LATVIJAS UNIVERSITĀTE

FIZIKAS UN MATEMĀTIKAS FAKULTĀTE

MATEMĀTIKAS NODAĻA

KRĀPNIECISKU KREDĪTŅĒMĒJU NOTEIKŠANA, IZMANTOJOT GADĪJUMA MEŽA UN GRADIENTA PASTIPRINĀŠANAS ALGORITMU

BAKALAURA DARBS

Autors: **Ieva Brantevica**

Stud. apl. Nr.: ib11256

Darba vadītājs: asoc. prof. Dr. math. Jānis Valeinis

Rīga, 2016

Anotācija

Mašīnmācīšanās nozare attīstās ļoti strauji un mūsdienās tā tiek plaši pielietota dažādās nozarēs, ne tikai zinātnē. Viena no šādām nozarēm ir nebanku kreditēšanas sektors. Darbā aplūkoti trīs mašīnmācīšanās algoritmi, kas balstīti uz klasifikācijas kokiem - lēmumu koks, gadījuma mežs un gradienta pastiprināšanas metode. Darba mērķis ir teorētiski iepazīt un praktiski noskaidrot piemērotāko no modeļiem, lai spētu izvērtēt un prognozēt krāpnieciskus kredītu pieteikumus. Darba izpildei tiek izmantota programma R un tajā iebūvētās funkcijas.

Atslēgas vārdi: krāpnieciski kredītņēmēji, mašīnmācīšanās, lēmumu koks, gadījuma mežs, gradienta pastiprināšanas mašīna

Abstract

As Machine Learning becomes more common, it is widely used in various industries. One of these industries is the non-bank lending sector. This thesis examines three machine learning algorithms that are based on classification trees - Decision tree, Random Forest and Gradient Boosting Machine. The goal is to gather the theory of these algorithms and to find the best fraud detection model in consumer financing. Data analysis and predictive modelling is done using R software.

Key words: fraud detection, machine learning, Decision tree, Random Forest, Gradient Boosting machine

Satura rādītājs

Ievads

Lielo datu (*Big Data*) laikmetā ir svarīgi apstrādāt milzīgu datu apjomu un saprast, kā tos var izmantot sev vajadzīgā virzienā. Lai šo problēmu būtu vieglāk risināt, ir izveidojusies jauna skaitļošanas nozare - mašīnmācīšanās (*Machine Learning*). Mašīnmācīšanās ir datu analīzes metode, kas automatizē analītiskā modeļa veidošanu. Izmantojot algoritmus, kas iteratīvi mācās no vēsturiskiem datiem, mašīnmācīšanās ļauj datoram atrast apslēptas datu īpašības bez skaidras ieprogrammēšanas, kur konkrēti tās meklēt. Vēl pirms pāris gadiem, kā tas tradicionāli bija pieņemts, ar statistiku nodarbojās matemātiķi-statistiķi, bet ar mašīnmācīšanos - datoriķi, savukārt mūsdienās šīs nozares ir saplūdušas un abu veidu speciālisti nodarbojas gan ar statistiku, gan mašīnmācīšanos. [9]

Vēl viena nozare, kas pēdējo gadu laikā ir ievērojami attīstījusies, ir nebanku kreditēšanas sektors. Ja cilvēkam steidzami ir nepieciešams aizdevums vai viņš to vēlas saņemt uz īsu laika periodu, tad viņam ir iespēja pieteikties aizdevumam internetā un saņemt naudu savā kontā pāris minūšu laikā. Lai aizdevēju kompānijas to spētu nodrošināt, ir jābūt ļoti strukturētai un operatīvai kredītņēmēju izvērtēšanai. Tāpēc arī šajā nozarē ienākusi mašīnmmācīšanās nozare, jo nepieciešams apstrādāt lielos datus ātri un efektīvi. Šīm kompānijām ir nepieciešamas datorprogrammas, kas spēj atpazīt maksātspējīgus klientus pēc aizdevumu pieteikumiem. Tiek meklētas īpašības, kas piemīt krāpnieciskiem klientiem, un īpašības, kas piemīt labiem klientiem, atbilstoši tām izdarot lēmumu - klientam izsniegt aizdevumu vai tomēr neizsniegt.

Bakalaura darba mērķis ir teorētiski iepazīt un praktiski pārbaudīt atsevišķu prognozējošo modeļu piemērotību krāpniecisku kredītu pieteikumu atpazīšanai.

Lai šo mērķi sasniegtu, izvirzīti vairāki uzdevumi:

- iepazīties ar mašīnmācīšanās nozari,
- iepazīties ar lēmumu koka, gadījuma meža un gradienta pastiprināšanas modeļa teorētisko pamatojumu,
- apzināt modeļu prognozes kvalitātes noteikšanas metodes,
- izstrādāt R kodu, ar kura palīdzību ir iespējams prognozēt krāpniecisku klientu atpazīšanu ar aplūkotajiem modeļiem;
- salīdzināt iegūto modeļu rezultātus un izdarīt secinājumus.

Bakalaura darba pirmajā nodaļā ir iepazīta mašīnmācīšanās nozare, tās algoritmu tipu veidi un trīs klasifikācijas modeļi, kas balstās uz koku metodes - lēmuma koks, gadījuma mežs un

gradienta pastiprināšanas mašīna. Otrajā nodaļā ir teorētiski aprakstīti modeļa izveides procesa soļi, kas nepieciešami, lai spētu izveidot prognozējošu modeli. Trešajā nodaļā tiek iepazītas divas modeļa kvalitātes novērtēšanas metodes - ROC līkne un $GINI$ koeficients, kas vēlāk tiek izmantotas, lai salīdzinātu modeļu prognozēšanas spējas.

Darba ceturtā nodaļa veltīta modeļu praktiskai izpētei. Tiek izmantoti reāli kredītu pieteikumu dati, kas ņemti no kādas konkrētas finanšu kompānijas. Šie dati nav brīvi pieejami publiskajās datubāzēs, tāpēc darbā netiek atklāti mainīgo nosaukumi - tie ir kodēti. Uz šiem daudzdimensionālajiem datiem tiek izveidots lēmuma koks un izvērtēta tā efektivitāte. Izmantojot to pašu datu kopu, tiek veidots gadījuma meža modelis un salīdzināta abu šo modeļu efektivitāte. Tiek izveidots arī gradienta pastiprināšanas modelis, kuram tiek izvērtēta precizitāte. Darba noslēgumā, izmantojot ROC līkni un $GINI$ koeficientu, izdarīti secinājumi par efektīvāko no apskatītajām metodēm.

1. Mašīnmācīšanās

1.1. Kas ir mašīnmācīšanās?

Pēc būtības, mašīnmācīšanās tiek definēta kā metožu kopums automātiskai sakarību noteikšanai datos un nākotnes vērtību prognozēšanai pēc jauniem datiem. Jaunie dati ”mācās” no iepriekš veiktajiem aprēķiniem, šādi iegūstot atkārtojamus lēmumus un rezultātus, kas balstīti uz vienādām īpašībām. [9]

Pateicoties jaunajām skaitļošanas tehnoloģijām, mašīnmācīšanās mūsdienās vairs nav tāda kā agrāk. Kaut arī daudzi mašīnmācīšanās algoritmi tiek izmantoti jau ļoti ilgu laiku, pēdējie uzlabojumi ir veikti tieši to spējā automātiski atkārtot un daudz ātrāk piemērot sarežģītus matemātiskus aprēķinus lielajiem datiem. Pāris piemēri, kuros tiek izmantota mašīnmācīšanās:

- Krāpniecisku darījumu atpazīšana;
- Meklēšanas rezultāti internetā;
- Noskaņojuma analīze, balstoties uz uzrakstīto tekstu;
- Kredītskorings un īpašie piedāvājumi klientiem;
- Iekārtu kļūdu prognozēšana;
- Attēlu atpazīšana;
- Mēstuļu filtrēšana e-pastā.

1.2. Mašīnmācīšanās algoritmu tipi

Parasti mašīnmācīsanās algoritmi tiek iedalīti trīs grupās:

1. Uzraudzītās apmācības algoritmi. Ieejas datu kopu sauc par apmācības datiem un tiem ir skaidri zināmas mērķa mainīgā vērtības. Modelis tiek būvēts caur apmācības procesu, kura laikā tas prognozē katram novērojumam mērķa mainīgo un tiek labots vietās, kad prognoze ir izteikta nepareizi. Šis apmācības process turpinās tik ilgi, līdz modelis sasniedz vēlamo apmācības precizitāti. Uzraudzītās apmācības algoritmu problēmas ir klasifikācijas un regresijas uzdevumi.

2. Neuzraudzītās apmācības algoritmi. Ieejas dati ir bez noteikta mērķa mainīgā un bez zināmiem rezultātiem. Modelis tiek veidots, nosakot datos līdzīgas struktūras ar mērķi iegūt vispārējus novērojumu nosacījumus. Šie algoritmi spēj iemācīties matemātisku procesu, lai

sistemātiski samazinātu datu pārmērību vai sakārtotu tos pēc līdzības. Parasti tie ir klāsterēšanas vai dimensiju samazināšanas uzdevumi.

3. Daļēji uzraudzītās apmācības algoritmi. Daļai no ieejas datiem ir zināmas mērķa mainīgo vērtības, bet pārējiem datiem nav zināmas. Modelim ir nepieciešams iemācīties datu struktūras, lai tās varētu sakārtot un pēc tam veikt prognozes. Arī daļēji uzraudzītās apmācības algoritmi visbiežāk strādā ar klasifikācijas un regresijas uzdevumiem. [11]

Turpmāk darbā tiks apskatīts uzraudzītās apmācības klasifikācijas uzdevums, jo datu kopai ir zināms mērķa mainīgais.

1.3. Mašīnmācīšanās algoritmi

1.3.1 Lēmuma koks

Lēmumu koki ir vieni no vispopulārākajām šobrīd izmantotajiem klasifikatoriem. Zinātnieki, kas nodarbojas tādās nozarēs kā statistika, mašīnmācība, tēlu atpazīšana (*pattern recognition*) un datizrace (*data mining*), ir plaši pētījuši, kā būvēt lēmumu kokus no pieejamajiem datiem.

Lēmumu koka mērķis ir izveidot modeli, kas prognozēs mērķa mainīgo atkarībā no dažādiem ievades datiem. Kokam ir trīs veida mezgli:

- iztekas mezgls - tam nav ienākošu zaru un ir 0 vai vairāki izejoši zari;
- iekšējie mezgli - katram ir tieši viens ienākošais zars un divi izejošie zari;
- gala mezgli jeb lapas - katram ir tieši viens ienākošais zars un neviens izejošais zars. [5]

Katru novērojumu raksturo varbūtība ar kādu tas no iztekas mezgla nonāk gala mezglā. Katrs gala mezgls pieder vienai konkrētai klasei un tas reprezentē vispiemērotāko mērķa mainīgā vērtību. Tas nozīmē, ka lapa var saturēt varbūtību vektoru, kas norāda uz varbūtību, ka mērķa mainīgais pieņems konkrētu vērtību. [3]

Modeļa formulējums

Klasifikācijas uzdevumā ievades dati ir novērojumu kopums

$$D = \{(X_1, Y_1), (X_2, Y_2), ..., (X_m, Y_m)\}. \tag{1.1}$$

Katrs ieraksts ir apzīmēts kā pāris (X_i, Y_i), kur $X_i = X_i^1, X_i^2, ..., X_i^n$, $i = 1, ..., m$ ir mainīgo kopa un Y_i, $i = 1, ..., m$ ir klasifikators (mērķa mainīgais). Mainīgie var būt gan kategoriāli, gan nepārtraukti (skaitliski).

1.1. definīcija. Par kategoriālu mainīgo $X_i^k, k = 1, ..., n$ sauksim tādu mainīgo, kas pieņem l konkrētas vērtības $\{a_1, a_2, ..., a_l\}$.

Piemēram, kad ir nepieciešams noteikt, vai klients kredītu atmaksās, kā kategoriālos mainīgos var izvēlēties - dzimums, e-pasta domēns, banka, savukārt skaitliskie mainīgie - vecums, ienākumi, bērnu skaits utt. Tomēr mērķa mainīgajam klasifikācijas uzdevumos jābūt diskrētam, tā ir galvenā pazīme, kas ļauj atpazīt klasifikācijas uzdevumu no regresijas uzdevuma.

1.2. definīcija. Klasfikācija ir mērķa funkcijas f mācīšanās uzdevums, kas katru kopas X ierakstu attēlo par vienu no definētajām Y klasēm.

Ļoti bieži mērķa funkciju neformāli sauc par klasifikācijas modeli. Klasifikācijas koks ir vienkārši saprotams - katrs iekšējais mezgls ir atkarīgs tikai no vienas mainīgā vērtības. Tas ir, ja mainīgais X_i^k, $k = 1, ..., n$ ir kategoriāls un pieņem vērtības $X_i^k = \{a_1, a_2, ...a_l\}$, tad mezgls būs atkarīgs tikai no vienas kategorijas a_r, $r = 1, ..., l$. Mezgls tālāk sadalās divos zaros - pozitīva atbilde un negatīva atbilde. Ja mainīgais X_i^k ir skaitlisks $X_i^k = [a, b]$, tad tiek atrasts punkts $c \in [a, b] \subset \mathbb{R}$, kurš skaitliskās vērtības sadala divos intervālos $[a, c]$ un $(c, b]$, katram intervālam seko savs zars.

