Tartu Ülikool
Matemaatika-informaatikateaduskond
Matemaatilise statistika instituut

Valentin Skokov

Krediidiskooringu süsteemi loomine tehisõppe meetodite abil

Bakalaureusetöö (9 EAP)

Juhendaja: Kalev Pärna

Tartu 2015

Krediidiskooringu süsteemi loomine tehisõppe meetodite abil

Käesoleva töö eesmärk on luua tehisõppe meetoditel baseeruv süsteem isiku individuaalse krediidiskoori määramiseks ja selle põhjal krediidiandmise otsuse tegemiseks. Potentsiaalseteks kasutajateks on pangad ja laenufirmad, kelle otsene huvi on vähendada ning võimalusel vältida finantskahju, mis tekib seoses krediidi mittetagastamisega laenusaaja poolt.

Töö esimeses peatükis antakse ülevaade tehisõppest ja selle meetoditest koos nende rakenduste näidetega. Töö teises peatükis keskendukse CART meetodile, selle rakenduste piiridele, puudustele ja headele külgedele.

Töö kolmandas peatükis rakendatakse kirjeldatud meetodit andmestikule, mis modelleerib reaalselt võimalikku olukorda ning mis sisaldab informatsiooni isikute sissetulekute, krediidiajaloo ning mõnede teiste isiklike andmete kohta.

Märksõnad: tehisõpe, andmeteadus, otsustuspuu

Creation of Credit Score system using Machine Learning methods

The purpose of this thesis is to create ready to use, off the shelf system for setting individual credit score and making decisions about loan giving, implementing Machine Learning. Potential users are banks and loan firms, whose interest is to minimize and potentially eliminate financial loss that is brought by loans not being paid off.

The first and the smallest part of the thesis covers the overview of Machine Learning with examples of its implementations. The second part of the thesis concentrates on CART method, its implementation borders, advantages and disadvantages.

The third part of the thesis covers use of CART with a dataset, which models real-life situation and consists of personal data like credit history, income, marriage status and such.

Keywords: machine learning, data science, decision tree

Sisukord

Sissejuhatus

Laenud ja nendega seotud probleemid ja protsessid on tänapäevase elu lahutamatud osad, ning tõenäosus leida isikut, kes ei ole oma elus võtnud ühtegi laenu, väheneb iga aastaga. Panga või laenufirma jaoks on väga tähtis vältida finantskadu ning saada laenatud raha tagasi koos kasumiga. Isikute jaoks on väga tähtis võtta enda jaoks võimalikult parimatel tingimustel laenu. Ei ole saladus, et pangad lisavad laenuintressi sisse ka riski oma raha kaotada ning mida vähem pank vaatab, kellele laenu anda, seda suurema riskiga pank arvestab intressis, mis omakorda teeb laenutingimused halvemaks. Vähendada riski ning teha laenutingimused paremaks on siis nii pankade kui ka laenuvõtjate huvides.

Kõige rohkem huvi pakub aga tehisõpe, mis haarab endasse suure hulga meetodeid, mille rakendusi on veelgi rohkem: käekirja ja kõne eristamine, isejuhtivad autod, tehisintellekt ning palju-palju muud. Võrreldes teiste probleemidega on krediidiskooringu süsteem suhteliselt lihtne, aga vaatamata sellele huvitav proov ja esimene samm tehisõppes.

Töö autor tänab juhendajat Kalev Pärna paranduste, selgituste ning kasulike nõuannete eest.

Bakalaureusetöö on kirjutatud tekstitöötlusprogrammiga OpenOffice Writer, andmete analüüs ja tehisõppe meetodite rakendamine viidi läbi tarkvarapakettide R ja RStudio abil.

1. Tehisõpe

Enne tehisõppest rääkimist tuleb lahti seletada mõned terminid:

- Treeningandmed (algandmed, ing. *training data*) - andmed, mille peal tehisõppe algoritm/programm õpetab otsitavat funktsiooni.
- Sihtfunktsioon (ing. *target function*) – funktsioon, mis võib esineda hüpoteesi, programmi, arvulise funktsiooni või mistahes teisel kujul. See on funktsioon, mida me tahame optimeerida.

1.1 Ülevaade tehisõppest

Foster Provost [1] defineerib tehisõppe niimoodi: ***Tehisõpe*** *on teadusharu mis keskendub algoritmidele, mille kohta võib öelda, et nad „õpivad“* (ing. ***Machine Learning*** *is the field of scientific study that concentrates on induction algorithms and on other algorithms that can be said to „learn“*). Õppimine selles kontekstis ei tähenda inimeste õppimist tavalises mõttes, vaid mingi tulemuse parandamist kogemusega. Väga hea definitsiooni õppimisele annab Tom M. Mitchell [2]: *Öeldakse, et programm* ***õpib****, kui programmi ülesannete T sooritamine parandab vastavalt headuse näitajale P kogemusega E* (ing. *A computer pogram is said to* ***learn*** *from experience E with respect to some class of tasks T and performace measure P, if its performance at tasks in T, as measured by P, improves with experience E*).

Krediidiskooringu süsteemi kontekstis võime defineerida parameetrid järgmiselt:

- Ülesanne T: krediidiskoori määramine ja otsuse tegemine.
- Headuse näitaja P: protsent inimesi, kelle kohta oli tehtud õige otsus või 2x2 vigade tabel.
- Kogemus E: süsteemi rakendamine reaalses elus.

Meie juhul see, et programm õpib tähendab õigete otsuste protsendi suurendamist programmi kasutamisega. Selle jaoks vajab programm nii algandmeid, mis sisaldaksid infot laenu tagasimakse kohta, kui ka tagasisidet tehtud otsustest.

Tehisõppe tähtsamad probleemid on:

- Missugused algoritmid eksisteerivad ja millised on nende rakenduste piirid?
- Kui palju algandmeid on vaja?

- Kuidas saab rakendada eelteadmisi probleemi kohta?
- Missugust funktsiooni peab süsteem õppima ning kas saab seda protsessi automatiseerida?

Mingil töö hetkel püüame leida vastuse nendele küsimustele krediidiskooringu süsteemi loomise vaatenurgast.

1.2 Põhilised tehisõppe meetodid

Tehisõpe on kiiresti arenev teadusharu, ning kõiki tehisõppe meetodeid on võimatu vaadelda selle töö raames. Edaspidi keskendume üksnes nendele meetoditele, millest räägitakse raamatus *Machine Learning* [2]. Lisaks tuleb arvestada et tehisõpe ei ole dogmaatiline ning et erinevate meetodite omavaheline kombineerimine on laialt levinud.

Kontsepti õppimine (ing. *Concept Learning)*

Kontsepti õppimine on formuleeritav kui ülesanne leida hüpotees eeldefineeritud hulgast, mis kõige paremini sobib treeningandmetele. Mõned selle meetodi algoritmid on *General-to-Specific Ordering* ning *Candidate-Elimination*. Meetodi miinuseks on vajadus defineerida hüpoteeside hulk: alati eksisteerib tõenäosus, et tegelikult kehtiv hüpotees ei ole selles hulgas.

Otsustuspuu õppimine (ing. *Decision Tree Learning*)

Otsustuspuu on üks praktikas kõige rohkem kasutatavatest meetoditest. Meetodi plussiks on see, et ta väga hästi käivitub mürarikaste andmetega (ing. *Noisy Data*). Põhilised meetodi algoritmid on ID3, C4.5 ja selle kõige uuem versioon C5.0, CART, ASSISTANT, CHAID, MARS. Selle töö teine peatükk keskendub just CART algoritmile, täpsemalt selle osale.

Kunstlikud närvivõrgud (ing. *Artificial Neural Networks*)

Kunstlikud närvivõrgud annavad üldise ning praktilise meetodi funktsioonide õppimiseks, kus funktsioonid võivad olla nii pidevad, diskreetsed, kui ka vektorilised. Õppematerjaliks on *Output-Input* paarid, väga hästi käivitub meetod vigaste treeningandmetega. Meetodit on edukalt rakendatud näiteks kõne ning piltide eristamise juures.

Vaatlusel põhinev õppimine (ing. *Instant Based Learning*)

Võrreldes teistega nimetatakse seda meetodit aeg-ajalt „laisaks“, kuna meetod ei konstrueeri mingeid üldistusi ning kirjeldusi otsitavaks funktsiooniks, vaid lihtsalt hoiab treeningandmeid. Üldistusi nende andmete pealt tehakse ainult siis, kui uus juhtum vajab klassifitseerimist; siis uuritakse uue juhtumi suhet treeningandmetega, et määrata talle funktsiooni väärtusi. Kasutatavad

algoritmid on *Nearest Neighbour* ja *Locally Weighted Regression*. Meetodi plussiks on võimalus iga uue juhtumi puhul hinnata funktsiooni väärtust lokaalselt ja erinevalt teistest juhtumitest. Kuigi töö ei keskendu sellele meetodile, kasutatakse mõnda tema elementi kolmandas peatükis.

Geneetilised algoritmid (ing. *Genetic Algorithms*)

Nagu kontsepti õppimisel, vajavad geneetilised algoritmid eeldefineeritud hüpoteeside hulka. Selle asemel, et lihtsalt leida kõige paremini sobiva hüpoteesi antud hulgast, defineeritakse hüpoteesi sobivuse näitaja, mis leitakse iga hulgas oleva hüpoteesi jaoks. Seejärel simuleeritakse nö „evolutsioon“, ehk hüpoteeside osad rekombineeritakse omavahel ning tõenäosus, et hüpoteesi osa kasutatakse rekombineerimisel sõltub selle hüpoteesi sobivusest. Protsessi korratakse järglaste hulgas, siis järglaste järglaste hulgas, kuni on saadud soovitud sobivusega hüpotees. Geneetilisi algoritme rakendati edukalt roboti kontrollimiseks ning koos kunstlike närvivõrkudega.

