KAUNO TECHNOLOGIJOS UNIVERSITETAS
MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

Arnas Dundulis

TAKSI MARŠRUTŲ DIDELIO DUOMENŲ KIEKIO APDOROJIMAS IR ANALIZĖ

Baigiamasis magistro projektas

Vadovai
dr. Mindaugas Kavaliauskas
prof. dr. Rimantas Gatautis

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS
MATEMATIKOS IR GAMTOS MOKSLŲ. FAKULTETAS

TAKSI MARŠRUTŲ DIDELIO DUOMENŲ KIEKIO APDOROJIMAS IR ANALIZĖ

Baigiamasis magistro projektas

Didžiųjų verslo duomenų analitika (kodas 621G12002)

Vadovai
dr. Mindaugas Kavaliauskas

prof. dr. Rimantas Gatautis

Recenzentai
prof. dr. Jūratė Banytė

lekt. dr. Evaldas Vaičiukynas

Projektą atliko
Arnas Dundulis

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Matematikos ir gamtos mokslų fakultetas
(Fakultetas)
Arnas Dundulis
(Studento vardas, pavardė)
Didžiųjų verslo duomenų analitika, 621G12002
(Studijų programos pavadinimas, kodas)

„Taksi maršrutų didelio duomenų kiekio apdorojimas ir analizė“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 17 m. birželio 2 d.
Kaunas

Patvirtinu, kad mano, **Arnas Dundulio**, baigiamasis projektas tema „Taksi maršrutų didelio duomenų kiekio apdorojimas ir analizė“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka) *(parašas)*

Dundulis, Arnas. *Taxi trip big data processing and analysis* .: *Master's* thesis in Business Big Data Analytics / supervisor dr. Mindaugas Kavaliauskas, prof. dr. Rimantas Gatautis. The Faculty of Mathematics and Natural Sciences, Kaunas University of Technology.

Research area and field: Physical sciences, mathematics (01 P)

Key words: taxi trip, big data, clustering, segmentation.

Kaunas, 2017. 42 p.

SUMMARY

Analyzing taksi trip data one of the criteria that allows drivers to increase their income is the whether the rider leaves a tip for the trip. Since this is not mandatory the ability to determine if a tip will be given after a ride is sought after. One of the ways to increase the accuracy of such a prediction is data grouping by some criteria. This article will explore the social aspects associated with gratuity and describe the ways in which taxi trip data can be clustered and classified. K-means method will be used for clusterization and classification accuracy will be compared between decision tree, random forest and neural network methods.

Dundulis, Arnas. Taksi maršrutų didelio duomenų kiekio apdorojimas ir analizė. Magistro baigiamasis projektas / vadovai: dr. Mindaugas Kavaliauskas, prof. dr. Rimantas Gatautis; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas.

Mokslo kryptis ir sritis: Fiziniai mokslai, matematika (01 P)

Reikšminiai žodžiai: taksi kelionės, didieji duomenys, klasterizavimas, klasifikavimas.

Kaunas, 2017. 42 p.

SANTRAUKA

Analizuojant taksi maršrutų duomenis, vienas iš kriterijų, leidžiančių vairuotojui padidinti pajamas yra tai, ar keleivis už kelionę davė arbatpinigių. Kadangi tai nėra privaloma, norima gebėti prognozuoti, ar bus duodama arbatpinigių už tam tikrą kelionę.Vienas iš būdų pagerinti tokio prognozavimo tikslumą – duomenų grupavimas pagal tam tikrus kriterijus. Šiame darbe apžvelgiama socialiniai aspektai susiję su arbatpinigiais, aprašomi būdai, kaip galima klasterizuoti ir klasifikuoti taksi keliones. Panaudojamas k-vidurkių metodas klasterizavimui, palyginama klasifikavimo tikslumas tarp sprendimų medžių, atsitiktinių miškų ir neuroninių tinklų.

Turinys

ĮVADAS

Šiuolaikinėms technologijoms fiksuojant vis daugiau mūsų gyvenimo aspektų atsiranda galimybė šiuos duomenis paversti į įžvalgas, leidžiančias pagerinti žmonių gyvenimus. Niujorko taksi ir limuzinų komisija nuo 2009 metų kaupia duomenis apie visas mieste atliktas keliones. Turint šiuos duomenis galima sukurti matematinius modelius, gebančius prognozuoti, po kurių kelionių klientai paliks vairuotojui arbatpinigių ir nustatyti, kuriose geografinėse vietose didžiausia tikimybė tiksliai nustatyti šį įvykį. Šis gebėjimas atneštų daug naudos taksi vairuotojams, nes suteiktų galimybę padidinti savo gaunamas pajamas. Vairuotojų pajamos yra priklausomos nuo įvairių aspektų. Vienas iš būdų kaip jas būtų galima padidinti – rinktis klientus, kurie mokėtų arbatpinigus. Kadangi arbatpinigiai dažnu atveju proporcingi paslaugų kainai, gebėjimas žinoti ar klientas mokės arbatpinigius žinant kelionės kainą leistų prognozuoti šios kelionės pelningumą. Šiame darbe naudojami nuo 2009 metų sausio iki 2016 liepos sukaupti Niujorko miesto taksi kelionių duomenys. Su jais atliekamas kategorizavimo uždavinys, su kurio rezultatais sudaromi klasifikavimo modeliai pagal sprendimų medžių, atsitiktinio miško ir daugiasluoksnio perceptrono metodus su skirtingais metodų parametrais. Modelių rezultatai palyginami tarpusavyje nustatant geriausią klasifikatorių pagal plotą po sprendimus priimančiojo ypatybių (angl. *receiver operating characteristics - ROC*) kreive.

Darbo tikslas – sudaryti klasifikavimo modelius rinkos segmentams.

Darbo uždaviniai:

1. Atlikti literatūros analizę, atskleisti arbatpinigių svarbą taksi vairuotojams.
2. Surinkti, susisteminti ir parengti duomenis analizei.
3. Atlikti Niujorko taksi paslaugų rinkos segmentavimą.
4. Pritaikyti klasifikavimo metodus arbatpinigių prognozavimui ir palyginti sudarytų modelių tikslumą.

1. LITERATŪROS APŽVALGA

Šioje darbo dalyje analizuojamos arbatpinigių suteikimo priežastys, veiksniai lemiantys vartotojų naudojimąsi taksi paslaugomis bei aptariami matematiniai duomenų tyrybos metodai rinkos segmentavimo ir prognozavimo uždaviniams spręsti.

1.1 Taksi vairuotojų pajamų išskirtis

Tirdami Šanchajaus taksi kelionių duomenis Yuanas Gao ir kiti nustatė, kad yra didelė pajamų išskirtis tarp daugiausiai ir mažiausiai uždirbančių taksi vairuotojų. Senesni tyrimai naudojo informaciją, surinktą iš vairuotojų pateikiant jiems klausimus, šiuolaikiniai duomenų rinkimo būdai leidžia turėti tikslią geografinę informaciją apie taksi keliones. Vairuotojai gali naudotis skirtingomis strategijomis, kaip pritraukti klientus, kokius maršrutus naudoti. Šios strategijos, dar žinomos kaip mobilumo intelektas (angl. mobility intelligence) nulemia vairuotojo gaunamas pajamas. Viena iš tyrimo prielaidų yra tai, kad yra koreliacija tarp vairuotojo pajamų ir regionų, kuriuose jis dirba.

Ankstesniuose darbuose nustatyta, kad taksi vairuotojų darbo pasiūlos elastingumas yra neigiamas, rodantis, kad vairuotojai anksčiau baigia darbą kuomet valandinis uždarbis yra aukštas ir dirba ilgiau kuomet jis yra žemas.

Iš tyrimo duomenų matyti, kad darbuotojų uždarbis yra gana stabilus tarp savaičių, egzistuoja koreliacijos tarp pajamų ir kelionių požymių kaip jų skaičius. Taipogi matoma didelis skirtumas tarp to kiek uždirba didžiausias ir mažiausias pajamas gaunantys vairuotojai. (Y., et al., 2012)

1.2 Arbatpinigių ekonominė vertė

Žmonės mokantys už įvairias paslaugas dažnai pasirenka kartu su įprasta paslaugų kaina sumokėti ir savarankišką mokestį. Šis mokestis dažnai vadinamas arbatpinigiais ir yra skirtas atsidėkoti darbuotojui už suteiktas paslaugas. Toks dėkingumo parodymas labiausiai paplitęs Jungtinėse Amerikos Valstijose (JAV), bet praktikuojamas ir daugumoje kitų pasaulio valstybių. Tokio mokesčio dydis priklauso nuo paslaugų rūšies, jų suteikimo kokybės, socialinių normų, nusistovėjusių įpročių ir kitų sąlygų. Įstaigos, kuriose dirba dauguma arbatpinigius gaunančių darbuotojų, gali didinti arba mažinti šio mokesčio populiarumą. Skatinimas gali pasireikšti įvairiais ženklais, įspėjančiais, kad darbuotojai tikisi arbatpinigių arba būtų pamaloninti juos gavę. Taip pat galima ir sumažinti ar išvis pašalinti tokį mokestį įskaičiuojant jį į paslaugų kainą, ar įspėjant klientus, kad arbatpinigiai nėra privalomi. Įtraukdami tokį mokestį į bendrą paslaugų kainą įmonės gali pakeisti siūlomų paslaugų patrauklumą klientams, darbuotojų gaunamą atlyginimą, bei kitus

ekonominius aspektus. Pagal JAV darbuotojų statistikos biurą (angl. *bureau of labor statistics*) arbatpinigius gaunančių profesijų atstovai gauna panašaus dydžio atlyginimus kaip ir tų profesijų, kurios negauna arbatpinigių, atstovai. Kadangi arbatpinigiai nėra valstybės reguliuojami jie dažniausiai būna nedeklaruojami, todėl tikrovėje profesijos gaunančios arbatpinigių uždirba daugiau lyginant su kitomis, panašų darbą atliekančiomis, profesijomis. (Lynn & Withiam, 2008)

Dėl pajamų nelygybės iškylą įvairių klausimų, koks turėtų būti atlyginimas darbuotojams, kurie gauna arbatpinigius. Kadangi milijonų darbuotojų pajamos tiesiogiai susiję su šiuo mokesčiu, tai turi įtakos darbo ekonomikai (angl. *labor economics*). Vienas iš pagrindinių klausimų yra ar darbuotojai, gaunantys arbatpinigius, turėtų gauti tokį patį minimalų atlyginimą kaip ir profesijos negaunančios arbatpinigių. JAV darbuotojams gaunantiems arbatpinigių gali būti taikomas mažesnis minimalus atlyginimas. Tačiau vienodo minimalaus atlyginimo taikymas gali turėti ir neigiamų aspektų. Izraelyje teismo sprendimu tapo privaloma kavinių padavėjams gauti minimalų atlyginimą. Prieš šį sprendimą įprastai padavėjai negaudavo jokio užmokesčio iš kavinės ir visos jų pajamos būdavo gaunamos iš arbatpinigių. Po teismo sprendimo dauguma restoranų pradėjo įtraukti arbatpinigius į paslaugų kainą ir už gautas papildomas pajamas mokėti padavėjams algas. Šių pokyčių rezultatas – sumažėjusios padavėjų gaunamos pajamos. (Azar O. H., 2003)

Daugumoje pasaulio šalių yra priimta už paslaugas atsidėkoti ne tik užmokesčiu už paslaugas, bet ir suteikti dovaną, dar vadinamą arbatpinigiais. Į profesijas, kurios dažnai gauna šiuos arbatpinigius įeina barmenai, viešbučių kambarinės, maisto išvežiotojai, kavinių muzikantai, taksi vairuotojai (Star, 1988). Šie mainai įvardijami kaip dovanos todėl, kad toks mokestis nėra privalomas. Nepaisant to, šių dovanų dydis dažnai prilygsta didelei daliai prekių ar paslaugų, už kurias ir duodama ši dovana, kainos (Lynn, Jabbour, & Kim, 2012). Vien JAV maisto rinkoje tokios dovanos sudaro apie 47 milijardus dolerių per metus (Azar, 2011). Toks elgesys prieštarauja pirkėjų dažnam polinkiui siekti sumažinti išleidžiamus pinigus prekėms ar paslaugoms (Azar, 2008).

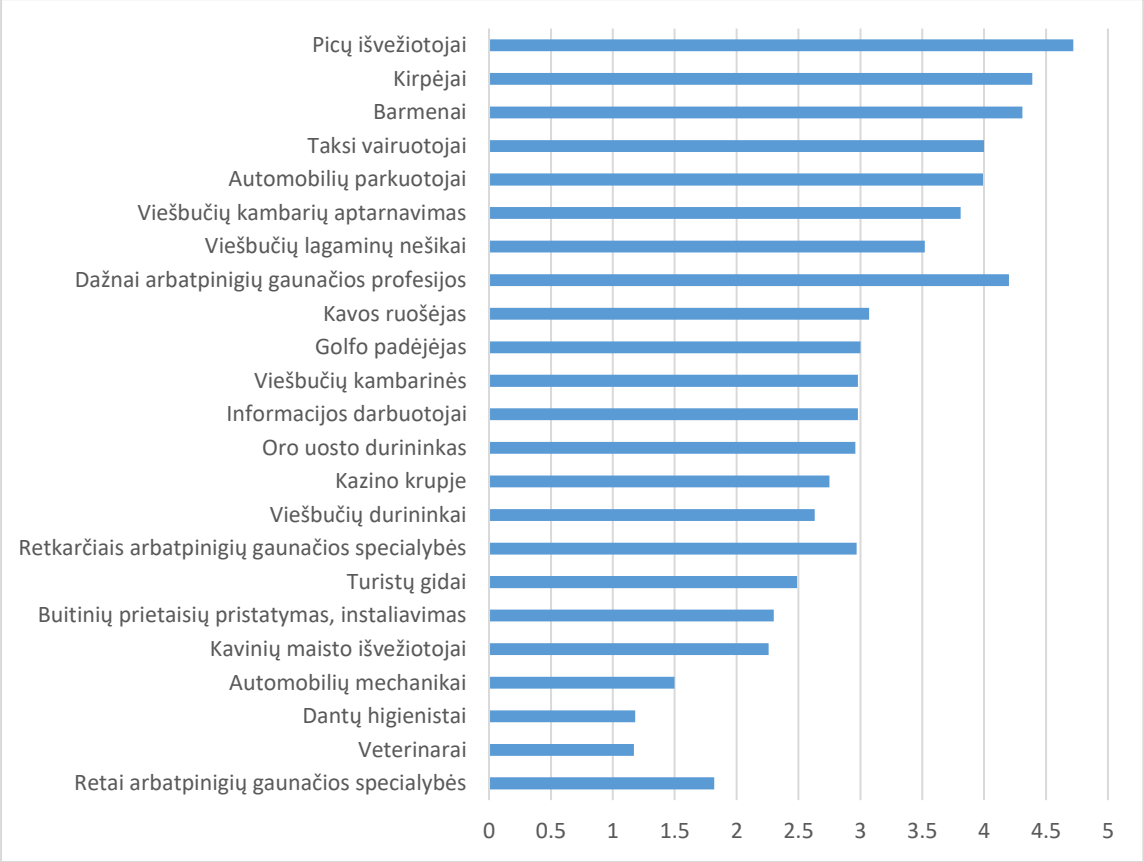
1.3 Arbatpinigių palikimo elgsenos priežastys

Siekiant nustatyti priežastis, kodėl paliekami arbatpinigiai, atlikta nemažai tyrimų (Azar, 2011; Becker, Bradley, & Zantow, 2012; Lynn, 2009, 2015b, Whalen, Douglas, & O'Neill, 2014). Vienos dažniausių priežasčių – siekis gauti geresnį aptarnavimą kito apsilankymo metu, noras įvertinti gerą aptarnavimą, sulaukti aplinkinių įvertinimo. Arbatpinigių davimo dažnio ir dydžio analizė parodo, kad toks atsidėkojimas priklauso nuo teikiamos paslaugos, paslaugos suteikimo kokybės ir kitų faktorių. Bekeris su kolegomis (2012) nustatė, kad stiprūs ateities aptarnavimo ir atlygio motyvai susiję su didesniais arbatpinigiais kai aptarnavimas būdavo geras, bet ne kai aptarnavimas buvo

blogas. Stipresnis atsidėkojimo motyvai asocijuojasi su mažasniais arbatpinigiais kuomet būna blogas aptarnavimas. Taip pat nustatyta, kad skirtingi motyvai nevienodai asocijuojasi su skirtingomis profesijomis (Lynn, 2015b). Geresnio ateities aptarnavimo motyvai labiau įtakos didesnius arbatpinigius barmenams, taksistam ar picų išvežiotojams, bet ne viešbučių darbuotojams.

Lynnas (2015a) iškėlė keletą skirtingų hipotezių, kodėl skiriasi motyvaciją duoti arbatpinigių tarp skirtingų profesijų. Jis taip pat pažymėjo, kad kuo dažniau tam tikros profesijos atstovai gauna arbatpinigių, tuo labiau jie linkę nemėgti arbatpinigių nepaliekančių klientų ir juos praščiau aptarnauti. Klientai nenorėdami gauti blogo aptarnavimo ateityje yra labiau linkę palikti arbatpinigių. Tuo pačiu dažnas aplinkinių arbatpinigių palikimas sukuria socialinį spaudimą taip pat atsidėkoti už paslaugas. Tačiau galioja ir išimčių, kuriose klientai paliekantys arbatpinigių gauna darbuotojo palankumą net ir jei ta profesija dažnai arbatpinigių negauna. Keletas neapžvelgtų aspektų Lynno tyrime yra skatinimas atsidėkoti už paslaugas iš altruistiškų paskatų, matant, kad kiti žmonės palieka arbatpinigių parodo, kad darbuotui galbūt reikalinga pagalba, parodo, kad darbuotojas priima arbatpinigius. Kitas faktorius tai, kad išoriniai motyvai palikti arbatpinigių slopina vidinius motyvus (Frey & Jegen, 2001; James, 2005). Taipogi aplinkinių skatinimas palikti arbatpinigių gali padidinti konkurenciją norint gauti geresnį aptarnavimą, kas pakeltų arbatpinigų dydį ir pradėtų neigiamai veikti motyvaciją palikti arbatpinigių (Lynn, 2016).

Maiklo Lynno atliktoje apklausoje (Lynn M. , Motivations for tipping: How they differ across more and less frequently tipped services, 2016) tyrinėta kokios profesijos dažniausiai gauna arbatpinigių. Tyrime surinkti duomenys iš trijų šimtų penkiasdešimt šešių respondentų. Apklausa atlikta internetu, gaunant atsakymus už nedidelį piniginį atlygį. Respondentai pagal savo bruožus sudarė heterogenišką imtį. Iš respondentų surinkti duomenys, kaip dažnai jie duoda arbatpinigių skirtingų profesijų atstovams. Pagal vidutinį arbatpinigių gavimo tikėtinumą profesijos suskirstytos į retai, retkarčiais ir dažnai gaunančias arbatpinigius profesijas.



