VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

Bakalauro darbas

Kredito rizikos prognozavimas naudojantis mašininio mokymosi algoritmu „XGboost".

Credit Risk Forecast Using Machine Learning Algorithm „XGboost".

Domantas Norkus

VILNIUS 2020

MATEMATIKOS IR INFORMATIKOS FAKULTETAS
MATEMATINĖS ANALIZĖS KATEDRA

Darbo vadovas doc. dr. Martynas Manstavičius

Darbo recenzentas _____

Darbas apgintas

Darbas įvertintas _____

Registravimo NR. _____

Įrašoma atidavimo į katedrą data _____

Kredito rizikos prognozavimas naudojantis mašininio mokymosi algoritmu „XGboost”

Santrauka

Šiame darbe bus išbandomas mašininio mokymosi algoritmas „XGboost” siekiant prognozuoti reitingo pokytį. Bus gilinimąsi į algoritmo teorinius pagrindus, kaip sprendžiamos regresijos ir reguliarizacijos problemos medžių erdvėje ir kaip įvertinti modelio rezultatus. Iš praktinės pusės, bus bandoma suformuoti logiką duotai prognozavimo užduočiai bei pasinaudojant „Python” programavimo kalbos bibliotekomis, skirtomis statistikai ir mašininiui mokymuisi, išspręsti užduotį. Nors regresinių medžių algoritmas nėra naujas dalykas, tačiau didėjantys duomenų kiekiai ir kompiuterių programiniai galingumai leidžia išplėsti panaudojimo sritis nuo medicinos vaistų testavimo iki analitinių šaškių programų kūrimo.

Raktiniai žodžiai : „XGboost”, regresiniai medžiai, reguliarizacija, optimizacija, nuostolių funkcija, prieaugio funkcija.

Credit Risk Forecast Using Machine Learning Algorithm „XGboost”

Abstract

In this bachelor thesis I will introduce machine learning algorithm „XGboost” which will try to predict rating downgrade. Attention will be drawn to algorithm's theoretical background, analysing how regression and regularization functions behave in tree space. Thus, methods for analysing model performence will be introduced and explained. From practical perspective, will try to set up explained algorithm for given prediction problem and solve it with use of „Python” libraries, created

for statistics and machine learning. Though regression trees is not a new subject, nevertheless, with increased amount of data and development of computer technologies, capabilities of solving different problems have widened from medicine drugs testing to implementing an analytical checkers program.

Key words : „XGboost", regression trees, regularization, optimizaton, loss function, gain function.

Turinys

1 Įvadas

1.1 Pasirinktos temos aprašymas

Pasirinkta tema – kliento rizikos vertinimas, susijęs su tikimybe bankrutuoti (angl. probability of default) - ar kliento reitingas per ateinančius 6 mėnesius pasikeis iš A į B naudojantis „XGboost" algoritmu, pasitelkiant įvairius duomenis (transakcijų, kredito ir finansų). Gebėjimas tinkamai įvertinti kliento riziką bankrutuoti, priskirti skaitinę reikšmę, kuri veliau konvertuojama į reitingą priklausomai nuo metodologijos, yra vienas esminių būdų kontroliuoti portfelio riziką, kadangi šis rodiklis koreliuoja su tokiais išvestiniais dydžiais kaip LGD, EAD, kurių sandauga lygi tikėtiniems nuostoliams (angl. expected loss). Šiuo atveju prognozavimui pasirinkau mašininio mokymosi (angl. machine learning) vieną iš algoritmų *XGboost* [8], kuris remiasi matematikos pagrindais, tokiais kaip: regresija, aproksimacija, normalizacija ir t.t. Tobulėjantys matematiniai modeliai, duomenų surinkimo ir apdirbimo metodai, kompiuterių galingumai ir operatyvumai leidžia vis plačiau naudoti mašininį mokymąsį įvairiose srityse: duomenų klasifikavime, paieškos sistemose, medicinos diagnostikoje ir t.t. Tačiau šio metodo problematika plati: duomenų surinkimo ir naudojimo teisiniai apribojimai, duomenų kaina, energetiniai kaštai, netinkamai parinktas modelis, duomenų trūkumas arba perteklius, nereikšmingi parametrai, plačiai interpretuojami rezultatai. Dažnai įvairios problemos yra aiškinamos empiriniu, o ne teoriniu būdu. Ne veltui šis metodas dažnai vadinamas „juodąją dėže" ir sulaukia skeptiškų vertinimų dėl logikos ir algoritmų aiškumo ir baigtumo trūkumų bei rezultatų subjektyvumo. Todėl ateityje modeliuotojams bus svarbu sugebėti paaiškinti duomenų priežastingumą, algoritmo struktūrą bei rezultatus lemiančius parametrus – tai pasiekti padeda tinkami matematiniai ir statistiniai įrankiai.

1.2 Bakalauro darbo tikslo ir proceso aprašymas

Bakalauro darbo tikslas – įsigilinti iš teorinės pusės, kaip veikia duotasis algoritmas, susipažinti su regresinių medžių bei reguliarizacijos teorija, modelio įvertinimo galimybėmis bei įgyti programavimo patirties naudojantis „Python" programavimo kalba. Bakalauro darbo procesas susideda iš teorinės ir praktinės dalies. Teorinėje dalyje bandysiu gilintis

į regresinių medžių mokinimosi algoritmo logiką, su kokiomis problemomis susiduria šis modelis ir kaip jos sprendžiamos. Taip pat bandysiu apibrėžti metodus, kaip galima įvertinti algoritmo rezultatą, lyginti su kitais modeliais. Praktinėje dalyje išbandysiu „XGBoost" algoritmą naudojantis banko duomenimis bei išbandant skirtingus teorinėje dalyje aprašytus parametrus ir įvertinti jų įtaką rezultatams.

2 Teorinė dalis

XGBoost algoritmas, angliškai verčiamas kaip eXtreme Gradient Boosting [8], yra 2014 m. sukurtas kinų mokslininko Tianqi Chen. Šis algoritmas yra itin populiarus tarp modeliuotojų, ne vieną kartą laimėjęs statistinio modeliavimo rungtyse „Kaggle" tinklalapyje. Tokią sėkmę lemia du faktoriai – modelio greitumas ir tikslumas. Greitumo tematika teorinėje dalyje nebus paliesta, nes ji pagrinde susijusi su programavimo architektūra, todėl pagrinde bus gilinimąsi į matematinę dalį. Teorinėje dalyje bandysiu apibrėžti algoritmą ir jo parametrus bei funkcijas, pateiksiu naudingų formulių išvedimus ar nuorodas į jas bei plačiau panagrinėsiu parametrų svarbą modelyje. Remsiuosi sąvokomis, išverstomis iš anglų kalbos, pagal šį tinklapį `https://klevas.mif.vu.lt/~linp/page/savokos.html`. Teorinėje dalyje aptarsiu tik pagrindinius modelyje naudojamus parametrus.

2.1 XGBoost algoritmas

2.2 Mašininio mokymosi įvadas ir apibrėžimai

Kiekvieno mašininio mokymosi tikslas toks pat, kaip ir tiesinės ar logistinės regresijos – iš turimų duomenų x_i atlikti prognozę, kurios rezultatas lygus $\widehat{y_i}$ ir apibrėžiamas formule:

$$\widehat{y_i} = \sum_{j}^{d} w_j x_{ij},$$

čia $x_{ij} \in \mathbb{R}^{n \times d}$, kur i indeksas žymi elemento poziciją duomenų aibėje

$$X = \{x_1, x_2, \ldots, x_i, \cdots, x_n\},$$

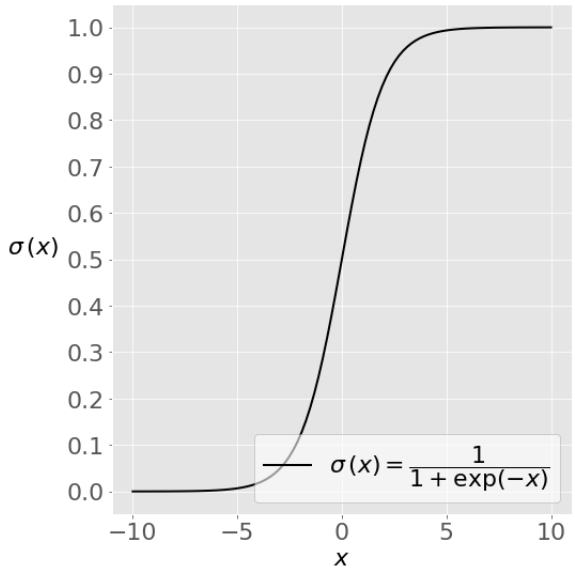
čia n – paskutinis duomenų elementas, w_j yra svorio koeficientas, kurio reikšmės mes ieškosime, j – svorio koeficiento pozicija aibėje

$$\theta = \{w_j | j = 1, \ldots, d\},$$

čia d žymi maksimalų svorio koeficientų skaičių. Duomenų pavyzdžiai x_i finansų modelyje būtų tokie: kliento skola, vidutinis transakcijų skaičius per dieną, pavėluotų mokėjimų skaičius ir t.t. Prognozės y_i rezultatas priklauso nuo to, kokią regresiją naudosime. Jeigu naudojame tiesinę regresiją, $\widehat{y_i}$ yra prognozuojamas rezultatas, jeigu naudojame logistinę regresiją, rezultatas lygus:

$$\frac{1}{(1 + e^{-\widehat{y_i}})}$$

Šis rezultatas reiškia prognozuojamą tikimybę, jog, pavyzdžiui, klientas bankrutuos arba ne. Žemiau pavaizduotas logistinės funkcijos grafikas.



Pav. 1: Kaip matoma, prognozės įgyja reikšmes intervale $[0, 1]$. Grafike vietoj $\widehat{y_i}$ naudojamas x kintamasis [14].

Logistinės regresijos privalumas tas, kad galime pasirinkti slenkstinę vertę intervale $[0, 1]$, priklausomai nuo kurios galutinė prognozės reikšmė lygi 0 arba 1, kaip minėtas pavyzdys, bankrutuos arba ne. Tarkime, jeigu slenkstinė vertė lygi 0.5, o

$$\frac{1}{(1 + e^{-\widehat{y_i}})} = 0.6,$$

tada kadangi $0.6 > 0.5$ ir todėl prognozės reikšmė lygi 1 – klientas bankrutuos. Tačiau jeigu mūsų apibrėžime klientas bankrutuoja, kai tikimybė lygi 0.8, tada kadangi $0.6 < 0.8$, prognozės reikšmė lygi 0 – klientas nebankrutuos. Tam, kad įvertinti, ar prognozės reikšmė statistiškai skiriasi nuo žinomos reikšmės, naudojama nuostolių funkcija l. Tiesinėje regresijoje ji lygi

$$l(y_i, \widehat{y_i}) = (y_i - \widehat{y_i})^2$$

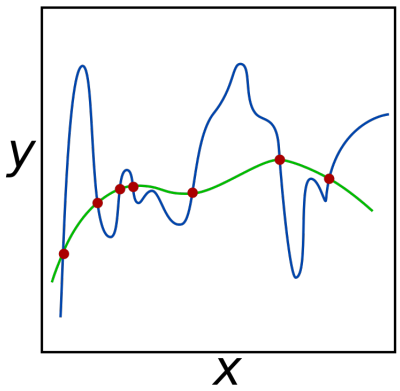
Visgi praktikoje spręsiu klasifikavimo užduotį, todėl reikia apibrėžti *logistinę nuostolių funkciją*.

