_pred,K98_ctr_pred)
ctr_pred <- cbind(ctr_pred, new.bids)
for (i in (1:31)){
  for (j in (1:24)){
    if (ctr_pred[j,31+i]<=0){
      ctr_pred[j,i]=0
    }
  }
}
for (i in (1:31)){
  for (j in (1:24)){
    if (ctr_pred[j,i]< 0){
      ctr_pred[j,i]=0
    }
  }
}
ctr_pred <- ctr_pred[,1:31]
est.pos$Time <- as.numeric(est.pos$Time)
est.pos <- as.matrix(est.pos)
est.pos <- est.pos[,2:32]
ctr_pred <-cbind(ctr_pred,est.pos)

for (i in (1:31)){
  for (j in (1:24)){
    if (ctr_pred[j,31+i]==0){
      ctr_pred[j,i]=0
    }
  }
}
ctr_pred <- ctr_pred[,1:31]
#Estimating Impressions
max.imp <- read.table("max.imp.txt", dec = ",", header = TRUE)
max.imp$Time <- as.factor(max.imp$Time)
imp.means <- aggregate(max.imp[, 2:32], list(max.imp$Time)
  , na.rm=TRUE, mean)
imp.means <- read.table("imp.means.txt")
share <- read.table("share.txt", dec = ",", header = TRUE)
share <- share[,2:32]
```

```
bids <- read.table("bids.txt", dec = ",", header = TRUE)
share.bids <- cbind(share,bids)
share.bids <- as.matrix(share.bids)
est.share <- matrix(nrow = 24, ncol = 31)
for (i in (1:31)){
  est.share[,i] <- share.bids[,i]/(exp(1-new.bids[,i]/share.bids[,31+i]))
}
est.share[is.na(est.share)]=0
new.bids[is.na(new.bids)]=0
est.share.bids <- cbind(est.share,new.bids)

for (i in(1:31)){
  for (j in (1:24)){
    if (est.share.bids[j,i]>1){
      est.share.bids[j,i]=1
    }
  }
}
for (i in(1:31)){
  for (j in (1:24)){
    if (est.share.bids[j,i]<0){
      est.share.bids[j,i]=0
    }
  }
}
for (i in(1:31)){
  for (j in (1:24)){
    if (est.share.bids[j,31+i]==0){
      est.share.bids[j,i]=0
    }
  }
}
est.share <-est.share.bids[,1:31]
est.share <- as.matrix(est.share)
imp.means <- as.matrix(imp.means)
imp.means <- imp.means[,2:32]
imp <- matrix(nrow = 24, ncol = 31)
for (i in(1:31)){
  imp[,i]<- est.share[,i]*imp.means[,i]
}
imp[is.na(imp)] <- 0
#Estimating Clicks
clicks <- matrix(nrow = 24, ncol = 31)
for (i in (1:31)){
  for (j in (1:24)){
    clicks[j,i] <- imp[j,i]*ctr_pred[j,i]
  }
}

cost <- matrix(nrow = 24, ncol = 31)
for (i in (1:31)){
  for (j in (1:24)){
    cost[j,i] <- new.bids[j,i]*clicks[j,i]
```

```
    }
  }
  #Estimating conversions
  values <- read.table("values.txt")
  values <- as.matrix(values)
  rate <- read.table("rate.txt")
  rate <- as.matrix(rate)
  revenue <- matrix(nrow = 24, ncol = 31)
  for (i in (1:31)){
    for (j in (1:24)){
      revenue[j,i] <- clicks[j,i]*rate[i]*values[i]
    }
  }
  mean.profit <- matrix(nrow = 1, ncol = 31)
  profit <- revenue - cost
  for ( i in (1:31)){
    mean.profit[i] <- mean(profit[,i])
  }
  #Final results
  results[k,] <- mean.profit
}
write.table(results, file = "auction_model.txt")
auction <- read.table("auction_model.txt")
auction <- as.matrix(auction)
bid <- matrix(nrow = 100, ncol = 1)
for (i in (1:100)){
  bid[i,1] <- i/100
}
auction <- cbind(bid, auction)
optimal.bids <- matrix(nrow = 1, ncol = 31)
j=0
for (i in (1:31)){
  j=which.max(auction[,i+1])
  if (max(auction[,i+1])>0){
    optimal.bids[i] <- auction[j,1]
  } else {
    optimal.bids[i] <- 0
  }
}
auction <- data.frame(auction)
colnames(auction) <-c("Bid","K01","K02","K04","K07","K08","K10",
    "K12","K14","K15","K24","K32","K36","K39","K44","K54","K57"
    ,"K60","K65","K68","K71","K72","K73","K83","K85","K88","K91"
    ,"K92","K93","K95","K97","K98")
optimal.bids <- data.frame(optimal.bids)
colnames(optimal.bids) <-c("K01","K02","K04","K07","K08","K10"
    ,"K12","K14","K15","K24","K32","K36","K39","K44","K54",
    "K57","K60","K65","K68","K71","K72","K73","K83","K85",
    "K88","K91","K92","K93","K95","K97","K98")

write.table(optimal.bids, file = "optimal_bids.txt")
```