



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

MATEMATINĖS STATISTIKOS KATEDRA

Egidijus Griška

**KLIENTŲ ELGSENOS ANALIZĖ
TELEKOMUNIKACIJOS SRITYJE**

**Analysis of customer behavior in the field of
telecommunications**

Baigiamasis magistro darbas

**Duomenų mokslo ir statistikos magistro studijų
programa, valstybinis kodas 6211AX009**

Statistikos studijų kryptis

Vilnius, 2023

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
MATEMATINĖS STATISTIKOS KATEDRA

TVIRTINU

Katedros vedėjas

(Parašas)

doc. dr. Kazimieras Padvelskis
(Vardas, pavardė)

(Data)

Egidijus Griška

KLIENTŲ ELGSENOS ANALIZĖ
TELEKOMUNIKACIJOS SRITYJE

Analysis of customer behavior in the field of
telecommunications

Baigiamasis magistro darbas

Duomenų mokslo ir statistikos magistro studijų
programa, valstybinis kodas 6211AX009

Statistikos studijų kryptis

Vadovas

doc. dr. Nomedą Bratėkovienė
(Pedagoginis vardas, vardas, pavardė)

(Parašas)

(Data)

Vilnius, 2023

Anotacija

Baigiamajame magistro darbe buvo siekiama suprognozuoti telekomunikacijos klientų elgseną. Norint išlaikyti klientus šiame sektoriuje, šis uždavinys tampa itin svarbiu ir aktuali. Magistro darbo literatūros apžvalgoje yra aprašoma dažniausiai taikoma metodologija, sprendžiant panašius binario klasifikavimo uždavinius. Atlikus literatūros analizę nuspręsta analizuoti „XGBoost“ metodą ir palyginti jo tikslumą su kitais mašininio mokymosi metodais. Apžvelgus taikomą metodologiją yra atliekamas skirtingų mašininio mokymosi metodų modeliavimo rezultatų palyginimas po pirminio modeliavimo, kryžminės validacijos, hiperparametrų paieškos, taip pat atliktas palyginimas tarp skirtingų duomenų balansavimo metodų, identifikuojami prasmingi kintamieji, kurie labiausiai daro įtaką klientų elgsenai.

Darbą sudaro 7 dalys: įvadas, literatūros apžvalga, modeliavimo metodų apžvalga, tiriamoji dalis, išvados, literatūros sąrašas ir priedai. Darbo apimtis 52 p. teksto be priedų, 19 iliustr., 11 lent., 30 bibliografinių šaltinių.

Prasminiai žodžiai: klientų elgsenos prognozavimas, dvireikšmis klasifikavimas, mašininis mokymas, XGBoost.

Annotation

The master's thesis aimed to predict the behavior of telecommunications customers. In order to retain customers in this sector, this task becomes extremely important and relevant. The literature review of the master's thesis describes the most commonly used methodology for solving similar binary classification tasks. After the literature analysis, it was decided to analyze the "XGBoost" method and compare its accuracy with other machine learning methods. After reviewing the methodology, comparison between the applied machine learning methods are made after initial simulation, cross-validation, parameter search, as well as a comparison between various class balancing methods and identifying meaningful data that influence the behavior of customers.

The work consists of 7 parts: introduction, review of literature, review of modeling methods, research part, conclusions, bibliography and appendix. Scope of work 52 p. text without appendices, 19 illustrations, 11 tables, 30 bibliographic sources.

Keywords: customer churn prediction, binary classification, machine learning, XGBoost.

Turinys

Įvadas	7
1. Literatūros apžvalga	9
2. Modeliavimo metodų apžvalga	11
2.1. Modelių metodologija	11
2.1.1. Logistinė regresija	11
2.1.2. K-artimiausių kaimynų metodas	12
2.1.3. Sprendimų medis	13
2.1.4. Atsitiktinis miškas	15
2.1.5. Medžių didinimas	16
2.2. Modelio tikslumo metrikos	17
2.3. Modelio įvertinimas kryžminės patikros metodu	19
2.4. Kintamųjų svorių metrikos	20
2.5. Hiperparametrų paieškos metodai	22
2.5.1. Gardelės paieška	22
2.5.2. Atsitiktinė paieška	24
2.6. Klasių balansavimo metodai	24
2.6.1. Atsitiktiniai imties balansavimo metodai	25
2.6.2. SMOTE	25
2.6.3. NearMiss	26
3. Tiriamoji dalis	27
3.1. Duomenų rinkinys, analizė	27
3.1.1. Duomenų rinkinys ir kintamieji	27
3.1.2. Duomenų valymas	28
3.1.3. Skaitinių kintamųjų apžvalga	29
3.1.4. Kategorinių kintamųjų apžvalga	31
3.1.5. Tikslų kintamojo disbalansas	34
3.2. Modeliavimas	36
3.3. Sumaišties matricos	36
3.4. ROC kreivė	37
3.5. Modelių metrikos	38
3.6. Kintamųjų svoriai	38
3.7. Kryžminė patikra	40
3.8. Hiperparametrų paieška	40
3.9. Eksperimentas taikant klasių balansavimo metodus.	47

Išvados	50
Literatūros sąrašas	52

Iliustracijų sąrašas

1.	KNN metodika (Alaliyat, 2023)	13
2.	Sprendimų medis, sprendžiant, ar oras palankus tenisui	13
3.	Atsitiktinio miško prognozė (IBM, 2023)	16
4.	Medžių mokymas paeiliui, gradientiniu metodu, (Boehmke & Greenwell, 2019)	17
5.	K-perlenkimų metodo iliustracija (Patro, 2021)	19
6.	Gardelės paieškos vizualizacija, (Malato, 2021)	23
7.	Atsitiktinės paieškos vizualizacija, (Malato, 2021)	24
8.	SMOTE metodo sintetinių stebinių sugeneravimo iliustracija (Orellana, 2020)	26
9.	Fiktyvūs kintamieji	28
10.	Išskirties pavyzdys	28
11.	Koreliacinė matrica	30
12.	Skaitinių kintamųjų sklaidos diagramos tikslo kintamojo analizei	31
13.	Kategorinio kintamojo kiekių disbalansas ir balansas	32
14.	χ^2 testo p reikšmės tarp dviejų kintamųjų	33
15.	Atsisakiusių klientų kategorinių kintamųjų proporcijos	34
16.	Tikslo kintamojo klasių disbalansas	35
17.	Sumaišties matricos	37
18.	ROC kreivės atskiriems modeliams	38
19.	Kintamųjų reikšmingumo normalizuotos reikšmės	40

Lentelių sąrašas

1.	AUC metrika naudojant skirtingus metodus (Ahmad, Jafar & Aljoumaa, 2019)	10
2.	Sumaišties matrica	18
3.	Apibendrintos tikslumo metrikos	38
4.	Tikslumo metrikos atlikus k-perlenkimų kryžminę paiešką, kai $k = 5$	40
5.	Gardelės paieškos rezultatai su dviem skirtingais kombinacijų skaičiais XG-Boost metodui	42
6.	Gardelės paieškos rezultatai su dviem skirtingais kombinacijų skaičiais atsitiktinio miško metodui	43
7.	Gardelės paieškos rezultatai su dviem skirtingais kombinacijų skaičiais KNN metodui	44
8.	Gardelės paieškos rezultatai su dviem skirtingais kombinacijų skaičiais Logistinės regresijos metodui	45
9.	Tikslumo metrikos ir laikai su atsitiktine paieška, taikant kryžminę validaciją, kai $n = 50$	46

10.	XGBoost modelio tikslumo metrikos taikant klasių balansavimo metodus su balansuotu validavimo rinkiniu	47
11.	XGBoost modelio tikslumo metrikos taikant klasių balansavimo metodus su nebalansuotu validavimo rinkiniu	49

Ivadas

Šiomis dienomis žmonės be telekomunikacijų paslaugų sunkiai įsivaizduotų savo gyvenimą. 2022 m. 93 % Europos namų ūkių turėjo prieigą prie interneto ryšio ((Eurostat, 2022)), todėl natūralu, kad tarp telekomunikacines paslaugas siūlančių įmonių atsiranda vis didesnė konkurencija. Telekomunikacijų srityje, kaip ir kiekvienam versle, siekiant išlaikyti ir didinti pelną, vienas svarbiausių veiksnių yra išlaikyti esamus klientus, bei pritraukti naujus. Yra padaryta analizė (Gupta & Zeithaml, 2006), kurioje yra išsakoma, kad klientų išlaikymas yra svarbesnis ilgesnėje perspektyvoje, nei naujų pritraukimas, dėl kelių veiksnių:

- nereikia investuoti į telekomunikacijų srities rinkodarą, technines stotis,
- esamas klientas yra sukalbamesnis ir pelningesnis,
- ilgaaamžis klientas įprastai yra labiau pelningas, nes su laiku jo išlaikymas lengvėja.

Taigi, čia kyla klausimas: kaip turėtų elgtis tiekėjas, norintis išlaikyti savo klientus? Esminiai veiksniai, dėl kurių klientas galėtų atsisakyti paslaugų, yra geresni konkurentų pasiūlymai. Tam, kad klientas nesutiktų su pateiktu konkurentų pasiūlymu, įmonė galėtų imtis įvairių veiksmų, pavyzdžiui, pasiūlyti papildomų nuolaidų įrangai, kuria naudojasi paslaugomis, išlyginti pasiūlymo kainą, su konkurentų, pasiūlyti geresnes sąlygas gaunamos naudos ar klientų aptartavimo pusėje. Tačiau, drastiškas paslaugų kainų mažinimas bei papildomos išlaidos veda į bendro pelno mažinimą, ko, akivaizdu, sėkmingas verslas nesiekia, todėl yra naudinga nustatyti iš anksto, ar kuris nors specifinis klientas bus linkęs atsisakyti teikiamų paslaugų atskiram laikotarpiui (Hauge & Jamison, 2009). Siekiant priimti teisingus sprendimus, tokiais atvejais padeda klientų elgsenos statistinė analizė, kurios dėka yra prognozuojama, kurie klientai yra labiau linkę atsisakyti turimų paslaugų, kurie ne. Teisingai identifikavus klientus, kurie yra labiau linkę pasitraukti, žmonės gali strateguoti ir siųsti personalizuotus, patrauklesnius pasiūlymus būtent tokiems klientams, ir taip juos išlaikyti. Taip pat galima investuoti į technines stotis, kurios gerina telekomunikacijų paslaugų kokybę, nustačius, kad tam tikrose vietose gyvenantys žmonės yra linkę labiau pasitraukti.

Statistiniai metodai yra dažnai taikomi, prognozuojant klientų atsisakymą. Šie metodai įtraukia statistinius modelius, analizuojančius klientų duomenis ir prognozuojančius klientų atsisakymo tikimybę. Dažniausiai taikomi metodai įtraukia logistinę regresiją, sprendimų medžius, neuroninius tinklus. Modeliai yra apmokami iš istorinių klientų duomenų, į kuriuos įeina įvairūs demografiniai, techniniai, kiti klientą apibūdinantys kintamieji, bei tikslo kintamasis, atsakantis, ar klientas atsisakė paslaugos, ar ne. Naudojant įvairius matematinio modeliavimo metodus ieškomos tendencijos, kurios atsako į klausimą, kurie kintamieji buvo reikšmingiausi, testuojamas modelio tikslumas.

Darbo aktualumas. Konkurencingoje telekomunikacijų rinkoje, rinkos žaidėjų vienas iš svarbiausių tikslų yra didinti pelną, nukonkuruojant oponentus. Klientų elgsenos analizė padeda iš anksto nustatyti klientų polinkius ir atlikus tam tikrus veiksmus, pasiekti savo tikslą.

Tikslas. Taikant matematinius bei statistinius modelius apskaičiuoti, kurie klientai yra labiau linkę atsisakyti turimų telekomunikacijos paslaugų, nustatyti tai lemiančius veiksnius, nustatyti ir įvertinti geriausiai prognozuojantį statistinį metodą.

Uždaviniai. Siekiant įgyvendinti užsibrėžtą tikslą, yra iškeliami tokie uždaviniai:

1. Atlikti mokslinių darbų ir literatūros, kuriuose analizuojama ir modeliuojama klientų elgsena, literatūrinę apžvalgą.
2. Išanalizuoti taikomus matematinius metodus, bei panaudoti juos realiam telekomunikacijos duomenų modeliavimui.
3. Nustatyti reikšmingus veiksnius ir geriausius metodus, palyginti jų tikslumą.

1. Literatūros apžvalga

Telekomunikacijų kompanijos susiduria su dideliu pelno praradimu dėl klientų pasitraukimo, kompanijos ieško įvairių būdų, siekiant juos išlaikyti. Pastaraisiais metais mašininio mokymosi algoritmų naudojimas didėjo, norint tiksliau prognozuoti klientų atsisakymą telekomunikacijų sektoriuje. Šie algoritmai analizuoja klientų duomenis ir identifikuoja tendencijas, kurios indikuoja klientų atsisakymo tikimybę. Mašininio mokymosi algoritmų naudojimas klientų atsisakymo prognozavimui turi potencialo padėti telekomunikacijų kompanijoms imtis priemonių, kad būtų išvengiama klientų praradimo, taip sumažinant pajamų nuostolius ir didinant klientų išlaikymą. Šioje literatūros apžvalgoje bus išnagrinėti naujausi tyrimai, susiję su klientų atsisakymų telekomunikacijų srityje, sutelkiant dėmesį į metodus, naudojamus kuriant prognozuojamus modelius ir palyginamąsias metrikas.