Visi n mainīgie veido iekšējos mezglus, bet šāds koks datus apraksta pārāk smalki, tāpēc ir nepieciešams kritiski pieiet mainīgo izvēlei mezglā. Koka būvēšana notiek virzienā no augšas uz leju, tāpēc ļoti svarīga ir mainīgā izvēle katrā mezglā. Būvējot koku, ir jāatrod katrā mezglā tāds mainīgais, kas var sadalīt datus vistīrāk, t.i, konkrētais nosacījums ļoti precīzi attiecas uz kādu no klasēm. Lai to veiksmīgi varētu izdarīt, tiek izmantotas divas metrikas: entropija un informācijas pieaugums. Entropija binārā gadījumā tiek aprēķināta

$$H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}, \tag{1.2}$$

kur p - apmācības kopas pozitīvie elementi, n - apmācības kopas negatīvie elementi.

Entropijas formula vispārīgajā gadījumā tiek izteikta kā formula:

$$H(X) = -\sum_{i=1}^{s} p_i \log_2 p_i, \tag{1.3}$$

kur s - mērķa mainīgā klašu skaits, p_i - kopas X ierakstu proporcija, kas pieder i - tajai klasei.

1.3. definīcija Mezglu sauc par tīru, ja visi novērojumi šajā mezglā pieder vienai klasei. Tas nozīmē, ka šis mezgls ir arī gala mezgls.

Entropijas interpretācija:

- ja mezgls ir tīrs, t.i., $\exists i : p_i = 1$, tad $H(X) = 0$.

- ja mezglā visas klases sastāda vienādu proporciju, t.i., $p_1 = p_2 = ... = p_s = \frac{1}{s}$, tad $H(X) = 1$.

Pēc mezgla izveides tiek aprēķināta sagaidāmā entropija, kas vispārīgajā gadījumā tiek izteikta ar formulu

$$EH(X) = \sum_{i=1}^{K} \frac{p_i + n_i}{p + n} H\left(\frac{p_i}{p_i + n_i}; \frac{n_i}{p_i + n_i}\right), \tag{1.4}$$

kur $i = 1, 2, ..., K$ - no mezgla izejošo zaru daudzums,

p_i – "pozitīvo" elementu daudzums i - tajā mezglā,

n_i - "negatīvo" elementu daudzums i - tajā mezglā.[2]

Kad ir definēta entropija, varam definēt mainīgā klasificēšanas efektivitātes mēru, ko sauc par informācijas ieguvumu. Notikumu, ka X tiks šķelta, apzīmēsim ar c. Tad sagaidāmo entropiju var apzīmēt kā $EH(X) = H(X|a)$. Katram mainīgajam tiek aprēķināta entropiju starpība

$$I(X, a) = H(X) - H(X|a). \tag{1.5}$$

Izvēlas to mainīgo, kura šķelšanās rezultātā entropiju starpība ir vislielākā.

Dividimensiju gadījumā ar mainīgo, kam ir skaitliskas vērtības, datu šķelšanu var izteikt attiecībā pret punktu c, kas atrodas uz taisnes. Atliktais punkts sadala visu datu kopu divās apakškopās $X = X^L \cup X^K$. Iteratīvi šķelšanos var apzīmēt kā $X_j = X_j^L \cup X_j^K$. Tiek izvēlēta tādu mainīgo šķelšana, kuri dod vislielāko informācijas pieaugumu [2]

$$I_j = H(X_j) - \sum_{i \in \{L,K\}} \frac{|X_j^i|}{|X_j|} H(X_j^i). \tag{1.6}$$

Informācijas guvums norāda uz to, kā entropijas vērtība mainīgajam ir mainījusies pēc iekšējā mezgla sadales. To aprēķina kā starpību starp entropiju pirms mainīgā izmantošanas mezglā un entropiju, kas aprēķināta pēc mezgla izveides. Būvējot lēmuma koku, būtiskākais ir atrast tādu mainīgo, ko izmantot mezglā, lai informācijas guvums būtu vislielākais.

Lēmuma kokus pārsvarā neizmanto kā gala modeli, jo tie apmācības datus apraksta pārāk specifiski. Pārbaudot izveidoto modeli uz testa kopas, rodas liela testa kļūda. Tomēr, tos ļoti labi var pielietot ansambļu algoritmos, un viens no tādiem ir gadījuma meža algoritms.

1.3.2 Gadījuma meža algoritms

Gadījuma mežs ir ļoti populārs un efektīvs algoritms, kas tiek veidots, agregējot modeļus gan klasifikācijas, gan regresijas gadījumā. Šo algoritmu ieviesa Leo Breimanis (*Leo Breiman*). Algoritms pieder ansambļu algoritmu grupai.

Modeļa apraksts

Ir dota datu kopa $D = (X_1, Y_1), (X_2, Y_2), ..., (X_m, Y_m)$, kas sastāv no m neatkarīgiem un vienādi sadalītiem novērojumiem (X_i, Y_i). Vektors $X_i = X_i^1, ..., X_i^n$ sastāv no n prediktoriem jeb atkarīgajiem mainīgajiem, t.i, $X_i \in \mathbb{R}^n$ un $Y_i \in \mathcal{Y}$, kur $\mathcal{Y}$ ir klases mērķa mainīgais. Klasifikācijas uzdevumos, klasifikators t ir attēlojums $t : \mathbb{R}^n \to \mathcal{Y}$. Gadījuma mežu princips ir kombinēt daudzus bināros lēmumu kokus, izmantojot dažādas apmācības kopas D apakškopas, un gadījuma veidā izvēlēties katrā mezglā apakškopu no prediktoru kopas X.

Tiek veidots lēmumu koku ansamblis $h = \{h_1(X), h_2(X), ..., h_K(X)\}$, kur $h_k(X)$, $k = 1, ..., K$ ir lēmumu koks un K - koku skaits ansamblī. Katram kokam $h_k(X)$, $k = 1, ..., K$ tiek definēts klasifikatoru vektors $\theta_k = \{\theta_{k1}, ..., \theta_{kp}\}$, kur nosacījumu skaitu apzīmē ar p. Šis klasifikatora vektors sevī ietver informāciju par prediktoriem, kas atrodas mezglos, koka mezglu skaitu, mainīgo biežumu, utt. Ja koks tiek apmācīts uz k - to kopu, tad ar $h_k(X) = h(X|\theta_k)$ apzīmēsim šo koku. No apmācības kopas D tiek veidotas K apakškopas ar atkārtojumiem. Šīs kopas apzīmē ar D_θ un uz katras no izveidotajām apmācības kopām D_θ, tiek apmācīts lēmumu koks (skaitā K).[1] Parametrs θ_k nosaka gadījuma apakškopu D_{θ_k}. Veidojot apakšizlases, pazīmes tiek atlasītas nejauši, t.i, θ_k nosaka arī X_{θ_k}, kas ir $X = \{X^1, ..., X^n\}$ apakškopa, līdz ar to $(X|\theta_k)$ ir klasifikācijas koks ar reducētu dimensiju skaitu. Šādi veidots ansamblis sastāv no klasifikatoriem ar dažādām ieejas datu kopām.[2]

1.4. Definīcija Gadījuma mežs ir klasifikators, kas sastāv no kokiem $h(X, \theta_k)$, $k = 1, ..., K$, kur θ_k ir neatkarīgs, vienādi sadalīts gadījuma vektors un katrs koks dod balsi vispopulārākajai no klasēm, kas piemērota kopai X.

1.1 Teorēma Korelācija ρ starp kokiem $T_i, i = 1, ..., K$ samazinās, ja koku skaits K ansamblī palielinās. [12]

$$
\begin{aligned}
D\left(\frac{1}{K}\sum_{i=1}^{K} T_i\right) =& \frac{1}{K^2}\sum_{i=1}^{K}\sum_{j=1}^{K} Cov(T_i, T_j) \\
&= \frac{1}{K^2}\sum_{i=1}^{K}\left(\sum_{j\neq i}^{K} Cov(T_i, T_j) + D(T_i)\right)
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1}{K^2} \sum_{i=1}^{K} \left((K-1)\sigma^2\rho + \sigma^2 \right) \\
&= \frac{K(K-1)\sigma^2\rho + K\sigma^2}{K^2} \\
&= \frac{K(K-1)\sigma^2\rho}{K^2} + \frac{\sigma^2}{K} \\
&= \sigma^2\rho + \frac{\sigma^2(1-\rho)}{K}.
\end{aligned}
$$

Lai gadījuma meža modelis iegūtu prognozi, tas veido ”balsojumu” no izveidotajiem lēmumu kokiem. Nepieciešams, lai starp kokiem nebūtu stipra korelācija, tāpēc modeļa būvēšanā ieteicams izmantot lielu koku skaitu.

Lai uzbūvētais modelis prastu pēc iespējas precīzāk klasificēt ierakstus $X_i, i = 1, ..., m$, lielu ietekmi dod katrs koks $T_i, i = 1, ..., K$, jo no tā ir atkarīgs, kurai klasei $Y_i, i = 1, ..., m$ piederēs šis ieraksts. Varbūtība, ar kādu elements X piederēs klasei $Y_i, i = 1, ..., m$ tiek aprēķināta, izmantojot visu ansambļa koku rezultātus [2]:

$$
p(Y_i|X) = \frac{1}{K} \sum_{k=1}^{K} p_k \left(Y^{(i)}|X \right), (i = 1, ..., m). \tag{1.7}
$$

Gadījuma meža algoritms, veicot koku apmācību, izmanto tikai aptuveni 63% datu no apmācības kopas. Atlikušos 37% tas patur neskartus (*out-of-bag*), un, kad koks ir izveidots, pārbauda *out-of-bag* datus uz izveidoto koku, aprēķina tā kļūdu. [12] Šī metode darbojas ļoti līdzīgi kā krosvalidācija.

1.3.3 Gradienta pastiprināšanas mašīna

Šī nodaļa sarakstīta, balstoties uz publikācijas, kas izmantotajā literatūra norādīta kā [4].

Tādi modeļi kā gadījuma meži tiek veidoti kā vienkāršu modeļu apvienojums ansamblī. Pastiprināšanas (*boosting*) metodes pieeja ir ar citādāku, konstruktīvāku ansambļa veidošanas stratēģiju nekā gadījuma meži. Galvenā ideja ir izveidot vienkāršu, vāju modeli un tad pakāpeniski pievienot jaunus modeļus, izveidojot ansambli. Lai samazinātu apmācītā ansambļa kļūdu, katrā iterācijā tiek veidots jauns modelis.

Lai šo problēmu pietuvinātu statistikai, pastiprināšanas metodi noformulēja kā gradienta samazināšanās (*Gradient descent*) uzdevumu, kas atbilst optimizācijas uzdevumam. Pastiprināšanas metodes un to attiecīgie modeļi tiek saukti par Gradienta pastiprināšanas mašīnām (*Gradient Boosting Machines*). Šī pāreja piedāvā būtiskas izmaiņas modeļu hiperparametros, un nodibina metodoloģisku bāzi, lai turpmāk varētu attīstīt gradienta pastiprināšanas modeli.

Mācīšanās procedūra gradienta pastiprināšanas mašīnās (GBM) tiek veidota, secīgi veidojot jaunus modeļus, lai vēl precīzāk spētu novērtēt mērķa mainīgo. Algoritma ideja ir izveidot jaunus bāzes apmācāmos (*base-learner*) modeļus, lai tie būtu maksimāli korelēti ar zaudējuma funkcijas negatīvo gradientu, kas piemērots visam ansamblim. Zaudējuma funkcijas var būt dažādas - par kļūdas funkciju ņemot klasisko kvadrātisko kļūdu, mācīšanās procedūra rezultējas secīgā kļūdas pielāgošanā.

Modeļa apraksts

Kā jau klasifikācijas uzdevumā, dota datu kopa $D = (X, Y)_{i=1}^{m}$, kur X - prediktori un Y - mērķa mainīgais. Modeļa mērķis ir konstruēt nezināmu funkcionālo atkarību $X \xrightarrow{f} Y$ ar tādu novērtējumu $\hat{f}(X)$, ka zaudējuma funkcija $\Psi(Y, f)$ ir minimizēta:

$$\hat{f}(X) = Y,$$

$$\hat{f}(X) = \arg\min_{f(X)} \Psi(Y, f(X)). \tag{1.8}$$

Ja pārraksta modeļa novērtējuma problēmu kā prognozi, tad ekvivalents formulējums būtu: sagaidāmās zaudējuma funkcijas minimizēšana pret mērķa mainīgo $E_Y(\Psi(Y, f(X)))$ ar nosacījumu uz aprakstošajiem datiem X:

$$\hat{f}(X) = \arg\min_{f(X)} E_X \left[E_Y(\Psi(Y, f(X)))|X \right], \tag{1.9}$$

kur $E_Y(\Psi(Y, f(X)))$ ir sagaidāmais Y zaudējums, un $E_X(E_Y(\Psi(Y, f(X)))|X)$ - visu datu prognoze. Mērķa mainīgais Y var būt no dažādiem sadalījumiem. Tas automātiski noved mūs pie dažādām zaudējuma funkcijām. Ja Y ir binārs, t.i., $Y \in \{0, 1\}$, tad var izmantot binomiālo zaudējuma funkciju, savukārt, ja Y ir nepārtraukts, t.i, $Y \in \mathbb{R}$, var izmantot kvadrātisko zaudējuma funkciju. Lai funkcijas novērtēšana būtu izsekojama, var ierobežot funkcijas meklēšanas telpu uz parametrisko funkciju $f(X, \theta)$. Tad funkcijas optimizēšanas problēma kļūst par parametru novērtēšanas problēmu:

$$\hat{f}(X, \hat{\theta}) = Y,$$

$$\hat{\theta} = \arg\min_{\theta} E_X \left[E_Y \left(\Psi \left(Y, f(X, \theta) \right) \right) |X \right]. \tag{1.10}$$

Lai iegūtu novērtējumu, nepieciešamas skaitliskas, iteratīvas procedūras.

