LATVIJAS UNIVERSITĀTE FIZIKAS UN MATEMĀTIKAS FAKULTĀTE MATEMĀTISKĀS ANALĪZES KATEDRA

Lēmuma koku, gadījuma mežu un loģistiskās regresijas modeļu salīdzinājums klientu maksātspējas prognozēšanai

Bakalaura darbs

Autors: Kristīne Grundmane

Studenta apliecības Nr.: kg11097

Darba vadītājs: asoc. prof. Jānis Valeinis

Rīga 2017

Anotācija

Mūsdienu tehnoloģiju attīstība nodrošina iespēju jebkurā laikā un vietā saņemt naudas aizdevumu. Tas savukārt nozīmē, ka uzņēmumiem, kas izsniedz aizdevumus, ir jānodrošina ātra pieteikumu izvērtēšana un procesam jābūt maksimāli automatizētam. Balstoties uz vēsturiskiem datiem, var veidot modeļus, kas prognozēs jauno klientu maksātspēju. Mērķa pazīme pieņem tikai divas vērtības: 'labs' vai 'slikts'. Tā ir divu klašu klasifikācijas problēma. Darbā tiek apskatītas trīs metodes, kā risināt šo problēmu, - lēmumu koku algoritms, gadījuma mežu algoritms un loģistiskā regresija. Lēmumu koku un gadījuma mežu algoritmi ir mašīnmācīšanās algoritmi, kas tiek trenēti uz treniņa datiem. Ja treniņa dati ietver plašākas datu kombinācijas, tad algoritmu prognoze kļūst precīzāka. Tomēr, lai algoritmi neprognozētu labāku iznākumu nekā tas patiesībā ir sagaidāms, ir nepieciešams pārbaudīt modeļu pielāgotību.

Atslēgas vārdi: klasifikācijas un regresijas koki, lēmumu koki, koku apgriešana, gadījuma meži, loģistiskā regresija.

Anotation

Nowadays technology development gives opportunity to get loan at any place and at any time. That means that companies that gives loans have to ensure fast application evaluation and process have to bee as automatic as possible. Depending on historical data, there is possibility to create models for new client evaluation. Target have only two values: ’good’ or ’bad’. This is two class classification problem. In this work there are three methods that are compared - decision tree algorithm, random forest algorithm and logistic regression. Decision tree and random forest algorithms are machine learning algorithms, that are trained on training data. If training data contains more variable combinations, algorithm prediction is more accurate. However over fitting needs to be checked.

Key words: classification and regression trees, decision trees, tree pruning, random forest, logistic regression.

Saturs

IEVADS

Katra uzņēmuma mērķis ir gūt pēc iespējas lielāku peļņu. Uzņēmuma, kas izsniedz aizdevumus, peļņa ir atkarīga no tā, vai cilvēks, kam tiek izsniegts aizdevums, naudu atmaksās vai nē. Tātad ir nepieciešama sistēma, pēc kuras izvērtēt, kuriem klientiem izsniegt aizdevumu.

Noteikt, vai klients spēs atmaksāt aizdevumu, nozīmē atrisināt divu klašu klasifikācijas problēmu. Klientam tiek piešķirta kategorija 'labs', ja noteiktā laikā aizdevums ir atmaksāts, attiecīgi kategorija 'slikts' klients tiek piešķirta tiem klientiem, kas laikus nav atmaksājuši aizdevumu.Klasifikāciju var definēt kā uzdevumu iemācīties mērķa funkciju, kas katram novērojumam piešķir vienu no definētām klasēm.

Klasifikācijas modeļus visbiežāk izmanto divām sekojošām lietām:

1. Aprakstošā modelēšana. Klasifikācijas modeli var izmantot kā aprakstošu palīgu, lai atšķirtu novērojumus, kas pieder dažādām klasēm.

2. Prognozējošā modelēšana. Klasifikācijas modeli var izmantot, lai prognozētu nezināmas novērojumu klases.

Klasifikācijas tehnika ir sistemātiska pieeja klasifikācijas modeļa izveidošanai no ievades datu kopas. Ir tādas tehnikas kā lēmumu koki, neirona tīkli, atbalsta vektoru mašīnas, naivais Beijesa algoritms un daudzas citas. Katra tehnika lieto mācīšanās algoritmu, lai identificētu modeli, kas labāk raksturo attiecības starp pazīmju jeb faktoru kopu un datu kopas vērtībām. Mašīnmācīšanās algoritmi (lēmumu koku un gadījuma mežu algoritmi)tiek trenēti uz treniņa datiem. Tas nozīmē - ja datu kopa ietver plašāku klāstu dažādu faktoru kombinācijas, tad modelis spēs prognozēt precīzāk. Taču loģistiskās regresijas algoritms ir statistiskā modelēšana, kas nozīmē, ka attiecības starp faktoriem tiek aplūkotas matemātisku vienādojumu formā. Mācīšanās algoritma mērķis ir izveidot modeli ar labu ģeneralizētu spēju akurāti prognozēt iepriekš nezināmas vērtības.

Lēmumu koku algoritma pirmsākumi ir meklējami 1950. gadā, kā Hunta klasifikācijas metodes algoritmi, tad Kvinlins attīstīja Hunta algoritmu par algoritmu ID3 1975. gadā, 1993. gadā to uzlabojot un nosaucot par C4.5 algoritmu. Pavisam nesenā pagātnē 2011. gadā tika prezentēta algoritma C4.5 uzlabota versija - algoritms C5.0. Savukārt gadījuma mežu algoritms pirmo reizi tika minēts 2001. gadā.

Modeļa precizitāti raksturo tas, kādu skaitu no testa datu vērtībām modelis prognozēja pareizi un nepareizi. Tiek meklēts modelis ar visaugstāko precizitāti vai ekvivalenti zemāko kļūdas mēru. Tomēr ir svarīgi pārbaudīt to, vai modelis neprognozē pārāk pozitīvu rezultātu - rezultātu, kas ir labāks par sagaidāmo.

Darba mērķis ir izvēlēties piemērotāko modeli klientu maksātspējas prognozēšanai.

Lai sasniegtu mērķi ir izvirzīti tādi darba uzdevumi kā lēmumu koku izveides algoritmu aplūkošana, uzlabot lēmumu koku precizitāti tos apgriežot, apskatīt pārāku mašīnmācīšanās algoritmu - gadījuma mežu algoritmu. Salīdzināt iegūtos modeļus ar parametrisku metodi - loģistisko regresiju. Kā arī jāizvēlas mēri, kā noteikt, kurš algoritms prognozē precīzākus rezultātus.

Darbs sastāv no 9 nodaļām un 53. lapām, kur pēdējās sešas lapas ir pielikumi. Sākotnēji tiek aprakstīti lēmumu koku algoritmi, to uzlabošana, apgriežot kokus. Tālāk tiek aplūkots gadījuma mežu algoritms un loģistiskās regresijas modelis, tā koeficientu nozīmība. Pēc tam tiek aplūkoti mēri, ar kuriem salīdzināt iegūto modeļu prognozēšanas precizitāti un rezultātu daļa, kur teorētiskajā daļā aprakstītais tiek implementēts valodā R.

1 LĒMUMU KOKU ALGORITMS

Lēmumu koks ir viena no klasifikācijas metodēm, kas, balstoties uz konkrētu pazīmju kopumu, ļauj pieņemt lēmumu, kādu klasi piešķirt konkrētam novērojumam. Tā ir vienkārša tehnika, tādēļ tiek plaši lietota.

Piemērs 1. Klienti, kuriem tiek izsniegts aizdevums, tiek grupēti divās grupās: 'labs' klients vai slikts. Kad pieteikumu izveido jauns klients, tad nav zināms, kādu grupu viņam piešķirt. Ja klientam tiek piešķirta grupa 'slikts', tad aizdevums netiks izsniegts. Tātad, lai noskaidrotu, vai izsniegt aizdevumu, var uzdot jautājumu, vai klientam šobrīd ir parādi. Attiecīgi, ja klientam ir parādi, pieteikums netiek apstiprināts, jo tiek sagaidīts, ka klients būs grupā 'slikts'. Savukārt klientiem, kuriem nav parādu var uzdot nākamos jautājumus: vai kādreiz ir atmaksāts aizdevums, vai klientam ir parādi, vai ir pastāvīga darba vieta un citus.

Piemērs parāda, ka var atrisināt klasifikācijas problēmu, uzdodot virkni ar jautājumiem un katru reizi saņemot atbildi, tiek uzdota nākamā jautājumu sērija. Tā tas turpinās līdz nonāk pie konkrētas atbildes. Jautājumu sērijas un atbildes var organizēt lēmumu kokā, kas sastāv no trīs veidu virsotnēm un lokiem jeb zariem. Lēmumu kokus var definēt un raksturot, izmantojot grafu teorijas elementus.

Definīcija 1. Par grafu sauc kopu pāri $G = (X, U)$, kur X ir netukša kopa, bet U sastāv no šīs kopas elementu sakārtotiem pāriem $(x, y) : U \subset X \times X$. Kopas X elementus sauc par virsotnēm, bet kopas U elementus - par lokiem. [13]

Teorēma 1. Katru savienotu grafu ar n virsotnēm un $n - 1$ lokiem sauc par koku. [13]

Pierādījums Ar G apzīmē savienotu grafu ar n virsotnēm un $n-1$ lokiem. Jāparāda, ka G nesatur ciklus. Pieņem pretējo, ka G satur ciklus. Noņem vienu loku no cikla tā, ka rezultējošais grafs ir savienots. Turpina noņemt pa vienam lokam no viena cikla, kamēr rezultējošais grafs H ir koks. Tā kā H satur n virsotnes (jo tika noņemti tikai loki), tad H loku skaits ir $n - 1$. Tagad loku skaits grafam G ir lielāks par loku skaitu grafam H. Tātad $n - 1 > n - 1$, kas nav iespējams. Tātad ir pierādīts, ka G nav ciklu un tas ir koks.

Definīcija 2. Virsotni x sauc par izteku jeb saknes virsotni, ja neeksistē neviens loks, kas no jebkuras citas virsotnes y ved uz virsotni x:

$$\forall y \in X(y, x) \notin U, \tag{1.1}$$

kur X ir virsotņu kopa, U ir loku kopa, (y, x) ir loks jeb kopas U sakārtots pāris.

Definīcija 3. Par iekšēju virsotni sauc virsotni, kurai ir tieši viens ienākošs loks un divi vai vairāk izejoši loki.

Definīcija 4. Par lēmuma koka lapu jeb ieteku sauc virsotni x, ja neeksistē neviens loks, kas no šīs virsotnes ved uz jebkuru citu virsotni y:

$$\forall y \in X(x, y) \notin U, \tag{1.2}$$

kur X ir virsotņu kopa, U ir loku kopa, (y, x) ir loks jeb kopas U sakārtots pāris.

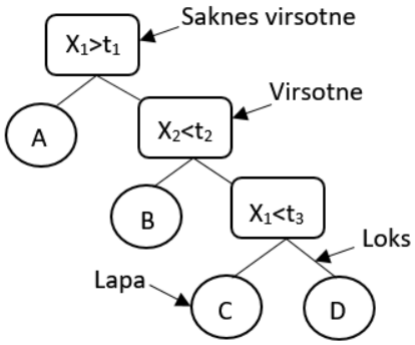
Lēmumu kokā katrai lapai tiek piedēvēta konkrēta klase. Tas nozīmē, ka brīdī, kad novērojums ir sasniedzis lapu, tam tiek piešķirta konkrētās lapas klase.

Definīcija 5. Par lēmuma koka zaru sauc loku, kas savieno divas lēmuma koka virsotnes.

Definīcija 6. Par lēmuma koka 'bērna' virsotni virsotnei x, sauc jaunizveidotu virsotni, kas izveidojas pēc virsotnes x šķelšanas.

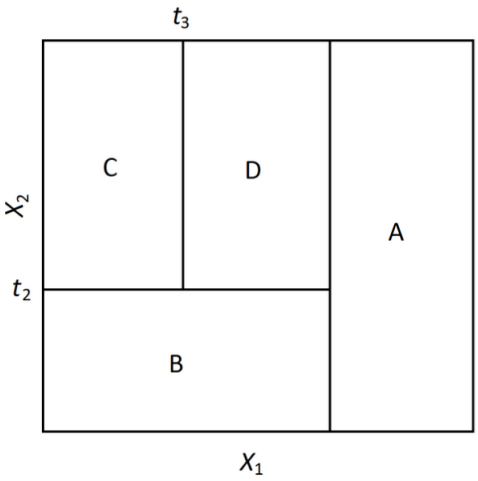
Definīcija 7. Par lēmuma koka 'vecāku' virsotni virsotnei x, sauc virsotni, kuru šķeļot ir izveidojusies virsotne x.

Attēlā 1.1. ir redzama lēmumu koka struktūra, kur X_1 un X_2 ir divi faktori, kas tiek sadalīti ar parametriem t_1, t_2, t_3.



Att. 1.1: **Lēmuma koka struktūra**

Attēlā redzamais lēmumu koks ir atbilstošs 1.2 attēlā redzamajam divu dimensiju faktora laukumam. Laukums ir rekursīvi bināri šķelts. Ja virsotnes nosacījums izpildās, tad koks dodas pa kreiso zaru. Attēlos ir redzams, ka lēmumu kokam ir četras lapas, kas reprezentē atbilstošas klases: A, B, C, D. Grafiska lēmumu koku reprezentācija ļauj viegli uztvert izvērtēšanas procesu. Virsotnēs definētos testa stāvokļus var saukt par noteikumiem. Pēc lēmumu koka izveidošanas ir iegūts noteikumu kopums, kuriem ir jāizpildās, lai klients saņemtu aizdevumu.



Att. 1.2: **Sadalīts divdimensionāls īpašību laukums**

No dotu pazīmju kopuma var konstruēt eksponenciāli daudz lēmumu koku. Ir attīstīti efektīvi algoritmi, kas samērīgā laikā izveido pamatoti precīzus lēmumu kokus. Viens no šādiem algoritmiem ir Hunta algoritms, kas ir pamatā daudziem eksistējošiem lēmumu koku algoritmiem, iekļaujot ID3(*Iterative Dichotomizer 3*, tulkojumā no angļu valodas – iteratīvais dalītājs), C4.5 un CART (angļu *classification and regression trees*, tulkojumā – klasifikācijas un regresijas koki).[11]

1.1 Hunta algoritms

Hunta klasifikācijas problēmu risināšanas metodes ar CLS (angļu *Concept Learning Systems*, tulkojumā – koncepta mācīšanās sistēmas) ir zināmas kopš 1950. gada.[11]

Hunta algoritmā lēmumu koki tiek audzēti rekursīvi dalot treniņa datus tīrākās apakškopās. Ar tīrāku apakškopu jāsaprot apakškopa, kurā konkrētas klases novērojumu skaits ir pārākumā.
Definīcija 8. Ar D_t apzīmē datu kopu lēmuma koka virsotnē t un ar $y = \{y_1, y_2, \ldots, y_n\}$ apzīmē klašu kopu, kur mērķa mainīgais pieņem n klases. Tad Hunta algoritmu definē sekojoši:

1.Solis. Ja visi D_t ieraksti pieder vienai klasei y_t, tad virsotne t ir lapa, kas reprezentē klasi y_t. Formula?

