Tartu Ülikool
Loodus- ja täppisteaduste valdkond
Matemaatika ja statistika instituut

Markus Haug

Juhumetsa mudelite loomine tasakaalustamata andmetega

Matemaatilise statistika eriala
Bakalaureusetöö (9 EAP)

Juhendaja: PhD Raivo Kolde

Tartu 2020

Juhumetsa mudelite loomine tasakaalustamata andmetega

Bakalaureusetöö

Markus Haug

Lühikokkuvõte

Antud bakalaureusetöö eesmärk on tutvustada erinevaid meetodeid, mida kasutatakse juhumetsa mudelite loomiseks tasakaalustamata andmetel. Töö esimeses pooles tutvustatakse klassifitseeriva juhumetsa olemust ning tasakaalustamata andmetel mudelite loomise probleemi. Töö teises osas pakutakse välja ja võrreldakse erinevaid meetodeid, mida saab kasutada tasakaalustamata andmete probleemi lahendamiseks.

CERCS teaduseriala: P160 Statistika, operatsioonianalüüs, programmeerimine, finants- ja kindlustusmatemaatika.

Märksõnad: Juhumets, tasakaalustamata andmed, klassifitseerimine, MIMIC-III.

Creating Random Forest Models With Imbalanced Data

Bachelor thesis

Markus Haug

Abstract

The aim of this thesis is to provide an overview of using random forest to learn imbalanced data. In the first part of the thesis random forest classifier and the problem of imbalanced data is introduced. In the second part of the work possible methods for dealing with imbalanced data are proposed. The methods are later compared with each other using a classification problem.

CERCS research specialisation: P160 Statistics, operations research, programming, financial and actuarial mathematics.

Key Words: Random forest, imbalanced data, classification, MIMIC-III.

Sisukord

Sissejuhatus

Andmete analüüsimisel, mudelite ja seoste loomisel kasutatakse tänapäeval mitmeid masinõppe meetodeid. Masinõppe meetodid on efektiivsed erinevat tüüpi seoste leidmisel suurte andmemahtude puhul. Üheks edukaimaks masinõppe meetodiks on juhumets (ingl *random forest*). Kuigi juhumetsa mudelid on läbipaistvad ja väheste parameetritega (Biau *et al.*, 2015), võivad nad teatud juhtudel ebaõnnestuda (Tang *et al.*, 2018). Üheks selliseks juhuks on tasakaalustamata andmetel mudelite loomine.

Antud bakalaureusetöö keskendubki selle probleemi lahendamisele. Täpsemalt uurime, milliseid meetodeid on mõistlik rakendada, kui tuleb koostada klassifitseerivaid juhumetsa ennustusmudeleid tasakaalustamata andmetel. Töö eesmärk on anda ülevaade võimalikest kasutatavatest meetoditest, tuua konkreetseid näiteid nende rakendamisest ja tulemusi võrrelda. Meetodite võrdlemiseks kasutame meditsiinilist MIMIC-III andmestikku. Selle põhjal koostame klassifitseerivaid juhumetsa mudeleid, mille eesmärk on prognoosida patsiendi suremist ühe nädala jooksul peale viimast arstivisiiti.

Bakalaureusetöö koosneb kuuest peatükist. Esimeses peatükis anname ülevaate juhumetsa sh otsustuspuude tööpõhimõtetest. Teises peatükis avame tasakaalustamata andmete probleemi tausta ja kirjeldame võimalike lahendusi. Kolmandas peatükis tutvustame töö praktilises osas kasutatavat andmestikku. Neljandas peatükis avaldame lugejale, kuidas lõime mudeleid. Viiendas peatükis kirjeldame töö käigus saadud tulemusi. Viimases, kuuendas, peatükis võtame käesoleva bakalaureusetöö kokku.

Bakalaureusetöö läbiviimiseks on kasutatud statistikatarkvara R (versioon 3.5.1). Töö on kirjutatud kasutades LaTeX veebirakenduse liidest Overleaf. Töö läbiviimiseks vajalik arvutusresurss on saadud Tartu Ülikooli HPC klastrist Rocket.

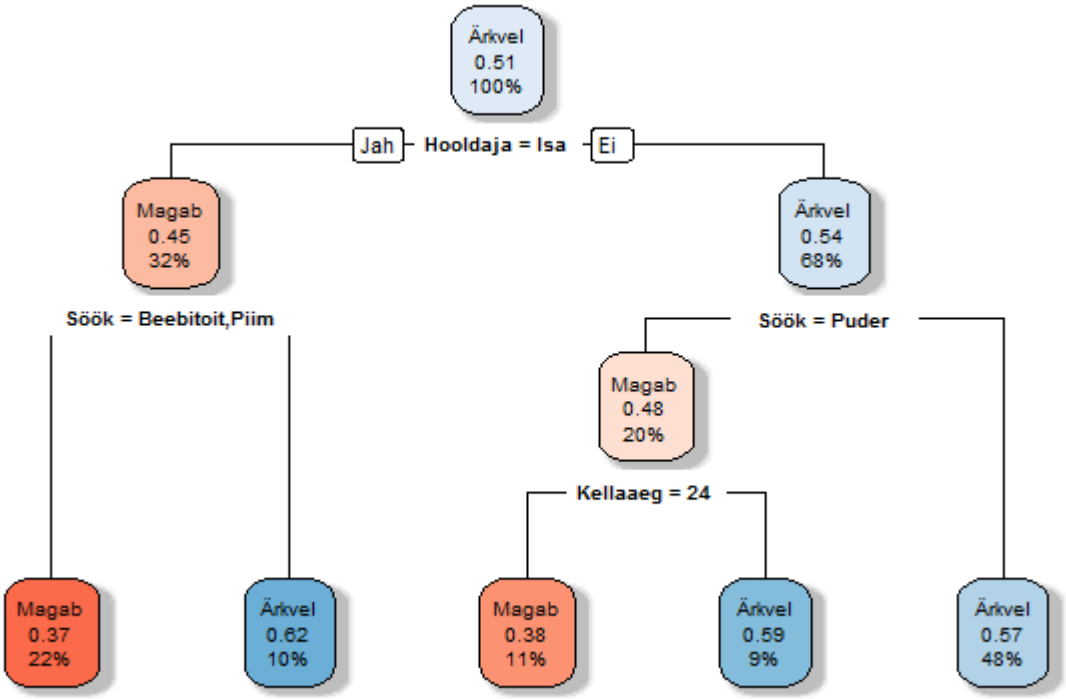
1. Otsustuspuud ja juhumets

Üks enim kasutatavatest meetoditest masinõppe ülesannete lahendamisel on juhumetsa meetod. Juhumetsa meetod arendati välja 1990. aastate keskel, mille üheks esimeseks arendajaks oli USA statistik Leo Breiman (Breiman, 2001). Selles peatükis anname ülevaate juhumetsa olemusest ja selle ennustustäpsust kirjeldavatest suurustest.

1.1. Otsustuspuud

Otsustuspuud on juhumetsade alustalaks - nad moodustavad juhumetsa. Selles alapeatükis tutvustame otsustuspuude olemust. Me tutvustame ainult klassifitseerivat otsustuspuu meetodit, kuna antud töös kasutame vaid klassifitseerimist (veel eristatakse regressiooni meetodit).

Klassifitseeriva otsustuspuu meetodi eesmärk on leida vaatluse ühe tunnuse klassile väärtus tuginedes tema teiste tunnuste väärtustele. Klassifitseerivat otsustuspuud kasutades on meil enamasti valim n vaatlusega ja sõltuv tunnus Y, mis võtab väärtuseid klassidest $1, 2, 3, \ldots, k$. Samuti on meil p kirjeldavat tunnust $X_1, \ldots, X_p$. Meie eesmärk on teadaolevate tunnuste X_i ($i = 1, \ldots, p$) väärtuste põhjal klassifitseerida Y väärtus. Seda tehakse kahendpuul, kus igal tipul jaotatakse objektid mingi tingimuse järgi kaheks, kuni nad lehtedel lõpliku klassifikatsioonini jõuavad (Loh, 2011). Joonisel 1 on kujutatud näiteandmestikul koostatud klassifitseeriv otsustuspuu, mis prognoosib noorte laste ärkvel olemist. Igal vaatlusel on kolm nominaalset kirjeldavat tunnust. Moodustunud otsustuspuu koosneb neljast tipust ja viiest lehest.



Joonis 1: Otsustuspuu näide. Igal kahendpuu tipul on kujutatud vastavalt prognoositavat klassi, noore lapse ärkveloleku tõenäosust ja treeningandmete protsentuaalset tipu läbivust.

Otsustuspuu tippudes peavad olema efektiivsed vaatluste jagamise tingimused. Traditsiooniliselt kasutatakse otsustuspuul tipu tingimuse hindamiseks Gini indeksit, mida arvutatakse järgmiselt:

$$I_G(t) = 1 - \sum_{i=1}^{k} (p_i)^2.$$

Eelnevas tähistab p_i $(i = 1, \ldots, k)$ tõenäosust, et antud tippu t jõudnud juhusliku vaatluse puhul $Y = i$. Põhimõtteliselt kirjeldab Gini indeks tõenäosuse suurust, et tippu jõudnud objekt klassifitseeritakse valesti, kui ta klassifitseeritaks samasse tippu jõudnud objektide jaotuse järgi. Seega on lahknemise tingimus, millele vastab väikesem Gini indeks, eelistatud kõrgemate üle (Loh, 2011).

1.2. Juhumets

Mõistmaks juhumetsa eelist otsustuspuu ees, peame tutvustama *bootstrap* agregeerimise (ingl *bagging*) mõistet. Oletame, et me lahendame K klassi klassifitseerimisprobleemi. Me loome otsustuspuu mudeli treeningandmestikul $Z = \{(y_1, x_1), \ldots, (y_N, x_N)\}$ ja saame selle mudeli järgi prognoosi $\hat{G}(x)$ vaatluse x klassile. Defineerime indikaatorvektori funktsiooni $\hat{f}(x)$, mille väärtus koosneb $K-1$ nullist ja ainsast ühest. Sel juhul $\hat{G}(x) = argmax_k \hat{f}(x)$ ehk $\hat{G}(x)$ on $\hat{f}(x)$ suurima väärtusega elemendi indeks. *Bootstrap* agregeerimine keskmistab selle prognoosi üle *bootstrap* valimite, vähendades sealhulgas dispersiooni. Iga *bootstrap* valimi Z^{*b}, $b = 1, \ldots, B$ jaoks tekitame otsustuspuu mudeli, mis annab meile prognoosi $\hat{f}^{*b}(x)$. Seega saame *bootstrap* agregeerimise hinnangu

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x) = [p_1(x), p_2(x), \ldots, p_K(x)],$$

kus $p_k(x)$ on võrdne otsustuspuude osakaaluga, mis prognoosisid vaatlust x klassi k. *Bootstrap* agregeerimise klassifitseerija $\hat{G}_{bag}(x) = argmax_k \hat{f}_{bag}(x)$ prognoosib vaatluse x klassi k enamushääletuse põhjal. Põhiline *bootstrap* agregeerimise idee on keskmistada ligikaudu erapooletuid, kuid müratundlikuid mudeleid, ning selle läbi vähendada dispersiooni (Hastie *et al.*, 2001).