Reeglite õppimine (ing. *Learning Set of Rules*)

Üks kõige paremini esitatav viis hüpoteesi õppimiseks on hulk „kui-siis“ reegleid. Kõige tähtsamad meetodi algoritmid on *Sequential Covering* ning *Inductive Logic Programmind ILP.*

Analüütiline õppimine (ing. *Analytical Learning*)

Analüütilise õppimise põhimõtteks on täiendada eeltoodud meetodeid kõrvalteadmistega, mis ei ole saadavad treeningandmetest. Üheks niisuguseks algoritmiks on *Explanation Based Learning EBX*, mis hästi sobib planeerimisülesannete jaoks.

Tugevdatud õppimine (ing. *Reinforcement Learning*)

Võib-olla kõige huvitavam, aga seega ka kõige raskem tehisõppe meetod. Tugevdatud õpe keskendub sellele, kuidas autonoomne agent, mis tunneb oma ümbrust ning tegutseb, saab valida optimaalseid tegevusi oma eesmärgi saavutamiseks. Iga kord, kui agent tegutseb, võib treener anda kas „tasu“ või „karistust“ agendile. Agendi ülesanne on õppida sellest kaudsest tagasisidest, et valida käitumine, mis toob kokkuvõttes kaasa suurima tasu.

2. CART algoritm

CART on akronüüm, mis tähendab *Classification and Regression Trees* ning on üks otsustuspuude õppimise algoritmidest. Juba nimest on näha, et see algoritm koosneb kahest osast ning edasi keskendub töö viimasele osale - regressioonipuudele. See peatükk põhineb Leo Breimani raamatul *Classification and Regression Trees* [3].

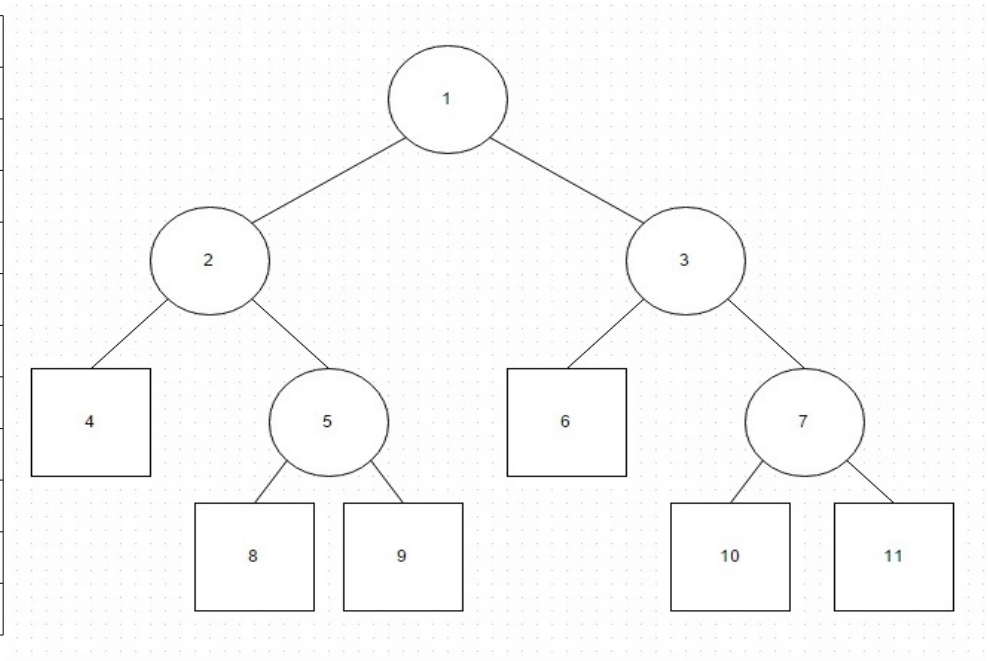
2.1 Puude terminoloogia

Puu on struktuur, mis koosneb lõplikust mittetühjast arvude hulgast T ning funktsioonidest $v()$ ning $p()$, mis rahuldavad järgmist tingimusi:

- Iga $t \in T$ jaoks kas $v(t) = p(t) = 0$ või $v(t) > t \wedge p(t) > t$.
- Iga $t \in T$, peale väikseima hulga T arvu, jaoks leidub paraajasti üks $s \in T$ niimoodi, et $t = v(s) \vee t = p(s)$.

Lihtsustamiseks: edaspidi T nimetatakse *puuks* ning iga T elementi nimetatakse *puu sõlmeks*.

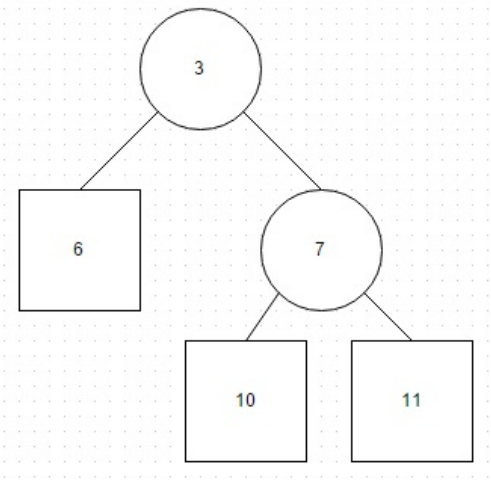
t	v(t)	p(t)
1	2	3
2	4	5
3	6	7
4	0	0
5	8	9
6	0	0
7	10	11
8	0	0
9	0	0
10	0	0
11	0	0



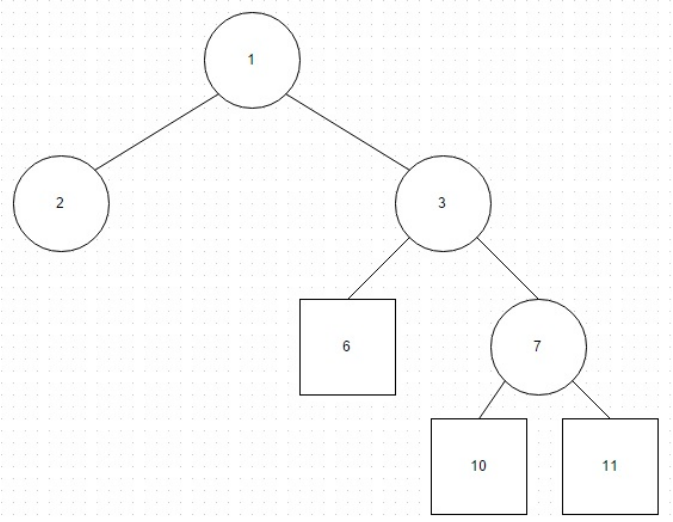
Joonis 1. Puu tabeli kujul ja skeemina.

Puu vähimat sõlme nimetatakse *juureks*. Sõlme s nimetatakse t *järglaseks*, kui

$t = v(s) \vee t = p(s)$, sõlme t nimetatakse siis s *vanemaks*. Puu juurel ei ole vanemat. Puu sõlm, mis ei ole mingi teise sõlme vanemaks, on lõplik, ning teda nimetatakse *leheks*. Olgu meil mittetühi alamhulk $T_1 \subset T$. Defineerime f-d $v_1(t) = \begin{cases} v(t), kui\ v(t) \in T_1 \\ 0, muul\ juhul \end{cases}$ ning $p_1(t) = \begin{cases} p(t), kui\ p(t) \in T_1 \\ 0, muul\ juhul \end{cases}$. T_1 on *osapuu*, kui T_1 , $v_1(t)$, ning $p_1(t)$ moodustavad puud. Antud $t \in T$ jaoks tema ning kõikide tema järglaste ühendit T_t nimetatakse nimetatakse *haruks*. Võttes ülaltoodud puu esialgseks, võime teha järgmised joonised.



Joonis 2.



Joonis 3.

Joonisel 2 olev struktuur on nii *haru* kui ka *osapuu*.

Joonisel 3 olev struktuur on *osapuu* aga ei ole *haru*.

Seos osapuude ja harude vahel on järgmine: iga *haru* on samas ka *osapuu,* aga suvaline *osapuu* ei pruugi olla *haru*. Osapuud T_1 nimetatakse T kärbitud osapuuks, kui nendel on sama juur.

2.2 Regressioonipuu

Regressioonipuu on mingis mõttes konkurent lineaarsetele regressioonimudelitele ning kasutab sarnast terminoloogiat. Olgu X mõõtmiste hulk, siis iga juhtum koosneb andmetest (x, y), kus $x \in X$ ning y on sihtfunktsiooni väärtus. Ennustus reegel, ehk prognoos $d(x)$ on reaalväärtustega funktsioon X -st. Regressioonanalüüsi ülesanneks ongi leida prognoos $d(x)$ kasutades andmeid L.

Puu-kujuline prognoos on struktuur, mis lahutab hulga X binaarsete tükelduste (poolituste) abil lõplikeks puu leheteks. Igas sõlmes ennustatud väärtus $y(t)$ on konstantne.

Et ehitada puu-kujulist prognoosi alustades treeningandmetest L on vaja 3 asja:

- Poolituste meetod igas mittelõplikus puu harus.
- Reegel lõplike lehtede määramiseks.
- Reegel väärtuse $y(t)$ määramiseks igas lõplikus lehes.

Osutub, et kõige lihtsam on viimane osa, seega sellest on mõistlik alustada.

2.3 Ennustusväärtuse määramine lõplikus lehes

Olgu meil treeningandmestik L, mis koosneb N vaatlusest $(x_{1,}y_1),\ldots,(x_N,y_N)$ ning olgu prognoos $d(x)$ konstrueeritud nende andmete põhjal. Defineerime keskmise ruutvea

$$R(d)=E(Y-d(X))^2.$$

Siis $R(d)$ on oodav keskmine ruutviga kasutades prognoosi $d(X)$. $R(d)$ hinnanguks võime kasutada suurust

$$\hat{R}(d)=\frac{1}{N}\sum_{i=1}^{N}(y_i-d(x_i))^2.$$

Meie ülesandeks on leida niisugune $y(t)$, mis minimiseerib summa $\hat{R}(d)$.