1 pav. Vidutinis tikėtinumas gauti arbatpinigių pagal profesiją

Pagal pateiktą pavyzdį (žr. 1 pav.) matome, kaip skiriasi tikėtinumas gauti arbatpinigių dirbant skirtingas profesijas. Tyrimas nustatė, kad pagarbos motyvai turi mažesnį poveikį arbatpinigių davimo tikėtinumui dažnai juos gaunančiose nei rečiau gaunančiose profesijose. Priešingai altruizmo motyvas turi didesnį poveikį arbatpinigių davimui profesijose, kurios dažnai gauna tokį atsidėkojimą, nei rečiau gaunančiose.

Pagal šią analizę autorius siūlo keletą būdų, kaip padidinti tikėtinumą, kad darbuotojas gautų arbatpinigių. Profesijose, kuriose arbatpinigiai paliekami retai ar retkarčiais galima padidinti galimybės gauti arbatpinigių sakant, kad už arbatpinigius klientai gaus geresnį aptarnavimą ir, kad tai padės darbuotojams. Sakymas, kad darbuotojai tikisi arbatpinigių turėtų mažesnį poveikį šiose profesijose. Tuo tarpu profesijose, kuriose dažnai gaunami arbatpinigiai didesnį poveikį turėtų sakymas, kad arbatpinigiai prisideda prie mažų darbuotojų algų ir jų yra tikimasi, o sakymas, kad už arbatpinigius bus teikiamas geresnis aptarnavimas padarytų mažesnį poveikį tikėtinumui gauti arbatpinigių.

Segmentavimas – tai procesas, kurio metu rinkos dalinamos į grupes žmonių, turinčių panašius poreikius ar bruožus dėl kurių jie elgtūsi panašiai perkant. Šis procesas tapo vienu pagrindinių įrankių kuriant marketingo strategijas JAV ir kitose šalyse. Segmentavimo tyrimo tikslas – tirti rinką

ir rasti nišines galimybes ir jas išnaudoti. To galima pasiekti surandant grupę vartotojų ir jiems kuriant unikalias marketingo programas.

1.4 Rinkų segmentavimas

Iš praktinės pusės segmentavimas turi būti valdomas tam, kad būtų sėkmingas. Neįmanoma pasinaudoti visomis rinkos galimybėmis, todėl reikalinga strategiškai spręsti, kurios galimybės svarbiausios. Pirmiausia svarbu suprasti, kad produktas ar paslaugos turi įtikti kiekvienam žmogui. Firmos gali būti labai sėkmingos dirbdamos tik su tam tikro segmento klientais. Antra, firmos turėtų valdyti savo produkto savybes tam, kad būtų pasiektas maksimalus efektyvumas. Dėl įvairių netiesioginių kaštų svarbu, kad produktas atitiktų klientų segmentų norus ir lūkesčius duotuoju metu ir išliktų vienu žingsniu priekyje prieš konkurentus.

Yra daug alternatyvų kaip galima segmentuoti rinką. Dauguma šių būdų atsirado iš vartotojų elgsenos tyrimo. Sprendimų priėmimas yra veikiamas racionalių ir emocinių faktorių, pavyzdžiui demografija, geografija, privalumai, motyvacija, pirkimo įpročiai ir kiti.

Kaip pavyzdį galima nagrinėti naftos kompaniją. Norint segmentuoti šios kompanijos rinką galima tirti šios firmos klientų klientus. Teritorinė pardavimų analizė galėtų būti pradinis taškas. Vartotojų demografija ir socioekonominiai požymiai kaip amžius, lytis, pajamos, ir kiti gali būti tyrinėjami. Produktų vartojimas – dizelino ar benzino vartojimas kiekis gali būti analizuojamas. Papildomai prekinių ženklų prisirišimas ar pakeitimo tendencijos, kainų jautrumas gali sutiekti įžvalgų segmentavimui. Pateikiami šeši trumpi rinkos segmentavimo pavyzdžiai:

- Geografinis: medicininių instrumentų firma gali gauti duomenis iš ligoninių asociacijos ir susisiekti su ligoninėmis pagal jų regioną, lovų skaičių. Toks sprendimas padėjo apibrėžti rinką naujam produktui.
- Verslo demografija: grafikos priemonių įmonė gali susisiekti su grafinio dizaino įmonėmis pagal jų verslo demografinius kintamuosius pagal jų specializacija, siūlomas paslaugas, didžiausius klientus ir kitais.
- Įsisavintojų kategorijos: potencialūs klientai skirstomi į įsisavintojus ir neįsisavintojus. Toks skirstymas gali labiausiai padėti naujiems produktams atliekant tyriamąsias studjas ir kokybines procedūras.
- Privalumai: ko siekia įmonė pirkdama ofiso priemones? Ar tai kaina, aptarnavimas ar specialios savybės – greitis, kiekis, spalvos, ar pardavėjo reputacija? Privalumas vienam klientui gali būti trūkumas kitam.

- Produkto naudojimas: verslo rinkos gali būti segmentuojamas pagal vartojimo kiekį – didelis, vidutinis, mažas. Papildomai geriausi klientai gali būti apibūdinami keliais bruožais: užsakymų kiekis, pardavimų kiekis, pelnas, pajamos ir kiti.
- Pirkimo pobudžiai: kompanija ”Dell” strategija ieško išmanančių ir turtingų klientų klientų, kuriems nereiktų per daug pagalbos.

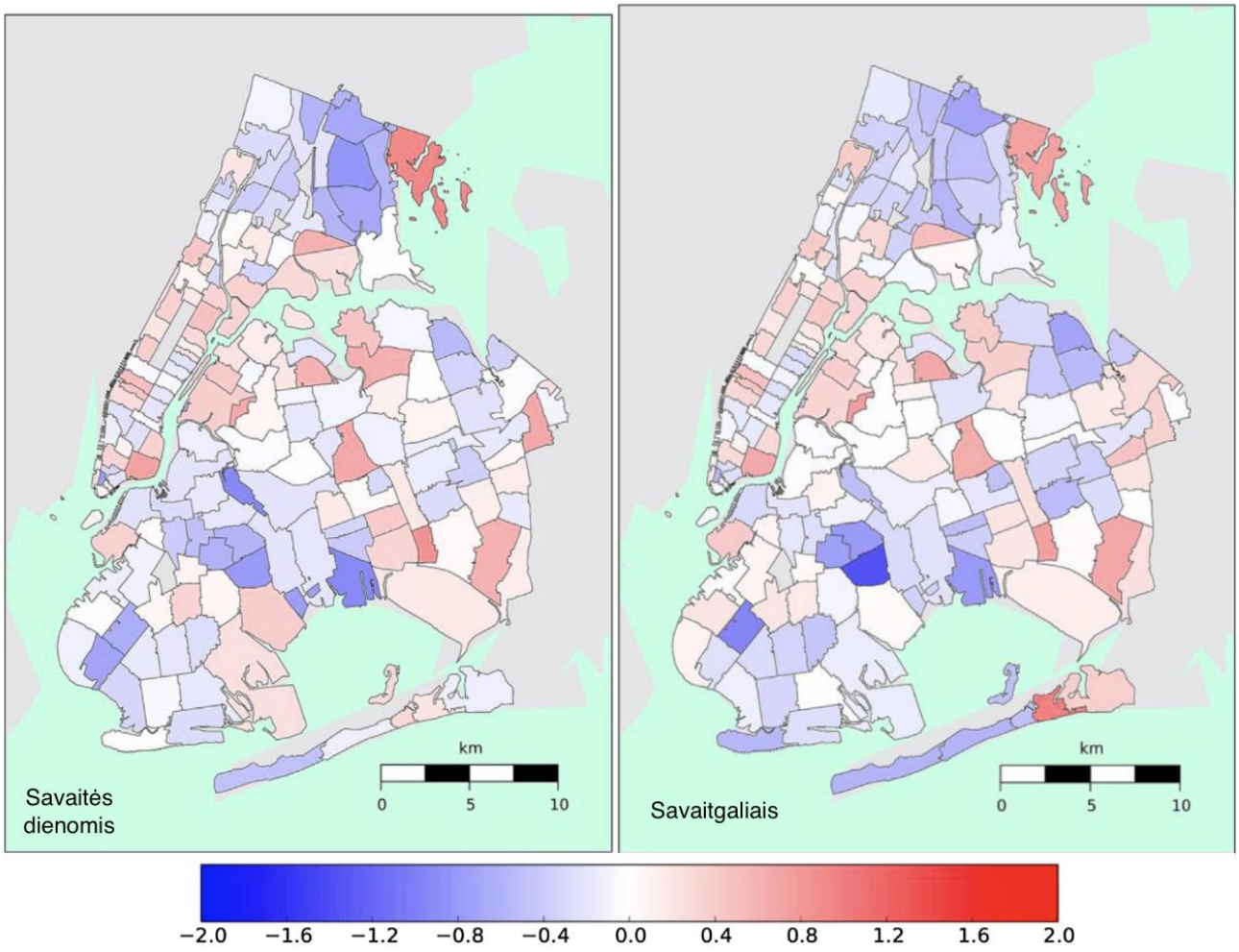
Kompanijos turi du pagrindinius sprendimus: segmentuoti rinką arba vertinti visą rinką kaip potencialius klientus. Antrasis variantas lemia, kad kompanija naudoja nediferencijuotos rinkos modelį. Yra tik keletas kompanijų, kam toks modelis būtų į naudą. Nors nediferencijavimas praktiškai nei vienai įmonei netinkai, daug jų naudoja šį modelį vertinant visus kaip potencialius klientus vietoj to, kad kreiptų save link labiausiai tikėtinų klientų. (Weinstein, 2010)

1.5 Geografijos įtaka taksi naudojimuisi

Tiriant su viešiuoju transportu susijusius duomenis svarbu atsižvelgti į jų geografinį išsidėstymą. Esami tyrimai dažnai atsižvelgia į tokius paklausą lemiančius faktorius kaip populiacijos dydis ir darbingumo lygis (McNally, 2008). Tačiau vidiniai faktoriai kaip kelionės trukmė, kaina ir paslaugų kokybė yra susiję su viešiojo transporto paslaugomis (Kanafani, 1983). Dauguma viešojo transporto tyrimų apsiriboja autobusais arba metro ar tramvajais (Haire & Machemehl, 2007; Hamberger & Chatterjee, 1987; Kain & Liu, 1999; Kuby, Barranda, & Upchurch, 2004; Tao, Corcoran, Mateo-Babiano, & Rohde, 2014) tačiau taksi transporto tyrimų nėra daug. Tai galima paaiškinti duomenų apie taksi keliones trūkumu. Ankstesni tyrimai bandė spręsti šią problemą matuojant pravažiuojančių taksi mašinų skaičių tam tikrose miesto gatvėse (Yang, Lau, Wong, & Lo, 2000). Dėl miestų dydžio ir didelio taksi automobilių skaičiaus toks būdas nebuvo labai efektyvus. Naudojant šiuolaikines technologijas tokio tipo duomenų rinkimas tapo daug paprastesnis. Didelio tikslumo geografiniai duomenys leidžia fiksuoti taksi automobilių keliavimo trajektorijas ir jas susieti su klientų poreikiais.

Tradiciniai būdai tirti viešojo transporto užimtumą naudoja keturių žingsnių modelį (McNally, 2008) ir paprastąjį mažiausių kvadratų (angl. *ordinary least squares – OLS*) daugiamatės regresijos modelį. Lyginant su keturių žingsnių metodu OLS modelis ir gana greitas, pigesnis ir labiau tinkantis analizei didesniu masteliu (Cardozo, García-Palomares, & Gutierrez, 2012). Kaip globalios regresijos modelis, viena iš OLS prielaidų yra tai, kad visi kintamieji nejuda tiriamojoje erdvėje. Tačiau taksi užimtumas yra labai priklausomas nuo aplinkos ir aiškinamieji kintamieji kinta erdvėje. Tai nulemia, kad OLS naudojimas nebūtų pats optimaliausias sprendimas. Alternatyva naudoti geografiškai pasvertą regresiją (angl. *geographical weighted regresion – GWR*) (Brunsdon,

Fotheringham, & Charlton, 1996). Šis metodas leidžia nepriklausomiems kintamiesis kisti erdvėje ir suteikia galimybę vizualizuoti regresijos koeficientus, kas padeda vizualiai nustatyti duomenų heterogeniškumą.



2 **pav.** Erdvinis išliekamosios vertės ir paklaidų kvadratų santykio pasiskirstymas geografiškai pasvertam modeliui. Adaptuota (Qian & Ukkusuri, 2015).

1.6 Kitų veiksnių įtaka taksi naudojimuisi

Tam, kad nustatyti išorinių veiksnių įtaką taksi paslaugų naudojimui Niujorko mieste duomenys sugruopuoti pagal pašto kodus. Įvertinta tokių išorinių veiksnių kaip ispanų kilmės populiacijos dydis, bakalauro ar aukštesnį išsilavimo laipsnį turinčių asmenų populiacijos dydis, nedarbo lygis, vidutinės pajamos, kelių tankumas, komercinių plotų dydis, gyvenamųjų plotų dydis, kelių tankumas, dviračių takų tankumas, parkavimo vietų skaičius, metro pasiekiamumas, autobusų pasiekiamumas, kelionės į darbą trukmės įtaka taksi paslaugų populiarumui. Nustatyta, kad vidutinės pajamos ir kelionės į darbą laikas neigiamai koreliuoja su taksi naudojimu, o turinčių bakalauro ar aukštesnį išsilavinimą populiacijos dydis, komercinio ploto dydis, kelių tankumas ir

metro pasiekiamumas veikia teigiamai. Intuityviai taksi paslaugos mažiau prieinamos žemas pajamas turintiems žmonėms ir rečiau naudojamos žmonių, turinčių didesnes pajamas. Tačiau nustatyta, kad pajamų dydis taksi populiarumą gali veikti ir teigiamai, ir neigiamai. Tai priklauso nuo geografinės lokacijos. Komercinio ploto įtaka taip pat priklausoma nuo lokacijos. Aplink miesto centrą ji turi mažesnę įtaką, tačiau toliau nuo centro, tuo didesnis komercinio ploto dydis skatina taksi naudojimąsi. Bakalauro ar aukštesnio laipsnio išsilavinimą turinčių asmenų kiekis teigiamai veikia taksi naudojimą, bet šis poveikis stipriausias aplink miesto centrą ir mažėja tolstant nuo jo. Tai gali būti dėl to, kad tokį išsilavinimą turintys žmonės turi daugiau galimybių gauti didesnes pajamas ir gyventi arčiau centro. Kelionės į darbą trukmė neigiamai veikia keliones taksi – ilgesnės kelionės atrodo mažiau patrauklios. Gatvių tankis koreliuoja su populiacijos dydžiu. Stebėtinai metro pasiekiamumas didina važiavimų skaičių. Intuityviai atrodytų, kad dėl mažesnių kainų metro turetų mažinti taksi kelionių skaičių, bet pastebimas priešingas poveikis. Tai būtų galima bandyti paaiškinti paskutinio atstumo kelionėmis, kuomet taksi važiuojama nuo metro stoties iki kelionės tikslo. (Qian & Ukkusuri, 2015)

1.7 Klasifikavimo metodai

Sprendimų medžių klasifikavimo metodai yra patrauklūs (Kanal, 1979; Landeweerd, Timmers, Gersema, Bins, & Halic, 1983; Li & Dubes, 1986; Nadler, 1970; Swain & Hauska, 1977; Wang & Suen, 1987) dėl šių savybių:

1. Globalūs sudėtingų sprendimų regionai (ypač daugiamatėse erdvėse) gali būti apytiksliai apskaičiuoti sujungiant paprastesnius lokalius sprendimų regionus įvairiuose medžio lygiuose.
2. Priešingai nei vieno etapo klasifikatoriuose, kur kiekvienas įrašas lyginamas su visais požymiais, medžio klasifikatoriuje įrašas lyginamas su požymių poaibiu, kas sumažina skaičiavimų kiekį.
3. Vieno etapo klasifikatoriuose naudojama visų savybių poaibis nustatant visus požymius. Poaibis nustatomas pagal globalaus tinkamumo kriterijų siekiant maksimalaus vidutinio požymių išskyrimo (Swain & Hauska, 1977). Sprendimų medžių metoduose vidinės viršūnės gali turėti skirtingus savybių poaibius tam, kad būtų naudojamas optimalus poaibis nustant požymius toje viršūnėje. Toks lankstumas gali suteikti geresnį našumą lyginant su vieno etapo klasifikatoriais (Mui & Fu, 1980; You & Fu, 1976).

4. Daugiamatėje analizėje su dideliu savybių ir požymių skaičiumi dažnai reikia apytikriai apskaičiuoti aukštų dimensijų pasiskirstymą (galimai daugiarūšį) ar tam tikrų požymių pasiskirstymo parametrus, tokius kaip a priori tikimybės iš nedidelio dydžio apmokymo duomenų imties. Tai sukelia daugiamačių dimensijų problemą, kurios galima išvengti sprendimų medžiuose naudojant mažesnį skaičių savybių vidinėse viršūnėse taip nepatiriant didelio sumažėjimo našume.

Tačiau turi ir neigiamų aspektų:

1. Požymių persidengimas nulemia didesnį lapų skaičių medyje nei yra požymių, kas pailgina skaičiavimo laiką ir naudojamos atminties kiekį.
2. Klaidų kiekis gali kauptis einant iš viršūnės į viršūnę. Neįmanoma optimizuoti ir tikslumui ir efektyvumui (Wu, Landgrebe, & Swain, 1975). Norint pasiekti tam tikrą tikslumą reikia paaukoti efektyvumą.
3. Sukurti optimalų sprendimų medį yra sudėtinga. Metodo efektyvumas stipriai priklauso nuo to kaip gerai sudarytas medis.

Kolektyvinio (angl. *ensemble*) mokymosi metodai, tokie kaip atsitiktinio miško metodas, tampa vis populiaresni dėl savo tikslumo ir atsparumo duomenų triukšmui. Pagrindinė tokių metodų idėja tai, kad imtis klasifikatorių bus tikslesni nei vienas klasifikatorius. Sprendimų medžių privalumai:

1. Gali veikti su daugybių savybių (angl. *features*) be savybių trynimo.
2. Pateikia savybių svarbumo reikšmes.

1.8 Apibendrinimas

Arbatpinigiai – svarbus pajamų šaltinis taksi automobilių vairuotojams. Nors nėra visiškai aišku, kodėl klientai palieka arbatpinigių, dažnai tai susiję su socialinėmis normomis ar įsitikinimais. Kadangi tai yra papildomas pajamų šaltinis, arbatpinigiai gali būti naudojami, kaip papildomas kriterijus, pagal kurį būtų galima segmentuoti rinką. Kelionių pasibaigusių arbatpinigių gavimu padidinas gali lemti didesnes pajamas. Kadangi geografinė padėtis svarbi taksi kelionių kiekiui, ji gali padėti segmentuoti rinką. Segmentavimui vienas populiariausių būdų yra k-vidurkių metodas, klasifikavimui – sprendimų medžių metodas ir atsitiktinių miškų metodas, taip pat neuroninių tinklų metodai.