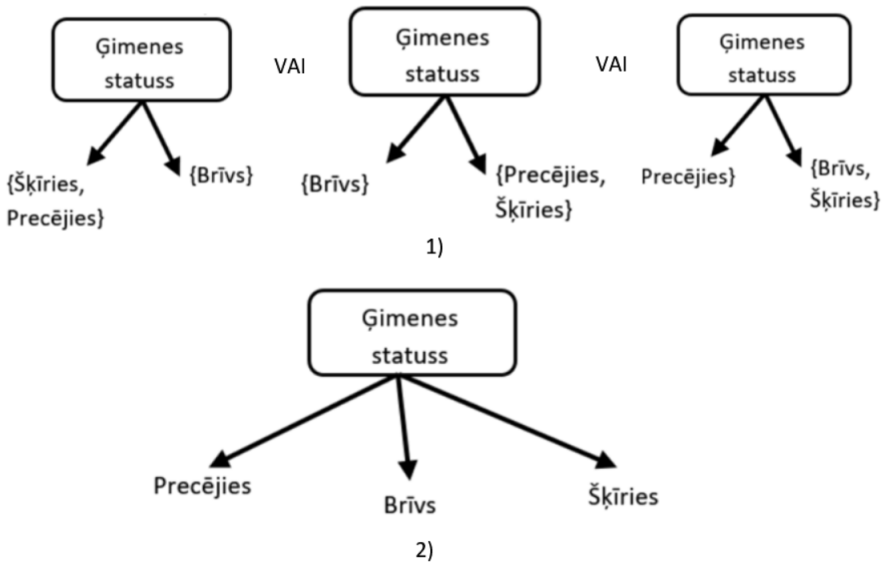
2.Solis. Ja kopa D_t satur novērojumus, kas pieder vairākām klasēm, tiek izvēlēts pazīmes nosacījums, kas ierakstus sadala apakškopās. Tiek izveidotas jaunas virsotnes (‘bērnu’ virsotnes) un kopas D_t ieraksti tiek attiecīgi sadalīti. Tad algoritms rekursīvi tiek pielietots katrai jaunajai virsotnei. Formula?

Hunta algoritms darbosies, ja katra kombinācija no pazīmes vērtībām ir treniņa datos un katrai kombinācijai ir unikāla klase. Šie pieņēmumi ir pārāk strikti lielākajā daļā praktisku gadījumu. Ir nepieciešami papildu nosacījumi, lai tiktu galā ar sekojošiem gadījumiem:

1.)Pastāv iespēja, ka kāda no ‘bērna’ virsotnēm, kas ir radīta otrajā solī, ir tukša, tas ir, nav ierakstu, kas atbilst šai virsotnei. Šādā gadījumā virsotne tiek atzīta par lapu ar tādu klasi, kāda ir vairākumam treniņa datu tās ’vecāku’ virsotnē.

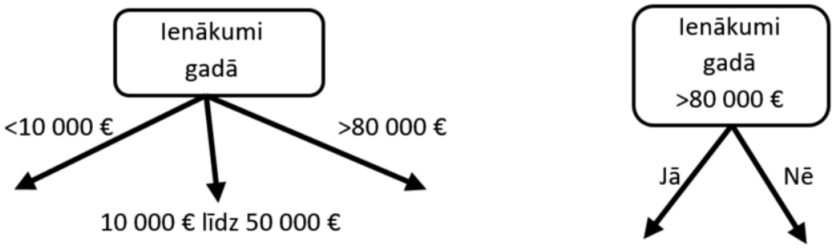
2.)Solī divi, ja visiem ierakstiem saistītiem ar D_t ir identiskas pazīmes vērtības (izņemot klases), tad nav iespējams turpināt datu šķelšanu. Šajā gadījumā virsotne kļūst par lapu ar tādu klasi, kāda ir vairākumam novērojumu treniņa datos saistītiem ar šo virsotni.

Testa stāvokļu interpretācija Lēmumu koka virsotnes tiek dalītas pēc pazīmes testa stāvokļa. Pazīmes var būt bināras (klientam ir parāds vai nav), nominālas (klienta ģimenes status) un nepārtrauktas (klienta alga). Bināro pazīmju testa stāvokļi ģenerē divus iespējamus iznākumus, tātad no virsotnes, kurā testa stāvoklis ir binārs, vienmēr izies divi zari. Tā kā nominālas pazīmes var saturēt vairākas vērtības, tad testa stāvoklis var tikt izteikts divos veidos, kas attēloti 1.3 attēlā. Pirmajā gadījumā dati tiek šķelti bināri, grupējot pazīmes vērtības, otrajā gadījumā dati tiek šķelti vairākās grupās. Otrajā gadījumā no virsotnes izies tieši tik zari, cik vērtības ir konkrētajai pazīmei.



Att. 1.3: **Testa nosacījumi nominālām pazīmēm**

Savukārt nepārtrauktas pazīmes testa stāvokli var izteikt kā salīdzinājumu (ienākumi ir lielāka vai mazāka par izvēlētu konstanti) ar bināru iznākumu, vai grupēt mainīgā vērtības (klienta ienākumus sadalīt trīs grupās).



Att. 1.4: **Testa nosacījumi nepārtrauktām pazīmēm**

Kad dotās pazīmes ir aplūkotas, jāsaprot cik ilgi turpināt koka audzēšanu. Pāris izplatīti apstāšanās noteikumi:

- Visiem novērojumiem treniņu datu kopā ir viena vērtība.
- Ir sasniegts koka maksimālais dziļums (visas pazīmes ir apskatītas).
- Labākais šķelšanas kritērijs (plašāk aprakstīts otrajā nodaļā) nav lielāks par noteiktu sliekšni.

Minētie apstāšanās kritēriji ir viegli saprotami un implementējami. Tomēr apstāšanās kritēriji nav plaši izmantoti, jo vieglāka un precīzāka pieeja ir koku apgriešana (ir aprakstīta trešajā nodaļā) pēc to izveidošanas.

1.2 ID3 algoritms

J.R. Kvinlins izstrādāja ID3 algoritmu. Pirmo reizi algoritms tika minēts 1975. gadā grāmatā 'Machine Learning'. [3] Algoritms veido kokus balstoties uz informāciju (informācijas ieguvumu), kas iegūta no treniņa datiem un tad pielieto to, lai klasificētu jaunus datus.
Definīcija 9. Dotai pazīmju kopai $X_1, X_2, \ldots, X_n$,mērķa pazīmei C un treniņa datu kopai S algoritms tiek definēts sekojoši:[3]

1. Ievades dati: R - pazīmju kopa (bez mērķa pazīmes), C- mērķa pazīme, S - treniņa dati.
2. Pārbauda vai S ir tukša kopa. Ja tā ir, tad lēmumu koks sastāv tikai no vienas virsotnes ar vērtību, kas apzīmē neveiksmīgu mēģinājumu.
3. Pārbauda vai visiem S elementiem mērķa vērtība ir vienāda. Ja apgalvojums ir patiess, tad lēmumu koks sastāv no vienas virsotnes ar to pašu mērķa vērtību.

4. Pārbauda vai pazīmju kopa ir tukša. Ja tas izpildās, tad lēmumu kokam ir tikai viena virsotne ar vērtību, kas visbiežāk ir mērķa pazīmei kopā S.

5. Izvēlas pazīmi ar vislielāko informācijas ieguvumu (vai vismazāko entropiju) starp visām pazīmēm un šķeļ kopu S pēc izvēlētās pazīmes.

6. Algoritms rekursīvi turpina šķelt apakškopas, izmantojot tikai tās pazīmes, kuras vēl nav izmantotas, līdz brīdim, kad iestājas kāds no apstāšanās kritērijiem.

Piektajā algoritma solī kā šķelšanas kritērijs tiek izmantots informācijas ieguvuma mērs. Informācijas ieguvuma aprēķināšanā tiek izmantota entropija.

Definīcija 10. Ar $p(i|t)$ apzīmē daļu no ierakstiem, kas pieder klasei i dotā virsotnē t. (Var arī izlaist atsauci uz virsotni t un apzīmēt daļu ar p_i.) Tad algoritma entropiju definē sekojoši:[10]

$$Entropija(t) = -\sum_{i=0}^{n-1} p(i|t) log_2 p(i|t), \tag{1.3}$$

kur n ir klašu skaits.

Definīcija 11. Informācijas ieguvums ir kritērijs, kas tiek uzdots ar formulu:[10]

$$IG(F) = Entropija(S_1) - Entropija(S_2), \tag{1.4}$$

kur F ir pazīme, kurai tiek aprēķināts informācijas ieguvums, S_1 ir datu kopa pirms šķelšanas un S_2 ir datu kopa pēc šķelšanas. Tā kā pēc šķelšanas izveidojas vairākas virsotnes, $Entropija(S_2)$ ir kopējā entropija jaunajām virsotnēm. Katra jaunā dalījuma entropija tiek svērta attiecībā no ierakstu skaitu proporcijas konkrētajā dalījumā. To var uzrakstīt sekojošā formulā:

$$Entropija(S_2) = \sum_{i=0}^{k-1} w_i Entropija(P_i), \tag{1.5}$$

kur k ir jauno virsotņu skaits, w_i ir novērojumu proporcija $i-$tajā jaunizveidotajā virsotnē un P_i ir $i-$tā jaunizveidotā virsotne.

Divu klašu problēmai, klases sadalījumu katrā virsotnē var uzrakstīt kā (p_0, p_1), kur $p_1 = 1-p_0$. Klienta maksātspējas noteikšana ir divu klašu problēma. Tātad katrā virsotnē novērojumi piederēs vienai vai otrai klasei un klašu sadalījumu katrā virsotnē var uzrakstīt kā (p_0, p_1).

Lai ID3 algoritms darbotos pareizi, treniņa datu kopai jāapmierina sekojoši nosacījumi:

- Treniņa datos ir jābūt mērķa pazīmes vērtībām.

- Klasēm jābūt diskrētām. Arī nepārtrauktām vērtībām jābūt sadalītām grupās (piemēram, tādai pazīmei kā vecums ir jābūt grupētai: zem 30 gadiem, no 30 līdz 60 gadiem, virs 60 gadiem).
- Katru novērojumu raksturo visas pazīmes – nevar būt trūkstošas vērtības (tukšumi).

1.3 C4.5 algoritms

Algoritmu C4.5 J.R. Kvinlins piedāvāja 1993. gadā, lai pārvarētu ID3 algoritma ierobežojumus. [4]

C4.5 uzlabojumi, salīdzinot ar ID3 algoritmu:

- Iespēja lietot treniņa datus ar trūkstošām vērtībām.
- Ir iespējama koka apgriešana pēc tā izveides.
- Iespēja lietot pazīmes ar diskrētām un nepārtrauktām vērtībām.

Viens no ID3 algoritma ierobežojumiem: ID3 algoritms ir pārāk jutīgs pret lielu skaitu vērtībām. Piemēram, personas kods katram cilvēkam ir atšķirīgs, testējot datus ar šādu pazīmi, vienmēr būs zemas entropijas nosacījuma vērtības. Lai novērstu šādu problēmu, C4.5 algoritms izmanto informācijas ieguvuma proporcijas mēru.

Definīcija 12. Ieguvuma proporcijas mērs GR pazīmei F ir mērs, kas uzdots ar sekojošu formulu: [6]

$$GR(F) = \frac{IG(F)}{SI(F)}, \tag{1.6}$$

kur $SI(F)$ apzīmē formulu:

$$SI(F) = -\sum_{j=1}^{n} p'\left(\frac{j}{p}\right) log\left(p'\left(\frac{j}{p}\right)\right), \tag{1.7}$$

.

kur $p'\left(\frac{j}{p}\right)$ ir proporcija no elementiem, kas atrodas pozīcijā p, pieņemot $j-$tā testa vērtību, IG ir informācijas ieguvums.[2]

Pazīme ar augstāko normalizēto informācijas ieguvumu tiek izvēlēta veikt lēmumu, kā šķelt datus. Jāņem vērā, ka, atšķirībā no entropijas, šī definīcija ir neatkarīga no novērojumu sadalījuma dažādās klasēs.

2011. gadā tika prezentēta uzlabota versija C4.5 algoritmam – algoritms C5.0.[6] C5.0 algoritma uzlabojumi pār algoritmu C4.5: [6]

- Ātrums – C5.0 algoritms darbojas ievērojami ātrāk nekā C4.5 algoritms.
- Atmiņas lietojums – C5.0 algoritma rezultāti aizņem mazāk atmiņu.
- Mazāki lēmumu koki – C5.0 algoritms iegūst līdzīgus rezultātus kā C4.5 ar mazākiem lēmumu kokiem.
- Iespēja lietot metodes kokus un to precizitātes uzlabošanai.
- Atsijāšana – viena no C5.0 algoritma opcijām ir automātiska atsijāšana, kas izslēdz pazīmes, kuras nedod uzlabojumu.

1.4 Klasifikācijas un regresijas koki

Klasifikācijas un regresijas koku algoritms pieļauj gan kategoriskus, gan skaitliskus mainīgos. Katrai iekšējai virsotnei ir tieši divi izejas loki. Metode ir robusta pret izlēcējiem. Parasti šķelšanas algoritms izolē izlēcējus atsevišķā virsotnē vai virsotnēs.

Definīcija 13. Par klasifikācijas un regresijas koku algoritmu sauc algoritmu, kas sastāv no trim sekojošiem soļiem:

1. Maksimuma koka konstruēšana.
2. Pareizā izmēra koka izvēle jeb koka apgriešana.
3. Jaunu datu klasifikācija, izmantojot konstruēto koku.

Definīcija 14. Par maksimuma koku sauc tādu koku, kas sadala treniņa datus līdz pat pēdējam novērojumam, tas ir brīdis, kad virsotne satur novērojumus tikai no vienas klases. Lai izveidotu maksimuma koku jāapskata gan klasifikācijas, gan regresijas koki.

Klasifikācijas koki tiek izmantoti gadījumos, kad katram novērojumam no kopas ir zināma klase. Modelis klientu maksātspējas prognozēšanai tiek būvēts uz vēsturiskiem datiem. Šajos datos karam klientam ir atbilstoša klase – 'labs' vai 'slikts' klients.