Teoreem 1

Keskmise ruutvea minimiseerib igas lehes olevate vaatluste väärtuste keskmine ehk

$$\bar{y}(t)=\frac{1}{N}(t)\sum_{x_N\in t}y_N.$$

Tõestus:

On teada, et suurust $\sum_{i=1}^{N}(y_i-a)^2$ minimiseerib arv $a=\frac{1}{N}\sum_{i=1}^{N}y_i$. Sarnaselt, iga y_n alamhulka $y_{n^{\check{}}}$ suurust $\sum_{n^{\check{}}}(y_{n^{\check{}}}-a)^2$ minimiseerib $y_{n^{\check{}}}$ keskmine.

m.o.t.t.

Edasi võtame iga lõpliku lehe ennustatud väärtuseks $\bar{y}(t)$. Siis, tähistades $R(d)$ asemel $R(T)$ kirjutame

$$R(T)=\frac{1}{N}\sum_{t\in\tilde{T}}\sum_{x_n\in t}(y_n-\bar{y}(t))^2,$$

kus $\tilde{T}$ on lõplike lehtede hulk. Lehe t keskmine ruutviga on

$$R(t)=\frac{1}{N}\sum_{x_n\in t}(y_n-\bar{y}(t))^2,$$

seega

$$R(T)=\sum_{t\in\tilde{T}}R(t).$$

2.4 Parim poolitus

Edasi meid huvitab, kuidas jagada meie puu harud ehk milline poolitus valida. Olgu meil S poolituste hulk praeguse lõpliku haru t jaoks. Defineerime parima poolituse $\dot{s}$:

Parim t poolitus $\dot{s}\in S$ on niisugune poolitus, mis kõige rohkem vähendab $R(T)$ väärtust. Kui haru t poolitus s jagab haru t_L -ks ja t_R -ks, defineerime

$$\Delta R(s,t)=R(t)-R(t_L)-R(T_R).$$

Siis parim poolitus $\dot{s}$ on niisugune, et

$$\Delta R(\dot{s},t)=max(\Delta R(s,t)\mid s\in S).$$

Seega regressionipuu ehitatakse jagades puu harud iteratiivselt niimodi, et minimiseerida $R(T)$.

2.5 Puu kärpimine

Kui oleme otsustanud, kuidas puud ehitada tekib viimane ja kõige täthsam küsimus: millal peatuda? Probleem seisneb selles, et $R(t)$ -l on järgmine omadus

Omadus 1

Iga t poolitus, mis jagab t t_L -ks ning t_R -ks korral kehtib $R(t)\geqslant R(t_L)+R(t_R)$.

Omaduse 1 tõestus:

lahutame $R(t)$ kaheks liidetavaks ja hindame neid liidetavaid alt **teoreemi 1** abil:

$$R(t)=\frac{1}{N}\sum_{x_n\in t}(y_n-\bar{y}(t))^2=\frac{1}{N}[\sum_{x_n\in t_R}(y_n-\bar{y}(t))^2+\sum_{x_n\in t_L}(y_n-\bar{y}(t))^2]$$
$$\geqslant\sum_{x_n\in t_R}(y_n-\bar{y}_R(t))^2+\sum_{x_n\in t_L}(y_n-\bar{y}_L(t))^2=R(t_L)+R(t_R)$$

m.o.t.t.

Näeme, et eelkirjeldatud iteratiivne protsess jagab puu sõlmed niikaua kuni on olemas andmeid jagamiseks. Et leida optimaalne puu, ehitatakse kõigepealt puu T_{max}, korrates iteratiivset protsessi, kuni iga $t\in\tilde{T}_{max}$, $N(t)\leqslant N_{min}$, kus N_{min} on suvaliselt valitud väike arv, näiteks 5. Defineerime vea-keerukuse määra

$$R_\alpha(T):=R(T)+\alpha|\tilde{T}|\ ,$$

kus α on reaalarv. Iga α väärtuse jaoks on nüüd võimalik leida osapuu $T(\alpha)\subseteq T_{max}$ mis minimiseerib $R_\alpha(T)$, ehk

$$R_\alpha(T(\alpha))=min(R_\alpha(T)\mid T\subseteq T_{max})$$

Funktsioon $R_\alpha(T)$ on lineaarkombinatsioon puu veast ja tema lõplike lehtede arvust, arvu α võib interpreteerida nagu „karistust“ iga lõpliku lehe jaoks. Kuigi α on reaalarv, siis suurimal puul T_{max} saab olla ainult lõplik hulk osapuid, seega leidub $T(\alpha)$, mis on minimiseeriv puu antud α jaoks ning ta jääb minimiseerivaks, kuni α suureneb kuni mingi hüppepunktini $\dot{\alpha}$ ning uus puu $T(\dot{\alpha})$ on minimiseeriv kuni järgmise hüppepunktini $\ddot{\alpha}$ jne. Märkame, et kui α on piisavalt suur, siis $R_\alpha(T)$ minimiseerib puu, mis koosneb ainult juurest $\{t_1\}$, kui $\alpha=0$, siis $R_\alpha(T)$ minimiseerib T_{max}. Tekivad küsimused, kas leidub ühene $T\subseteq T_{max}$, mis minimiseerib $R_\alpha(T)$ ning kui jadas $T_1, T_2, T_3\ldots$ iga järgmine puu on saadud eelmise puu kärpimisega, siis kas säilib sisaldavus $T_1\supset T_2\supset\ldots\supset\{t_1\}$?

Defineerime vähima minimiseeriva osapuu $T(\alpha)$ tingimustega:

$$\begin{aligned}&1.\ R_\alpha(T(\alpha))=min\ (R_\alpha(T)\mid T\subseteq T_{max})\\&2.\ kui\ R_\alpha(T)=R_\alpha(T(\alpha))\ siis\ T(\alpha)\subseteq T\end{aligned}\qquad(1)$$

Tingimusi rahuldav osapuu on vähim osapuudest, mis minimiseerivad $R_\alpha(T)$. Selge, et kui niisugune osapuu eksisteerib, siis ta on ühene, seega küsimus on tema eksisteerimises. Osutub, et kehtib järgmine teoreem.

Teoreem 2

Iga α jaoks leidub vähim minimiseeriv osapuu, mis rahuldab tingimusi (1) .

Teoreemi 2 tõestuse annab alltoodud algoritm puu kärpimiseks, mis annab võimaluse iga α jaoks leida vähima minimiseeriva osapuu, mis rahuldaks tingimusi (1) . Alustame nüüd puu kärpimist.

Kärpimise alguspunktiks ei ole T_{max} vaid $T_1 = T(0)$, ehk T_1 on vähim T_{max} osapuu, mis rahuldab $R(T_1) = R(T_{max})$. Olgu t_L ning t_R suvalised kaks T_{max} lehte, mis on saadud nende otsese vanema t jagamisega. Omaduse 1 tõttu teame, et $R(t) \geqslant R(t_L) + R(t_R)$; kui $R(t) = R(t_L) + R(t_R)$, siis kärbime ära t_L ning t_R . Jätkame seni, kuni ei ole võimalik midagi kärpida ning lõpuks saamegi puu T_1 . On selge, et $R(T_1) = R(T_{max})$: kui ei olnud võimalik mitte mingeid lehti ära kärpida, siis $T_1 = T_{max}$, kui aga mõned lehed kärbiti paariviisi ära , siis $R(T_1)$ erineb $R(T_{max})$ - st just nendes kohtades, ehk

$$R(T_1) - R(T_{max}) = \sum_{t \in \tilde{T}_{max}} R(t) - \sum_{t \in \tilde{T}_1} R(t) = \sum_{t \in T_{kärbitud}} R(t) - R(t_L) - R(t_R) = 0 \ .$$

Iga T_1 haru T_t jaoks defineerime nüüd

$$R(T_t) = \sum_{t \in \tilde{T}_t} R(t) \ ,$$

kus $\tilde{T}_t$ on T_t lehede hulk.

Omadus 2

Iga mittelõpliku T_1 juure t puhul $R(t) > R(T_t)$.

Omaduse 2 tõestus

Kõigepealt meenutame, et

$$R(t)=\frac{1}{N}\sum_{x_n\in t}(y_n-\bar{y}(t))^2 \text{ ning } R(T)=\sum_{t\in\tilde{T}}R(t)\ .$$

Kuna t ei ole lõplik, eksisteerivad t_L ja t_R ning kehtib $R(t)\geqslant R(t_L)+R(t_R)$.

Kui t_L ja t_R ei ole lõplikud, siis kehtib

$$R(t)\geqslant R(t_{\mathrm{LR}})+R(t_{\mathrm{LL}})+R(t_{\mathrm{RR}})+R(t_{\mathrm{RL}}) \text{ jne.}$$

Kokku saame, et $R(t)\geq R(T_t)$. Vaatleme aga juhtu, kus $R(t)=R(T_t)$, siis peab kehtima ka $R(t)=R(t_L)+R(t_R)$, millest järeldub, et ei ole mingit mõtet t jagada ehk t on lõplik, mis on vastuolus eeldusega. Seega $R(t)>R(T_t)$.

m.o.t.t.

Alustades T_1 -st kärbime puud kasutades *nõrgima koha lõikamise* meetodit. Iga juure $t\in T_1$ jaoks tähistame haru T_t osa, mis koosneb ainult ühest juurest, $\{t\}$ -ga. Määrame $R_\alpha(\{t\})=R(t)+\alpha$, iga haru T_t jaoks defineerime $R_\alpha(T_t)=R(T_t)+\alpha|\tilde{T}_t|$. Nüüd niikaua kui $R_\alpha(T_t)<R_\alpha(\{t\})$ harul T_t on väiksem vea-keerukus kui puul, mis koosneb ainult juurest $\{t\}$, aga mingil α väärtusel kaks vea-keerukuse määra saavad võrdseteks. Sel hetkel osaharu $\{t\}$ on väiksem, kui haru T_t , ning seega eelistatavam. Et leida seda α väärtust, lahendame võrratuse $R_\alpha(T_t)<R_\alpha(\{t\})$ ning saame

$$\alpha<\frac{R(t)-R(T_t)}{|\tilde{T}_t|-1},\ (2)\ .$$

Omaduse 2 tõttu on võrratuse (2) parem pool positiivne.