Otsustuspuud on ideaalsed *bootstrap* agregeerimise kandidaadid. Esiteks suudavad nad leida komplekseid struktuure, jäädes üsna erapooletuks. Teiseks on nad väga müratundlikud. Nende ainuke võimalus paremaid tulemusi saada on dispersiooni vähendamine. B sõltumatu sama jaotuse juhusliku suuruse, igaüks dispersiooniga σ^2, keskmise dispersioon on $\frac{1}{B}\sigma^2$. Kui need suurused on sama jaotusega ja omavad positiivset paariviisilist korrelatsiooni ρ, siis on keskmise dispersioon

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

Kui B väärtus tõuseb, siis jääb keskmise dispersiooni kirjeldama vaid $\rho\sigma^2$. Juhumetsade idee seisneb *bootstrap* agregeerimise dispersiooni vähendamises. Juhumetsade eesmärgiks on vähendada korrelatsiooni otsustuspuude vahel, dispersiooni märgatavalt tõstmata. Selline olukord saavutatakse, kui treeningandmestikul luuakse puid juhusliku hulga tunnuste ja *bootstrap* valimeid kasutades (Hastie *et al.*, 2001).

Klassikalise reegli kohaselt valitakse ühe otsustuspuu tunnuste arvuks ruutjuur kõikide tunnuste arvust. Selline käitumine aitab vältida ülesobitamist (ingl *overfitting*). Ülesobitamine tähendab, et moodustatav mudel töötab hästi andmestiku peal, mida kasutati selle mudeli arendamiseks, aga halvasti üldistatud uute näidete peal (Breiman, 2001).

Kokkuvõttes võib juhumetsa kirjeldada kui otsustuspuude kogumit, millel viiakse läbi enamushääletus. Iga otsustuspuu loomisel kasutatakse vaid alamhulka kõikidest vaatlustest ja tunnustest (seega need erinevad). Klassifitseeritavaid vaatluseid klassifitseerivad kõik juhumetsa otsustuspuud ning juhumets prognoosib vaatluse klassi, mis saab proportsionaalselt enim hääli.

1.3. Mudeli ennustustäpsuse mõõtmine

Üheks võimaluseks juhumetsa mudeli ennustustäpsust hinnata on tema prognooside kaardistamine. See eeldab andmestiku jaotamist enne mudeli koostamist kaheks: treeningandmestikuks ja testandmestikuks. Kui mudel on treeningandmete peal loodud, lastakse sellel klassifitseerida testandmeid, ning saadud tulemused kaardistatakse (vt tabel 1) neljaks: tõesed positiivsed (TP), väärad positiivsed (VP), tõesed negatiivsed (TN), väärad negatiivsed (VN).

Tabel 1: Prognooside kaardistamine kahe klassi korral.

	Prognoositakse uuritavasse klassi	Ei prognoosita uuritavasse klassi
Reaalselt on uuritavas klassis	Tõesed positiivsed	Väärad negatiivsed
Reaalselt ei ole uuritavas klassis	Väärad positiivsed	Tõesed negatiivsed

Mudeli eesmärgiks on saavutada võimalikult suur täpsus (Provost, 2000). Täpsus iseloomustab mudeli võimekust eraldada andmeid testandmestikul:

$$Täpsus = \frac{TP+TN}{TP+TN+VP+VN}.$$

Tasub märkida, et selline mudeli ennustustäpsuse hindaja ei ole väga palju mõjutatud proportsionaalselt

väikeste klasside valesti klassifitseerimisest.

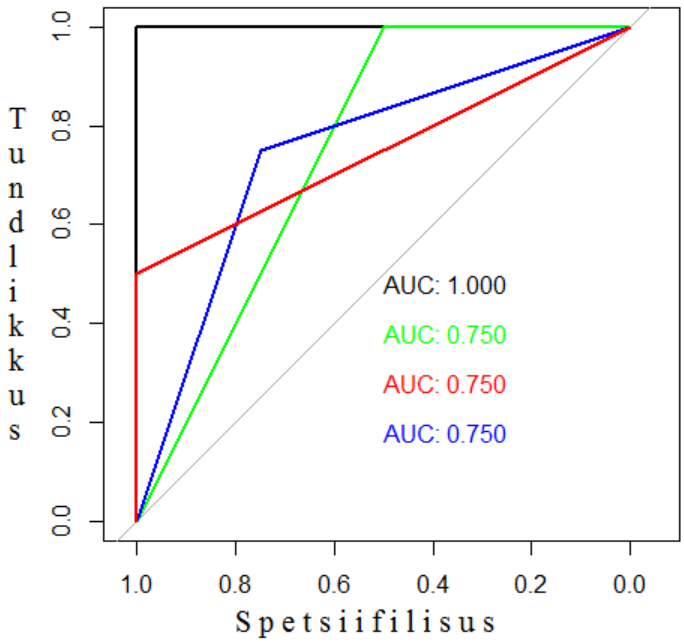
Mudeli täpsuse puhul räägitakse tihti ka mudeli tundlikkusest (ingl *sensitivity*) ja spetsiifilisusest (ingl *specificity*). Tundlikkus iseloomustab mudeli täpsust prognoosida tegelikke positiivseid vaatlusi:

$$Tundlikkus = \frac{TP}{TP+VN}.$$

Spetsiifilisus iseloomustab mudeli täpsust prognoosida tegelikke negatiivseid vaatlusi:

$$Spetsiifilisus = \frac{TN}{TN+VP}.$$

Mudeli ennustustäpsust kujutatakse tihti joonisel ROC-kõveraga (ingl *receiver operating characteristics curve*). See kasutab mudeli hindamiseks tema tundlikkuse ja väär-positiivsete ($1 - spetsiifilisus$) määra graafikul. Graafikule tekkinud ROC alust pindala AUC (ingl *area under curve*) peetakse klassifitseerimismudelite üheks tähtsaimaks hinnanguks (Flach, 2016). Kui $AUC < 0.5$ on mudel halvem juhuslikult klassifitseerivast mudelist. Kui $AUC = 1$, siis töötab mudel valideerimisandmetel ideaalselt. Joonisel 2 on toodud näited neljast erinevast ROC-kõverast ja nende AUC väärtustest. Kui tegeletakse tasakaalustamata andmestikega, kus vale klassifitseerimine põhjustab erinevates klassides erinevaid tagajärgi, siis on mõistlikum kasutada ROC-kõveraga seotud mudeli hinnanguid (Bowyer *et al.*, 2011).



Joonis 2: ROC-kõverad. Igale ROC-kõverale vastavat AUC väärtus on kirjeldatud joonisel sama värviga.

Lisaks eelnevale kasutatakse juhumetsade hindamiseks ka *out-of-bag* (OOB) viga. See ei eelda andmete jagamist treening- ja testandmeteks. Kuna juhumetsa treenimisel iga otsustuspuu loomiseks rakendatakse vaid osa kasutamiseks mõeldud andmetest, siis saab sellest valimist välja jäänud vaatlusi kasutada antud otsustuspuu ennustustäpsuse hindamiseks. Olgu meil vaatlused (y_i, x_i) ja andmete põhjal valminud otsustuspuud T_k, kus i ja k on indeksid. Me hindame iga vaatluse (y_i, x_i) klassi y_i puudel T_k, mille koostamisel vastav vaatlus valimisse ei kuulunud. Sarnaselt juhumetsa klassifitseerimiseeskirjale (enamushääletus) prognoositakse (y_i, x_i) klass, ning kõikide vaatluste hindamisel kujunenud veahinnang ongi *out-of-bag* viga (Breiman, 2001).

2. Tasakaalustamata andmed

Andmestikud koosnevad vaatlustest ja vaatlustel on tunnused, mille järgi saab nad erinevatesse klassidesse jagada. Kui uuritava tunnuse klassides olevad vaatlused ei jagune ühtlaselt, siis tekivad enamus- ja vähemusklassid ehk uuritav andmestik ei ole tasakaalus. Tasakaalustamata andmestikud esinevad paljudes elulistes rakendustes. Eriti problemaatiline on see meditsiinis (Ali *et al.*, 2019).

Enamus klassifitseerimise algoritmidest tegutsevad kahe eelduse järgi.

- Mudeli eesmärgiks on maksimeerida täpsust.
- Treeningandmestik on sama jaotusega nagu testandmestik.

Tasakaalustamata andmete kasutamise puhul annavad sellised mudelid mitterahuldavaid tulemusi. Tihti on vähemusklass proportsionaalselt väga väike, moodustades koguni alla 1% kogu andmestikust. Kui sellisele andmestikule rakendada traditsioonilisi klassifitseerimismudeleid, siis enamasti ennustab mudel kõik vaatlused enamusklassi. See on probleem, kui uurija peamiseks eesmärgiks on vähemusklassi elementide õigesti klassifitseerimine (Provost, 2000).

Selles peatükis kirjeldame erinevaid andmestike ja mudelite kohandamise meetodeid, mida kasutatakse klassifitseerimismudelite loomise eel nende tegevustulemuse võimendamiseks. Peamiselt on allpool kirjeldatavad meetodid lahti mõtestatud parandamaks klassifitseeriva juhumetsa mudeli näitajaid. Neid, üsna universaalseid meetodeid, saab rakendada ka teist tüüpi mudelite loomisel.

2.1. Taasvalimismeetodid

Üks enim kasutatavaid strateegiaid tasakaalustamata andmete puhul on andmete taasvalimine (ingl *resampling*) treeningandmete hulgas. Taasvalimine tähendab olemasolevate andmete põhjal uue andmestiku moodustamist. Seda tehakse, et balansseerida mõne klassi osakaalu treeningandmestikus. Enamjaolt kasuta-

takse selleks vähemusklassi ülevalimist või enamusklassi alavalimist. Mõlemal taasvalimise tüübil on selgeid kitsaskohtasid, kuna nad muudavad algandmestiku jaotust (Marqués *et al.*, 2013). Järgnevalt anname ülevaate tuntumatest taasvalimise meetoditest.

2.1.1. Ülevalimine

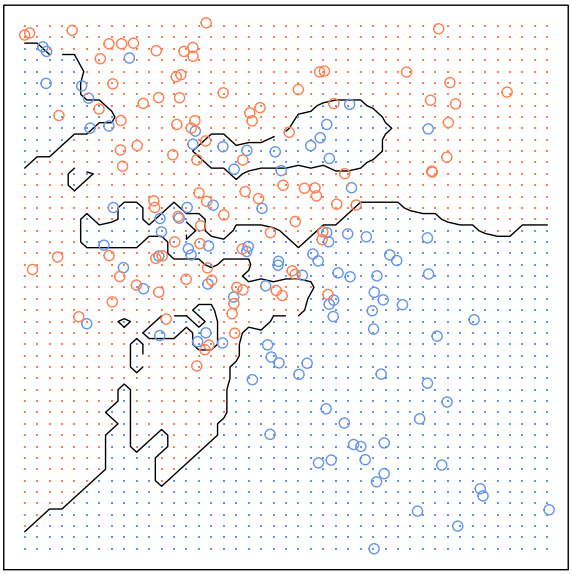
Traditsiooniliselt kasutatakse andmete tasakaalustamiseks ülevalimist (ingl *oversampling*). Selle lähenemise käigus valitakse vähemusklassidele juhuslikult tagasipanekuga vaatluseid juurde, kuni andmestiku igal klassil on ligikaudu sama palju esindajaid. Ülevalimise kasutamisel on probleemideks ülesobitavuse ja suurema arvutusressursi vajaduse teke (Kuhn *et al.*, 2013).