Savukārt regresijas koki tiek veidoti, kad novērojumiem nav atbilstošas klases. Tā vietā ir atbildes vektors, kas reprezentē atbildes vērtības katram mainīgajam. Regresijas kokos šķelšana tiek veikta atbilstoši kļūdu kvadrātu minimizēšanas algoritmam, kas nozīmē, ka jāminimizē sagaidāmā dispersiju summa divām rezultējošām virsotnēm. Kļūdu kvadrātu summas minimizēšanas algoritms ir identisks Gini šķelšanas kritērijam.

Tā kā regresijas koku lapas prognozē konkrētu skaitli nevis klasi, tad klientu maksātspējas prognozēšanai tos nevar izmantot.

2 MĒRI LĒMUMA KOKA VIRSOTNES LABĀKĀ ŠĶĒLUMA IZVĒLEI

Mēri, kuri tiek izmantoti, lai atrastu labāko šķēlumu, bieži tiek balstīti uz virsotņu nekārtības līmeņa. Jo mazāks nekārtības līmenis, jo asimetriskāks klases sadalījums.

Definīcija 15. Mainīgajam P ar k diskrētām vērtībām, sadalītām atbilstoši $P = (p_1, p_2, \ldots, p_k)$ par nekārtības mēru sauc funkciju $\phi : [0,1]^k -> R$, kas apmierina sekojošus nosacījumus: [10]

- $\phi(P) \geq 0$,
- $\phi(P)$ ir minimums, ja $\exists i$ tāds, ka komponente $p_i = 1$,
- $\phi(P)$ ir maksimums, ja $\forall i, 1 \leq i \leq k, p_i = \frac{1}{k}$,
- $\phi(P)$ ir simetriska attiecībā pret P komponentēm ,
- $\phi(P)$ ir gluds (diferencējams katrā punktā)tā diapazonā.

Gadījumā, kad varbūtību vektors satur vienu komponenti (mainīgajam x tiek piešķirta tikai viena vērtība), mainīgais ir definēts kā tīrs. No otras puses, ja visas komponentes ir vienādas, tad nekārtības līmenis sasniedz maksimumu.

Lai noskaidrotu, cik labi testa nosacījumi darbojas, jāsalīdzina nekārtības līmenis 'vecāku' virsotnei (pirms dalīšanas) ar 'bērnu' virsotnēm (pēc dalīšanas). Jo lielāka starpība, jo labāki testa nosacījumi.

Definīcija 16. Ieguvums Δ ir kritērijs, kuru var izmantot, lai noteiktu šķelšanas kritērija vērtību:

$$\Delta = I(n) - \sum_{v_j=1}^{k} \frac{N(v_j)}{N} I(v_j), \tag{2.1}$$

kur $I(n)$ – nekārtības mērs dotajai virsotnei n, N - kopējais ierakstu skaists 'vecāku' virsotnē, k - faktoru skaits, $N(v_j)$- ierakstu skaits, kas asociēti ar bērna virsotni v_j.

Lēmumu koku indukcijas algoritmi bieži izvēlas testa nosacījumus, lai maksimizētu ieguvumu Δ. Tā kā izvēlētais nekārtības mērs $I()$ ir vienāds visiem testa nosacījumiem, tad maksimizēt ieguvumu nozīmē minimizēt svērto vidējo nekārtības mēru 'bērnu' virsotnēm.

Definīcija 17. Klasifikācijas kļūda ir mērs, kas tiek aprēķināts dalot nepareizi klasificētos novērojumus ar visu novērojumu skaitu.

Definīcija 18. Gini indekss ir uz nekārtības bāzēts kritērijs, kas mēra diverģenci starp mērķa pazīmju vērtību varbūtību sadalījumu. Virsotnei t tas tiek definēts sekojoši: [10]

$$Gini(t) = 1 - \sum_{i=0}^{n-1} [p(i|t)]^2, \tag{2.2}$$

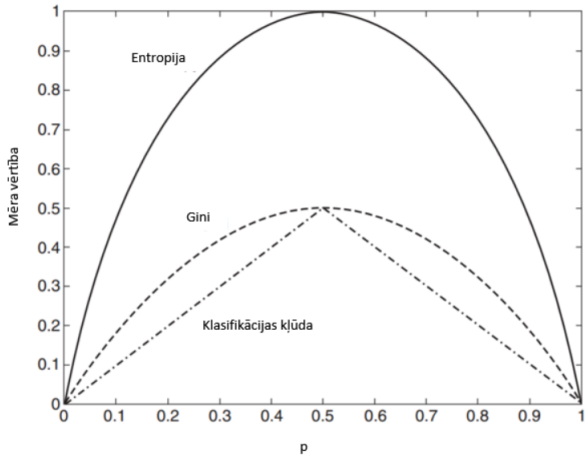
kur n ir klašu skaits un $p(i|t)$ ir daļa no ierakstiem, kas pieder klasei i dotā virsotnē t.

Definīcija 19. Informācijas ieguvums ir uz nekārtības bāzēts kritērijs, kas izmanto entropijas mēru. Tas tiek definēts ar sekojošu formulu:

$$IG(S, A) = Entropija(S) - \sum_{v \in (A)} w_v * Entropija(S_v), \tag{2.3}$$

kur $Entropija(S) = \sum_{i=0}^{n-1} p(i|t) * log_2[p(i|t)]$, $p(i|t)$ ir daļa no ierakstiem, kas pieder klasei i dotā virsotnē t, n ir klašu skaits S ir datu kopa pirms šķelšanas, S_v ir datu kopa jaunizveidotajā virsotnē v, A ir kopa no jaunizveidotajām virsotnēm un w_v ir svari jeb novērojumu proporcija konkrētajā virsotnē v.

Attēlā 2.1 redzams entropijas, Gini indeksa un klasifikācijas kļūdas mēru salīdzinājums bināra klasifikatora problēmai. P apzīmē ierakstus, kas pieder vienai no divām klasēm.



Att. 2.1: **Bināra klasifikatora problēmas mēru salīdzinājums**

Visi trīs mēri sasniedz maksimālo vērtību, kad $p = 0.5$. Minimuma vērtības tiek sasniegtas, kad visi ieraksti pieder vienai klasei ($p = 0$ vai $p = 1$).

Definīcija 20. Ticamības proporcijas Hī- kvadrāta statistika G^2 tiek definēta sekojoši: [10]

$$G^2(A, S) = 2 * ln(2) * \mid S \mid * IG(A, S), \tag{2.4}$$

kur $IG(A, S)$ ir informācijas ieguvums, A ir kopa no jaunizveidotajām virsotnēm, S ir datu kopa pirms šķelšanas.

Šī proporcija ir noderīga mērot statistisko nozīmību informācijas ieguves kritērijam. Nulles hipotēze (H_0): ievades īpašība un mērķa īpašība ir nosacīti neatkarīgas. Ja H_0 izpildās, testa statistikai ir Hī kvadrāta sadalījums.

Definīcija 21. Ieguvumu proporcijas mērs GR ir normalizēts uz nekārtības bāzēts kritērijs. Tas normalizē informācijas ieguvumu sekojošā veidā:

$$GR(A,S) = \frac{IG(A,S)}{Entropija(A,S)}. \tag{2.5}$$

Uz nekārtības bāzēti kritēriji ir novirzīti uz īpašību pusi, kurām ir lielāks vērtību apjoms. Ieguvuma proporcijas kritēriju ir ieteicams lietot divos posmos. Vispirms informācijas ieguvums tiek aprēķināts visām īpašībām. Kā rezultātā, tiek apskatītas tikai īpašības, kuras ir vismaz tik labas kā vidējais informācijas ieguvums. Tiek izvēlēta īpašība, kas ir ieguvusi labāko proporcijas ieguvumu. Ieguvumu proporcijai ir tendence pārspēt informācijas ieguvuma kritēriju – gan no precizitātes aspekta, gan no klasifikatoru sarežģītības aspekta.

Definīcija 22. Par *Twoing* kritēriju virsotnei t sauc mēru, kas maksimizē sekojošu funkciju:[11]

$$\frac{P(t_L)P(t_R))}{4}\left(\sum_{c\in a_i} \mid P(c \mid t_L) - P(c \mid t_R) \mid \right)^2, \tag{2.6}$$

kur $P(t_L)$ un $P(t_R)$ ir varbūtības doties pa labi vai kreisi (izvēlēties vienu vai otru grupu), $P(c \mid t_L)$ un $P(c \mid t_R)$ attiecīgi ir datu proporcijas t_L un t_R, kas pieder klasei c, un a_i pazīme.

Twoing kritērijs ir binārs kritērijs. Bināro kritēriju izmanto, lai izveidotu bināros lēmumu kokus. Šādi mēri ir bāzēti uz ievades īpašību kopuma sadalīšanu divās apakšgrupās. Gini indekss var sastapties ar problēmām, kad kopas mērķa pazīmei ir relatīvi plašas, šādos gadījumos var lietot *Twoinga* kritēriju. Kad mērķa īpašība ir bināra, Gini un Twoinga kritēriji ir ekvivalenti.

3 LĒMUMU KOKA APGRIEŠANAS METODES

Lēmumu koki klasificē datus sašķirojot tos no saknes mezgla līdz lapas mezglam. Katrs mezgls kokā pārbauda vienas pazīmes vērtības. Jo vairāk pazīmju, jo koks izaug lielāks. Taču lielāks koks nebūt nenozīmē arī precīzākus lēmumus. Pie tam kad koks ir izveidots, to var konvertēt kā sarakstu ar noteikumiem. Ja koks ir pārāk liels, tad arī noteikumu saraksts būs pārāk garš un neizprotams. Tā kā klientu maksātspējas noteikšanai tiek izmantots liels skaits faktoru, pēc tā izveidošanas ir nepieciešams koku samazināt.

Kļūdas sarežģītības funkcijas optimizēšana Apgriešanu, izmantojot kļūdas sarežģītību, prezentēja L. Breimans 1984. gadā.

Definīcija 23. Koka T kopējās izmaksas $C_\alpha(T)$ definē ar sekojošu formulu :[10]

$$C_\alpha(T) = R(T) + \alpha \mid \tilde{T} \mid \ , \alpha >= 0, \tag{3.1}$$

kur $R(T)$ ir koka T lapu svērta summēta kļūda, α ir sarežģītības parametrs (sods par koka $\tilde{T}$ sarežģītību) un $\tilde{T}$ apzīmē visas lapas kokā T.

Fiksētai vertībai α ir unikāls mazākais minimizējošs apakškoks $T(\alpha)$ no pilnā koka T_{max}, kam izpildās sekojoši nosacījumi:[5]

$$C_\alpha(T(\alpha)) = \min_{T \subseteq T_{max}} C_\alpha(T) \tag{3.2}$$

$$\text{Ja } C_\alpha(T) = C_\alpha(T(\alpha)), \text{ tad } T(\alpha) \subseteq T \tag{3.3}$$

Pirmais nosacījums apgalvo, ka nav tāda apakškoka no T_{max}, kam būtu mazākas izmaksas par $T(\alpha)$ konkrētajam α. Otrais nosacījums nodrošina, ka gadījumā, kad vairāk kā viens koks sasniedz minimumu, kļūdu sarežģītības apgriešanas algoritms izvēlas mazāko koku. Tā kā T_{max} ir galīgs, arī atšķirīgo koku $T(\alpha)$ no T_{max} skaits ir galīgs.

Kļūdas minimuma metode T.Nibletts un I.Bratko piedāvāja metodi, kas atrod tādu koku, kas minimizē sagaidāmo kļūdu.

Definīcija 24. k klašu problēmai, sagaidāmo varbūtību, ka novērojums, sasniedzot virsotni t, pieder $i-$tajai klasei definē sekojoši:[2]

$$p_i(t) = \frac{n_i(t) + p_{ai}m}{N(t) + m}, \tag{3.4}$$

kur $n_i(t)$ ir treniņa datu piemēru skaits virsotnē t, kas ir klasificēti kā $i-$tās klases elementi, m ir novērtējuma metodes parametrs un $N(t)$ ir treniņa datu piemēru skaits, kas ir sasnieguši virsotni t.

Kad jauns novērojums, sasniedzot virsotni t, tiek klasificēts, tiek aprēķināta sagaidāmā kļūda E:[2]

$$E(t) = \min_i (1 - p_i(t)). \tag{3.5}$$

Katrai iekšējai koka virsotnei tiek aprēķinātas divas kļūdas:

1. Statiskā kļūda- sagaidāmā kļūda, kad virsotne t tiek apgriezta.

2.Dinamiskā kļūda - svērta summa no 'bērnu' virsotņu kļūdām, kur svari ir varbūtības, ka novērojums sasniegs konkrēto 'bērna' virsotni.

Ja statiskā kļūda ir mazāka par dinamisko kļūdu, tad virsotne tiek apgriezta.

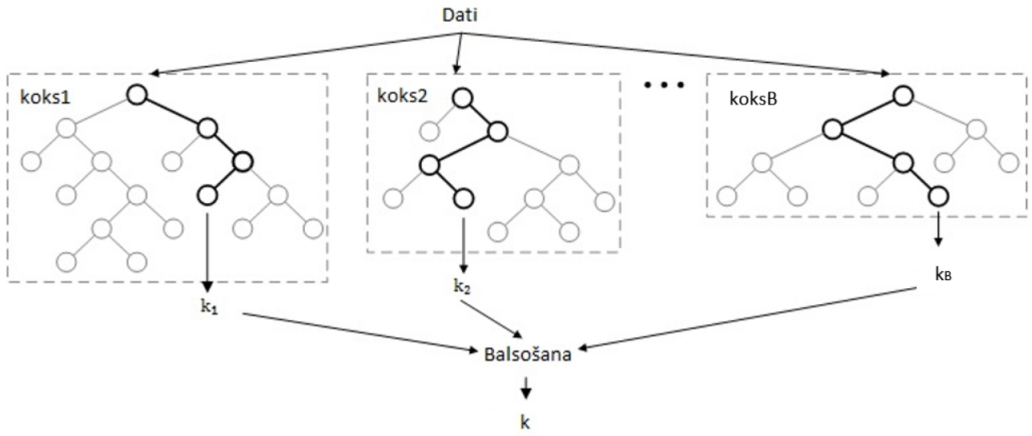
4 GADĪJUMA MEŽU ALGORITMS

Gadījuma mežu algoritmu izstrādāja Breimans 2001. gadā. Tas ir neparametrisks klasifikācijas algoritms.

Definīcija 25. Gadījuma mežs ir lēmumu koku ansamblis.

Gadījuma mežu algoritms veido lēmumu kokus, kas tiek nejauši ģenerēti nejauši izvēlētās datu apakškopās. Izveidotie lēmumu koki reprezentē noteikumus, kas ir veidojušies no modelēšanas. Algoritmā arī labākā pazīme, pēc kuras šķelt, dažādos lēmuma koku līmeņos tiek izvēlēta nejauši. Gadījuma koku algoritmu var izmantot gan klasifikācijā, gan prognozēšanā, gan mainīgo izvēlē, gan izlēcēju atklāšanai.