2. TYRIMO METODAI

2.1. K-vidurkių segmentavimo metodas

Tarkime, kad X = $\{x_i\}, i = 1, \dots, n$ yra n dydžio d dimensijų taškų aibė, kurią norime suskirstyti į K skaičių poaibių - klasterių. Algoritmas suranda poaibius, kuriuose visų taškų atstumų iki poaibio centro, arba poaibio taškų vidurkio, bendra kvadratinė paklaida būtų minimizuota. Jeigu $\mu_k$yra klasterio c_k vidurkis. Kvadratinė paklaida tarp μ_k ir klasterio c_k taškų yra.

$$J(c_k) = \sum_{x_i \in c_k} ||x_i - \mu_k||^2 \quad (2.1.1)$$

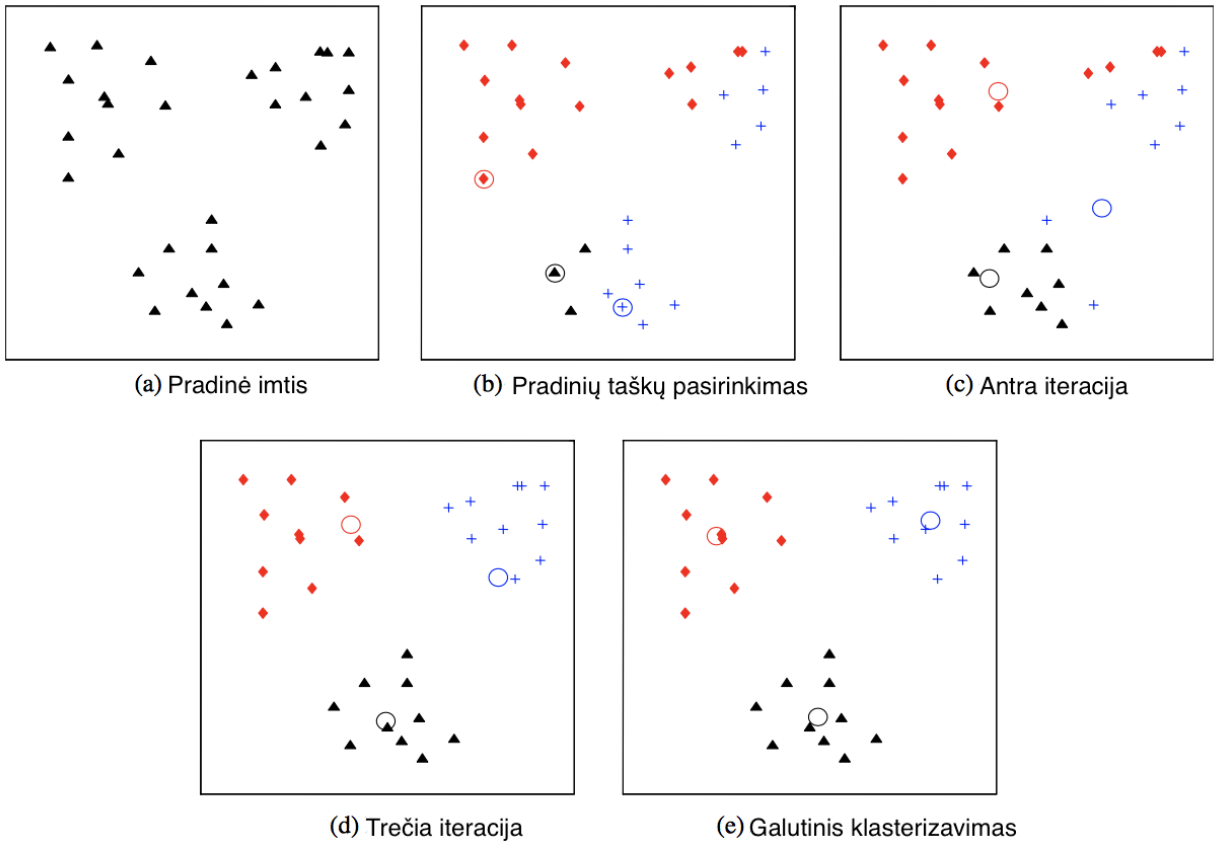
K-vidurkių algoritmo tikslas rasti mažiausią kvadratinę paklaidą visiem K klasteriam.

$$J(c_k) = \sum_{k=1}^{K} \sum_{x_i \in c_k} ||x_i - \mu_k||^2. \quad (2.1.2)$$

Šios funkcijos minimizavimas yra žinomas kaip NP-sudėtinga problema (net kai K = 2) (Drineas, Frieze, Kannan, Vempala, & Vinay, 1999). Todėl K-vidurkių algoritmas, kuris yra godus algoritmas, gali rasti tik lokalius minimumus, nors tyrimai yra parodę, kad šis algoritmas su gana didele tikimybe gali rasti globalų minimumą, kai klasteriai yra pakankamai atsiskyrę (Meila, 2006). K-vidurkių algoritmas prasideda su aibe taškų ir juos priskyrinėja klasteriams siekdamas sumažinti kvadratinę paklaidą. Kadangi kvadratinė paklaida mažėja didėjant klasterių K skaičiui ir yra lygi nuliui kai K = n, algoritmas veikia tik su fiksuotu klasterių skaičiumi. Pagrindiniai algoritmo žingsniai (Jain & Dubes, Algorithms for Clustering Data. , 1988):

1. *Pasirinkti pradinį padalijimą su K klasterių, kartoti 2 ir 3 žingsnius kol klasteriai stabilizuojasi.*
2. *Atlikti naują padalijimą priskiriant taškus prie jiem arčiausiai esančių klasterių.*
3. *Perskaičiuoti klasterių centrus.*

2 pav. pateikta dvimačių taškų aibės klasterizavimas į tris centrus.



3 pav. K-vidurkių algoritmo veikimas. Adaptuota (Jain, 2009).

Algoritmui reikalinga trys parametrai: Klasterių skaičius K, pradiniai klasterių taškai ir atstumo metrika. Svarbiausiais – K. Nėra matematinio kriterijau, kuris nustatytų geriausią K reikšmę, tačiau yra euristikų leidžiančių ją nustatyti (Tibshirani, Walther, & Hastie, 2001). Dažniausiai surandami klasteriai su skirtingomis K reikšmėmis ir dalykinės srities ekspertas nustato geriausią variantą. Skirtingi pradiniai taškai gali nulemti skirtingą taškų pasiskirstymą tarp klasteriu. Norint rasti geriausią pasiskirstymą galima keletą kartų kartoti algoritmą su ta pačia K reikšme, bet skirtingais pradiniais taškais ir išrenkant klasterius su mažiausia kvadratine paklaida. Algoritmas dažniausiai naudojamas su Euklidine atstumo metrika naudojama skaičiuojant atstumus tarp klasterių centrų ir taškų juose. Naudojant šią metriką klasteriai įgauna apvalią apskritimo formą. Naudojant kitokias atstumo metrikas galima gauti skirtingų formų klasterių.

2.2. Klasifikavimo metodai

2.2.1. Sprendimų medžio klasifikavimo metodas

Sprendimų medžių klasifikatoriai sėkmingai naudojami daugelyje skirtingų sričių. Tokių sričių kaip radaro signalo klasifikavimas, balso atpažinimas, medicininė diagnozė ir kt. Viena svarbiausių sprendimų medžio savybių – gebėjimas sprendimo priėmimo procesą apibūdinti kaip seką paprastesnių sprendimų. Norint apibūdinti šio algoritmo veikimą įvedame šias sąvokas:

1. Jeigu turime grafą $G = (V, E)$ kur V – grafo viršūnės, o E – briaunos. Jeigu briaunos yra surikiuotos poros (v,w) tuomet grafas orientuotas.
2. Kelias grafe laikas seka briaunų $(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n)$. Sakome, kad kelias nuo v_1 iki $v_n$yra n ilgio.
3. Orientuotas grafas be ciklų vadinamas orientuotu aciklišku grafiku. Orientuotas medis yra orientuotas acikliškas grafas tenkinantis šias sąlygas:
    a. Yra tik viena šaknies viršūnė į kurią neįeina jokios briaunos. Šaknies viršūnė turi visus priklausomo kintamojo požymių pavadinimus.
    b. Visos kitos viršūnės turi tik po vieną įeinančią briauną.
    c. Kiekviena viršūnė turi unikalų kalią iki šaknies.
4. Jeigu (v,w) yra briauna medyje, tuomet v yra vadinamas w tėvu, o w yra v sūnus. Jeigu yra kelias nuo v iki w ($v \neq w$), tuomet v yra tikrinis w protėvis, o w yra tikrinis v palikuonis.
5. Viršūnė be palikuonių vadinama lapu arba terminalu. Visos kitos viršūnės, išskyrus šaknį, vadinamos vidinėmis viršūnėmis.
6. Viršūnės v gylis medyje yra atstumas nuo šaknies iki viršūnės v. Aukštis viršūnes v medyje yra atstumas didžiausio kelio nuo v iki lapo. Medžio aukštis yra šaknies aukštis. Viršūnės v lygis medyje yra medžio aukštis atėmus viršunės v gylį.
7. Surikiuotas medis yra medis, kurio visų viršūnių sūnūs surikiuoti (paprastai iš kairės į dešnę).
8. Dvejetainis medis – surikiuotas medis, kuriame:
    a. Kiekvienas viršūnės sūnus apibūdinamas kaip kairys arba dešnys sūnus.
    b. Nė viena viršūnė turi daugiau nei vieną kairį ir vieną dešnį sūnų.
9. Viršūnės v balansas dvejetainiame medyje yra (1+L)/(2+L+R), kur L ir R yra skaičius viršūnių kairiame ir dešniame viršūnės pomedžiuose. Dvejetainis medis yra α subalansuotas, $0 < \alpha \leq 1$, jeigu kiekviena viršūnė turi balansą tarp α ir $1 - \alpha$. Dvejetainis medis, kurio balansas yra ½ dar vadinamas pilnu medžiu.

Medžius žymėsime kaip T. 3 pav. Parodo bedrinį sprendimų medį.

10. Jeigu dvi vidinės viršūnės turi bent vieną bendrą priklausomo kintamojo požymį, tuomet sakoma, kad yra požymių persidengimas. (žr 4 pav.)
11. Vidutinis skaičius sluoksnių nuo šaknies iki lapų vadinamas vidutiniu medžio gyliu. Vidutinis skaičius vidinių viršūnių kiekviename sluoksnyje vadinamas vidutiniu medžio pločiu. Bendrai vidutinis medžio plotis atspindės suteikiamą svorį tikslumui tuo tarpu medžio gylis atspindės svorį duodamą našumui.

Įveskime bendrai pasiskirsčiusių atsitiktinių kintamųjų poros $(\underline{X}, Y)$ sąvoką, kur q-dimensijų vektorius $\underline{X}$ pažymi savybių vektorių, o Y nurodo X požymį.

12. $\underline{X}$ vadinamas *surikiuotu* arba *skaitiniu* (Breiman, Friedman, Olshen, & Stone, 1984) šablonu jeigu jo savybės ima reikšmes iš surikiuoto rinkinio ir *kategoriniu* jei savybės iš rinkinio neturinčio natūralaus surikiavimo. Surikiuotos arba numerinės savybės gali turėti diskrečias arba tolydžias reikšmes. Dėl žymėjimo paprastumo manykime, kad $\underline{X}$ yra tolydaus, surikiuoto tipo, taip pat manykime, kad $\underline{X}$ ima reikšmes iš R^q. Tarkime, kad Y įgauna vertes iš $\{1,2,\dots,J\}$, kitaip tariant yra J skaičius požymių. Tuomet klasifikavimo tikslas nustatyti Y pagal $\underline{X}$.
13. Sprendimo taisyklė $d(\cdot)$ yra funkcija, kuri paverčia R^q į $\{1,2,\dots,J\}$, kur $d(\underline{X})$ atstovauja savybių vektoriaus $\underline{X}$ požymį. Tikrasis neteisingų klasifikavimų santykis d žymimas kaip $R^*(d)$, kur $p(\cdot)$ žymi tikimybę.

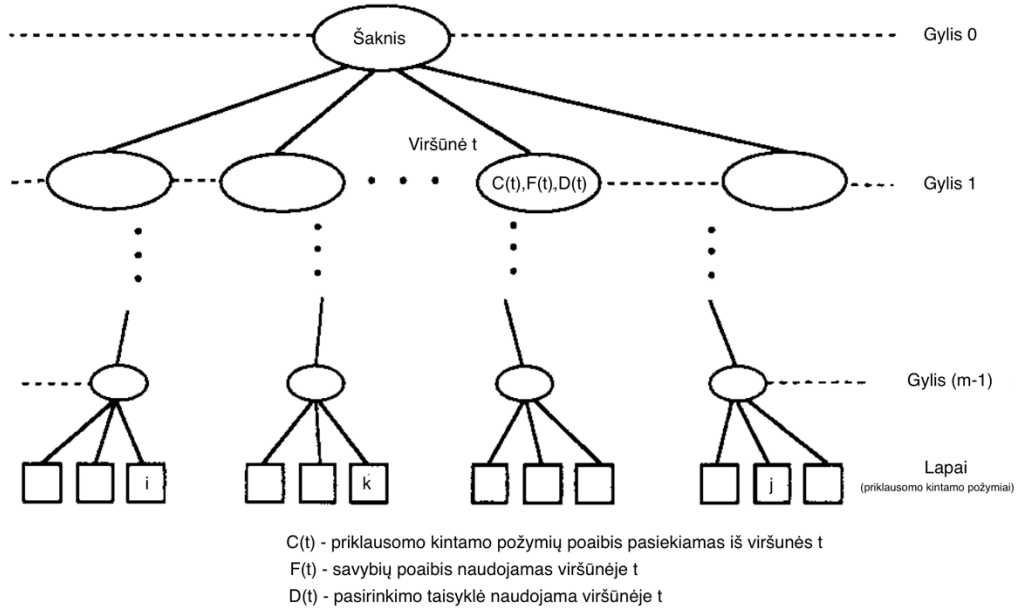
$$R^*(d) = p(d(\underline{X}) \neq Y) \tag{2.2.1.1}$$

Pažymėkime turimą klasifikuotą imtį kaip

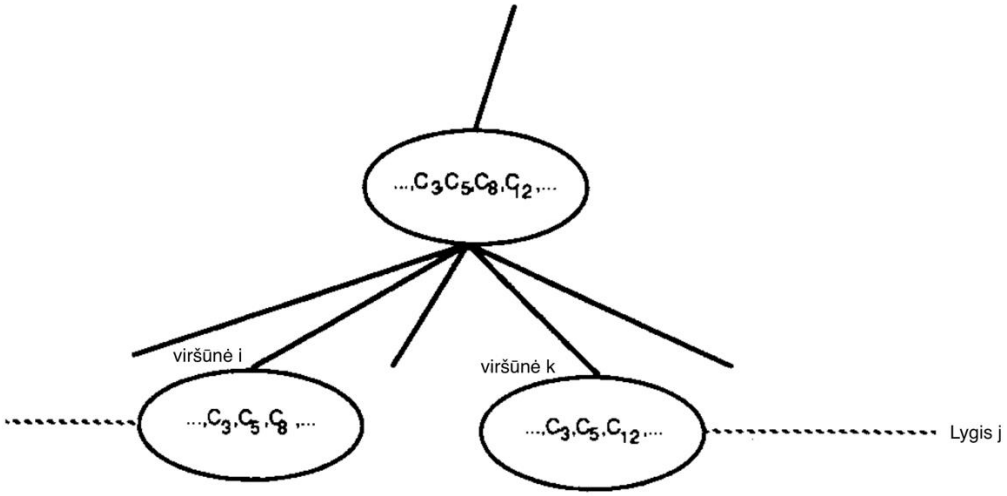
$$\mathcal{L} = \{(\underline{X}_n, Y_n), n = 1,2,\dots,N\}. \tag{2.2.1.2}$$

14. Dažniauisiai turimos klasifikuotos imtįs dalinamos į dvi dalis: apmokymo imtį $\mathcal{L}^{(1)}$ ir testavimo įmtį $\mathcal{L}^{(2)}$. Dažniausiai apmokymui naudojama 2/3 imties, o testavimui likusi 1/3 imties.
15. Kadangi sudėtinga apskaičiuoti tikrąjį neiteisingo klasifikavimo santykį $R^*(d)$, jis dažniauisia aptyksliai apskaičiuojamas naudojant apmokymo arba testavimo imtį. Pažymėkime apytikslį neiteisingų klasifikavimų sanktykį kaip $R(d)$. Kai apmokymo imtis naudojama apskaičiuoti $R^*(d)$, $R(d)$ vadinamas $R^*(d)$ apkeitimo įverčiu, o kai naudojama testavimo imtis $R(d)$ vadinamas testinės imties $R^*(d)$ įverčiu. Betkokiu atveju neteisingų klasifikavimų santykis apskaičiuojamas padalinant neitisingai klasifikuotų reikšmių skaičių padalinus iš visų reikšmių. Sudėtingesnis metodas įvertinti neteisingų klasifikavimų santykį yra K-blokų kryžminio patikrinimo metodas. Jis padalina turimą klasifikuotą imtį į k apylygias dalis. Tuomet viena dalis naudojama testavimui, o likusios dalys apmokymui.

Metodas kartojamas kol visos dalys būna panaudotos testavimui.



4 pav. Bendrinio sprendimų medžio pavyzdys. Adaptuota (Safavian & Landgrebe, 1991)



5 pav. Medžio dalis su persidengimu viršūnėse i ir k (požymiai 3 ir 5 kartojasi). Adaptuota (Safavian & Landgrebe, 1991)

Pagrindiniai sprendimų medžio klasifikavimo metodo tikslai yra: 1) teisingai klasifikuoti kuo didesnę apmokymo imties dalį 2) nebūti priklausomam nuo apmokymo duomenų, kad pritaikius naujus duomenis tikslumas būtų kuo aukštesnis 3) būti lengvai atnaujinamam kuomet atsiranda naujų apmokymo duomenų 4) turėti kuo paprastesnę medžio struktūrą. Tuomet sprendimų medžio klasifikatoriaus sukūrimo procesas gali būti apibrėžtas šiais uždaviniais (Kulkarni & Kanal, 1976; Kurzynski,1983):

1. Pasirinkti tinkamą medžio struktūrą.

2. Pasirinkti optimalius savybių poaibius kiekvienoje vidinėje viršūnėje.
3. Nustatyti sprendimo taisyklę kiekvienai vidinei viršūnei.

Optimalaus medžio sudarymo problemą galima išreikšti kaip optimizavimo problemą:

$$\min_{T,F,d} p_e(T,F,d) \tag{2.2.1.3}$$

kur p_e yra bendra klaidos tikimybė, T yra konkreti medžio struktūra, F yra savybių poaibis, o d yra sprendimo taisyklės, naudojami vidinėse medžio viršūnėse. Šią problemą galima spresti dviem žingsniais:

$$p_e(T,F,d^*(T,F)) = \min_d P_e(T,F,d) \tag{2.2.1.4}$$

$$p_e(T^*,F^*,d^*(T^*,F^*)) = \min_{T,F} p_e(T,F,d^*(T,F)) \tag{2.2.1.5}$$

Pažymėtina, kad problemoje nėra įtraukiama nei laiko nei skaičiavimų greičio faktorių. Juos įtraukus problema taptų dar labiau sudėtinga.

Žiūrint iš informacijos teorijos pusės optimaliausiais medis yra tas, kuris kiekviename medžio lygyje maksimizuoja informacijos gavimą.