Ahmad, Jafar and Aljoumaa (2019) savo darbe tyrinėja mašininio mokymosi algoritmų panaudojamumą klientų atsisakymo prognozėms Sirijos telekomunikacijų įmonei, prognozuojant mobilių paslaugų klientų pasitraukimą. Autoriai darbą pradeda diskutuodami apie klientų atsisakymo prognozės svarbą šioje industrijoje ir tikslų prognozavimo modelių poreikį. Tada, jie aprašo savo tyrime naudotą metodiką, kuri apima imties duomenų padalijimą į mokymosi ir testavimo duomenų rinkinius. Savo darbe, autoriai naudojo demografinius, naudojamumo (skambučių, interneto), prenumeratos tipo (plano tipo, periodiškumo) duomenis, taip pat pateikė naudotus duomenų pertvarkymo etapus:

1. papildė trūkstamas reikšmes vidurkio arba medianos reikšmėmis,
2. atliko duomenų normalizavimą,
3. kategorinių kintamųjų transformaciją į binarius kintamuosius,
4. pritaikė pagrindinių komponentų analizę, siekiant sumažinti kintamųjų skaičių,
5. siekiant išlyginti klasių disbalansą panaudojo imčių metodus, sumažinant arba padidinant trūkstamos arba perteklinės klasės stebinių skaičių.

Toliau panaudoję keturis skirtingus medžiais grįstus algoritmus t.y. sprendimų medį, atsitiktinius miškus bei gradientinio didinimo metodus GBM ir XGBoost, autoriai prognozavo klientų paslaugų atsisakymo polinkį. Autoriai teigia, jog patys svarbiausi veiksniai duomenyse jų modelių sėkmėje buvo kintamųjų paruošimas, įtraukiant socialinės medijos naudojamumą. Analizuojant 1 lentelę, XGBoost algoritmas buvo pripažintas efektyviausiu metodu, nedarant jokio klasių balanso, kurio AUC reikšmė siekė 93.3 %.

1 lentelė. AUC metrika naudojant skirtingus metodus (Ahmad, Jafar & Aljoumaa, 2019)

Technique used for unbalanced dataset	XGBOOST (%)	GSM (B) (%)	Random Forest (%)	Decision Tree (%)
Oversampling	92	90.01	84.2	76.25
Undersampling	93.12	90.21	87.76	83
Without balancing	93.3	90.89	78.47	72.2

Bogaert and Delaere (2023) savo darbe pristato ansamblinių metodų, naudojamų klientų atsisakymui vertinti, palyginamąją analizę. klientų atsisakymo prognozavimo uždavinyje įvairiuose sektoriuose. Ansambliniai metodai mašiniame mokyme kombinuoja kelis skirtingus modelius siekiant pagerinti prognozavimo tikslumą ar siekiant sukurti tikslesnį ir stipresnį rezultatą negu bet kuris kitas individualus modelis (Zhou, 2012). Autoriai naudojo įvairius duomenų rinkinius iš įvairių sektorių (telekomunikacijų, banko, laikraščių), jiems pritaikant įvairius ansamblių metodus, kaip didinimas, pakavimas, krovimas kartu su individualiais klasifikatoriais kaip sprendimų medžiai, atraminių vektorių klasifikatorius ir neuroniniai tinklai. Darbe yra pažymima, jog ansamblių metodai pranoksta individualius klasifikatorius, o iš homogeninių ansamblių metodų labiausiai išsiskiria XGBoost algoritmas, kuris jau buvo paminėtas (Ahmad, Jafar & Aljoumaa, 2019) šaltinyje.

Vafeiadis, Diamantaras, Sarigiannidis and Chatzisavvas (2015) darbas siekia įvertinti populiarių mašinių mokymo algoritmų tikslumą klientų atsisakymo uždavinyje telekomunikacijų srityje. Buvo vertinami penki klasifikavimo metodai - dviejų sluoksnių atgalinio platavimo tinklas, sprendimų medis, atraminiai vektoriai, naivusis Bajesas ir logistinė regresija su viešai prieinamais telekomunikacijų duomenimis. Klasifikatoriai buvo bandomi be ansamblių didinimo ir neuroninio tinklo, tačiau medžių metodai pasirodė geriausiai. Tada trim likusiems klasifikatoriams buvo pritaikytas didinimo AdaBoost algoritmas (Freund & Schapire, 1997), kuris pagerino visų trijų modelių tikslumą. Padidintas SVM klasifikatorius pasirodė, kaip geriausias klasifikatorius su 97 % tikslumu ir 84 % F-metrika. Tyrimas parodė didinimo metodų taikymo pranašumus ir siūlo patyrinėti kitus didinimo metodus, ne tik AdaBoost.

López Oliva, Egea, Museros and Sanz (2021) darbas, kaip ir pastarieji siūlo naudoti mašininio mokymosi metodus klientų pasitraukimo aptikimui. Šiame darbe jis prognozuoja Ispanijos internetinės televizijos klientų pasitraukimą, jame yra naudojami demografiniai, prenumeratos, mokėjimų, naudojamumo ir kliento sąveikos su tiekėju duomenys. Autorius savo apžvalgoje pabrėžia, kad standartiniai metodai, kaip logistinė regresija, sprendimų medžiai, neuroniniai tinklai metodai kenčia nuo mažo tikslumo ir yra limituoti naudojant didelius duomenų rinkinius su daug kintamųjų. Autorius turi daug kintamųjų, todėl išven-giant šių apribojimų, klasifikavimui naudoja atraminių vektorių klasifikatorių (SVM), o naudingiausiems kintamiesiems atrinkti, autorius naudojo MDI (Hastie, Tibshirani, Friedman & Friedman, 2009) metriką iš atsitiktinio miško modelio, kurią panaudojo, atrenkant

20 pagrindinių kintamųjų iš 66.

Autorius atlieka palyginimus tarp skirtingų metodų ir rezultai rodo, kad pasiūlytas SVM modelis pralenkia logistinę regresiją, sprendimų medžius ir atsitiktinį mišką tikslumo, preciziškumo ir jautrumo atžvilgiu. Pavyzdžiui, tikslumo metrika SVM atveju siekė 83.96 %, o kitų modelių tikslumas varijavo tarp 74.66 % ir 80.08 %. Tačiau reiktų atkreipti dėmesį, kad autorius atsisakė naudoti medžių didinimo metodą, kuris galėtų pralenkti pastarąjį. Autorius apibendrina, teigdamas, kad mašininio mokymosi metodai gali būti naudingas įrankis televizijos servisų tiekėjams, siekiant prognozuoti klientų atsisakymą ir priimti sprendimus išlaikant juos. Taip pat, autorius priduria, kad šis sprendimas galėtų būti naudojamas ne tik minėtoje industrijoje.

Remiantis literatūros apžvalga, panašu, kad algoritmai, naudojant medžių didinimo (ang. boosting) metodiką gerai veikia daugelyje skirtingų šaltinių, klientų atsisakymo prognozavime. Medžių didinimo algoritmai yra populiarūs mašininio mokymosi algoritmų klasė, kuri gali pagerinti prognozių tikslumą sujungdama kelis silpnus mokinius į vieną stiprų ansamblį. Yra matoma, kad medžių grįsti didinimo metodai tokie kaip gradiento didinimo, AdaBoost ir XGBoost yra ypač veiksmingi gerinant modelio našumą. Šie algoritmai apdoroja didelius ir sudėtingus duomenų rinkinius ir dažnai naudojamos ne tik klientų atsisakymo prognozavime, bet ir sukčių aptikimo uždavinyje, ar kituose dvireikšmio klasifikavimo uždaviniuose. Bendrai tariant, panašu, kad medžių didinimo metodai yra galingas įrankis mašininio mokymosi specialistams, į kuriuos reikia atsižvelgti sprendžiant įvairius prognozuojamuosius uždavinius. Tačiau reikia nepamiršti, kad bet kurio algoritmo efektyvumas priklausys nuo daugiau veiksnių, įskaitant problemos sritį, duomenų kokybę ir pasirinktus hiperparametrus. Nors ir Vafeiadis, Diamantaras, Sarigiannidis and Chatzisavvas (2015) šaltinyje geriausiai pasirodė AdaBoost su atraminių vektorių klasifikatorium, tačiau straipsnis buvo parašytas 2015 metais, kuomet XGBoost metodas nebuvo plačiai naudojamas, kaip yra dabar. Dėl plataus naudojimo pastaruoju metu ir gerų rezultatų vėliau pasirodžiusiuose šaltiniuose, šiame darbe yra pasirinktas medžių didinimo metodas XGBoost ir jis bus palygintas su kitais metodais, kuriuose medžių didinimas nėra atliekamas, t.y. paprastąja logistinė regresija, k -artimiausių kaimynų metodu ir paprastuoju atsitiktiniu mišu.

2. Modeliavimo metodų apžvalga

2.1. Modelių metodologija

2.1.1. Logistinė regresija

Logistinė regresija yra statistinis modelis, kurio pagrindinė forma naudoja logistinę funkciją dvejetainiam priklausomam kintamajam modeliuoti (Hosmer Jr, Lemeshow & Sturdivant, 2013). Tikimybės, $P(Y = 1)$ įvertis, kai nepriklausomų kintamųjų reikšmių $x_1, x_2, \dots, x_k$

vektorius yra $\vec{x} = (x_1, x_2, \dots, x_k)$, skaičiuojamas pagal formulę

$$\hat{P}(Y = 1|\vec{x}) = \frac{\exp\{\vec{z}(\vec{x})\}}{(1 + \exp\{\vec{z}(\vec{x})\})}$$

čia $\hat{z}(\vec{x}) = \hat{a} + \hat{b}_1x_1 + \hat{b}_2x_2 + \dots + \hat{b}_kx_k$. Klasifikavimas yra atliekamas pagal taisykles:

Jeigu $\hat{P}(Y = 1|\vec{x}) > 0.5$, tai prognozuojame, kad Y reikšmė yra 1.

Jeigu $\hat{P}(Y = 1|\vec{x}) < 0.5$, tai prognozuojame, kad Y reikšmė yra 0.

2.1.2. K-artimiausių kaimynų metodas

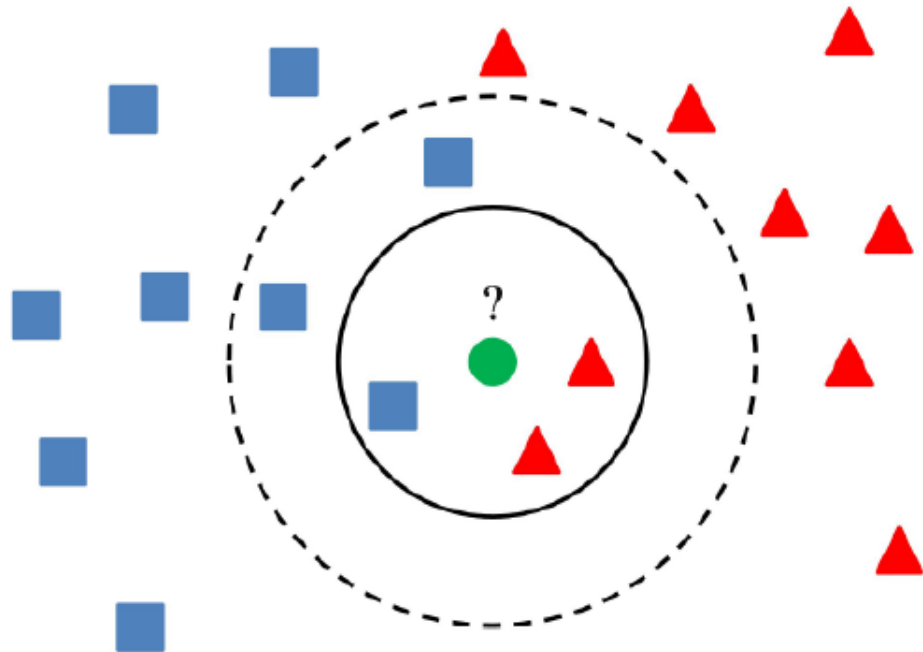
K-artimiausių kaimynų algoritmas (KNN) yra paprastas, bet plačiai naudojamas mašininio mokymosi algoritmas, kuris yra naudojamas tiek klasifikavimo tiek regresijos uždaviniuose. Šis metodas yra neparametrinis, tai reiškia, kad jis nedaro jokių išankstinių prielaidų apie duomenų pasiskirstymus. KNN yra vienas iš stebinių pagrįstų (angl. instance-based) mokymosi algoritmų, kuriame algoritmas saugo visus mokymo stebinius ir naudoja juos spėjant naujų elementų prognozes, ieškant panašumų į treniruojamuosius duomenis (Bishop & Nasrabadi, 2006).

Turint mokymo duomenų rinkinį $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ su atitinkamom tikslu kintamojo žymom $\mathbf{y} = y_1, y_2, \dots, y_n$, ir testavimo stebiniu $\mathbf{z}$:

1. Apsibrėžti atstumo metriką $d(\mathbf{x}, \mathbf{z})$ (Euklido, Manheteno kosinusų ar kita), kad būtų galima išmatuoti atstumus tarp treniravimo stebinio $\mathbf{x}$ ir testinio stebinio $\mathbf{z}$.
2. Apskaičiuoti atstumą $d_i = d(\mathbf{x}_i, \mathbf{z})$ tarp testinio stebinio $\mathbf{z}$ ir visų kitų treniruojamųjų stebinių $\mathbf{x}_i$.
3. Išrinkti K treniruojamųjų stebinių $\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \dots, \mathbf{x}_{i_K}$ su mažiausiais atstumais $d_{i_1}, d_{i_2}, \dots, d_{i_K}$ iki testinio stebinio $\mathbf{z}$. K skaičius yra parametras, kuris nusprendžiamas prieš pradedant skaičiavimus.
4. Klasifikavimo uždavinyje, prognozuojama klasė testiniam stebiniui $\mathbf{z}$ yra apskaičiuojama kaip dažniausia klasė tarp K artimiausių kaimynų $\hat{y} = \text{moda}\{y_{i_1}, y_{i_2}, \dots, y_{i_K}\}$.