Skaitliskā optimizācija

Ja tiek veiktas M iterācijas, tad parametra novērtējumu var rakstīt formā

$$\hat{\theta} = \sum_{i=1}^{M} \hat{\theta}_i. \tag{1.11}$$

Visbiežāk izmantotā parametru novērtēšanas procedūra ir straujākā gradienta samazināšanas metode (*Steepest Gradient Descent*). Nepieciešams samazināt empīrisko zaudējuma funkciju $J(\theta)$ dotajiem $(X_i, Y_i), i = 1, ...m$ novērojumiem:

$$J(\theta) = \sum_{i=1}^{m} \Psi \left(Y_i, f(X_i, \hat{\theta}) \right). \tag{1.12}$$

Straujākās gradienta optimizācijas metodes pamatā ir secīgi uzlabojumi zaudējuma funkcijas gradienta $\nabla J(\theta)$ virzienam. Parametra novērtējums $\hat{\theta}$ tiek aprēķināts iterācijās, tāpēc tas tiks apzīmēts ar diviem dažādiem apzīmējumiem. Parametrs ar apakšindeksu θ_t tiks apzīmēts kā t - tais iterācijas solis. Ar augšējo indeksu θ^t tiks apzīmēta visa ansambļa novērtējumu samazināšanās, t.i., saskaitīti visu iterāciju parametru novērtējumi no 1 līdz t. Metode tiek izpildīta atbilstošā secībā:

1. Tiek noteikts sākuma parametrs θ_0. Atkārtot katrai iterācijai t:

2. Aprēķina kopējo parametra novērtējumu, kas aprēķināts iepriekšējās iterācijās:

$$\hat{\theta}^t = \sum_{i=0}^{t-1} \hat{\theta}_i. \tag{1.13}$$

3. Aprēķina gradientu zaudējuma funkcijai $\nabla J(\theta)$ pret ansambļa parametru novērtējumiem:

$$\nabla J(\theta) = \{\nabla J(\theta_i)\} = \left[\frac{\partial J(\theta)}{\partial J(\theta_i)} \right]_{\theta=\hat{\theta}^t}, \tag{1.14}$$

4. Aprēķina jauno parametra novērtējumu $\hat{\theta}_t$:

$$\hat{\theta}_t \leftarrow -\nabla J(\theta). \tag{1.15}$$

5. Pievieno jaunu parametra novērtējumu $\hat{\theta}_t$.

Funkciju optimizācija

Atšķirība starp pastiprināšanas metodēm un parastajām mašīnmācības tehnikām ir tā, ka optimizētas tiek funkcijas, t.i., funkciju novērtējumi tiek parametrizēti aditīvā formā:

$$\hat{f}(X) = \hat{f}^M(X) = \sum_{i=1}^{M} \hat{f}_i(X), \tag{1.16}$$

kur M ir iterāciju skaits, $\hat{f}_0$ ir sākotnējais minējums un $\{\hat{f}_i\}_{i=1}^M$ ir funkcijas pieaugums, ko sauc arī par pastiprinājumu (*"boosts"*).

Tiek ieviestas bāzes apmācāmās (*base-learner*) funkcijas $h(X, \theta)$. Lai tās atšķirtu no kopējā ansambļa funkciju novērtējumiem $\hat{f}(X)$, var izmantot dažādas bāzes funkcijas. Šajā darbā par pamatu tiek ņemti lēmumu koki.

Tagad var formulēt funkcijas palielināšanu ar bāzes apmācāmajām funkcijām. Šim nolūkam optimālais soļa lielums ϵ tiek noteikts katrā iterācijā. Lai aprēķinātu funkcijas novērtējumu t -tajā iterācijā, optimizācijas likums tiek definēts kā:

$$\hat{f}_t \leftarrow \hat{f}_{t-1} + \epsilon_t h(X, \theta), \tag{1.17}$$

$$(\epsilon_t, \theta_t) = \arg\min_{\epsilon,\theta} \sum_{i=1}^{N} \psi(Y_i, \hat{f}_{t-1}) + \epsilon h(X_i, \theta). \tag{1.18}$$

Algoritms

Tā kā gan zaudējuma funkcijai $\psi(Y, f)$, gan bāzes apmācāmajai funkcijai $h(X, \theta)$ var būt dažādi sadalījumi, var gadīties, ka parametru novērtējumus ir grūti atrast. Lai no tā izvairītos, nepieciešams izvēlēties jaunu funkciju $h(X, \theta_t)$, lai dotie dati būtu paralēli negatīvajam gradientam $\{g_t(X_i))_{i=1}^m$:

$$\{g_t(X) = E_Y \left[\frac{[\partial \psi(Y, f(X))}{\partial f(X)} \right]_{f(X)=\hat{f}^{t-1}(X)}. \tag{1.19}$$

Tā vietā, lai meklētu galveno funkcijas pastiprināšanas pieaugumu, var vienkārši izvēlēties jaunās funkcijas pieaugumu, kurš visvairāk korelē ar $-g_t(X)$. Tas palīdz potenciāli grūtu optimizācijas uzdevumu aizstāt ar mazāko kvadrātu metodi:

$$(\epsilon_t, \theta_t) = \arg\min_{(\epsilon,\theta)} \sum_{i=1}^{N} \left[-g_t(X_i) + \epsilon h(X_i, \theta) \right]^2. \tag{1.20}$$

Tagad var formulēt pilno gradienta pastiprināšanas algoritmu, kā to definēja Frīdmans (*Freidman*) 2001. gadā. Viss algoritms ir atkarīgs no bāzes apmācāmās funkcijas $h(X, \theta)$ un zaudējuma funkcijas $\psi(Y, f)$.

Bāzes apmācāmā funkcija un zaudējuma funkcija

Gadījumā, kad mērķa mainīgais ir kategoriāls, tas pieņem bināras vērtības $Y \in \{0, 1\}$ un pieņem, ka tas nāk no Bernulli sadalījuma. Lai atvieglotu notāciju, mērķa mainīgais tiek transformēts uz $\bar{Y} = 2Y - 1$, un transformētais atkarīgais mainīgais pieņem vērtības $\bar{Y} \in \{-1, 1\}$. Šajā gadījumā klases varbūtību var novērtēt, minimizējot negatīvo ticamības funkciju, kas apzīmēta ar jaunajiem klases apzīmējumiem:

$$\psi(Y, f)_{Bern} = \log(1 + \exp(-2\bar{Y}f)). \tag{1.21}$$

Zaudējuma funkcija $\psi(Y, f)_{Bern}$ tiek saukta par Bernulli zaudējuma funkciju.

Vēl viena kategoriālā mērķa mainīgā zaudējumu funkcija, kas bieži tiek izmantota, ir eksponenciālā funkcija, kas tiek izmantota *Adaboost* algoritmā

$$\psi(Y, f)_{Ada} = \exp(-\bar{Y}f). \tag{1.22}$$

Lēmumu koku modeļi tiek izmantoti, lai starp GBM modeļiem ātri aprēķinātu sakarības. Lai arī dažas sakarības starp atkarīgajiem mainīgajiem varētu zaudēt aditīvo modeļu interpretācijas īpašību, to tomēr nevar uzskatīt par nozīmīgu trūkumu, jo GBM interpretācijai tiek izmantoti arī citi uz kokiem bāzēti rīki. Lēmumu koku ideja ir veidot ieejas mainīgo sadali homogēnos taisnstūru laukos ar lēmumu koku veiktu nosacījumu sistēmu. Katrs koka mezgls veido *if - then* nosacījumu kādam konkrētam ieejas mainīgajam. Lēmumu koku struktūra dabiski kodē un veido sakarības starp prediktoriem. Kokiem parasti tiek noteikts parametrs, kurš apzīmē mezglu skaitu vai sakarību dziļumu (*interaction depth*). Vienam mainīgajam ir iespēja atkārtoties lēmuma kokā vairākas reizes.

2. Mašīnmācīšanās modeļa izveides process

Lai izveidotu modeli no zināmiem datiem un prognozētu mērķa mainīgo, balstoties uz jauniem datiem, ir nepieciešams saprast, kādi darbi jāveic, lai prognoze būtu sekmīga. Vissvarīgākais posms modelēšanā ir datu sagatavošana.

1. Definē projektu - jāsaprot, kāds ir prognozes mērķis, kāpēc modelis tiek būvēts, ko prognoze parādīs, pie kādiem secinājumiem analītiķis grib nonākt. Jāsaprot, kādi dati būs vajadzīgi;

2. Datu iegūšana - iegūst datus no dažādiem avotiem, nodrošina pilnīgu priekšstatu par prediktoriem, kas varētu ietekmēt prognozi;

3. Datu apstrāde un analīze - jāizpēta iegūtie dati, jāiztīra, jāpārveido, varbūt pie kādiem secinājumiem var nonākt jau pirms modeļa izveides. Jāizspriež, kuri prediktori ietekmēs mērķa mainīgo, ja nepieciešams, dati jābalansē, trūkstošās vērtības jāaizvieto vai novērojumi ar trūkstošajām vērtībām jāatmet. Vienmēr jāpārbauda arī prediktoru korelācija savā starpā.

4. Statistiskā analīze - ļauj apstiprināt vai noraidīt pieņēmumus, hipotēzes, un pārbauda ar statistikas modeļiem.

5. Modelēšana - prognozes modeļa veidošana dod iespēju automātiski izveidot precīzus modeļus par nākotni, balstoties uz zināmiem datiem.

6. Modeļa pielietošana - ļauj iegūtos analītiskos rezultātus izmantot ikdienas lēmumu pieņemšanā. [10]

3. ROC līkne un $GINI$ koeficients

Modeļa kvalitātes novērtēšanā var izmantot ROC līkni (*Receiver-Operating Characteristic curves*) un $GINI$ koeficientu.

ROC līkne ir divdimensionāls grafiks, kas vizuāli attēlo klasifikācijas modeļa sniegumu. Tā ļauj novērtēt, cik labi konkrētais modelis spēj atšķirt objektus, kuriem piemīt kāda īpašība, no objektiem, kuriem šī īpašība nepiemīt.[8] Konkrētajā gadījumā, ROC līkne palīdz novērtēt, vai modelis spēj atšķirt uz krāpnieciskiem nolūkiem tendētus klientus no godīgiem.

Tiek pieņemts, ka eksistē modelis, kurš prot atpazīt krāpniecisku klientu no laba klienta. Modeļa rezultāta vērtība tiek apzīmēta ar S. Tiek uzskatīts, ka klients ir krāpniecisks, ja vērtība S ir mazāka par slieksni c, savukārt klients tiek uzskatīts par labu, ja S ir lielāka par slieksni c. Lai konstruētu ROC līkni, nepieciešams definēt divas jaunas metrikas. Ar TPF (*true positive fraction*– patiesi pozitīva daļa) tiks apzīmēta tā klientu daļa, kuriem modeļa rezultāts ir pozitīvs, t.i. klients ir krāpniecisks. Ar FPF (*false positive fraction*– nepatiesi pozitīva daļa) tiks apzīmēta tā klientu daļa, kuriem modeļa rezultāts bija pozitīvs, neskatoties uz to, ka klients ir labs. [8]

$$TPF = \frac{TP}{TP + FN}, \tag{3.1}$$

$$FPT = \frac{FP}{TN + FP}, \tag{3.2}$$

kur TP - pareizi noteikti krāpnieciskie klienti, TN - pareizi noteikti labie klienti, FP - klienti, kuri bija slikti, prognozēti kā labi klienti, un FN - labie klienti klasificēti kā krāpnieciski. Prognozētās vērtības tiek sastādītas klasificēto novērojumu matricā (Tabula 3.1). No šīs matricas var

Tabula 3.1
Klasificēto novērojumu matrica

		Prognozētā vērtība	
		Patiesa	Nepatiesa
Esošā vērtība	Patiesa	Patiesi pozitīvs (TP)	Nepatiesi pozitīvs (FP)
	Nepatiesa	Nepatiesi negatīvs (FN)	Patiesi negatīvs (TN)

viegli aprēķināt modeļa precizitāti (*Accuracy*), kas ir izteikta kā:

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}. \tag{3.3}$$

3.1. definīcija. Ar D apzīmēsim klasificēšanas pazīmes indikatorfunkciju, t.i., $D = 1$, ja klients ir labs, un $D = 0$, ja klients ir krāpniecisks.