Sünteetiline vähemusklassi ülevalimise tehnika (ingl *synthetic minority over-sampling technique*) ehk SMOTE on ülevalimise tehnika, mis kasutab olemasolevatest andmetest uue vaatluse sünteesimist, selleks, et tekitada uusi andmepunkte.

SMOTE meetodi mõistmiseks peame tutvustama nii vaatluste kauguse kui ka k-lähima naabri (ingl *k-Nearest-Neighbor*) klassifitseerija mõistet. Olgu meil vaatlused x_0 ja y_0, siis vaatluste vaheline kaugus $\delta(x_0, y_0)$ on nendevaheline vahemaa tunnuste ruumis. Tavaliselt, kui tunnused on reaalarvulised, kasutatakse eukleidilist kaugust

$$\delta(x_0, y_0) = \|x_0 - y_0\|.$$

K-lähima naabri (edaspidi k-NN) klassifitseerija on mälupõhine ning ei vaja mudeli sobitamist. Olgu meil uuritav vaatlus x_0, mida soovime klassifitseerida. Me leiame treeningandmestikul x_0 suhtes k kauguse mõttes lähimat naabrit (vaatlust) $x_{(r)}$, $r = 1, \ldots, k$. Vaatlus x_0 klassifitseeritakse k-NN klasside enamushääletuse järgi (Hastie *et al.*, 2001). Joonisel 3 on kujutatud kahe klassi klassifitseerimisülesande lahendamiseks 5-NN algoritmi rakendamist näidisandmestikul. Klassifitseerija prognoosid on eraldatud nii kontuurjoontega kui ka erineva värviga tähistatud teljestiku punktikestega (vt joonis 3).



Joonis 3: Näidisandmestik 5-NN algoritmi kasutamisel kahe klassi puhul. Ringid märgistavad vaatlusi ja punktid märgistavad klassifitseerija järgi määratud klassi ala. Erinevad klassid on tähistatud oranži ja sinise värviga.

SMOTE meetod sünteesib uusi andmepunkte järgmise algoritmi järgi:

1. Vähemusklassist valitakse juhuslik vaatlus.
2. Treeningandmestikult leitakse sellele vaatlusele k-lähimat naabrit.
3. Uuele sünteesitud objektile antakse vähemusklassi omadus (tunnus).
4. Sünteesitud objektile valitakse juhuslikult tunnuste väärtused esimeses ja teises punktis valitud vaatluste vastavate tunnuste väärtuste valimi hulgast.

Sünteesitud andmepunktid lisatakse vähemusklassi, et balansseerida vähemusklasse ja enamusklasse (Kuhn *et al.*, 2013). Selle tehnika peamine eesmärk on vähendada traditsioonilisel ülevalimisel tekkivat ülesobitavust.

2.1.2. Alavalimine

Alavalimine (ingl *downsampling*) on vastand ülevalimisele. Kui meil on ühes klassis vaatlusi rohkem kui teises, siis seisneb alavalimise idee enamusklassi vaatluste vähendamises, kuni vaatlusi on klassides enamvähem võrdselt. Seda tehakse juhusliku valimi võtmisega enamusklassist. Alavalimise selgeimaks probleemiks on infokadu (Kuhn *et al.*, 2013).

Selleks, et infokao probleemi vähendada, on välja käidud lisastrateegiaid. Näiteks OSS (ingl *One-Sided Sampling*) meetod, mille käigus eemaldatakse ainult neid enamusklassi vaatlusi, mis piirnevad alamklassi vaatlustega (tekitavad müra) ja on ülearused. Müra tekitavad vaatlused põhjustavad ülesobitamist treeningandmestikul (Sammut *et al.*, 2011). Ülearused vaatlused ei ohusta korrektset klassifitseerimist, aga suurendavad mudeli loomise ressursi vajadust (Kubat *et al.*, 1997).

OSS meetodi rakendamisel leitakse piirnevad vaatlused näiteks Tomeki sidususe meetodiga. Ülearused enamusklassi vaatlused leitakse Harti tehnikaga (tuntakse ka kui CNN algoritmi ehk ingl *condensed nearest neighbor algorithm*). Joonisel 4 on näide andmestikust, kus on nii enamusklassi müra (tähistatud kolmnurkadega) kui ka ülearuseid vaatluseid.

Müra eemaldamine käib järgmise algoritmi (Tomeki sidususe meetod) järgi.

1. Vaatleme kahte objekti x ja y. Olgu x vähemusklassist ja y enamusklassist ning $\delta(x,y)$ nendevaheline kaugus.

2. Kui paari (x,y) puhul ei leidu ühtegi objekti z, et kehtiks

   $$\delta(x,z) < \delta(x,y)$$

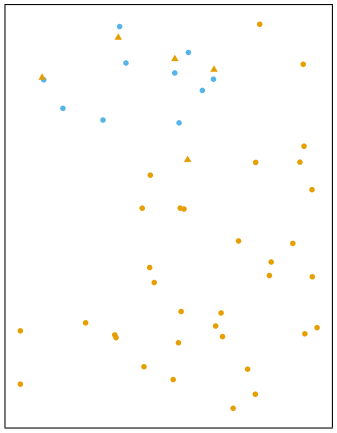
   või

   $$\delta(y,z) < \delta(x,y)$$

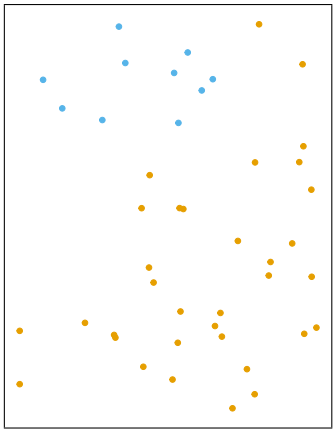
   siis võime pidada enda jaoks vaatlust y müraks.

3. Eemaldame vaatluse y.

Joonisel 5 on kujutletud näiteandmestiku peale müra eemaldamist Tomeki sidususe meetodiga.



Joonis 4: Näidisandmestik enne OSS meetodi rakendamist. Vähemus- ja enamusklassi vaatlused on kujutatud vastavalt sinise ja oranži värviga.

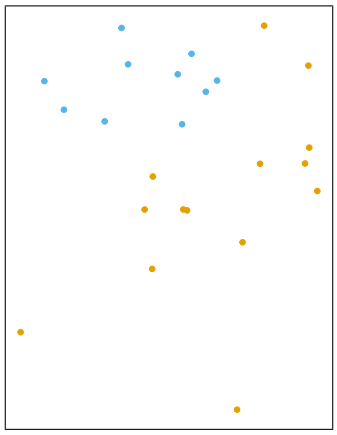


Joonis 5: Andmestik peale Tomeki sidusus meetodi rakendamist.

Ülearuste enamusklassi vaatluste eemaldamine käib modifitseeritud Harti algoritmi järgi.

1. Olgu meil treeningandmestik T ja selle osahulk P.
2. P koosneb kõikidest vähemusklassi vaatlustest ja ühest juhuslikust enamusklassi vaatlusest.
3. Ükshaaval klassifitseeritakse $T \setminus P$ elemente, kasutades $1-NN$ (1-lähima naabri) meetodit hulgal P.
4. Enamusklassi elemendid, mis klassifitseeritakse $1-NN$ klassifitseerijaga valesti, lisatakse P'sse.

Joonisel 6 on visualiseeritud näiteandmestiku (jooniselt 5) pärast modifitseeritud Harti algoritmi rakendamist. Kubat ja Matwin (1997) pakkusid sellise alavalimise meetodi välja ja tähendasid selle võimalikku efektiivsust (Kubat *et al.*, 1997).



Joonis 6: Andmestik peale Harti algoritmi rakendamist.

2.2. Kaalutundlik juhumets

Kaalutundlik (ingl *weighted, cost-sensitive*) juhumets on meetod, kus klassifitseerijat modelleeritakse tasakaalustamata andmete puhul alamklassi eelistama. Kuna tasakaalustamata andmete puhul on juhumets kallutatud klassifitseerima vaatlusi enamusklassi, siis on mõistlik seadistada vähemusklassi õigesti klassifitseerimine prioriteediks. Selleks kasutame erinevatele klassidele erineva kaalu andmist, andes tavapäraselt vähemusklassile suurema kaalu. Klassi kaalu kasutatakse juhumetsa algoritmi kahes kohas. Esiteks kasutatakse kaale otsustuspuude loomisel, et mõjutada andmete poolitamist Gini indeksiga (Chen *et al.*, 2004). Otsustuspuude tippudes arvutatakse kaalutud Gini indeksit järgmiselt (Breiman *et al.*, 1984):

$$IW_G(t) = 1 - \sum_{i=1}^{k} (\frac{w_i n_i}{s_t})^2,$$

kus

- n_i $i = 1, \ldots, k$ on vastava klassi vaatluste arv tipus t;
- w_i $i = 1, \ldots, k$ on vasatava klassi kaal;
- $s_t = \sum_i^k w_i n_i$.

Nagu ka tavalise Gini indeksi puhul, eelistatakse ka siin andmete poolitamisel võimalikult väikest kaalutud Gini indeksi väärtust. Teiseks kasutatakse klasside kaale modelleeritud juhumetsa klassifitseerimise lõppotsusel. Otsustuspuu terminalis saab puu enda prognoositud klassifitseeringule vastava kaalu kordse hääle, andes sellevõrra suurema või väiksema panuse juhumetsa lõpphääletusel (Chen *et al.*, 2004).

2.3. Kombineeritud meetodid

Bowyeri (2011) läbi viidud uuring leidis, et erinevate eelmainitud meetodite kombineerimine võib tõsta klassifitseerimismudeli prognoosimise võimekust. Bowyer leidis, et traditsioonilise alavalimise asemel võib paremini töötada SMOTE ja alavalimise kombinatsioon (edaspidi Bowyeri meetod). Meetodi läbiviimiseks muudetakse treeningandmestiku. Vähendatakse enamusklassi vaatluseid alavalimise meetodiga ja suurendatakse vähemusklassi vaatluste arvu SMOTE meetodiga (Bowyer *et al.*, 2011).

Lisaks pakume ka omalt poolt välja järgmised kolm hübriidset meetodit. Üheks võimaluseks oleks kombineerida traditsiooniline ülevalimine ja eespool mainitud (vt ptk 4.2.5.) modifitseeritud Harti algoritm (CNN). Alguses tuleks treeningandmestikus vähendada CNN algoritmiga enamusklassi vaatluste arvu ning peale seda traditsioonilise ülevalimisega vähemusklassi vaatluste arvu tasakaalustada allesjäänud enamusklassi vaatluste arvuga. Selline kombineerimine võiks suurendada OSS alavalimise mudeli tundlikkust.