Defineerime funktsiooni $g_1(t), t\in T_1$ järgmiselt: $g_1(t)=\begin{cases}\dfrac{R(t)-R(T_t)}{|\tilde{T}_t|-1}, t\notin\tilde{T}_1\\ +\infty, t\in\tilde{T}_1\end{cases}$

Nüüd saame defineerida *nõrgema koha* $\bar{t}_1\in T_1$ nii, et $g_1(\bar{t}_1)=min(g_1(t)\mid t\in T_1)$

ning paneme kirja $\alpha_2 = g_1(\bar{t}_1)$. Sõlm $\bar{t}_1$ on nüüd *nõrgim koht* selles mõttes, et kui α kasvab, siis $\bar{t}_1$ on esimene niisugune sõlm, et $R_\alpha(\{\bar{t}_1\}) = R_\alpha(T_{\bar{t}_1})$, siis $\{\bar{t}_1\}$ saab eelistatavamaks kui $T_{\bar{t}_1}$ ning α_2 on väärtus, mille korral võrdus kehtib. Defineerime puu $T_2 \subset T_1$, kärpides haru $T_{\bar{t}_1}$ ehk $T_2 = T_1 - T_{\bar{t}_1}$. Nüüd kasutades T_2 leiame *nõrgima koha* T_2 -s. Täpsemalt, kui T_{2t} on see osa harust T_t , mis sisaldub T_2 -s, defineerime

$$g_2(t) = \begin{cases} \dfrac{R(t) - R(T_{2t})}{|\tilde{T}_{2t}| - 1}, t \in T_2, t \notin \tilde{T}_2 \\ +\infty, t \in \tilde{T}_2 \end{cases}$$

ning $\bar{t}_2 \in T_2$ et $g_2(\bar{t}_2) = min(g_2(t) \mid t \in T_2)$, millest saame $\alpha_3 = g_2(\bar{t}_2)$. Kordame protseduuri defineerides $T_3 = T_2 - T_{\bar{t}_2}$, leides $\bar{t}_3$ ning parameetri α_4 jne. Kui mingil etapil leidub mitu *nõrgimat kohta*, näiteks $g_k(\bar{t}_k) = g_k(\dot{t}_k)$, siis defineerime $T_{k+1} = T_k - T_{\bar{t}_k} - T_{\dot{t}_k}$ ning jätkame kärpimist.

2.6 Optimaalse suurusega puu

Selliselt toimides tekivad puude jadad $T_{max} \supseteq T_1 \supset T_2 \supset \ldots \supset \{t_1\}$ ning vastav α -de jada $0 = \alpha_1 < \alpha_2 < \ldots$ nii, et $\alpha : \alpha_k < \alpha < \alpha_{k+1}$ jaoks T_k on väiksem osapuu, mis minimiseerib $R_\alpha(T)$. Et valida nüüd õige suurusega puu jadast $T_{max} \supseteq T_1 \supset T_2 \supset \ldots \supset \{t_1\}$ tuleb leida hinnangud $R(T_k)$ -dele. Selleks jagatakse meil olevad vaatlused L juhuslikut treeninghulgaks L_1 ja testhulgaks L_2 .

Olgu $d_k(x)$ puule T_k vastav prognoos. Kui $|L_2| = N_2$ defineerime

$$R^{ts}(T_K) = \frac{1}{N_2} \sum_{(x_n, y_n) \in L_2} (y_n - d_k(x_n))^2 \ .$$

Praktikas kasutatakse *V-kordset ristvalideerimist.* L jagatakse juhuslikult treeninghulkadeks

$L_{1,}\ldots, L_v$ nii, et igal treeninghulgal on võimalikult võrdne arv vaatlusi. Treeninghulk *v* näeb siis välja niimoodi: $L^{(v)}=L-L_v$. Siis iga *v* jaoks kordame puu kasvamist ja kärpimist, saades nii puud $T^{(v)}(\alpha)$, mis on minimaalse keerukuse-veaga puu parameetri α jaoks.

Kogu L kasutades saame jadad (T_k) ning (α_k) . Defineerime $\acute{\alpha}_k=\sqrt{\alpha_k \alpha_{k+1}}$ ning tähistame $d_k^{(v)}(x)$ -ga vastava puu $T^{(v)}(\acute{\alpha}_k)$ prognoosi. Rist-valideerimise hinnangud $R^{cv}(T_K)$ ning $RE^{cv}(T_K)$ on antud valemitega:

$$R^{cv}(T_K)=\frac{1}{N}\sum_{v=1}^{V}\sum_{(x_n,y_n)\in L_v}(y_n-d_k^{(v)}(x_n))^2 \ ,$$

$$RE^{cv}(T_K)=R^{cv}(T_K)/R(\bar{y}) \ .$$

Praktikas selgub, et pärast mingit piiri kujutab suuruste $R^{cv}(T_K)$ graafik ennast praktiliselt x-teljega paralleelset joont ehk suuri muutusi $R^{cv}(T_K)$ -ga ei toimu. Et siis valida õige suurusega puu rakendame $1\,SD$ – reeglit, nimelt valime piisavalt väikse T_k et

$$R^{cv}(T_k)\leqslant R^{cv}(T_{k0})+SD \ ,$$

kus $R^{cv}(T_{k0})=min(R^{cv}(T_k))$ ning SD on standardhälbe hinnang $R^{cv}(T_{k0})$ jaoks.

2.7 Rakendus R tarkvaras

Tarkvarapakettis R on CART-algoritmid realiseeritud moodulis rpart. Koodi autorid on Therry Thernau, Beth Atkinson, Brian Ripley. Töö kirjutamise hetkel oliviimane versioon 4.1-9. Moodul töötab kõikide viimaste R-versioonidega alustades versioonist 2.15.0 ning on vabalt levitatav litsentsite GPL-2 ning GPL-3 all [4].

3. CART algoritmi rakendus krediidiskooringu süsteemi loomiseks.

3.1 Andmestiku kirjeldus ja kodeerimine

Krediidiskooringu süsteemi loomisel kasutati ühte treeningandmestiklu, milles olevad andmed on suhteliselt realistlikud. Töö autori jaoks oli andmestik antud valmiskujul. Esialgne andmestik sisaldas 1800 rida ja 17 tunnust. Ühes reas puudusid kolme tunnuse väärtused, tühjad kohad imputeeriti selle tunnuse üldkeskmisega.

Andmestikus olevate tunnuste lühikirjeldus:

Tunnuse nimi vastavalt andmestikule	Tunnuse tüüp	Min	Max	Tasemete arv	Koodis kasutatav nimi
Staatus	Binaarne (1/0)			2	staatus
Sugu	Binaarne (F/M)			2	sugu
Vanus	Diskreetne arvuline	18	67		vanus
Maakond	Faktor			15	maakond
Emakeel	Binaarne (est/rus)			2	keel
Summa	Pidev arvuline	50	2000		summa
Periood	Pidev arvuline	1	720		periood
Sissetulek	Pidev arvuline	95	14000		sissetulek
Väljaminek	Pidev arvuline	4	2000		outflow
Pereseis	Faktor			5	pereseis
Haridus	Faktor			6	haridus
Töökogemus	Faktor			4	kogemus
Laste arv	Diskreetne arvuline	0	5		lapsed
Kinnisvara	Diskreetne arvuline	0	8		kinnisvara
Aktiivsed maksuhäired	Diskreetne arvuline	0	12		active
Lõpetatud maksuhäired	Diskreetne arvuline	0	22		ended
Maksuhäired kokku	Diskreetne arvuline	0	27		total

Igas reas on andmed ühe isiku kohta, tunnus *staatus* tähendab laenu tagastamist: 1 kui on, 0 kui ei ole. Eelmise peatüki valguses sisaldas tunnus *staatus* y_i väärtusi, ning vastavad x_i -d olid

vektorite $(vanus_i, sugu_i, \ldots, total_i)$ kujul. Töö autori ülesandeks oli koosatada prognoosi $d(x)$ tegev programm, kasutades käesolevaid andmeid.

Enne mudeli ehitamist kodeerisime järgmised tunnused binaarseteks:

- Maksuhäired kokku – 0, kui aktiivseid ega lõpetatud maksuhäireid ei ole; 1 kui on.
- Aktiivsed maksuhäired – 0, kui aktiivseid maksuhäireid ei ole; 1 kui on.
- Kinnisvara – 0, kui oma kinnisvara ei ole; 1 kui on.
- Haridus – 1, kui isikul on kesk- või kõrgharidus; 0 muidu.
- Töökogemus – 1, kui töökogemus on „rohkem kui aasta“; 0 muidu.
- Pereseis – 1, kui pereseis on „vallaline“ või „vabaabielus“; 0 muidu.
- Maakond – 1, kui maakond on „Harjumaa“; 0 muidu.

Kodeerimine kujunes välja proovides erinevaid kodeerimisviise ning kontrollides saadud kodeeritud tunnuseid Kruskal - Wallis testiga. Kruskal - Wallis testi eesmärgiks on välja selgitada, kas vaatlused on pärit samast jaotusest ehk võime öelda et Kruskal - Wallis test kontrollib kas mõned hulgad on samad mingi tunnuse poolest. Meie juhul kontrollisime, kas tunnuse kodeerimine toob kaasa vastavates rühmades tunnuse *staatus* erineva jaotuse. Tunnuste histogrammid enne/pärast kodeerimist ning testi R väljundid on esitatud lisades.

3.2 Regressioonipuu valikust

Nüüd, pärast andmestikuga tutvumist, tekib ilmne küsimus: millist CART meetodi kasutada - kas klassifitseerimispuud, mis kasutab parameetrilisi (binaarse funktsioonitunnusega) andmeid või regressioonipuud, mis kasutab arvulist funktsioonitunnust. Töö autori otsus oli regressioonipuu ning sellele on olemas põhjus: autor tahtis luua mitte ainult klassifitseerivat süsteemi, vaid kombineerida otsustuspuud edaspidi kirjeldatud meetodiga ning saada pideva väärtustega funktsiooni, mida oleks võimalik interpreteerida kui tõenäosust, et mingi isik maksab krediidi tagasi. Kui see funktsioon eksisteeriks, siis oleks võimalik määrata ning muuta „otsustuspiiri“ - funktsiooni väärtust, mille ületamisel isikule antakse krediiti. Niisugune „otsustuspiir“ või „riskifaktor“ annaks võimaluse muuta süsteemi tundlikkust ja spetsiifilisust sõltuvalt kasutaja soovidest, mis loodetavasti teeks kogu süsteemi paremaks tööriistaks. Töö käigus saadud süsteemil on see idee edukalt realiseeritud.