Apibrėžimas 1. *Logistinė nuostolių funkcija* l yra apibrėžiama kaip:

$$l(y_i, \widehat{y_i}) = y_i \ln(1 + e^{-\widehat{y_i}}) + (1 - y_i) \ln(1 + e^{\widehat{y_i}}), \tag{1}$$

čia y_i žinoma reikšmė, o $\widehat{y_i}$ – prognozuojama.

Taigi, išmokus įvertinti prognozės ir tikros reikšmės skirtumą, galime optimizuoti duotąją funkciją, tačiau net ir gavus tikslią prognozę, svarbu įvertinti modelio kompleksiškumą. Mat gali būti, jog prognozės tikslumas didelis, tačiau ir standartinis nuokrypis didelis, kaip nuotraukoje apačioje:



Pav. 2: Kaip matoma, žalios ir mėlynos linijos nuostolių funkcijos atžvilgiu atlieka puikias prognozes, tačiau, ant mėlynosios funkcijos grafiko esančių taškų, standartinis nuokrypis gerokai didesnis [15].

Šią problema padeda išspręsti *reguliarizacijos procesas*.

Apibrėžimas 2. *Reguliarizacija* – tai procesas padedantis išvengti persimokymo, o jos funkcija $\Omega(w)$ yra lygi:

$$\Omega(w) = \lambda \sum_{i=1}^{d} |(w_i)|^p,$$

čia w_i yra regresijos svorio koeficientas, λ yra reguliarizacijos koeficientas, įgyjantis reikšmes intervale $[0; 1]$, o p koeficientas įgijantis reikšmes intervale $[0; \infty)$.

Minėtame antrame paveikslėlyje, po tam tikro iteracijų skaičiaus, didinant λ koeficiento reikšmę ir pridedant tam tikras pasvertas reikšmes prie mėlynojo grafiko, jis taptų panašus į žaliąjį ir tokiu būdu jo prognozės taptų mažiau jautrios ateičiai.

2.3 Kodėl naudoti regresinius medžius o ne logistinę regresiją?

Iš esmės, šie du algoritmai yra gana panašūs, nes regresiniai medžiai remiasi logistinės regresijos principais – jie naudojasi esamais duomenimis išmokti įvertinti svorius ir atlikti prognozę bei kaip ją optimizuoti. Tačiau manau yra keletas priežąsčių rinktis regresinius medžius:

- Vizualizavimas. Medžių struktūra yra patogi norint atvaizduoti procesą ir jį paaiškinti.
- Gebėjimas reguliarizuoti algoritmą įvairiau bei optimizuoti skirtingus parametrus, kurių nėra logistinėje regresijoje.

2.3.1 Reguliarizuotas mokymosi algoritmas

Praeitame skyriuje apibrėžėme prognozės formulę $\widehat{y_i}$. Tiesa, ji, ir kitos formulės bei apibrėžimai, buvo siejami su tiesine erdve, tačiau praktikoje naudosiu „regresinių medžių" erdvę, kurią reikėtų apibrėžti.

Apibrėžimas 3. *Regresinių medžių prognozė* $\widehat{y_i}$ lygi:

$$\widehat{y_i} = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{G}, \tag{2}$$

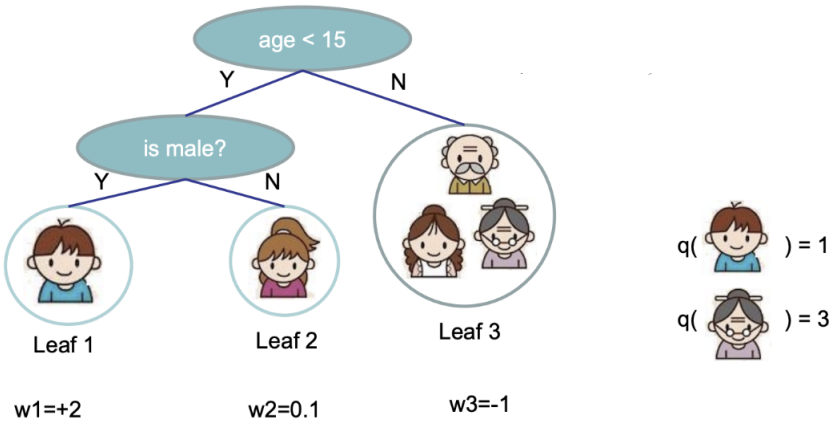
čia K yra medžių skaičius, $x_i \in \mathbb{R}^d$ yra duomenų pavyzdžiai, f_k yra funkcijos, priklausančios regresinių medžių erdvei $\mathcal{G}$, o $\mathcal{G}$ yra regresinių medžių erdvė.

Apibrėžimas 4. *Regresinių medžių erdvė* $\mathcal{G}$ apibrėžiama

$$\mathcal{G} = \{f_t(\cdot) = w_{q(\cdot)}, w \in \mathbb{R}^T, q : \mathbb{R}^d \longrightarrow \{1, 2, \ldots, T\}\},$$

čia $f_t()$ yra funkcija, kuri moka įvertinti lapo svorį ir bendrai tariant – medžio struktūros kokybę, w žymi lapo svorį, T – lapų skaičių, o q yra medžio struktūros funkcija, kuri duomenų pavyzdžiui x_i priskiria lapo indeksą. Anksčiau minėtuose pavyzdžiuose, svorio koeficientas w_i buvo ieškomas dydis, o regresiniuose medžiuose šį dydį atitinka funkcija f_t, todėl atitinkamai

$$\Theta = \{f_1, f_2, \cdots, f_K\}$$



Pav. 3: Kaip matoma, $f(\text{berniukas}) = w_1 = 2$ – tai pirmojo lapo svoris lygus 2 ir $q(\text{berniukas}) = 1$ – čia q funkcija parodo, kuriame lape yra kuris duomenų pavyzdys x_i. Kaip matoma, kairėje atšakoje esantys duomenų pavyzdžiai išpildo tam tikrą sąlyga – yra jaunesni nei 15 metų, o dešinėje jos netenkina [8].

Reikia apibrėžti *reguliarizacijos funkciją* medžių erdvėje.

Apibrėžimas 5. *Reguliarizacijos funkcija* Ω medžių erdvėje lygi:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2,$$

čia γ yra šakų naikinimo parametras, įgyjantis reikšmes intervale $[0; \infty)$, o λ - lapų. Šaka apibrėžiama kaip viršūnė ir dvi jos atšakos: kairioji ir dešinioji. Čia Ω reikšmė trečiajame paveikslėlyje lygi

$$\gamma 3 + \frac{1}{2}\lambda(4 + 0.01 + 1)$$

Taigi, apibendrinus nuostuolių ir reguliarizacijos funkcijas, reikia apibrėžti *tikslo funkciją* $\mathcal{G}$ regresinių medžių erdvėje.

Apibrėžimas 6. *Tikslo funkcija* $\mathcal{F}$ lygi:

$$\mathcal{F} = \sum_{i}^{n} l(y_i, \widehat{y_i}) + \sum_{k}^{K} \Omega(f_k), \tag{3}$$

čia l yra nuostolių funkcija, o Ω – reguliarizacijos funkcija, padedanti kontroliuoti modelio kompleksiškumą. Įprastai, reguliarizacijos funkcijoje parametras p lygus 1 arba 2.

Kadangi minėtoje tikslo funkcijoje (3) ieškomų svorių aibę Θ sudaro funkcijos f_k, funkcija negali būti optimizuojama įprastais optimizavimo metodais Euklidinėje erdvėje [8]. Todėl, modelis yra apmokomas iteraciniu būdu. Kaip tai daroma? Pradedame nuo pradinės prognozės, kuri dažniausiai yra konstanta:

$$\widehat{y_i}^{(0)} = 0$$

ir prie jos iteratyviai pridedame naują funkciją $f_t(x_i)$ ir gauname:

$$\widehat{y_i}^{(1)} = f_1(x_i) = \widehat{y_i}^{(0)} + f_1(x_i)$$

Galiausiai momentu t turime:

$$\widehat{y_i}^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \widehat{y_i}^{(t-1)} + f_t(x_i)$$

Tada (3) formulė lygi:

$$\mathcal{F}^{(t)} = \sum_{i=1}^{n} l(y_i, \widehat{y}^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{4}$$

Kaip nuspręsti, kokią funkciją f pridėti? Tikslas rasti tokį f_t, kuris minimizuotų (4) funkciją. Tačiau duotoji funkcija atrodo sudėtinga. Antros eilės Teiloro skleidinys padeda paprasčiau ir greičiau optimizuoti duotąją (4) funkciją. Pageidautina sąlyga – nuostolių funkcija l yra du kartus diferencijuojama. Teiloro skleidinys atrodo taip:

$$f(x) \approx f(a) + f'(a)(x-a) + \frac{1}{2} f''(a)(x-a)^2$$

Iš čia:

$$\mathcal{F}^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \widehat{y}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t), \tag{5}$$

kur

$$g_i = \partial_{\widehat{y}^{(t-1)}} l(y_i, \widehat{y}^{(t-1)}),$$

o

$$h_i = \partial^2_{\widehat{y}^{(t-1)}} l(y_i, \widehat{y}^{(t-1)})$$

Kadangi optimizuojama yra $f_t(x_i)$ funkcija, nuostolių funkciją $l(y_i, \widehat{y}^{(t-1)})$ galime prilyginti nuliui, nes ji neturi įtakos optimizavimui.Tada tikslo funkcija atrodo taip:

$$\mathcal{F}^{(t)} \simeq \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t),$$

Tada, pritaikę minėtą Teiloro aproksimaciją ir skaičiavimo paprastumui prilyginę γ nuliui, gaunama tokia transformuotos (5) lygties dešinioji pusė momentu t:

$$g f_t(x_t) + \frac{1}{2}(h + \lambda) f_t^2(x_t), \tag{6}$$

čia

$$h = \sum_{i=1}^{n} h_i, \quad g = \sum_{i=1}^{n} g_i$$

Norėdami minimizuoti (6) funkciją $f_t(x_t)$ atžvilgiu, prilyginame jos išvestinę nuliui:

$$\frac{d}{df_t(x_t)}(g f_t(x_t) + \frac{1}{2}(h + \lambda) f_t^2(x_t)) = 0 \tag{7}$$

Iš čia,

$$g + (h + \lambda) f_t(x_t) = 0$$

gauname, jog

$$f_t(x_t) = \frac{-g}{h + \lambda}$$

Dabar apibrėšiu formules regresinių medžių erdvėje. Pažymėkime duomenų pavyzdžių aibę lape su indeksu j lygu $I_j = \{i | q(x_i) = j\}$. Tada atitinkamai

$$G_j = \sum_{i \in I_j} g_i, \quad H_j = \sum_{i \in I_j} h_i,$$

o

$$\begin{aligned}\mathcal{F}^{(t)} &\simeq \sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \Omega(f_t) \\ &= \sum_{i=1}^{n}[g_i w_{g(x_i)} + \frac{1}{2}h_i w_{g(x_i)}^2] + \gamma T + \frac{1}{2}\sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}((\sum_{i\in I_j} h_i + \lambda)w_j^2)] + \gamma T \\ &= \sum_{j=1}^{T}[G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2)] + \gamma T,\end{aligned}$$

Tada, tariame jog medžio struktūra $q(x)$ yra fiksuota, tada atitinkmai optimalus lapo svoris w_j^* kiekviename lape yra lygus:

$$w_j^* = -\frac{G_j}{H_j + \lambda}, \tag{8}$$

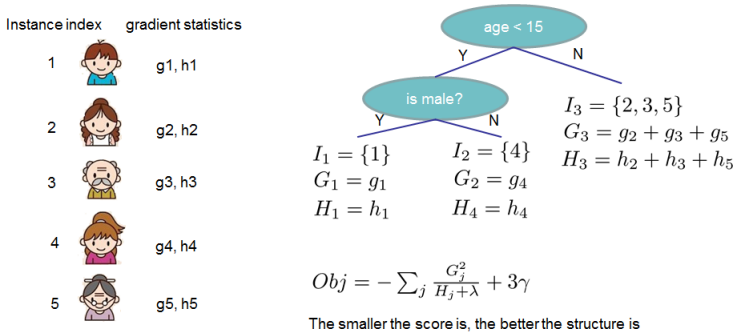
atitinkamai optimali tikslo funkcija $\mathcal{F}^{(t)}$ lygi

$$\mathcal{F}^{(t)} = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T, \tag{9}$$

čia

$$\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda}$$

įvertina medžio struktūrą.