Vertinant medžio struktūrą naudojami kriterijai: minalus klaidų kiekis, minimalių ir maksimalių kelių ilgis, minimalus viršūnių skaičius, minimalus tikėtino kelio ilgis, maksimalus vidutinis abipusės informacijos gavimas. Kadangi medžiai konstruojami rekursyviai, daug optimizavimo sprendinė negali būti lengvai pritaikomi dėl skaičiavimų greičio ir atminties talpos dydžio.

Pagrindinis kriterijus leidžiantis kurti sprendimus yra informacijos gavimas. Šis terminas sprendimo medžiuose apibūdina skirtumą tarp tėvo viršūnės neskaidrumo (angl. *inpurity)* ir pasvertų vaikų viršūnių neskaidrumų. Dvejetainio medžio atveju informacijos gavimo formulė atrodo taip:

$$IG(D,s) = Neskaidrumas(D) - \frac{N_{kairė}}{N} Neskaidrumas(D_{kairė}) - \frac{N_{dešnė}}{N} Neskaidrumas(D_{dešnė}) \tag{2.2.1.6}$$

Kur D – duomenų rinkinys, s – išskyrimas, padalijantis duomenų rinkinį, N – duomenų rinkino dydis. Neskaidrumui nustatyti galima naudoti skirtingus formules:

$$\sum_{i=1}^{C} f_i(1-f_i) \tag{2.2.1.7}$$

Gini neskaidrumas, kur f_i požymio i dažnumas viršunėje, o C – unikalių požymių skaičius, naudojamas klasifikavime.

$$\sum_{i=1}^{C} -f_i \log(f_i) \tag{2.2.1.8}$$

Entropija, naudojama klasifikavime.

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - \mu)^2 \tag{2.2.1.9}$$

Dispersija, naudojama regresijoje, y_i-atvejo požymis, N-atvejų skaičius, μ – vidurkis, gaunamas su $\frac{1}{N}\sum_{i=1}^{N}\mu$.

Medžio dydis ribojamas naudojant gylio maksimalaus gylio limitą arba minimalaus informacijos gavimo limitą, kuris nustoja kurti naujas šakas jeigu gauta informacija neviršija šio limito. Tam, kad išvengti permokymo patartina patikrinti medžio tinkamumą su atskirais testiniais duomenimis. (Apache spark, 2017).

2.2.2. Atsitiktinio miško klasifikavimo metodas

Atsitiktinis miškas priskiriamas prie kolektyvinio (angl. *ensemble*) mokymosi grupės. Šis metodas papildo apjungimo (angl. *bagging*) metodą papildomu atsitiktinumo žingsniu. Apjungimo metodas sukuria daug sprendimų medžių su atsitiktiniais duomenų poaibiais ir požymiui nustatyti imami visų medžių rezultatai ir daugumos principu nustatomas teisingas atsakymas. Atsitiktinio miško metodas prie šios strategijos prisideda pakeisdamas medžių konstravimo principą. Paprastai medžiuose kiekviena viršūnė padalijama pagal geriausią išskyrimą tarp turimų savybių. Atsitiktinio miško atveju viršūnės dalija pagal atsitiktinį savybių poaibį kiekvienoje viršūnėje. Toks pakeitimas pagerina tikslumą lyginant su apjungimo metodu.

Algoritmo žingsniai:

- Paimti iš duomenų rinkinio tiek pat atsitiktinių poaibių kiek bus naudojama medžių.
- Su kiekvienu poaibiu sukurti sprendimų medį su tokiais pakeitimais: kiekvienoje viršūnėje paimti visų savybių poaibį ir su juo išrinkti geriausią išskyrimą.
- Nustatyti naujų duomenų požymius pasirenkant daugumos išrinktą požymį.

Apytikslį tikslumą galima gauti:

- Kiekviename poaibio ėmimo stadijoje gauti tikslumą su duomenimis nesančiais gautame poaibyje naudojant medį apmokytą su poaibio duomenimis.
- Apskaičiuoti bendrą visų medžių tikslumą.

(Liaw & Wiener, 2002)

2.2.2. Neuroninių tinklų klasifikavimo metodas

Neuroniniai tinklai tai skaičiavimo sistema sudaryta iš sujungtų skaičiavimo vienetų vadinamų neuronais. Šie taip vadinami dėl savo panašumo į smegenyse esančius neuronus. Vienas iš

išskiriančių neuroninių tinklų bruožų yra gebėjimas mokintis ir apibendrinti iš patirties ir pavyzdžių, prisitaikyti prie besikeičiančių sąlygų. Šie tinklai gali sujungti poveikio ir rezultato modelius arba atvirkščiai, pagal rezultatus nustatyti priežastis.

Modelio apmokymas vyksta parodant sistemai pavyzdžius duomenų ir jų rezultatų. Sistema pakeičia svorius tarp vidinių neuronų sumažindama paklaidą tarp savo rezultatų ir apmokymo rezultatų. Kai modelis pakankamai apmokytas, jis gebės apibendrinti taisykles ir numatyti rezultatus su nematytais duomenimis.

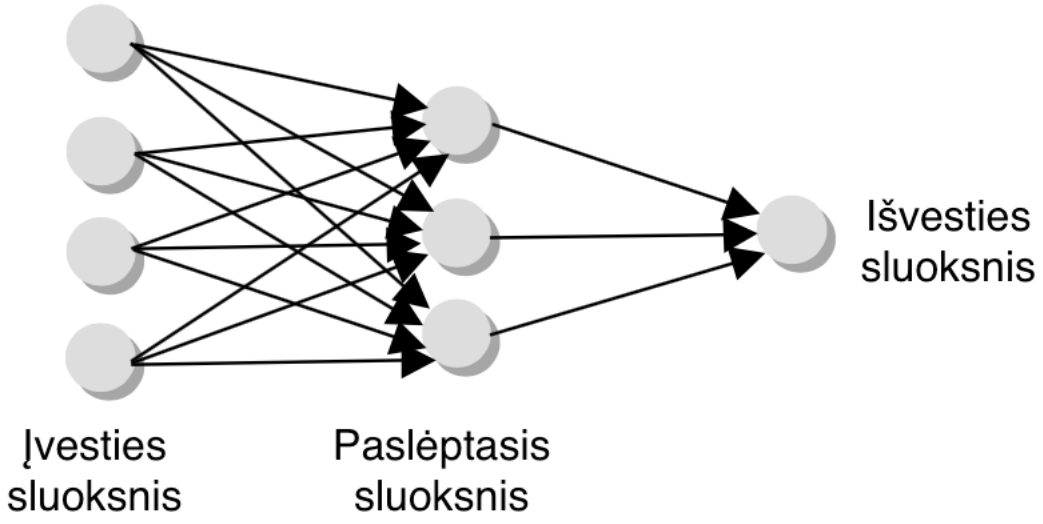
Neuroniniai tinklai gali atlikti tiek klasifikaciją su linijiniais ar nelinijiniais apribojimais, tiek funkcijų reikšmių apytikslį prognozavimą.

Tradiciškai kompiuteriai naudoja taisykles ar funkcijas, kurios aiškiai apibrėžia problemą ir žingsnius, kuriuos reikia atlikti, kad būtų rastas sprendimas. Yra daug situacijų kuomet taisyklės ar funkcijos nėra žinomos arba sunkiai nustatomos. Tokių problemų negalima spręsti tradiciniais skaičiavimo būdais, tačiau galima spręsti naudojantis neuroniniais tinklais.

- Neuroniniai tinklai gali apibendrinti iš pavyzdžių ir duoti prasmingus sprendimus.
- Jie susitvarko su duomenų paklaida, netolygumu ar išskirtimis.
- Tinklai gali adaptuoti rezultatus laikui bėgant ir kompensuoti besikeičiant sąlygoms.
- Duomenys duodami neuroniniam tinklui gali būti teoriniai, eksperimentiniai, istoriniai ar bet kokia išvardintų kombinacija. Duomenys gali būti keičiami srities eksperto siekiant pagerinti modelį.

Nors neuroniniai tinklai gali prognozuoti rezultatus, kuomet nėra žinoma taisyklių ar funkcijų, kurios leistų gauti rezultatus kitais būdais, bet nėra galimybės paaiškinti, kodėl gaunami vienoki ar kitokie rezultatai.

Toliau nagrinėsime neuroninio tinklo variantą – daugiasluoksnį perceptroną (angl. *multilayer perceptron*).

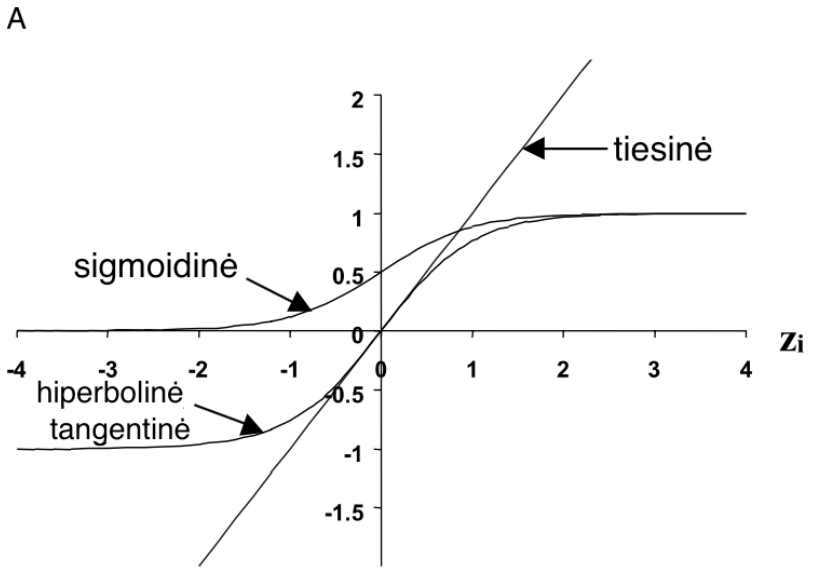


6 pav. Tipinis neuroninio tinklo išsidėstymas. Adaptuota (Rafiq, Bugmann, & Easterbrook, 2001).

Tinklas sudarytas iš įvesties sluoksnio, paslėptų sluoksnių ir išvesties sluoksnio. Visi sluoksniai sujungti (žr. 5 pav. rodykles). Visos jungtys turi svorius. Pradžioje visom jungtims suteikiama atsitiktiniai svoriai nuo -1 iki 1. Vėliau šie svoriai kinta vykstant mokymosi procesui. Įvesties sluoksnis gauna reikšmes iš išorės, o išvesties sluoksnis atiduoda rezultatus. Paslėptųjų sluoksnių užduotis išgauti duomenų savybes, kurių pagalba nustatoma požymiai. Pagrindiniai skirtumai tarp neuroninių tinklų yra neuronų aktyvavimo funkcijos. Nuo jų priklauso kokią reikšmę toliau perduoda neuronas. Daugiasluoksnio perceptrono atveju paslėptojo sluoksnio neuronų aktyvavimo funkcija dažnai būna sigmoidinė funkcija $\boldsymbol{S_i(\vec{x})}$:

$$S_i(\vec{x}) = \frac{1}{1+\exp(-z_i)}; \qquad (2.2.2.1)$$

$\boldsymbol{z_i} = \sum_{j=1}^{n} \boldsymbol{w_{ij} x_j}$, $\boldsymbol{x_j}$ – j-tojo įvesties neurono reikšmės, $\boldsymbol{S_i(\vec{x})}$ – i-tojo paslėptojo sluoksnio neurono reikšmės. Ši funkcija daugiamatę įvestį padalina į dvi dalis, su aukšta išvestim vienai daliai ir žema išvestim kitai daliai (žr. 6 pav.).



7 pav. Aktyvavimo funkcijų grafikai. Adaptuota (Rafiq, Bugmann, & Easterbrook, 2001)

Apmokymas daugiasluoksniame perceptrone apima ir atgalinį paskirstymą (angl. *back propagation*) tinklo paklaidos ir jungčių svorių pritaikymo. Tai esminis mokymosi algoritmo žingsnis.

Neuronų skaičiaus sprendimas tinkluose priklauso nuo sprendžiamos problemos. Įvesties sluoksnyje neuronų skaičius nusakomas pagal naudojamų kintamųjų kiekį, o išvesties – nuo išvadų kintamųjų kieko. Paslėptų sluoksnių ir juose esančių neuronų kiekis nustatomas eksperimentiniu būdu. Vientisoms funkcijoms turėtų pakakti vieno paslėpto sluoksnio, o nevientisoms dviejų sluoksnių su optimaliais neuronų kiekiais, bet tai nėra bendra taisyklė. Kadangi nėra vienareikšmiškai teisingų atsakymų į klausimą kiek neuronų ir jų sluoksnių turėtų būti naudojama, geriausias būdas - palyginti modelio tikslumą su skirtingais jų kiekiais.

2.3 Analizavimo priemonės

”Apache spark” yra paralelių skaičiavimų įrankis palaikantis keletą programavimo kalbų. Šis įrankis yra greitas, patikimas. Kadangi spark palaiko skaičiavimus operatyvinėje atmintyje jis yra greitesnis nei įrankiai, kurie duomenis ima iš ilgalaikės atminties, kaip ”Apache hadoop”. Spark taip pat turi daug laisvai prieinamų įrankių, kaip ”MLLib” ir ”Hive”, leidžiančių lengviau tirtis duomenis.

Šio įrankio privalumai:

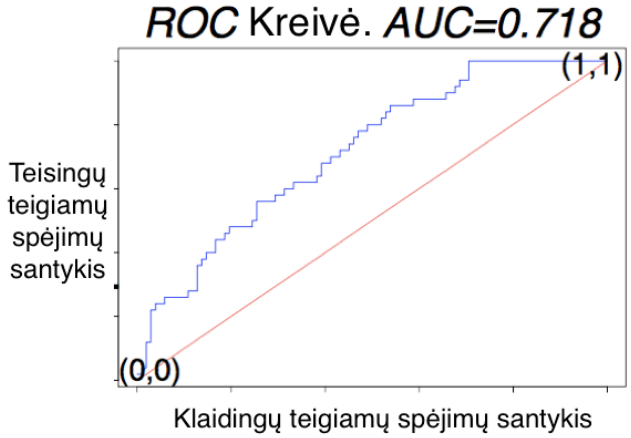
- Vienas greičiausių bendro naudojimo įrankių duomenų tyrimui
- Aktyvus atvirojo kodo bendromenės palaikymas

- Patikimumas įvertintas ”Intel” kompanijos rekomendacija naudoti medicinos srityje
- Galimybė rinkti duomenis iš skirtingų šaltinių

Tačiau spark taip pat turi ir trūkumų – nėra pats tinkamiausiais kuomet tiriami srautiniai duomenys, kuries reikalingas greitas reagavimo laikas ir resursų poreikis – reikalauja turėti nemaža operatyvinės atminties kiekį atliekant tyrimus su dideliais duomenų kiekiais. (Shoro & Soomro, 2015)

2.4 Klasifikavimo modelių tikslumo įvertinimas

Vienas iš būdų įvertinti klasifikavimo metodą yra įvertinti kiekį padarytų klaidų – neteisingų klasifikacijų. Šis būdas legvai pritaikomas ir tinka įvairiems klasifikacijos tipams. Tačiau toks vertinimas gali neatspindėti modelio nesubalansuotumo. Dvejetainiui klasifikavimui galima pritaikyti ploto po sprendimus priimančiojo ypatybių (angl. *receiver operating characteristics - ROC*) kreive metriką.



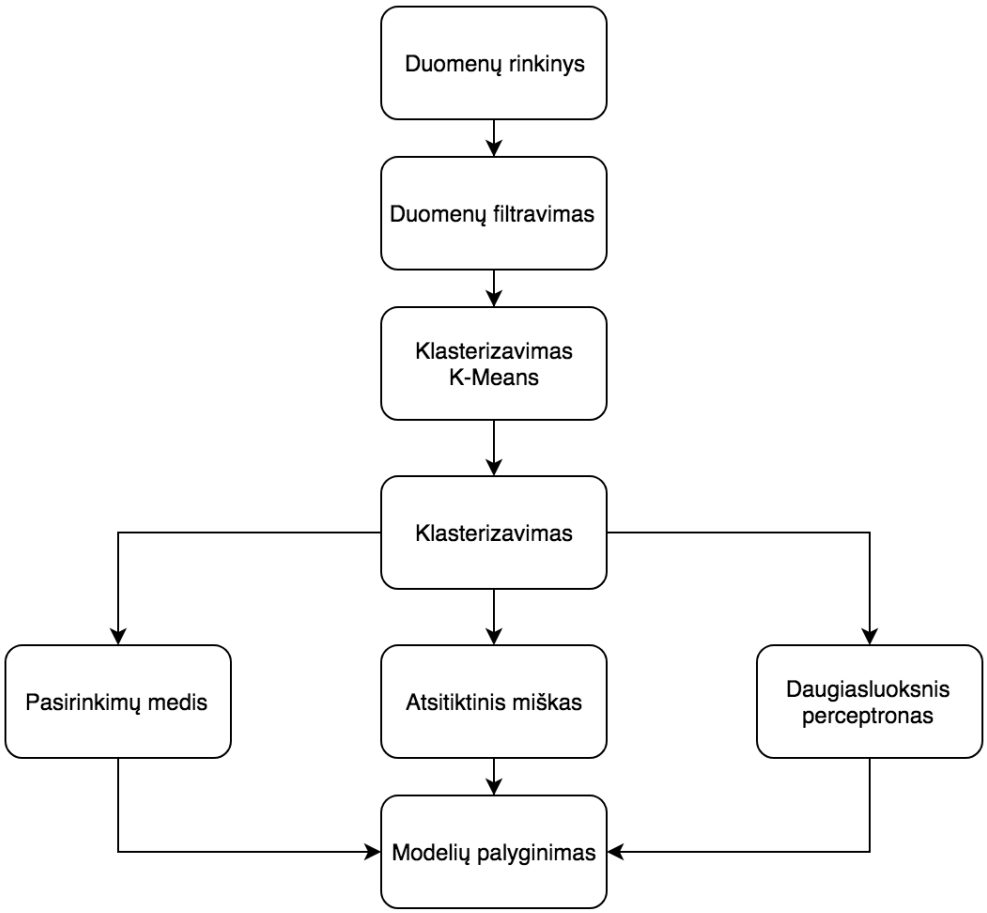
8 pav. Sprendimus priimančiojo ypatybių kreivė

Kreivė vertikalioje ašyje žymima teisingai spėtų teigiamo rezultato reikšmių santykis su visomis teigiamo rezultato reikšmėmis, o horizontalioje ašyje – neteisingai atspėtų neigiamo rezultato reikšmių santykis su visomis neigiamo rezultato reikšmėmis.

Šios kreivės originaliai buvo naudojamos signalų aptikimo teorijoje, bet nuo to laiko buvo pritaikytos daugelyje kitų sričių.