3.3 Kaugus-kaalutud keskmine

Töös eespool nimetatud „laiskade“ meetodite hulka kuuluvad ka *k-lähima naabri algoritm* ning selle modifikatsioon - kaugusega-kaalutud lähima naabri algoritm [1]. Viimast on väikese muudatusega kasutatud meie krediidiskooringu süsteemis. Olgu meil mingi treeninghulk X, $|X|=n$, milles on antud vaatlused x_i koos nende sihtfunktsiooni $f(x_i)$ väärtustega. Me tahame kasutada seda hulka uue vaatluse $\dot{x}$ sihtfunktsiooni väärtuse hindamiseks. Selleks kasutame järgmist hinnangut:

$$\hat{f}(\dot{x})=\frac{\sum_{i=1}^{n} w_i f(x_i)}{\sum_{i=1}^{n} w_i},$$

kus $w_i=\frac{1}{d(\dot{x}, x_i)}$ on mingi kaugus-funktsiooni pöördväärtus ning $x_i \in X$.

Põhimõtteliselt ei ole see hinnang midagi muud kui hulga X kaalutud keskmine, kus kaaludena on kasutatud kaugust otsitavast punktist. Selles töös rakendatakse meetodit koos regressioonipuuga: kõigepealt leitakse regressioonipuu abil uuele vaatlusele vastav leht ning siis lehe sees rakendatakse kaugus-kaalutud keskmist.

Hinnangufunktsioonil on aga üks nõrk koht – kui otsitav punkt on täpselt sama nagu mingi juba olev punkt X -st ehk $\dot{x}=x_i$ mingi i jaoks siis $w_i \to \infty$. Et seda vältida on programmi koodis hinnanguks kasutatud järgmist funktsiooni:

$$\check{f}(\dot{x})=\begin{cases} \left(\frac{\sum_{x\in X_1} w_i f(x_i)}{\sum_{x\in X_1} w_i}+\frac{\sum_{x\in X_2} f(x_i)}{|X_2|}\right)/2; & |X_2|\neq 0 \\ \hat{f}(\dot{x}); & |X_2|=0 \end{cases},$$

kus $X_1:=(x\in X | x\neq \dot{x})$
$X_2:=(x\in X | x=\dot{x})$.

Näide

Olgu meil andmestik L kujul

Tabel 1. andmestik L .

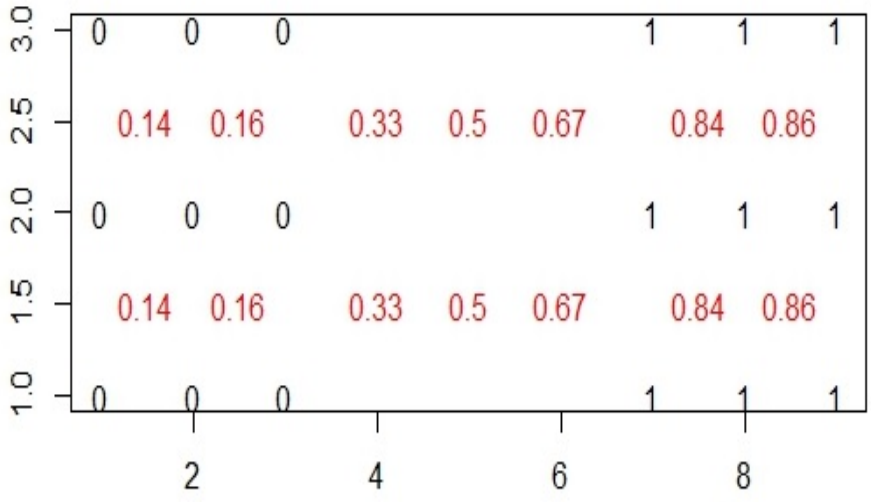
x	1	1	1	2	2	2	3	3	3	7	7	7	8	8	8	9	9	9
y	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
z	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1

kus x ja y on mingid tunnused, mida on võimalik vaadelda punktide koordinaatidena ning z on sihtfunktsiooni väärtus. Tahame, et meie prognoos andmetest L tagastaks antud andmete (x, y) jaoks prognoosi $\hat{z} = f(x, y)$. Olgu siis meil teine andmestik L_2 , mis sisaldab ainult x ja y väärtused. Leiame $\hat{z}$ iga $(x, y) \in L_2$ jaoks:

Tabel 2. Andmestik L_2 koos prognoosiga $\hat{z}$.

x	1.5	2.5	1.5	2.5	4	5	6	4	5	6	7.5	8.5	7.5	8.5
y	1.5	1.5	2.5	2.5	1.5	1.5	1.5	2.5	2.5	2.5	1.5	1.5	2.5	2.5
$\hat{z}$	0.14	0.16	0.14	0.16	0.33	0.5	0.67	0.33	0.5	0.67	0.84	0.86	0.84	0.86

Joonisele 4 on kantud L ning L_2 punktid koos vastavate z ja $\hat{z}$ väärtustega ning on võimalik näha, kuidas $\hat{z}$ sõltub „lähimate naabrite“ z väärtustest.



Joonis 4.

Kaugusfunktsiooniks on näite jaoks võetud tavaline eukleidiline kaugus

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \ .$$

3.4 Tunnuste valik puu ehitamiseks ja kaugus-kaalutud keskmise rakendamiseks

Pärast ümberkodeerimist muutus enamus tunnuseid andmestikus binaarseteks, mis on väga mugav otsustuspuu koostamiseks. Välja arvates tunnust *staatus* jääb andmestikku alles 16 tunnust, millest kolme (*sissetulek*, *outflow*, *summa*) on kasutatud kaugus-kaalutud keskmise leidmiseks, 12 otsustuspuu ehitamiseks ning ühte (*ended*) ei kasutata. Selline tunnuste jaotus on saadud katseliselt - töö autor proovis käsitsi erinevaid tunnuste kombinatsioone mõlemates rollides ning sai kõige paremad tulemused just sellise tunnuste rühmitamisega. Regressioonipuu ehitamiseks kasutatavad tunnused on seega : *vanus, sugu, keel, periood, lapsed, kinnisvara, total, active, haridus, pereseis, kogemus, maakond*; ning kaugus-kaalutud keskmise leidmiseks kasutatavad tunnused on: sissetulek, outflow, summa. Kaugus-funktsioonina on kasutatud eukleidilist kaugust

$$d(x, y) = \sqrt[3]{(x_1 - y_1)^3 + (x_2 - y_2)^3 + (x_3 - y_3)^3} \ .$$

3.5 Süsteemi ristvalideerimine

Ristvalideerimine on üks kõige rohkem kasutatav viis hinnata mingi meetodi/algoritmi/funktsiooni täpsust, kasutades ainult algandmeid. Töö käigus, kui süsteeemi kuju ei olnud veel lõplikult selge, kasutati just ristvalideerimist selleks, et otsustada, milline idee on parem. Lõpliku süsteemi puhul teostati 27 ristvalideerimist 3 erineva „otsustuspiiri“ väärtusega. Iga ristvalideerimise puhul oli andmestik jagatud kahte ossa: ühes oli 200 vaatlust, teises – 1600. Teist andmestiku poolt kasutati puu ehitamiseks ja otsuse arvutamiseks, esimese poole peal kontrolliti hinnangute täpsust. Nii korrati 9 korda iga „otsustuspiiri“ väärtuse jaoks, võttes esimesse ossa esmalt vaatlused nr 1-200, siis 201-400, siis 401-600 jne. Iga kord pandi kirja 2x2 veatabel, lõpuks keskmistati iga „otsuspiiri“ väärtuse jaoks kõik 9 tabelit. Ristvalideerimisel leiti ka iga „otsustuspiiri“ väärtuse jaoks spetsiifilisus ja tundlikkus ning valepositiivsete ja õigepositiivsete vaatluste suhe.

Olgu veatabel kujul

Tabel 3.

	Prognoositud seisund -	Prognoositud seisund +
Tegelik seisund -	a	c
Tegelik seisund +	b	d

Tundlikkus (ing. *sensitivity*) näitab, kui suure osa uuritava sündmuse toimumistest ennustab kasutatud mudel õigesti. Arvutatakse valemiga $\frac{d}{b+d}$ (tähised antuud tabelis 3).

Spetsiifilisus (ing. *specificity*) näitab, kui suure osa (kui mitu protsenti) uuritava sündmuse mittetoimumistest ennustab kasutatud mudel õigesti. Arvutatakse valemiga $\frac{a}{a+b}$.

Kadu näitab antud juhul valepositiivsete ja õigepositiivsete vaatluste suhet, ehk kui suur osa nendest, kellele krediiti anti, ei maksa seda tagasi. Arvutatakse valemiga $\frac{c}{d}$.

Edasi on toodud ristvalideerimise tulemused tunnuse *staatus* jaoks. Meenutame, et 1 tähendab edukat laenu tagasimaksmist ja 0 tähendab, et laen ei maksta tagasi.

Ristvalideerimise tulemused:

Otsustuspiir: 0.75, tundlikkus: 0.582, spetsiifilisus: 0.666, kadu: 0.202 .

Tabel 4.

	Programmi otsus: 0	Programmi otsus: 1
Reaalsus: 0	0.205	0.102
Reaalsus: 1	0.290	0.403

Otsustuspiir: 0.6, tundlikkus: 0.740, spetsiifilisus: 0.498, kadu: 0.231 .

Tabel 5.

	Programmi otsus: 0	Programmi otsus: 1
Reaalsus: 0	0.155	0.154
Reaalsus: 1	0.180	0.513

Otsustuspiir: 0.5, tundlikkus: 0.825, spetsiifilisus: 0.367, kadu: 0.254 .

Tabel 6.