Eelnevaga sarnast efekti võiks omada meetod, kus kombineeritakse modifitseeritud Harti algoritm ja kaalutundlik juhumetsa mudel. Alguses tuleb treeningandmestikus vähendada CNN algoritmiga enamusklassi vaatluste arvu ning peale seda tõsta vähemusklassi kaalu juhumetsa treenimisel. Selle meetodi eelis eelneva meetodi ees on väiksem võimalus ülesobitamiseks.

Viimasena pakume välja meetodi, kus kasutatakse koos traditsioonilist ala- ja ülevalimist. Sellise meetodi eesmärgiks on vähendada infokadu enamusklassis (võrreldes traditsioonilise alavalimisega) ja mudeli ülesobitamist vähemusklassi suhtes (võrreldes traditsioonilise ülevalimisega). Meetodi rakendamiseks tuleb treeningandmestikus suurendada vähemusklassi vaatluste arvu traditsioonilise ülevalimismeetodiga ja

vähendada enamusklassi vaatluste arvu traditsioonilise alavalimismeetodiga.

2.4. Prognoosiläve muutmine

Prognoosiläve muutmise meetod põhineb liberaalsemal vähemusklassi eelistamisel. See eeldab, et mudeli eesmärk on klassifitseerida elemente kahe klassi vahel. Sellist klassifitseerimist saab teostada, kui mudel arvutab klassifitseeritava objekti jaoks tõenäosuse, mis iseloomustab tema kuulumist mõnda klassi (seda teevad ka juhumetsa mudelid). Tavaliselt on klassifitseerimise otsustusläveks 50%. Kuid seda läve võime muuta lähtuvalt meie eesmärkidest. Selle tulemusena võib mudel meie tahtmise järgi kaotada kas tundlikkuses või spetsiifilisuses ja võita vastavalt spetsiifilisuses või tundlikkuses. Üldist mudeli täpsust selle meetodiga tõsta ei saa. Kuna ROC-kõver määratakse läbi tundlikkuse ja spetsiifilisuse, siis uueks klassifitseerimisläveks saab valida tundlikkuse väärtuse ROC-graafiku punktilt, mis on kõige lähemal optimaalsele väärtusele (vasakul asuv ülemine nurk). Samuti saame ROC-kõveralt valida ka sellise klassifitseerimiskvoodi, mis tagab meile kas soovitud tundlikkuse või spetsiifilisuse. Klassifitseerimislävendi tuletamiseks peaks mudeli koostaja kasutama eraldiseisvat andmestikku. Mudeli testimiseks mõeldud andmestiku kasutamisel saame kallutatud hinnangu mudeli ennustustäpsusele (Kuhn *et al.*, 2013). Kuna eelnev meetod ei tõsta mudeli täpsust ja on mõeldud tundlikkuse või spetsiifilisuse tõstmiseks, siis hiljem töö praktilises osas me seda meetodit ei kasuta. Pidasime oluliseks seda meetodit tutvustada, sest see leiab laialdast kasutust.

3. Andmestik

Bakalaureusetöö raames võrdleme erinevaid võimalusi tasakaalustamata andmetega juhumetsade mudelite loomisel. Selleks on meil vaja tasakaalustamata andmetega andmestikku. Antud töö läbiviimisel kasutame MIMIC-III andmebaasi. Nende andmete põhjal loome erinevate meetoditega juhumetsa ennustusmudeleid, mis prognoosivad, kas inimene sureb ära nädala jooksul peale oma viimast ülestähendatud visiiti.

3.1. MIMIC-III

MIMIC andmebaas ehk meditsiinilise informatsiooni kogum intensiivravist (ingl *Medical Information Mart for Intensive Care*) on andmebaas, mis põhineb andmetel Beth Israel'i Diakoonia meditsiinikeskusest (Boston, Massachusetts, USA). Andmestikus on anonüümsed andmed inimeste kohta, kes olid antud meditsiinikeskusega seotud aastatel 2001-2012. Andmed on mõeldud retrospektiivse andmeanalüüsi teostamiseks (Johnson *et al.*, 2016). Vaatlused on tehtud anonüümseks iga inimese sünni juhusliku arvu päevade edasilükkamisel (vastavalt sellele nihkuvad ka inimesega seotud muud kuupäevad).

Kõikidest pakutavatest andmetest kasutame antud analüüsi tegemiseks andmetabeleid surmajuhtumitest, diagnoosidest, inimeste üldinfost ja läbivaatustest. Info inimeste diagnoosidest ja läbivaatustest on talletatud rahvusvahelist haiguste klassifikatsiooni (RHK-9) kasutades. Kasutatavas andmestikus on andmed 46520 inimese kohta, kellest 14849 puhul on registreeritud surm.

3.2. Lõplik andmetabel

Lõpliku andmetabeli panime kokku eelnevalt mainitud MIMIC-III andmebaasist valitud andmetabelitest. Lõpliku andmetabeli loomise R'i kood on toodud välja lisades (vt lisa 1).

Meie eesmärgiks on antud andmete baasil koostada mudel prognoosimaks, kas inimene sureb ära nädal aega peale oma viimast visiiti. Selliseid inimesi on andmestikus 6306. Seega on meie andmestiku enamusklassis $86,44\%$ ja vähemusklassis $13,56\%$ vaatlustest. Sellist kõrget suremust põhjendab asjaolu, et MIMIC-III andmed on pärit intensiivravi meditsiinikeskusest. Tegemist on tasakaalustamata andmetega.
Prognoosimiseks kasutame tunnuseid:

- inimestele tehtud erinevatest protseduuridest (binaarne, 88 tunnust);
- inimestele antud erinevatest diagnoosidest (binaarne, 6023 tunnust);
- inimeste soost (binaarne, 2 tunnust);
- inimeste päritolu (binaarne, 23 tunnust);
- inimeste viimasest teadaolevast vanusest (aastates) enne surma (pidev, 1 tunnus);
- inimeste olek nädal aega pärast viimast visiiti: elus või surnud (binaarne, 1 tunnus).

Seega on lõplikus tabelis 46520 rida ja 6136 tunnust, millest kõiki peale viimase teadaoleva vanuse (numbriline) käsitletatkse faktorina.
Tunnuseid on küll palju ning nendest ainult tähtsamate kasutamine võiks parandada nii mudeli ennustustäpsust kui ka ökonoomsust. Andmetabelitega, kus on ainult olulised tunnused, on tegelemine kergem ja annab tavaliselt paremaid tulemusi (Rudnicki, 2011). Kuna tunnuste valimise meetodite kirjeldamine ja võrdlemine ei ole selle töö eesmärgiks, siis vastavaid meetodeid me antud töös ei rakenda.

4. Mudelite loomine

Selles peatükis esitleme viise, kuidas me rakendame peatükkides 2.1., 2.2. ja 2.3. kirjeldatud erinevaid meetodeid. Esiteks tutvustame mudelite paremaks hindamiseks loodud ristvalideerimise kontseptsiooni. Teiseks anname ülevaate statistikatarkvara R kasutamisest mudelite loomisel.

4.1. Ristvalideerimine

Mudelite hindamiseks kasutame ristvalideerimist (ingl *cross-validation*). Ristvalideerimine tähendab, et andmestik jagatakse treening- ja testandmeteks. Treeningandmetel koostatakse mudel ja hiljem hinnatakse mudeli ennustustäpsust testandmete peal. Selline lähenemine annab tavaliselt parema hinnangu mudeli ennustustäpsusele kui hinnang treeningandmetelt (James *et al.*, 2014).

Eraldasime oma andmestiku treening- ja testandmeteks vastavalt 70% (32564 vaatlust) ja 30% (13956 vaatlust) kogu andmestikust, jättes meid huvitavate juhtude proportsioonid samaks. Seega sattus treeningandmestikku 28149 vaatlust inimeste kohta, kes ei surnud seitsme päeva jooksul (enamusklass), ja 4414 vaatlust inimeste kohta, kes surid seitsme päeva jooksul (vähemusklass). Kogu töö vältel kasutame ühte ja sama eraldust st mudelid koostatakse alati sama treeningandmestiku põhjal ning hinnatakse sama valideerimisandmestiku järgi. Andmete eraldamiseks kasutatud kood on välja toodud lisades (vt lisa 2).

4.2. Juhumetsa mudelite loomine

Siin alapeatükis kirjeldame ükshaaval erinevate juhumetsa mudelite loomist statistikatarkvaras R. Juhumetsa mudelite koostamisel toetume R'i paketile *ranger*. See on Marvin N. Wright'i loodud pakett, mis on mõeldud kiireks juhumetsade rakendamiseks ehk töötamiseks andmetabelitega, millel on suured dimensioonid (Wright *et al.*, 2017). See pakett sobib meie töö läbiviimiseks, kuna see võimaldab kasutada li-

saks tavalisele juhumetsa meetodile ka kaalutud juhumetsa meetodit ja kergemaid taasvalimise meetodeid. Ülejäänud meetodite kasutamiseks loome ise võimalused, kuid toetume ka nende käigus paketile *ranger*.

4.2.1. Traditsiooniline juhumets

Esmalt loome mudeli lihtsa juhumetsa meetodiga. Mudeli treenimiseks anname algoritmile ettevalmistatud treeningandmed. Otsustuspuude moodustamisel lubame algoritmil kasutada klassikalise reegli järgi ruutjuurt kõikide tunnuste arvust ehk 78 tunnust. Võrdlusmomendi saamiseks loome juhumetsi puude arvuga 128, 512, 1024 ja 4096. Juhumetsa mudelite koostamiseks kasutame R'i paketi *ranger* meetodit *ranger* (vt lisa 3).

4.2.2. Traditsiooniline ülevalimine

Traditsioonilise ülevalimise meetodi läbiviimiseks võrdsustame vähemusklassi ja enamusklassi vaatluste arvud treeningandmestikus. Valime juhuslikult tagasipanekuga vähemusklassist vaatlusi treeningandmestiku juurde, kuni neid on sama palju kui enamusklassis. Võrdlusmomendi saamiseks loome juhumetsi puude arvuga 128, 512, 1024 ja 4096. Juhumetsa mudelite koostamiseks kasutame R'i paketi *ranger* meetodit *ranger* (vt lisa 4).

4.2.3. SMOTE ülevalimine

Sünteetilise vähemusklassi ülevalimise tehnika rakendamiseks kasutame R'i paketi *performanceEstimation* meetodit *smote*. See pakett on loodud prognoosimisülesannete lahendamise hõlbustamiseks (Torgo, 2014). Meile sobib see pakett, kuna selles on SMOTE algoritmi teostav funktsioon. Laseme võrdluse saamiseks *smote* meetodil sünteesida vähemusklassi vaatluseid nii viit kui ka kahte lähimat naabrit kasutades (tekivad kaks erinevat mudeli tüüpi). Otsustuspuude moodustamisel lubame juhumetsa

algoritmil kasutada 78 tunnust. Võrdlusmomendi saamiseks loome juhumetsi puude arvuga 128, 512, 1024 ja 4096. Taaskord kasutame selleks paketti *ranger* (vt lisa 5).