Apibendrinant metodiką, 1 pav. yra matomas paprastas pavyzdys, kaip KNN metodas atrodo praktikoje. Žalias apskritimas yra testinis stebinis, kurį norima suprognozuoti, o mėlyni kvadračiukai ir raudoni trikampiai iliustruoja elementus iš dviejų skirtingų klasių treniruojame duomenų rinkinyje. Šiuo atveju, jeigu K parinktume 3, tada 3 artimiausi kaimynai (iliustruota juodam apskritime) nuspręs, kuri klasė bus priskirta testiniam stebiniui. Jeigu K yra 5, tada 5 artimiausi kaimynai spręs išvestį (iliustruotas juodas apskritimas su taškuota linija). Kaip matoma, pirmu atveju, modelis prognozuotų trikampį, o kitų atveju - keturkampį. (Alaliyat, 2023)

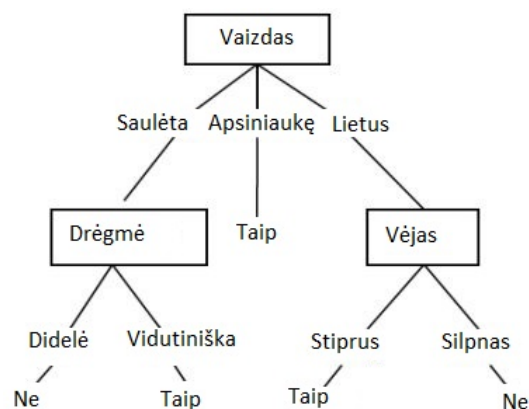
KNN paprastumas ir gebėjimas lengvai interpretuoti padaro šį metodą populiariu metodu įvairiems pritaikymams, bet labai dideliems duomenims jis nėra pritaikomas dėl galimo resursų stygiaus, nes kiekvienoje iteracijoje yra įdarbinami mokymosi duomenys. Surandant tinkamą K ir atstumo metriką, KNN gali būti efektyvus mašininio mokymo algoritmų metodas įvairioms užduotims spręsti.



pav 1. KNN metodika (Alaliyat, 2023)

2.1.3. Sprendimų medis

Sprendimų medis yra grafinė reprezentacija kokio nors specifinės situacijos, naudojama, kai strukturizuotame sprendime įvyksta išsišakojimas (Quinlan, 1986). 2 pav. yra pateikiamas tokio medžio pavyzdys, kuriame yra sprendžiama, ar oras yra palankus teniso žaidimui. Šis medis turi tris dalis.



pav 2. Sprendimų medis, sprendžiant, ar oras palankus tenisui

Medžio struktūrą sudaro trys skirtingos viršūnės, kurias reikia įvardinti norint geriau suprasti tiek įvairių metrikų skaičiavimus, naudojant sprendimų medžius, tiek skirtingų modelių supratime (Hastie et al., 2009):

- **Lapo viršūnė** – ši viršūnė nurodo priklausomo kintamojo reikšmę.
- **Sprendimo viršūnė** – ši viršūnė nurodo testą, kuris yra tikrinamas, siekiant surasti prognozuojamą reikšmę.
- **Šaknies viršūnė** – pirma medžio struktūros viršūnė, nuo kurios prasideda visa struktūra.

Sprendimų medžiai turi dvi rūšis:

- **klasifikavimo** naudojamas, kai prognozuojamas kintamasis yra kategorinis,
- **regresinis** naudojamas, kai prognozuojamas kintamasis yra tolydus.

Kad lengviau suprastume einančius metodus, aptarkime, kaip yra sugeneruojamas sprendimų medis pažingsniui praktikoje. (Hastie et al., 2009)

1. Pradedama nuo viso duomenų rinkinio kaip šakninio mazgo.
2. Pasirenkamas kintamasis, su kuria norima padalinti duomenų rinkinį, tą lemia tokios metrikos kaip informacijos padidėjimas (ang. information gain) arba Gini koeficientas (Hastie et al., 2009).
3. Duomenų rinkinys padalinamas į poaibius pagal pasirinkto kintamojo reikšmes.
4. Kartojamas 2 ir 3 žingsniai kiekvienam duomenų poaibiui, iki kol viena iš nurodytų sąlygų bus patenkinta:
 - visi poaibyje esantys elementai priklauso tai pačiai klasei,
 - nebėra kintamųjų, su kuriais būtų galima dalinti duomenis,
 - pasiektas iš anksto nurodytas sustabdymo kriterijus, kaip maksimalus medžio gylis (maksimalus atstumas nuo šaknies iki lapo viršūnių) arba minimalus elementų skaičius sprendime
5. Lapo viršūnėms priskiriama tikslo kintamojo reikšmė, atsižvelgiant į elementų daugumą, nukeliavusią iki numatyto lapo viršūnės.

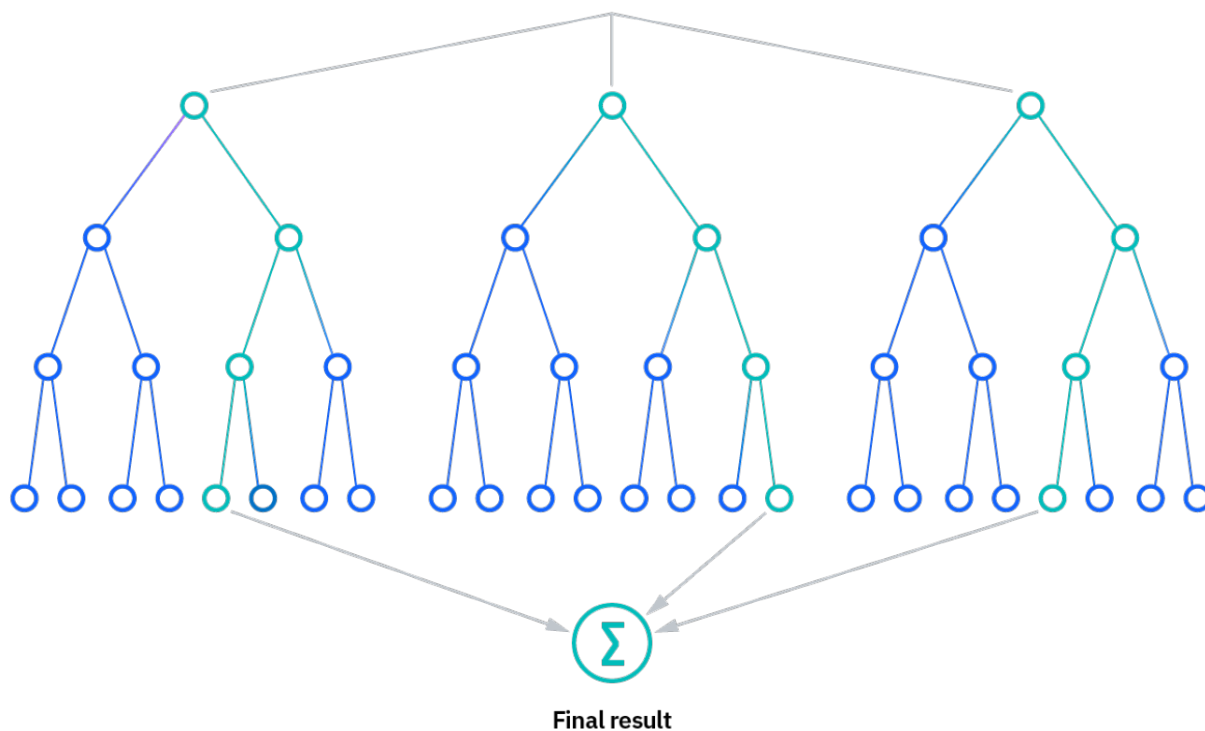
Sprendimų medžio algoritmo tikslas - sukurti kuo mažesnę medį, tuo pačiu tiksliai klasifikuojant stebinius. Siekiant medžio sumažinimo taip pat yra siekiama išvengti persotinimo, kad tą pasiektų algoritmas gali „genėti“ medį pašalindamas šakas, kurios ženkliai nepagerina medžio našumo. Praktikoje yra daug sprendimų medžio algoritmų patobulinimų, pvz., naudojant ansamblio metodus, tokius kaip atsitiktiniai miškai ir medžių padidėjimas. Šie metodai sujungia kelių sprendimų medžių išvestis, kad prognozės tikslumas būtų didesnis.

2.1.4. Atsitiktinis miškas

Atsitiktinio miško algoritmas yra ansamblio mašininio mokymosi metodas, naudojamas klasifikavimo, regresijos ir kito tipo prognozavimo užduotims. Jo veikimas yra grindžiamas sukuriant kelis sprendimų medžius su atsitiktiniais duomenų ir jų kintamųjų poaibiais, o tada sujungiant jų išvestis padaroma galutinė prognozė. Atsitiktinio miško sprendimų medžiai paprastai auginami iki didžiausio prieš tai nurodyto gylio be jokio genėjimo, o visų medžių prognozės yra apskaičiuojamos vidurkiu arba suma, kad būtų gaunama galutinė išvestis. (Breiman, 2001)

Vienas iš pagrindinių atsitiktinių miškų privalumų yra tai, kad jie gali apdoroti didelių matmenų ir triukšmingus duomenis ir yra mažiau linkę persisotinti nei atskiri sprendimų medžiai. Atsitiktiniai miškai padeda pasirinkti kintamuosius, kadangi kiekviename medyje paimamas ne tik duomenų poaibis (imtis), bet ir kintamųjų. Pagrindiniai žingsniai, kaip susigeneruoja atsitiktinio miško modelis:

1. Atsitiktinai parenkama imtis duomenų ir sukuriamas naujas duomenų rinkinys.
2. Iš pastarojo duomenų rinkinio atsitiktinai parenkami kintamieji.
3. Sprendimų medis yra didinamas iki maksimalaus iš anksto nurodyto gylio be jokio genėjimo, naudojant išrinktą duomenų imtį bei atrinktus kintamuosius.
4. Atkartoti 1-3 žingsnius, kol bus sugeneruotas iš anksto nurodytas sprendimų medžių skaičius.
5. Suagreguoti prognozes iš visų medžių, siekiant sudaryti galutinę prognozę. 3 pav. yra iliustratyviai parodyta, kaip yra atliekama prognozė. Į tris medžius yra nusiunčiamas vienas tas pats stebiny, kiekvienas medis buvo sugeneruotas skirtingai, todėl to stebinio kelionė per viršūnes taip pat buvo nevienoda. Kiekviena prognozė yra unikali, todėl galutinė miško išvestis yra susumuojama arba suvidurkinama.



pav 3. Atsitiktinio miško prognozė (IBM, 2023)

Atsitiktiniai miškai turi nemažai hiperparametrų, kurie gali būti derinami siekiant pagerinti jų kokybę, tokie kaip medžių skaičius, maksimalus medžių gylis, kintamųjų skaičius kiekvienam paskirstymui ir minimalus stebinių skaičius, reikalingas padalinant vidinę viršūnę. Įprastas metodas optimizuoti hiperparametrus yra kryžminė paieška.

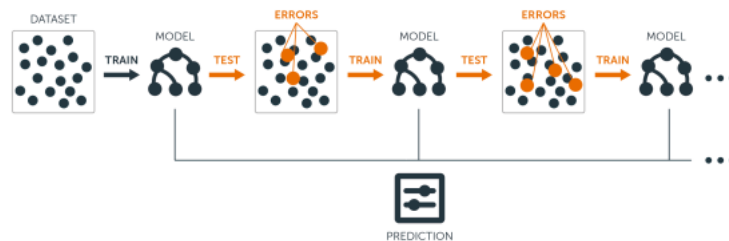
2.1.5. Medžių didinimas

XGBoost metodas yra pastaraisiais metais praktikoje vienas iš labiausiai pritaikomų algoritmų mašiniame mokyme. Šis metodas iš esmės yra gradiento didinimo metodas, kuris yra labiau reguliarizuotas, leidžiantis išvengti modelio persotinimo (Chen & Guestrin, 2016). Šis metodas veikia treniruojant didelį skaičių sprendimo medžių, kiekvienas medis yra treniruojamas su duomenų imtimi, o galutinė prognozė yra daroma atsižvelgiant visų medžių prognozes. Taigi, principas yra toks pats, kaip ir atsitiktinio miško, skirtumas yra tas, kad šiame metode yra naudojamas gradientinis medžių didinimas, o kad jį suprasti, reiktų trumpai susipažinti, kas yra medžių didinimas (ang. *Boosting*) ir gradientinis didinimas (ang. *Gradient boosting*).

Medžių didinimas yra iteracinis metodas, kuris remiasi ansamblio technika, nes daugelis medžių yra sujungiami kartu, kad būtų gaunamas galutinis rezultatas. Tačiau medžių didinimas skiriasi tuo, kad užuot mokę visus medžius atskirai vienas nuo kito, didinimas treniruoja modelius iš eilės, o kiekvienas naujas modelis mokomas, siekiant ištaisyti ankstesnių medžių padarytas klaidas. Medžiai yra pridedami atskirai, kol nebeišeina atlikti patobilinimų (Friedman, 2001).