Tātad $TPF = P[S > c|D = 0]$, bet $FPF = P[S > c|D = 1]$, kur S ir iespējamā modeļa novērtējuma varbūtība, $c \in (0, 1)$. [7]

Var noteikt modeļa novērtējuma varbūtību S populācijām ar klasificēšanas pazīmes indikatorfunkcijām $D = 0$ un $D = 1$, šādā veidā tiek iegūtas divas sadalījuma funkcijas. Ja sliekšņa c vērtības tiek mainītas no 0 līdz 1, tad katram c iegūst $TPF(c)$ un $FPF(c)$. Ja $TPF(c)$ tiek attēlota uz y ass un $FPF(c)$ tiek attēlota uz x ass, iegūst ROC līkni. [7]

3.2. definīcija. $ROC = \{(TPF(c), FPF(c)), c \in (0, 1)\}$.

No ROC līknes definīcijas seko, ka līkne vienmēr atradīsies apgabalā $[0, 1]\times[0, 1]$, jo gan $TPF(c)$, gan $FPF(c)$ ir varbūtības, kuras pēc definīcijas var pieņemt vērtības no 0 līdz 1. [6]

Laukums, kas atrodas zem ROC līknes tiek saukts par AUC (*area under curve*). Tas ir visplašāk lietotais ROC līknes raksturojošais lielums un tiek definēts kā:

$$AUC = TPF(FPF)dFPF. \tag{3.4}$$

AUC var pieņemt vērtību no intervāla $[0.5, 1]$. Piemēram, salīdzinot divus modeļus, tas modelis, kuram AUC lielāks, tiek uzskatīts par kvalitatīvāku modeli. [7] Salīdzinoši nesen tikai ieviests jauns statistikas mērs $GINI$ koeficients (indekss)

$$GINI = 2AUC - 1, \tag{3.5}$$

kas nosaka sakarības starp sadalījumiem. Ar šo mēru var salīdzināt dažādu prediktīvo modeļu kvalitāti. Krāpniecisku kredītņēmēju novērtēšanas kontekstā, modelis ar lielāku $GINI$ koeficentu labāk spēj noteikt atšķirību starp labiem un krāpnieciskiem klientiem.

4. Praktiskais pielietojums

Šajā nodaļā tiks praktiski pārbaudīts, kā strādā iepriekšminētie koku algoritmi. Tiks izveidota datorprogramma vidē R, kas, izmantojot dažādas iebūvētās funkcijas, izveidos lēmuma koka modeļus, gadījuma meža modeli un gradienta pastiprināšanas modeli. Tiks izmantoti dati no kādas finanšu kompānijas, kas vēlējās, lai patiesie mainīgo nosaukumi netiktu izmantoti. Modeļa mērķis būs prognozēt, vai klients ir ar krāpnieciskiem nolūkiem, t.i, aizņēmums netiks atdots, vai tomēr klients plāno atdot aizņemto summu. Kad modeļi būs izveidoti, tad to prognozes spējas tiks savā starpā salīdzinātas ar ROC līknēm un $GINI$ koeficientiem.

Izmantotie dati nav brīvi pieejami publiskajās datubāzēs, tie ņemti no konkrētas finanšu kompānijas un ir konfidenciāli. Datu kopa sākotnēji sastāv no 16400 apstiprinātiem aizņēmumiem un 47 kolonnām- to skaitā ir 1 atkarīgais mainīgais, un 46 prediktori. Prediktoru nosaukumi tiks šifrēti ar apzīmējumiem $x_1, x_2, ..., x_{46}$. Tie ietver tādas pieteikuma pazīmes kā klienta vecums, profesija, e-pasta garums, ienākumi, kredītvēsture utt.

4.1. Mērķa mainīgā definīcija

Ļoti svarīgs solis prognozēšanas modeļa izveidē ir korekti definēt mērķa mainīgo. Lai prognozētu varbūtību, ka klients ir ar krāpnieciskiem nolūkiem, ir nepieciešams precīzi nodefinēt krāpniecisku un labu klientu. Atkarīgā mainīgā definīcija tiks balstīta uz klientu maksājumu kavēto dienu skaitu un tā būs apzīmēta kā $target$ ar vērtībām „GOOD" un „BAD".

4.1. definīcija. Par labiem klientiem ar $target$ vērtību „GOOD" sauksim tādus klientus, kuriem maksājumu kavēto dienu skaits ir mazāks nekā 60.

4.2. definīcija. Par krāpnieciskiem klientiem ar $target$ vērtību „BAD" sauksim tādus klientus, kuriem 60 dienu laikā nav veikts neviens maksājums.

Tiklīdz ir zināmas krāpniecisko un labo klientu definīcijas, un ir pieejamas šo klientu pazīmes, ir iespējams izveidot dažādus modeļus, noteikt to kvalitāti un salīdzināt savā starpā.

4.2. Datu analīze

Tā kā koku algoritms neprot interpretēt un izveidot modeli, kurā mainīgajiem ir trūkstošas vērtības, noskaidroju, ka nav tādu mainīgo, kam būtu vairāk kā 10% tukšu ierakstu. Trūkstošās vērtības kategoriāliem mainīgajiem ir aizvietotas ar jaunu kategoriju "MISSSING". Programma R pieņem tikai tādus kategoriālos mainīgos, kuros nav vairāk kā 52 kategoriju. Šādi mainīgie tika sagrupēti pēc to biežuma: ja "BAD" proporcija mainīgajā ir mazāka par 0,05 un "GOOD"

proporcija ir mazāka par 0,01, tad tiek piešķirta vērtība ”Others”, savukārt kategorijas uz kurām šis nosacījums neattiecas, ir biežāk sastopamas, tāpēc tās tiek atstātas kā iepriekš. Datu kopa ir pietiekami liela, tāpēc ieraksti, kuriem skaitlisko mainīgo vērtības ir trūkstošas, tika izslēgti no datu kopas, izmantojot R bibliotēkas *stats* funkciju *complete.cases()*. Rezultātā, modelēšanas datu kopa sastāv no 12531 ierakstiem, 13 skaitliskiem prediktoriem, un 33 kategoriāliem mainīgajiem. Dati ir par 11 mēnešu periodu, tie tiek sadalīti apmācibas kopā un testa kopā. Modelis tiks būvēts uz 8 mēnešu periodu, un pārbaudīts uz atlikušo 3 mēnešu pieteikumiem. Tabulā 4.1 attēlotas visu skaitlisko prediktoru aprakstošās statistikas.

Tabula 4.1
Skaitlisko mainīgo aprakstošās statistikas

Mainīgais	Min.	Mediāna	Vid.vērtība	Max.
x_1	0	3	2,84	6
x_2	21	39	40,46	70
x_6	700	2500	3022	600000
x_7	0	0,47	342,51	45658
x_10	0	1	0,97	5
x_12	0	0	0,32	15
x_14	1	10	10,74	30
x_25	-30	0	-0,01	30
x_27	0	0	0,20	6
x_28	0	0	0,29	6
x_29	1	1	1,17	72
x_37	0	734	758,10	1456
x_38	0	72	50,83	480

Aplūkojot aprakstošās statistikas, redzams, ka mainīgie x_6 un x_7 satur izlecējus, jo to maksimālā vērtība krietni atšķiras no mediānas un vidējās vērtības. Paturēsim šos mainīgos prātā, un iekļausim tos modelēšanā. Ja šie mainīgie nebūs nozīmīgi, tad modelis pats tiem dos mazu svaru. 1. pielikumā ir pievienota skaitlisko mainīgo Pīrsona korelāciju matrica. Prediktori savā starpā korelē ļoti vāji, tāpēc modeļa būvēšanā var iekļaut visus skaitliskos mainīgos.

Tabulā 4.2 norādīts skaits, cik kategoriju ir katrā kategoriālajā mainīgajā.

Apmācības kopā, kas apzīmēta ar $train$ ir 721 ”BAD” klientu un 7749 ”GOOD”, savukārt testa kopā $test$ ir attiecīgi 453 ”BAD” un 3608 ”GOOD” klientu. Skaidri redzams, ka mērķa mainīgā klases ir neproporcionālas un tas ievērojami sarežģī modeļa izveidi. Šādā gadījumā nepieciešams klases sabalansēt. Teorijā piedāvā trīs klases balansēšanas iespējas:

1. Biežāk sastopamās klases novērojumu samazināšanu līdz retāk sastopamās klases skaitam;

2. Retāk sastopamās klases novērojumu pavairošanu līdz biežāk sastopamās klases skaitam;

Tabula 4.2
Klašu skaits datu kategoriālajos mainīgajos

Mainīgais	Kategoriju skaits	Mainīgais	Kategoriju skaits
target	2	x_24	6
x_3	2	x_26	2
x_4	2	x_30	3
x_5	7	x_31	6
x_8	2	x_32	2
x_9	7	x_33	2
x_11	2	x_34	2
x_13	3	x_35	2
x_15	2	x_36	2
x_16	2	x_39	2
x_17	2	x_40	2
x_18	2	x_41	3
x_19	2	x_42	2
x_20	2	x_43	4
x_21	11	x_44	3
x_22	2	x_45	6
x_23	8	x_46	4

3. Retāk sastopamās klases novērojumu pavairošanu līdz konkrētam skaitam un biežāk sastopamās klases novērojumu skaita samazināšanu līdz tam pašam skaitam.

Literatūras avotā [13] minēts, ka, ja tiek izmantota 1.pieeja, tad apmācības kopa pazaudē ļoti svarīgu novērojumu informāciju, un modeļa prognozes precizitāte ievērojami samazinās, kad modelis tiek testēts uz jaunu novērojumu kopu. Līdzīgi notiek ar 3. balansēšanas pieeju, tāpēc turpmāk apmācības kopai izmantoju ”BAD” klases pavairošanu no 721 līdz 7749 ierakstiem. Lai arī informācija par sliktajiem klientiem atkārtojas, netiek pazaudētas labo klientu īpašības. R bibliotēka *caret* piedāvā funkciju *upSample()*, kas izpilda 2. balansēšanas pieeju, kā rezultātā $train$ ieraksti tiek palielināti līdz 15498 novērojumiem, kur ”BAD” un ”GOOD” proporcija ir 7749 pret 7749.

4.3. Modelēšana

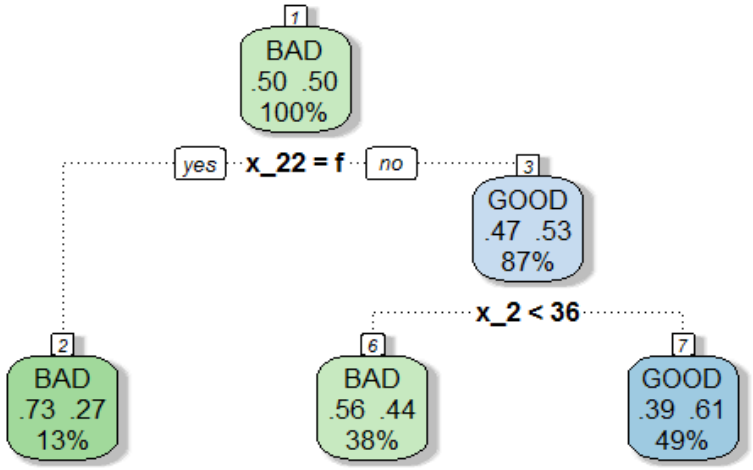
Izmantojot programmas R iebūvētās funkcijas, jābūt ļoti uzmanīgam ar modeļa pārpielāgošanu (*overfitting*). Par pārpielāgošanu sauc modeļa pārāk lielu pielāgošanos apmācības kopai - šis modelis attēlo ne tikai sakarības starp prediktoriem un atkarīgo mainīgo, bet arī kļūdas. Tas nozīmē, ka modeļa precizitāte uz apmācības kopu ir tuva 100%, bet veicot prognozi uz jauniem datiem, precizitāte ir krietni mazāka. Tāpēc ir nepieciešams atrast parametrus, kas neveic pārpielāgošanu un izveido modeli, kas norāda uz sakarībām starp mērķa mainīgo un prediktoriem. R bibliotēka *caret* ir izveidota tieši šādam mērķim, tāpēc to izmantoju savā darbā. Bibliotēkas

caret galvenais autors ir Makss Kūns (*Max Kuhn*). Šī bibliotēka tika izveidota, lai 147 dažādiem prognozēšanas modeļiem būtu vienāda sintakse un to vieglāk būtu pielietot programmā R, kā arī iegūt visoptimālākos modeļa parametrus. Par labāko prognozes modeli tiks izvēlēts tas modelis, kuram AUC un $GINI$ koeficients testa kopai un apmācības kopai būs vistuvākais, tādā veidā izvairīsimies no apmācības kopas pārgludināšanas.