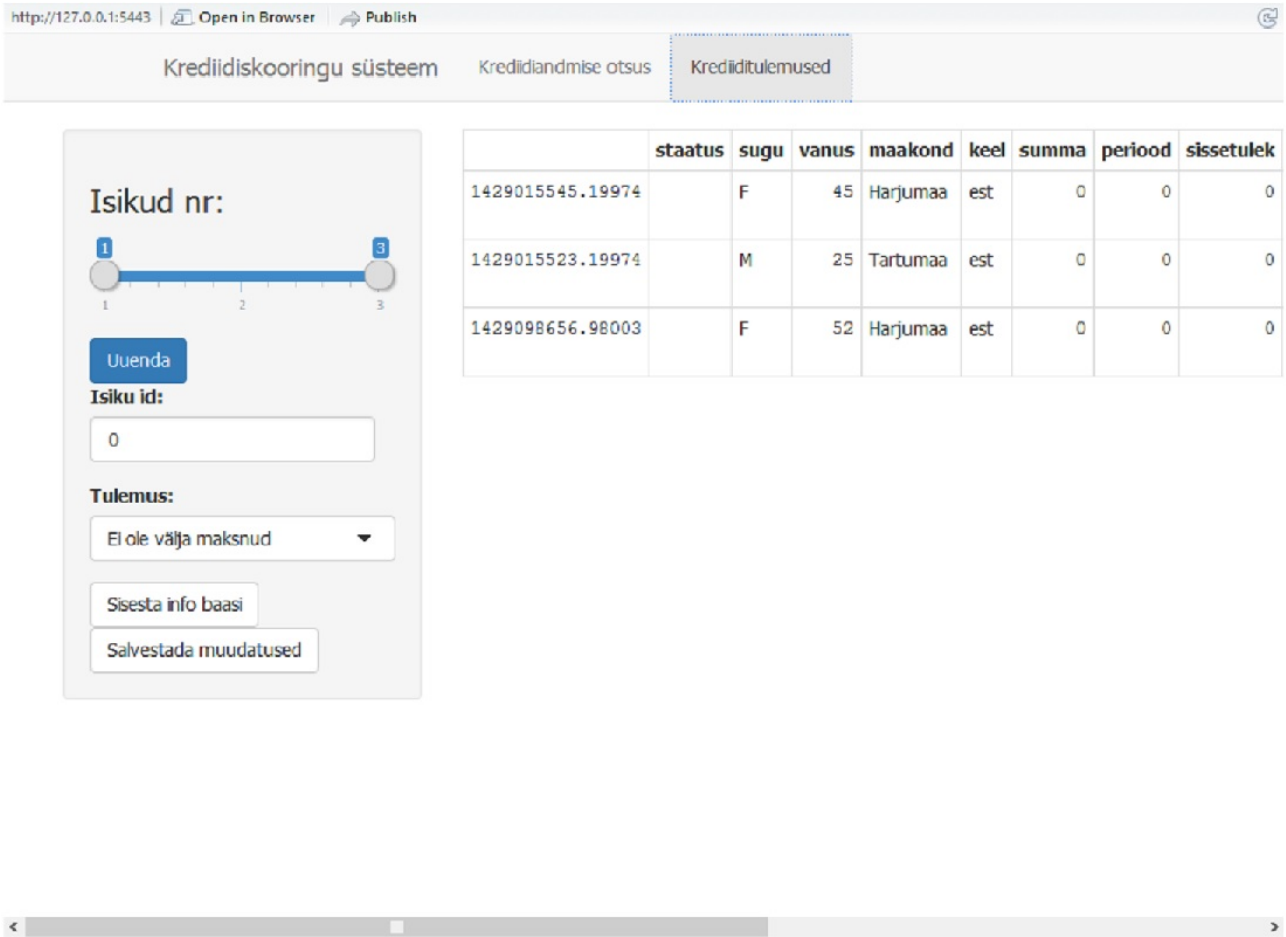
	Programmi otsus: 0	Programmi otsus: 1
Reaalsus: 0	0.112	0.196
Reaalsus: 1	0.121	0.571

Ristvalideerimise tulemused on väga hea näide selle kohta, et kui tõsta testi tundlikkust, siis langeb spetsiifilisus ning vastupidi. See töö ei anna mingeid soovitusi, milline otsuspiir on parem ning laseb kasutajal selle ise valida, lähtudes oma kogemusest/teadmistest/ettevõtte poliitikast. Autor isiklikult eelistaks vähendada kadu ehk kasutada kõrgemat otsuspiiri, mis annaks võimaluse rakendada paremaid krediiditingimusi.

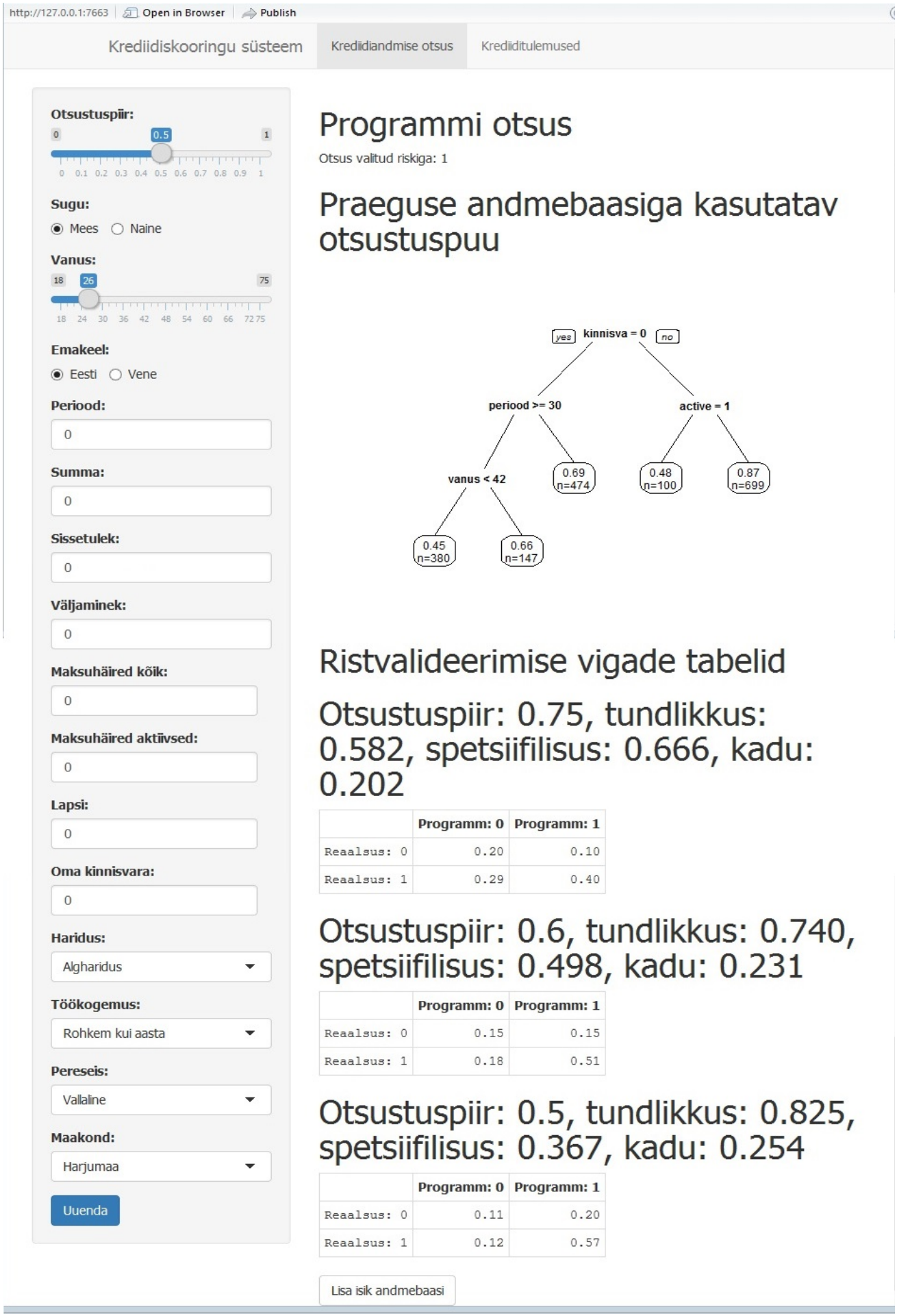
3.6 Süsteemi visuaalne kuju

Et süsteem oleks kasutajale võimalikult sõbralik, oli loodud visuaalne rakendus kasutades R moodulit *Shiny*. Osutus, et *Shiny* ei ole kõige parem moodul antud ülesandeks, mistõttu rakenduse kasutamine ei ole nii mugav nagu see võiks olla. Vaatamata sellele on süsteem täiesti kasutuskõlblik; tuleb ainult korralikult jälgida kasutusjuhendit. Rakendus koosneb 2 osast: esimeses on võimalik arvutada süsteemi otsus sõltuvalt otsuspiirist ning isiku andmetest, näha otsustuspuud ning vigade tabelit, salvestada isik, kellele anti krediiti, et tulevikus tema krediidistaatus koos tema andmetega andmebaasi lisada, mille peale ehitatakse otsustuspuu; teises on võimalik vaadata varem salvestatuid isikuid ning kirja panna nende krediidistaatus.