XGBoost naudoja **gradientinį didinimą**, kurio principas yra toks, kad nauji modeliai yra mokomi numatant ankstesnių modelių klaidas, 4 pav. yra matomas metodo procesas. O nueinant labiau į detales, šį metodą galima aprašyti per kelis žingsnius. (Boehmke & Greenwell, 2019)

1. Suskaičiuoti tikslo kintamojo vidurkį $\bar{y}$.
2. Apskaičiuoti liekanas $\epsilon_i = \bar{y} - \hat{y}_i$, kur $\hat{y}_i$ šiame žingsnyje atitinka tikras reikšmes.
3. Sukonstruojamas sprendimų medis, kuriame kiekvienas lapas, t.y. prognozuojama reikšmė atitinka liekanos reikšmes, o jeigu į lapą papuola daugiau negu viena reikšmė, imamas tų reikšmių vidurkis.
4. Atliekama prognozė, naudojant medžius iš ansamblio, t.y. (po pirmos iteracijos imamas vienas medis, po antros pirmas ir antras ir t.t.), nauja prognozė yra $\hat{y}_i = \bar{y} + \alpha \epsilon_i$, kur α yra hiperparametras, nurodantis mokymosi greitį.
5. Perskaičiuojamos naujos klaidų reikšmės, kurios yra naudojamos naujame medyje kaip reikšmės, esančios lapuose.
6. Kartojami 3-5 žingsniai tiek kartų, kiek nurodyta hiperparametre.
7. Apskaičiuoti galutinę prognozę, naudojant visą medžių ansamblį.



pav 4. Medžių mokymas paeiliui, gradientiniu metodu, (Boehmke & Greenwell, 2019)

Taigi, tiek XGBoost, tiek gradientinis medžių didinimas remiasi gradientinio didinimo principu, tačiau skirtumas yra jau modeliavimo detalėse, tiksliau, XGBoost metodas naudoja labiau reguliarizuotą modelio formulizavimą, kad išvengtų modelio persotinimo, o tas užtikrina didesnę modelio našumą (Chen & Guestrin, 2016).

2.2. Modelio tikslumo metrikos

Modelių tikslumo metrikos yra esminis būdas įvertinti mašininio/statistinio modelių tikslumą. Yra keleri įvertinimo metrikų tipai, kurie gali būti naudojami priklausomai nuo

modelio, dažniausiai naudojamos metrikos yra tikslumas, precizija, jautrumas, F-1 metrika, ROC kreivė.

Tikslumas (ang. accuracy) yra pati elementariausia tikslumo metrika, ji yra apibrėžiama, teisingai išprognuotų stebinių skaičių, padalinus iš visų stebinių skaičiaus (Hastie et al., 2009):

$$\text{Tikslumas} = \frac{\text{teisingų prognozių skaičius}}{\text{bendras stebinių skaičius}} \cdot 100$$

Sumaišties matrica iš esmės, nėra tikslumo matas, bet jį suprasti yra būtina, norint suprasti kitas metrikas.

Sakykime, kad nulinė hipotezė „Stebinis yra linkęs atsisakyti paslaugų“, sumaišties matricoje, atsispindi keturios baigtys, kurios yra atvaizduotos 2 lentelėje. Jų reikšmės:

- Teisingas teigiamas reiškia, kiek teigiamų išvesčių modelis suprognozavo teisingai
- Klaidingas teigiamas reiškia, kiek teigiamų išvesčių modelis suprognozavo neteisingai
- Klaidingas neigiamas, reiškia, kiek neigiamų išvesčių modelis suprognozavo klaidingai
- Teisingas neigiamas reiškia, kiek neigiamų išvesčių modelis suprognozavo teisingai

2 lentelė. Sumaišties matrica

		Suprognozuota	
		Atsisakė paslaugų	Neatsisakė paslaugų
Pagrindinė tiesa	Atsisakė paslaugų	Teisingas teigiamas (TT)	Klaidingas teigiamas(KT)
	Neatsisakė paslaugų	Klaidingas neigiamas(KN)	Teisingas neigiamas(TN)

Idealiu atveju, klaidingai teigiamų ir klaidingai neigiamų suprognozuotų stebinių skaičius turėtų būti lygus nuliui.

Precizija nurodo teisingai ir teigiamai suprognozuotų atvejų skaičiaus santykį su teisingai ir teigiamai suprognozuotų atvejų bei klaidingai teigiamai suprognozuotų atveju suma (Powers, 2020).

$$\text{Precizija} = \frac{TT}{TT+KT}$$

Jautrumas nurodo teisingai ir teigiamai suprognozuotų atvejų skaičiaus santykį su teisingai ir teigiamai suprognozuotų atvejų bei klaidingai neigiamai suprognozuotų atveju suma (Powers, 2020).

$$\text{Jautrumas} = \frac{TT}{TT+KN}$$

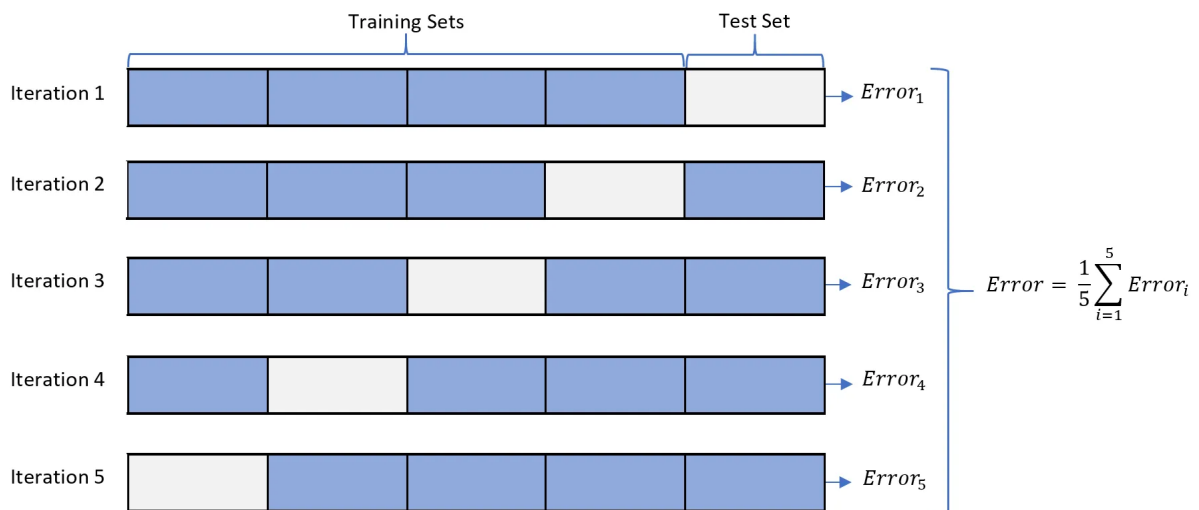
F1 metrika yra precizijos ir jautrumo harmoninis vidurkis (Powers, 2020).

$$F_1 = \frac{2}{\frac{1}{\text{Precizija}} + \frac{1}{\text{Jautrumas}}}$$

Ši metrika reprezentuoja precizijos ir jautrumo balansą taip apibendrinama šias metrikas.

2.3. Modelio įvertinimas kryžminės patikros metodu

Iš vienos testinių ir validavimo rinkinių padalinimų nuspręsti modelio tikslumą ne visada yra teisingai. Tam, kad būtų padaromas tikslesnis modelių palyginimas, yra taikoma **kryžminė patikra** (ang. cross-validation). Kryžminė patikra yra mašininio mokymosi algoritmo našumo įvertinimo metodas. Šio metodo idėja yra padalinti duomenis į du rinkinius: mokymo ir testavimo. Testavimo rinkinys yra naudojamas modelio paklaidai įvertinti, t.y., kaip gerai modelis veiks su naujais, nematytais duomenimis. Vienas iš dažnų būdų, atliekančių kryžminę patikrą yra k-perlenkimų kryžminė patikra. Šiame būde, duomenys yra padalinami į k vienodo dydžio poaibių. Tada modelis yra treniruojamas su $k - 1$ poaibiu, o paskutinė imtis yra paliekama validavimui. Šis procesas yra kartojamas k kartų, kad kiekviena imtis vieną kartą pasitarnautų kaip validavimo rinkinys. Paprastą iliustraciją galima pamatyti 5 pav., kur yra atvaizduotas procesas su $k = 5$, matosi, kai per penkias iteracijas kiekvienas duomenų poaibis yra panaudojamas kaip testinis duomenų rinkinys ir iš kiekvienos iteracijos yra suskaičiuojami nuostoliai, kurie galiausiai yra suvidurkinti, kadangi modelio tikslumas yra įvertinamas kaip vidurkis einant per visus testus k (Ozdemir, 2016).



pav 5. K-perlenkimų metodo iliustracija (Patro, 2021)

Kalbant matematiškiau, tegul $\mathcal{D} = (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$ yra duomenų rinkinys, kur $\mathbf{x}_i$ yra vieno taško predikatų vektorius, o y_i - tikslo kintamasis. Tegul $f(\mathbf{x}; \theta)$ aprašo mašininio mokymosi modelį su parametrais θ . Tikslas yra surasti optimalią θ reikšmę, kuri minimizuos modelio klaidą (Hastie et al., 2009).

1. Duomenys yra padalinami į k poaibių: $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_k$.
2. Su kiekviena iteracija $i = 1, 2, \dots, k$ yra treniruojamas modelis su likusiais $k - 1$ poaibiais:

$$\theta_i = \arg \min_{\theta} \sum_{j \neq i} \mathcal{L}(y_j, f(\mathbf{x}_j; \theta)),$$

kur $\mathcal{L}$ yra nuostolių funkcija.

3. Įvertinti i -os iteracijos paklaidą:

$$\text{Err}_i = \frac{1}{|\mathcal{D}_i|} \sum_{(\mathbf{x}, y) \in \mathcal{D}_i} \mathcal{L}(y, f(\mathbf{x}; \theta_i)).$$

4. Apskaičiuoti vidutinę paklaidą per visas iteracijas:

$$\text{CV}(\theta) = \frac{1}{k} \sum_{i=1}^k \text{Err}_i.$$

5. Parenkama θ reikšmė, kuri minimizuoja vidutinę paklaidą.

Kryžminis patikrinimas yra galingas įrankis mašininio mokymosi modelių našumui įvertinti. Jis leidžia tiksliau ir patikimiau įvertinti modelio tikslumą, naudojant kelias iteracijas ir skirtingus duomenų pogrupius mokymams ir bandymams, taip pat sumažinant persotinimo riziką, kadangi kiekvienoje iteracijoje modelis parodo savo gebėjimą prognozuoti nematytus duomenis.

2.4. Kintamųjų svorių metrikos

Kintamųjų svarbumo analizė yra svarbi dalis klientų elgsenos analizėje. Mūsų atveju, ši analizė atskleidžia, kurie kintamieji duomenų rinkinyje, apibūdinantys klientą yra svarbiausi, prognozuojant jų pasitraukimą. Vien iš šios analizės, telekomunikacijų kompanija gali imtis pirminių veiksmų, darant specializuotas kampanijas didelėms auditorijoms, suprantant, kurie veiksniai yra labiausiai problematiški. Kadangi šiame darbe bus naudojamas ne vienas mašininio mokymosi metodas, šiame skyrelyje bus pristatyti pagrindiniai jų kintamųjų svorių apskaičiavimo būdai.

XGBoost metode yra keletas skirtingų metrikų, apskaičiuojančių kintamųjų svarbą, populiariausios - **svorio**, **prieaugio** ir **padengiamumo** metrikos (Chen & Guestrin, 2016).

Svorio metrika yra pagrįsta skaičiumi, kiek kartų kintamasis yra panaudojamas duomenims padalinti į visus sprendimų medžius. Šią metriką galima atvaizduoti tokia formule:

$$\text{svoris}_i = \sum_{j \in I} I_{j=i}.$$

Čia I reprezentuoja visų duomenų rinkinio mokymų atvejų aibę, o $I_{j=i}$ yra indikatorinė funkcija, kuri lygi 1, kai kintamasis i yra panaudotas dalinant medį duomenų rinkinio mokymo atvejo j metu, 0 kitu atveju. Didesnė svorio reikšmė nurodo didesnį kintamojo reikšmingumą.

Prieaugis yra naudojamas kaip metrikos matas, nurodantis, kiek sumažėjo netikslumas, kai sprendimų medis yra atliekamas su tam tikru duomenų poaibių. Jis yra skaičiuojamas padalinant duomenis pagal kiekvieną kintamąjį tikrinant, kaip pasikeitė nuostoliai neatlikus padalinimo ir jį atlikus. Šią metriką galima konkretizuoti su tokia formule:

$$\text{prieaugis} = \frac{\text{balas}_{\text{kairė}}^2}{\text{svoris}_{\text{kairė}} + \lambda} + \frac{\text{balas}_{\text{dešinė}}^2}{\text{svoris}_{\text{dešinė}} + \lambda} - \frac{\text{balas}_{\text{viso}}^2}{\text{svoris}_{\text{viso}} + \lambda} - \gamma.$$

Kur:

- $\text{balas}_{\text{kairė}}$ tikslo kintamojo reikšmių suma kairėje šakoje,
- $\text{svoris}_{\text{kairė}}$ kintamojo svorių suma kairėje mokymų šakoje,
- $\text{balas}_{\text{dešinė}}$ tikslo kintamojo reikšmių suma dešinėje šakoje,
- $\text{svoris}_{\text{dešinė}}$ kintamojo svorių suma dešinėje mokymų šakoje,
- $\text{balas}_{\text{viso}}$ tikslo kintamojo reikšmių suma visame mazge,
- $\text{svoris}_{\text{viso}}$ svorių suma visame mazge,
- λ and γ yra reguliuojami parametrai.

Padengiamumo metrika matuoja vidutinį padengiamumą kiekvienam kintamajam per elementus medžiuose. Vienam medžiui padengiamumą galima suskaičiuoti tokia formule:

$$\text{padengiamumas}(i,t) = \sum_{j \in \text{lapas}(t)} \frac{|I_j|}{h_j^2} \sum_{k \in I_j} I_{k \in \text{split}(j,i)}.$$

Čia:

- $\text{lapas}(t)$ lapų viršūnių rinkinys medyje t ,
- $|I_j|$ yra imčių skaičius viršūnėje j ,
- h_j yra suprognuota reikšmė viršūnėje j ,
- $I_{k \in \text{split}(j,i)}$ yra indikatorinė funkcija su reikšme 1, jei elementas k naudojo kintamą i savo dalomajam sprendime ir atsidūrė viršūnėje j , 0 kitu atveju.

Padengiamumo metrika kintamajam i yra apskaičiuojama susumuojant $\text{cover}(i,t)$ reikšmės visiems medžiams t .