Modeļa konstruēšana Ar dotu treniņa datu kopu ar gadījumu skaitu N , treniņa datu izlase izmērā N tiek izvēlēta ar aizvietošanu no oriģinālās treniņu datu kopas. Izlases kopa satur ievades mainīgos vai pazīmes M. Ievades mainīgo skaits m paliek konstants koka būvēšanas procesā, kur $(m <= M)$ tāds, ka katrā virsotnē m mainīgie tiek nejauši izvēlēti un labākais šķēlums no šiem m tiek izvēlēts, lai šķeltu virsotni. Koki tiek audzēti šādā veidā (nejaušu pazīmju izvēle) līdz maksimālajam izmēram bez apgriešanas. Šis process tiek atkārtots vairākas reizes un procesa beigās tiek izveidota kolekcija no prognozētājiem. Nejaušības iekļaušana gadījuma mežu algoritmā nodrošina, ka vājām pazīmēm gadījuma mežos ir zema nobīde , zema korelācija un augsta dispersija, padarot tos par labiem prognozētajiem.



Att. 4.1: **Gadījuma mežu ilustrācija**

Ilustrācijā ir attēlots gadījuma mežs, kas sastāv no kopas ar lēmumu kokiem skaitā B $(koks_1, koks_2, \ldots, koks_B)$, kur katram kokam atbilst konkrēta klase $(k_1, k_2, \ldots, k_B)$. Kad katrā kokā ir izvēlēta klase, notiek balsošana un klase, kura ir izvēlēta lielākajā daļā koku, tiek piešķirta konkrētajam novērojumam.

Modeļa pielietošana prognozēšanā Lai klasificēt jaunu objektu, tam jādodas cauri katram kokam, kas atrodas mežā. Katrs koks klasificēs jauno objektu. Rezultātā tiek izvēlēta tā klase, kam ir visvairāk balsu no visiem kokiem mežā.

Pieaugot koku skaitam gadījuma mežā, testa kopas kļūdas konverģē uz noteiktu limitu. Tas nozīmē, ka arī lielos gadījuma mežos nav novērojama modeļa pārpielāgošana (*overfitting*). Gadījuma meži dod labus rezultātus retiem novērojumiem un datiem ar trūkstošām vērtībām.

Algoritma priekšrocība ir spēja tikt galā ar vairākiem faktoriem, kas ir nozīmīgi maksātspējas prognozēšanas procesā:

1. Aizdevumu pieprasījumu skaits nav vienmērīgs, kas noved pie nesabalansētiem datiem un samazina parametru skaitu, kas varētu tikt efektīvi novērtēti.

2. Liels pazīmju skaits mazam novērojumu skaitam. Šādos gadījumos loģistiskā regresija kļūst nestabila.

3. Dažādu pazīmju mijiedarbība ir bieži novērojama, piemēram, vecums un nodarbinātība. Viena no galvenajām gadījuma mežu algoritma priekšrocībām ir tā spēja efektīvi tikt galā ar dažādu pazīmju mijiedarbību.

5 LOĢISTISKĀS REGRESIJAS ALGORITMS

Logit transformācija Lai definētu modeli ir nepieciešama sistemātiska datu struktūra. Ir nepieciešams, lai varbūtības π_i ir atkarīgas no vektora ar novērojumu ietekmējošajiem faktoriem x_i. Ideja ir izteikt varbūtības kā lineāru funkciju no ietekmējošajiem faktoriem:

$$\pi_i = x_i'\beta, \tag{5.1}$$

kur β ir regresijas koeficientu vektors.

Problēma ar šo modeli ir tāda, ka varbūtības kreisajā pusē var pieņemt vērtību no 0 līdz 1, taču lineārā daļa labajā vienādojuma pusē var pieņemt jebkuru reālu vērtību. Tātad nav garantijas, ka paredzētās vērtības būs vēlamajā intervālā.

Vienkāršs risinājums ir transformēt varbūtību, lai noņemtu intervāla ierobežojumus un modelēt transformāciju kā lineāru funkciju no ietekmējošajiem faktoriem. Pirmkārt, varbūtība π_i tiek pārveidota kā attiecība, kuru apzīmē ar $odds$:

$$odds_i = \frac{\pi_i}{1-\pi_i}, \tag{5.2}$$

kas ir definēta kā attiecība labvēlīgiem pret nelabvēlīgiem notikumiem.

Otrkārt, attiecība tiek logaritmēta, iegūstot funkciju, kuru apzīmē ar $logit$:

$$logit(\pi_i) = log\frac{\pi_i}{1-\pi_i}, \tag{5.3}$$

kas noņem apakšējo robežu (ja varbūtība tiecas uz 0, $logit$ tiecas uz $-\infty$). Attiecīgi, ja varbūtība tiecas uz 1, $odds$ tiecas uz $+\infty$ un tāpat arī $logit$. Tātad $logit$ atzīmē varbūtības no intervāla $(0, 1)$ uz visas reālās taisnes.[9]

Loģistiskās regresijas pieņēmumi Tā kā $\frac{\pi_i}{1-\pi_i}$ pieņem vērtības no 0 līdz $+\infty$, $log\frac{\pi_i}{1-\pi_i}$ pieņem vērtības no $-\infty$ līdz $+\infty$. Loģistiskajā regresijā netiek ņemti vērā vairāki svarīgi lineārās regresijas pieņēmumi, tomēr jāņem vērā sekojoši nosacījumi:

- mērķa mainīgajam jābūt kategoriskam,
- jāpastāv lineārai sakarībai starp logaritma attiecību un neatkarīgajiem mainīgajiem ,
- neatkarīgas kļūdas, katram novērojumam ir jābūt neatkarīgam, neatkarīgajiem mainīgajiem jābūt neatkarīgiem vienam no otra,
- prediktoriem jābūt nekorelētiem,

- jāizmanto tikai nozīmīgi mainīgie, lai izmantotu visus nozīmīgos mainīgos, labs paņēmiens ir soļu metode.

Definīcija 26. Lai definētu loģistiskās regresijas modeli, pieņem, ka ir skaitā k neatkarīgi novērojumi $y_1, ..., y_k$ un ka i-tais novērojums var tikt uztverts kā realizācija no brīvi izvēlēta mainīgā Y_i, kur Y_i ir binomiāli sadalīts:

$$Y_i \sim Bin(n_i, \pi_i) \tag{5.4}$$

ar parametriem n_i - mēģinājumu skaits un π_i - varbūtība, ka notikums izpildīsies . Individuāliem datiem $n_i = 1$ visiem i. Pieņem, ka $logit$ no varbūtības π_i ir lineāra funkcija:

$$logit(\pi_i) = x_i'\beta, \tag{5.5}$$

kur β ir regresijas koeficientu vektors un x_i ir ietekmējošie faktori. Tad modelis, kas ir definēts ar diviem augstāk minētajiem vienādojumiem ir ģeneralizēts lineārs modelis ar binomiālu rezultātu un linka funkciju $logit$.

Eksponencējot augstāk redzamo vienādojumu, iegūst:

$$\frac{\pi_i}{1-\pi_i} = exp(x_i'\beta). \tag{5.6}$$

Šī izteiksme definē multiplikatīvu modeli. Piemēram, ja tiek mainīts j-tais prediktors par vienu vienību, kamēr pārējie nemainās, daļa tiktu reizināta ar $exp(\beta_j)$. Pieņemot, ka lineārs prediktors ir $x_i'\beta$ un x_i tiek palielināts par 1 , tas ir : $x_i'\beta + \beta_j$. Tātad eksponentētais koeficients $exp(\beta_j)$ reprezentē attiecību.[9] Uztvert rezultātus kā multiplikatīvos efektus uz daļu vai daļas attiecību, reizēm ir izdevīgi, jo tā ir pazīstamāka skala relatīvi vienkāršiem modeļiem.

5.1 Loģistiskās regresijas koeficientu novērtējums

Ticamības funkcija n neatkarīgiem binomināliem novērojumiem ir rezultāts no binomiālā sadalījuma varbūtību funkcijas: [1]

$$P(Y_i = y_i) = \binom{n_i}{y_i} \pi_i(1-\pi_i)^{n_i - y_i}. \tag{5.7}$$

Logaritmējot iegūst, ka izņemot konstantei, logaritmiskā ticamības funkcija ir :

$$logL(\beta) = \sum \Big(y_i log(\pi_i) + (n_i - y_i) log(1 - \pi_i) \Big), \tag{5.8}$$

kur π_i ir atkarīgs no ietekmējošajiem faktoriem x_i un vektora p parametriem β caur logaritmiskās funkcijas transformāciju vienādojumu :

$$logit(\pi_i) = x_i' \beta. \tag{5.9}$$

Tālāk tiek izmantota iteratīva svērto mazāko kvadrātu metode . Pēc dota pašreizēja novērtējuma $\widehat{\beta}$, tiek aprēķināts lineārs prognozējums $\widehat{\eta} = x_i \widehat{\beta}$ un vērtība $\widehat{\mu} = logit^{-1}(\eta)$. Ar šīm vērtībām aprēķina atkarīgo mainīgo z, kura elementi ir:

$$z_i = \widehat{\eta_i} + \frac{y_i - \widehat{\mu_i}}{\widehat{\mu_i}(n_i - \widehat{\mu_i})} n_i, \tag{5.10}$$

kur n_i ir binomiālā sadalījuma parametrs. Tad regresē z uz ietekmējošajiem mainīgajiem, aprēķinot svērto mazāko kvadrātu metodes novērtējumu:

$$\widehat{\beta} = (X'WX)^{-1} X'Wz, \tag{5.11}$$

kur W ir diagonāl matrica no svariem :

$$w_{ii} = \widehat{\mu_i} \frac{(n_i - \widehat{\mu_i})}{n_i}. \tag{5.12}$$

Svari ir inversi proporcionāli novērtētai dispersijai no atkarīgā mainīgā. Rezultējošais novērtējums β tiek lietots, lai iegūtu uzlabotu pielāgotās vērtības un procedūra tiek atkaŗtota līdz konverģencei. [12] Piemērotas sākuma vērtības var tikt iegūtas pielietojot savienojumu ar datiem. Lai novērstu problēmas ar 0 skaitu vai n_i , aprēķina empīrisko $logit$, pievienojot $\frac{1}{2}$ abiem, respektīvi, rēķina :

$$z_i = log \frac{y_i + \frac{1}{2}}{n_i - y_i + \frac{1}{2}}. \tag{5.13}$$

Tad iegūtais vienādojums tiek regresēts uz x_i, lai iegūtu sākuma novērtējumu β.

Rezultējošais novērtējums ir konsistents un tā lielas izlases dispersija ir uzdota ar formulu :

$$var(\widehat{\beta}) = (X'WX)^{-1}, \tag{5.14}$$

kur W ir svaru matrica, kas iegūta pēdējā iteracijā.

5.1.1 Loģistiskās regresijas koeficientu nozīmības pārbaude

Tiek pārbaudīta nulles hipotēze, ka j-tais regresijas koeficients ir vienāds ar 0 jeb nav nozīmīgs :

$$H_0 : \beta_j = 0. \tag{5.15}$$

Koeficienta nozīmība tiek pārbaudīta aprēķinot attiecību no koeficienta novērtējuma pret tā standartkļūdu:

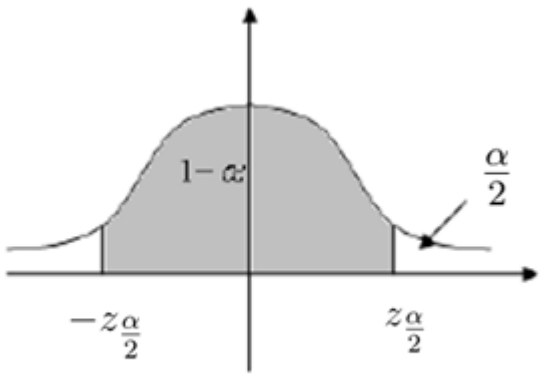
$$z = \frac{\widehat{\beta}_j}{\sqrt{\widehat{var}(\widehat{\beta}_j)}}. \tag{5.16}$$

Šai statistikai lielās izlasēs ir aptuveni standart normālais sadalījums. Alternatīvi kvadrātu no šīs statistikas var uztver kā aptuveni Hī-kvadrāta sadalījumu ar brīvības pakāpi 1.

Tests var tikt izmantots, lai aprēķinātu ticamības intervālu koeficientam β_j. Var pieņemt, ka ar $100(1-\alpha)\%$ procentu drošību īstais parametrs atrodas intervālā ar robežām:

$$\widehat{\beta}_j + -z_{1-\frac{\alpha}{2}}\sqrt{\widehat{var}(\widehat{\beta}_j)}, \tag{5.17}$$

kur $z_{1-\frac{\alpha}{2}}$ ir normālā sadalījuma kritiskā vērtība, divpusējam testam izmērā α, skatīt 5.1. attēlu [9] :



Att. 5.1: **Normālā sadalījuma kvantile**

Tests var tikt pielietots testa hipotēzēm vairākiem koeficientiem aprēķinot parasto kvad-

rātisko formu. Šis tests var tikt invertēts, lai noteiktu ticamības reģionus vektor-vērtību parametriem.

Vispārīgākām problēmām apskatīšu ticamības attiecibas testu. Atslēga šo testu konstruēšanā ir agrāk pieminētā D statistika. Ticamības attiecības tests salīdzina divus modeļus balstoties un starpību starp to attālumu.

5.2 Modeļa pielāgotības pārbaude

Kad modelis ir izveidots, jāpārbauda, cik labi tas atbilst dotajiem datiem. Mērs, kas apraksta atšķirību starp novērotajām un pielāgotajām vērtībām ir statistika, kuru apzīmē ar D un kas ir uzdota ar formulu:

$$D = 2\sum\left(y_i log\left(\frac{y_i}{\widehat{\mu_i}}\right) + (n_i - y_i) log\left(\frac{n_i - y_i}{n_i - \widehat{\mu_i}}\right)\right), \tag{5.18}$$

kur y_i ir novērojumi un $\widehat{\mu_i}$ ir pielāgotās vērtības i-tajam novērojumam. Perfektā gadījumā attiecība starp novērojumiem un sagaidāmajām vērtībām ir viens un tā logaritms ir nulle, rezultātā D ir nulle.[9]

Grupētiem datiem statistikas sadalījums, grupas apjomam palielinoties no n_i uz ∞ visiem i, konverģē uz Hī-kvadrāta sadalījumu ar $n - p$ brīvības pakāpēm, kur n ir grupu skaits un p ir parametru skaits modelī, iekļaujot konstanti. Tātad apjaušami lielām grupām, statistika nodrošina pielāgotības pārbaudi modelim. Individuāliem datiem statistikas sadalījums nekonverģē uz Hī-kvadrāta vai kādu citu zināmu sadalījumu un nevar tikt izmantots kā pielāgotības pārbaude.