Visiškai atsitiktinį klasifikavimą kreivė atspindi kaip tiesią iš taško (0,0) į tašką (1,1). Bent kiek geresnis nei atsitiktinis klasifikavimo atsispindėtų kreive, kuri yra aukščiau atsitiktinio klasifikavimo tiesės. Skaitine reikšmei tai galima įvertinti plotu po kreive (angl. *area under curve – AUC*) – reikšmė didesnė nei 0.5 rodo geresnį nei atsitiktinis klasifikavimą. (Cortes & Mohri, 2003)

2.5 Tyrimo metodika



9 pav. Tyrimo metodikos grafikas

Tyrimui naudota ”Apache Spark” programinė įranga, veikianti ”Microsoft HDInsight” aplinkoje. Pagrindinis kriterijus renkatis šią aplinką buvo galimybė naudotis užrašų knygučių (angl. *notebooks*) įrankiu leidžiančiu greitai paleisti programinį kodą. Kita bandyta aplinka buvo ”Amazon EMR”, tačiau dėl komplikuoto užrašų knygučių įrašymo proceso šios aplinkos buvo atsisakyta. Naudota 40-ties virtualių branduolių dydžio grupė. Kadangi duomenų kiekis didelis buvo pasirinkta naudoti ”Apache Spark” ir susijusius įrankius. Šie įrankiai patogūs naudoti dėl to, kad leidžia duomenis apdoroti nausaugant visko operatyvinėje atmintyje. Kiti dažnai naudoji įrankiai kaip ”R” reikalauja papildomų priemonių norint dirbti su tokiais duomenų kiekiais. ”Spark” taip pat turi didelį atvirojo kodo palaikymą, dėl ko mašininio mokymosi metodai lengvai pasiekiami ir lengvai naudojami. Duomenys ”csv” formatu saugoti ”azure storage” talpykloje, vėliau apdoroti ir saugomi ”Hive” duomenų bazėje. Duomenys filtruojami pašalinant neteisingas reikšmes, kaip neigiamos pinigų sumos, dešimtimis tūkstančių skaičiuojami atstumai, pradžios ar pabaigos taškai esantys per toli nuo Niujorko. Klasterizavimui panaudotas ”k-vidurkių” metodas, nustatytas 20 iteracijų skaičius,

apskaičiuoti paklaidų dydžiai išbandant nuo 1 iki 10 klasterių skaičių. Išrinkus optimaliausią klasterių kiekį klasterizuojami mokomieji duomenys. Modeliavimui naudojamas 5-ių blokų (angl. *folds*) kryžminė patikra, kartu nustatant ir geriausius parametrus. Atlikus kryžminę validaciją gaunami rezultatai kiekvienai parametrų kombinacijai. Kadangi sprendimų medžio ir atsitiktinio miško metodai yra linkę būti permokinti, patikrinimui naudojami visi originalūs duomenys, turintys kelionių pradžios ir pabaigos koordinates. Kadangi duomenų schemos skiriasi tarp tam tikrų duomenų rinkmenų jas jungiant į bendrą vienetą stengtasi išsaugoti kintamųjų reikšmių prasmingumą.

3. REZULTATAI

3.1 Tyrimo duomenys

Tyrimui panaudoti duomenys surinkti Niujorko taksi ir limuzinų komisijos (TLC Trip Record Data, 2017). Pateikti duomenys nuo 2009 metų sausio mėnesio pradžios iki 2016 metų gruodžio pabaigos. Kiekvieno mėnesio duomenys išsaugoti kaip atskiros ”csv” formato rinkmenos. Duomenys rinkmenose išdėstyti lentelės principu. Kiekvienos rinkmenos pirmoje eilutė įrašyti stulpelių pavadinimai. Stulpelių pavadinimai ir toliau einantys duomenys išskirti kableliais. Lentelių struktūra laikui bėgant pakito: keitėsi stulpelių pavadinimai ir kai kurių stulpelių tipai, atsirado papildomų stulpelių. Modelių apmokymui panaudoti 2016 metų pirmų šešių mėnesių duomenys. Šie duomenys pasirinkti dėl naujumo – kelionių (Apache spark, 2017) tendencijos turėtų būti aktualesnės nei senesnių metų duomenyse. Taip pat šie duomenys paskutiniai turintys pilnas kelionių pradžios ir pabaigos geografinės koordinates – nuo 2016 metų liepos mėnesio duomenyse pateikia tik pradžios ir pabaigos vietų kodai. Testavimui panaudota visi duomenys turintys kelionės pradžios ir pabaigos koordinates bei kitus klasterizavimui bei klasifikavimui reikalingus stulpelius. Iš viso apmokymo imtyje yra 68009668 eilutės, o testavimo imtyje – 782479057 eilutės duomenų. 3.1.1 lentelėje pateikta duomenų rinkinių stulpelių struktūra.

3.1.1 lentelė duomenų rinkinio stulpelių struktūra.

Stulpelio pavadinimas	Reikšmės tipas	Paaiškinimas
VendorID	Sveikas skaičius	Įmonės, suteikusios duomenis, identifikacinis numeris
tpep_pickup_datetime	Data	Kelionės pradžios data
tpep_dropoff_datetime	Data	Kelionės pabaigos data
passenger_count	Sveikas skaičius	Keleivių skaičius
trip_distance	Skaičius	Kelionės atstumas
pickup_longitude	Skaičius	Kelionės pradžios geografinė ilguma
pickup_latitude	Skaičius	Kelionės pradžios geografinė platuma
RateCodeID	Sveikas skaičius	Kelionės tipo kodas

Stulpelio pavadinimas	Reikšmės tipas	Paaiškinimas
store_and_fwd_flag	Simbolinis	Ar kelionė buvo išsaugota prietaiso atmintyje
dropoff_longitude	Skaičius	Kelionės pabaigos geografinė ilguma
dropoff_latitude	Skaičius	Kelionės pabaigos geografinė platuma
payment_type	Simbolinis	Mokėjimo tipas
fare_amount	Skaičius	Kelionės kaina
extra	Skaičius	Sumuoti papildomi mokesčiai
mta_tax	Skaičius	Eismo ministerijos papildomas mokestis
tip_amount	Skaičius	Arbatpinigių suma
tolls_amount	Skaičius	Kelių mokesčių suma
improvement_surcharge	Skaičius	Tobulinimų mokestis
total_amount	Skaičius	Bendra suma

3.2 Duomenų filtravimas

Atlikus pradinę analizuojamų duomenų analizę pastebėta, kad geografinę lokaciją apibrėžiančių kintamųjų reikšmės yra neteisingos. Kelionės pradžios platumos ir ilgumos reikšmes buvo pernelyj nutolusios nuo Niujorko. Pavyzdžiui, kelionės pradžios taškas buvo nurodytas Ramiajame vandenyne, todėl suformuluoti papildomi reikalavimai duomenų rinkiniui.

Tam, kad pašalinti neteisingus duomenis iš apmokymo imties atliktas filtravimas, pašalinti duomenys:

- kurių ”trip_distance” daugiau nei 1000 ar mažiau nei 0,
- kurių ”fare_amount” mažiau nei 0,
- kurių ”pickup_longitude” mažiau nei -75.528844 ar daugiau nei -71.658250,
- kurių ”pickup_latitude” mažiau nei 39.858698 ar daugiau nei 42.188717,
- kurių ”dropoff_longitude” mažiau nei -75.528844 ar daugiau nei -71.658250,
- kurių ”dropoff_latitude” mažiau nei 39.858698 ar daugiau nei 42.188717.

Duomenim pridedami 2 papildomi išvestiniai stulpeliai: ”didTip” – ar ”tip_amount” buvo daugiau nei 0, “pickupHour” – kelionės pradžios valanda. Žemiau pateikiama stulpelių reikšmių statistika.

3.1.2 lentelė mokomojo duomenų rinkinio stulpelių skaitinės statistikos.

Stulpelis	Kiekis	Vidurkis	Standartinis nuokrypis	Minimumas	Maksimumas
VendorID	68009668	1,5351	0,4988	1	2
passenger_count	68009668	1,6650	1,3148	0	9
trip_distance	68009668	2,9779	3,6947	0,01	807,7
pickup_longitude	68009668	-73,9735	0,0380	-75,5145	-71,9190

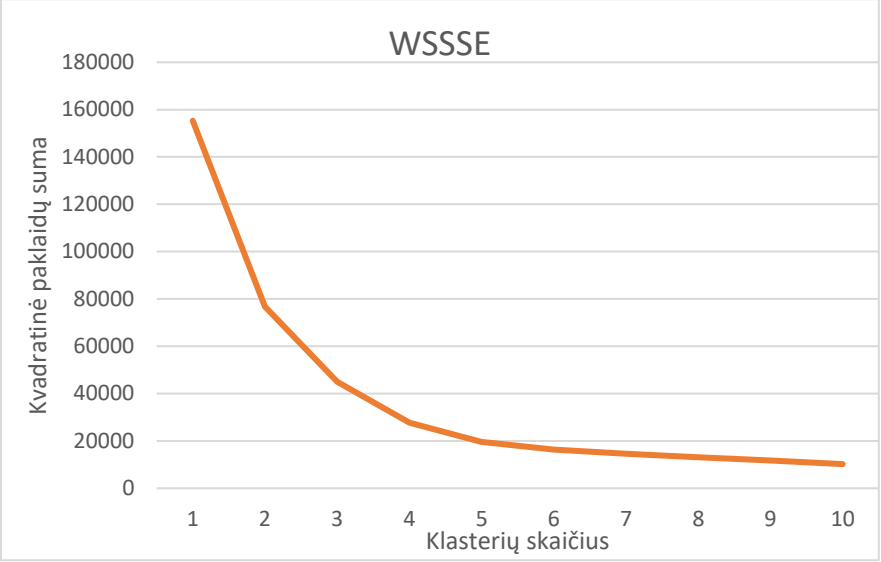
Stulpelis	Kiekis	Vidurkis	Standartinis nuokrypis	Minimumas	Maksimumas
pickup_latitude	68009668	40,7510	0,0280	39,8914	42,1598
RatecodeID	68009668	1,0318	0,3399	1	99
dropoff_longitude	68009668	-73,9734	0,0359	-75,5108	-71,7317
dropoff_latitude	68009668	40,7519	0,0323	39,8618	42,1316
payment_type	68009668	1,3392	0,4862	1	5
fare_amount	68009668	12,8424	32,8394	0,01	187440,96
extra	68009668	0,3331	0,4670	-0,5	648,87
mta_tax	68009668	0,4986	0,0316	0	80,5
tip_amount	68009668	1,7961	2,5023	0	998,14
tolls_amount	68009668	0,3140	1,6367	-5,54	1410,32
improvement_surcharge	68009668	0,3000	0,0012	0	0,3
total_amount	68009668	16,0847	33,7816	0,01	187442,26
didTip	68009668	0,6415	0,4795	0	1
pickupHour	68009668	13,6103	6,3923	0	23

Pateiktos skaitinės charakteristikos 3.2.2 lentelėje rodo, kad vidutinis kelionės atstumas yra 2,97 mylios, taksi dažniausiai vienos kelionės metu naudojasi daugiau nei vienas keleivis, kadangi vidutinis klientų skaičius yra 1,6. Taip pat įvertinta arbatpinigių palikimo suma yra 2,5.

3.3 Klasterizavimas

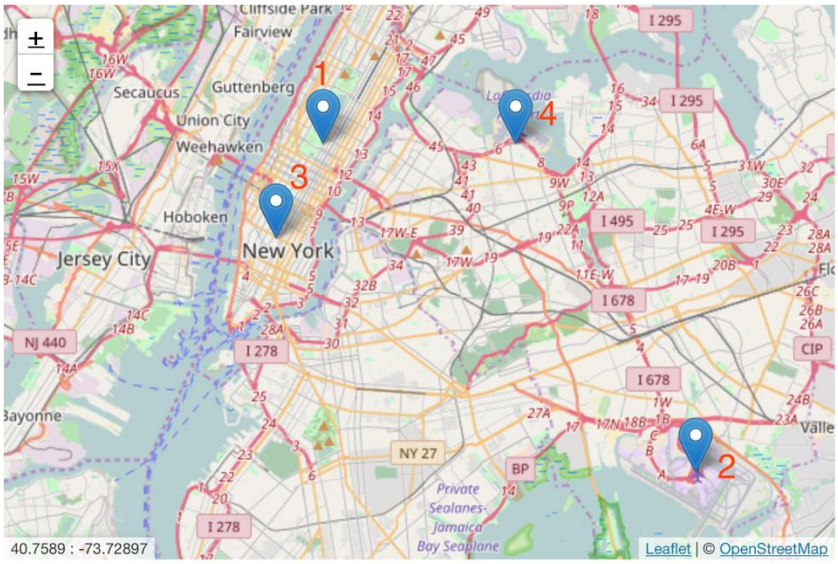
Siekiant segmentuoti Niujorko taksi rinką, pasirinktas k vidurkių metodas. Klasterizavime naudotos tik kelionės pradinio taško koordinatės. Papildomų kintamųjų naudojimo klasterizavime atsisakyta, nes tuomet klasteriai geografiškai persidengdavo. Klasterizavimo tikslas buvo išskirti konkrečius geografinius plotus tam, kad būtų galima surasti geriausią prognozavimo modelį kiekvienam rinkos segmentui.

Klasterių skaičiaus nustatymui įvertinta vidutinė kvvadratinė paklaida tarp klasterio centro ir stebėtų reikšmių. Naudojant ”alkūnės” metodą pasirinktas optimalus 4 klasterių sprendinys.



10 pav. Klasterių WSSSE metrikos

Pasirinkus klasterių sprendinį, kai vertinami kintamieji yra kelionės pradžios koordinatės, pavyko identifikuoti nepersidengiančius klasterius. Apskaičiavus klasterių centrus, pavaizduoti grafiškai 11 paveiksle.



11 pav. Klasterių centrų geografinis pasiskirstymas

Grafinė rezultatų interpretacija rodo, kad klasterių centrai yra strategiškai svarbiausiose miesto vietose. Čia 1 rodo miesto centrą, 2 – oro uostą, 3 – miesto centro pietinė dalis, 4 – John Keneddy oro uostą.

Sudarytų klasterių skaitinės charakteristikos pateiktos 3.3.1 lentelėje. Gauti rezultatai rodo, kad didžiausi klasteriai yra 1 ir 3 Niujorko miesto centre. Tai reiškia, kad dažniausiai klientai

naudojasi taksi paslaugomis, kad keliautų iš miesto centro. Klasteriai, kurie apibūdina kelionės pradžios tašką – oro uostas, yra mažiau paklausūs.

3.3.1 lentelė Duomenų pasiskirstymas tarp klasterių mokymo imtyje.

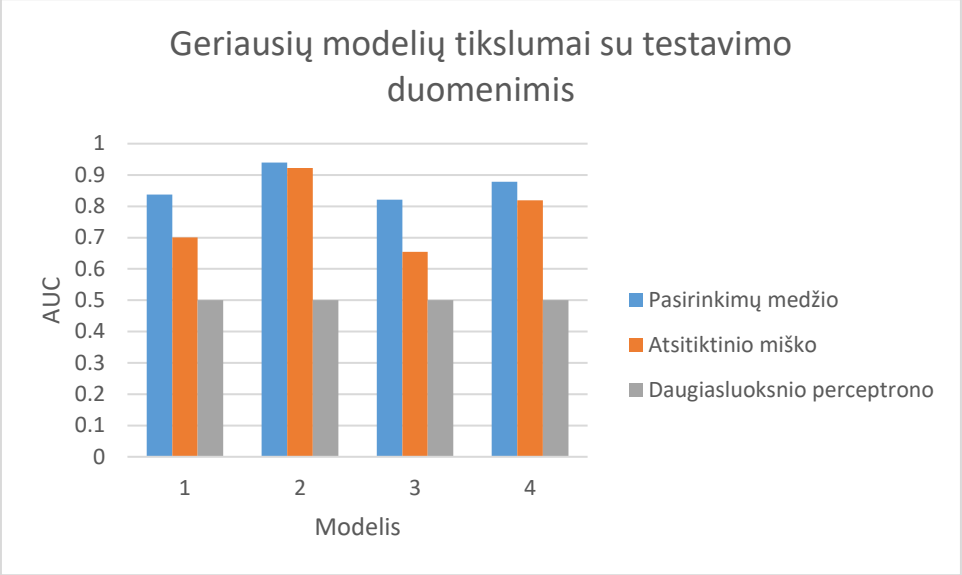
Klasteris	Įrašų skaičius	Teigiamų klasių skaičius	Teigiamų klasių procentas	Neigiamų klasių skaičius	Neigiamų klasių procentas
1	32658114	20203414	61,86%	12454700	38,14%
2	1522660	876783	57,58%	645877	42,42%
3	31713265	21129710	66,63%	10583555	33,37%
4	2115629	1421262	67,18%	694367	32,82%
Suma:	**68009668**	**43631169**	**64,15%**	**24378499**	**35,85%**

Pagal charakteristikas (3.3.1 lent.) pastebime, kad dažniausiai arbatpinigius palieka klientai, kurių kelionės pradžios taškas yra Johno Kenedžio oro uostas, tuo tarpu, klientai keliaujantys iš La Guardijos oro uosto yra nelinkę palikti arbatpinigių.

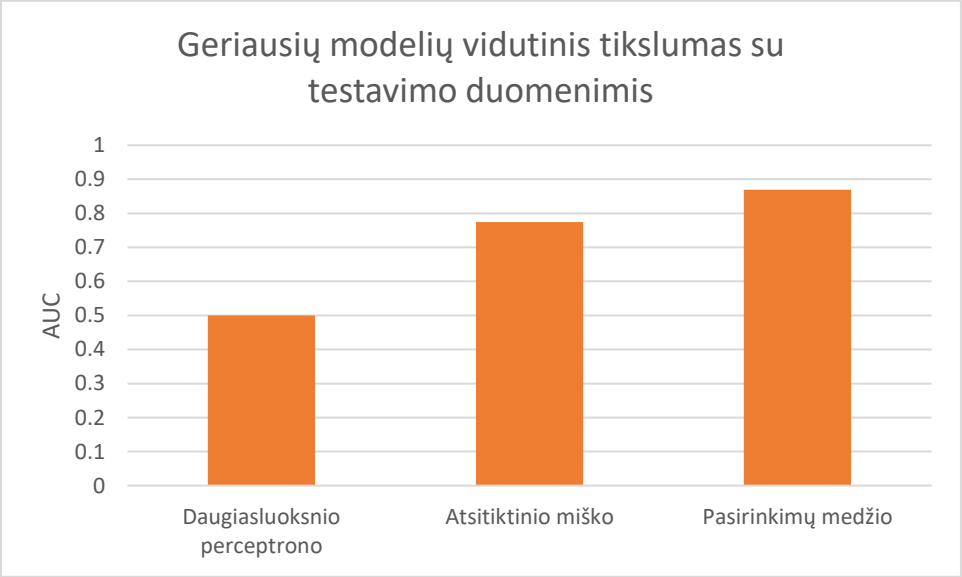
3.4 Geriausio modelio nustatymas

Suskirsčius Niujorko taksi rinką į klasterius, buvo nuspręsta taikyti klasifikavimo metodus, kurių tikslas prognozuoti arbatpinigių palikimo faktą. Siekiant įvertinti klasifikavimo metodų kokybės tikslumą, kiekvienam klasteriui atskirai įvertintos AUC reikšmės.