Atsitiktiniuose miškuose dažniausiai naudojama „vidutinio sumaišymo sumažėjimas“ (mean decrease impurity), kuris yra skaičiuojamas taikant Gini svarbumo metrika. Ji yra gaunama, apskaičiuojant bendrą „priemaišų“ sumažėjimą (išmatuota Gini indeksu), kai kintamasis yra įtraukiamas į visus atsitiktinio miško medžius. **Gini indeksas** yra metrika, nurodanti duomenų taškų įvairovę. Binarinio klasifikatoriaus uždavinyje, ji iš esmės nurodo tikimybę, kad atsitiktinai parinktas taškas bus neteisingai suklasifikuotas. Maža Gini indekso reikšmė reiškia labiau homogeninius duomenis, didelė - heterogeninius.

Gini svarbumo metrika kintamųjų atrinkime yra suskaičiuojama sumuojant Gini indekso sumažėjimus per visas viršūnes, kur kintamasis buvo naudotas duomenų paskirstymui. Gini indekso sumažėjimas paskirstymui yra skirtumas tarp pirminės viršūnės Gini indekso ir antrinių viršūnių Gini indeksų svertinio vidurkio. Antrinės viršūnės svoris yra stebinių skaičius, pasiekęs šią viršūnę. Detaliau, kintamųjų svarba atsitiktiniame miške yra įvertinami darant šiuos žingsnius. (Géron, 2022)

1. Kiekvienam medžiui yra paskaičiuojamas Gini indeksas kiekvienai viršūnei žymimas $I_{\text{medis}} = \sum_{i=1}^{n_{\text{viršūnė}}} p(i) - (1 - p(i))$. Čia $n_{\text{viršūnė}}$ yra bendras viršūnių skaičius medyje, $p(i)$ yra stebinių proporcija viršūnėje i , priklausančių teigiamai klasei.
2. Kiekvienam kintamajam j yra paskaičiuojamas Gini kintamojo svarba duotam medžiui. Tai yra padaroma sumuojant Gini indekso sumažėjimus tarp visų paskirstymų tam kintamajam. $GI_j = \sum_{t \in \text{splits}(j)} \frac{n_t}{n} I_{\text{left}}(t) + \frac{n_t}{n} I_{\text{right}}(t) - I_{\text{tree}}$ kur $\text{splits}(j)$ yra visi kintamojo j paskirstymai, n_t yra stebinių skaičius, kurie nueina per paskirstymą t , n yra bendras stebinių skaičius medyje, $I_{\text{left}}(t)$ ir $I_{\text{right}}(t)$ yra Gini indeksai kairiniame ir dešiniame antrinės viršūnės paskirstyme.
3. Apskaičiuoti Gini svarbumo metriką (vidutinio sumaišymo sumažėjimas) kiekvienam kintamajam j skaičiuojant kintamojo j Gini svarbumus visuose atsitiktinio miško medžiuose. $MDI_j = \frac{1}{N_{\text{tree}}} \sum_{k=1}^{N_{\text{tree}}} GI_{j,k}$, kur N_{medis} yra bendras medžių skaičius atsitiktiniame miške, $GI_{j,k}$ is the Gini svarbumas kintamajam j medyje k .
4. Suranguoti kintamuosius pagal jų MDI svarbumo balus, didesnis balas reiškia didesnę svarbumą modelyje.

Pristatant logistinės regresijos kintamųjų svarbumą, intuityvus būdas yra pristatyti modelio koeficientus. Tai leidžia aiškiai ir glaustai interpretuoti modelio rezultatus. Norint suprasti, kaip koeficientai yra įvertinami, reikia suprasti, kaip veikia logistinė regresija, tai trumpai yra aprašyta 2.1.1 skyrelyje.

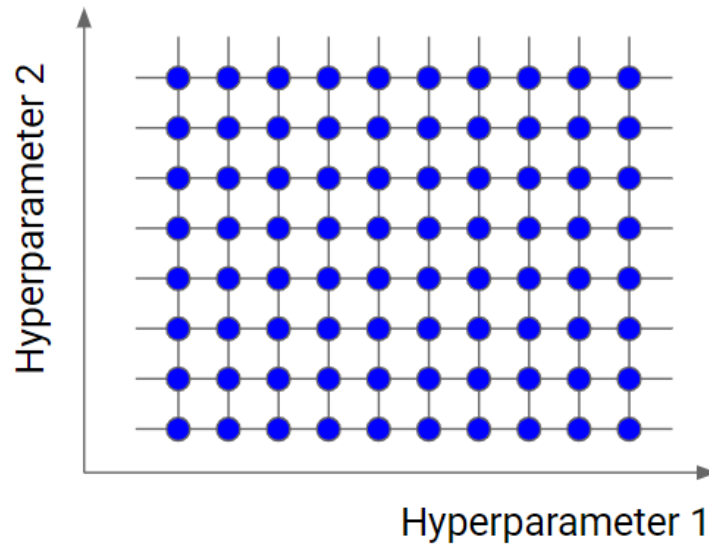
2.5. Hiperparametrų paieškos metodai

Mašininio mokymosi modelių našumas labai priklauso nuo hiperparametrų pasirinkimo, kurie apibrėžia mokymosi algoritmo elgesį su duomenimis. Šiame skyrelyje bus apžvelgiami plačiausiai taikomi paieškos metodai - gardelės ir atsitiktinė paieška.

2.5.1. Gardelės paieška

Gardelės paieška yra metodas, naudojamas mašiniame mokyme, siekiant surasti geriausią hiperparametrų rinkinį paskirtam modeliui. Hiperparametrai yra mašininio mokymosi algoritmo parametrai, kurie nėra išmokstami treniravimosi metu, bet yra nustatomi rankiniu būdu pačio tyrėjo. Hiperparametrų pavyzdžiai galėtų būti medžių skaičius atsitiktiniame miške, mokymosi žingsnis neuroniniuose tinkluose ir kaimynų skaičius k -artimiausių kaimynų metode. Šiam metodui yra nurodomos įvairios įvairių parametrų reikšmės ir tada modelis treniruojamas ir vertinamas su kiekviena įmanoma pateikta hiperparametrų kombinacija. Modelio tikslumas yra įvertinamas naudojant testinius duomenis, arba taikant 3.7 skyriuje apkalbėtą kryžminę validaciją. Tiksliausių hiperparametrų rinkinys yra pasirenkamas kaip

optimalus ir yra panaudojamas praktikoje, kaip geriausias modelis (Géron, 2022). Šio metodo pavadinimas „gardelės paieška“ kyla iš to, kad pateiktos hiperparametrų reikšmės suformuoja gardelę, arba visų įmanomų kombinacijų lentelę, 6 pav. matomas pavyzdys, kaip yra sukombinuojamos dviejų hiperparametrų reikšmės gardelės pavidale, 10 vieno ir kito parametro reikšmės sugeneruoja 10x10 dydžio gardelę ir gardelės paieška patikrintų modelių su 100 skirtingų hiperparametrų kombinacijų. Gardelės paieška gali būti brangi resursų prasme, ypač jeigu hiperparametrų diapozonas yra didelis, taip pat, ne visada yra lengva parinkti hiperparametrų reikšmes, iš kurių būtų prasminga rinktis, tačiau tai yra paprastai suprantamas ir visgi efektyvus būdas rasti geresnius hiperparametrus, negu taikant atsitiktinį spėjimą.



pav 6. Gardelės paieškos vizualizacija, (Malato, 2021)

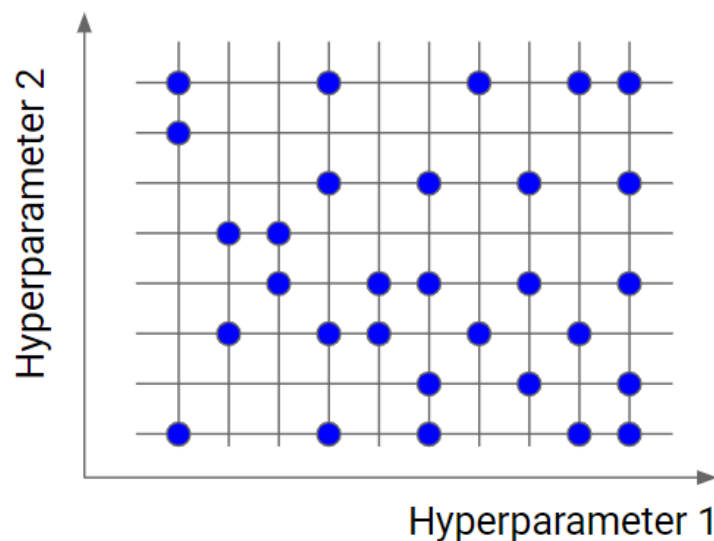
Gardelės metodą galima aprašyti keliais žingsniais.

1. Apibrėžti parametrų reikšmes: tam tikriems hiperparametrms nurodyti atitinkamas reikšmes arba reikšmių intervalus. Tegul $H = h_1, h_2, \dots, h_n$ pažymi hiperparametrų rinkinius. Kiekvienam hiperparametrui h_i pažymėkime gardelę $G_i = g_{i1}, g_{i2}, \dots, g_{im}$. Kur m yra galimų reikšmių hiperparametrui h_i skaičius.
2. Nurodyti paieškos erdvę: suskaičiuoti visą kombinacijų skaičių. Tegul N žymi bendrų derinių skaičių paieškos erdvėje. Bendra paieškos erdvė S yra hiperparametrų gardelės Dekarto sandauga $S = G_1 \times G_2 \times \dots \times G_n$.
3. Sugeneruojamos parametrų kombinacijos: $P_i = h_{i1}, h_{i2}, \dots, h_{in}$, kur h_{ij} nurodo j -tojo hiperparametro reikšmę i -tai kombinacijai.
4. Kiekvienam parametrų deriniui P_i , apmokyti modelį naudojant atitinkamas hiperparametrų reikšmes. Įvertinti modelio tikslumą naudojant kryžminę ar kitą validavimo būdą. Tegul M_i nurodo apmokytą modelį su parametrų kombinacija P_i .

5. Pagal pasirinktą modelio našumo metriką, iš visų įvertintų modelių pasirinkti geriausią modelį M_i . Tarkime, pasirinkus našumo metriką, geriausią modelį galima apibrėžti $M_{\text{geriausias}} = \underset{i}{\operatorname{argmax}} \operatorname{tikslumas}(M_i)$, kur $\operatorname{tikslumas}(M_i)$ nurodo modelio M_i tikslumą.

2.5.2. Atsitiktinė paieška

Gardelės paieška turi aiškius trūkumus - vienas iš jų yra iš anksto pateiktos skaitinių parametrų reikšmės, kitas - apskaičiavimo kaina resursų prasme, pateikus didelį diapozoną skirtingų parametrų. Alternatyvus būdas gardelės paieškai yra **atsitiktinė paieška**, kuri yra kitas metodas siekiant surasti optimalius hiperparametrus mašininio mokymosi modelyje. Šioje paieškoje, vietoje to, kad būtų ieškomos visos įmanomos kombinacijos, yra parenkamas fiksuotas skaičius hiperparametrų kombinacijų iš tam tikros hiperparametrų erdvės. Čia privalumas yra tas, kad atsitiktinė paieška gali ištirti platesnį diapozoną hiperparametrų reikšmių per trumpesnę laiką, lyginant su gardelės paieška. Atliekant atsitiktinę paiešką, metodo įgyvendinimo žingsniai yra beveik vienodi kaip ir buvo įvardinta 2.5.1 skyriuje, esminis skirtumas yra, kad vietoj visų galimų parametrų kombinacijų yra pasirenkamas fiksuotas skaičius paibių, kuris yra parenkamas atsitiktinai iš paieškos erdvės. 7 pav. yra pateikiamas pavyzdys, kaip vietoje visos aibės, pavaizduotos 6 pav. yra parenkama atsitiktinė imtis.



pav 7. Atsitiktinės paieškos vizualizacija, (Malato, 2021)

2.6. Klasių balansavimo metodai

Klasių disbalansas yra dažnas iššūkis klasifikavimo uždaviniuose, įskaitant telekomunikacijos klientų elgsenos prognozavimą. Klasių disbalansas atsiranda, kai viena klasė yra daug labiau paplitusi nei kita ir tokiais atvejais modeliai, parengti remiantis nesubalansuotais duomenimis yra linkę prognozuoti į daugumos klasę.

Siekiant išvengti šios problemos, yra siekiama išlyginti tikslo kintamąjį, tam yra du būdai:

- prigeneruoti trūkstamos klasės stebinių tiek, kiek yra daugumos klasės stebinių (ang. Oversampling), populiariausi metodai – atsitiktinis, SMOTE, ADASYN, ROSE.
- panaikinti daugumos klasės stebinius iki tokio kiekio, kad išsilygintų su mažumos klasės kiekiu (ang. Undersampling), populiariausi metodai – atsitiktinis, NearMiss, klasterių centroidų, Tomek.

Abu metodai turi trūkumų, prigeneruodami fiktyvius duomenis yra pradedama dirbti su netikrais duomenimis, o naikinant turimus duomenis, prarandama jau turima informacija. Šiame darbe apžvelgsime atsitiktinius, SMOTE ir NearMiss.

2.6.1. Atsitiktiniai imties balansavimo metodai

Paprasčiausi metodai yra atsitiktinio trūkstamos imties didinimo ar daugumos imties mažinimo, jų apibendrinti veikimo principai.