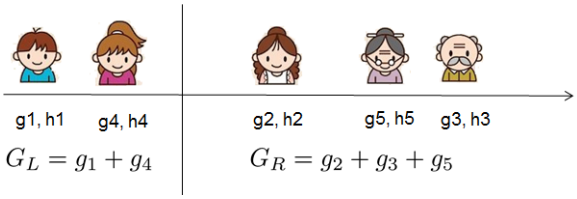
Pav. 4: Duotame pavyzdyje parodyta, kaip reikia apskaičiuoti medžio struktūros kokybę. Kuo $\mathcal{F}^{(t)}$ reikšmė mažesnė, tuo kokybė geresnė [8].

Kadangi sunumeruoti visas galimas medžių struktūras q yra sunku, jų gali būti begalybė, vietoj to yra naudojamas gobšus algoritmas (angl. „Greedy algorithm") [8], kuris pradeda

nuo viršutinio lapo ir iteratyviai formuoja medžio šakas. Tarkime, jog I_L ir I_R yra koeficientų skaičius atitinkamai kairėje ir dešinėje šakos pusėje. Tada, kai $I = I_L \cup I_R$, tai (9) funkciją remiantis [8] keičiame į:

$$\mathcal{F}_{split} = \frac{1}{2}\Big[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\Big] - \gamma \tag{10}$$

Iš esmės, $\mathcal{F}_{split}$ apibrėžiama kaip prieaugio funkcija, įvertinanti naujos šakos įtaką prognozei. Čia pirmas dėmuo skliaustuose įvertina kairiosios atšakos rezultatą, antrasis – dešiniosios atšakos, o trečiasis įvertina rezultatą, jeigu nuspręstumėme neformuoti naujos atšakos. Natūralu, jeigu $\mathcal{F}_{split} < 0$, kas reiškia modelio neigiamą tobulėjimą, arba $\gamma > \mathcal{F}_{split}$, tuomet atšaka iš medžio yra pašalinama. Klausimas, kaip optimaliausiai skirstyti duomenų pavyzdžius x_i į kairę ar dešinę atšaką?



Pav. 5: Duotame pavyzdyje, kairioji ir dešinioji atšaka formuojama pagal taisyklę $x_j < a$, kur x_j yra amžius [8].

Mums reikia g_i ir h_i sumos ir tada galime apskaičiuoti $\mathcal{F}_{split}$ reikšmę. Optimalią reikšmę gauname iteratyviai išbandę visas įmanomas atšakas, šiuo atveju, tai priklauso nuo a. Procesą tęsiame augindami naujas atšakas su skirtingomis savybėmis. Naudinga paminėti, jog H_j yra vadinamas *Cover* parametras, apie jį rašoma dokumentacijoje [13]. Jis XGBoost modelyje atitinka *min child weight* parametrą, kuris apibrėžia minimalią galimą svorių sumą lape. Įprastai, *Cover* $= 1$, tačiau jo apibrėžimo sritis $[0; \infty)$. Kuo *Cover* didesnis, tuo konservatyvesnis modelis. Svarbu paminėti, jog medis yra auginamas iki tol, kol pasiekiamas modelio pradžioje apibrėžtas maksimalaus šakų kiekis. Modelyje šis parametras apibrėžiamas kaip *max depth* [13] ir jis įprastai yra lygus 6, o jo apibrėžimo sritis lygi $[0; \infty)$. Taigi, išmokus rasti optimalią medžio struktūrą, prognozė yra tobulinama pridedant naują medį f_t, tačiau prieaugį reikia pasverti tam tikru *mokymosi koeficientu* η.

Apibrėžimas 7. *Mokymosi koeficientas* yra parametras, žymimas η ir įgyjantis reikšmes intervale $[0; 1]$. Jo pasirenkamas dydis leidžia skirtingai įvertinti naujo medžio įtaką prognozei.

2.3.2 Algoritmo apibendrinimas

- Kiekvienos iteracijos pradžioje yra pridedamas naujas medis. Pirmojo medžio reikšmė lygi pasirinktai konstantai.

- Kiekvienos iteracijos pradžioje apskaičiuojame:

$$g_i = \partial_{\widehat{y}^{(t-1)}} l(y_i, \widehat{y}^{(t-1)}) \quad \text{ir} \quad h_i = \partial^2_{\widehat{y}^{(t-1)}} l(y_i, \widehat{y}^{(t-1)})$$

- Naudodamiesi išvesta (10) formule, auginame medį $f_t(x)$ ir surandame geriausią medžio struktūrą su atitinkamais lapų svoriais.

- Grąžiname rastą optimalią reikšmę $f_t(x)$ į modelį

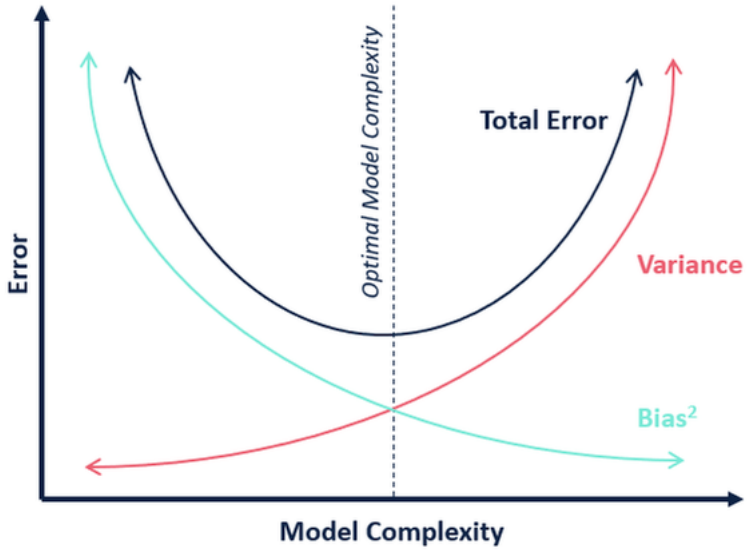
$$\widehat{y}_i^{(t)} = \widehat{y}_i^{(t-1)} + \eta f_t(x),$$

  čia η yra minėtasis mokinimosi koeficientas. Tai reiškia, jog kiekviename medyje nėra pasiekiama galutinė optimizacija, tai vienas iš būdų išvengti persimokymo.

- Procesas tęsiamas iki pasirinkto iteracijų (medžių) skaičiaus arba iki tol, kol naujas medis nepagerina prognozės rezultatų.

2.4 Reguliarizacija

Kaip minėta anksčiau, reguliarizacija yra būdas išvengti persimokymo. Ji reikalinga, nes naudojant tik mažiausiųjų kvadratų regresijos metodą, prognozės gali būti nestabilios - turėti didelią dispersiją ar poslinkį, (angl. „Bias-Variance tradeoff"), kurio grafikas žemiau pateiktas.



Pav. 6: Kaip matoma, tikslas rasti dispersijos ir funkcijos poslinkio susikirtimo tašką [12] .

Šiame skyriuje aptarsiu du reguliarizacijos metodus: *Ridge*, ir *Lasso* bei paminėsiu *Elastic Net* regresiją. [1].

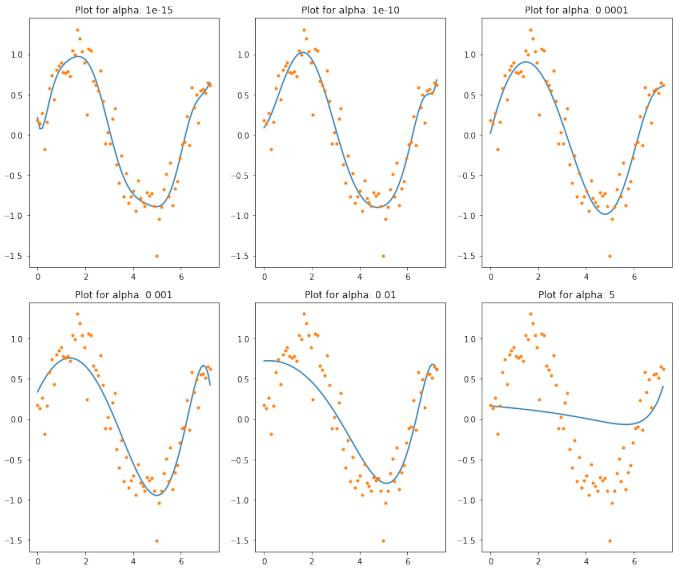
2.4.1 Ridge (α) ir Lasso (λ) reguliarizacija

Tarkime, turime duomenų rinkinį (x^i,y_i), $i = (1, 2, \ldots, N)$, kur $x^i = (\ x_{i1},\ldots,x_{ip})^T$ yra prognozės kintamieji, o y_i yra stebėtos prognozės reikšmės. Darome prielaidą, jog stebėjimai yra nepriklausomi, o y_i yra sąlygiškai tarpusavyje nepriklausomi. Darome prielaida, jog x_{ij} yra standartizuoti taip, kad $\sum_i \frac{x_{ij}}{N} = 0$, o $\sum_i \frac{x_{ij}^2}{N} = 1$. Tarkime, $\hat{\beta} = (\hat{\beta}_1, ..., \hat{\beta}_p)^T$, tada apibrėžiame *Lasso regresiją.*

Apibrėžimas 8. *Lasso regresija* $(\hat{\alpha}, \hat{\beta})$ yra apibrėžiama kaip:

$$(\hat{\alpha}, \hat{\beta}) = \text{argmin} \sum_{i=1}^{N} (y_i - \alpha - \sum_{j}^{p} \beta_j x_{ij})^2, \quad kur \sum_{j}^{p} |\beta_j| \leq t. \tag{11}$$

Čia $t \geq 0$ bei $\alpha \in [0; \infty)$ yra parametrai, kurie lemia reguliarizacijos funkcijos dydį duotai (3) formulei. Kuo α didesnis, tuo t atitinkamai mažesnis remiantis [1].



Pav. 7: Paveikslėlyje matoma α koeficiento įtaka regresijos funkcijai [9].