Metodes, lai pārbaudītu, cik labi modelis apraksta datus un tos prognozē ir ļoti daudz. Darba ietvaros tika apskatītas vairākas metodes, taču galvenās, kuras beigās tika izmantotas ir ROC (*receiver operating characteristics*) līkne un Kolmogorova-Smirnova statistika.

5.3 Analogi R^2 mēri

McFadden 1973. gadā prezentēja analogu R^2, ko sauc arī par maksimālās ticamības daļas indeksu. Tas tiek aprēķināts salīdzinot modeli bez prediktoriem ar modeli, kurā iekļauti visi preditkori.

Definīcija 27. McFadden R^2 analogs R^2_{MF} ir mērs, kas tiek definēts sekojoši:[9]

$$R^2_{MF} = 1 - \left(\frac{\widehat{L}_1}{\widehat{L}_0}\right), \tag{5.19}$$

kur $\widehat{L}_0$ ir maksimizēta maksimālā ticamība modelim bez preditoriemun $\widehat{L}_1$ ir maksimizēta maksimālā ticamība modelim ar visiem preditoriem.

1983. gadā Maddala izstrādāja citu pieeju R^2 analoga aprēķināšanai, kas var tikt pielietots katram modelim, kas novērtēts ar maksimālās ticamības metodi.

Definīcija 28. Maddala R^2 analogs R^2_M ir mērs, kas uzdots ar formulu:[9]

$$R^2_M = 1 - \left(\frac{\widehat{L}_0}{\widehat{L}_1}\right)^{\frac{2}{n}}, \tag{5.20}$$

kur $\widehat{L}_0$ ir maksimizēta maksimālā ticamība modelim bez preditoriem, $\widehat{L}_1$ ir maksimizēta maksimālā ticamība modelim ar visiem preditoriem

6 MODEĻU PRIEKŠROCĪBAS UN TRŪKUMI

Loģistiskā regresija Nozīmīgākais loģistiskās regresijas trūkums ir ietilpīgie aprēķini, it īpaši situācijās, kurās modelis ir jāuzlabo neskaitāmas reizes kosmētisku izmaiņu dēļ. Tomēr mūsdienās šo vairs nevar atzīt par trūkumu, jo aprēķinus var veikt ar programmu palīdzību datorā. Būtiskākās loģistiskās regresijas priekšrocības:

- Tā ir veidota, lai apstrādātu bināru iznākumu.
- Tā sniedz varbūtības novērojumiem, tādejādi ļaujot iegūt novērtējumu un izvēlēties precīzu robežu, līdz kurai klientiem izsniegt aizdevumu.
- Galējā varbūtība atrodas robežās no 0 līdz 1.
- Tā sniedz robustu novērtējumu īstajai varbūtībai no pieejamās informācijas.

Loģistiskās regresijas galvenie trūkumi:

- Rezultāti kļūst neprecīzi, ja pazīmju skaits ir liels.
- Modeļa prognozes precizitāte samazinās, kad datos ir daudz trūkstošu vērtību.
- Pārāk daudz kategorisku mainīgo arī samazina loģistiskās regresijas efektivitāti.

Lēmumu koku algoritms Lēmumu koku algoritmu ir vienkārši pielietot un rezultātus ir viegli interpretēt. Tiešā veidā netiek sniegts varbūtības novērtējums kā loģistiskajā regresijā, taču tā vietā var izmantot mērķa pazīmes klašu varbūtības katrā virsotnē. Lēmumu koku galvenās priekšrocības:

- Viegli interpretēt iegūtos rezultātus kā noteikumus, kuriem ir jāizpildās, lai klients saņemtu aizdevumu.
- Rezultātus var attēlot grafiski, kas ļauj viegli prezentēt rezultātus citiem un ir izdevīgi no biznesa viedokļa.
- Metode ir robusta pret izlēcējiem. Tā izlēcējus nodala atsevišķās virsotnēs, tādejādi neietekmējot rezultātus.
- Kad klasifikācijas noteikumi vienreiz ir izveidoti, klasifikācija noris ļoti ātri.
- Tā kā metode ir neparametriska, nav formālu pieņēmumu un ir viegli iekļaut virkni skaitliskus vai kategoriskus datus un nav nepieciešams izvēlēties vienveidīgus treniņa datus.

Lēmumu koki tiek trenēti uz treniņa datiem. Tādēļ prognozes uz treniņa datiem var tikt veiktas ļoti precīzi, bet testa datu prognozes precizitāte var būt ļoti zema, ja testa datos parādās jaunas mainīgo kombinācijas, grupas vai vērtības. Ievērojamākie lēmumu koku trūkumi:

- Ļoti ietekmējas no treniņa datiem, kā rezultātā notiek modeļa pārpielāgošana.
- Lēmumu koku algoritms parasti nesniedz tik precīzus rezultātus kā citi algoritmi.
- Nestabilitāte - nedaudz mainot datus, koka izskats var kardināli mainīties.

Samazināt lēmumu koku pārpielāgošanu un uzlabotu koku prognozēšanas precizitāti var, izmantojot koku apgriešanu vai metodes, kas audzē vairākus kokus un tiek izvēlēts viens rezultāts. Kombinējot lielu koku apjomu, prognozējuma precizitāte tiek uzlabota, tomēr interpretācija ir sarežģītāka.

Gadījuma mežu algoritms Gadījuma mežu lielākais trūkums ir sarežģīta rezultātu interpretācija. Gadījuma mežu algoritmam, līdzīgi kā lēmumu koku algoritmam nav daudz izteiktu trūkumu. Viens no ievērojamākajiem mīnusiem : ja datos ir vairāki kategoriski mainīgie ar dažādu skaitu līmeņiem, tad gadījuma mežs ir nobīdīts uz mainīgajiem ar lielāku skaitu līmeņu. Pie tam mainīgo nozīmības novērtējums nav uzticams šādiem datiem. Salīdzinot ar lēmuma koku algoritmu, situācijā, kad dati tiek nedaudz izmainīti, individuāli koki var izmainīties, taču mežs ir stabils, jo to veido kombinācija no daudziem kokiem. Gadījuma mežu algoritmam parasti ir ļoti augsta prognozēšanas precizitāte. Nozīmīgākie ieguvumi, izmantojot gadījuma mežu algoritmu:

- Līdzīgi kā lēmumu koku algoritms, arī gadījuma mežu algoritms ir neparametriska metode, tātad algoritmam nav formālu pieņēmumu.
- Nav problēmu ar pārpielāgošanu.
- Nav nepieciešama koku apgriešana.
- Spēj strādāt ar datiem bez pirmsapstrādes.
- Efektīvi novērtē trūkstošus datus un saglabā precizitāti, kad iztrūkst liela daļa datu.
- Automātiski aprēķina precizitāti un novērtējumu, kuri mainīgie ir nozīmīgi klasifikācijas procesā.
- Modeļa veidošanas procesā jau tiek ietverta arī mainīgo savstarpējā mijiedarbība.
- Spēj ātri izveidot modeli un automātiski atlasīt mainīgos, ja mainīgo skaits ir ļoti liels.

6.1 Pārpielāgošana

Labam modelim jābūt zemai treniņa datu kļūdai un arī zemai ģeneralizētajai kļūdai. Kā teicis Occam's Razor: "Visam jābūt cik vien vienkārši ir iespējams, bet ne vienkāršāk." Pārpielāgošana notiek, kad algoritms turpina attīstīt hipotēzi, kas reducē treniņa datu kļūdu uz testa datu kļūdas palielināšanas rēķina. Tas rada nepatiesu optimismu par izveidotā modeļa precizitāti.

Ar h apzīmē hipotēzi. Treniņa datu kļūda ir $error_{train}(h)$ un kļūda sadalījuma D datiem ir $error_D(h)$. Hipotēze h pārpielāgo treniņa datus, ja eksistē tāda alternatīva hipotēze h', ka: [4]

$$error_{train}(h) < error_{train}(h'), \tag{6.1}$$

$$error_D(h) > error_D(h'). \tag{6.2}$$

Citiem vārdiem treniņa dati tiek pārpielāgoti, ja eksistē cita alternatīva ar lielāku treniņa datu kļūdu un mazāku kopējo datu kļūdu. Pārpielāgošanas iemesli:

- Neklasificētas pazīmes var apstrīdēt klases līdzīgiem ierakstiem.
- Modelis ir pārāk sarežģīts (pārāk daudz mainīgo).
- Reprezentatīvu pazīmju trūkums treniņa datos var samazināt algoritma precizitāti.
- Vairākkārtēja salīdzināšanas procedūra. Algoritmi, kas apskata daudz alternatīvas var novest pie neīsta rezultāta.

Lēmuma kokiem ir tendence modeli pārpielāgot, kad lēmumu koks ir dziļāks. Lēmumu koku pārpielāgošanu var novērst divos veidos:

- Pārtraukt koka audzēšanu, kad datu šķelšana nav statistiski nozīmīga.
- Izaudzēt pilnu koku un tad apgriezt.

Lai samazinātu lēmuma koku modeļa pārpielāgotību klientu maksātspējas prognozēšanai, tika veikta lēmumu koka apgriešana. Lai uzlabotu gan loģistiskās regresijas modeli, gan lēmumu koku modeli tika samazināts mainīgo skaits, palielināta treniņu datu kopa un attīrīti dati.

Kā jau minēts iepriekš, viens no lielākajiem ieguvumiem, lietojot gadījuma mežu algoritmu, ir tas, ka modelim nav problēmu ar pārpielāgošanu.

Nodaļas kopsavilkums Katrai metodei ir savas stiprās un vājās puses. Lai prognozētu klientu maksātspēju, ir ļoti svarīgi spēt saprotami interpretēt un paskaidrot rezultātus. Rezultātu interpretācija ir nepieciešama no biznesa viedokļa puses. Lēmuma pieņemšanas procesā datos var būt tukšumi vai neatbilstoši mainīgie, kā arī liels skaits mainīgo. Tātad galvenās iezīmes, kurām ir jāpiemīt izveidotajam modelim: ātrums, precizitāte, vienkāršība, viegli interpretējami rezultāti, spēja tikt galā ar nestandarta mainīgajiem. Aplūkojot modeļu priekšrocībām un trūkumiem, redzams, ka katram modelim piemīt kāda no šīm iezīmēm. Lai spētu izvēlēties piemērotāko modeli klientu maksātspējas prognozēšanai, nozīmīga ir salīdzināt modeļu prognozēšanas precizitāti.

7 MODEĻU PROGNOZĒŠANAS PRECIZITĀTES SALĪDZINĀJUMA MĒRI

Modeļa precizitāti raksturo tas, kādu skaitu no testa datu vērtībām modelis prognozē pareizi vai nepareizi. Šos datus var apkopot tabulā:

Tabula 7.1: **Datu vērtības pret prognozētajām**

	Prognozētais iznākums	
Patiesais iznākums	f_{01}	f_{11}
	f_{10}	f_{11}

Tabulā ir attēlota bināras klasifikācijas problēmas matrica. Katrs elements f_{ij} apzīmē vērtību skaitu, cik novērojumi no klases i tika prognozēti kā piederoši klasei j. Piemēram, elements f_{01} apzīmē elementu skaitu, kam patiesā klase ir 0 un modeļa prognozētā klase ir 1.

Tātad ir četri dažādi iespējamie rezultāti:

- patiesā vērtība ir labvēlīga un prognozētais rezultāts ir patiess
- patiesā vērtība ir labvēlīga un prognozētais rezultāts nav patiess
- patiesā vērtība ir nelabvēlīga un prognozētais rezultāts ir patiess
- patiesā vērtība ir nelabvēlīga un prognozētais rezultāts nav patiess

Matricā ir apkopota nepieciešamā informācija, lai noteiktu, cik precīzi klasifikācijas modelis prognozē nezināmas vērtības (klases). Lai vieglāk definētu precizitātes mērus, ar n apzīmē kopējo novērojumu skaitu, ar PL apzīmē patiesi prognozētās labvēlīgas vērtības, ar PN apzīmē patiesi prognozētos nelabvēlīgos iznākumus, ar NL apzīmē nepatiesi prognozētās labvēlīgās vērtības un ar NN apzīmē nepatiesi prognozētās nelabvēlīgās vērtības.

Definīcija 29. Par patieso pozitīvo rezultātu kvantitāti jeb precizitāti TP sauc mēru, kas uzdots ar formulu:

$$TP = \frac{PL}{\textit{kopējais labvēlīgo iznākumu skaits}}. \tag{7.1}$$

Definīcija 30. Par nelabvēlīgo nepareizi prognozēto rezultātu kvantitāti jeb kļūdas mēru FP sauc mēru, kas uzdots ar sekojošu formulu :

$$FP = \frac{NN}{\textit{kopējais nelabvēlīgo iznākumu skaits}}. \tag{7.2}$$

Tiek meklēts modelis ar visaugstāko precizitāti vai ekvivalenti zemāko kļūdas mēru.

Definīcija 31. Mērs C, kas apzīmē patiesi prognozēto novērojumu proporciju abās klasēs kopā tiek uzdots ar formulu: [4]

$$C = \frac{PL + PN}{n} \tag{7.3}$$

Definīcija 32. Sensitivitāte Sn ir mērs, kas mēra proporciju no pozitīviem iznākumiem, kas ir pareizi prognozēti: [4]

$$Sn = \frac{PL}{PL + NN} \tag{7.4}$$

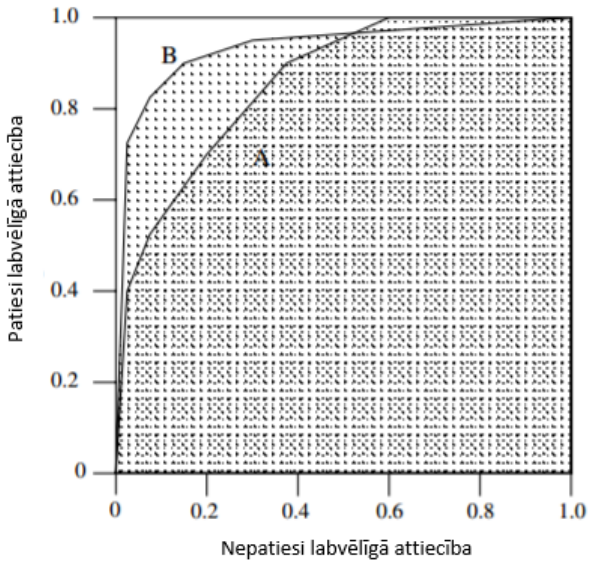
Definīcija 33. Specifika Sp ir mērs, kas mēra proporciju no negatīviem iznākumiem, kas ir pareizi identificēti: [4]

$$Sp = \frac{NN}{NN + NL} \tag{7.5}$$

Definīcija 34.ROC līkne (*Receiver Operating Characteristic curve*) tiek izmantota, lai grafiski attēlotu patieso mainīgā klasi salīdzinot ar prognozēto. Grafikā uz horizontālās ass tiek atzīmētas nepareizi prognozētās nelabvēlīgās vērtības attiecība un uz vertikālās ass - patieso pozitīvo rezultātu kvantitāte. Abi mainīgie pieņem vērtības no 0 līdz 1. [4]

Lielākā nozīme šajā grafikā ir laukumam, kas atrodas zem ROC līknes, to apzīmē ar AUC(*Area Under Curve*). Tas pieņem vērtības no 0 līdz 1.