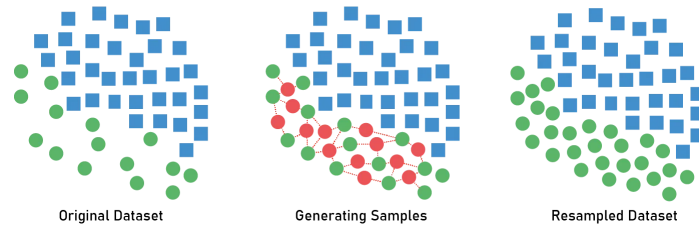
Tegul N žymi visą stebinių skaičių duomenyse, o N_{dauguma} atitinka daugumos klasės skaičių ir $N_{\text{mažuma}}$ atitinka mažumos. Atsitiktinis trūkstamos imties didinimas duplikuoja mažumos klasės stebinius tol, kol mažumos skaičius sutaps su daugumos skaičiumi N_{dauguma} . Atsitiktinis daugumos imties mažinime, daugumos klasės stebiniai yra atsitiktinai ištrinami, kol likusių daugumos stebinių skaičius sutaps su mažumos skaičiumi $N_{\text{mažuma}}$. (He & Garcia, 2009)

2.6.2. SMOTE

Vienas populiariesnių metodų, atliekant duomenų balansavimą mažumos klasės imtį didinti yra SMOTE (ang. Synthetic Minority Over-sampling Technique). Šis metodas generuoja sintetinius mažumos klasės stebinius, kurdamas duomenis, kurie yra panašūs į jau esančius mažumos duomenis. Tai padeda subalansuoti klasių pasiskirstymą ir pagerinti klasifikatoriaus našumą. Šį metodą galima apibrėžti keliais iteruojamais žingsniais (Chawla, Bowyer, Hall & Kegelmeyer, 2002).

1. Tegul X yra visi turimi stebiniai, o $X_{\min}$ yra stebiniai iš mažumos klasės.
2. Mažumos klasės stebiniui $x_i \in X_{\min}$, tegul $X_k(x_i)$ yra k artimiausi kaimynai stebiniui x_i iš $X_{\min}$, pagal kurią nors atstumo metriką $d(\cdot, \cdot)$. Čia galima apibrėžti KNN sugeneruotą imtį $X_k(x_i) = x_{j_1}, x_{j_2}, \dots, x_{j_k}$ kur $d(x_i, x_{j_1}) \leq d(x_i, x_{j_2}) \leq \dots \leq d(x_i, x_{j_k})$ ir $x_{j_1}, x_{j_2}, \dots, x_{j_k} \in X_{\min}$.
3. Toliau yra sugeneruojamas atsitiktinis kaimynas x_{j_r} iš $X_k(x_i)$, kur r yra atsitiktinai sugeneruotas sveikasis skaičius tarp 1 ir k imtinai.

4. Sugeneruojamas naujas sintetinis stebiny x_{naujas} interpoliuojant tarp taškų x_i ir x_{j_r} atsižvelgiant į atsitiktinį svorį λ tarp 0 ir 1, pvz., $x_{naujas} = x_i + \lambda(x_{j_r} - x_i)$. Tokių stebinių sugeneravimo pavyzdys matomas 8 pav., kaip antrame žingsnyje matosi tarp įvairių mažumos klasės taškų yra sugeneruojami panašūs raudoni taškai, kurie po klasių balansavimo tampa žaliais.
5. 2-4 žingsniai yra kartojami, kol norimas stebinių kiekis susibalansuoja.



pav 8. SMOTE metodo sintetinių stebinių sugeneravimo iliustracija (Orellana, 2020)

SMOTE yra paprastas ir efektyvus išbalansuotų duomenų rinkinių tvarkymo būdas. Svarbu paminėti, kad kartais jis gali sugeneruoti triukšmingus stebinius, ypač, jeigu mažumos klasės duomenys yra panašūs į daugumos klasės, bet reiktų pastebėti, kad su tokiais duomenimis, kai skirtingų klasių stebiniai yra panašūs, bet kokiam metodui mašininio mokymo metodui būtų sunkiau prognozuoti teisingą atsakymą.

2.6.3. NearMiss

NearMiss imties mažinimo metodu siekiama sumažinti daugumos klasę, selektyviai pašalinant stebinius iš daugumos klasės, taip, kad būtų sukurtas labiau subalansuotas daugumos rinkinys, stebinius iš daugumos klasės, kurie yra artimesni mažumos klasei (Lemaître, Nogueira & Aridas, 2017). Šio metodo įgyvendinimo žingsniai:

1. Surasti atstumus tarp stebinių iš daugumos ir mažumos klasių:
 - Apsibrėžti tikslumo metriką $d(x_i, x_j)$ kuri nurodys skirtumą tarp dviejų stebinių.
 - Apskaičiuoti atstumus tarp kiekvieno daugumos stebinio x_i ir kiekvieno mažumos stebinio x_j .
2. Kiekvienam stebiniui iš daugumos klasės, surasti N artimiausių stebinių iš mažumos klasės, atsižvelgiant į apskaičiuotus atstumus.
3. Atrinkti daugumos klasės stebinius, kurie turi mažiausią vidutinį atstumą iki kitų N mažumos stebinių.

3. Tiriamoji dalis

3.1. Duomenų rinkinys, analizė

Bet kurio mašininio mokymosi uždavinio sprendimo sėkmė labai priklauso nuo duomenų kokybės ir išsamumo. Šiame skyrelyje, bus išanalizuojamas duomenų rinkinys, kuris bus naudojamas dvireikšmio klasifikavimo uždaviniui spręsti žiūrint į struktūrą, turinį, duomenų kokybę. Išplėstinė duomenų analizė yra atliekama, siekiant suprasti sąryšį tarp tikslo ir prognozuojamųjų kintamųjų, siekiant surasti problemas duomenų rinkinio kokybėje. Analizės metu, bus vizualiai atvaizduojami individualių prognozuojamųjų kintamųjų pasiskirstymai tikslo kintamojo atžvilgiu, bus patikrinamas klasių balansas. Atlikus šią analizę bus identifikuotos problemos duomenyse, kurios leis priimti standartinius sprendimus, tvarkant duomenų rinkinį.

3.1.1. Duomenų rinkinys ir kintamieji

Šioje darbo dalyje yra pateikiama duomenų rinkinio apžvalga, kintamieji, jų tipai. Duomenys yra gauti iš telekomunikacijų įmonės, kuriuose yra tokie kintamieji:

- **Skaitiniai kintamieji** - amžius, mokesčio už paslaugas dydis, produkto ilgaamžiškumas, kliento ilgaamžiškumas, laikas iki kontrakto pabaigos, užregistruotų gedimų skaičius per pastaruosius metus, paslaugos naudojimo srautas savaitgalį, paslaugos naudojimo srautas šiokiadienį,
- **Kategoriniai kintamieji** - reikšmė, parodanti kurias kitas paslaugas klientas yra nusipirkęs iš to paties tiekėjo, papildomos įrangos išpirkimo indentifikatorius, kliento kontrakto tipas (terminuotas, ar ne), gyvenamosios vietos informacija (ar gyvena didmiesty, bute), paslaugos plano pavadinimas,
- **Tikslo kintamasis** - dvireikšmis, nurodantis, ar klientas atsisakė paslaugos, ar ne.

Kadangi dauguma kintamųjų yra kategoriniai, modeliuojant juos reikia perprogramuoti į binarius. Pavyzdžiui, gyvenamą vietą galima sukonkretinti iki binario, nurodančio, ar klientas gyvena didmiestyje, ar ne, o interneto tipas, jeigu turi daugiau nei dvi skirtingas reikšmes, galėtų būti perprogramuotas į kelias skirtingas dimensijas, naudojant fiktyvius kintamuosius (Cohen, Cohen, West & Aiken, 1983). Šis procesas yra atvaizduotas 9 pav. Po pertvarkių duomenų rinkinį sudaro 274 tūkstančiai stebinių ir 29 stulpeliai.

$$\begin{bmatrix} \text{Grade} \\ 7 \\ 7 \\ 8 \\ 9 \\ 7 \\ 9 \\ 8 \\ 7 \end{bmatrix} = \begin{bmatrix} \text{Grade}_7 & \text{Grade}_8 & \text{Grade}_9 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

pav 9. Fiktyvūs kintamieji

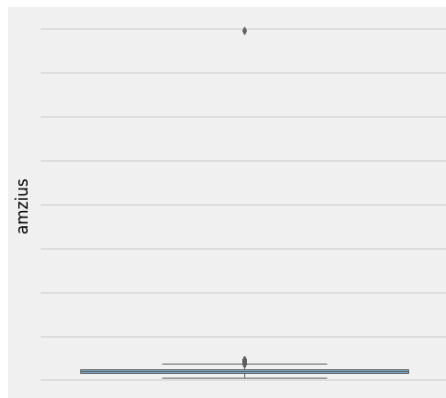
Kai kurie statistiniai metodai reikalauja, kad skaitinių kintamųjų reikšmės būtų normalizuotos, t.y. atitiktų vieningą skalę. Čia duomenų normalizavimui buvo pasirinkta minimumo-maksimumo mastelio mažinimo technika, kuri normalizuoja skaitinius kintamuosius tarp 0 ir 1, šio metodo formulė (Pedregosa et al., 2011):

$$X_{\text{norm}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}.$$

Čia X reprezentuoja originalią reikšmę, $X_{\min}$ ir $X_{\max}$ nurodo minimalią ir maksimalią reikšmes duomenų rinkinyje nurodytam kintamajam., o X_{norm} yra normuota reikšmė.

3.1.2. Duomenų valymas

Duomenų rinkinyje yra randami keli šimtai stebinių, kurie neturi priskirtos amžiaus reikšmės. Kadangi keli šimtai iš kelių šimtų tūkstančių nėra didelis skirtumas, tokius stebinius galime tiesiog pašalinti, neprarasdami didelės dalies informacijos. Taip pat žiūrint į skaitinių kintamųjų pasiskirstymą yra pastebima, kad pasitaiko išskirčių. Tokį pavyzdį galima pamatyti 10 pav. kur yra atvaizduojamos amžiaus reikšmės y ašyje, kur kelios labai išsiskiria nuo pasiskirstymo.



pav 10. Išskirties pavyzdys

Bet kuriuo atveju, ar išskirtis atsirado dėl įvesties klaidos, ar dėl kitos priežasties, jų suvaldymas yra svarbi užduotis atliekant bet kokią duomenų analizę. Binarinio klasifikavimo

atveju, išskirtys gali nukreipti prognozę į mažumą, pavyzdžiui, jeigu pradinuose duomenyse klasei B priklauso kelios išskirtys, o klasei A neturi jokių išskirčių, modelis gali priskirti visus naujus atvejus, kuriuose nebus išskirties klasei A. Tokiu būdu atsiranda didesnis statistinis reikšmingumas prie galimai, ne tokių reikšmingų kintamųjų, kas veda prie mažesnio modelio tikslumo. Išskirčių identifikavimui yra naudojamas interkvartilių metodas, kurio formulė:

$$\begin{cases} \text{Išskirtis,} & \text{jei } x < Q_1 - k \cdot \text{IQR arba } x > Q_3 + k \cdot \text{IQR} \\ \text{Neišskirtis,} & \text{kitu atveju.} \end{cases}$$

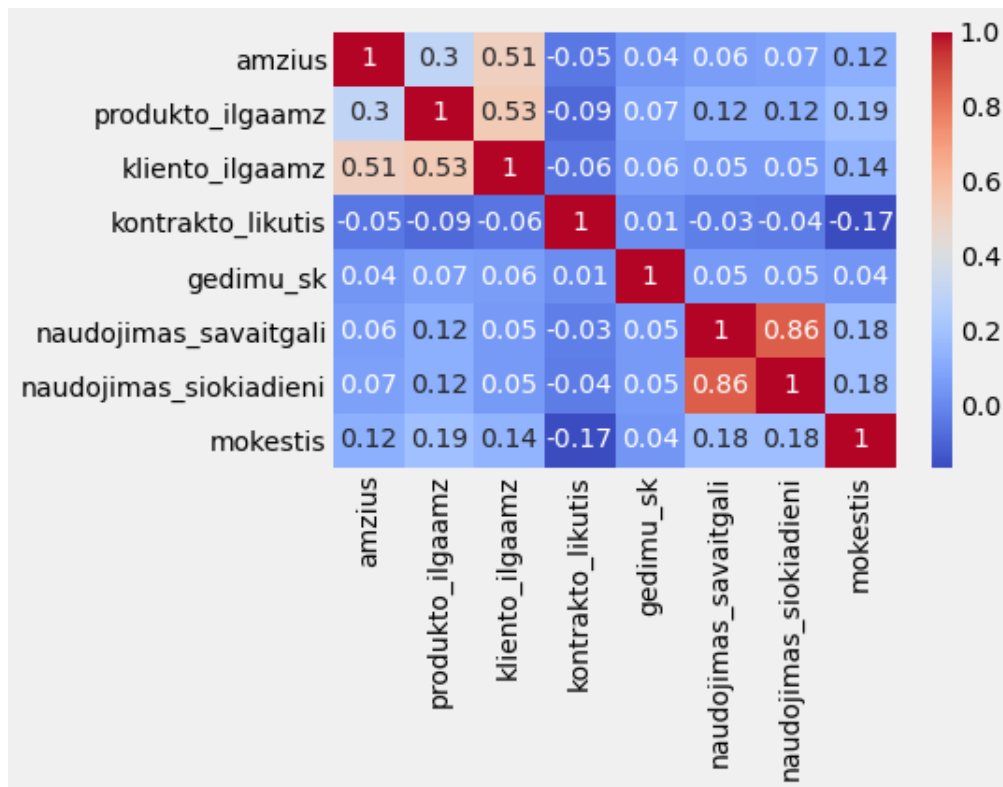
Kur x yra duomenų reikšmė, Q_1 ir Q_3 yra kintamojo pirmas ir trečias kvartilis, o $\text{IQR} = Q_3 - Q_1$ yra interkvartilių diapozonas, o k yra konstanta, nulemianti nuokrypių mastą. Įprastinė k reikšmė yra 1.5. Pritaikius šį metodą mūsų skaitiniams duomenims su k reikšme 1.5, 13 % turėtų stebinių, kas yra nemaža dalis duomenų, todėl ši reikšmė padidinam iki 2 ir yra prarandama vos 6 % stebinių.