Tiksli klasifikavimo metodų prognozė leistų taksi vairuotojams pasirinkti labiausiai pelningus klientus, kadangi jie bus linkę sumokėti ne tik kelionės kainą, bet ir atsidėkodami palikti arbatpinigių. Todėl siekiant rasti geriausią klasifikavimo modelį kiekvienam klasteriui, panaudota 5 blokų kryžminė patikra kartu su skirtingų parametrų rinkiniu. Tokiu metodu nustačius galimus parametrų variantus sukuriama parametrų aibė su visom unikaliomis parametrų kombinacijomis, sukurtas modelis su šiais parametrais, 5 kartus apmokytas ir atliktas testavimas, nustatomas šių modelių tikslumų vidurkis bei nustatomas tiksliausiais modelis. Prognozės modelio sudarymui taikyti daugiasluoksnio perceptrono, atsitiktinių miškų ir sprendimų medžių metodai. Toliau analizuosime kiekvieno iš metodų gautus rezultatus.



11 pav. Geriausių modelių tikslumas pagal klasterį.



12 pav. Geriausių modelių vidutinis tikslumas pagal metodą visiems segmentams

3.5 Sprendimų medžio metodas

Šiam metodui buvo parinkta trijų parametrų kombinacijos. Neskaidrumo funkcijos – 2 kombinacijos, maksimalaus kategorijų skaičiaus – 2 kombinacijos, maksimalaus gylio – 2 kombinacijos. Kadangi testuojama su 5 blokų kryžmine validacija ir 4-iais klasteriais iš viso sukuriami 8 modeliai, kurių kiekvienas mokomas ir testuojamas 5 kartus. Iš lentelės (žr. 2 priedą) matome, kad geriausi rezultatai pasiekiami su 96 kategorijomis ir turint 15-a kaip maksimalų gylį, ”gini” neskaidrumo funkciją. Naudojant mokymo duomenis šis metodas yra pirmas pagal tikslumą (žr. 11, 12 pav.), tačiau naudojant testavimo duomenis (žr. 15 pav.) yra antras. Pagal 4 priedo 1

lentelę matome, kad prognozę labiausiai įtakoja kelionės kaina, nukeliautas atstumas, kelionės pradžios valanda, kelionės pabaigos koordinatės.

3.6 Atsitiktinio miško metodas

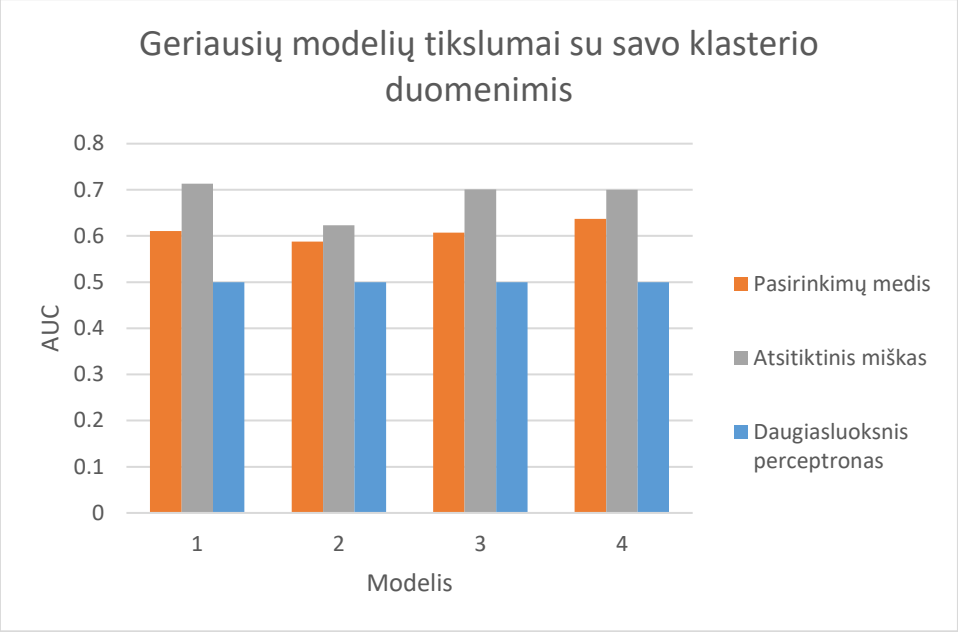
Šiam metodui buvo parinkta trijų parametrų kombinacijos. Neskaidrumo funkcijos – 2 kombinacijos, maksimalaus gylio – 2 kombinacijos, medžių skaičiaus – 2 kombinacijos. Ieškant geriausio modelio sukurta 8 modeliai, testuojant ir apmokant po 5 kartus. Iš lentelės (žr. 2 priedą) matoma, kad didžiausią įtaką modelio tikslumui daro maksimalus gylis, ir "gini" neskaidrumo funkcija mažiau svarbus medžių skaičius, tiksliausi modeliai tarp klasterių gali būti gaunami su skirtingais medžių skaičiais. Su mokymosi duomenimis šis metodas yra antras pagal tikslumą (žr. 11, 12 pav.), bet su visais duomenimis yra tiksliausias (žr. 15 pav.). Tai būtų galima paaiškinti mažesniu maksimaliu gyliu (žr. 2 priedo 1, 2 lentelės). Pagal 4 priedo 2 lentelę matome, kad kaip ir sprendimų medžio metode prognozę labiausiai įtakoja kelionės kaina, nukeliautas atstumas, kelionės pradžios valanda, kelionės pabaigos koordinatės.

3.7 Daugiasluoksnio perceptrono modelis

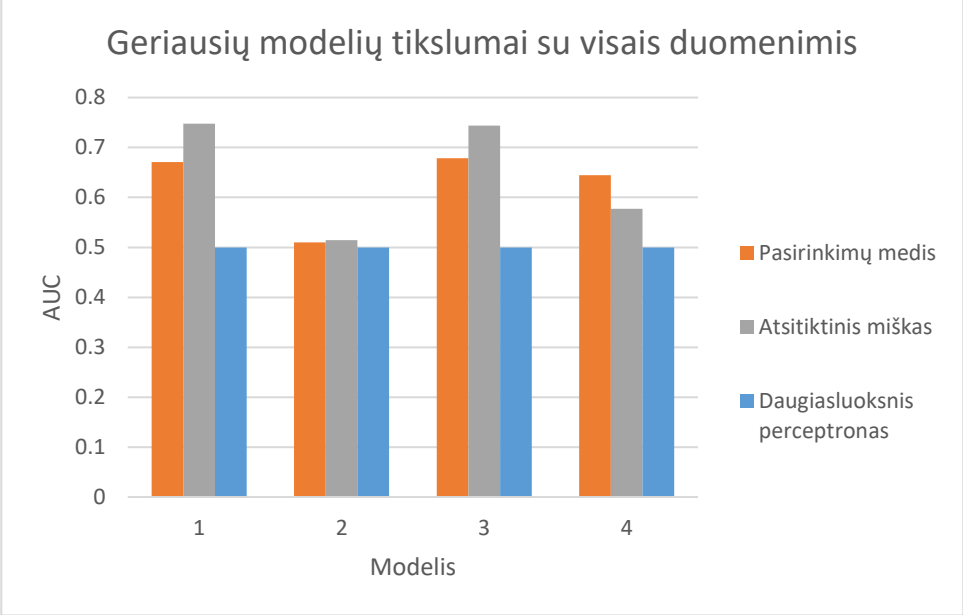
Šiam metodui buvo parinkta dviejų parametrų kombinacijos. Sluoksnių – 2 kombinacijos, maksimalaus iteracijų skaičiaus – 2 kombinacijos.Kiekvienam klasteriui sukurta po 4 modelius, apmokant ir testuojant po 5 kartus. Pagal 2 priedo 3 lentelę matome, kad didesnę įtaką modelio tikslumui daro neuroninio tinklo sluoksnių ir viršūnių skaičius negu maksimalus iteracijų skaičius. Pagal mokymosi duomenis šis metodas yra netiksliausias (žr. 11,12 pav.), tas pats matoma ir su visais duomenimis (žr. 15 pav.). Tyrimo metu pastebėta, kad šis metodas įgauną laibai aukštą tikslumą kuomet naudoja kintamieji, koreliuojantys su priklausomu kintamuoju, tačiau kuomet tokie kientamieji nenaudojami šis metodas turi prasčiausią tikslumą iš trijų naudotų metodų.

3.8 Metodų prognozės tikslumo palyginimas su visu duomenų rinkiniu

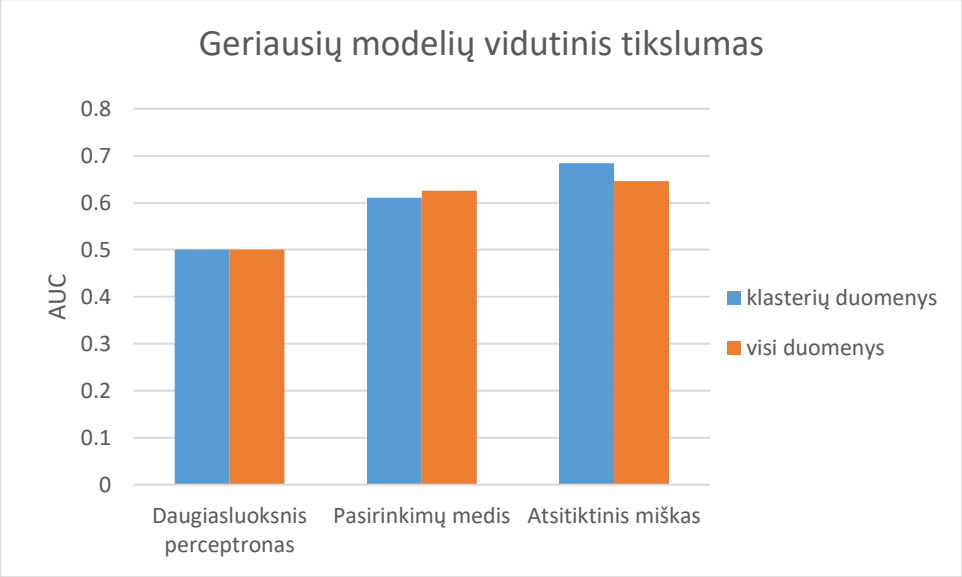
Norint patikrinti ar sudaryti ir atrinkti geriausi modeliai nebuvo permokyti, modelių įvertinimui panaudota visa duomenų imtis. Pateiksime modelių AUC reikšmių palyginimą.



13 pav. Geriausių modelių tikslumai su savo klasterio duomenimis



14 pav. Geriausių modelių tikslumai su visais duomenimis



15 pav. Geriausių modelių tikslumo palyginimas su savo klasterio duomenimis ir su visais duomenimis.

Naudojant testavimo duomenis matome, kad tikslumas sumažėjo, tačiau išlieka tos pačios tendencijos tarp metodų. Pastebėtina, kad kai kurių modelių tikslumas geresnis naudojant visus duomenis įskaitant ir kitų klasterių duomenis, nei klasifikuojant tik savo klasterio duomenis (žr. 13,14,15 pav.). Matoma, kad antrame klasteryje tikslumas prasčiausiais su visais metodais. Atsitiktinio miško metodas tiksliausias naudojant visus duomenis, nors naudojant testavimo duomenis šis metodas buvo mažiau tikslus nei sprendimų medžio metodas. Daugiasluoksnio perceptrono modelis parodė prasčiausią tikslumą. Kandagi geriausi modeliai gauti su dižiausiais modelio parametrais kaip maksimalus medžio gylis, medžių skaičius ir kt. norint pagerinti bendrą modelio tikslumą vertėtų išbandyti daugio modelio kūrimo parametrų kombinacijų.

4. IŠVADOS

- Atliekant literatūros analizę nustatyta, kad arbatpinigiai yra svarbi taksi vairuotojų, kaip ir kitų, paslaugas teikiančių, profesijų atstovų, pajamų dalis. Taksi vairuotojų uždarbis nėra tolygiai pasiskirstęs, priklausomas, kaip ir pačios taksi kelionės, nuo geografinių savybių.
- Didžiulis duomenų kiekis – virš 700 milijonų eilučių surinkti naudojant didžiųjų duomenų analitikos įrankius. Skirtingų laikotarpių duomenys sujungti į bendrumą. Modelių apmokymui skirti duomenys išvalyti – pašalintos neteisingos reikšmės, skirtingų tipų duomenys aprašantys tuos pačius bruožus apibendrinti, duomenys neturintys reikšmių apie tam tikrus bruožus papildyti.
- Pasitelkus klasterizavima gauti keturi geografiniai regionai. Du iš jų padalina miesto centrą į dvi dalis. Kiti du sutampa su miesto oro uostais. Šie klasteriai sutampa su taksi kelionių tankumo centrais. Pagal šiuos klasterius galima segmentuoti taksi rinką nustatant, kokios duomenų savybės yra kiekviename, kokia vidutinė kelionės kaina ar astumas. Turint tokius duomenis gali susidaryti nuomonę kuriame iš segmentų geriausia dirbti, norint uždirbti didžiausias pajamas. Taip pat galima susidaryti strategijas, leidžiančias keisti vietas, kuriose ieškoma laukiama klientų taipogi siekiant padidinti uždirbamas pajamas. Pagal apžvelgtą literatūrą, geriausia strategija nulemia daugiausiai uždirbančius vairuotojus. Tačiau dirbant rinkoje ši informacija įgaunama su patirtimi.
- Sukurti klasifikavimo modeliai suteikia papildomą priemonę – gebėjima prognozuoti ar klientas mokės arbatpinigius ar ne. Vairuotojui naudojančiam tokią priemonę tai suteiktų pranašumą prieš kitus vairuotojus. Galimybė rinktis tik klientus, kurie palieka arbatpinigių leistų vairuotojui maksimizuoti pajamas. Tiriant modelių tikslumą taip pat nustatyta, kad skirtinguose klasteriuose modelių tikslumai skiriasi. Tai nulemia, kad tokių modelių naudojimo efektyvumas būtų didžiausiais naudojant juos tik tam tikruose klasteriuose, tam tikruose rinkos segmentuose. Iš tirtų metodų geriausius rezultatus parodė atsitiktinio miško metodas, šiek tiek prasčiau sprendimų medžio metodas. Daugiasluoksnio perceptrono metodas parodė prasčiausius rezultatus.
- Tolimesniam modelio tobulinimui būtų naudinga įtraukti papildomų išorinių faktorių, kurie leistų identifikuoti taksi vairuotojus, tai leistų sudaryti tikslesnes prognozes.

5. LITERATŪROS SĄRAŠAS

(2017 m. 05 12 d.). Nuskaityta iš Apache spark: https://spark.apache.org

Azar, O. (2008). Strategic behavior and social norms in tipped service industries. *B.E. J. Econ. Anal. Policy 8. Article 7.*

Azar, O. (2011). Business strategy and the social norm of tipping. *Journal of Economic Psychology, 32*, 515–525.

Azar, O. H. (2003). The implications of tipping for economics and management. *International Journal of Social Economics, 30*(10), 1084-1093.

Becker, C., Bradley, G., & Zantow, K. (2012). The underlying dimensions of tipping behavior: an exploration, confirmation and predictive model. *Internation Journal of Hospitality Management, 31*(1), 247–256.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees.* Belmont, CA: Wadsworth Int.

Brunsdon, C., Fotheringham, A. S., & Charlton, M. E. (1996). Geographically weighted regression: a method for exploring spatial nonstationarity. . *Geographical Analysis, 281e298.*(28(4)).

Cardozo, O. D., García-Palomares, J. C., & Gutierrez, J. (2012). Application of geographically weighted regression to the direct forecasting of transit ridership at station-level. . *Applied Geography, 548e558.*(34).

Cortes, C., & Mohri, M. (2003). AUC Optimization vs. Error Rate Minimization. *NIPS, 9*, 10.

Drineas, P., Frieze, A., Kannan, R., Vempala, S., & Vinay, V. (1999). Clustering large graphs via the singular value decomposition. . *Machine Learn. , 56*(1–3), 9–33.

Frey, B., & Jegen, R. (2001). Motivation crowding theory. . *Journal of Economic Surveys, 15*, , 589–611.

Haire, A. R., & Machemehl, R. B. (2007). Impact of rising fuel prices on US transit ridership. *Transportation Research Record: Journal of the Transportation Research Board, 1992(1), 11e19.*

Hamberger, C. B., & Chatterjee, A. (1987). Effects of fare and other factors on express bus ridership in a medium-sized urban area.

Jain, A. K. (2009). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 651-654.

Jain, A. K., & Dubes, R. C. (1988). Algorithms for Clustering Data. . *Prentice Hall.*

James, H. (2005). Why did you do that? An economic examination of the effect of extrinsic compensation on intrinsic motivation and performance. . *Journal of Economic Psychology, 26*, 549–566.

Kain, J. F., & Liu, Z. (1999). Secrets of success: assessing the large increases in transit ridership achieved by Houston and San Diego transit providers. . *Transportation Research Part A: Policy and Practice, 33(7), 601e624.*

Kanafani, A. (1983). Transportation demand analysis. .

Kanal, L. N. (1979). Problem-solving methods and search strategies for pattern recognition. *IEEE Trans. Patt. Anal. Mach. Intell., PAMI-1*, 193-201.

Kuby, M., Barranda, A., & Upchurch, C. (2004). Factors influencing light-rail station boardings in the United States. *Transportation Research Part A: Policy and Prac- tice, 38(3), 223e247.*

Kulkarni, A. V., & Kanal, L. N. (1976). An optimization appearch to hierarchical classifier design. *Proc. 3rd Int. Joint Conf. Pattern Recognittion.* San Diego.

Kurzynski, M. W. (1983). Decision rules for a hierarchical classifier. *Pattern Recog. Lett., 1*, 305-310.

Kurzynski, M. W. (1983). The optimal strategy of a tree classifier. *Pattern Recog. , 16*, 81-87.

Landeweerd, G., Timmers, T., Gersema, E., Bins, M., & Halic, M. (1983). Binary tree versus single level tree classification of white blood cells. *Pattern Recognition, 16*, 571-577.

Li, X., & Dubes, R. C. (1986). Tree classifier design with a Permutation statistic. *Pattern Recognition, 19*, 229-235.

Liaw, A., & Wiener, M. (2002). Classification and Regression by randomForest. *R News, 2*, 18-19.

Lynn, M. (2009). Individual differences in self-attributed motives for tipping: antecedants, consequences, and implications. *International Journal of Hospitality Management, 28*(3), 432–438.

Lynn, M. (2015b). Explanations for service gratuities and tipping: Evidence from individual differences in tipping motivations and tendencies. . *Journal of Behavioral and Experimental Economics , 55*, 65–71.

Lynn, M. (2015b.). Explanations for service gratuities and tipping: Evidence from individual differences in tipping motivations and tendencies. . *Journal of Behavioral and Experimental Economics , 55*, 65–71.

Lynn, M. (2016). Motivations for tipping: How they differ across more and less frequently tipped services. *Journal of Behavioral and Experimental Economics*, 39.

Lynn, M. (2016). Motivations for tipping: How they differ across more and less frequently tipped services. *Journal of Behavioral and Experimental Economics*, 39-48.

Lynn, M., & Withiam, G. (2008). Tipping and its alternatives: business considerations and directions for research. *Journal of Services Marketing*, 328-332.

Lynn, M., Jabbour, P., & Kim, W. (2012). Who uses tips as a reward for service and when? An examination of potential moderators of the service-tipping relationship. *Journal of Economic Psychology, 33*(1), 90–103.