Definīcija 35. AUC modelim ir ekvivalents varbūtībai, ka modelis augstāk vērtēs nejauši izvēlētu labvēlīgu gadījumu nekā nejauši izvēlētu negatīvu gadījumu. Skatīt piemēru 7.1 attēlā.



Att. 7.1: AUC piemērs

Attēlā modelis B ierobežo lielāku laukuma daļu nekā modelis A un līdz ar to no šī skatu punkta modelis B ir labāks par modeli A.

8 MODEĻU PROGNOZĒŠANAS PRECIZITĀTES SALĪDZINĀJUMS

Dati Modeļi tika veidoti no datiem, kas satur 1790 negatīvus iznākumus (klients neatmaksāja aizdevumu laikā), 15197 pozitīva iznākuma (klients atmaksāja laikā) un 1457 gadījumos vēl nav zināms iznākums. Šie dati jau ir apstrādāti - ir atsijāti pavisam nederīgi mainīgie, atlikušie ir sagrupēti. Tad no datiem tika izvēlēta izlase, kas satur 15881 novērojumus, kur ir 14141 (89%) pozitīvs un 1740 (11%) negatīvi iznākumi. Tad izlase tiek sadalīta treniņa un testa datos. Treniņa datu izlase sastāv no 70% (11116) nejauši izvēlētiem novērojumiem un atlikušie 30% (4765) novērojumi veido testa datu izlasi. Attiecīgi treniņa datos ir 9898 (89%) labvēlīgi gadījumi un 1218 (11%) nelabvēlīgi. Savukārt testa datos ir 4243 (89%) labvēlīgi un 522 (11%) nelabvēlīgi gadījumi. Datu pirmsapstrādē tiek apskatīta mainīgo savstarpējā korelācija, to nozīmīgums, informācijas vērtība, izskaidrojamība, stabilitāte un pamatotība. Tad attiecīgi pēc minētajiem kritērijiem tiek izvēlēti spēcīgākie mainīgie. Darbā visi modeļi ir veidoti, izmantojot vienu datu kopu, kā arī testa un treniņa datu izlases.

ID3 algoritma implementācija Darba ietvarā izveidoju programmas kodu valodā R ID3 algoritma implementācijai (skatīt 1.pielikumu). Programmā RStudio ir iebūvēta paciņa *data.tree*, kurā ir funkcija *Node*, ar kuru var izveidot saknes virsotni. Lai implementētu algoritmu, izveidoju funkciju, kas pārbauda, vai dati ir tīri - vai kopā ir tikai vienai klasei piederoši dati. Implementēju arī entropijas un informācijas ieguvuma formulas. Tad paša algoritma implementācijas formula atbilst iepriekš aprakstītajiem algoritma soļiem - tiek pārbaudīta kopas tīrība, ja kopa ir tīra, tad tiek veidota lapa. Situācijā, kad kopa nav tīra, izvēlas mainīgo ar augstāko informācijas ieguvumu, attiecīgi šķeļ virsotni un rekursīvi atkārto darbības līdz visi dati ir klasificēti.

Sākotnējais informācijas ieguvums mainīgajiem apskatāms tabulā 8.1.

Tabula 8.1: **Faktoru informācijas ieguvumi pirmajā algoritma solī**

Nr	Faktors	Informācijas ieguvums
1	faktors1	0.02499
2	faktors2	0.01686
3	faktors3	0.11498
4	faktors4	0.00561
5	faktors5	0.02201
6	faktors6	0.09014
7	faktors7	0.20484
8	faktors8	0.13960
9	faktors9	0.03734

Pirmajā tabulā ir redzams, ka augstākais informācijas ieguvums ir septītajam faktoram. Tad pirmajā solī šis faktors veido virsotni un pēc šī faktora tiek šķelta datu kopa. Pēc tam attiecīgi atkal tiek veikta pārbaude jaunizveidotajās virsotnēs, vai datu kopas ir tīras. Ja ir, tad tā ir lapa, ja nav, tad tiek aprēķināts informācijas ieguvums un tiek izvēlēts nākamais faktors ar lielāko informācijas ieguvumu. Tā tas turpinās līdz visi novērojumi ir klasificēti.

C5.0 algoritma implementācija 2015. gadā valodā R tika izveidota paciņa $C50$, kurā pieejamās funkcijas ļauj implementēt C5.0 algoritmu valodā R. R valodas kodu var apskatīt 2.pielikumā. Tika izmantota iebūvētā funkcija $C5.0()$, kura izveido gadījuma koku. Izmantojot funkciju $summary()$, iegūst, ka modeļa kļūda ir 11.0%. Tomēr prognozējot iznākumu ar testa datiem, tiek iegūts, ka visi novērojumi ir labvēlīgi jeb 100% klientu atmaksās aizdevumu. Problēma ir tajā, ka datos 89% no novērojumiem ir labvēlīgi (klients atmaksās aizdevum).

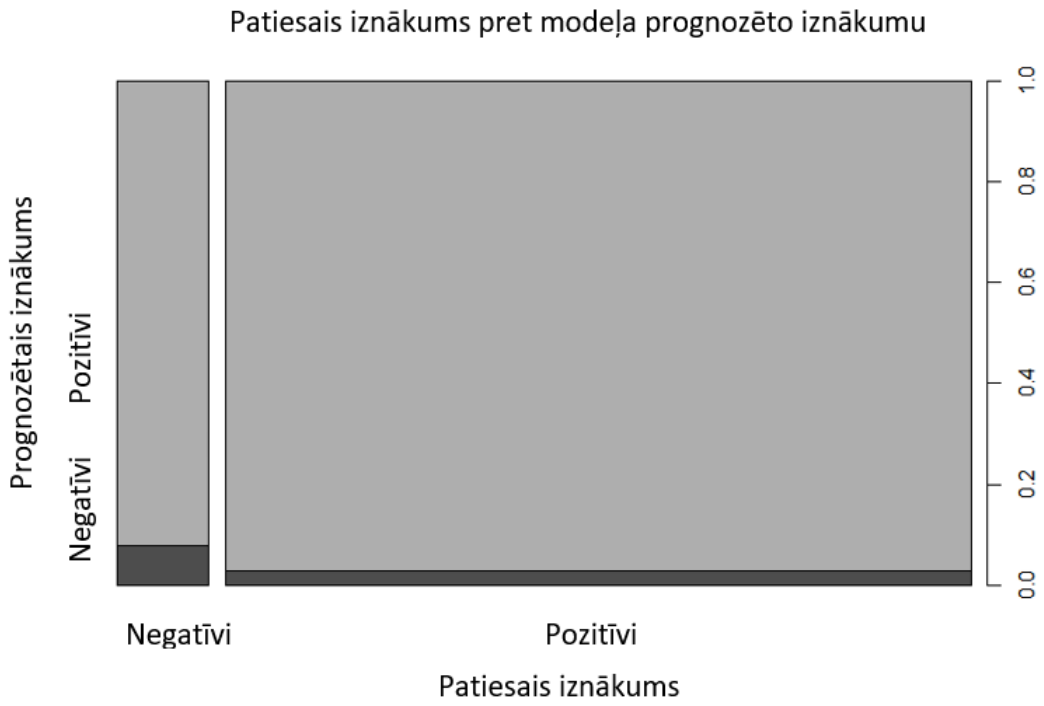
CART algoritma implementācija Programmā RStudio ir iebūvēta paciņa $rpart$, kas ir izveidota 2017. gadā un satur funkcijas rekursīvai datu dalīšanai klasifikācijas nolūkos, kā arī regresijas un izdzīvošanas kokiem. Izmantojot iebūvētās funkcijas izveidoju modeli klientu maksātspējas prognozēšanai. Lēmumu koku izveido ar funkciju $rpart()$.

Lēmumu koks tika veidots uz treniņa datiem un pēc tam tika prognozēts testa datu novērojumu iznākums. 8.2 tabulā ir redzamas patiesās testa datu vērtības pret prognozētajām:

Tabula 8.2: **Patiesās testa datu vērtības pret prognozētajām**

		Prognozētais iznākums	
		Pozitīvi	Negatīvi
Patiesais iznākums	Pozitīvi	4120	123
	Negatīvi	482	40

Grafiskā interpretācija 8.2.tabulai redzama 8.1 attēlā:



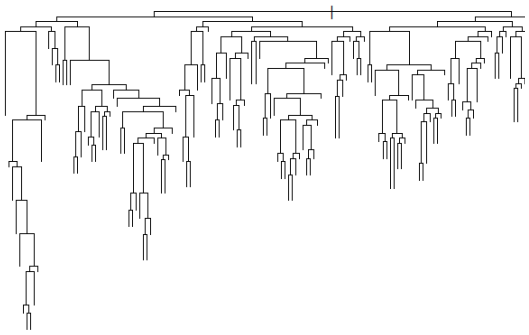
Att. 8.1: **Patiesās testa datu vērtības pret prognozētajām**

Attiecīgi precizitāti var aprēķināt dalot patiesi prognozētās vērtības pret novērojumu skaitu. Iegūst, ka šī modeļa precizitāte ir 83%.

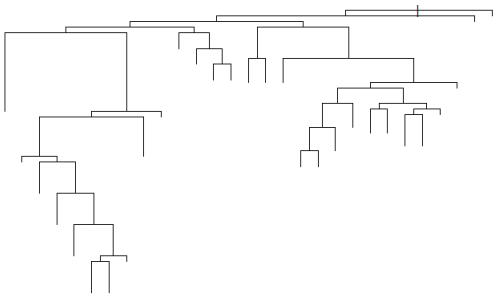
Lēmuma koka apgriešana Izveidojot klasifikācijas un regresijas koku ar funkciju $rpart()$, iegūtajam kokam var attēlot sarežģītības parametrus ar funkciju $plotcp()$, kā arī izvadīt tabulas veidā ar funkciju $printcp()$. Iebūvētā funkcija $prune()$ apgriež koku attiecīgi līdz norādītam sarežģītības parametram. Tātad sekojoša rindiņa :

$$prune(koks, cp = koks\$cptable[which.min(koks\$cptable[, "xerror"]), "CP"]) \qquad (8.1)$$

veic koka apgriešanu pēc mazākā kļūdas sarežģītības parametra. Attēlā redzama lēmumu koka struktūra pirms un pēc tā apgriešanas :



Att. 8.2: **Lēmuma koka struktūra pirms apgriešanas**



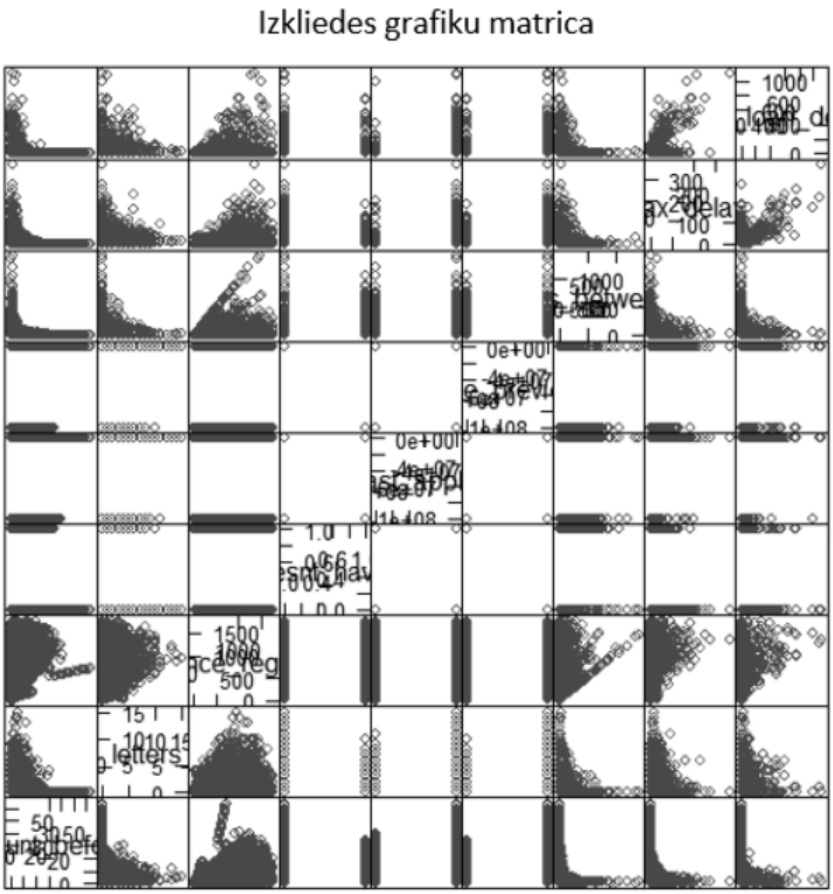
Att. 8.3: **Apgrieztā koka struktūra**

Sarežģītības parametrs cp tiek izmantots, lai kontrolētu lēmuma koka izmēru un izvēlētos optimāla izmēra koku. Ja izmaksas par vēl viena mainīgā pievienošanu kokam no pašreizējās virsotnes ir virs cp vērtības, tad koka būvēšana vairs neturpinās.

Piešķirot vērtību 0, tiks veidots koks līdz tā maksimālajam dziļumam, kas visticamāk ir ļoti, ļoti liels koks. Tas ir noderīgi, ja ir nepieciešams apskatīt cp vērtības dažādiem koka izmēriem.

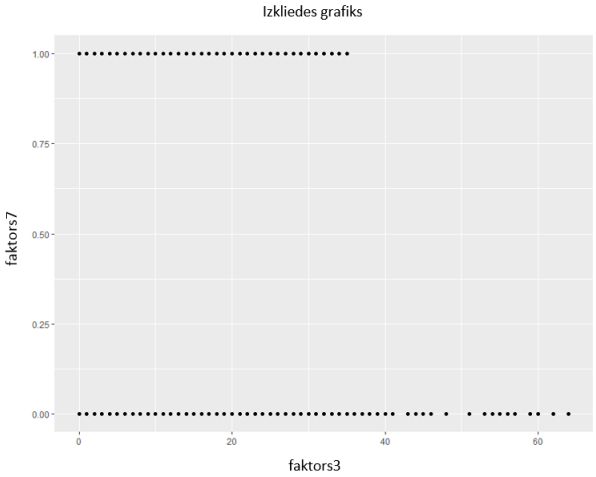
Aplūkojot grafiski sākotnējo koku un apgriezto, atšķirība ir ievērojama. Apgrieztā koka rezultātus ir vieglāk interpretēt. Līdzīgi kā iepriekš tiek aprēķināta precizitāte, dalot patiesi prognozētās vērtības ar novērojumu skaitu. Apgrieztā koka precizitāte ir 89% (pirms lēmuma koka apriešanas tā precizitāte bija 83%).