4.2.4. Traditsiooniline alavalimine

Traditsioonilise alavalimise meetodi läbiviimiseks võrdsustame vähemusklassi ja enamusklassi vaatluste arvud. Valime juhuslikult tagasipanekuta enamusklassist vaatlusi, kuni neid on sama palju kui vähemusklassis. Otsustuspuude moodustamisel lubame juhumetsa algoritmil kasutada 78 tunnust. Võrdlusmomendi saamiseks loome juhumetsi puude arvuga 128, 512, 1024 ja 4096. Taaskord kasutame selleks paketti *ranger* (vt lisa 6).

4.2.5. OSS alavalimine

One-Sided Sampling alavalimise teostamiseks kasutame R'i paketti *UBL*. Pakett *UBL* on koostatud läbiviimaks mitmeid taasvalimise protseduure nii klassifitseerimise kui ka regressiooni ülesannete jaoks. Paketist *UBL* kasutame Tomeki sidusus meetodi ja Harti algoritmi läbiviimiseks vastavalt funktsioone *TomekClassif* ja *CNNClassif* (Branco *et al.*, 2016). Kui alguses oli treeningandmestikus 28149 vaatlust enamusklassis, siis Tomeki sidusus meetodi kasutamine eemaldas 61 vaatlust ning Harti algoritmi kasutamine 21243 vaatlust. Peale OSS alavalimise meetodi rakendust jäi seega treeningandmestikku 6845 enamusklassi objekti. Otsustuspuude moodustamisel lubame juhumetsa algoritmil kasutada 78 tunnust. Võrdlusmomendi saamiseks loome juhumetsi puude arvuga 128, 512, 1024 ja 4096. Taaskord kasutame selleks paketti *ranger* (vt lisa 7).

4.2.6. Kaalutundliku juhumetsa meetod

Kaalutundliku juhumetsa mudeli loomiseks saame kasutada R'i paketi *ranger* meetodi *ranger* argumendi *class.weights* täpsustamist (Wright *et al.*, 2017). Jätame enamusklassi kaalu üheks ning treenime juhumetsa

mudeleid võrdluse saamiseks kahe erineva vähemusklassi kaaluga. Valime nendeks kaaludeks ühe- ja kahekordse klassidevahelise suhte (vaatluste arv enamusklassis jagatud vaatluste arvuga vähemusklassis). Otsustuspuude moodustamisel lubame juhumetsa algoritmil kasutada 78 tunnust. Võrdlusmomendi saamiseks loome juhumetsi puude arvuga 128, 512, 1024 ja 4096 (vt performance 8).

4.2.7. Bowyeri meetod

Bowyer'i (2011) esitatud meetodi kasutamiseks on mitmeid võimalusi. Alavalimise ja SMOTE meetodi ülevalimise mahud võivad mitmeti varieeruda. Toome näite juhust, kus kasutame traditsioonilist alavalimist enamusklassil vähendades treeningandmestikus selle klassi mahtu nii, et see suhestuks vähemusklassiga nagu 2:1 (meie andmestikul jääb enamusklassi 8828 vaatlust). Kasutades SMOTE algoritmi (viie lähima naabriga) suurendame vähemusklassi vaatluste arvu nii palju, et klasside suhted oleksid võrdsed. Otsustuspuude moodustamisel lubame juhumetsa algoritmil kasutada 78 tunnust ning loome juhumetsa puude arvuga 1024 (vt lisa 9).

4.2.8. CNN ja ülevalimise kombinatsiooni meetod

CNN algoritmi rakendamiseks kasutame taaskord R'i paketi *UBL* meetodit *CNNClassif*. Traditsioonilise ülevalimise rakendamiseks valime juhuslikult tagasipanekuga vähemusklassist vaatlusi testandmestiku juurde. Kuna antud meetodi väljapakkumisel seadsime eesmärgiks OSS alavalimise meetodi tundlikkuse tõstmise (ptk 2,3,), siis ülevalime vähemusklassi nii, et treeningandmestikus on algse enamus- ja vähemusklassi suhe vastavalt 2:3. Otsustuspuude moodustamisel lubame juhumetsa algoritmil kasutada 78 tunnust ning loome juhumetsa puude arvuga 1024 (vt lisa 10).

4.2.9. CNN ja kaalutundliku juhumetsa kombinatsiooni meetod

Nagu ka eelnevalt kasutame CNN algoritmi rakendamiseks R'i paketi *UBL* meetodit *CNNClassif*. Kaalutundliku juhumetsa meetodi rakendamisel kasutame *ranger* argumendi *class.weights* täpsustamist. Kuna ka selle kombinatsiooni eesmärk on OSS alavalimise meetodi tundlikkuse tõstmine, siis jätame enamusklassi kaalu üheks ning treenime juhumetsa mudeleid vähemusklassi kaaluga, mis on kahekordne klassidevaheline suhe. Otsustuspuude moodustamisel lubame juhumetsa algoritmil kasutada 78 tunnust ning loome juhumetsa puude arvuga 1024 (vt lisa 11).

4.2.10. Üle- ja alavalimise kombinatsiooni meetod

Üle- ja alavalimise kombinatsiooni meetodi rakendamiseks on mitmeid võimalusi. Olgu T vaatluste kogu arv treeningandmestikus. Alavalime (juhuslikult tagasipanekuta) enamusklassi vaatluseid nii, et treeningandmestikku jääks $\frac{T}{2}$ enamusklassi vaatlust, ja ülevalime (juhuslikult tagasipanekuga) vähemusklassi vaatluseid nii, et treeningandmestikku jääks samuti $\frac{T}{2}$ vähemusklassi vaatlust. Otsustuspuude moodustamisel lubame juhumetsa algoritmil kasutada 78 tunnust ning loome juhumetsa puude arvuga 1024 (vt lisa 12).

5. Tulemused

Selles peatükis esitleme töö käigus saadud tulemusi. Oleme koostanud eelmises peatükis kirjeldatud juhumetsa mudelid ning nende kõigi kohta välja arvutanud erinevad mudeli ennustustäpsust kirjeldavad suurused (AUC, tundlikkus, spetsiifilisus, OOB viga, täpsus). Bowyer (2011) väitis, et tasakaalustamata andmete puhul on ennustustäpsuse kirjeldamisel tähtsad ROC-kõveraga seotud suurused. Sellest tulenevalt hindame ka meie kõrgemalt mudeleid, mille AUC väärtus on teistest suurem.

Üldiselt peetakse aktsepteeritavateks klassifitseerijateks mudeleid, mille AUC on 0,7 ja 0,8 ühiku vahel. Mudeleid, mille AUC on 0,8 kuni 0,9 ühikut, peetakse väga headeks klassifitseerijateks ja mudeleid, mille AUC on üle 0,9 ühiku, peetakse suurepärasteks klassifitseerijateks (Mandrekar, 2010).

Me lõime juhumetsi puude arvuga 128 (vt tabel 2), 512 (vt lisa 13), 1024 (vt lisa 14) ja 4096 (vt tabel 3). Juhumetsade kasvatamise jaoks valitud puude arvu ja juhumetsade ennustustäpsuse vahel esines positiivne korrelatsioon. Keskmised AUC väärtused (vastavalt 0,7488; 0,7525; 0.7535; 0.7573) tõusid puude arvu kasvul kokku vaid 0.0049 ühikut. Sedavõrd väike võit AUC väärtuses üsna suure ressursi kasutamise erinevuse korral näitab, kui oluliseks võib osutuda andmete töötlemiseks valitud meetod.

Konkurentsitumalt kõige väiksema AUC väärtusega on esindatud traditsioonilise juhumetsa mudel. Traditsioonilisel mudelil on väga kõrge spetsiifilisus (keskmiselt $99,2\%$), kuid väga madal tundlikkus (keskmiselt $15,6\%$). Tulenevalt kõrgest spetsiifilisusest on ka keskmine täpsuse näitaja kõrge. Siin joonistub hästi välja tasakaalustamata andmete alusel treenitud juhumetsa mudeli probleem. Sellised mudelid suudavad vaid väikese osa vähemusklassi kuuluvatest vaatlustest õigesti klassifitseerida. Traditsioonilise mudeli puhul on märgata ka seda, et juhumets 128 puuga ennustas testandmestikul paremini kui temast komplekssemad juhumetsad.

Rakendatud meetoditest esines kõige halvemini SMOTE meetod. SMOTE meetod, mis kasutas vaatluste sünteesimiseks kahte lähimat naabrit (2-NN), esines testandmestikul omakorda halvemini kui SMOTE mee-

Tabel 2: Juhumetsa mudelite ennustustäpsuse näitajad 128 puu korral.

Juhumetsa mudelite ennustustäpsuse näitajad 128 puu korral					
Meetodi nimetus	**AUC**	**Tundlikkus**	**Spetsiifilisus**	**OOB**	**Täpsus**
Traditsiooniline	0,5773	0,1602	0,9813	0,1163	0,8822
Traditsiooniline ülevalimine	0,8117	0,7415	0,8818	0,0660	0,8628
SMOTE (5-NN)	0,7460	0,5703	0,9218	0,0751	0,8741
SMOTE (2-NN)	0,6700	0,4535	0,9464	0,0706	0,8796
Traditsiooniline alavalimine	0,8187	0,9091	0,7283	0,1732	0,7528
OSS alavalimine	0,7617	0,5867	0,9368	0,2552	0,8893
Kaalutundlik (1)	0,7910	0,6633	0,9187	0,1230	0,8841
Kaalutundlik (2)	0,8143	0,7558	0,8727	0,1468	0,8568

tod, mis kasutas sünteesimiseks viit lähimat naabrit (5-NN). Klassifitseerimise võimekuse tõus võrreldes traditsioonilise juhumetsa kasutamisega on märkimisväärne, seda peamiselt tundlikkuse tõusu tõttu. Mõlema SMOTE mudeli tundlikkus on kehvem kui teiste tasakaalustamismeetodite puhul. Esinev OOB viga on võrreldes teiste meetoditega (v.a traditsiooniline ülevalimine) madal.

OSS alavalimise rakendamisel saime keskmiseks AUC väärtuseks $0,7598$. Need alavalimise meetodi mudelid omasid kõige kõrgemat OOB viga (keskmiselt $24,94\%$), kuid vaatamata sellele olid valideerimisandmestikul üsna täpsed. OSS mudeli eesmärk vähendada alavalimisel tekkivat infokadu enamusklassis realiseerus, kuna traditsioonilise alavalimise ja OSS mudeli spetsiifilisuse vahe on ligikaudu 20 protsendipunkti viimase kasuks.

Kaalutundlikutest juhumetsa meetodite mudelitest prognoosis inimeste suremist nädala jooksul paremini mudel, mis kasutas vähemusklassi kaaluna kahekordset klassidevahelist suhet. Ühtlasi on see mudel Mandrekari (2010) jaotuse järgi väga hea klassifitseerija ($\text{AUC} > 0,8$). Mudel, mis kasutas suuremat vähemusklassi kaalu, saavutas keskmiselt märkimisväärselt suurema tundlikkuse (ligikaudu 10 protsendipunkti võrra kõrgema).