Rakenduse hetktõmmised (ing. *Screenshot*):



Joonis 5.


http://127.0.0.1:7663
Open in Browser
Publish
Krediidiskooringu süsteem
Krediidiandmise otsus
Krediiditulemused
Otsustuspiir:
0
0.5
1
Sugu:
Mees
Naine
Vanus:
18
26
75
Emakeel:
Eesti
Vene
Periood:
0
Summa:
0
Sissetulek:
0
Väljaminek:
0
Maksuhäired kõik:
0
Maksuhäired aktiivsed:
0
Lapsi:
0
Oma kinnisvara:
0
Haridus:
Algharidus
Töökogemus:
Rohkem kui aasta
Pereseis:
Vallaline
Maakond:
Harjumaa
Uuenda
Programmi otsus
Otsus valitud riskiga: 1
Praeguse andmebaasiga kasutatav otsustuspuu
yes
kinnisva = 0
no
periood >= 30
active = 1
vanus < 42
0.69
n=474
0.48
n=100
0.87
n=699
0.45
n=380
0.66
n=147
Ristvalideerimise vigade tabelid
Otsustuspiir: 0.75, tundlikkus: 0.582, spetsiifilisus: 0.666, kadu: 0.202
Programm: 0
Programm: 1
Reaalsus: 0
0.20
0.10
Reaalsus: 1
0.29
0.40
Otsustuspiir: 0.6, tundlikkus: 0.740, spetsiifilisus: 0.498, kadu: 0.231
Programm: 0
Programm: 1
Reaalsus: 0
0.15
0.15
Reaalsus: 1
0.18
0.51
Otsustuspiir: 0.5, tundlikkus: 0.825, spetsiifilisus: 0.367, kadu: 0.254
Programm: 0
Programm: 1
Reaalsus: 0
0.11
0.20
Reaalsus: 1
0.12
0.57
Lisa isik andmebaasi

Joonis 6

3.7 Vajalik tarkvara ning kasutusjuhend

Et käivitada rakendus kasutaja arvutis peavad olema installeeritud:

- Tarkvarapakett R
- Tarkvarapakett RStudio
- R-i moodul shiny
- R-i moodul rpart
- R-i moodul rpart.plot
- Mingi veebibrauser

Rakendus on loodud R moodulit *Shiny* kasutades ning *Shiny* omapäradest tuleb põhiline ebamugavus: vajadus kogu aeg vajutada nuppu „Uuenda“. Täpsemad juhendid iga tegevuse kohta:

Et arvutada süsteemi otsus:

1. Käivita rakendus.
2. Sisesta isiku andmed.
3. Vajuta nuppu „Uuenda“.
4. Lehe üleval poolel tekst „Otsus valitud riskiga:“ nüüd näitab programmi otsust.

Et salvestada isiku andmeid, kellele anti krediiti:

1. Arvuta selle isiku jaoks programmi otsus.
2. Vajuta nuppu „Lisa isik andmebaasi“.
3. Vajuta nuppu „Uuenda“.

Et vaadata varem salvestatud isikute andmeid ning panna kirja krediiditulemus:

1. Sule rakendus kui see oli avatud ja käivita see veelkord.
2. Ava rakenduse teine pool vajutades üleval nuppu „Krediiditulemused“.
3. Vali paremal lehekülje poolel nähtava tabeli suurus ning vajuta nuppu „Uuenda“.
4. Leia otsitav isik tabelis, vaata esimeses veerus tema unikaalset ID-d.
5. Kopeeri ID sisendile „Isiku ID“.
6. Vali tema krediiditulemus.
7. Vajuta nuppu „Sisesta info baasi“.
8. Vajuta nuppu „Uuenda“.

9. Tabel ei muutu, aga programm pani kirja vastava otsuse.
10. Kui töö on lõpetatud, vajuta nuppu „Salvesta muudatused“.
11. Vajuta nuppu „Uuenda“ ning sule rakendus.

Kokkuvõte

Käesoleva töö eesmärk oli luua tehisõppe meetoditel baseeruv süsteem isiku individuaalse krediidiskoori määramiseks ja selle põhjal krediidiandmise otsuse tegemiseks.

Töö koosnes teoreetilisest ja rakenduslikust osadest. Esimeses osas tutvusime üldiselt tehisõppega ning lähimalt CART algoritmiga. Teises osas rakendasime saadud teadmisi konkreetse probleemi lahendamiseks. Krediidiskooringu süsteemi loomine oli väga loominguline ülesanne, paljude asjade lahendus leiti katseliselt ning kuigi töö seda ei peegelda, tööprotsess oli autorile huvitav ja põnev.

Süsteem sai edukalt loodud ning süsteemi töö kontrollitud ristvalideerimisega. Töö käigus sai autor uusi teadmisi mitte ainult tehisõppest vaid ka rakendustarkvarast R ja RStudio ning parandas oma programmeerimisoskusi.

Süsteeemi tööks vajalikud failid koos esialgse treeningandmestikuga on esitatud eraldi DVD-kettal.

Seega kokkuvõttes võib öelda, et töö täitis oma eesmärgi. Suutsime luua töötava krediidiskooringu süsteemi ning tutvusime tehisõppega.

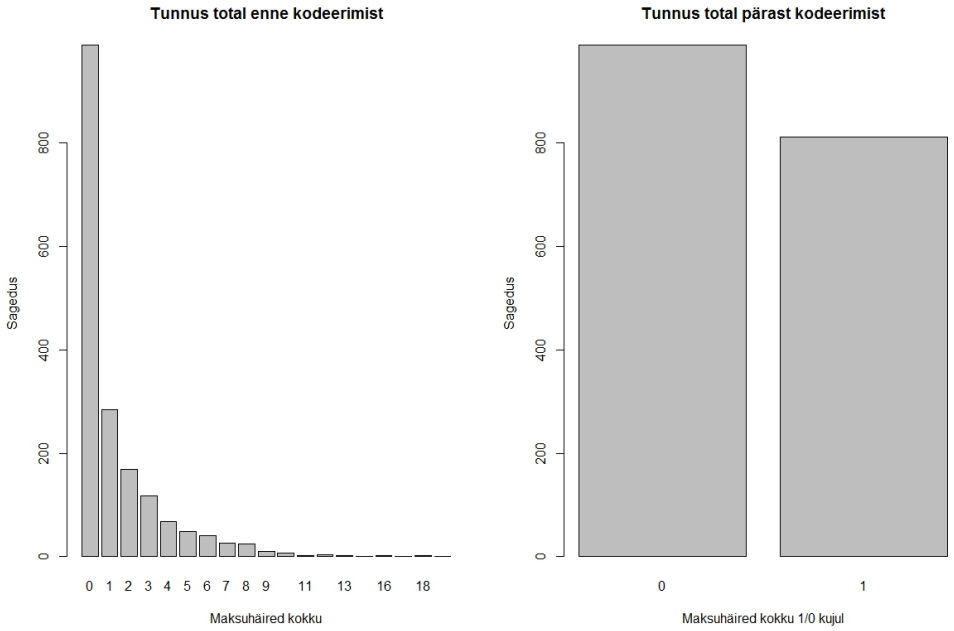
Kasutatud kirjandus

1. Ron Kohavi; Foster Provost , Glossary of Terms, 1998 (veebileheküljel) http://ai.stanford.edu/~ronnyk/glossary.html (vaadatud 27.05.2015).
2. Tom M. Mitchell, Machine Learning, 1997.
3. Leo Breiman; Jerome H. Friedman; Richard A. Olshen; Charles J. Stone, Classification and Regression Trees, 1984.
4. Terry Therneau, Beth Atkinson, Brian Ripley, rpart: Recursive Partitioning and Regression Trees, 2015 (veebileheküljel) http://cran.r-project.org/web/packages/rpart/index.html (vaadatud 27.05.2015).

Lisad

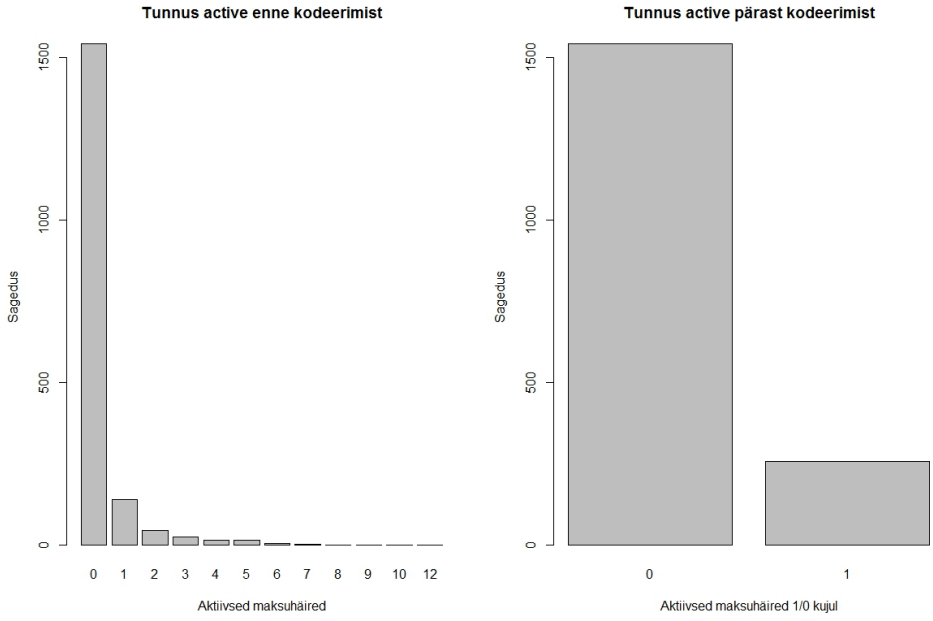
Tunnuste histogrammid enne/pärast kodeerimist ning kodeeritud tunnuse Kruskal-Wallis testi väljund.