Gadījuma mežu algoritms Valodā R tika izveidots modelis, balstoties uz gadījuma mežu algoritmu, izmantojot paciņu $randomForest$. Izvēlētajā izlasē ir 453 faktori un 17338 novērojumu, no kuriem $14141(82\%)$ ir labvēlīgs iznākums un $1740(10\%)$ nelabvēlīgs. Tika veikta pārbaude, kurās kolonnās ir trūkstoši dati un kurās ir faktora mainīgie ar vairāk kā 100 faktoriem. Pēc datu izpētes, tika attēloti grafiski pāris daudzsološākie mainīgie, izmantojot paciņu $ggplot2$ un $caret$. Attēlā redzama izkliedes grafiku matrica, kurā mainīgie attēloti pa pāriem.



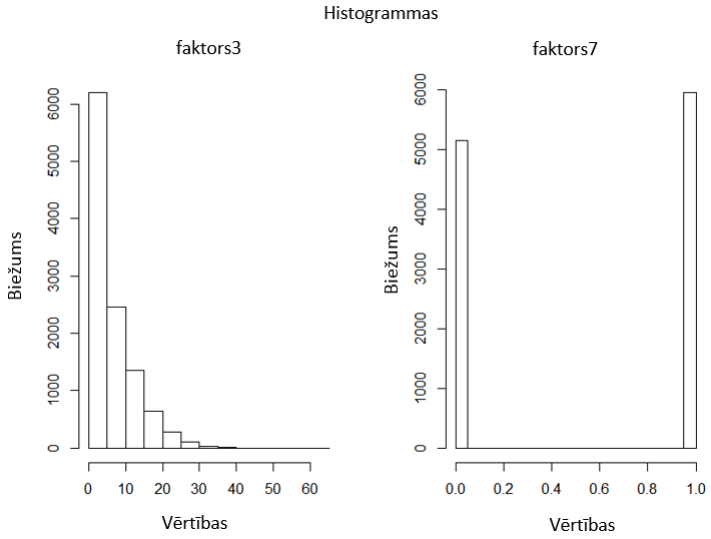
Att. 8.4: **Mainīgo izkliedes grafiku matrica**

Lai precīzāk apskatītu mainīgos, katram pārim tika izveidots atsevišķs izkliedes grafiks. Piemērs:



Att. 8.5: **Divu faktoru izkliedes grafiks**

Lai saprastu mainīgo grupēšanu, izveidoju mainīgo histogrammas:

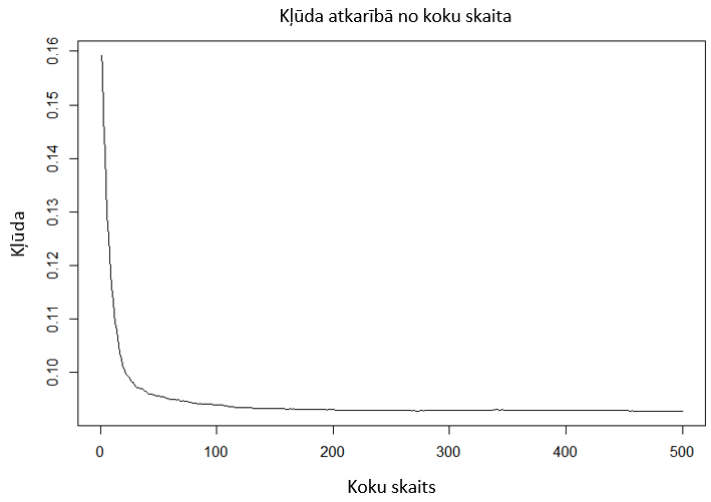


Att. 8.6: **Divu faktoru histogrammas**

Grafikos ir redzams, ka septītajam faktoram ir tikai divas vērtības.

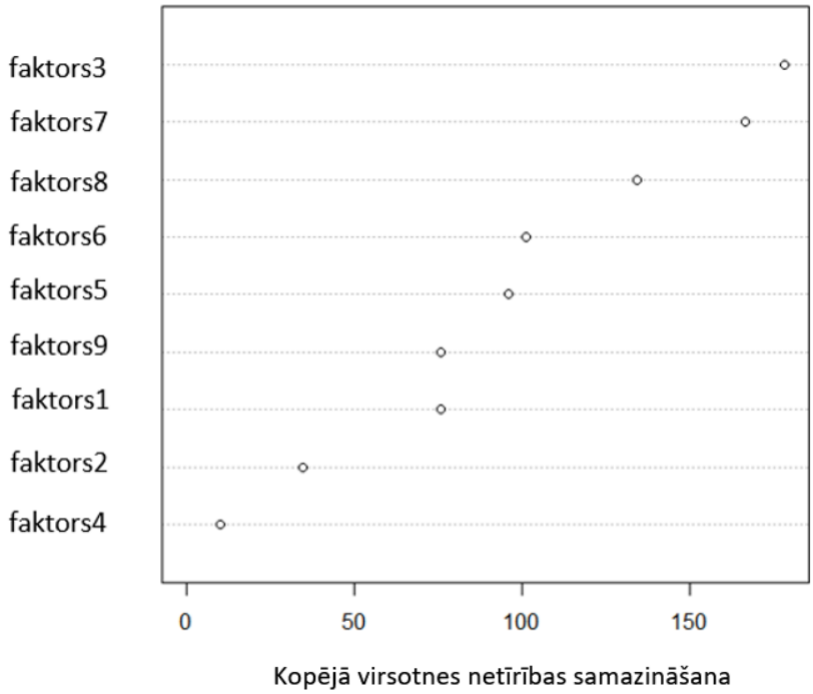
Pēc tam izveido un uzzīmē gadījuma mežu no treniņa datiem, izmantojot iebūvēto funkciju randomForest ar 500 lēmumu kokiem. Kļūdas novērtējums ir 11.35%.

Attēlojot gadījuma mežu, redzams, ka pie aptuveni 100 lēmumu kokiem vairs nav nozīmīgs kļūdas samazinājums.



Att. 8.7: **Kļūda atkarībā no koku skaita**

Izmantojot iebūvēo funkciju $varImpPlot()$, grafiski tiek attēlota mainīgo nozīmība:



Att. 8.8: **Mainīgo nozīmības grafiks**

Katra faktora nozīmību var attēlot arī tabulas veidā:

Tabula 8.3: **Faktoru virsotnes netīrības samazināšana**

Nr	Faktors	Kopējā virsotnes netīrības samazināšana
1	faktors1	75.85
2	faktors2	34.82
3	faktors3	178.57
4	faktors4	10.01
5	faktors5	96.22
6	faktors6	101.12
7	faktors7	166.79
8	faktors8	134.24
9	faktors9	75.94

Kopējā virsotnes netīrības samazināšana tiek mērīta pēc Gini indeksa no mainīgā šķelšanas, vidēji visos kokos kopumā.

Pēc modeļa izveides un izpētes, tiek veikta prognozēšana uz atlikušajiem testa datiem. Tabulā redzams gadījuma meža algoritma prognozētais iznākums pret patieso:

Tātad pēc augstāk izveidotajiem apzīmējumiem, gadījuma mežiem $NN = 509$, $NL = 27$, $PN = 13$ un $PL = 4216$.

Salīdzinot testa datu patieso vērtību ar prognozēto, iegūst skaitu, cik daudziem novērojumiem prognoze ir bijusi pareiza. Attiecīgi, izdalot šo skaitu ar kopējo novērojumu skaitu, iegūst,

Tabula 8.4: **Gadījuma mežu algoritma patiesās testa datu vērtības pret prognozētajām**

		Prognozētais iznākums	
		Pozitīvi	Negatīvi
Patiesais iznākums	Pozitīvi	4216	27
	Negatīvi	509	13

ka 89% tika prognozēti pareizi.

Loģistiskās regresijas modeļa izveidošana Loģistiskās regresijas modeļi tika veidoti programmā R ar iebūvēto funkciju $glm()$. Šī formula ir izskatā:

$$glm(formula, sadaljums, linkafunkcija), \tag{8.2}$$

kur $formula$ - loģistiskās regresijas funkcija kā ietekmējošo faktoru summa, $sadaljums$ - binomiālais sadalījums, $linkafunkcija$ - logit funkcija (automātiski tiek izvēlēta, ja tiek noradīts binomiālais sadalījums) . Protams, ka šajā funkcijā ir iespējams izvēlēties vēl daudzus papildus nosacījums, taču šoreiz tie nav nepieciešami. Lai apskatītu procesu, kā no datiem tika iegūta izvēlēto ietekmējošo faktoru formula un pielietota funkcija $glm()$, skatīt 3.pielikumu.

Loģistiskās regresijas koeficientu nozīmības pārbaude Svarīgs nosacījums veidojot loģistiskās regresijas modeli: ir jāizmanto tikai nozīmīgi mainīgie. Lai pārbaudītu, vai visi izvēlētie ietekmējošie faktori ir nozīmīgi, tika pārbaudīta hipotēze, ka konkrēts mainīgā regresijas koeficients ir vienāds ar 0. Respektīvi, ja koeficients ir vienāds ar 0, tad mainīgais šajā modelī neko neietekmē un nav nepieciešams. Tas tiek atkārtots visiem mainīgajiem. Arī šī procedūra programmā R aizņem tikai vienu rindiņu, skatīt 4.pielikumu. Lai veiktu šo hipotēzi tiek izmantota komanda $summary()$, kur tiek izvadīts kopsavilkums iepriekš izveidotajam modelim. Rezultāti redzami 8.9 attēlā:

```
Call:
glm(formula = frm, family = "binomial", data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0163  -0.7865  -0.5370   0.7294   2.7215

Coefficients:
                                        Estimate Std. Error z value Pr(>|z|)
(Intercept)                              -0.7890     0.1187  -6.646 3.01e-11 ***
  V1                                     -1.1080     0.2406  -4.604 4.14e-06 ***
  V2                                     -0.5180     0.2734  -1.895 0.058077 .
  V3                                     -1.0530     0.4436  -2.374 0.017617 *
  V4                                     -0.8807     0.2954  -2.982 0.002867 **
  V5                                     -0.9266     0.2931  -3.161 0.001573 **
  V6                                     -1.4546     0.3934  -3.698 0.000218 ***
  V7                                     -0.9741     0.2737  -3.559 0.000372 ***
  V8                                     -1.2072     0.3531  -3.419 0.000629 ***

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 531.98  on 441  degrees of freedom
Residual deviance: 441.91  on 433  degrees of freedom
AIC: 459.91

Number of Fisher Scoring iterations: 5
```

Att. 8.9: **Koeficientu nozīmības līmeņi**

, kur ir redzams, ka otrais mainīgais ir nenozīmīgs, jo tā koeficienta nozīmības līmenis ir 0. Tad attiecīgi no modeļa tiek izslēgts attiecīgais mainīgais. Pēc mainīgā izslēgšanas atkal pārbaudām pārējo koeficientu nozīmību, skatīt 8.10 attēlu :

```
Call:
glm(formula = frm, family = "binomial", data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0458  -0.7845  -0.5365   0.8129   2.5864

Coefficients:
                                        Estimate Std. Error z value Pr(>|z|)
(Intercept)                              -0.8229     0.1173  -7.016 2.29e-12 ***
 V1                                      -1.1225     0.2395  -4.687 2.77e-06 ***
 V3                                      -1.0436     0.4440  -2.351 0.018748 *
 V4                                      -0.8728     0.2937  -2.971 0.002964 **
 V5                                      -0.8591     0.2862  -3.002 0.002685 **
 V6                                      -1.4514     0.3870  -3.751 0.000176 ***
 V7                                      -1.0050     0.2729  -3.682 0.000231 ***
 V8                                      -1.2261     0.3541  -3.463 0.000534 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 531.98  on 441  degrees of freedom
Residual deviance: 445.43  on 434  degrees of freedom
AIC: 461.43

Number of Fisher Scoring iterations: 5
```

Att. 8.10: **Koeficientu nozīmības līmeņi pēc viena mainīgā izslēgšanas**

Šajā brīdī situācija ir uzlabojusies un šāda mainīgo kombinācija tiek izvēlēta modeļa izveidošanai .

Modeļu precizitātes parametru salīdzinājums Tabulā 8.5 ir attēlotas mēru vērtības, kuri tika aprakstīti iepriekšējā nodaļā.