McNally, M. G. (2008). The four step model. *UC Irvine: Center for Activity Systems Analysis.* .

Meila, M. (2006). The uniqueness of a good optimum for k-means. *In: Proc. 23rd Internat. Conf. Machine Learning, pp. ,* (p. 625–632).

Mui, J., & Fu, K. S. (1980). Automated classification of nucleated blood cells using a binary tree classifier. *IEEE Trans. Patt. Anal. Mach. Intell., PAMI-2*, 429-443.

Nadler, M. (1970). Error and reject rates in a hierarchical pattern recognizer. *IEEE Trans. Comput., C-19*, 1598-1601.

Qian, X., & Ukkusuri, S. V. (2015). Spatial variation of the urban taxi ridership using GPS data. *Applied Geography, 59*, 31-42.

Rafiq, M. Y., Bugmann, G., & Easterbrook, D. J. (2001). Neural network design for engineering applications. *Computers and Structures, 79*, 1541-1552.

Safavian, S. R., & Landgrebe, D. (1991). A Survey of Decision Tree Classifier Methodology. *IEEE Transactions on Systems, Man, and Cybernetics, 21*(3).

Shoro, A. G., & Soomro, T. R. (2015). Big Data Analysis: Ap Spark Perspective. *Global Journal of Computer Science and Technology: C Software & Data Engineering, 15*(1), 1-9.

Star, N. (1988). *The International Guide to Tipping.* New York: Berkley Books.

Swain, P., & Hauska, H. (1977). The decision tree classifier design and potential. *IEEE Trans. Geosci. Electron, GE-15*, 142-147.

Tao, S., Corcoran, J., Mateo-Babiano, I., & Rohde, D. (2014). Exploring Bus Rapid Transit passenger travel behaviour using big data. *Applied Geography, 90e104.*(53).

Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. . *J. Roy. Statist. Soc. B*, 411–423.

TLC Trip Record Data. (2017 m. 05 01 d.). Nuskaityta iš NYC Taxi and Limousine Commission : http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

Wang, Q. R., & Suen, C. Y. (1987). Large tree classifier with heuristic search and global training. *IEEE Trans. Patt. Anal. Mach. Intell., PAMI-9*, 91-102.

Weinstein, A. (2010). *Handbook of Market Segmentation Strategic Targeting for Business and Technology Firms.* New York: Routledge.

Whalen, J., Douglas, A., & O'Neill, M. (2014). What's in a tip? The creation and refinement of a restaurant-tipping motivations scale: A consumer perspective. *International Journal of Hospitality Management, 37*, 121–130.

Wu, C., Landgrebe, D., & Swain, P. (1975). The decision tree approach to classification. *School Elec. Eng., RE-EE 75-17*.

Y., G., P., X., L., L., H., L., S., L., & H., Q. (2012). Visualization of Taxi Drivers' Income and Mobility Intelligence. *International Symposium on Visual Computing*, 275-284.

Yang, H., Lau, Y. W., Wong, S. C., & Lo, H. K. (2000). A macroscopic taxi model for passenger demand, taxi utilization and level of services. *Transportation, 317e340.* (27).

You, K. C., & Fu, K. S. (1976). An approach to the design of a linear binary tree clasiffier. *Proc. 4rd Symp. Machine Processing of Remotely Sensed Data*.

1 PRIEDAS. DUOMENŲ RINKINIO STRUKTŪRA

1 lentelė duomenų rinkinio stulpelių struktūra.

Stulpelio pavadinimas	Reikšmės tipas	Paaiškinimas
vendor_name	Simbolinis	Įmonės, suteikusios duomenis, pavadinimas
Trip_Pickup_DateTime	Data	Kelionės pradžios data
Trip_Dropoff_DateTime	Data	Kelionės pabaigos data
Passenger_Count	Sveikas skaičius	Keleivių skaičius
Trip_Distance	Skaičius	Kelionės atstumas
Start_Lon	Skaičius	Kelionės pradžios geografinė ilguma
Start_Lat	Skaičius	Kelionės pradžios geografinė platuma
Rate_Code	Sveikas skaičius	Kelionės tipo kodas
store_and_forward	Simbolinis	Ar kelionė buvo išsaugota prietaiso atmintyje
End_Lon	Skaičius	Kelionės pabaigos geografinė ilguma
End_Lat	Skaičius	Kelionės pabaigos geografinė platuma
Payment_Type	Simbolinis	Mokėjimo tipas
Fare_Amt	Skaičius	Kelionės kaina
surcharge	Skaičius	Tobulinimų mokestis
mta_tax	Skaičius	Kelių inspekcijos papildomas mokestis
Tip_Amt	Skaičius	Arbatpinigių suma
Tolls_Amt	Skaičius	Kelių mokesčių suma
Total_Amt	Skaičius	Bendra suma

2 lentelė duomenų rinkinio stulpelių struktūra.

Stulpelio pavadinimas	Reikšmės tipas	Paaiškinimas
vendor_id	Sveikas skaičius	Įmonės, suteikusios duomenis, identifikacinis numeris
pickup_datetime	Data	Kelionės pradžios data
dropoff_datetime	Data	Kelionės pabaigos data
passenger_count	Sveikas skaičius	Keleivių skaičius
trip_distance	Skaičius	Kelionės atstumas
pickup_longitude	Skaičius	Kelionės pradžios geografinė ilguma
pickup_latitude	Skaičius	Kelionės pradžios geografinė platuma
rate_code	Sveikas skaičius	Kelionės tipo kodas
store_and_fwd_flag	Simbolinis	Ar kelionė buvo išsaugota prietaiso atmintyje
dropoff_longitude	Skaičius	Kelionės pabaigos geografinė ilguma
dropoff_latitude	Skaičius	Kelionės pabaigos geografinė platuma
payment_type	Simbolinis	Mokėjimo tipas
fare_amount	Skaičius	Kelionės kaina

Stulpelio pavadinimas	Reikšmės tipas	Paaiškinimas
surcharge	Skaičius	Tobulinimų mokestis
mta_tax	Skaičius	Kelių inspekcijos papildomas mokestis
tip_amount	Skaičius	Arbatpinigių suma
tolls_amount	Skaičius	Kelių mokesčių suma
total_amount	Skaičius	Bendra suma

3 lentelė duomenų rinkinio stulpelių struktūra.

Stulpelio pavadinimas	Reikšmės tipas	Paaiškinimas
VendorID	Sveikas skaičius	Įmonės, suteikusios duomenis, identifikacinis numeris
tpep_pickup_datetime	Data	Kelionės pradžios data
tpep_dropoff_datetime	Data	Kelionės pabaigos data
passenger_count	Sveikas skaičius	Keleivių skaičius
trip_distance	Skaičius	Kelionės atstumas
RatecodeID	Sveikas skaičius	Kelionės tipo kodas
store_and_fwd_flag	Simbolinis	Ar kelionė buvo išsaugota prietaiso atmintyje
PULocationID	Sveikas skaičius	Kelionės pradžios vietos kodas
DOLocationID	Sveikas skaičius	Kelionės pabaigos vietos kodas
payment_type	Simbolinis	Mokėjimo tipas
fare_amount	Skaičius	Kelionės kaina
extra	Skaičius	Sumuoti papildomi mokesčiai
mta_tax	Skaičius	Kelių inspekcijos papildomas mokestis
tip_amount	Skaičius	Arbatpinigių suma
tolls_amount	Skaičius	Kelių mokesčių suma
improvement_surcharge	Skaičius	Tobulinimų mokestis
total_amount	Skaičius	Bendra suma

2 PRIEDAS. MODELIŲ TIKSLUMAS NAUDOJANT MOKYMOSI DUOMENIS

1 lentelė Sprendimų medžio modelių rezultatai su mokymosi duomenimis

Klasteris	Neskaidrumo funkcija	Maks. Kategorijos	Maks. Gylis	AUC
1	entropy	64	6	0,712253279
1	entropy	96	6	0,716710812
1	gini	64	6	0,723475573
1	gini	96	6	0,729481032
1	entropy	64	15	0,81094584
1	gini	64	15	0,82023747
1	entropy	96	15	0,824816089
1	gini	96	15	0,837039961
2	entropy	64	6	0,897915771
2	entropy	96	6	0,897971163
2	gini	96	6	0,900286015
2	gini	64	6	0,900555585
2	entropy	64	15	0,938146584
2	gini	64	15	0,938568222
2	entropy	96	15	0,938575424
2	gini	96	15	0,939357607
3	entropy	64	6	0,683041567
3	entropy	96	6	0,696097434
3	gini	64	6	0,69993207
3	gini	96	6	0,714667674
3	entropy	64	15	0,797982406
3	gini	64	15	0,805655548
3	entropy	96	15	0,809342645
3	gini	96	15	0,820875612
4	entropy	96	6	0,802168499
4	gini	96	6	0,803434135
4	entropy	64	6	0,808041393
4	gini	64	6	0,810370858
4	entropy	64	15	0,872242866
4	gini	64	15	0,875624541
4	entropy	96	15	0,876437289
4	gini	96	15	0,878535219

2 lentelė Atsitiktinio miško modelių rezultatai su mokymosi duomenimis

Klasteris	Neskaidrumo funkcija	Maks. Gylis	Medžių skaičius	AUC
1	entropy	4	40	0,59769507
1	gini	4	40	0,598140941
1	gini	4	20	0,629251268
1	entropy	4	20	0,634440415
1	entropy	10	40	0,69317993
1	entropy	10	20	0,697355963
1	gini	10	40	0,697463782
1	gini	10	20	0,700589518
2	entropy	4	40	0,864132275
2	gini	4	40	0,86635026
2	entropy	4	20	0,868096145
2	gini	4	20	0,86886419
2	entropy	10	20	0,919547519
2	gini	10	20	0,920715646
2	entropy	10	40	0,921721765
2	gini	10	40	0,921895934
3	gini	4	20	0,588814698
3	entropy	4	20	0,589062685
3	gini	4	40	0,589119723
3	entropy	4	40	0,589391817
3	gini	10	20	0,647765063
3	entropy	10	40	0,64954187
3	entropy	10	20	0,651730893
3	gini	10	40	0,654486681
4	entropy	4	40	0,675110607
4	entropy	4	20	0,679566365
4	gini	4	40	0,688125043
4	gini	4	20	0,702138772
4	entropy	10	40	0,810265197
4	gini	10	40	0,813484252
4	entropy	10	20	0,81577037
4	gini	10	20	0,818962563

3 lentelė Daugiasluoksnio perceptrono modelių rezultatai su mokymosi duomenimis

Klasteris	Sluoksniai	Iteracijų sk.	AUC
1	[16, 5, 4, 3]	50	0,5
1	[16, 5, 4, 3]	100	0,5
1	[16, 20, 16, 3]	50	0,5
1	[16, 20, 16, 3]	100	0,5
2	[16, 5, 4, 3]	50	0,5
2	[16, 20, 16, 3]	50	0,5
2	[16, 5, 4, 3]	100	0,5
2	[16, 20, 16, 3]	100	0,5
3	[16, 5, 4, 3]	50	0,5
3	[16, 20, 16, 3]	50	0,5
3	[16, 5, 4, 3]	100	0,5
3	[16, 20, 16, 3]	100	0,5
4	[16, 5, 4, 3]	50	0,5
4	[16, 20, 16, 3]	50	0,5
4	[16, 5, 4, 3]	100	0,5
4	[16, 20, 16, 3]	100	0,5

3 PRIEDAS. MODELIŲ TIKSLUMAI NAUDOJANT TESTINIUS DUOMENIS

1 lentelė Modelių tikslumai naudojant testinius duomenis

Metodas	Klasteris	Duomenys	AUC
Sprendimų medis	1	Klasterio	0,610686078
	1	Visi	0,670994467
	2	Klasterio	0,587964936
	2	Visi	0,509959348
	3	Klasterio	0,607061637
	3	Visi	0,678193362
	4	Klasterio	0,636959833
	4	Visi	0,64461758
Atsitiktinis miškas	1	Klasterio	0,713351269
	1	Visi	0,747887751
	2	Klasterio	0,622947336
	2	Visi	0,514684017
	3	Klasterio	0,700547043
	3	Visi	0,743884514
	4	Klasterio	0,700283735
	4	Visi	0,576922961
Daugiasluoksnis perceptronas	1	Klasterio	
	1	Visi	
	2	Klasterio	
	2	Visi	
	3	Klasterio	
	3	Visi	
	4	Klasterio	
	4	Visi	0,5

4 PRIEDAS. KINTAMŲJŲ SVARBUMAS MODELIUI

1 lentelė Sprendimų medžio kintamųjų svarbumas prognozės nustatymui

sprendimų medis				
klasteris	**1**	**2**	**3**	**4**
tpep_pickup_datetime	0,000362	0,004327	0,000547	0,013454
tpep_dropoff_datetime	0,000026	0,000084	0,000228	0,001338
VendorID	0,001180	0,000784	0,001636	0,001012
passenger_count	0,000079	0,000655	0,000159	0,001485
trip_distance	0,184893	0,087463	0,211947	0,254216
pickup_longitude	0,003101	0,002096	0,003107	0,007885
pickup_latitude	0,003137	0,005168	0,005846	0,011311
RateCodeID	0,000255	0,040206	0,001859	0,011218
dropoff_longitude	0,006104	0,020995	0,007234	0,029182
dropoff_latitude	0,015120	0,008021	0,016488	0,028780
extra	0,012361	0,048337	0,012246	0,022218
mta_tax	0,000005	0,000155	0,000052	0,000117
tolls_amount	0,000572	0,088380	0,001190	0,073418
improvement_surcharge	0,000027	0,000022	0,000000	0,000013
total_amount	0,750557	0,678389	0,700295	0,504562
pickupHour	0,022223	0,014917	0,037167	0,039791

2 lentelė Atsitiktinio miško kintamųjų svarbumas prognozės nustatymui

atsitiktinis miškas				
klasteris	**1**	**2**	**3**	**3**
tpep_pickup_datetime	0,000381	0,001138	0,000654	0,001315
tpep_dropoff_datetime	0,000339	0,000850	0,000679	0,001449
VendorID	0,001311	0,001501	0,002193	0,001020
passenger_count	0,002947	0,002224	0,004889	0,002189
trip_distance	0,176925	0,079696	0,219828	0,239382
pickup_longitude	0,016535	0,003066	0,010708	0,037787
pickup_latitude	0,009402	0,004783	0,033435	0,017962
RateCodeID	0,001421	0,043128	0,002173	0,005520
dropoff_longitude	0,021989	0,046883	0,019095	0,128490
dropoff_latitude	0,031977	0,030441	0,031728	0,039950
extra	0,017024	0,010225	0,016348	0,008444
mta_tax	0,000480	0,001428	0,000798	0,000936
tolls_amount	0,002470	0,041877	0,002374	0,047234
improvement_surcharge	0,000176	0,000051	0,000000	0,000072
total_amount	0,700866	0,724838	0,628819	0,445754
pickupHour	0,015758	0,007870	0,026279	0,022496