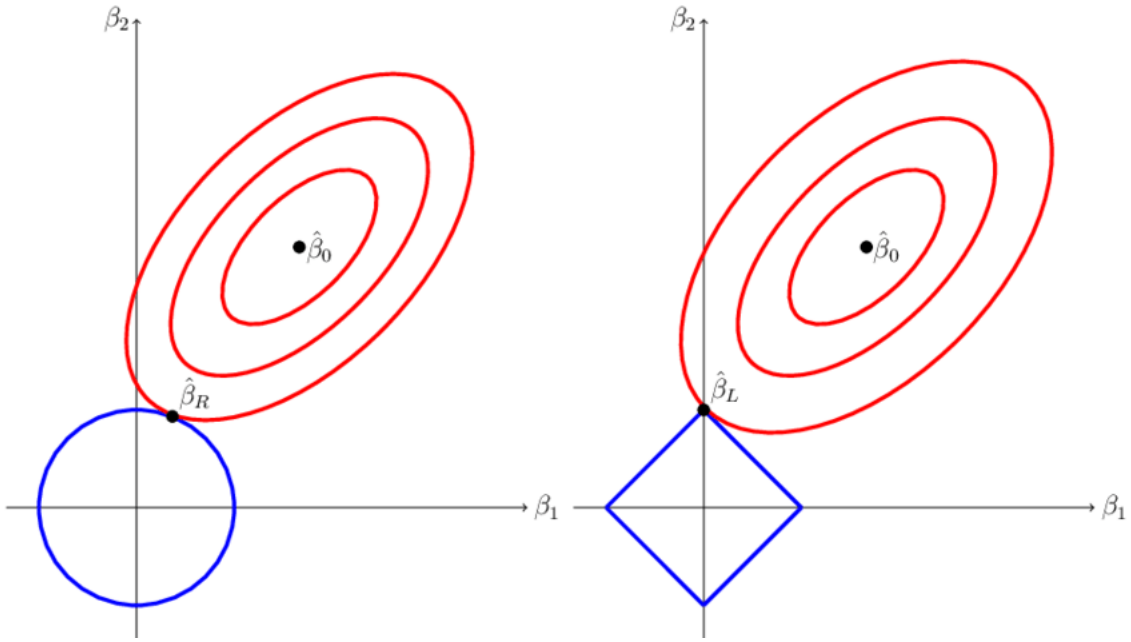
Visiems t, sprendinys α atžvilgiu yra lygus $\widehat{\alpha} = \overline{y}$. Neprarandant bendrumo, galima laikyti, jog $\overline{y} = 0$, tokiu būdu pašalinti α iš lygties [1]. Tarkime, $\widehat{\beta_j^0}$ yra mažiausiųjų kvadratų metodo įvertis ir $t_0 = \sum |\widehat{\beta_j^0}|$. Kai $t < t_0$, tuomet, lygties sprendiniai artės prie nulio, o kai kurie bus tiksliai lygus nuliui. Taigi, remiantis [1], supaprastinta *Lasso* regresijos formulė lygi:

$$\sum_{i=1}^{N}(y_i - \sum_{j}^{p} \beta_j x_{ij})^2 + \alpha \sum_{j} |\beta_j| \tag{12}$$

Atitinkamai, *Ridge* regresija yra panaši į *Lasso* regresiją, tik skiriasi β įverčio laipsnis. *Ridge* regresijos formulė apibrėžiama taip:

$$\sum_{i=1}^{N}(y_i - \sum_{j}^{p} \beta_j x_{ij})^2 + \alpha \sum_{j}^{p} \beta_j^2 \tag{13}$$

Esminis skirtumas tarp *Ridge* ir *Lasso* normalizuojančių regresijų yra tai, jog *Lasso* sugeba panaikinti, prilyginti nuliui tam tikrus paramterus, o *Ridge* tik artina link nulio. Skirtumą tarp šių skirtingų $\sum_j |\beta_j| \leq t$ ir $\sum_j \beta_j^2 \leq t$ apibrėžimų galima pavaizduoti grafiškai, kai parametrų skaičius $p = 2$ ir tuo įsitikinti.



Pav. 8: Čia kairėje pusėje *Ridge* regresija, dešinėje *Lasso*, o $\widehat{\beta_0}$ yra mažiausųjų kvadratų regresijos sprendinys. Plotai po mėlynom linijom yra atitinkamai apibrėžti kaip $\beta_1^2 + \beta_2^2 \leq t$ ir $|\beta_1| + |\beta_2| \leq t$.

Lasso sprendinys, matomas dešinėje nuotraukos pusėje, yra pirma vieta, kur elipsės kontūras liečiasi su apskritimo kontūru, ir tai kartais atsitinka kampe, kuris atitinka nulinę koeficiento reikšmę. Kairėje pusėje matomas *Ridge* regresijos sprendinys nesiliečia kampuose, todėl parametrų reikšmės nebus lygios nuliui. Taip pat, įdomus atviras klausimas, kaip skiriasi *Lasso* ir mažiausųjų kvadratų regresijos sprendiniai, kai p $>$ 2, apie tai rašoma [1].

Kaip minėta, taip pat egzistuoja šių dviejų regresijų kombinacija - *Elastic Net* regresija [3], lygi:

$$\sum_{i=1}^{N}(y_i - \sum_{j}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j}^{p}(\alpha|\beta_j| + (1-\alpha)\beta_j^2), \quad \text{kur } \alpha = \frac{\lambda_2}{\lambda_2 + \lambda_1} \tag{14}$$

Iš (14) lygties išplaukia, jog kai $\lambda_2 = 0$, tai regresija lygi *Ridge* regresijai (13), ir atvirkščiai, kai $\lambda_1 = 0$ tada regresija lygi *Lasso* regresijai (12). Priešingu atveju, kai tiek $\lambda_1 > 0$, tiek $_2 > 0$, turime *Elastic Net* regresiją.

Taigi, remiantis [1], Tibshirani apie *Lasso* ir *Ridge* regresijas priėjo tokių išvadų, jog kai stebimi nuo vidutinės iki didelės apimties mažas skaitines vertes turintys parametrai - svoriai, naudojama *Ridge* regresija, kai imtis tarp mažos ir vidutinės, o parametrų reikšmės taip pat - naudojama *Lasso* regresija. O jeigu koreliacija tarp parametrų stipri - naudojama *Elastic*

Net regresija. Apie tai daugiau rašoma [3]. Nors pradinėje reguliarizacijos formulėje (3) naudojama $Ridge(\lambda)$ regresija, tačiau algoritmas leidžia pasirinkti tiek λ, tiek α parametrus. Taigi, jeigu α ir $\lambda > 0$, tuomet taikoma *Elastic Net* reguliarizacija.

2.4.2 Mokinimosi koeficientas η ir algoritmo konvergavimas

Skyriaus pradžioje apibrėšiu sąvokas, kurias naudosiu vėliau.

Apibrėžimas 9. *Kritinis taškas, lokalus ekstremumas ir balno taškas.* Tarkime $z = f(x, y)$ yra dviejų kintamųjų funkcija, kuri yra du kartus tolydžiai diferencijuojama taške (x_0, y_0) ir apibrėžta aibėje $\mathbb{X} \times \mathbb{Y} \subset \mathbb{R}^m \times \mathbb{R}^n$. Taškas (x_0, y_0) yra vadinamas funkcijos f *kritiniu tašku*, jeigu tenkinama vieną iš dviejų sąlygų:

- $f_x(x_0, y_0) = f_y(x_0, y_0) = 0$, čia f_x ir f_y žymi išvestines atitinkamai x ir y atžvilgiu.
- Viena iš funkcijų $f_x(x_0, y_0)$ arba $f_y(x_0, y_0)$ neegzistuoja.

Sakome, kad funkcija f turi *lokalų minimumą* taške (x_0, y_0), jeigu

$$f(x_0, y_0) \geq f(x, y)$$

su visais $x \in \mathbb{X}$ ir visais $y \in \mathbb{Y}$.
Sakome, kad funkcija f turi *lokalų maksimumą* taške (x_0, y_0), jeigu

$$f(x_0, y_0) \leq f(x, y)$$

su visais $x \in \mathbb{X}$ ir visais $y \in \mathbb{Y}$.
Sakome, kad funkcija f turi *balno tašką* (x_0, y_0), jeigu

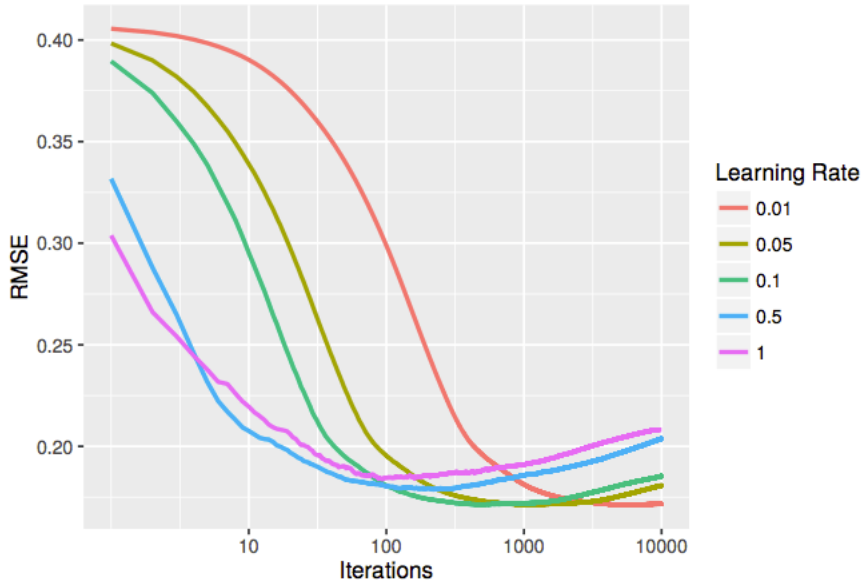
$$f_x(x_0, y_0) = 0$$

ir

$$f_y(x_0, y_0) = 0$$

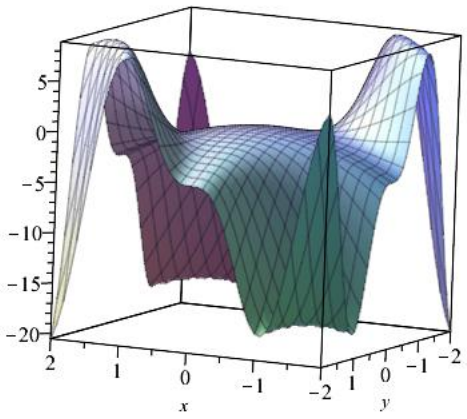
, tačiau funkcija f neturi lokalaus ekstremumo taške (x_0, y_0).

Koeficientas η yra vienas iš normalizavimo parametrų, aprašytas Friedman [2], kur $0 \leq \eta \geq 1$. Duotame straipsnyje, empiriškai pastebėta, kuo η reikškmės mažesnės, tuo modelis tikslesnis. Kita vertus, kuo η maženis, tuo iteracijų skaičius n didesnis, o tai reikalauja daugiau skaičiavimo išteklių. Anot Bühlmann [6], η pasirinkimas nėra svarbus, iki tol, kol jis ganėtinai mažas, t.y, $\eta \in (0; 0, 1]$. Kad geriau tai įsivaizduoti, pavaizduosiu įteracijų kiekio n ir η pasirinkimo įtaką vidutinei kvadratinei paklaidai remiantis Bostono nekilnojamo turto duomenimis [7]. Regresinių medžių modelis buvo išbandytas su skirtingomis η reikšmėmis, maksimalus įteracijų skaičius lygus 10000, o duomenų imtys buvo sumaišytos ir procesas pakartotas 3 kartus. Rezultatai matomi grafike žemiau.