Traditsioonilised taasvalimise meetodid olid mõlemad väga hea klassifitseerimise võimekusega. Traditsioonilise ülevalimise meetodi kasutamise baasil loodud juhumetsa mudelid olid väga head klassifitseerijad (keskmine AUC väärtus 0,8135). Selle meetodi kasutamisel saavutatud OOB vea väärtus oli võrreldes teiste meetoditega väikseim, mis viitab parimale sobivusele koostatud mudeli ja treeningandmete vahel. SMOTE meetodi eesmärgiks oli vähendada ülevalimisel võimalikku ülesobitavust. Meie klassifitseerimisülesande korral osutus see ebaefektiivseks. Kuigi loodud SMOTE mudelite OOB vea väärtus tõusis ligikaudu ühe protsendipunkti võrra, kaotasid nad samal ajal märgatavalt AUC väärtuses.

Tabel 3: Juhumetsa mudelite ennustustäpsuse näitajad 4096 puu korral.

Juhumetsa mudelite ennustustäpsuse näitajad 4096 puu korral					
Meetodi nimetus	**AUC**	**Tundlikkus**	**Spetsiifilisus**	**OOB**	**Täpsus**
Traditsiooniline	0,5762	0,1570	0,9954	0,1172	0,8818
Traditsiooniline ülevalimine	0,8143	0,7463	0,8822	0,0631	0,8638
SMOTE (5-NN)	0,7456	0,5687	0,9243	0,0687	0,8761
SMOTE (2-NN)	0,7030	0,4571	0,9481	0,0675	0,8815
Traditsiooniline alavalimine	0,8214	0,9109	0,7320	0,1684	0,7563
OSS alavalimine	0,7616	0,5867	0,9365	0,2451	0,8891
Kaalutundlik (1)	0,7886	0,6575	0,9198	0,1153	0,8842
Kaalutundlik (2)	0,8185	0,7643	0,8727	0,1394	0,8579

Kõikidest väljapakutud kombineerimata meetoditest (vt ptk 2.1. ja 2.2.) klassifitseeris juhtumeid kõige paremini traditsioonilise alavalimise meetod. See meetod saavutas keskmise AUC väärtuse $0,8205$. Alavalimine saavutas eelmainitud meetoditest kõrgeima tundlikkuse (ligikaudu 91%). Tasub välja tuua, et traditsioonilise alavalimise meetodi üldine täpsus on teiste mudelite täpsusest madalam. Seda põhjustab madal spetsiifilisus, mida omakorda põhjustab infokadu enamusklassis. Infokao vähendamiseks on mõeldud OSS alavalimise meetod, kuid OSS meetodi suur kaotus tundlikkuses tähendab traditsioonilise meetodi paremat AUC väärtust. Mõlema alavalimise meetodi OOB vea näitajad võrreldes teiste meetoditega kõrged, mis on oodatav enamusklassis tekkinud infokao tõttu.

Kombineeritud meetodite rakendamisel lõime juhumetsi 1024 puuga (vt tabel 4). Kombineeritud meetodite kasutamisel näeme, et saavutatud tundlikkuse ja spetsiifilisuse näitajad on palju stabiilsemad kui kombineerimata meetodite korral. Kõik koostatud kombineeritud meetodite mudelid on väga head klassifitseerijad ($AUC > 0,8$).

Tabel 4: Juhumetsa mudelite ennustustäpsuse näitajad 1024 puu korral (kombineeritud meetodid).

Juhumetsa mudelite ennustustäpsuse näitajad 1024 puu korral					
Meetodi nimetus	**AUC**	**Tundlikkus**	**Spetsiifilisus**	**OOB**	**Täpsus**
Bowyeri meetod	0,8245	0,8388	0,8102	0,1304	0,8141
CNN ja ülevalimine	0,8319	0,8367	0,8271	0,1800	0,8284
CNN ja kaalutundlik	0,8319	0,8599	0,8038	0,2823	0,8114
Üle- ja alavalimine	0,8282	0,8198	0,8366	0,0882	0,8343

Kõikidest pakutud kombineeritud meetoditest (ptk 2.3.) omasid parimat AUC väärtust kombinatsioonid Harti modifitseeritud algoritmiga. Nendest veidi halvemini esinesid üle- ja alavalimise kombinatsioon ja Bowyeri meetod. Bowyeri meetod saavutas omakorda paremaid tulemusi kui traditsioonilise alavalimise meetod. Kombinatsioonid nii traditsioonilise ülevalimise kui ka kaalutundliku juhumetsaga saavutasid AUC väärtuse $0,8319$. Mõlemad variandid pakkusime välja selleks, et tõsta OSS alavalimise meetodi tundlikkust. Võrreldes OSS meetodiga tõusis mõlema kombineeritud meetodi tundlikkus vähemalt 25 protsendipunkti võrra. Kaalutundliku variandi OOB viga on ligikaudu 10 protsendipunkti võrra suurem kui traditsioonilise ülevalimise vastav näitaja. Mõlema mudeli kohta saab öelda, et nad olid testandmestikul täpsemad kui treeningandmestikul. Kuna mudelid on AUC väärtuse järgi sama head klassifitseerijad, tekib küsimus kumba neist kasutada? See oleneb suuresti uurimisülesandest. Antud juhul soovime prognoosida inimese suremist nädala jooksul peale viimast arstivisiiti. Sellest tulenevalt võiksime eelistada suurema tundlikkusega mudelit, kuna uurijatele oleks eeldatavasti tähtsam teada inimese võimalikust suremisriskist.

6. Kokkuvõte

Antud töö eesmärgiks oli tutvustada ja võrrelda võimalusi efektiivsete juhumetsa mudelite loomiseks tasakaalustamata andmete korral. Töö esimeses osas andsime ülevaate juhumetsa mudelite tööpõhimõtetest ja nende hindamisest. Tutvustasime tasakaalustamata andmetega klassifitseerimismudelite loomise probleemi tausta. Töö teises osas tutvustasime erinevaid lahendusi eelmainitud probleemile. Selleks kasutasime erinevaid väljapakutud lahendusi, et lahendada tasakaalustamata andmetega klassifitseerimisprobleemi MIMIC-III andmestikul. Töö käigus lõime kokku 28 erinevat juhumetsa mudelit. Tavalise juhumetsa algoritmi kasutamisel saavutasime mudeli, mille ennustusvõimekus oli peaaegu võrdväärne juhuslikult klassifitseeriva mudeliga. Mudelid, mille koostasime väljapakutud lahenduste järgi, saavutasid aktsepteeritavaid ja väga häid prognoosimise tulemusi. Koostatud mudelitest saavutasid kõige suuremaid AUC väärtusi kombineeritud meetodid. Kaks parimat klassifitseerijat olid esiteks mudel, kus kasutati treeningandmete modifitseerimiseks CNN algoritmi ning traditsioonilist ülevalimist, ja teiseks mudel, kus kasutati treeningandmete modifitseerimiseks CNN algoritmi ja mudeli loomiseks kaalutundliku juhumetsa meetodit.

Meie töö oli edukas, kuna õnnestus näidata tutvustatud meetodite efektiivsust. Kindlasti vajaks antud bakalaureusetöö ka jätku. Võimalik oleks uurida suurest andmestikust tunnuste valimise (ingl *feature selection*) metoodikaid ning hiljem kombineerida nende kasutamist tasakaalustamata andmetel.

Kasutatud kirjandus

Ali, H., Mohd, S., Saedudin, R., Hussain, K. ja Mushtaq, M. (2019). *Imbalance class problems in data mining: A review*. Indonesian Journal of Electrical Engineering ja Computer Science.

Biau, G. ja Scornet, E. (2015). *A Random Forest Guided Tour*, lk 1, 2.

Bowyer, K.W., Chawla, N.V., Hall, L.O. ja Kegelmeyer, W.P. (2011). *SMOTE: Synthetic Minority Over-sampling Technique*. URL: http://arxiv.org/abs/1106.1813 (vaadatud 18.05.2020).

Branco, P., Ribeiro, R.P. ja Torgo, L. (2016). *UBL: an R Package for Utility-Based Learning*. URL: http://arxiv.org/abs/1604.08079 (vaadatud 18.05.2020).

Breiman, L. (2001). *Random Forests. Machine Learning 45*. Toim. Schapire, Robert E. Holland: Kluwer Academic Publishers, lk 5–32.

Breiman, L., Friedman, J., Olshen, R. ja Stone, C. (1984). *Classification and Regression Trees*. Chapman ja Hall/CRC, lk 114.

Chen, C. ja Breiman, L. (2004). *Using Random Forest to Learn Imbalanced Data*. USA: University of California.

Flach, P.A. (2016). *ROC Analysis*. Toim. Sammut, Claude ja Webb, Geoffrey I. Springer, lk 1, 4. URL: https://research-information.bris.ac.uk/files/94977288/PeterFlachROCAnalysis.pdf (vaadatud 18.05.2020).

Hastie, T., Tibshirani, R. ja Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., lk 282, 283, 463–465, 587–590.

James, G., Witten, D., Hastie, T. ja Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer, lk 176–184, 319.

Johnson, A.E.W. *et al.* (2016). *MIMIC-III, a freely accessible critical care database.* Versioon v1.4. PhysioNet. URL: https://physionet.org/content/mimiciii/1.4/ (vaadatud 18.05.2020).

Kubat, M. *et al.* (1997). *Adressing the Curse of Imbalanced Training Sets: One-Sided Selection.* Kanada: University of Ottawa.

Kuhn, M. ja Johnson, K. (2013). *Applied Predictive Modeling.* Springer, lk 419–435.

Loh, W. (2011). *Classification and Regression Trees.* Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery.

Mandrekar, J.N. (2010). *Receiver Operating Characteristic Curve in Diagnostic Test Assessment.* Journal of Thoracic Oncology, lk 1315 –1316.

Marqués, A., García, V. ja Sánchez, J. (2013). *On the suitability of resampling techniques for the class imbalance problem in credit scoring.* Journal of the Operational Research Society.

Provost, F. (2000). *Machine learning from imbalanced data sets 101.* Proceedings of the AAAI'2000 workshop on imbalanced data.

Rudnicki, W. (2011). *The All Relevant Feature Selection using Random Forest.* Poola: University of Bialystok.

Sammut, C. ja Webb, G.I. (2011). *Encyclopedia of Machine Learning.* 1st. Springer Publishing Company, Incorporated, lk 707.

Tang, T., Garreau, D. ja Luxburg, U. (2018). *When do random forests fail?*

Torgo, Luis. (2014). *An Infra-Structure for Performance Estimation and Experimental Comparison of Predictive Models in R.* URL: http://arxiv.org/abs/1412.0436 (vaadatud 18.05.2020).

Wright, M.N. ja Ziegler, A. (2017). *ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R.* 1, lk. 1–17.