4.3.1 Lēmuma koka modelis

Gadījuma meža un gradienta pastiprināšanas algoritmu pamatā ir kombinēti lēmumu koku ansambļi. Lai arī minēts, ka viens lēmuma koks ir salīdzinoši vājš klasifikators, veiksim lēmuma koka modeļa izveidi. Tam tiks izmantota programmas R bibiliotēka *rpart*. Lai izveidotu modeli, tiek izmantota funkcija *rpart()*. Funkcija pēc noklusējuma izmanto $cp = 0,01$ un izveido modeli ar vismazāko kļūdu. Parametru cp sauc par sarežģītības parametru (*complexity parameter*). Koka būvēšanā tiek izmantoti tikai tie mezgli, kas kopējo kļūdu samazina par cp reizēm. Parametra galvenā loma ir paātrināt datora ātrdarbību, nogriežot mezglus, kas nedod pietiekamu guvumu, un neļaut modelim pārpielāgot apmācības datus.

```
rpart_koks<-rpart(target~.,data=train)
rpart_koks\$cptable
      CP       nsplit  rel error    xerror     xstd
1 0.11885405      0    1.0000000 1.0215512 0.008030846
2 0.09510905      1    0.8811460 0.8811460 0.007975774
3 0.01000000      2    0.7860369 0.7860369 0.007846688
```



Att. 4.1: **Funkcijas *rpart()* izveidotā lēmuma koka attēls ar parametru** $cp = 0,01$

Gala modelis tiek izveidots ar $cp = 0,01$ un 2 mezgliem. Attēlā 4.1 var aplūkot lēmuma koku,

kas tika izveidots. To var interpretēt šādi:

1. mezglā dati vēl nav sašķelti, BAD un GOOD proporcija ir 50:50.

2. mezglā redzam, ka 13% pieteikumu no apmācāmās kopas ir ar mainīgā x_{22} vērtību f, no kuriem 73% ir BAD, bet 27% ir GOOD.

3. mezglā nonākuši ieraksti, kam x_{22} vērtība nav f, tie ir 87% ieraksti no visas apmācības kopas. No šiem 87% ierakstiem, 53% klasificējās kā labi.

4.mezglā klasificējās 38% no visas apmācāmo datu kopas, un tie ir ieraksti, kam $x_{22} \neq f$ un $x_2 < 36$. Pieteikums 56% šādu gadījumu tiek klasificēts kā krāpniecisks.

5.mezglā nonāk tādi novērojumi, kam $x_{22} \neq f$ un $x_2 > 36$. Tādi ir 49% no visas $train$ kopas, un šajā gadījumā varbūtība, ka klients būs labs, ir 0,61.

Tālāk tiek aplūkota klasifikācijas novērojumu matrica un kļūda katrai klasei gan apmācības datiem, gan testa datiem.

Tabula 4.3
Izveidotā *rpart()* lēmuma koka klasifikācijas novērojumu matricas un kļūdas

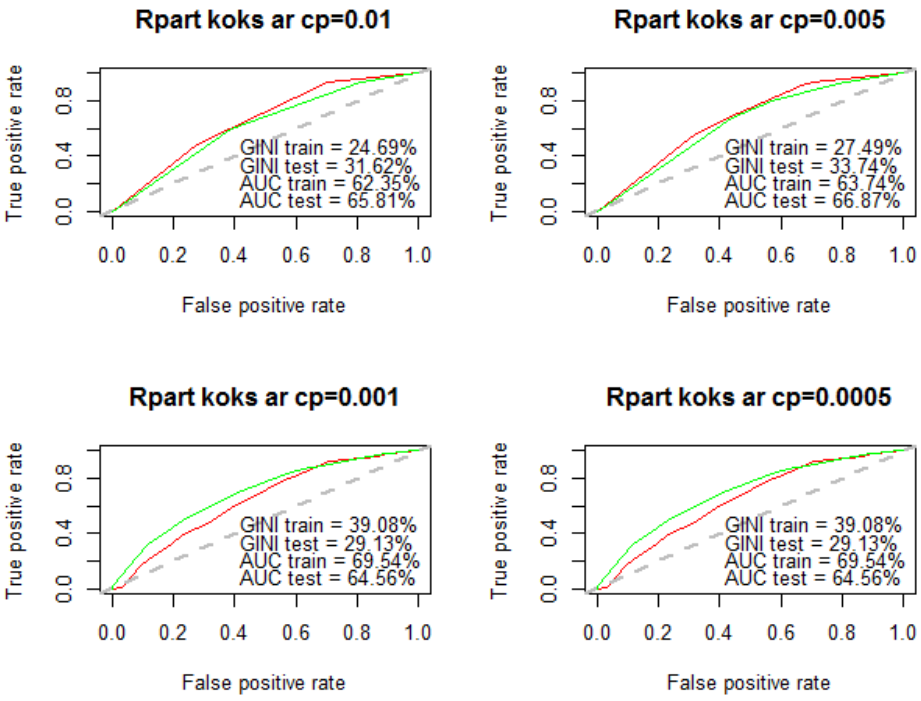
	Apmācības kopa				**Testa kopa**		
	BAD	GOOD	Klasif.kļ.		BAD	GOOD	Klasif.kļ.
BAD	4755	2994	0.3863	BAD	327	126	0.2781
GOOD	3097	4652	0.3996	GOOD	1848	1760	0.5121

Tabulā 4.3 redzam, ka modelis ar līdzīgu klasifikācijas kļūdu prognozē gan labos, gan krāpnieciskos klientus apmācības kopā - aptuveni 40% no visiem klientiem klasificēti nepareizi. Precizitāte šajā gadījumā ir $\frac{4755+4652}{15498} \approx 60,7\%$. Testa kopā atkarīgā mainīgā vērtību proporcijā ir krasa atšķirība, tāpēc modelis vairs nespēj pareizi klasificēt labos klientus. Redzam, ka klasifikācijas kļūda krāpniecisku klientu atpazīšanā ir daudz zemāka kā labo klientu atpazīšanā - $27,8\%$ un $51,2\%$. Modeļa precizitāte uz testa kopas ir $\frac{327+1760}{15498} \approx 51,4\%$. Redzams, ka pie $cp = 0,01$, izveidotais modelis prognozēšanā izmanto tikai divus mainīgos. Aplūkosim *rpart()* koku modeļus ar $cp = \{0,01; 0,005; 0,001; 0,0005\}$.

Attēlā 4.2 pēc ROC līknēm var secināt, ka pie cp vērtībām, kas mazākas par $0,005$, modelis pārgludina apmācības kopu. Savukārt iepriekš izveidotā modeļa ($cp = 0,01$) $GINI$ koeficienti norāda, ka tas labāk apraksta testa kopu nekā apmācības kopu. Tā tam nevajadzētu būt, tas nozīmē, ka modelis nav labs.

Modelēšanas mērķis ir izveidot tādu modeli, kurš uz jaunas datu kopas spēs klasificēt jaunos datus ar pēc iespējas lielāku precizitāti. Tas nozīmē, ka abu kopu $GINI$ un AUC vērtībām jābūt pēc iespējas tuvākām.

Vēl viens nozīmīgs lēmuma koka parametrs ir $maxdepth$ - lēmuma koka maksimālais mez-



Att. 4.2: **Funkcijas *rpart* izveidotā koka ROC līknes apmācības (zaļā krāsā) un testa kopai (sarkanā krāsā) pie** $cp = \{0,01; 0,005; 0,001; 0,0005\}$

glu skaits. Pie parametra $cp = 0.005$ izveidosim kokus ar dažādu maksimālo mezglu skaitu un salīdzināsim $GINI$ un AUC gan apmācības, gan testa kopai. Tabulā 4.4 redzami iegūtie rezultāti.

Tabula 4.4

***rpart()* koka $GINI$ un AUC vērtības atkarībā no $maxdepth$ lieluma, kad $cp = 0,005$**

maxdepth	gini_train	gini_test	AUC_train	AUC_test
1	24,69%	31,62%	62,35%	65,81%
2	24,69%	31,62%	62,35%	65,81%
3	27,07%	33,52%	63,53%	66,76%
4	27,49%	33,74%	63,74%	66,87%
5	33,65%	33,46%	66,82%	66,73%
6	35,10%	33,12%	67,55%	66,56%
7	36,36%	33,22%	68,18%	66,61%
8	37,47%	34,09%	68,73%	67,04%
9	37,47%	34,09%	68,73%	67,04%

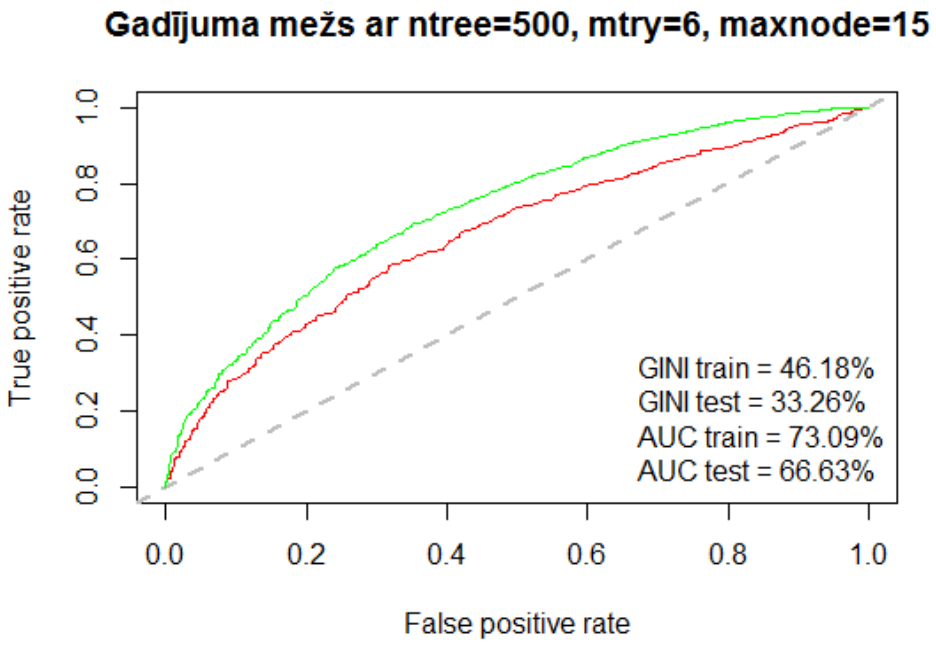
Secinām, ka labākais lēmuma koka modelis ir ar 5 mezgliem, un tas prot precīzi klasificēt aptuveni 67 klientus no 100.

4.3.2 Gadījuma meža modelis

Lai būvētu gadījuma meža modeli, tiks izmantotas divas bibliotēkas: *caret* un gadījuma meža bibliotēka *randomForest* [14]. Iebūvētajā funkcijā *randomForest()* ir svarīgi 3 parametri:

- $maxnode$ - norāda, cik dziļš būs katrs lēmuma koks ansamblī,
- $mtry$ - norāda, cik prediktori tiks gadījuma veidā izmantoti katra lēmuma koka mezgla veidošanā. Ieteikts sākt ar noklusējuma vērtību $mtry = \sqrt{n}$, kas konkrētajā gadījumā būtu $mtry = \sqrt{46} \approx 7$.
- $ntree$ - norāda, cik lēmuma koki tiks būvēti, lai izveidotu gadījuma mežu.

Parametram $ntree$ nevajadzētu būt pārāk mazam, tam vajadzētu nodrošināt, ka katram pieteikumam tiek prognozēta $target$ vērtība vismaz pāris reizes. Tomēr, jo lielāka $ntree$ vērtība, jo lēnāk algoritms strādā. Tā kā apmācāmo datu kopa ir ļoti liela, tad, lai uzlabotu modeļa ātrdarbību un kompilācijas laiku, tiks pārbaudītas trīs $ntree$ vērtības: $ntree = \{100, 300, 500\}$. Manuāli tiek sastādīts cikls, kurš pie dotajām $ntree$ vērtībām veido gadījuma mežu ar $mtry = \{1, 2, ..., 15\}$ un $maxnode = \{2, 3, ..., 15\}$. 2.pielikumā pievienoti iegūto modeļu *out-of-bag* kļūdu novērtējumi. Lai arī vismazākā kļūda ir modelim ar ieejas parametriem $mtry = 6$, $ntree = 500$ un $maxnode = 15$, attēlā 4.3 redzamās ROC līknes skaidri norāda, ka apmācības kopa ir pārgludināta.



Att. 4.3: **Gadījuma meža ROC līknes apmācības kopai (zaļā krāsā) un testa kopai (sarkanā krāsā), kad parametru vērtības ir $mtry = 6$, $ntree = 500$ un $maxnode = 15$**

Lai to nepieļautu, jāpanāk, ka lēmuma koki, kas veido ansambli, nav pārāk dziļi. Tas nozīmē, ka nepieciešams atrast atbilstošu $maxnode$ vērtību. Atkal sastādīts cikls, kurā tiek veidoti gadījuma meži ar parametra $maxnode$ vērtībām $\{2, 3, ..., 15\}$. Tabulā 4.5 attēlotas šo modeļu kvalitātes mēri.