5 PRIEDAS. PROGRAMINIS KODAS

```
import org.apache.spark.sql.types.{StructType, IntegerType, StringType, DoubleType, TimestampType}
import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.types.{LongType, StructField, StructType}
import org.apache.spark.sql.Row
val spark = org.apache.spark.sql.SparkSession.builder
    .config("spark.kryo.registrator", "org.bdgenomics.adam.serialization.ADAMKryoRegistrator")
    .master("local")
    .getOrCreate;
val didTip: (Double => Double) = (arg: Double) => {if (arg > 0) 1.0 else 0.0 }
val didTipUdf = udf(didTip)
val initial = spark.read.format("com.databricks.spark.csv")
            .option("header", "true")
            .option("mode", "DROPMALFORMED")
             .option("inferSchema", "true")
            .load(f"notebooks/taxai/yellow_tripdata_2016-01.csv")
val before_count = initial.count
val initialReduced = initial.filter($"pickup_longitude" >= -75.528844)
             .filter($"pickup_longitude" <= -71.658250)
             .filter($"pickup_latitude" >= 39.858698)
             .filter($"pickup_latitude" <= 42.188717)
             .filter($"dropoff_longitude" >= -75.528844)
             .filter($"dropoff_longitude" <= -71.658250)
             .filter($"dropoff_latitude" >= 39.858698)
             .filter($"dropoff_latitude" <= 42.188717)
val after_count = initialReduced.count
val withDidTip = initial.withColumn("didTip", didTipUdf(col("tip_amount")))
val withPickupHour = withDidTip.withColumn("pickupHour", hour(col("tpep_pickup_datetime")))
val withTime = withPickupHour
            .withColumn("tpep_pickup_datetime_int",
withPickupHour("tpep_pickup_datetime").cast(IntegerType))
            .withColumn("tpep_dropoff_datetime_int",
withPickupHour("tpep_dropoff_datetime").cast(IntegerType))
            .drop("tpep_pickup_datetime")
            .drop("tpep_dropoff_datetime")
            .withColumnRenamed("tpep_pickup_datetime_int","tpep_pickup_datetime")
            .withColumnRenamed("tpep_dropoff_datetime_int","tpep_dropoff_datetime")
withTime.createOrReplaceTempView("data_temptable")
spark.sql("drop table if exists taxi_inital")
```

```
spark.sql(f"create table taxi_initial as select * from data_temptable")
println("before count: " + before_count)
println("after count: " + after_count)
var before_count: Long = 0
var after_count: Long = 0
for(j <- 2 to 6) {
  val initial = spark.read.format("com.databricks.spark.csv")
                .option("header", "true")
                .option("mode", "DROPMALFORMED")
                 .option("inferSchema", "true")
                .load(f"notebooks/taxai/yellow_tripdata_2016-$j%02d.csv")
  before_count = before_count + initial.count()
  val initialReduced = initial.filter($"pickup_longitude" >= -75.528844)
                 .filter($"pickup_longitude" <= -71.658250)
                 .filter($"pickup_latitude" >= 39.858698)
                 .filter($"pickup_latitude" <= 42.188717)
                 .filter($"dropoff_longitude" >= -75.528844)
                 .filter($"dropoff_longitude" <= -71.658250)
                 .filter($"dropoff_latitude" >= 39.858698)
                 .filter($"dropoff_latitude" <= 42.188717)
  after_count = after_count + initialReduced.count
  val withDidTip = initial.withColumn("didTip", didTipUdf(col("tip_amount")))
  val withPickupHour = withDidTip.withColumn("pickupHour", hour(col("tpep_pickup_datetime")))
  val withTime = withPickupHour
                .withColumn("tpep_pickup_datetime_int",
withPickupHour("tpep_pickup_datetime").cast(IntegerType))
                .withColumn("tpep_dropoff_datetime_int",
withPickupHour("tpep_dropoff_datetime").cast(IntegerType))
                .drop("tpep_pickup_datetime")
                .drop("tpep_dropoff_datetime")
                .withColumnRenamed("tpep_pickup_datetime_int","tpep_pickup_datetime")
                .withColumnRenamed("tpep_dropoff_datetime_int","tpep_dropoff_datetime")
  withTime.write.mode(org.apache.spark.sql.SaveMode.Append).insertInto("taxi_initial")
  println("before count " + before_count)
  println("after count " + after_count)
}
initial.schema.fields.foreach { p => println(p.name + " " + initial.filter(p.name + " is null").count) }
import org.apache.spark.sql.types.{StructType, IntegerType, StringType, DoubleType, TimestampType}
import org.apache.spark.sql.DataFrame
```

```
import org.apache.spark.sql.types.{LongType, StructField, StructType}
import org.apache.spark.sql.Row
val spark = org.apache.spark.sql.SparkSession.builder
    .config("spark.kryo.registrator", "org.bdgenomics.adam.serialization.ADAMKryoRegistrator")
    .master("local")
    .getOrCreate;
val didTip: (Double => Double) = (arg: Double) => {if (arg > 0) 1.0 else 0.0 }
val didTipUdf = udf(didTip)
for(i <- 9 to 16) {
   for(j <- 1 to 12) {
      val initial = spark.read.format("com.databricks.spark.csv")
                  .option("header", "true")
                  .option("mode", "DROPMALFORMED")
                   .option("inferSchema", "true")
                  .load(f"notebooks/taxai/yellow_tripdata_20$i%02d-$j%02d.csv")
      val withDidTip = initial.withColumn("didTip", didTipUdf(col("tip_amount")))
      val withPickupHour = withDidTip.withColumn("pickupHour", hour(col("tpep_pickup_datetime")))
      val withTime = withPickupHour
                  .withColumn("tpep_pickup_datetime_int",
withPickupHour("tpep_pickup_datetime").cast(IntegerType))
                  .withColumn("tpep_dropoff_datetime_int",
withPickupHour("tpep_dropoff_datetime").cast(IntegerType))
                  .drop("tpep_pickup_datetime")
                  .drop("tpep_dropoff_datetime")
                  .withColumnRenamed("tpep_pickup_datetime_int","tpep_pickup_datetime")
                  .withColumnRenamed("tpep_dropoff_datetime_int","tpep_dropoff_datetime")
      val vectors = withTime.rdd.map(r => Vectors.dense(r.getDouble(5),
                                          r.getDouble(6)))
      vectors.persist(StorageLevel.MEMORY_AND_DISK_SER)

      /* duomenim, kurie neturi VendorID
      val indexer = new StringIndexer()
        .setInputCol("vendor_name")
        .setOutputCol("VendorID")
      val indexed = indexer.fit(withTime).transform(withTime)
      // duomenim, kuriu payment_type string tipo
      val indexer2 = new StringIndexer()
        .setInputCol("payment_type")
        .setOutputCol("payment_type_int")
```

```
        val indexed2 = indexer2.fit(indexed).transform(indexed)
        val withPayments = indexed2.drop("payment_type").withColumnRenamed("payment_type_int","payment_type")
        */
        val clusters = KMeansModel.load(sc, "notebooks/taxai/models_v3/km/4")
        val predictions = clusters.predict(vectors)
        val predictionsDF = predictions.toDF("cluster")
        val rows = withTime.rdd.zip(predictionsDF.rdd).map{
          case (rowLeft, rowRight) => Row.fromSeq(rowLeft.toSeq ++ rowRight.toSeq)
        }

        val schema = StructType(withPickupHour.schema.fields ++ predictionsDF.schema.fields)

        val clustered: DataFrame = spark.createDataFrame(rows, schema)
        vectors.unpersist()
        // Duomenim neturintiem extra lauko
//        clustered.withColumn("extra", lit(0.0)).createOrReplaceTempView("data_temptable")
        clustered.write.mode(org.apache.spark.sql.SaveMode.Append).insertInto("taxi_all")
      }
    }
    import org.apache.spark.storage.StorageLevel
    import org.apache.spark.mllib.linalg.Vectors
    import org.apache.spark.mllib.clustering.{KMeans}
    import org.apache.spark.sql.DataFrame
    import org.apache.spark.sql.Row
    import org.apache.spark.sql.types.{StructField, StructType, LongType}
    val spark = org.apache.spark.sql.SparkSession.builder
        .config("spark.kryo.registrator", "org.bdgenomics.adam.serialization.ADAMKryoRegistrator")
        .master("local")
        .getOrCreate;
    val data = spark.sql("SELECT * FROM taxi_initial")
    data.count
    val vectors = data.rdd.map(r => Vectors.dense(r.getDouble(5), //keliones pradzios coordinates
                                  r.getDouble(6)))
    vectors.persist(StorageLevel.MEMORY_AND_DISK_SER)
    val numIterations = 20
    var wsseArray: Array[(Int, Double)] = Array()
    for( i <- 1 to 10){
      val tempClusters = KMeans.train(vectors, i, numIterations)
```

```
  val WSSSE = tempClusters.computeCost(vectors)
  println( "Value of WSSSE: " + WSSSE + " i: " + i );
  wsseArray = wsseArray :+ ((i, WSSSE))
 }
 display(wsseArray)
 val clusters = KMeans.train(vectors, 4, numIterations)
 clusters.save(sc, "notebooks/taxai/models_v3/km/4")
 val predictions = clusters.predict(vectors)
 val predictionsDF = predictions.toDF("cluster")
 val rows = data.rdd.zip(predictionsDF.rdd).map{
  case (rowLeft, rowRight) => Row.fromSeq(rowLeft.toSeq ++ rowRight.toSeq)
 }

 val schema = StructType(data.schema.fields ++ predictionsDF.schema.fields)

 val clustered: DataFrame = spark.createDataFrame(rows, schema)
 clustered.createOrReplaceTempView("clustered_temp")
 spark.sql("DROP TABLE IF EXISTS taxi_2016_clustered")
 spark.sql("CREATE TABLE taxi_2016_clustered AS SELECT * from clustered_temp")
 clusters.clusterCenters
 val values = List(List(40.770324472464246, -73.96967481337708) ,List(40.64703491515005, -
73.78456990487321) ,List(40.73482343371666, -73.99321540947628), List(40.77010862599982,-
73.87419179965413)).map(x =>(x(0), x(1)))
 import spark.implicits._
 val df = values.toDF
 GeoPointsChart(df, latLonFields=Some(("_1","_2")))
 import org.apache.spark.ml.classification.MultilayerPerceptronClassifier
 import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
 import org.apache.spark.ml.feature.{StringIndexer, VectorIndexer, VectorAssembler}
 import org.apache.spark.ml.classification.BinaryLogisticRegressionSummary
 import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator
 import org.apache.spark.ml.linalg.DenseVector
 import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics
 import org.apache.spark.ml.tuning.{ParamGridBuilder, CrossValidator, CrossValidatorModel}
 val spark = org.apache.spark.sql.SparkSession.builder
     .config("spark.kryo.registrator", "org.bdgenomics.adam.serialization.ADAMKryoRegistrator")
     .master("local")
     .getOrCreate;
 val data = spark.sql("SELECT * FROM taxi_2016_clustered")
```

```
val clusterCount = 4
val assembler = new VectorAssembler().
setInputCols(Array(
    "tpep_pickup_datetime",
    "tpep_dropoff_datetime",
    "VendorID",
    "passenger_count",
    "trip_distance",
    "pickup_longitude",
    "pickup_latitude",
    "RateCodeID",
    "dropoff_longitude",
    "dropoff_latitude",
    "extra",
    "mta_tax",
    "tolls_amount",
    "improvement_surcharge",
    "total_amount",
    "pickupHour"
  )).setOutputCol("features")
val assembled = assembler.transform(data)
var models = Array[CrossValidatorModel]()
for( i <- 0 until clusterCount){
  val classifier = new MultilayerPerceptronClassifier()
    .setBlockSize(128)
    .setLabelCol("didTip")

  val paramGrid = new ParamGridBuilder()
    .addGrid(classifier.maxIter, Array(50,100))
    .addGrid(classifier.layers, Array(Array(16, 5, 4, 3), Array(16, 20, 16, 3)))
    .build()
  val evaluator = new BinaryClassificationEvaluator()
    .setLabelCol("didTip")
    .setRawPredictionCol("prediction")
    .setMetricName("areaUnderROC")

  val crossval = new CrossValidator()
  crossval.setEstimator(classifier)
  crossval.setEvaluator(evaluator)
```

```
  crossval.setEstimatorParamMaps(paramGrid)
  crossval.setNumFolds(5)

  val modelCV = crossval.fit(assembled.filter("cluster == " + i))
  models = models :+ modelCV
  modelCV.avgMetrics.foreach(println)
  modelCV.params.foreach(println)
  modelCV.save(f"notebooks/taxai/models_v3/mp/$i")
  println("===")
}
val area = models.flatMap(m => m.avgMetrics)
val cluster = Array(1,2,3,4)
val results = sc.parallelize(area zip cluster).toDF("areaUnderROC", "cluster")
results.createOrReplaceTempView("multilayer_results_temp")
spark.sql("DROP TABLE IF EXISTS multilayer_results")
spark.sql("CREATE TABLE multilayer_results AS SELECT * from multilayer_results_temp")
import org.apache.spark.ml.classification.{RandomForestClassifier, RandomForestClassificationModel}
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics
import org.apache.spark.ml.feature.{StringIndexer, VectorIndexer, VectorAssembler}
import org.apache.spark.ml.tuning.{ParamGridBuilder, CrossValidator, CrossValidatorModel}
import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator
import org.apache.spark.sql.types.{StructField, StructType, IntegerType}
val spark = org.apache.spark.sql.SparkSession.builder
        .config("spark.kryo.registrator", "org.bdgenomics.adam.serialization.ADAMKryoRegistrator")
        .master("local")
        .getOrCreate;
val data = spark.sql("SELECT * FROM taxi_2016_clustered")
val clusterCount = 4
val assembler = new VectorAssembler().
setInputCols(Array(
    "tpep_pickup_datetime",
    "tpep_dropoff_datetime",
    "VendorID",
    "passenger_count",
    "trip_distance",
    "pickup_longitude",
    "pickup_latitude",
    "RateCodeID",
```

```
    "dropoff_longitude",
    "dropoff_latitude",
    "extra",
    "mta_tax",
    "tolls_amount",
    "improvement_surcharge",
    "total_amount",
    "pickupHour"
  )).setOutputCol("features")
val assembled = assembler.transform(data)
var models = Array[CrossValidatorModel]()
for( i <- 0 until clusterCount){
  val rf = new RandomForestClassifier()
    .setLabelCol("didTip")
    .setFeaturesCol("features")

  val paramGrid = new ParamGridBuilder()
    .addGrid(rf.maxDepth, Array(6, 15))
    .addGrid(rf.numTrees, Array(20, 40))
    .addGrid(rf.impurity, Array("entropy","gini"))
    .build()
  val evaluator = new BinaryClassificationEvaluator()
    .setLabelCol("didTip")
    .setRawPredictionCol("prediction")
    .setMetricName("areaUnderROC")

  val crossval = new CrossValidator()
  crossval.setEstimator(rf)
  crossval.setEvaluator(evaluator)
  crossval.setEstimatorParamMaps(paramGrid)
  crossval.setNumFolds(5)

  val modelCV = crossval.fit(assembled.filter("cluster == " + i))
  models = models :+ modelCV
  modelCV.avgMetrics.foreach(println)
  modelCV.save(f"notebooks/taxai/models_v3/rf/$i")
  println("===")
}
val area = models.flatMap(m => m.avgMetrics)
```

```
val cluster = Array(1,2,3,4)
val results = sc.parallelize(area zip cluster).toDF("areaUnderROC", "cluster")
results.createOrReplaceTempView("random_forest_results_temp")
spark.sql("DROP TABLE IF EXISTS random_forest_results")
spark.sql("CREATE TABLE random_forest_results AS SELECT * from random_forest_results_temp")
import org.apache.spark.ml.classification.DecisionTreeClassificationModel
import org.apache.spark.ml.classification.DecisionTreeClassifier
import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics
import org.apache.spark.mllib.util.MLUtils
import org.apache.spark.ml.feature.{StringIndexer, VectorIndexer, VectorAssembler}
import org.apache.spark.ml.tuning.{ParamGridBuilder, CrossValidator, CrossValidatorModel}
import org.apache.spark.ml.{Pipeline, PipelineModel}
import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator
val spark = org.apache.spark.sql.SparkSession.builder
      .config("spark.kryo.registrator", "org.bdgenomics.adam.serialization.ADAMKryoRegistrator")
      .master("local")
      .getOrCreate;
val data = spark.sql("SELECT * FROM taxi_2016_clustered")
val clusterCount = 4
val assembler = new VectorAssembler().
setInputCols(Array(
   "tpep_pickup_datetime_int",
   "tpep_dropoff_datetime_int",
   "VendorID",
   "passenger_count",
   "trip_distance",
   "pickup_longitude",
   "pickup_latitude",
   "RatecodeID",
   "dropoff_longitude",
   "dropoff_latitude",
   "extra",
   "mta_tax",
   "tolls_amount",
   "improvement_surcharge",
   "total_amount",
   "pickupHour"
 )).setOutputCol("features")
```

```
val assembled = assembler.transform(data)
var models = Array[CrossValidatorModel]()
for( i <- 0 until clusterCount){
  val evaluator2 = new BinaryClassificationEvaluator()
    .setLabelCol("didTip")
    .setRawPredictionCol("prediction")
    .setMetricName("areaUnderROC")

  // Train a DecisionTree model.
    val dt = new DecisionTreeClassifier()
      .setLabelCol("didTip")
      .setFeaturesCol("features")

  val paramGrid = new ParamGridBuilder()
    .addGrid(dt.maxDepth, Array(6, 15))
    .addGrid(dt.maxBins, Array(64, 96))
    .addGrid(dt.impurity, Array("entropy","gini"))
    .build()

  val crossval = new CrossValidator()
  crossval.setEstimator(dt)
  crossval.setEvaluator(evaluator2)
  crossval.setEstimatorParamMaps(paramGrid)
  crossval.setNumFolds(5)

  val modelCV = crossval.fit(assembled.filter("cluster == " + i))
  models = models :+ modelCV
  modelCV.avgMetrics.foreach(println)
  modelCV.save(f"notebooks/taxai/models_v3/dt/$i")
  println()
}
val area = models.flatMap(m => m.avgMetrics)
val results = sc.parallelize(area).toDF("areaUnderROC")
results.createOrReplaceTempView("decision_tree_results_temp")
spark.sql("DROP TABLE IF EXISTS decision_tree_results")
spark.sql("CREATE TABLE decision_tree_results AS SELECT * from decision_tree_results_temp")
import org.apache.spark.ml.tuning.CrossValidatorModel
import org.apache.spark.ml.classification.DecisionTreeClassificationModel
import org.apache.spark.ml.classification.RandomForestClassificationModel
```

```
import org.apache.spark.ml.param.ParamMap
import scala.runtime.ScalaRunTime._
for (i <- 0 to 3) {
 val model = CrossValidatorModel.load("notebooks/taxai/models_v3/dt/"+i)
 println(model.bestModel.asInstanceOf[DecisionTreeClassificationModel].featureImportances)
 println
}
for (i <- 0 to 3) {
 val model = CrossValidatorModel.load("notebooks/taxai/models_v3/rf/"+i)
 println(model.bestModel.asInstanceOf[RandomForestClassificationModel].featureImportances)
 println
}
for (i <- 0 to 3) {
 val model = CrossValidatorModel.load("notebooks/taxai/models_v3/dt/"+i)
 model.getEstimatorParamMaps.zip(model.avgMetrics).foreach(println)
 println
}
for (i <- 0 to 3) {
 val model = CrossValidatorModel.load("notebooks/taxai/models_v3/rf/"+i)
 model.getEstimatorParamMaps.zip(model.avgMetrics).foreach(println)
 println
}
for (i <- 0 to 3) {
 val model = CrossValidatorModel.load("notebooks/taxai/models_v3/mp/"+i)
 val seq = model.getEstimatorParamMaps.map(p => (stringOf(p.toSeq(0).value),
stringOf(p.toSeq(1).value))).zip(model.avgMetrics).foreach(println)
 println
}
import org.apache.spark.ml.tuning.CrossValidatorModel
import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.sql.types._
val spark = org.apache.spark.sql.SparkSession.builder
    .config("spark.kryo.registrator", "org.bdgenomics.adam.serialization.ADAMKryoRegistrator")
    .master("local")
    .getOrCreate;
val data = spark.sql("SELECT * FROM initial_2016_clustered")
val assembler = new VectorAssembler().
setInputCols(Array(
```

```
      "tpep_pickup_datetime_int",
      "tpep_dropoff_datetime_int",
      "VendorID",
      "passenger_count",
      "trip_distance",
      "pickup_longitude",
      "pickup_latitude",
      "RatecodeID",
      "dropoff_longitude",
      "dropoff_latitude",
      "extra",
      "mta_tax",
      "tolls_amount",
      "improvement_surcharge",
      "total_amount",
      "pickupHour"
    )).setOutputCol("features")

  val assembled = assembler.transform(data)
  var clusterResultsMP = Array[Double]()
  var allResultsMP = Array[Double]()
  for (i <- 0 to 3) {
    val bestModel = CrossValidatorModel.load("notebooks/taxai/models_v3/mp/"+i).bestModel
    val transform = bestModel.transform(assembled.filter("cluster == " + i))
    val evaluator = new BinaryClassificationEvaluator().setLabelCol("didTip").setRawPredictionCol("prediction").setMetricName("areaUnderROC")
    val result1 = evaluator.evaluate(transform)
    println("cluster " + i + " local results: " + result1)
    clusterResultsMP = clusterResultsMP :+ result1
    val transform2 = bestModel.transform(assembled)
    val result = evaluator.evaluate(transform2)
    println("cluster " + i + " global results: " + result)
    allResultsMP = allResultsMP :+ result
  }
  var clusterResultsDT = Array[Double]()
  var allResultsDT = Array[Double]()
  for (i <- 0 to 3) {
    val bestModel = CrossValidatorModel.load("notebooks/taxai/models_v3/dt/"+i).bestModel
```

```
    val transform = bestModel.transform(assembled.filter("cluster == " + i))
    val evaluator = new BinaryClassificationEvaluator().setLabelCol("didTip").setRawPredictionCol("prediction").setMetricName("areaUnderROC")
    val result1 = evaluator.evaluate(transform)
    println("cluster " + i + " local results: " + result1)
    clusterResultsDT = clusterResultsDT :+ result1
    val transform2 = bestModel.transform(assembled)
    val result = evaluator.evaluate(transform2)
    println("cluster " + i + " global results: " + result)
    allResultsDT = allResultsDT :+ result
  }
  var clusterResultsRF = Array[Double]()
  var allResultsRF = Array[Double]()
  for (i <- 0 to 3) {
    val bestModel = CrossValidatorModel.load("notebooks/taxai/models_v3/rf/"+i).bestModel
    val transform = bestModel.transform(assembled.filter("cluster == " + i))
    val evaluator = new BinaryClassificationEvaluator().setLabelCol("didTip").setRawPredictionCol("prediction").setMetricName("areaUnderROC")
    val result1 = evaluator.evaluate(transform)
    println("cluster " + i + " local results: " + result1)
    clusterResultsRF = clusterResultsRF :+ result1
    val transform2 = bestModel.transform(assembled)
    val result = evaluator.evaluate(transform2)
    println("cluster " + i + " global results: " + result)
    allResultsRF = allResultsRF :+ result
  }
```