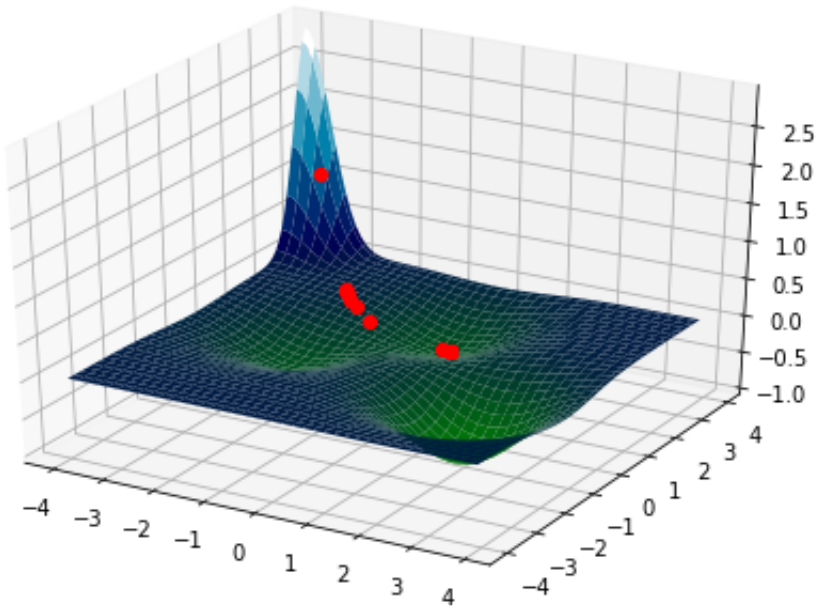
Pav. 9: Čia „RMSE" reiškia vidutinę kvadratinę paklaidą, kuri lygi $\sqrt{(y_i - \widehat{y_i})^2}$ [11]

Kaip matyti iš grafiko, nors mažesnės η reikšmės pradžioje lemia didesnes paklaidas, tačiau atlikus daugiau iteracijų, tikslumas gerėja ir galiausiai pasiekiamas funkcijos minimumas, po kurio didėjant iteracijų skaičiui, tikslumas vėl ima mažėti. Visgi, dažnu atveju, mažas η koeficientas nebūtinai lemia tikslius rezultatus – algoritmas gali ne konverguoti ir pasiekti balno tašką (angl. „Saddle point"). Bendruoju atveju turi būti viena kryptis, kurios atžvilgiu konkretus taškas tarsi minimumo, o kita – tarsi maksimumo ir tos kryptys nebūtinai ortogonalios ar sutampa su koordinačių ašimis.



Pav. 10: Pavaizduotas $f(x,y) = -(x^2+y^2) + 10sin(xy) - 5sin(2xy)$ grafikas, kur $x, y \in [-2;2]$. Koordinačių pradžioje dalinės išvestinės x ir y atžvilgiu bus lygios nuliui, o žvelgiant x ir y ašių kryptimis turėsime maksimumo tašką, tačiau įstrižainių krypčių požiūriu tas taškas atrodys kaip minimumo.

Kadangi balno taške x_0 gradientas $\nabla = 0$, tai netoli šio taško $\nabla \simeq 0$. Atsižvelgiant į gradientinio nuolydžio formulę (16), turime $x_{k+1} \simeq x_k$. Todėl pasiekus balno tašką, kiekviena papildoma įteracija nepadeda pagerinti rezultatų.



Pav. 11: Matoma, kaip taškai koncentruojasi ties balno tašku.

Todėl svarbu, jog algoritmą būtų galima optimizuoti. O tai galima atlikti tik tada, kai algoritmas konverguoja.

Teiginys 1. (Konvergavimas į lokalų minimumą). *Jeigu $f : \mathbb{R}^d \to \mathbb{R}$ yra du kartus diferencijuojama funkcija ir tenkina griežto balno taško savybę, teigiama, jog gradientinis nuolydis*

$$x_{k+1} = x_k - \alpha \nabla f(x_k) \tag{15}$$

su atsitiktiniu iteravimu ir sąlyginai mažu mokinimosi koeficientu α beveik tikrai konverguos į lokalų minimumą [10].

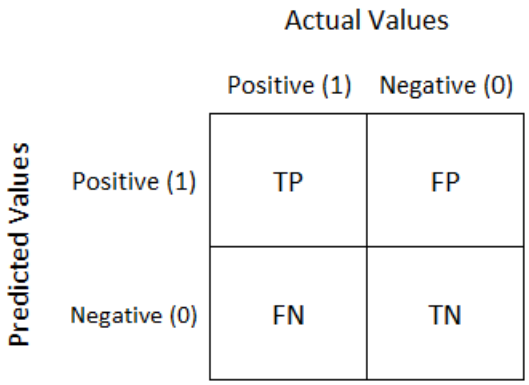
Šio itin svarbaus teiginio įrodymų aš nepateiksiu, bet jie randami šiame straipsnyje [10].

2.5 Modelio analizė

Kai modelis yra išbandomas ant tam tikrų testinių duomenų, svarbu turėti įrankius, kuriais būtų galima palyginti prognozavimo rezultatus su tikrosiomis reikšmėmis. Tai padeda įvertinti modelio tikslumą bei palyginti su kitais modeliais. Įrankių tai atlikti yra įvairių, tačiau šiame darbe nauduosiu šiuos: *Netikrumo matrica, AUC - ROC ir Precision - Recall kreives.*

2.5.1 Netikrumo matrica

Sprendžiant klasifikavimo užduotį, algoritmo galutinis rezultatas apibrėžtas aibėje $[0, 1]$. Supaprastinimo dėlei, yra naudojama minėta slenkstinė vertė, kuri įprastai lygi 0.5, bet priklauso intervalui $[0, 1]$. Slenkstinė vertė žymi tam tikrą tyrimo pasitikėjimo lygį. Taigi, pritaikius įprastą slenkstinę vertę 0.5, prognozės rezultatai apibrėžti aibėje $\{1, 0\}$, kur 1 reiškia spėjimą, jog hipotezė pasitvirtins, pavyzdžiui klientas bankrutuos, o 0 – hipotezė nepasitvirtins. Tada rezultatai lyginami su tikrosiomis reikšmėmis ir gaunama 2×2 *netikrumo matrica*, pavaizduota apačioje.



Pav. 12: Netikrumo matrica ir keturios galimos reikšmės. Stulpeliai žymi žinomas reikšmes, eilutės – spėjamas.

Duotoje matricoje, žymenys atitinkamai reiškia:

- *TP* – *true positive*, prognozė, jog klientas bankrutuos yra teisinga.
- *FP* – *false positive*, prognozė, jog klientas bankrutuos yra neteisinga. Dar vadinama, pirmos rūšies klaida.
- *FN* – *false negative*, prognozė, jog klientas nebankrutuos yra neteisinga. Dar vadinama, antros rūšies klaida.
- *TP* – *true negative*, prognozė, jog klientas nebankrutuos yra teisinga.

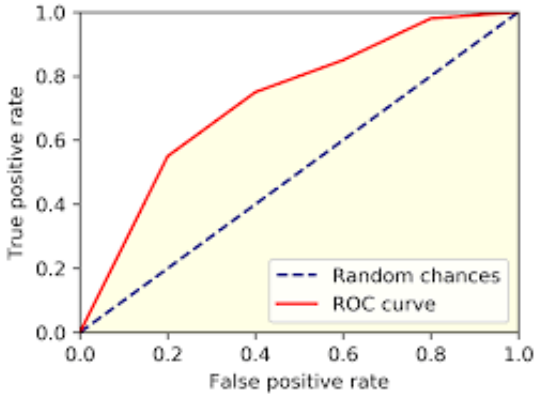
Gauti skaičiai pagrindinėje įstrižainėje reiškia teisingas prognozes, šalutinėje – neteisingas. Iš šių dydžių gaunami įvairūs išvestiniai rodikliai, padedantys vertinti modelio tikslumą.

- Preciziškumas (angl. *precision*) lygus $\frac{TP}{TP+FP}$
- Atšaukimo statistika (angl. *recall*) lygus $\frac{TP}{TP+FN}$
- Tikslumas (angl. *accuracy*) lygus $\frac{TP+TN}{TP+FP+FN+TN}$
- FP santykis (angl. *false positive rate*) lygus $\frac{FP}{FP+TN}$
- F rodiklis (angl. *F-measure*) lygus $2\frac{\text{precision}\times\text{recall}}{\text{precision} + \text{recall}}$, įvertinantis tiek *precision*, tiek *recall* koeficientus.

Kaip minėta, netikrumo matrica atvaizduoja rezultatus, priklausomus nuo pasirinktos slenkstinės vertės dydžio, tačiau įdomu, koks mūsų prognozės jautrumas slenkstinės vertės pokyčiui. Tam naudojama *AUC-ROC kreivė*.

2.5.2 AUC-ROC

Apibrėžimas 10. *ROC kreivė* yra kreivė, atvaizduota plokštumoje, kur Y ašyje yra TP reikšmės, o X – FP reikšmės. Ši kreivė parodo prognozės jautrumą šiems dviems rodikliams, t.y, kaip keičiant slenkstinės vertės koeficientą, keičiasi TP ir FP reikšmių pasiskirstymas.



Pav. 13: Raudona linija – ROC kreivė, o geltonai pažymėtas plotas po ja - AUC [5].

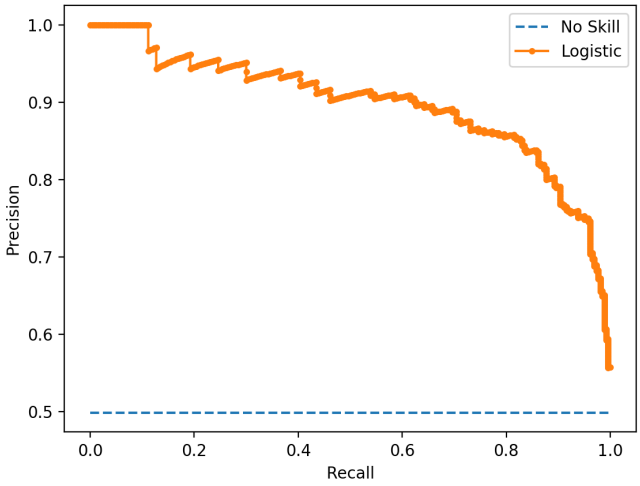
Verta paminėti tam tikrus kreivės taškus. Taškas (0,0) žymi strategiją, kai neatliekama nei viena TP bei FP prognozė. Priešinga strategija apibrėžiama taške (1,1). Tašką (0,1) galima laikyti geriausia strategija, kai teisingai klasifikuojami teigiami atvejai ir nepadaryta

nei viena pirmos rūšies klaida. Priešinga strategija taške (1,0). Įstrižaine einanti linija žymi atsitiktinę, neutralią strategiją. Teigiama, jog viena strategija yra pranašesnė už kitą, jeigu taškas, reprezentuojantis vieną strategiją, yra labiau į šiaurės vakarus nutolęs nei kitas, t.y – TP santykis didesnis, FP mažesnis arba tenkinamos abi sąlygos. Iš kitos pusės, jeigu strategiją žymintis taškas yra kairiajame grafiko kampe, jis laikomas *konservatyviu* – TP prognozės atliekamos tik su įtikinamais įrodymais, todėl mažiau FP prognozių, tačiau tuo pačiu ir mažesnis TP santykis. Priešinga liberali strategija dešiniajame plokštumos kampe. Plotas, esantis po ROC kreivė, yra vadinamas AUC *(angl. Area under an ROC curve).* [5]

Apibrėžimas 11. *AUC, arba plokštuma po ROC kreive* yra svarbus statistinis rodiklis įgyjantis reikšmes intervle $[0, 1]$, ekvivalentus tikimybei, jog vienas klasifikavimo modelis atsitiktinai pasirinktą teigiamą pavyzdį TP klasifikuos dažniau nei atsitiktinai pasirinktą neigiamą pavyzdį FP [5].