Tabula 4.5
Gadījuma meža $GINI$ un AUC vērtības atkarībā no $maxnode$ parametra, kad $ntree = 500$ un $mtry = 5$

maxnode	gini_train	gini_test	AUC_train	AUC_test
2	34,8%	30,0%	67,4%	65,0%
3	36,3%	30,9%	68,2%	65,5%
4	37,7%	31,1%	68,9%	65,5%
5	38,3%	31,2%	69,1%	65,6%
6	39,6%	32,3%	69,8%	66,1%
7	40,7%	31,4%	70,4%	65,7%
8	41,8%	34,3%	70,9%	67,2%
9	42,3%	32,9%	71,1%	66,4%
10	42,6%	33,0%	71,3%	66,5%
11	44,0%	33,2%	72,0%	66,6%
12	44,4%	33,3%	72,2%	66,6%
13	44,8%	34,2%	72,4%	67,1%
14	45,7%	34,2%	72,9%	67,1%
15	46,6%	34,4%	73,3%	67,2%

Redzams, ka vislabākais gadījuma meža modelis krāpniecisku kredītņēmēju noteikšanā, ir ar parametriem $mtry = 5$, $ntree = 500$ un $maxnode = 2$.

Tabula 4.6
Izveidotā gadījuma meža klasifikācijas novērojumu matricas un kļūdas

	Apmācības kopa				**Testa kopa**		
	BAD	GOOD	Klasif.kļ.		BAD	GOOD	Klasif.kļ.
BAD	4298	3451	0.4452	BAD	289	164	0.3620
GOOD	2178	5571	0.2818	GOOD	1469	2139	0.4072

Apskatot klasifikācijas novērojumu matricas, klasifikācijas kļūdas un modeļa precizitāti uz abām kopām, redzam, ka testa kopas labie pieteikumi tiek klasificēti ar lielāku kļūdu nekā apmācības kopā, savukārt krāpnieciskie klienti tiek klasificēti ar lielāku precizitāti testa kopā. Interesanti, ka izveidotais gadījuma meža algoritms dod mazliet sliktāku AUC kā lēmuma koks.

4.3.3 Gradienta pastiprināšanas modelis

Gradienta pastiprināšanas modelis tiks modelēts, izmantojot bibliotēku *caret*. Šī bibliotēka piedāvā izmantot funkciju *train()* un kā metodi izvēlēties ”gbm”. Funkcija dod iespēju pielāgot 4 parametrus:

- $n.trees$ - pastiprināšanas iterāciju skaits,
- $interaction.depth$ - koka maksimālais dziļums,
- $shrinkage$ - parametrs, kas samazina katra bāzes apmācāmā ietekmi. Teorijā ieteikts labāk ņemt mazāku $shrinkage$ vērtību nekā lielu. [4]
- $n.minobsinnode$ - minimālais gala mezglu skaits.

Lai modeļa parametrus varētu labāk noteikt un tas netiktu trenēts tikai uz vienas, unikālas apmācības kopas, bet gan uz vairākām, tiks izmantota 3 slāņu krosvalidācija. Tas nozīmē, ka modelis tiks trenēts 3 reizes, un katrā no šīm reizēm būs ņemta citādāka apmācības kopas apakškopa. Apskata 3 slāņu krosvalidācijas modeli ar noklusējuma parametriem $shrinkage = 0.1$, $n.minobsinnode = 10$.

Tabula 4.7
***train()* funkcijas izveidoto modeļu precizitāte pie $shrinkage = 0.1$, $n.minobsinnode = 10$, un dažādām $interaction.depth$, $n.trees$ vērtībām**

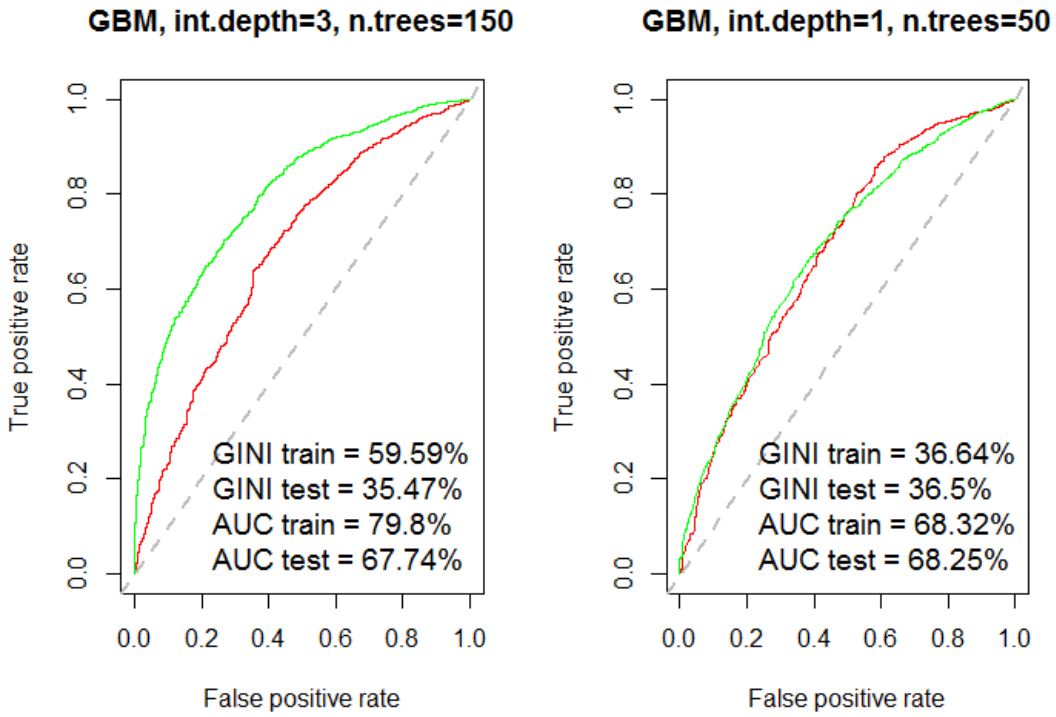
interaction.depth	n.trees	Precizitāte
1	50	63,2%
1	100	64,3%
1	150	65,2%
2	50	64,7%
2	100	66,3%
2	150	67,8%
3	50	66,0%
3	100	68,8%
3	150	70,0%

Redzams, ka noklusējuma parameteri $shrinkage = 0.01$, $n.minobsinnode = 10$ pie $interaction.depth = 3$, $n.trees = 150$ jau dod labu modeli ar precizitāti 70,18%. Aplūkosim modeļa klasifikācijas matricas, klašu kļūdas un precizitāti $train$ un $test$ datiem.

Tabula 4.8
Izveidotā *rpart()* lēmuma koka klasifikācijas novērojumu matricas un kļūdas

	Apmācības kopa				**Testa kopa**		
	BAD	GOOD	Klasif.kļ.		BAD	GOOD	Klasif.kļ.
BAD	5460	2289	0.2954	BAD	296	157	0.3466
GOOD	2149	5600	0.2773	GOOD	1489	2119	0.4127

Skatoties klasificēto novērojumu matricā, apmācības kopas dati pārāk labi tiek klasificēti, ja to salīdzina ar testa kopas prognozēm, tā pat tas ir pamanāms attēlā 4.4, skatoties uz ROC līknēm.



Att. 4.4: **Funkcijas *train()* GBM modeļu** ROC **līknes ar dažādiem parametriem, zaļā līkne atbilst apmācības kopai, sarkanā līkne- testa kopai**

Tālāk tiek definēti parametri, kurus būs nepieciešams pielāgot. Tiek iestatīti 3 dažādi koku dziļumi $interaction.depth = \{2, 4, 6\}$, iterāciju skaits $n.trees = \{25, 50, 75, ..., 225\}$, $shrinkage$ tiek samazināts uz 0,05 savukārt mezglu skaits netiek mainīts, tas ir, $n.minobsize = 10$. Modeļa precizitāti var apskatī 3.pielikumā. Kā gala modelis tiek izvēlēts GBM ar parametriem $shrinkage = 0.01$, $n.minobsinnode = 10$, $interaction.depth = 1$, $n.trees = 50$, jo AUC apmācības datiem un AUC testa datiem ir ļoti tuvas, tas nozīmē, ka modelis nepārgludina apmācības kopu, toties testa kopas prognozes arī ir precīzākas.

Secinājumi

Pētot šos trīs mašīnmācīšanās koka algoritmus un ar tiem strādājot R vidē, tika atklātas modeļu labās un sliktās īpašības. Kā jau tika paredzēts, lēmumu koks ir viegli interpretējams un saprotams. Teorijā minēts, ka viens lēmumu koks ir pārāk vājš klasifikators. Tas vai nu spēj uzbūvēt modeli, kurš ir pārāk pielāgots apmācības kopai, vai izveido pārāk triviālu modeli. Izveidotais lēmuma koka modelis ar sarežģītības parametru $cp = 0,005$ pārsteidzoši labi prognozē krāpnieciskos pieteikumus. Apmācības kopas AUC vērtība 66,82% norāda, ka aptuveni 67 gadījumos no 100, lēmuma koks spēj atpazīt krāpnieciskus pieteikumus no labiem pieteikumiem, un ar tādu pašu precizitāti izveidotais lēmuma koks spēs pareizi klasificēt arī kredīta pieteikumus testa kopā. Arī $GINI$ koeficienti gandrīz sakrīt, tas nozīmē ka izveidotais modelis ir labs.

Gadījuma meža algoritms spēja klasificēt labos klientus apmācības kopā ar 28% kļūdu, tomēr uz jauniem datiem klasifikācijas kļūda pieauga par 12%. Gradienta pastiprināšanas modelis šajā gadījumā izrādījās labāks. Tas uzrāda par vairāk nekā 3% augstāku $GINI$ koeficientu gan testa, gan apmācības kopai, salīdzinot ar lēmuma koku, un pat par 6% augstāku $GINI$, salīdzinot ar gadījuma meža testa kopu. Lai arī šajā gadījumā gradienta pastiprināšanas modelis izrādījās labāks, tas nenozīmē, ka uz citu datu kopu tas arī viennozīmīgi būs labāks.

Modeļa būvēšanas laikā jebkuram no šiem trim modeļiem ir nepieciešams nodefinēt koka dziļumu, citādāk tas vienmēr tiek pārgludināts. Ansambļu modeļi ir ļoti grūti interpretējami. Ja teorētiski tos var formulēt ar iterācijām, tad ar R iebūvētajām funkcijām ir praktiski neiespējami interpretēt modeļa uzbūvi.

Darba izstrādes laikā secināju, ka materiāli par mašīnmācīšanos ir ļoti daudz, tomēr tikai retos gadījumos tie ir ar matemātisku pamatu. Tā pat reti definētas ieteicamās modeļu parametru vērtības, tās ir jāmeklē katrai datu kopai un modelim individuāli.

Mašīnmācīšanās algoritmi strādā lēni, kad datu kopa ir apjomīga. Darbā izmantotie dati ir sarežģīti, jo *out-of-bag* kopas elementi ne pārāk labi apraksta visu apmācības kopu un nesniedz precīzus rezultātus. Tas ir iemesls, kāpēc lēmumu koks uzrādīja tik negaidīti labus rezultātus, jo, ar salīdzinoši vienkāršu modeli bija iespējams aprakstīt krāpniecisko klientu īpašības tik pat labi kā ar koku ansambļa modeli. Tomēr jāsaka, ka kreditēšanas biznesā nevajadzētu izmantot šos modeļus, jo to klasifikācija nav pietiekami precīza.

Literatūras saraksts

[1] Breiman L., *Random Forests*, Statistics Department, University of California, Berkeley, 2001.

[2] Kabaļina E., *Netipisku datu un krāpniecisku transakciju noteikšana, izmantojot gadījuma meža algoritmu: Bakalaura darbs,* Rīga: LU FMF Matemātikas nodaļa, 2015. 16-17 lpp

[3] Maimon O., Rokach L., *Data Mining and Knowledge Discovery Handbook*, Springer, 2005.

[4] Natekin A., Knoll A., *Gradient boostong machines, a tutorial.* Pieejams: https://www.researchgate.net/publication/259653472_Gradient_boosting_machines_a_tutorial [skatīts 25.05.2016]

[5] Pang-Ning Tan, Michael Steinbach, Vipin Kumar *Introduction to Data Mining*, Pearson, 2005. Page 150.

[6] Rezač, M., *How to measure the quality of credit scoring models. Journal of Economics and Finance No 5.*, 2011, Pages 487-507.

[7] Tetereva A., *Nepārtraukto testu ROC līknes: Kursa darbs.* Rīga: LU FMF Matemātikas nodaļa, 2009.

[8] Wang J., *Encyclopedia of Data Warehousing and Mining*, IGI Global, 2009. Pages 1316-1324.

[9] Wassermann L.,*All of statistics: A Concise Course in Statistical Inference*, Springer, 2004.

[10] Tiešsaiste: http://www.predictiveanalyticstoday.com/what-is-predictive-analytics/ [Skatīts 06.09.2016]

[11] Tiešsaiste: http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/ [Skatīts 05.04.2016.]