3.1.3. Skaitinių kintamųjų apžvalga

Šiame skyrelyje pasitelkus grafinį atvaizdavimą, bus atliekamos pirminės įžvalgos apie tikslo kintamąjį, atsižvelgiant į duomenų rinkinyje esančius skaitinius kintamuosius, bandoma įžvelgti prasmingų priklausomybių. Skaitiniams kintamiesiems analizuoti, yra nupiešiama dėžutės sklaidos diagrama kiekvienam kintamajam, atsižvelgiant į tikslo kintamojo grupes. Pirmiausiai bus atliekama koreliacinė analizė. Multikolinearumas dvireikšmio klasifikavimo uždavinyje gali kilti dėl tokių priežasčių kaip:

- persotinimas - didelis kintamųjų skaičius tampa problema, kadangi modelis gali per daug prisitaikyti treniruojamajam duomenų rinkiniui, ir modelis tampa labai jautrus duomenų pasikeitimui,
- koeficientų reikšmių išsikreipimas - esant multikolinearumui, tampa sudėtinga nustatyti kiekvieno individualaus predikto indėlį į modelio prognozę, kadangi stipriai koreliuojantys kintamieji sukuria perteklinę informaciją ir modeliui gali būti sunku parinkti atitinkamus teisingus koeficientus.

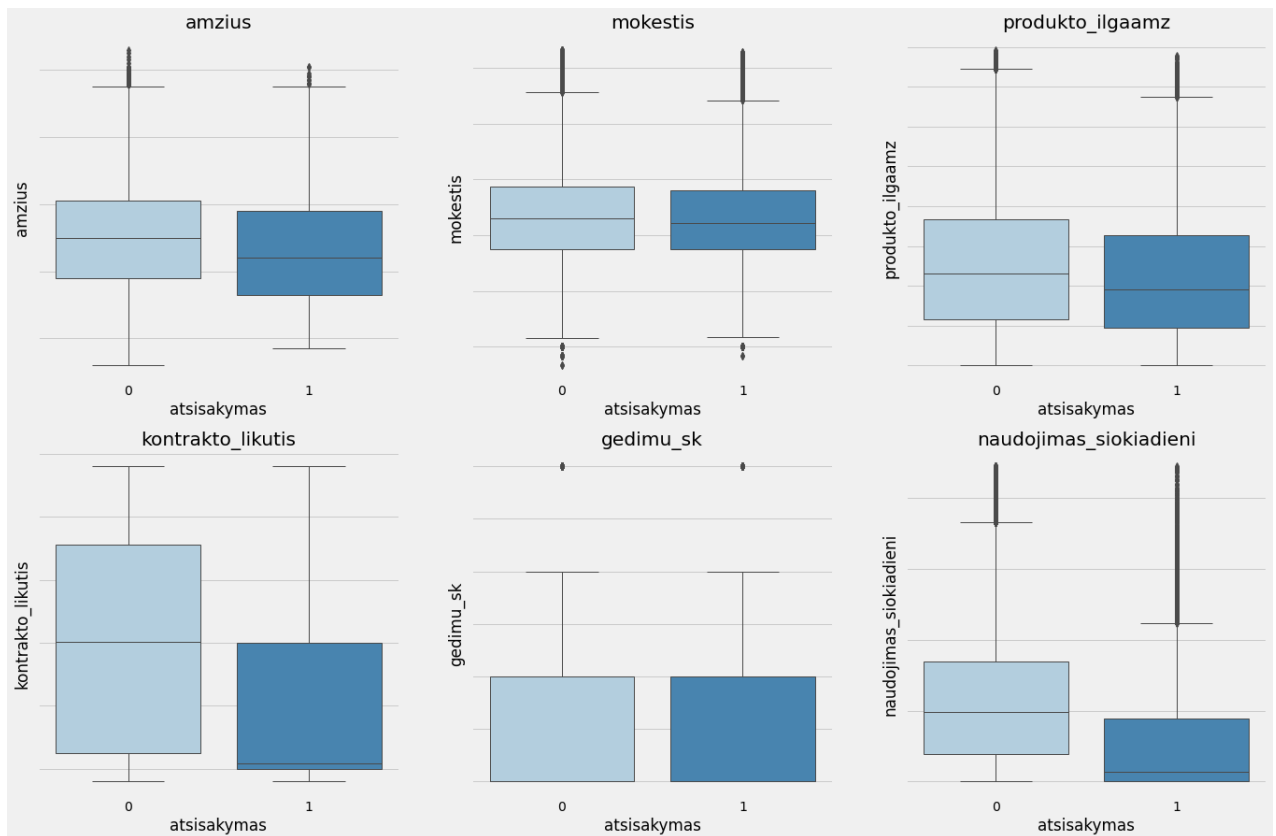
Šie abu veiksniai veda prie klaidingos prognozės, ko binarinio klasifikavimo uždavinys stengiasi išvengti, todėl susitvarkyti su tarpusavyje koreliuojančiais kintamaisiais yra svarbu. Analizuojant 11 pav. yra matoma, kad yra labai stipri koreliacija tarp naudojamo kintamųjų (savaitgalį ir šiokiadienį), taip pat kliento ilgaamžiškumas ganėtinai stipriai koreliuoja su amžiaus kintamuoju ir produkto ilgaamžiškumu. Po koreliacinės analizės, siekiant išvengti multikolinearumo yra atsisakoma paminėtų dviejų kintamųjų – paslaugos naudojimo savaitgalį ir kliento ilgaamžiškumas. Amžiaus ir produkto ilgaamžiškumo kintamieji nors ir turi ryškesnę koreliaciją, tačiau ji nėra tokia didelė, todėl šie kintamieji yra paliekami.



pav 11. Koreliacinė matrica

Toliau pateikiama pasiskirstymo analizė, lyginant tikslo kintamąjį su likusiais skaitiniais kintamaisiais. Nagrinėjant 12 pav. galima įžvelgti tokias tendencijas:

1. amžiaus kintamasis potencialiai gali būti reikšmingas - atsisakančių klientų amžius yra pasiskirstęs žemiau,
2. mokestis, t.y. mokami pinigai už paslaugas neatrodo kaip reikšmingas kintamasis,
3. produkto ilgaamžiškumo pasiskirstyme dvejose klasėse matosi panaši tendencija, kaip ir amžiaus kintamajame, galimai vieno iš šių kintamųjų teks atsisakyti, nors koreliacija ir neatrodė labai reikšminga sprendimas bus priimtas galutinio modeliavimo žingsnyje,
4. kontrakto likutis nurodo, ar terminuotas kontraktas yra jo pabaigoje (jeigu kontraktas yra neterminuotas, reikšmė yra nulinė), čia matoma, kad atsisakantys klientai yra labiau linkę atsisakyti paslaugų, kai yra kontrakto pabaigoje,
5. registruotų gedimų skaičius neatrodo kaip reikšmingas kintamasis, tai gali būti ir dėl to, kad dažnai ši reikšmė yra lygi nuliui,
6. naudojumas - mažesnis naudojumas linksta prie didesnio klientų atsisakymo.

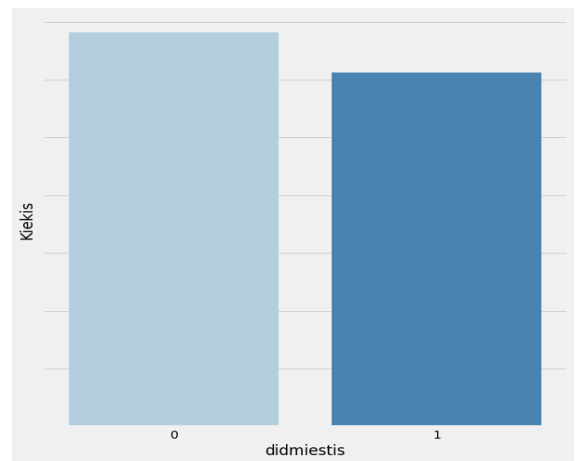
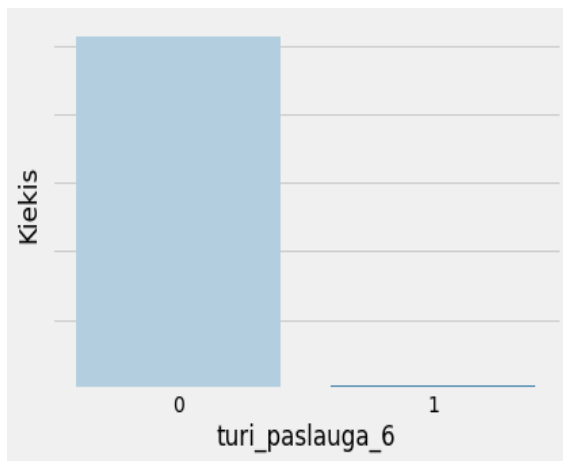


pav 12. Skaitinių kintamųjų sklaidos diagramos tikslo kintamojo analizei

Taigi, iš analizės galime nujausti, kad potencialiai reikšmingi kintamieji gali būti amžiaus, produkto ilgaamžiškumo, laiko iki kontrakto pabaigos ir naudojamumo kintamieji.

3.1.4. Kategorinių kintamųjų apžvalga

Skaitinių duomenų analizėje sumažinant dimensijų skaičių, siekiant sustiprinti modelio prognozę pašalinome kelis skaitinius kintamuosius. Kategorinių duomenų analizėje taip pat galima padaryti kelis žingsnius, siekiant atsisakyti neinformatyvių duomenų. Pirmiausiai patikrinama, ar nėra mažo kintamumo kintamųjų. Mažas kintamumas dažniausiai nesuteikia daugiau informacijos tikslo kintamojo prognozei, pavyzdžiui, jeigu kategorinis kintamasis turi tik vieną reikšmę labai didelei proporcijai stebinių, nėra didelio tikslo jo atskirti per kelias klases. Mūsų atveju radome kelis tokius atvejus, pavyzdžiui - 13 pav. yra matomas toks pavyzdys, t.y. didžioji dalis duomenų yra tokiam segmente, kad neturi penktos paslaugos, taip pat palyginimui yra pridėta subalansuotas kategorinis kintamasis, kurių kiekiai abejose kategorijose yra artimos. Kintamieji su disbalansu buvo išimti, kadangi jų buvo rasta dar keli, o su didesniu balansu kintamieji potencialiai yra labiau patikimi, kadangi jie atskleidžia daugiau informacijos.

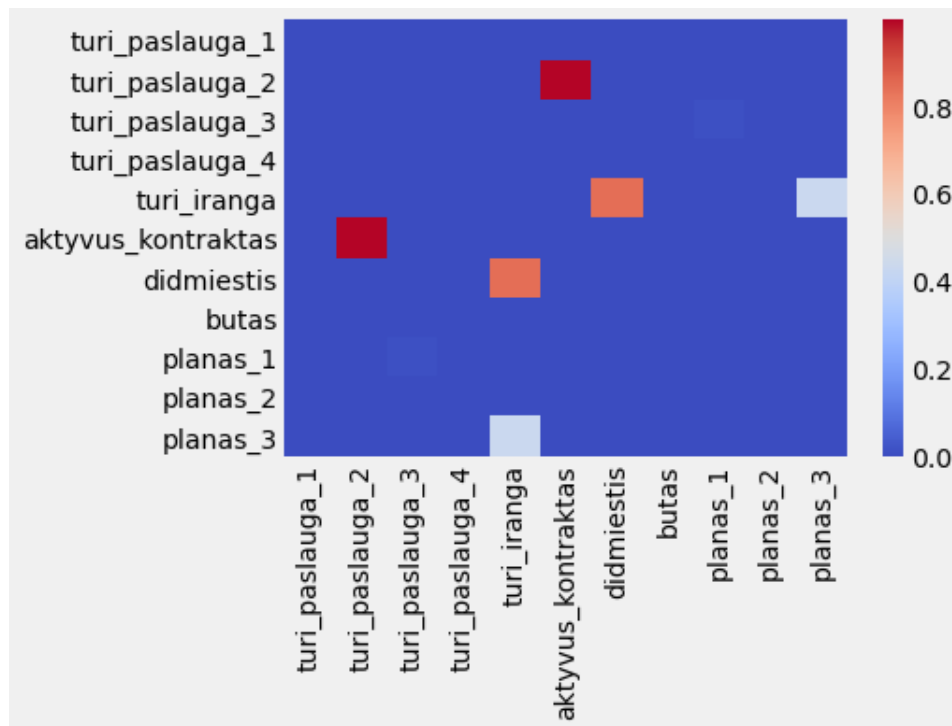


pav 13. Kategorinio kintamojo kiekių disbalansas ir balansas

Su kategoriniais kintamaisiais koreliaciją skaičiuoti nebūtų matematiškai teisinga. Čia gali padėti χ^2 testas, kuris tikrina:

$$\begin{cases} H_0 : \text{nėra priklausomybės tarp dviejų kintamųjų,} \\ H_1 : \text{yra priklausomybė tarp dviejų kintamųjų.} \end{cases}$$