Maksuhäired kokku:



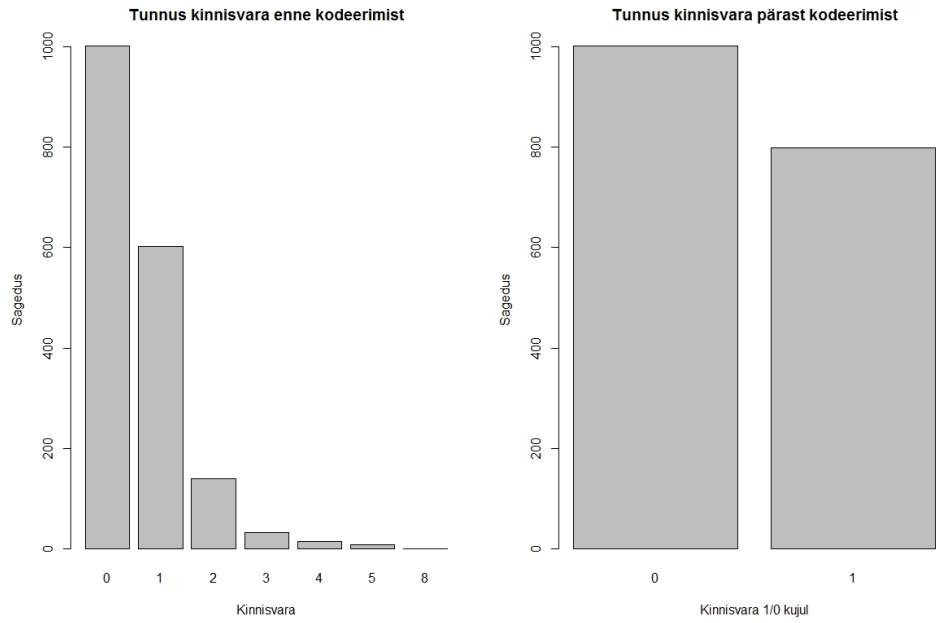

```
        data:  data$staatus by data$total1
Kruskal-Wallis chi-squared = 79.4758, df = 1, p-value < 2.2e-16
```

Maksuhäired aktiivsed:



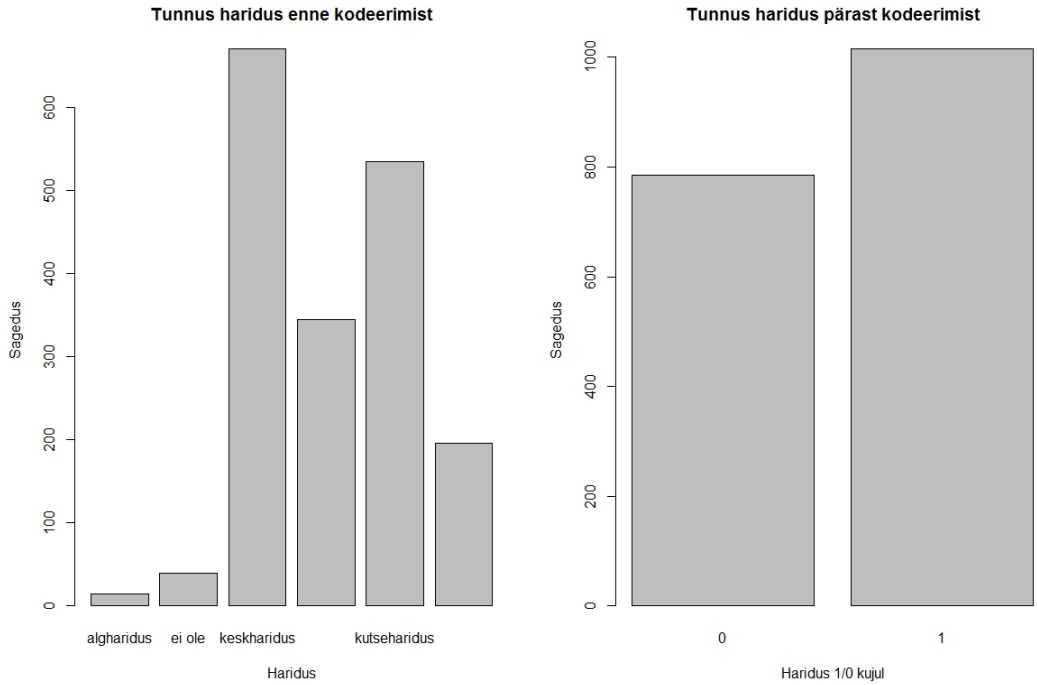

```
        data:  data$staatus by data$active1
Kruskal-Wallis chi-squared = 72.3012, df = 1, p-value < 2.2e-16
```

Kinnisvara:



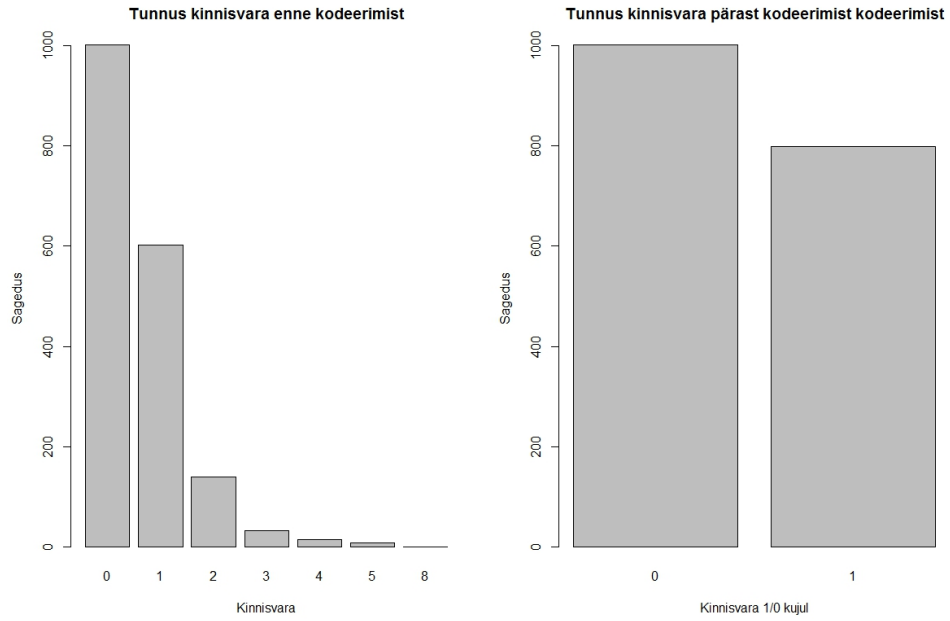

```
        data:  data$staatus by data$kinnisvara1
Kruskal-Wallis chi-squared = 104.5666, df = 1, p-value < 2.2e-16
```

Haridus:



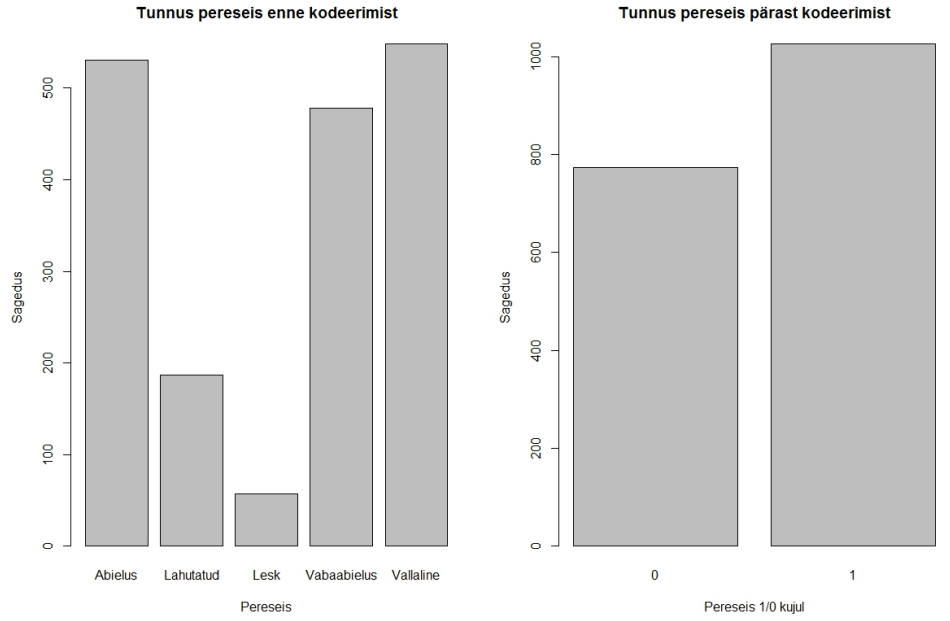

```
        data:  data$staatus by data$har
Kruskal-Wallis chi-squared = 19.4606, df = 1, p-value = 1.027e-05
```

Töökogemus:



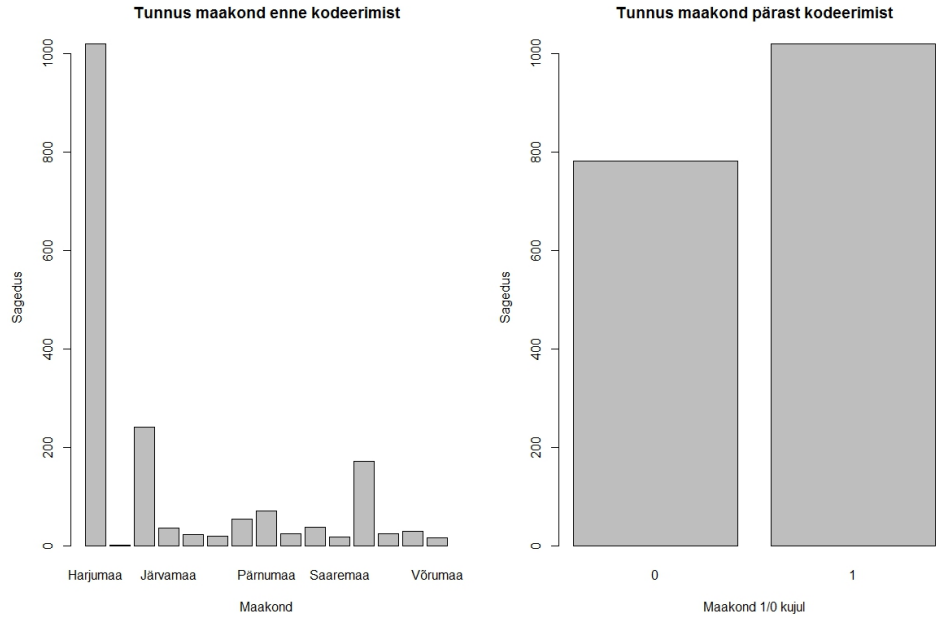

```
        data:  data$staatus by data$kog
Kruskal-Wallis chi-squared = 46.5679, df = 1, p-value = 8.85e-12
```

Pereseis:




```
        data:  data$staatus by data$per
Kruskal-Wallis chi-squared = 46.0698, df = 1, p-value = 1.141e-11
```

Maakond:




```
                data:  data$staatus by data$maak
Kruskal-Wallis chi-squared = 18.7883, df = 1, p-value = 1.461e-05
```

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina,

Valentin Skokov,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

Krediidiskooringu süsteemi loomine tehisõppe meetodite abil,

mille juhendaja on

professor Kalev Pärna,

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus 29.04.2015