2.5.3 Precision-recall kreivė

Apibrėžimas 12. *Precision-recall kreivė* yra kreivė atvaizduota plokštumoje, kur Y ašyje yra paskaičiuotos santykio *Precision* reikšmės, o X – *Recall* reikšmės. Ši kreivė parodo jautrumą šiems dviems santykiams, t.y, kaip keičiant slenkstinės vertės koeficientą, keičiasi *precision* ir *recall* reikšmių pasiskirstymas.



Pav. 14: Precision-recall funkcija. Čia taškas (1,1) žymi idealų prognozavimą, tiek FP, tiek $FN = 0$, t.y nėra nei pirmos nei antros rūšies klaidų. *No Skill* linija žymi strategiją, kurios prognozės yra atsitiktinės arba lygios konstantai. Ši strategija nesugeba klasifikuoti duomenų

Klausimas, pagal kurią kreivę reikėtų vertinti modelį? Kas jeigu vienas modelis atlieka geresnės prognozes *ROC* plokštumoje, o kitas modelis *Precision-recall* plokštumoje?

Teorema 1. *Fiksuotam skaičiui teigiamų ir neigiamų prognozių, viena kreivė (modelis) dominuoja kitą kreivę ROC plokštumoje t.t.t jeigu pirmoji dominuoja antrąją Precision-recall plokštumoje.*

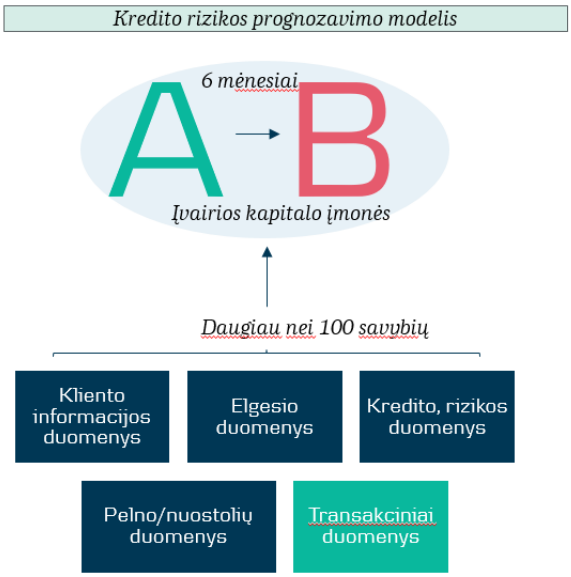
Įrodymas yra šiame straipsnyje [4].

3 Praktinė dalis

Šioje dalyje aprašysiu sukurto modelio logiką, pabandysiu ją paaiškinti ir pavaizduoti, aprašysiu kaip duomenys buvo ruošiami modeliavimui ir aprašysiu išbandytą modelį, išskirdamas du skirtingus atvejus pagal pasirinktas parametrų reikšmes. Galiausiai, įvertinsiu modelio rezultatus pagal minėtus metodus.

3.1 Modelio logikos aprašymas

Kaip minėta pradžioje, modelio tikslas įvertinti, ar kliento reitingas per ateinančius 6 mėnesius pasikeis iš A į B reitingą remiantis *XGboost* algoritmu. Vizualiai tai atrodo taip:



Pav. 15: Modelio šablonas.

Ką reiškia pereiti iš A į B? Reitingavimo metodika skiriasi įvairiose įmonėse, kartais reitingai žymimi AAA arba $A+$, tačiau šiuo atveju reitingų aibę sudaro tokie reitingai: A1, A2, A3, A4, A5, A6, A7, B1, B2, B3, B4. Taigi mus dominanti imtis bus A1, A2, A3, A4, A5, A6, A7. Kaip minėta anksčiau, kiekvienas reitingas atitinka tam tikra tikimybė bankrutuoti, pavyzdžiui B4 žymi tai, jog klientas bankrutavo, todėl jo PD = 1. Kodėl pasirinkta prognozuoti perėjimą iš A į B? Kaip taisyklė, klientai, kurių reitingas žemesnis nei A6, yra padidintos rizikos zonoje, iš jų reikalaujama papildomų finansinių, apyvartos rodiklių ataskaitų ar net tenka papildomai skolinti lėšų. Todėl svarbu iš anksto bandyti įvertinti ateities galimas rizikas ir papildomus kaštus. Kaip bus mokomasi? Tai panašu į pradžioje aprašyta regresijos uždavinį:

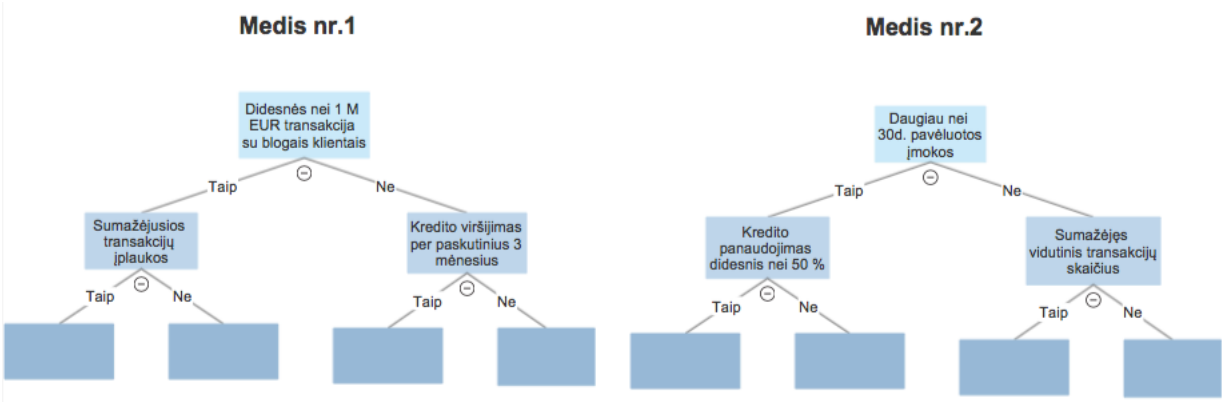
- Turime daug įvairių kredito savybių x_ij, tokių kaip kredito rizikos duomenys, pelno/nuostolių duomenys ir t.t.
- Žinomas rezultatas y_i yra 0, jeigu klientas turi A reitingą ir 1, jeigu jo reitingas B, t.y

mus domina perėjimas iš būsenos 0 į 1 ir atvirkščiai.

- Prognozė $\widehat{y_i}$ šiuo atveju lygi tikimybei, jog klientas yra reitinguotas A arba B po šešių mėnesių.

- Reikia rasti svorio koeficientus w_j, kuriais įvertinsime skirtingas rizikos savybes. Kadangi modelis naudoja regresinius medžius, ieškosime funkcijos f_t reikšmių, kurios moka įvertinti lapo svorį, o tuo pačiu ir medžio struktūrą, optimizuodami $\mathcal{F}^{(t)}$ funkciją.

Kaip atrodys modelio regresinis medis ir kaip jis mokinsis? Apačioje pateiktas poros atšakų „medelis", kadangi atvaizduoti galutinių medžių struktūrą yra sudėtinga.



Pav. 16: Algoritmas skirsto klientus į atitinkančius tam tikrus kriterijus arba ne. Optimaliausia kriterijaus, arba *split* taisyklė (10), šiuo atveju transakcijos dydis ar kredito viršijimo laikotarpis, yra nustatoma įteratyviai, optimizuojant (10) formulę. Kai bus surasta geriausia medžio struktūra, bus pridedamas sekantis medis ir t.t, t.y iki tol, kol pasieksime pasirinktą maksimalų medžių kiekį arba naujas medis nebeturės įtakos prognozės gerinimui.

Modelio logikos dalį papildomai paaiškins duomenų analizės skyrius.

3.2 Duomenų analizė

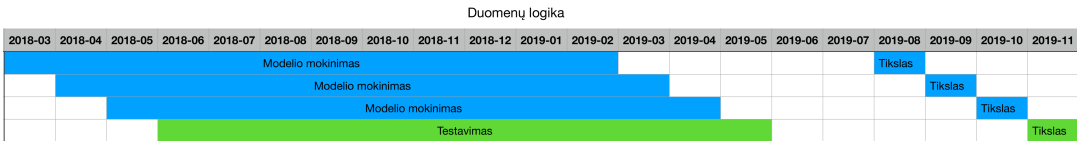
Duomenų analizės ir programavimo etapą sudarys savybių paruošimas modeliavimui: duomenų išskaidymas į apmokymo ir testavimo duomenis bei apmokymo intervalų parinkimas ir logika. Iš programinės pusės, prie duomenų savybių paruošimo neteko prisidėti, nes didžioji dalis duomenų yra iš kito skyriaus ir jų paruošimui reikalingos duomenų inžinerijos žinios,

todėl pagrindinė dalis buvo duomenų sujungimas ir modelio logikos kūrimas. Duomenų savybių naudotas intervalas lygus vieneriems metams, atitinkamai, rodikliai yra agreguojami skirtingais periodais: vieni rodikliai, kaip, pavyzdžiui, kredito rizikos dydis, atspindi paskutinio mėnesio duomenis, kiti, pavyzdžiui, vartojimo kredito viršijimo svyravimas, atspindi 12 mėnesių vidurkio reišmę. Apačioje pateikta dalis savybių ir kokie intervalai naudojami:

Nr.	Savybės	Pasirinktas periodas (vidurkis)
1	Transakcijų kiekis su blogais klientais	12 mėnesių
2	Dienų skaičius esant banko klientu	Paskutinis mėnuo
3	Vidutinis kredito rizikos dydis	12 mėnesių
4	Transakcijų apimtis su blogais klientais	12 mėnesių
5	Kredito perviršis	12 mėnesių
6	Kredito rizikos dydis	Paskutinis mėnuo
7	Mėnesinių transakcijų skaičiaus svyravimas	12 mėnesių
8	Kredito panaudojimo procentas	12 mėnesių
9	Vidutinis lizingo rizikos dydis	12 mėnesių
10	Mėnesinių transakcijų sumos svyravimas	12 mėnesių
11	Vidutinė kapitalo grąža	12 mėnesių
12	Pavėluota įmoka (tarp 0 ir 5 dienų)	12 mėnesių
13	Vartojimo kredito viršijimas	Paskutinis mėnuo
14	Vartojimo kredito viršijimo svyravimas	12 mėnesių
15	Sąskaitų skaičius	12 mėnesių

Pav. 17: Savybių pavyzdžiai ir agregavimo periodai.

Mašininio mokymosi modeliuose įprasta skaidyti turimus duomenis į mokinimosi ir testavimo. Nėra bendro sutarimo, kokio dydžio proporcija geriausia, tačiau dažniausiai 70 procentų visos imties sudaro mokinimosi duomenys, likusieji 30 procentų skirti testavimui. Mokinimosi duomenų imtis bus sudaryta iš trijų mėnesių slenkančio periodo, kuriam naudojamos agreguotos duomenų savybės x_ij. Šios logikos priežastis – kai kurios pramonės šakos yra sezoniškos - t.y jų rodikliai priklausomai nuo metų laiko, klimato sąlygų ar kitų priežasčių yra skirtingi. Pavyzdžiui, žemės ūkio pelno rodikliai yra jautrūs sezoniškumui ar virusinėms ligoms. Duomenys į mokinimosi ir testavimo yra išskaidomi pradžioje, kad tas pats klientas nepatektų į abejas imtis. Paruoštuose duomenyse, priklausomai nuo paskutinės imties datos, pridėjus šešis mėnesius, yra pridedamas tikslo – prognozės stulpelis, kuriame modelis pateiks prognozuotą $\widehat{y_i}$ reikšmę.