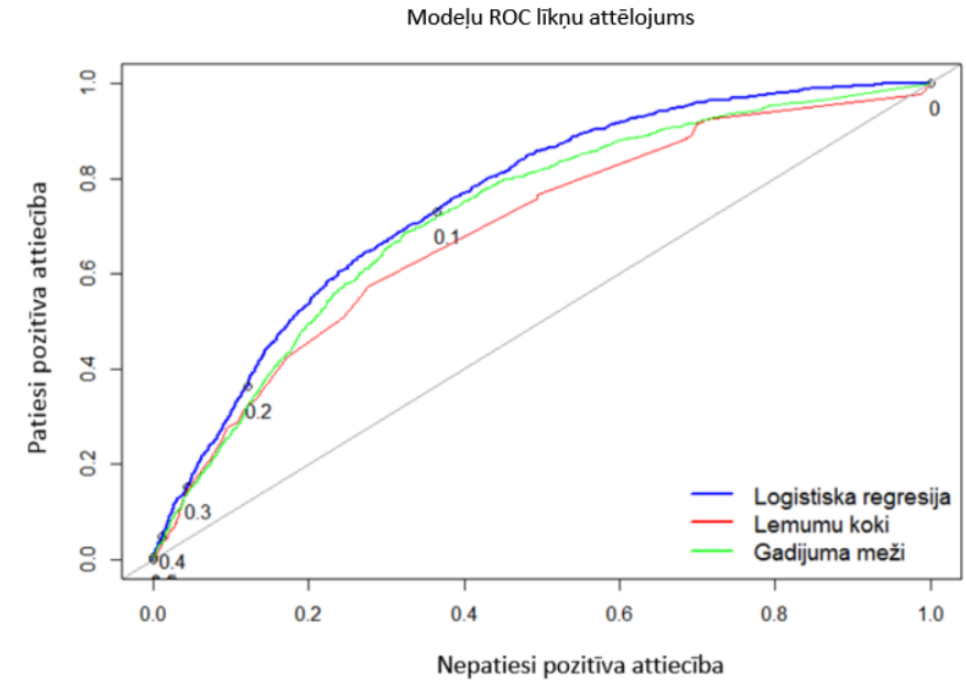
Tabula 8.5: **Prognozes precizitātes mēru salīdzinājums**

Metode	TP	FP	C	Sn	Sp
Loģistiskā regresija	0.9995	0.0127	0.9995	0.9989	0.0041
Lēmumu koku algoritms	0.9834	0.7845	0.8788	0.6974	0.07364
Gadījuma mežu algoritms	0.9947	0.9753	0.8882	0.8928	0.9753

Tabula 8.6: **AUC kritērija vērtības**

Modeļa algoritms	AUC kritērija vērtība
Lēmumu koks	0.613
Gadījuma meži	0.726
Loģistiskā regresija	0.756

Grafiskos rezultātus var apskatīt 8.11 attēlā :



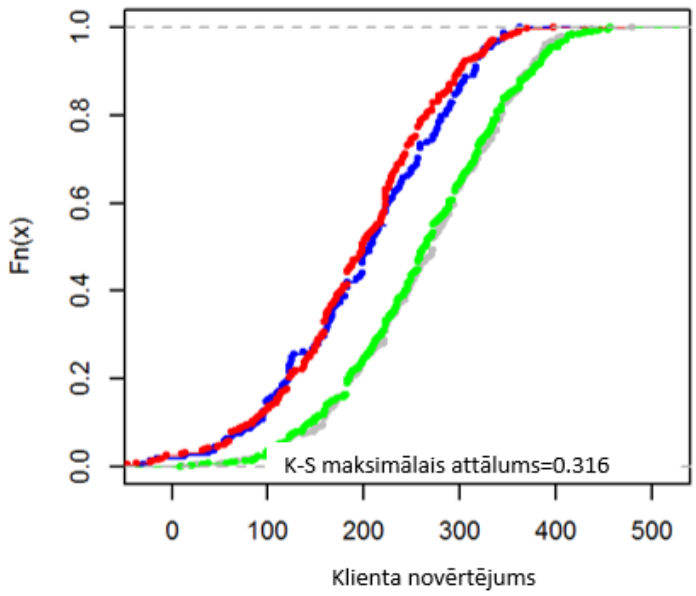
Att. 8.11: **ROC līknes salīdzinājums**

Grafikā ir redzams, ka vislielākais laukums (lielākā AUC vērtība) ir loģistiskajai regresijai, otrā lielākā vērtība ir gadījuma mežiem un vismazākā AUC vērtība ir lēmumu koku modelim.

Apskatot iegūtos rezultātus un ņemot vērā algoritmu priekšrocības un trūkumus, kā labākais modelis tiek izvēlēts loģistiskās regresijas modelis.

Kolmogorova-Smirnova statistika Ar $Fn(x)$ apzīmē kumulatīvo sadalījuma funkciju. Attiecīgi tiek apskatīta kumulatīvā sadalījuma funkcija atsevišķi klientiem, kuriem ir sliktāks novērtējums pēc modeļa un atsevišķi klientiem, kuriem ir labvēlīgs iznākums no modeļa. Abas kumulatīvās sadalījuma funkcijas tiek zīmētas vienā grafikā un mērķis ir maksimizēt starpību starp abām funkcijām. Tiek apskatītas divas apakšgrupas, kuras tiks sauktas par kreiso grupu labo grupu . Tad Kolmogorova-Smirnova kritērijs ir sekojošs: atrast dalījuma vietu starp kreiso un labo grupu , kas maksimizē sekojošu izteiksmi [2]: *varbūtība, ka nelabvēlīgie klienti atrodas kreisajā grupā mīnuss varbūtība, ka labvēlīgie klienti atrodas kreisajā grupā*.

Loģistiskās regresijas Kolmogorova-Smirnova statistikas pārbaudes rezultāti ir redzami 8.12 attēlā:



Att. 8.12: **Kolmogorova-Smirnova tests loģistiskās regresijas modelim**

Šajā grafikā ir redzamas kumulatīvās sadalījuma funkcijas. Attiecīgi sarkanā līnija ir modeļa prognozētās nelabvēlīgo klientu vērtības un zaļā ir modeļa prognozētās labvēlīgās vērtības. Attiecīgi zilā un pelēkā sadalījuma funkcija ir zīmēta no testa datiem. Kā redzams, labvēlīgo klientu sadalījuma funkcija tiek prognozēta nedaudz precīzāk nekā nelabvēlīgo iznākumu funkcija.

9 SECINĀJUMI

1. Darba mērķis ir izveidot piemērotāko modeli, ar kuru prognozēt, vai klients laikā atmaksās aizdevumu (labvēlīgs iznākums), vai arī klients neatmaksās laikā (nelabvēlīgs iznākums), kas arī tika izpildīts.

2. Modeļi tika veidoti divas reizes. Pirmajā reizē modeļi tika veidoti ar mazāku datu apjomu un otrajā reizē datu apjoms bija lielāks. Otrajā reizē modeļus izveidot bija vieglāk un tie bija stabilāki, tātad datu apjomam ir liela nozīme.

3. Manuprāt, efektīvākais veids kā izvēlēties visus nozīmīgākos mainīgos ir izveidot korelāciju matricu, pirmajā kolonnā pievienot informācijas vērtību, sakārtot dilstošā secībā un tad attiecīgi izslēgt visus korelētos mainīgos un no pārējiem izvēlēties loģiski pamatotos.

4. Svarīgi ir sadalīt datus treniņa un testa datos, lai pēc modeļa izveidošanas pārbaudītu ar testiem un grafikiem, cik labi izveidotais modelis apraksta datus.

5. Koku apgriešana ir nozīmīga gan lai uzlabotu precizitāti, gan vienkāršotu rezultātu implementāciju.

6. Salīdzinot modeļus pēc AUC kritērija un attēlojot to grafiski ROC līknes grafikā, labākais ir loģistiskās regresijas modelis, taču gadījuma mežu algoritma modeļa precizitāte ir tuvu loģistiskās regresijas modeļa precizitātei.

7. Loģistiskās regresijas modeļa priekšrocība pār gadījuma mežu algoritmu: loģistiskās regresijas modelis sniedz varbūtības novērtējumu.

8. Izveidotais modelis tika implementēts un tiek izmantots uzņēmuma klientu pieteikumu izvērtēšanas procesā.

Literatūras saraksts

[1] Anderson R., “The Credit Scoring Toolkit: Theory and Practice for Retail Credit Risk Management and Decision Automation” , *OUP Oxford* , 2007.

[2] Cestnik B., Bratko I. “On estimating probabilities in tree pruning. Proceedings of the E=S¸-91.”*Springer*, 1991.

[3] Fawcett T., ’Center for Computer Research in Music and Acoustics’ mājaslapas sadaļa ”An introduction to ROC analysis” `https://ccrma.stanford.edu/workshops/mir2009/references/ROCintro.pdf` , skatīts 09.01.2017

[4] Kuhn M. and Johnson K., “Applied Predictive Modeling”, *Springer*, 2013.

[5] Mingers J., “An empirical comparison of selection measures for decision-tree induction”In: Machine learning 3.4 ,1989.

[6] Quinlan J.R., “Machine Learning”, 1986

[7] Quinlan J.R. “Simplifying decision trees.”*International Journal of Man-Machine Studies*, 1987

[8] Quinlan J.R. “C4.5: programs for machine learning”*Morgan Kaufmann Publishers*, 1993.

[9] Rodríguez G., Princeton Universitātes Statistikas kursa mājaslapas sadaļa ”Logit Models for Binary Data” `http://data.princeton.edu/wws509/notes/c3.pdf` , skatīts 01.06.2017

[10] Rokach L., “Data Mining with Decision Trees: Theory and Applications. Series in machine perception and artificial intelligence.” *World Scientific Publishing Company, Incorporated,* 2008.

[11] Tan P.N., Steinbach M., Kumar V. “Introduction to data mining”, 2005

[12] Thomas L.C., Edelman D.B., Crook J.N., “Credit Scoring And Its Applications (Monographs on Mathematical Modeling and Computation) , ”*SIAM* , 2002.

[13] Ruohonen K., “GRAPH THEORY , ”, 2013.

PIELIKUMI

1.pielikums

```
#Funkcija, kas āprbauda, vai dati ir ītri (ir tikai viena klase):ī

  IrTra <- function(dati) {
    length(unique(dati[,ncol(dati)])) == 1 #āprbauda, vai ir tikai viena
        āunikla ēīvrtba}

#ēImplement entropijas ēmru:

  Entropija <- function( vls ) {
    res <- vls/sum(vls) * log2(vls/sum(vls))
    res[vls == 0] <- 0
    -sum(res)}

#ēDefin āinformcijas ieguvuma funkciju IG:

  IG <- function( dati ) {
    dati <- as.data.frame.matrix(dati)
    PimsEntropija <- Entropija(colSums(dati))
    s <- rowSums(dati)ē
    PcEntropija <- sum (s / sum(s) * apply(dati, MARGIN = 1, FUN =
        Entropija ))
    ig <- PimsEntropija - ēPcEntropija
    return (ig)}

#ID3 algoritms ēdefints āk funkcija:

  ID3 <- function(virsotne, dati) {
    virsotne$obsCount <- nrow(dati) #ēNovrojumu skaits ēvirsotn

        if (īIrTra(dati)) { #Ja datu kopa ir ītra, tad
            child <- virsotne$AddChild(unique(dati[,ncol(dati)]))
             #Tiek izveidota lapa ēkonkrtai klasei ('labs'/'slikts')

      virsotne$pazime <- tail(names(dati), 1)
      child$obsCount <- nrow(dati)
      child$feature <- ''
    } else {
```

```
36
37      #Ja datu kopa nav ītra, ēizvlas īpazmi ar ālielko āinformcijas ieguvumu:
38
39        ig <- sapply(colnames(dati)[-ncol(dati)],
40                     function(x) IG(table(dati[,x], dati[,ncol(dati)])))
41        pazime <- names(ig)[ig == max(ig)][1]
42        virsotne$pazime <- pazime
43
44        #Tiek šķelta kopa ēpc ēēāizvlts īpazmes:
45        childObs <- split(data[,!(names(dati) %in% pazime)], dati[,pazime],
              drop = TRUE)
46        for(i in 1:length(childObs)) {
47
48          #Tiek izveidots 'ēbrns' ar ēākonkrts īpazmes ēīvrtbu:
49          child <- virsotne$AddChild(names(childObs)[i])
50          ID3(child, childObs[[i]])
51
52          #Algoritms tiek āatkrtots īrekursvi
53        }
54      }
55    }
56    tree <- Node$new("client")
57    ID3(tree, train)
```

2.pielikums

```
1  library(C50)
2  c50data<-train
3  c50_result<-C5.0(formula=target~.,x=c50data, y=train$target)
4  summary(c50_result)
5  C5imp(c50_result,metric='usage')
6  predict_c50<-predict(c50_result, test, type='class')
7  summary(predict_c50)
```

3.pielikums

```
#āāSkum tiek īizlasti āēskotnji ēdefintie ēēizvltie īmaingie:
    var<-paste(model1[1], "+" )
    for(i in 2:(length(model1)-1))
    {
      var <- paste(var, model1[i], "+" )
    }
    var<-paste(var,model1[length(model1)])
    frm<-as.formula(paste("target ~",var)) #ēDefin formulu
    mylogit <- glm(formula = frm, family = "binomial", data = train) #ēDefin
        regresiju
    coeff<-as.data.frame(coef(mylogit)) #Regresijas koeficienti
    summary_data<-data.frame(variable = names(summary(mylogit)$coefficients
        [,1]),coef = coeff[,1])


    ml<-summary(mylogit) #āPrbauda koeficientu īīnozmbas īlmeni
```

4.pielikums

```
train$logodds <- predict(mylogit) #ēPrognoz ānkotnes ēīvrtbas
    train$prob <- predict(mylogit,type='response')
    train$score1 <- offset - factor*train$logodds
    pred<-prediction(train$score1,train$target,label.ordering =c(1,0))
    perf<-performance(pred,"tpr","fpr")
    auc<-performance(pred,"auc")@y.values[[1]]
    gini_train<-2*auc-1

    test$logodds <- predict(mylogit,test)
    test$prob <- predict(mylogit,type='response',test)
    test$score1 <- offset - factor*test$logodds
    predtest<-prediction(test$score1,test$target,label.ordering =c(1,0))
    perftest<-performance(predtest,"tpr","fpr")
    auctest<-performance(predtest,"auc")@y.values[[1]]
    gini_test<-2*auctest-1

        # ROC īlknes īēšzmana:
    plot(perf,colorize=T, print.cutoffs.at=seq(round(x_min,-2),round(x_max
        ,-2)-100,by=100), text.adj=c(0.1,2), lwd=2, main ="Model 1: ROC (
        Train vs Test)")
    plot(perftest,col = 'black', add = TRUE)
    legend("bottomright",c(paste("GINI train = ",round(100*gini_train,2),"%
        ", sep = "")
                           ,paste("GINI test = ",round(100*gini_test,2),"%"
                               , sep = "")),bty="n",cex=1.1)
    abline(a=0, b=1, col = "gray60")
```

5.pielikums

```
TPR <- attr(perf,'y.values')[[1]]
FPR <- attr(perf,'x.values')[[1]]
cutoff <- attr(perf,'alpha.values')[[1]]
distance <- TPR - FPR
KS <- max(distance)
performance <- data.frame(cutoff, TPR, FPR, distance)
KS_score <- performance[ which(performance$distance==KS),1]
i<-1
for (i in 1:length(performance[,1]))
{
  train$performance_temp <- ifelse((train$score1 >= performance$cutoff[
      i])==TRUE,1,0)
  train$bad_temp <- ifelse((train$score1 >= performance$cutoff[i])==
      TRUE & train$target == 1,1,0)
  performance$AR[i] <- round(100*sum(train$performance_temp)/nrow(train
      ),0)
  performance$BR[i] <- round(100*sum(train$bad_temp)/sum(train$
      performance_temp),3)
  i <- i + 1
}
col_to_delete <- grep("performance_temp", colnames(train))
train <- train[,-col_to_delete]
col_to_delete <- grep("bad_temp", colnames(train))
train <- train[,-col_to_delete]
```

Bakalaura darbs „Lēmuma koku, gadījuma mežu un loģistiskās regresijas modeļu salīdzinājums klientu maksātspējas prognozēšanai” 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: *(personiskais paraksts)* Kristīne Grundmane

Rekomendēju darbu aizstāvēšanai

Vadītājs: asoc. prof. Jānis Valeinis(*paraksts)* 05.06.2017.

Recenzents: Nadežda Siņenko

Darbs iesniegts Matemātikas nodaļā __.06.2017.

Dekāna pilnvarotā persona: vecākā metodiķe Dzintra Holsta

Darbs aizstāvēts Valsts pārbaudījuma komisijas sēdē

____ 06.2017. prot. Nr._____

Komisijas sekretāre: asociētā profesore Ingrīda Uļjane