Lisad

Lisa 1. Lõpliku andmetabeli moodustamise kood

```
library(parsnip)
library(recipes)
library(tidyverse)
library(dplyr)


# Vajaminevaid funktsioone
lisa_tulemus = function(tabelid){
  #Kogume kokku koik inimese visiitide kuupaevad
  kuupaevad = bind_rows(
    tabelid$condition_occurrence %>%
      select(person_id, date = condition_end_date),
    tabelid$drug_exposure %>%
      select(person_id, date = drug_exposure_start_date),
    tabelid$observation %>%
      select(person_id, date = observation_date) ,
    tabelid$procedure_occurrence %>%
      select(person_id, date = procedure_date),
    tabelid$visit_occurrence %>%
      select(person_id, date = visit_end_date),
    tabelid$visit_occurrence %>%
      select(person_id, date = visit_start_date),
    tabelid$condition_occurrence %>%
      select(person_id, date = condition_start_date)
  )
  #Leiame inimese koige viimase visiidi ning jatame selle alles
  kuupaevad = kuupaevad %>%
    arrange(person_id, desc(date)) %>%
    distinct(person_id, .keep_all = TRUE)
  #Leiame inimese sunnikuupaeva
  kuupaevad = kuupaevad %>%
    left_join(tabelid$person %>% select(person_id, year_of_birth, month_of_birth,
```

```
    day_of_birth), by = "person_id") %>%
    mutate(synnikp = as_date(str_c(year_of_birth,"-", month_of_birth,
    "-", day_of_birth)))
  #Sunnikuupaeva ja viimase visiidi kaudu arvutame inimese viimase teadaoleva vanuse
  kuupaevad$viimane_teadaolev_vanus = floor(time_length(difftime(as.Date(kuupaevad$date)
  ,as.Date(kuupaevad$synnikp)), "years"))
  #Liidame tabelid suremise ja vanuse kohta, arvutame tunnuse "tulemus",
  #mis utleb meile, kas
  #inimene suri ara nadala jooksul peale viimast visiiti
  tabelid$tulemus = kuupaevad %>%
    left_join(tabelid$death %>% select(person_id, death_date), by = "person_id") %>%
    mutate(death_date = coalesce(death_date, as_date("9999-01-01"))) %>%
    mutate(tulemus = factor(death_date - date < 7,labels = c("Ei", "Jah"))) %>%
    select(person_id, tulemus, viimane_teadaolev_vanus)
  return(tabelid)
}

# Loeme sisse andmed ------------------------------------------------------------
andmed = list(
  condition_occurrence = read_csv(str_glue("{TRAIN}/condition_occurrence.csv"),
  col_types = cols_only(person_id = col_double(), condition_concept_id = col_double(),
  condition_start_date = col_date(),condition_end_date = col_date())),
  procedure_occurrence = read_csv(str_glue("{TRAIN}/procedure_occurrence.csv"),
  col_types = cols_only(person_id = col_double(),
  procedure_concept_id = col_double(), procedure_date = col_date())),
  observation = read_csv(str_glue("{TRAIN}/observation.csv"),
  col_types = cols_only(person_id = col_double(), observation_concept_id = col_double(),
  observation_date = col_date())),
  drug_exposure = read_csv(str_glue("{TRAIN}/drug_exposure.csv"),
  col_types = cols_only(person_id = col_double(), drug_exposure_start_date = col_date())),
  visit_occurrence = read_csv(str_glue("{TRAIN}/visit_occurrence.csv"),
  col_types = cols_only(person_id = col_double(),visit_start_date = col_date(),
  visit_end_date = col_date())),
  death = read_csv(str_glue("{TRAIN}/death.csv"),
  col_types = cols_only(person_id = col_double(), death_date = col_date())),
  person = read_csv(str_glue("{TRAIN}/person.csv"),
  col_types = cols_only(person_id = col_double(), year_of_birth = col_double(),
 month_of_birth = col_double(),day_of_birth = col_double())),
  person_char = read_csv(str_glue("{TRAIN}/person.csv"),
```

```
  col_types = cols_only(person_id = col_double(),
gender_concept_id = col_double(), race_concept_id = col_double())))

# Tekitame tunnused "tulemus" ja viimane teada olev vanus ("viimane_teadaolev_vanus")
andmed = lisa_tulemus(andmed)

# Tekitame tunnused
#Tunnuste tekitamisel loeme sisse koik huvipakkuvad andmetabelid, valime sealt tunnuse,
#mis meid huvitab koos inimese id'ga.
#Seejarel tekitame valitud tunnusest omaette tunnuste jarjestuse,
#mis koosneb tunnuse vaartustest ja nende tunnuste vaartused on
#binaarsed (1,0) tahistamaks,
#kas inimesel esineb see tunnus voi mitte.
diagnoosid = andmed$condition_occurrence %>% select(-condition_end_date) %>%
  mutate(condition_concept_id = str_c("Diag", condition_concept_id)) %>%
  mutate(value = 1) %>%
  pivot_wider(id_cols = person_id, names_from = condition_concept_id,
  values_from = value, values_fn = list(value = max), values_fill = list(value = 0))

protseduurid = andmed$procedure_occurrence %>% select(-procedure_date) %>%
  mutate(procedure_concept_id = str_c("Proc", procedure_concept_id)) %>%
  mutate(value = 1) %>%
  pivot_wider(id_cols = person_id, names_from = procedure_concept_id,
  values_from = value, values_fn = list(value = max), values_fill = list(value = 0))


sugu = andmed$person_char %>%
  select(person_id, gender_concept_id) %>%
  mutate(gender_concept_id = str_c("Sex", gender_concept_id)) %>%
  mutate(value = 1) %>%
  pivot_wider(id_cols = person_id, names_from = gender_concept_id,
  values_from = value, values_fn = list(value = max), values_fill = list(value = 0))

rass = andmed$person_char %>%
  select(person_id, race_concept_id) %>%
  mutate(race_concept_id = str_c("Rass", race_concept_id)) %>%
  mutate(value = 1) %>%
  pivot_wider(id_cols = person_id, names_from = race_concept_id,
  values_from = value, values_fn = list(value = max), values_fill = list(value = 0))
```

```
# Liidame tabelid
#Kui tabelite liitmisel esineb olukord, kus inimesel pole teatud tunnust taidame selle
#lahtri tabelis nulliga.
andmed_loplik = andmed$tulemus %>%
  right_join(andmed$person, by = "person_id")

andmed_loplik = andmed_loplik %>%
  left_join(diagnoosid, by = "person_id") %>%
  mutate_at(.vars = vars(starts_with("X")), .funs = ~ coalesce(., 0))



andmed_loplik = andmed_loplik %>%
  left_join(protseduurid, by = "person_id") %>%
  mutate_at(.vars = vars(starts_with("Proc")), .funs = ~ coalesce(., 0))


andmed_loplik = andmed_loplik %>%
  left_join(sugu, by = "person_id") %>%
  mutate_at(.vars = vars(starts_with("Sex")), .funs = ~ coalesce(., 0))


andmed_loplik = andmed_loplik %>%
  left_join(rass, by = "person_id") %>%
  mutate_at(.vars = vars(starts_with("Rass")), .funs = ~ coalesce(., 0))

# Andmete korrastamine
#Selguse mottes asendame inimeste vanuse vaartuse "300" vaartusega "89".
#MIMIC-III andmete selgituses on valjatoodud, et koik 89+ vanusega inimeste
#vanuseks on 300
andmed_loplik = andmed_loplik %>% mutate(
  viimane_teadaolev_vanus = ifelse(
  viimane_teadaolev_vanus == 300, 89, viimane_teadaolev_vanus)) %>%
  select(-person_id, -year_of_birth, -month_of_birth, -day_of_birth)
save(andmed_loplik, file = "data_tables.RData")
```

Lisa 2. Andmestiku eraldamine treening- ja testandmeteks

```
# Andmete poolitamine
load("data_tables.RData")
andmed = andmed_loplik
set.seed(2020)
#Eraldame nadala jooksul surnud inimesed ja teised
andmed_jah = andmed[andmed$tulemus == "Jah",]
andmed_ei =  andmed[andmed$tulemus == "Ei",]

#Valime juhuslikult indeksid, mille jargi valime vaatluseid treeningandmestikku
#Valime treeningandmestikku 70% vaatlustest
indeksid_jah = sample(1:nrow(andmed_jah), size = floor(nrow(andmed_jah)*0.7))
indeksid_ei = sample(1:nrow(andmed_ei), size = floor(nrow(andmed_ei)*0.7))
#Leitud indeksite jargi eraldame andmed treeningandmestikuks ja testandmestikuks
treening_andmed = rbind(andmed_jah[indeksid_jah,], andmed_ei[indeksid_ei,])
valideerimis_andmed = rbind(andmed_jah[-indeksid_jah,],andmed_ei[-indeksid_ei,])
#Fikseerime tunnuste tyybid
#Kasitleme koiki tunnuseid faktortunnustena (va viimane vanus - pidev)
treening_andmed[] = lapply(treening_andmed, factor)
treening_andmed$viimane_teadaolev_vanus = as.numeric(treening_andmed$viimane_teadaolev_vanus)
valideerimis_andmed[] = lapply(valideerimis_andmed, factor)
valideerimis_andmed$viimane_teadaolev_vanus = as.numeric(valideerimis_andmed$viimane_teadaolev_vanus)

#Salvestame vastavad andmestikud
save(treening_andmed, file = "training.RData")
save(valideerimis_andmed, file = "validation.RData")
```

Lisa 3. Traditsioonilise juhumetsa mudeli loomine ranger paketiga

```
library(ranger)

juhumets1 = ranger(tulemus  ~ .,
                       data=treening_andmed, num.trees = 512 ,mtry = 78)
juhumets2 = ranger(tulemus  ~ .,
                       data=treening_andmed, num.trees = 1024 ,mtry = 78)
juhumets3 = ranger(tulemus  ~ .,
                        data=treening_andmed, num.trees = 4096 ,mtry = 78)
```

Lisa 4. Traditsioonilise ülevalimise rakendamine

```
#Traditsiooniline ulevalimine taasvalime vahemusklassist
#tagasipanekuga nii palju, et andmeid oleks vordselt
treening_jah = treening_andmed[treening_andmed$tulemus == "Jah",]
treening_ei = treening_andmed[treening_andmed$tulemus == "Ei",]
klasside_suhe = nrow(treening_ei)/nrow(treening_jah)
arv_taasvalitavaid = (floor(nrow(treening_jah)*klasside_suhe) - nrow(treening_jah))
treeningandmestiku_lisa = sample_n(treening_jah, size = arv_taasvalitavaid,
                        replace = TRUE)

treening_andmed <- rbind(treening_jah, treening_ei, treeningandmestiku_lisa)
# Treenime mudelid ------------------------------------------------------------
juhumets1 = ranger(tulemus  ~ .,
                         data=treening_andmed, num.trees = 512 ,mtry = 78)
juhumets2 = ranger(tulemus  ~ .,
                         data=treening_andmed, num.trees = 1024 ,mtry = 78)
juhumets3 = ranger(tulemus  ~ .,
                         data=treening_andmed, num.trees = 4096 ,mtry = 78)
```