[12] Tiešsaiste: https://stat.ethz.ch/education/semesters/ss2012/ams/slides/v10.2.pdf [Skatīts 07.05.2016]

[13] Tiešsaste: https://www3.nd.edu/ dial/publications/chawla2005data.pdf [Skatīts 07.04.2016]

[14] Tiešsaiste: https://cran.r-project.org/web/packages/randomForest/randomForest.pdf [Skatīts 04.02.2016]

[15] Tiešsaiste: http://topepo.github.io/caret/index.html [Skatīts 25.05.2016]

Pielikumi

1. pielikums. Pīrsona korelācijas matrica skaitliskajiem mainīgajiem.

Tabula 1

Pīrsona korelāciju matrica skaitliskajiem mainīgajiem

	x_1	x_2	x_6	x_7	x_10	x_12	x_14	x_25
x_1	1	-0,007	0,001	-0,006	0,008	0,005	-0,005	0,005
x_2	-0,007	1	0,029	-0,026	-0,264	-0,047	-0,057	-0,008
x_6	0,001	0,029	1	-0,026	-0,059	-0,024	-0,001	0,003
x_7	-0,006	-0,026	-0,026	1	-0,008	-0,009	0	0,01
x_10	0,008	-0,264	-0,059	-0,008	1	0,031	0,016	-0,004
x_12	0,005	-0,047	-0,024	-0,009	0,031	1	0,017	0,026
x_14	-0,005	-0,057	-0,001	0	0,016	0,017	1	-0,003
x_25	0,005	-0,008	0,003	0,01	-0,004	0,026	-0,003	1
x_27	0,001	-0,076	-0,017	-0,016	0,038	0,129	0	0,006
x_28	-0,001	-0,098	-0,021	-0,014	0,049	0,119	-0,006	0,006
x_29	-0,002	-0,004	-0,003	-0,006	0,007	0,017	0,008	-0,006
x_37	-0,012	0,105	0,033	-0,005	-0,034	-0,007	-0,033	-0,02
x_38	-0,02	0,054	-0,001	-0,003	-0,018	0,013	-0,021	-0,001

Tabula 2

Pīrsona korelāciju matrica skaitliskajiem mainīgajiem (turpinājums)

	x_27	x_28	x_29	x_37	x_38
x_1	0,001	-0,001	-0,002	-0,012	-0,02
x_2	-0,076	-0,098	-0,004	0,105	0,054
x_6	-0,017	-0,021	-0,003	0,033	-0,001
x_7	-0,016	-0,014	-0,006	-0,005	-0,003
x_10	0,038	0,049	0,007	-0,034	-0,018
x_12	0,129	0,119	0,017	-0,007	0,013
x_14	0	-0,006	0,008	-0,033	-0,021
x_25	0,006	0,006	-0,006	-0,02	-0,004
x_27	1	0,155	0,009	-0,005	0,013
x_28	0,155	1	0,003	-0,012	0,015
x_29	0,009	0,003	1	-0,061	-0,009
x_37	-0,005	-0,012	-0,061	1	0,101
x_38	0,013	0,015	-0,009	0,101	1

2. pielikums. Gadījuma meža algoritma *out-of-bag* kļūda, atkarībā no dažādiem $ntree$, $maxnode, mtry$ vērtībām.

Tabula 3

Gadījuma meža *out-of-box* kļūda, atkarībā no dažādām $mtry$ un $maxnode$ vērtībām, kad $ntree = 100$

	mtry							
maxnode	1	2	3	4	5	6	7	8
2	44,9%	42,0%	42,9%	43,1%	44,0%	41,1%	41,3%	41,5%
3	42,5%	42,1%	39,4%	41,2%	39,9%	41,5%	40,8%	39,9%
4	41,2%	40,3%	39,3%	38,1%	38,0%	38,5%	38,6%	37,9%
5	40,5%	39,8%	37,9%	38,7%	37,4%	37,8%	37,8%	38,4%
6	42,0%	39,4%	37,7%	37,7%	38,5%	37,8%	37,5%	37,3%
7	40,6%	39,1%	38,0%	36,9%	37,4%	36,5%	37,1%	36,5%
8	40,4%	38,9%	37,7%	36,5%	36,6%	36,6%	36,5%	36,3%
9	39,2%	38,1%	37,2%	37,0%	36,5%	37,0%	36,5%	36,3%
10	38,7%	38,0%	36,6%	36,5%	36,2%	35,6%	35,4%	36,6%
11	38,9%	38,0%	37,0%	36,2%	35,8%	35,7%	36,0%	36,2%
12	38,7%	37,7%	36,6%	36,5%	35,9%	34,9%	35,1%	35,6%
13	38,7%	36,6%	36,4%	35,7%	35,5%	35,7%	35,6%	35,2%
14	39,3%	36,6%	35,7%	35,9%	35,2%	35,6%	35,2%	34,7%
15	40,0%	37,5%	35,4%	35,5%	35,6%	34,4%	34,6%	35,1%

Tabula 4

Gadījuma meža *out-of-box* kļūda, atkarībā no dažādām $mtry$ un $maxnode$ vērtībām, kad $ntree = 100$ (turpinājums)

	mtry						
maxnode	9	10	11	12	13	14	15
2	42,5%	42,1%	41,2%	41,9%	41,4%	41,4%	42,0%
3	39,9%	40,7%	40,6%	40,9%	39,3%	41,1%	41,0%
4	38,3%	38,2%	38,5%	37,9%	37,8%	38,1%	37,9%
5	37,8%	37,2%	37,3%	38,3%	38,0%	37,9%	37,9%
6	37,2%	37,3%	37,6%	36,8%	37,5%	37,7%	37,9%
7	37,0%	36,8%	37,1%	37,1%	37,3%	37,1%	37,9%
8	35,9%	36,8%	37,0%	36,2%	36,4%	37,0%	37,4%
9	36,3%	36,4%	37,2%	36,8%	36,6%	36,5%	36,5%
10	36,2%	36,7%	36,8%	36,6%	36,4%	37,1%	37,2%
11	35,8%	36,1%	36,1%	36,0%	36,4%	36,2%	36,7%
12	35,9%	35,6%	35,5%	36,4%	36,7%	35,8%	36,5%
13	35,4%	35,2%	36,0%	36,3%	36,1%	36,9%	36,0%
14	35,3%	35,6%	35,8%	35,5%	36,2%	35,9%	35,9%
15	35,0%	35,1%	35,3%	35,0%	35,5%	36,1%	35,5%

Tabula 5

Gadījuma meža *out-of-box* kļūda, atkarībā no dažādām $mtry$ un $maxnode$ vērtībām, kad $ntree = 300$

	mtry							
maxnode	1	2	3	4	5	6	7	8
2	43,7%	42,4%	42,7%	42,3%	43,0%	40,6%	41,1%	40,9%
3	42,3%	40,8%	40,3%	39,5%	39,5%	40,2%	40,0%	39,7%
4	41,5%	39,4%	38,3%	37,9%	37,6%	38,0%	37,6%	37,4%
5	40,4%	39,1%	37,4%	37,7%	37,1%	37,6%	37,4%	37,5%
6	41,6%	39,1%	37,4%	37,0%	37,6%	37,1%	37,2%	36,8%
7	40,2%	38,5%	36,9%	36,7%	36,6%	35,9%	36,8%	36,9%
8	39,3%	38,1%	37,0%	36,1%	36,4%	36,1%	36,1%	36,7%
9	39,1%	37,8%	37,2%	36,2%	36,1%	36,0%	36,3%	36,1%
10	38,8%	36,7%	36,9%	36,0%	36,0%	35,6%	35,4%	36,0%
11	38,4%	37,3%	36,3%	35,6%	35,1%	35,4%	35,4%	35,5%
12	38,8%	36,9%	36,2%	35,4%	35,1%	34,9%	35,0%	35,3%
13	38,1%	36,4%	36,0%	34,9%	35,1%	35,1%	34,7%	35,2%
14	39,2%	35,4%	35,3%	35,2%	34,7%	34,8%	34,8%	34,7%
15	39,2%	36,2%	34,9%	35,0%	34,9%	34,2%	34,4%	34,7%

Tabula 6

Gadījuma meža *out-of-box* kļūda, atkarībā no dažādām $mtry$ un $maxnode$ vērtībām, kad $ntree = 300$ (turpinājums)

	mtry						
maxnode	9	10	11	12	13	14	15
2	41,9%	42,0%	41,7%	42,1%	42,1%	42,2%	42,2%
3	39,1%	39,6%	40,7%	39,8%	39,3%	39,9%	40,2%
4	38,0%	37,8%	37,9%	37,8%	37,7%	37,9%	37,8%
5	37,7%	37,1%	37,5%	37,7%	37,6%	37,5%	37,6%
6	37,1%	37,1%	37,6%	36,9%	37,2%	37,2%	37,2%
7	36,5%	36,7%	36,4%	36,6%	37,2%	37,0%	37,4%
8	36,2%	36,4%	36,6%	36,4%	37,1%	36,9%	36,8%
9	35,9%	36,1%	36,3%	36,6%	36,5%	36,7%	36,6%
10	36,2%	36,3%	36,2%	36,1%	36,4%	36,1%	37,0%
11	35,4%	35,6%	35,7%	36,4%	36,5%	36,2%	36,5%
12	35,5%	35,2%	35,0%	35,6%	36,2%	36,1%	36,3%
13	34,9%	35,2%	35,6%	35,6%	35,8%	36,3%	36,0%
14	34,9%	35,2%	35,6%	35,6%	35,9%	35,6%	36,1%
15	35,1%	34,9%	35,2%	35,0%	35,2%	35,5%	35,5%

Tabula 7

Gadījuma meža *out-of-box* kļūda, atkarībā no dažādām $mtry$ un $maxnode$ vērtībām, kad $ntree = 500$

	mtry							
maxnode	1	2	3	4	5	6	7	8
2	44,1%	42,7%	42,8%	42,5%	43,1%	42,0%	41,6%	41,2%
3	42,1%	40,3%	40,1%	39,8%	39,4%	40,2%	40,6%	39,8%
4	40,7%	39,5%	38,6%	37,7%	37,0%	37,9%	38,1%	37,3%
5	40,6%	39,4%	37,5%	37,5%	37,5%	37,3%	37,4%	37,2%
6	40,9%	39,3%	36,9%	37,2%	37,6%	37,5%	37,4%	37,0%
7	40,2%	38,4%	36,9%	36,5%	36,4%	35,5%	36,7%	36,5%
8	39,2%	38,4%	37,2%	36,0%	36,1%	36,2%	36,0%	36,8%
9	39,1%	37,5%	36,4%	36,2%	36,1%	35,8%	36,0%	35,9%
10	39,0%	37,4%	36,7%	35,8%	35,8%	35,6%	35,4%	35,7%
11	38,5%	37,2%	36,1%	35,4%	35,0%	35,1%	35,5%	35,4%
12	38,8%	37,2%	36,0%	35,2%	34,9%	34,9%	35,0%	34,9%
13	39,0%	36,2%	35,7%	34,7%	34,7%	34,7%	34,9%	34,9%
14	38,6%	35,7%	35,3%	34,8%	34,7%	34,6%	34,7%	34,5%
15	38,9%	36,0%	34,7%	34,7%	34,4%	34,1%	34,1%	34,4%

Tabula 8

Gadījuma meža *out-of-box* kļūda, atkarībā no dažādām $mtry$ un $maxnode$ vērtībām, kad $ntree = 500$ (turpinājums)

	mtry						
maxnode	9	10	11	12	13	14	15
2	41,6%	41,7%	41,7%	42,0%	41,7%	42,7%	42,7%
3	39,5%	39,4%	40,3%	39,7%	39,4%	39,6%	40,2%
4	37,6%	37,9%	37,8%	37,8%	37,7%	37,8%	37,8%
5	37,7%	37,4%	37,4%	37,4%	37,3%	37,4%	37,3%
6	37,2%	36,9%	37,1%	36,9%	37,1%	37,1%	37,1%
7	36,5%	36,4%	36,7%	36,8%	37,0%	37,1%	37,3%
8	36,0%	36,1%	36,2%	36,2%	37,0%	36,6%	37,1%
9	36,0%	36,2%	36,2%	36,5%	36,3%	36,5%	36,8%
10	36,1%	36,1%	36,1%	36,4%	36,2%	36,3%	36,9%
11	35,4%	35,4%	35,7%	36,0%	36,1%	36,5%	36,5%
12	35,1%	35,2%	35,1%	35,7%	36,3%	35,9%	36,5%
13	34,9%	35,0%	35,7%	35,5%	35,7%	36,0%	35,7%
14	34,9%	35,1%	35,2%	35,3%	35,5%	35,4%	35,6%
15	34,9%	34,7%	35,0%	35,1%	35,4%	35,8%	35,7%

3. pielikums. Gradienta pastiprināšanas modeļu metrikas ar dažādām parametru kombinācijām

```
> gbm_parameters
Stochastic Gradient Boosting
15498 samples
   46 predictor
    2 classes: 'BAD', 'GOOD'
No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 10332, 10332, 10332
Resampling results across tuning parameters:
```