Pav. 18: Kaip matoma, duomenys išskaidyti į skirtingus intervalus, siekiant mažesnės tarpusavio koreliacijos.

Tada, iš paruoštų duomenų, pašalinau klientus, kurie yra valstybei priklausantys ar gaunantys dideles subsidijas – šie klientai, kaip taisyklė, dažniausiai nebankrutuoja, jų reitingai yra labai aukšti, kaip A1, A2, tačiau jų finansiniai rezultatai gali neatitikti duotų reitingų ir todėl modeliuojant tam tikroms savybėms gali būti priskirti klaidingi svoriai. Taip pat pašalinami klientai, kurie neturi informacijos prognozės dienai – jie gali būti arba bankrutavę arba neturintys kredito pozicijos.

3.3 Testavimas

Šiame skyriuje bus išbandomi du skirtingi modelio algoritmai parametrų atžvilgiu ir pateikti jų rezultatai. Vėliau gauti rezultatai bus lyginami ir vertinami pagal minėtus kriterijus. Darbe naudoto kodo dalis pridėta prieduose. Modelio testavimui yra naudojama komanda *XGBClassifier*, kurioje galima pasirinkti įvairius paremetrus, apie kuriuos daugiau rašoma dokumentacijoje [13].

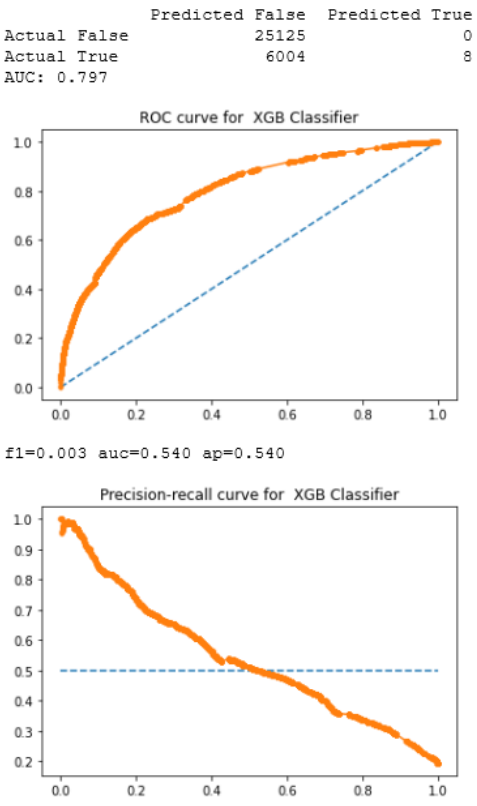
3.3.1 Modelis nr.1

Pirmojo modelio parametrų aprašas apačioje:

```
model = xgb.XGBClassifier(learning_rate=0.01,
                          n_estimators=500,
                          max_depth=5,
                          min_child_weight=0,
                          gamma=0,
                          reg_lambda = 0,
                          reg_alpha = 0,
                          subsample=0.6,
                          colsample_bytree=0.8,
                          objective='binary:logistic',
                          nthread=4,
                          scale_pos_weight=1,
                          seed=27)
```

```
model.fit(x_train, y_train)
return model
```

Kaip matoma šis modelis nėra reguliariziuotas, nes parametrai γ, λ, α, bei teorijoje aprašytas parametras *min child weight*, kuris apibrėžia minimalią galimą svorių sumą lape, yra lygūs 0. Parametras *objective = 'binary: logistic'* žymi, jog sprendžiama klasifikacijos problema. Gauti tokie modelio rezultatai:



(1) Viršuje netikrumo matrica ir ROC kreivė. Čia netikrumo matricoje *Predicted false* reiškia prognozę, jog klientas po šešių mėnesių turės A reitingą, o *Predicted true* – turės B reitingą. Apačioje Precision-recall kreivė.

	feature_name	weight
1	reminder_1_sum	0.136
0	daysinthebank_max	0.122
4	exposure_flag_sum	0.076
6	ej_forskud_flag_sum	0.056
2	delinquencydays_61_90_sum	0.054
11	customer_financial_trouble_flag_sum	0.054
16	utilisation_share_mean	0.046
15	exposure_mean	0.038
20	saldo_dkk_stddev	0.036
13	imum_excess_days_mean	0.030

(2) Duomenų savybių $x_i j$ svorio įvertinimas atsižvelgiant į lapų svorį w_j.

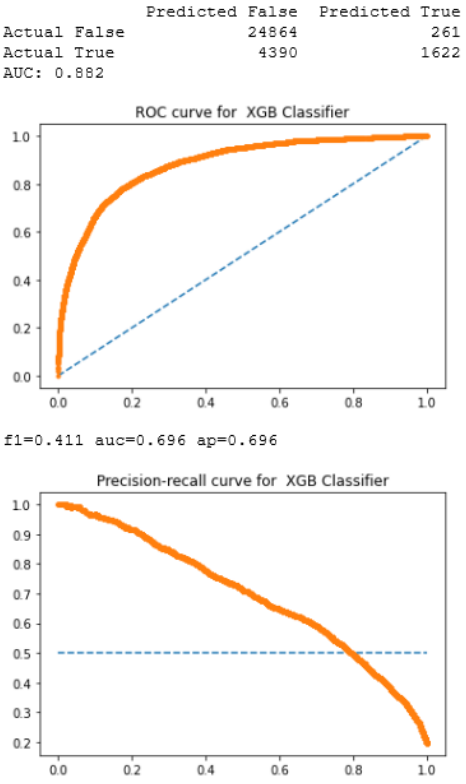
Pav. 19: Pirmojo modelio vertinimo grafikai bei lentelė.

3.3.2 Modelis nr.2

Antrojo modelio parametrų aprašas apačioje:

```
model = xgb.XGBClassifier(learning_rate=0.01,
                          n_estimators=500,
                          max_depth=5,
                          min_child_weight=1,
                          gamma=0.3,
                          reg_alpha = 2,
                          subsample=0.6,
                          colsample_bytree=0.8,
                          objective='binary:logistic',
                          nthread=4,
                          scale_pos_weight=1,
                          seed=27)
  model.fit(x_train, y_train)
```

Antrajame modelyje buvo atsižvelgima į reguliarizacijos funkciją, padidinus koeficientų γ, α, bei *min child weight* reikšmes. Gauti tokie rezultatai:



(1) Viršuje netikrumo matrica ir ROC kreivė. Čia netikrumo matricoje *Predicted false* reiškia prognozę, jog klientas po šešių mėnesių turės A reitingą, o *Predicted true* – turės B reitingą. Apačioje Precision-recall kreivė.

	feature_name	weight
17	roactotcostpct_mean	0.068385
85	transactions_amount_dkk_with_bad_customers_12m	0.058264
16	utilisation_share_mean	0.047528
0	daysinthebank_max	0.042673
144	exposure_current	0.042194
84	transactions_count_with_bad_customers_12m	0.038843
15	exposure_mean	0.038159
4	exposure_flag_sum	0.024345
11	customer_financial_trouble_flag_sum	0.023319
8	customer_on_watch_flag_sum	0.022841

(2) Duomenų savybių $x_i j$ svorio įvertinimas atsižvelgiant į lapų svorį w_j.

Pav. 20: Antrojo modelio vertinimo grafikai bei lentelė.

3.3.3 Modelių rezultatai ir palyginimas

Rezultatai bus lyginami pagal teorijoje aptartus rodiklius bei aptariami tam tikrų svorio koeficientų skirtumai.

3.3.4 Rodiklių analizė

Rodiklis	Modelis nr. 1	Modelis nr. 2	Palyginimas
Precision	1	0.861	Modelis nr 1.
Recall	0.001	0.27	Modelis nr 2.
Accuracy	0.807	0.85	Modelis nr 2.
FP rate	0	0.01	Modelis nr 1.
F measure	0.003	0.411	Modelis nr 2.

Pav. 21: Modelių palyginimas.

Apibendrinant viršuje pateiktus rodiklius, *AUC-ROC* ir *Precision-recall* grafikus, priėjau tokių išvadų:

- Nors pirmojo modelio preciziškumas didesnis bei *FP* santykis santykinai mažesnis nei antrojo modelio, tačiau tai lėmė *FP* rodiklis lygus nuliui. Didelės *Precision*, bet mažos *Recall* rodiklių reikšmės lemia sąlyginai didelį tikslumą, tačiau prastą gebėjimą klasifikuoti daugelį sudėtingų atvėjų, šiuo atveju pirmasis modelis tik 8 klientams prognozavo reitingo pokytį iš A į B, nors pagrindinis tikslas ir yra prognozuoti kliento reitingo suprastėjimą, todėl pirmojo modelio nauda itin maža.
- Vertinant modelius *AUC-ROC* ir *Precision-recall* kreivių atžvilgiu, antrasis modelis dominuoja pirmąjį remiantis pirmąja teorema [4]. Pirmasis modelis yra labiau jautrus slenkstinės vertės pokyčiui – tai matyt iš *Precision-Recall* grafiko, kadangi pirmasis modelis anksčiau kerta *No Skill* liniją. Be to, antrojo modelio *AUC* rezultatas didesnis – jis geriau sugeba klasifikuoti kliento reitingo pokytį.

3.3.5 Svorio koeficientų palyginimas

Kaip rašyta teorinėje dalyje, reguliarizacija padeda mažinti duomenų savybių svorio koeficientus. Tai matome palyginę pirmąjį modelį ir antrąjį, kur pirmajame reguliarizacijos koeficientai yra lygūs nuliui. Be to, verta paminėti, jog ne visada didinant λ ar α reikšmes,

svorių koeficientai mažės, nes jie taip pat priklauso nuo H_i ir G_i reikšmių (8), kurias galime kontroliuoti. Palyginimo išvados:

- Skirtingų savybių svoriai sumažėjo skirtingu santykiu. Labiausiai sumažėjo šių savybių svoriai: kliento dienų skaičius banke bei priminimai dėl įmokų. Nors savybių svorių vertinimas iš dalies yra subjektyvus, tačiau pirmojo modelio atveju buvo priskirti per dideli svoriai, ypač kliento dienų skaičiui esant banko klientu.

3.4 Praktikos rezultatų apibendrinimas

Nors antrojo modelio tikslumas gana aukštas, manau, jog jis dar nėra tinkamas prognozavimui dėl šių priežasčių:

- Didelis *FP* rodiklis – daugelio klientų, kuriems prognozuojamas A reitingas po šešių mėnesių, tikrasis reitingas yra B. Todėl modelis nesugeba pakankamai įvertinti svorių, kurie lemia kliento finansinės būklės prastėjimą.

- Klientas gali būti didelės kompanijos dukterinė imonė, todėl galimi nuostoliai būtų dengiami nepaisant blogų finansinių rezultatų, o tai ne visada atitiktų realius savybių svorius.

- Reikėtų nuodugnesnės savybių analizės. Pavyzdžiui, aukštas kredito panaudojimo santykis kartais priklauso nuo industrijos – vienur tai reiškia rizika, o pavyzdžiui lizingo ar nekilnojamo turto klientams aukštas santykis nėra rizikos indikatorius.

- Koronos viruso pandemija iškėlė klausimą. Galbūt ateityje vertinant kliento riziką, reikėtų atsižvelgti į tai, kokias pareigas užima klientas arba kokiai industrijai priklauso įmonė – reikia atsižvelgti į verslo ar paslaugų būtinumą pandemijos laikotarpiu.