Lisa 5. SMOTE ülevalimise rakendamine

```
library(performanceEstimation)

#Naide viie lahima naabri kasutamisest SMOTE algoritmi jaoks.
#Muu arvu lahima naabri kasutamiseks tuleb muuta funktsiooni smote argumendi k vaartust vastavalt.

#Funktsioon smote tagastab andmestiku vahemusklassiandmetega + uute synteesitud andmetega
#Kuna me tahame saada enamusklassiga vordselt vaatlusi, siis peame tekitama synteesitud andmeid
#vastavalt klassidele suhtele

treening_jah = treening_andmed[treening_andmed$tulemus == "Jah",]
treening_ei = treening_andmed[treening_andmed$tulemus == "Ei",]
klasside_suhe = nrow(treening_ei)/nrow(treening_jah)

#Uute vaatluste arvu maarab argument perc.over, lahutame sellest yhe,
#kuna tahame treenimisel kasutada ka reaalseid vahemusklassi vaatlusi
#(muidu tekib liiga palju uusi vahemusklassi vaatlusi)
synteesitud_andmed = smote(tulemus ~ ., data = treening_andmed,
perc.over = klasside_suhe - 1,k=5, perc.under = 0)

treening_andmed <- rbind(synteesitud_andmed, treening_ei)

# Treenime mudelid ------------------------------------------------------------
juhumets1 = ranger(tulemus ~ .,
                   data=treening_andmed, num.trees = 512 ,mtry = 78)
juhumets2 = ranger(tulemus ~ .,
                   data=treening_andmed, num.trees = 1024 ,mtry = 78)
juhumets3 = ranger(tulemus ~ .,
                   data=treening_andmed, num.trees = 4096 ,mtry = 78)
```

Lisa 6. Traditsioonilise alavalimise rakendamine

```
#Traditsiooniline alavalimine: taasvalime enamusklassist tagasipanekuta nii palju vaatlusi,
#et andmeid oleks vordselt
treening_jah = treening_andmed[treening_andmed$tulemus == "Jah",]
treening_ei = treening_andmed[treening_andmed$tulemus == "Ei",]


arv_taasvalitavaid = nrow(treening_jah)
treeningandmestiku_lisa = sample_n(treening_ei, size = arv_taasvalitavaid,
                          replace = FALSE)

treening_andmed <- rbind(treening_jah, treeningandmestiku_lisa)

# Treenime mudelid ------------------------------------------------------------
juhumets1 = ranger(tulemus  ~ .,
                         data=treening_andmed, num.trees = 512 ,mtry = 78)
juhumets2 = ranger(tulemus  ~ .,
                         data=treening_andmed, num.trees = 1024 ,mtry = 78)
juhumets3 = ranger(tulemus  ~ .,
                         data=treening_andmed, num.trees = 4096 ,mtry = 78)
```

Lisa 7. OSS alavalimise rakendamine

```
library(UBL)

#Valime dist = "HEOM" sest tegeleme molemal juhul andmetega,
#mis sisaldavad nii nominaalseid kui ka numbrilisi
#meetodeid (soovitatud dokumentatsioonis)

#Tomeki algoritmi jaoks maarame rem = "maj", kuna tahame,
#et eemaldataks vaid enamusklassi vaatluseid
treening_andmed = TomekClassif(tulemus ~ ., dat = as.data.frame(treening_andmed),
dist = "HEOM", Cl = "Ei", rem = "maj")[[1]]
treening_andmed = CNNClassif(tulemus ~ ., dat = as.data.frame(treening_andmed),
dist = "HEOM", Cl = "Jah")[[1]]

# Treenime mudelid ------------------------------------------------------------
juhumets1 = ranger(tulemus ~ .,
                   data=treening_andmed, num.trees = 512 ,mtry = 78)
juhumets2 = ranger(tulemus ~ .,
                   data=treening_andmed, num.trees = 1024 ,mtry = 78)
juhumets3 = ranger(tulemus ~ .,
                   data=treening_andmed, num.trees = 4096 ,mtry = 78)
```

Lisa 8. Kaalutundliku juhumetsa mudeli treenimine

```
#Arvutame klassidevahelise suhte
klasside_suhe = nrow(treening_andmed[treening_andmed$tulemus == "Ei",])/
nrow(treening_andmed[treening_andmed$tulemus == "Jah",])

# Treenime mudelid ------------------------------------------------------------
juhumets1_1 = ranger(tulemus  ~ .,
                        data=treening_andmed, num.trees = 512 ,mtry = 78,
                        class.weights = c(1, klasside_suhe))
juhumets2_1 = ranger(tulemus  ~ .,
                        data=treening_andmed, num.trees = 1024 ,mtry = 78,
                        class.weights = c(1, klasside_suhe))
juhumets3_1= ranger(tulemus  ~ .,
                        data=treening_andmed, num.trees = 4096 ,mtry = 78,
                        class.weights = c(1, klasside_suhe))

juhumets1_2 = ranger(tulemus  ~ .,
                          data=treening_andmed, num.trees = 512 ,mtry = 78,
                          class.weights = c(1, 2*klasside_suhe))
juhumets2_2 = ranger(tulemus  ~ .,
                          data=treening_andmed, num.trees = 1024 ,mtry = 78,
                          class.weights = c(1, 2*klasside_suhe))
juhumets3_2= ranger(tulemus  ~ .,
                         data=treening_andmed, num.trees = 4096 ,mtry = 78,
                         class.weights = c(1, 2*klasside_suhe))
```

Lisa 9. Bowyeri meetodi rakendamine

```
#Tekitame synteesitud vaatlusi sama palju nagu on vahemusklassis vaatlusi
#funktsioon smote tagastab andmestiku vahemusklassiandmetega + synteesitud andmetega
synteesitud_andmed = smote(tulemus ~ ., data = treening_andmed, perc.over = 1,k=5, perc.under = 0)

#Enamusklassis traditsioonilise alavalimise teostamiseks eraldame enamusklassi vaatlused
treening_ei = treening_andmed[treening_andmed$tulemus == "Ei",]
#Valime enamusklassist tagasipanekuta sama palju vaatluseid,
#kui on vahemusklassis + synteesitud vaatluste hulgas
arv_taasvalitavaid = nrow(synteesitud_andmed)
treeningandmestiku_lisa = sample_n(treening_ei, size = arv_taasvalitavaid,
                          replace = FALSE)
#Paneme tekitatud andmestikud kokku
treening_andmed <- rbind(synteesitud_andmed, treeningandmestiku_lisa)

# Treenime mudeli ------------------------------------------------------------

juhumets1 = ranger(tulemus ~ .,
                           data=treening_andmed, num.trees = 1024 ,mtry = 78)
```

Lisa 10. CNN ja ülevalimise kombinatsiooni meetod

```
#Viime labi CNN algoritmi
treening_andmed = CNNClassif(tulemus ~ ., dat = as.data.frame(treening_andmed),
dist = "HEOM", Cl = "Jah")[[1]]

#Jagame andmestiku kaheks, et teostada ulevalimist
treening_jah = treening_andmed[treening_andmed$tulemus == "Jah",]
treening_ei = treening_andmed[treening_andmed$tulemus == "Ei",]

#Suhte 3:2 saamiseks arvutame taasvalimiste arvu
arv_taasvalitavaid = max(0,floor(nrow(treening_ei)*3/2)-nrow(treening_jah))

treeningandmestiku_lisa = sample_n(treening_jah, size = arv_taasvalitavaid,
                          replace = TRUE)

treening_andmed <- rbind(treening_jah, treening_ei, treeningandmestiku_lisa)



# Treenime mudeli ------------------------------------------------------------
juhumets1 = ranger(tulemus ~ .,
                           data=treening_andmed, num.trees = 1024 ,mtry = 78)
```

Lisa 11. CNN ja kaalutundliku juhumetsa kombinatsiooni meetod

```
#Viime labi CNN algoritmi
treening_andmed = CNNClassif(tulemus ~ ., dat = as.data.frame(treening_andmed),
dist = "HEOM", Cl = "Jah")[[1]]

#Arvutame klasside esinemissuhte
klasside_suhe = nrow(
treening_andmed[treening_andmed$tulemus == "Ei",])/nrow(
treening_andmed[treening_andmed$tulemus == "Jah",])
#Kasutame vahemusklassi kaaluna kahe kordset klassidevahelist suhet
# Treenime mudeli ------------------------------------------------------------
juhumets1 = ranger(tulemus ~ .,
                           data=treening_andmed, num.trees = 1024 ,mtry = 78,
                           class.weights = c(1, 2*klasside_suhe))
```

Lisa 12. Traditsioonilise ala- ja ülevalimise kombinatsiooni meetod

```
#Eraldame andmed taasvalimismeetodite rakendamiseks
treening_jah = treening_andmed[treening_andmed$tulemus == "Jah",]
treening_ei = treening_andmed[treening_andmed$tulemus == "Ei",]
#Leiame vaatluste arvu, millega molemad klassid on esindatud
arv_taasvalitavaid = (nrow(treening_ei)+nrow(treening_jah))/2

#Teostame taasvalimismeetodid
treeningandmestiku_lisa = sample_n(treening_jah, size = arv_taasvalitavaid - nrow(treening_jah),
                          replace = TRUE)
treening_ei =  sample_n(treening_ei, size = arv_taasvalitavaid,
                        replace = FALSE)

#Paneme andmestikud kokku
treening_andmed <- rbind(treening_jah, treening_ei, treeningandmestiku_lisa)

# Treenime mudeli ------------------------------------------------------------
juhumets1 = ranger(tulemus ~ .,
                        data=treening_andmed, num.trees = 1024 ,mtry = 78)
```

Lisa 13. Kombineerimata meetodite tulemusi 512 puu korral

Tabel 5: Juhumetsa mudelite ennustustäpsuse näitajad 512 puu korral.

Juhumetsa mudelite ennustustäpsuse näitajad 512 puu korral					
Meetodi nimetus	**AUC**	**Tundlikkus**	**Spetsiifilisus**	**OOB**	**Täpsus**
Traditsiooniline	0,5743	0,1532	0,9956	0,1166	0,8812
Traditsiooniline ülevalimine	0,8120	0,7426	0,8815	0,0640	0,8627
SMOTE (5-NN)	0,7470	0,5714	0,9227	0,0703	0,8750
SMOTE (2-NN)	0,7030	0,4577	0,9484	0,0671	0,8819
Traditsiooniline alavalimine	0,8202	0,9102	0,7302	0,1699	0,7546
OSS alavalimine	0,7570	0,5788	0,9353	0,2518	0,8869
Kaalutundlik (1)	0,7890	0,6565	0,9215	0,1159	0,8856
Kaalutundlik (2)	0,8174	0,7637	0,8711	0,1418	0,8566

Lisa 14. Kombineerimata meetodite tulemusi 1024 puu korral

Tabel 6: Juhumetsa mudelite ennustustäpsuse näitajad 1024 puu korral.

Juhumetsa mudelite ennustustäpsuse näitajad 1024 puu korral					
Meetodi nimetus	**AUC**	**Tundlikkus**	**Spetsiifilisus**	**OOB**	**Täpsus**
Traditsiooniline	0,5746	0,1538	0,9954	0,1176	0,8814
Traditsiooniline ülevalimine	0,8161	0,7495	0,8826	0,0630	0,8646
SMOTE (5-NN)	0,7477	0,5714	0,9241	0,0682	0,8763
SMOTE (2-NN)	0,7040	0,4593	0,9486	0,0674	0,8824
Traditsiooniline alavalimine	0,8217	0,91068	0,7327	0,1696	0,7568
OSS alavalimine	0,7589	0,5808	0,9370	0,2455	0,8887
Kaalutundlik (1)	0,7879	0,6565	0,9193	0,1165	0,8836
Kaalutundlik (2)	0,8170	0,7632	0,8709	0,1409	0,8563

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Markus Haug,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose ”Juhumetsa mudelite loomine tasakaalustamata andmetega”, mille juhendaja on Raivo Kolde, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Markus Haug

18.05.2020