Tabula 9
***train()* funkcijas GBM modeļa precizitāte, atkarībā no dažādiem** $interaction.depth$ **un** $n.trees$ **vērtībām, kad** $shrinkage = 0,05$ **un** $n.minobsinnode = 10$

int.depth	n.trees	Precizitāte	int.depth	n.trees	Precizitāte
2	25	62,0%	4	150	68,9%
2	50	63,3%	4	175	69,6%
2	75	64,1%	4	200	70,0%
2	100	64,7%	4	225	70,8%
2	125	65,0%	5	25	64,2%
2	150	65,8%	5	50	66,7%
2	175	66,2%	5	75	67,6%
2	200	66,6%	5	100	68,6%
2	225	67,0%	5	125	69,4%
3	25	63,3%	5	150	69,9%
3	50	64,6%	5	175	71,1%
3	75	65,5%	5	200	72,0%
3	100	66,1%	5	225	72,8%
3	125	66,7%	6	25	65,3%
3	150	67,2%	6	50	66,7%
3	175	67,8%	6	75	68,1%
3	200	68,4%	6	100	69,4%
3	225	68,7%	6	125	70,4%
4	25	63,2%	6	150	71,2%
4	50	65,9%	6	175	72,6%
4	75	66,8%	6	200	73,6%
4	100	67,4%	6	225	74,6%
4	125	67,9%			

4.pielikums. Praktiskajā daļā izmantotais R kods

```
###### Ielādēju bibliotēkas ###########
require(rpart)
require(party)
library(plyr)
library(ranger)
library(randomForest)
library(ROCR)
library(pROC)
library(caret)
library(rattle)
library(rpart.plot)
library(RColorBrewer)
############### INPUT ##############
username<-Sys.getenv("USERNAME")
working_directory <- paste0("C:/Users/",username,"/Google Drive/Frauds")
working_file <- "Data.csv"
file_csv_seperator <<- ";"
target_name <- c("target")
set.seed(1)
seeds<-sample(1:33333, 5)
j<-1
seed<-seeds[j]
setwd(working_directory)
mydata<-read.csv(working_file, sep = file_csv_seperator,
na.strings=c("", "NA"), head=TRUE)
colnames(mydata)[colnames(mydata) == target_name] <- "target"
train<-mydata[mydata$sample == "train",]
test<-mydata[mydata$sample == "test",]
###### Klašu balansēšana train kopā#####
 set.seed(seed)
newdata1<-upSample(train,target,yname="target")
train<-newdata1[,-2]
train<-train[,!colnames(train) %in% c("transaction_mapping","sample","X")]
test<-test[,!colnames(test) %in% c("transaction_mapping","sample","X")]
colnames(test)[colnames(test) == target_name] <- "target"
```

```
colnames(train)[which(!colnames(test) %in% colnames(train))]
colnames(test)[which(!colnames(train) %in% colnames(test))]
#### klašu skaits katrā kopā ####
table(train$target)
table(test$target)
###### Aprēķina korelāciju skaitliskajiem mainīgajiem####
str(mydata)
skaitliskie<-c("x_1","x_2","x_6","x_7","x_10","x_12","x_14","x_25","x_27",
"x_28","x_29","x_37","x_38")
skait<-mydata[,which(colnames(mydata) %in% skaitliskie)]
korel.matr<-round(cor(skait),3)
 write.csv(korel.matr,file="final_korelacijas.csv")
#### saskaita kategorijas kategoriālajem mainīgajiem ####
for (i in 1:ncol(mydata)) { if (class(mydata[,i])=="factor"|
class(mydata[,i])=="character"|class(mydata[,i])=="logical")
 {cl[i]<-length(table(mydata[,i]))}}
c<-data.frame(colnames(mydata),cl)
########## Definēju funkcijas ##############
conf.matrix<-function(model,data,type){
  conf.matrix<- table(data\$target, predict(model,type=type,newdata = data))
  return(conf.matrix)}
class.error<-function(conf.matrix){
  class.error<-c()
  class.error[1]<-conf.matrix[1,2]/(conf.matrix[1,2]+conf.matrix[1,1])
  class.error[2]<-conf.matrix[2,1]/(conf.matrix[2,1]+conf.matrix[2,2])
  return(class.error)}
accuracy<-function(conf.matrix_train,conf.matrix_test){
  accuracy.train<-(conf.matrix_train[1,1]+conf.matrix_train[2,2])/18834
  accuracy.test<-(conf.matrix_test[1,1]+conf.matrix_test[2,2])/6093
  acc<-data.frame(paste(round(100*accuracy.train,2),"\%",sep="")
  ,paste(round(100*accuracy.test,2),"\%",sep=""))
  colnames(acc)<-c("accuracy.train","accuracy.test")
  return(acc) }
##### Rpart #############
par(mfrow = c(2, 2))
set.seed(seed)
```

```
cp<-c(0.01,0.005,0.001,0.0005)
for (i in 1:length(cp))
{ rpart_koks<-rpart(target~.,data=train,control=rpart.control
(maxdepth=5,cp=cp[i]))
rpart.koks.test.predict<-predict(rpart_koks,type="prob",test)
rpart.koks.test.prediction<- prediction
(rpart.koks.test.predict[,2], test$target)
rpart.koks.test.perf<-performance(rpart.koks.test.prediction,"tpr","fpr")
auc.test.rpart.koks<-performance(rpart.koks.test.prediction,"auc")@y.values[[1]]
gini.test.rpart.koks<-2*auc.test.rpart.koks-1
plot(rpart.koks.test.perf,main=paste("Rpart koks ar cp=",round(cp[i],4),sep="")
,col=2,lwd=1,xlim=c(0,1),ylim=c(0,1))
abline(a=0,b=1,lwd=2,lty=2,col="gray")
rpart.koks.train.predict<-predict(rpart_koks,type="prob",train)
rpart.koks.train.prediction<- prediction(rpart.koks.train.predict [,2], train$target)
rpart.koks.train.perf<- performance(rpart.koks.train.prediction,"tpr","fpr")
auc.train.rpart.koks<-performance(rpart.koks.train.prediction,"auc")@y.values[[1]]
gini.train.rpart.koks<-2*auc.train.rpart.koks-1
lines(rpart.koks.train.perf@x.values[[1]],rpart.koks.train.perf@
y.values[[1]], col = "green",lty=1,lwd=1)
legend("bottomright",c(paste("GINI train = ",round(100*gini.train.rpart.koks,2),"\%"
, sep = ""),paste("GINI test = ",round(100*gini.test.rpart.koks,
2),"\%", sep = ""),paste("AUC train
 =",round(100*auc.train.rpart.koks,2),"\%", sep = "")
,paste("AUC test = ",round(100*auc.test.rpart.koks,2),"\%"
, sep = "")),bty="n",cex=1)
conf.matrix_train<- conf.matrix(rpart_koks,train,"class")
conf.matrix_test<- conf.matrix(rpart_koks,test,"class")
class.error.train<-class.error(conf.matrix_train)
class.error.test<-class.error(conf.matrix_test)
accuracy(conf.matrix_train,conf.matrix_test)}
############## Random Forest ############
par(mfrow=c(3,3))
ntree_try<-c(100,300,500)
mtry<-seq(1:15,1)
maxnode<-seq(2,15,1)
```

```
i<-1
j<-1
k<-1
df<-data.frame()
for (i in 1:length(maxnode))
for (j in 1:length(mtry))
    {  set.seed(seed)
      rf<-randomForest(as.factor(target)~.,data=train,maxnode=maxnode[i]
      ,ntree=ntree_try[k],mtry=mtry[j])
      rf.test.predict<-predict(rf,type="prob",test) [,2]
      rf.test.prediction<- prediction(rf.test.predict, test$target)
      rf.test.perf<- performance(rf.test.prediction,"tpr","fpr")
      auc.test.rf<-performance(rf.test.prediction,"auc")@y.values[[1]]
      gini.test.rf<-2*auc.test.rf-1
      plot(rf.test.perf,main=paste("Random Forest ntree=", ntree_try[k], "
      ,mtry=",mtry[j],"maxnode=",maxnode[i]),col=2,lwd=1,xlim=c(0,1),ylim=c(0,1))
      abline(a=0,b=1,lwd=2,lty=2,col="gray")
      rf.train.predict<-predict(rf,type="prob",train) [,2]
      rf.train.prediction<- prediction(rf.train.predict, train$target)
      rf.train.perf<- performance(rf.train.prediction,"tpr","fpr")
      auc.train.rf<-performance(rf.train.prediction,"auc")@y.values[[1]]
      gini.train.rf<-2*auc.train.rf-1
      lines(rf.train.perf@x.values[[1]],rf.train.perf@y.values[[1]],
      col = "green",lty=1,lwd=1)
      legend("bottomright",c(paste("GINI train = ",round(100*gini.train.rf,2),"\%",
       sep = ""),paste("GINI test = ",round(100*gini.test.rf,2),"\%", sep = ""),
       paste("AUC train = ",round(100*auc.train.rf,2),"\%", sep = "")
       ,paste("AUC test = ",round(100*auc.test.rf,2),"\%", sep = "")),bty="n",cex=1)
      df[i,1]<-c(gini.train.rf)
      df[i,2]<-c(gini.test.rf)
      df[i,3]<-c(auc.train.rf)
      df[i,4]<-c(auc.test.rf)      }
colnames(df)<-c("gini_train","gini_test","AUC_train","AUC_test")
rf<-randomForest(target~.,train,maxnode=2,mtry=5,ntree=500)
###### GBM ###############
require(gbm)
```

```
require(dplyr)
par(mfrow=c(1,2))
set.seed(seed)
fitControl<-trainControl(method="cv", number=3)
gbmGrid <-  expand.grid(interaction.depth=c(2:6,by=2),
                        n.trees = seq(25,225,by=25),
                        shrinkage = 0.05,
                        n.minobsinnode = 10)
set.seed(seed)
gbm_default_minimize<-train(as.factor(target)~.,data=train,method="gbm",
trControl=fitControl,verbose=FALSE,metric = "Accuracy",
maximize=FALSE,tuneGrid=gbmGrid)
set.seed(seed)
gbm_default_maximize<-train(as.factor(target)~.,data=train,method="gbm",
trControl=fitControl,verbose=FALSE,metric ="Accuracy"
,maximize=TRUE,tuneGrid=gbmGrid)
set.seed(seed)
gbm_parameters<-train(as.factor(target)~.,data=train,method="gbm"
,trControl=fitControl,verbose=FALSE
,metric = "Accuracy",
maximize=FALSE,tuneGrid=gbmGrid)
gbm_parameters_max<-train(as.factor(target)~.,data=train,method="gbm"
,trControl=fitControl,verbose=FALSE,metric = "Accuracy",
maximize=TRUE,tuneGrid=gbmGrid)
set.seed(seed)
gbm<-gbm_parameters
gbm.test.predict<-predict(gbm,type="prob",test) [,2]
gbm.test.prediction<- prediction(gbm.test.predict, test$target)
gbm.test.perf<- performance(gbm.test.prediction,"tpr","fpr")
auc.test.gbm<-performance(gbm.test.prediction,"auc")@y.values[[1]]
gini.test.gbm<-2*auc.test.gbm-1
plot(gbm.test.perf,main="GBM, int.depth=1, n.trees=50",
col=2,lwd=1,xlim=c(0,1),ylim=c(0,1))
abline(a=0,b=1,lwd=2,lty=2,col="gray")
gbm.train.predict<-predict(gbm,type="prob",train) [,2]
gbm.train.prediction<- prediction(gbm.train.predict, train$target)
```

```
gbm.train.perf<- performance(gbm.train.prediction,"tpr","fpr")
auc.train.gbm<-performance(gbm.train.prediction,"auc")@y.values[[1]]
gini.train.gbm<-2*auc.train.gbm-1
lines(gbm.train.perf@x.values[[1]],gbm.train.perf@y.values[[1]],
 col = "green",lty=1,lwd=1)
legend("bottomright",c(paste("GINI train = ",round(100*gini.train.gbm,2),
"\%", sep = ""),paste("GINI test = ",round(100*gini.test.gbm,2),"\%", sep = ""),
paste("AUC train = ",round(100*auc.train.gbm,2),"\%", sep = ""),paste("AUC test = ",
round(100*auc.test.gbm,2),"\%", sep = "")),bty="n",cex=1.2)
conf.matrix_train<- conf.matrix(gbm,train,"raw")
conf.matrix_test<- conf.matrix(gbm,test,"raw")
class.error.train<-class.error(conf.matrix_train)
class.error.test<-class.error(conf.matrix_test)
accuracy(conf.matrix_train,conf.matrix_test)
```

Bakalaura darbs „Krāpniecisku kredītņēmēju noteikšana, izmantojot gadījuma meža un gradienta pastiprināšanas algoritmu" izstrādāts LU Fizikas un matemātikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: Ieva Brantevica

Rekomendēju darbu aizstāvēšanai

Vadītājs: asoc. Prof. Dr. mat. Jānis Valeinis 06.06.2016.

Recenzents: doc. Dr. mat. Nadežda Siņenko

Darbs iesniegts Matemātikas nodaļā __.06.2016.

Dekāna pilnvarotā persona: vecākā metodiķe Dzintra Holsta

Darbs aizstāvēts Valsts pārbaudījuma komisijas sēdē

____ 06.2016. prot. Nr._____

Komisijas sekretāre: docente Ingrīda Uļjane