4 Išvados ir rekomendacijos

Apibendrinant, būtų galima teigti, jog modelio logika ganėtinai neblogai sukonstruota ir prognozės sąlyginai tikslios, tačiau mano manymu yra kita medalio pusė. Fundamentinė problema kyla tuomet, kai naudojama vis daugiau medžių – nors tai dažniausiai lemia

geresnius rezultatus, tačiau tuo pačiu modelis tampa kompleksiškesnis ir sunkiau interpretuojamas, to pasekoje, sunku įvertinti kiekvieną savybę. Mano manymu, praktinėje dalyje galėtų būti daugiau parametrinės analizės, bei svorio koeficientų nuodugnesnių paaiškinimų – nepavyko pilnai įgyvendinti teorijoje aprašytų metodų. Be to, bakalauro darbas palietė tik dalį algoritmo funkcionalumo, todėl yra daug vietos tobulėjimui. Darbo metu pavyko įgauti įvairių teorinių bei programavimo žinių, kurios pravers ateityje.

5 Conclusion

All things considered, it is possible to claim that the predictions were conditionally precise and model's logic was constructed in a sound way, yet there is another side of the coin. Fundamental problem occurs when we are building more and more trees – though it results in better predictions, on the other hand, the model becomes more complex and less interpretable. Therefore, it becomes difficult to explain every feature and its weight to the prediction. In my opinion, practical side lacks parametrical analysis and explanation of learned feature weights, therefore, not all of the theoretical part was covered in practice. Besides, as only part of algorithm's functionality was explained, there is plenty of room for improvement.Throughout the process, I gained useful theoretcal and programming knowledge which will be useful in the future.

6 Nuorodos

Literatūros sąrašas

[1] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[2] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York, 2001.

[3] Hui Zou and Trevor Hastie. "Regularization and variable selection via the elastic net". In: *Journal of the Royal Statistical Society: Series B (statistical methodology)* 67.2 (2005), pp. 301–320.

[4] Jesse Davis and Mark Goadrich. "The relationship between Precision-Recall and ROC curves". In: *Proceedings of the 23rd International Conference on Machine Learning*. 2006, pp. 233–240.

[5] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern Recognition Letters* 27.8 (2006), pp. 861–874.

[6] Peter Bühlmann and Bin Yu Boosting. "Wiley Interdisciplinary Reviews". In: *Computational Statistics* 2.1 (2010), pp. 69–74.

[7] Moshe Lichman et al. *UCI machine learning repository, 2013*. 2013.

[8] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data mining*. 2016, pp. 785–794.

[9] Aarshay Jain. *A Complete Tutorial on Ridge and Lasso Regression in Python*. 2016. URL: `https://www.analyticsvidhya.com/blog/2016/01/ridge-lasso-regression-python-complete-tutorial/`.

[10] Jason D Lee et al. "Gradient descent only converges to minimizers". In: *Conference on Learning Theory*. 2016, pp. 1246–1257.

[11] David Li-Bland. *Saddle Points and Stochastic Gradient Descent*. 2018. URL: `https://davidlibland.github.io/posts/2018-11-10-saddle-points-and-sdg.html`.

[12] AI Pool. *Bias-Variance Tradeoff in Machine Learning*. 2019. URL: `https://ai-pool.com/a/s/bias-variance-tradeoff-in-machine-learning`.

[13] XGBboost developer. *XGBoost Documentation*. 2020. URL: `https://xgboost.readthedocs.io/en/latest/`.

[14] Mirko Stojiljkovic. *Logistic Regression in Python*. 2020. URL: `https://realpython.com/logistic-regression-python/`.

[15] Wikipedia. *Regularization (mathematics)*. 2020. URL: https://en.wikipedia.org/wiki/Regularization_(mathematics.

7 Priedai

Modelio logika

```
import pandas as pd
import pyspark.sql.functions as psf
import xgboost as xgb
from sklearn.model_selection
import train_test_split
from dateutil.relativedelta
import relativedelta
from datetime
import datetime

def run_training_of_ab_model(aggregated_features_on_ip_id_df, columns_to_drop_list,
    month_prediction):

  months_train_list = create_list_of_training_months(last_month = month_prediction)
aggregated_features_on_ip_id_df = \
  load_filtered_by_months_table_into_pandas(df = aggregated_features_on_ip_id_df,
    months_list = [month_prediction] + months_train_list)
train_knids_list, test_knids_list = create_train_and_test_knids_list(df =
    aggregated_features_on_ip_id_df,
  month_prediction = month_prediction)
test_df_pd, train_df_pd = \
  construct_train_and_test_datasets(aggregated_features_df_pd =
      aggregated_features_on_ip_id_df,
    train_knids_list = train_knids_list,
    test_knids_list = test_knids_list,
```

```
    month_prediction = month_prediction,
    months_train_list = months_train_list)
x_train, y_train = prepare_target_and_features(df = train_df_pd,
    columns_to_drop_list = columns_to_drop_list)
model = model_fit(x_train = x_train, y_train = y_train)
return model, test_df_pd, train_df_pd

def run_prediction(aggregated_features_df,
    columns_to_drop_list,
    classification_model,
    columns_to_drop_in_the_output,
    string_columns_list,
    last_month):

  prediction_df_pd = load_filtered_by_months_table_into_pandas(df =
      aggregated_features_df, months_list = [last_month])
x, y = prepare_target_and_features(df = prediction_df_pd, columns_to_drop_list =
    columns_to_drop_list)
y_prediction = model_predict(model = classification_model, x = x)
predicted_df = concatenate_original_df_with_prediction(df = prediction_df_pd,
    y_prediction = y_prediction)
predicted_df = filter_out_irrelevant_customers(df_pd = predicted_df,
  last_month = str(datetime.strptime(last_month, '%Y-%m') -
    relativedelta(months = 6))[: 7])
predicted_df = \
  rearrange_and_drop_irrelevant_columns_in_output_df(df = predicted_df,
    columns_to_drop_in_the_output = columns_to_drop_in_the_output)
predicted_df = cast_string_columns(predicted_df, string_columns_list)
return predicted_df

def create_list_of_training_months(last_month):
```

```
  list_of_dates = pd.date_range(end = pd.to_datetime(last_month), periods = 3, freq
      = 'M')
list_of_dates = list_of_dates.to_period('m').astype(str).tolist()
return list_of_dates

def load_filtered_by_months_table_into_pandas(df, months_list):

  df = df.filter(psf.col('month_target').isin(months_list))
df_pd = pd.DataFrame([])
for month in months_list:
  df_pd = df_pd.append(df.filter(psf.col('month_target') == month).toPandas())
return df_pd

def create_train_and_test_knids_list(df, month_prediction):

  df = df[df['month_target'].isin([month_prediction])][
    ['ip_id', 'pd_score_to_predict']
  ]
train_knids_df, test_knids_df = train_test_split(df,
  stratify = df['pd_score_to_predict'],
  test_size = 0.3,
  random_state = 42)
return train_knids_df.ip_id.tolist(), test_knids_df.ip_id.tolist()

def construct_train_and_test_datasets(aggregated_features_df_pd,
    train_knids_list,
    test_knids_list,
    month_prediction,
    months_train_list):

  test_df_pd = aggregated_features_df_pd[(aggregated_features_df_pd.month_target.
      isin([month_prediction])) &
```

```
    (aggregated_features_df_pd.ip_id.isin(test_knids_list))]
train_df_pd = aggregated_features_df_pd[(aggregated_features_df_pd.month_target.
    isin(months_train_list)) &
  (aggregated_features_df_pd.ip_id.isin(train_knids_list))]
return test_df_pd, train_df_pd

def prepare_target_and_features(df, columns_to_drop_list):

  x = df.drop(columns_to_drop_list, axis = 1)
y = df['ab_flag_to_predict']
return x.astype(float).values, y.astype(float).values

def model_fit(x_train, y_train):
  model = xgb.XGBClassifier(learning_rate = 0.01,
    n_estimators = 500,
    max_depth = 5,
    min_child_weight = 1,
    gamma = 0.1,
    subsample = 0.6,
    colsample_bytree = 0.8,
    objective = 'binary:logistic',
    nthread = 4,
    scale_pos_weight = 1,
    seed = 27)
model.fit(x_train, y_train)
return model

def model_predict(model, x):

  y_prediction = model.predict_proba(x)
return y_prediction
```

```
def concatenate_original_df_with_prediction(df, y_prediction):

  df['prediction'] = y_prediction[: , 1]
df['pd_score_difference'] = df['pd_score_to_predict'] - df['pd_score_current']
return df

def rearrange_and_drop_irrelevant_columns_in_output_df(df,
    columns_to_drop_in_the_output):

  df = df.drop(columns_to_drop_in_the_output, axis = 1)
cols = df.columns.tolist()
df = df[cols[-2: ] + cols[: -2]]
return df

def cast_string_columns(df, column_list):

  df[column_list] = df[column_list].astype(str)
return df

def filter_out_irrelevant_customers(df_pd, last_month):

  df_pd = df_pd[(df_pd.industrygrouplvl2_current != 'Public_Institutions') &
    (df_pd['dt_year_month_max'] == last_month)]
return df_pd
```

Modelio vertinimas

```
import pandas as pd
import sklearn.metrics as skm
import matplotlib.pyplot as plt

def convert_probability_to_zero_one_flag(df, threshold):
```

```
  return pd.to_numeric([1
    if y > threshold
    else 0
    for y in df['prediction']
  ])

def plot_confusion_matrix(validation_df_pd, prediction_threshold):

  y_true = validation_df_pd.ab_flag_to_predict.astype(int)
xgb_confusion_matrix = pd.DataFrame(
  skm.confusion_matrix(y_true, validation_df_pd['prediction'] >
      prediction_threshold),
  columns = ["Predicted False", "Predicted True"],
  index = ["Actual False", "Actual True"]
)
print('Confusion matrix with threshold {} for customers in defined scope'.format(
    prediction_threshold))
print(xgb_confusion_matrix)

def plot_roc_curve(validation_df_pd):

  auc = skm.roc_auc_score(validation_df_pd['ab_flag_to_predict'].astype(float),
      validation_df_pd['prediction'])
print('AUC: %.3f' % auc)
fpr, tpr, thresholds = skm.roc_curve(validation_df_pd['ab_flag_to_predict'],
    validation_df_pd['prediction'])
plt.plot([0, 1], [0, 1], linestyle = '--')
plt.plot(fpr, tpr, marker = '.')
plt.title('ROC curve for XGB Classifier')
plt.show()

def plot_precision_recall_curve(validation_df_pd, threshold):
```

```
  y_prediction = convert_probability_to_zero_one_flag(validation_df_pd, threshold)
y_test = pd.to_numeric(validation_df_pd['ab_flag_to_predict'])
precision, recall, _ = skm.precision_recall_curve(y_test, validation_df_pd['
    prediction'])
f1 = skm.f1_score(y_test, y_prediction)
auc_score = skm.auc(recall, precision)
ap = skm.average_precision_score(y_test, validation_df_pd['prediction'])
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc_score, ap))
plt.plot([0, 1], [0.5, 0.5], linestyle = '--')
plt.plot(recall, precision, marker = '.')
plt.title('Precision-recall curve for XGB Classifier')
plt.show()

def get_feature_importances(xgb_model, feature_names, importance_type):

  importance_values = xgb_model.booster().get_score(importance_type =
      importance_type)
importance_dict = {
  feature_names[i]: float(importance_values.get('f' + str(i), 0.)) for i in
    range(len(feature_names))
}
total = sum(importance_dict.values())
importance_percent_dict = {
  k: v / total
  for k,
  v in importance_dict.items()
}
return (pd.DataFrame(list(importance_percent_dict.items()), columns = ['
    feature_name', importance_type])
  .sort_values(importance_type, ascending = False))
```