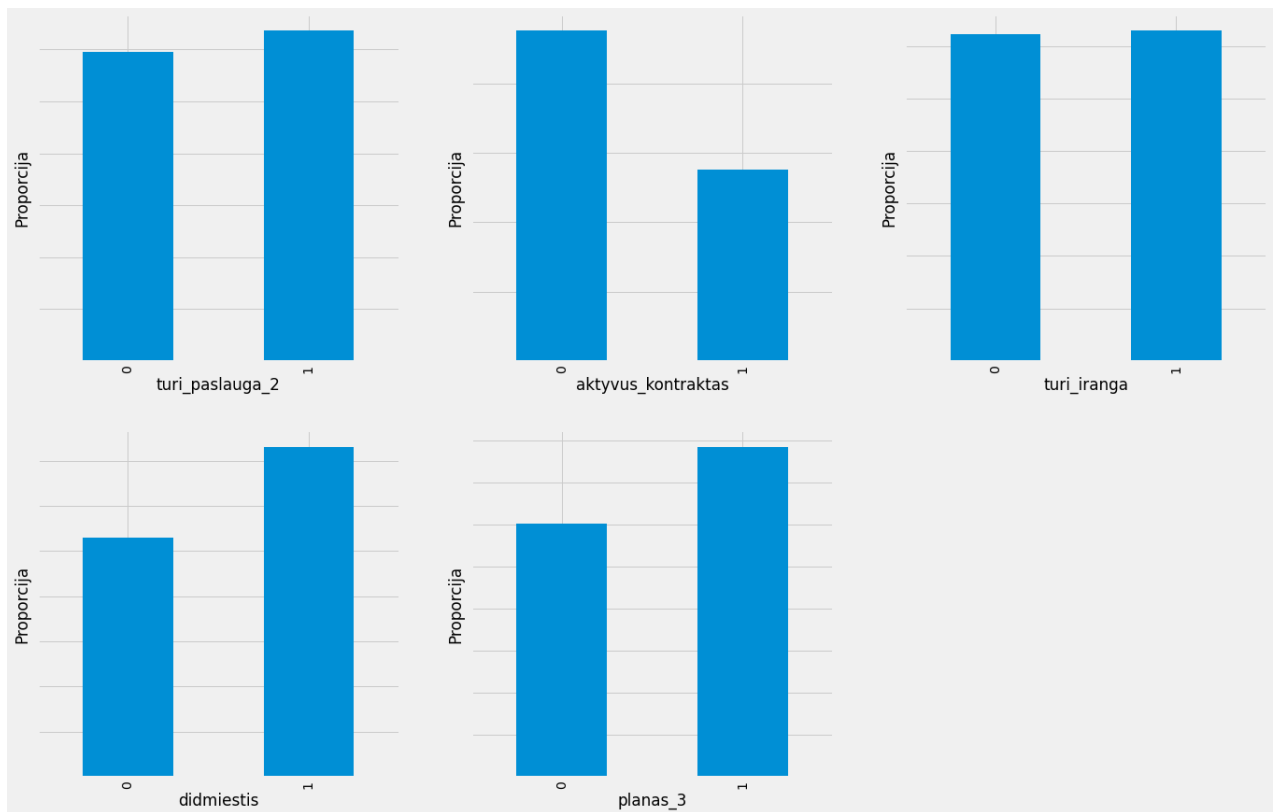
14 pav. yra atlikto χ^2 testo rezultatai tarp visų potencialių prognozuojamųjų kategorinių kintamųjų. Čia yra matoma, kad yra didelė priklausomybė tarp daugumos iš šių kintamųjų, kadangi p-reikšmės daug kur yra nulinės ir testą tenka atmesti. Šiuo atveju yra paliekami tie kategoriniai kintamieji, kurie turi bent viena priklausomybę su kažkuriuo kitu kintamuoju, tai reiškia, kad iš pateiktų kategorinių kintamųjų paliekame tik penkis (antros paslaugos turėjimas, įrangos nusipirkimo faktas, trečio plano turėjimas, kontrakto aktyvumo reikšmė, gyvenimo didmiestyje reikšmė).



pav 14. χ^2 testo p reikšmės tarp dviejų kintamųjų

Padarykime šių kintamųjų analizę, lyginant jų kiekius skirtingose grupėse vizualiai, tikslo kintamojo atžvilgiu. 15 pav. yra matomas grafikas, kuriame galima padaryti kategorinių kintamųjų analizę. Čia yra atvaizduojamos atisakusių klientų kategorinių kintamųjų santykinės proporcijos, rodančios, kuri dalis teigiamą arba neigiamą reikšmę turinčių klientų atsisakė paslaugos. Analizuojant šį grafiką galima iškelti tokias hipotezes:

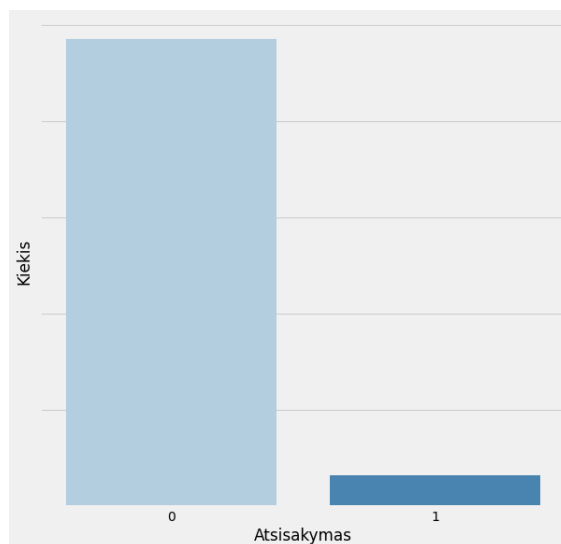
- antros paslaugos, ar įrangos prisipirkimas turėjimas neturi didelės įtakos klientų atsisakymui,
- klientai be aktyvios terminuotos sutarties yra labiau linkę atsisakyti paslaugų,
- klientai gyvenantys didmiestyje yra labiau linkę išeiti,
- trečio plano turėtojai labiau atsisako paslaugų.



pav 15. Atsisakiusių klientų kategorinių kintamų proporcijos

3.1.5. Tikslo kintamojo disbalansas

Dvireikšmio klasifikavimo uždaviniuose klasių disbalansas atsitinka tuomet, kai vienoje klasėje turimų duomenų skaičius yra ženkliai didesnis, nei kitoje. Tarkime, jeigu duomenų rinkinyje esantys stebiniai yra 90 % vienoje klasėje yra sakoma, kad tikslo kintamasis yra išbalansuotas. Toks atvejis nutinka mūsų duomenyse, **16** pav. yra matoma, kad neatsisakančių klientų skaičius yra ženkliai didesnis, nei atsisakiusių. Tai gali būti problema, nes mašininio mokymosi algoritmai dažnai siekia maksimizuoti bendrą tikslumą, o prognozuojant išbalansuotus duomenis, galima gauti didelį tikslumą vien prognozuojant į vieną pusę. Ypatingai tai yra problema, kada mažumos prognozavimas yra aktualesnis, kas ir yra mūsų atveju, norime stebėti klientus kurie yra linkę išeiti.



pav 16. Tikslo kintamojo klasių disbalansas

Kadangi duomenyse nėra didelės dinamikos, t.y. dauguma stebinių yra sąlyginai panašūs, pirminiame eksperimente bus dirbama su stebinių pašalinimu atsitiktiniu metodu, tai taip pat pagreitins modeliavimo skaičiavimus, nes stebinių skaičius bus mažesnis. Yra du būdai, kaip galima atlikti duomenų balansavimą mašininio mokymosi modeliuose - vienas iš jų, subalansuoti visą duomenų rinkinį ir atlikti mokymo ir validavimo rinkinių padalinimą su balansuotais duomenimis, arba atlikti balansavimą mokymo duomenimis, taip paliekant validavimo rinkinį nesubalansuotą. Prieš priimant sprendimą, apžvelkime privalumus ir trūkumus balansuojant visą duomenų rinkinį.

Mokymo-validavimo duomenų skirstymo privalumai balansuotam rinkiniui:

- Reprezentatyvumas: subalansavus duomenis prieš padalinimą, rinkiniai turės panašų klasių pasiskirstymą, o tai suteikia reprezentatyvesnį modelio įvertinimą.
- Nuoseklumas: balansuojant duomenis prieš paskirstymą į dvi grupes užtikrins nuoseklesnes modelio įverčių metrikas įvairiose eksperimentų eigose.

Mokymo-validavimo duomenų skirstymo trūkumai balansuotam rinkiniui:

- Nesutapimas su realybe: nesubalansuotas rinkinys geriau atspindi tikrą problemą, kadangi atsisakančių klientų visada bus mažiau, negu išsilaikiusių.
- Pernelyg optimistiški rezultatai: subalansavus duomenis prieš jų padalijimą, modelio tikslumas gali nuteikti per gerai.

Atsižvelgiant į šias pastabas, priimamas sprendimas visus duomenis iš pradžių subalansuoti, o tik tada atlikti padalinimą, nes šiame darbe pirmiausiai norima paeksperimentuoti su skirtingais mašininio mokymosi metodais, o stabilesnės modelių tikslumo metrikos bus lengviau interpretuojamos. Kadangi duomenyse nėra didelės dinamikos, t.y. dauguma

stebinių yra sąlyginai panašūs, pirminiame eksperimente bus dirbama su stebinių pašalinimu atsitiktiniu metodu, tai taip pat pagreitins modeliavimo skaičiavimus, kadangi stebinių skaičius bus mažesnis.

3.2. Modeliavimas

Šiame skyriuje sukuriama binarinio klasifikavimo modeliai, bandant suprognozuoti klientų elgseną. Tarp modelių bus atliekamas palyginimas, naudojant įvairias metrikas, lyginant sumaišties matricas bei ROC kreives. Taip pat patikrinamos kintamųjų svarbos modelyje, padaromas palyginimas su įžvalgomis, kurios buvo atliktos naudojant vaizdinę duomenų analizę.

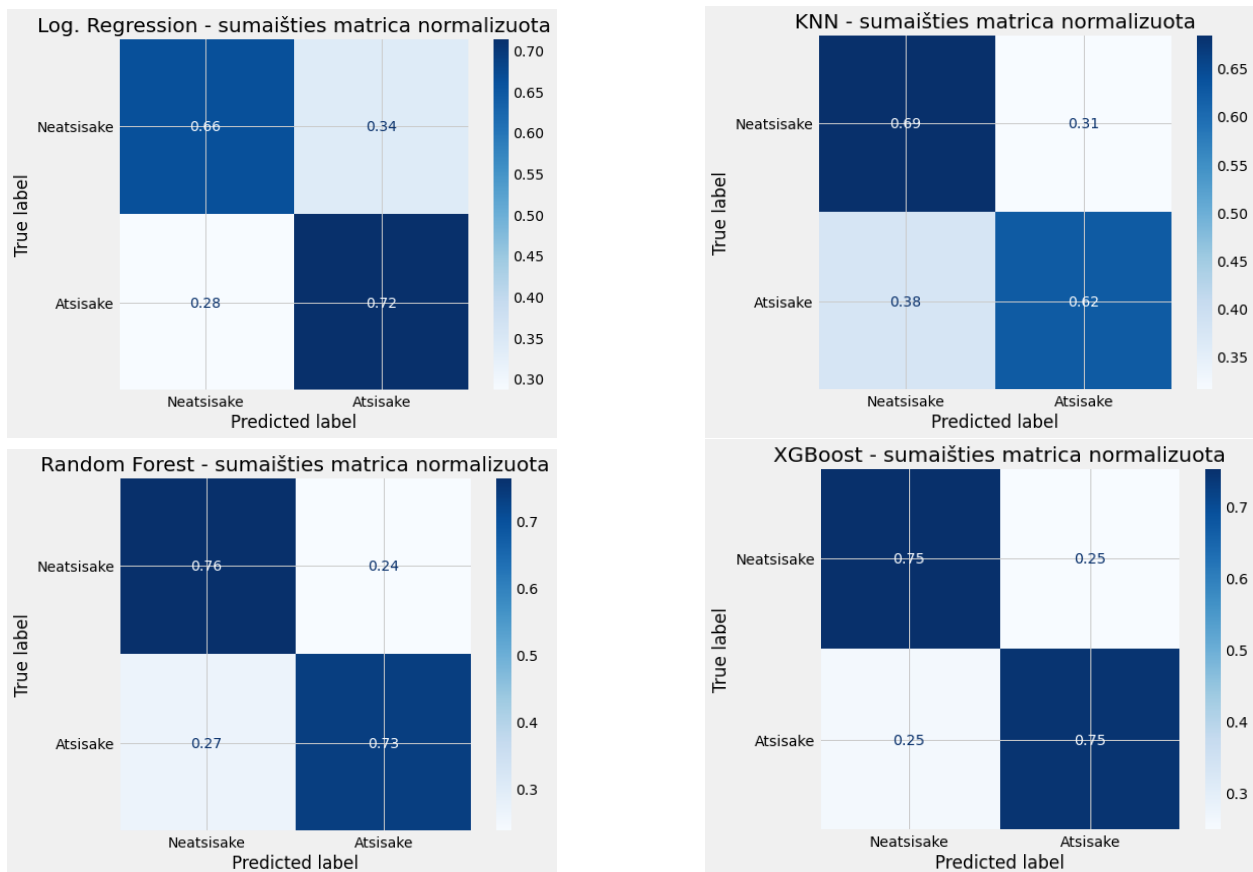
Testiniai ir treniruojamieji duomenys. Čia duomenų rinkinys yra įskaitomas į dvi dalis, treniruojamam rinkiniui paliekant 80 % stebinių, testiniam – 20 %. Modeliai bus kūrimi naudojant *scikit-learn* biblioteką (Pedregosa et al., 2011), kurioje yra implemenuota daug klasifikavimo metodų, bei *XGBoost* (Chen & Guestrin, 2016). Šiame darbe bus pritaikomi metodai su tokiomis funkcijomis:

- K artimiausių kaimynų metodas (KNN),
- paprastoji logistinė regresija (Log. Regression),
- atsitiktinis miškas (Random Forest),
- XGBoost (eXtreme Gradient Boosting).

K artimiausių kaimynų, logistinės regresijos ir atsitiktinio miško metodai yra užprogramuoti *scikit-learn* bibliotekos funkcijose ***KNeighborsClassifier***, ***LogisticRegression***, ***RandomForestClassifier*** (Géron, 2022), o XGBoost metodas *xgboost* bibliotekoje funkcija ***XGBClassifier*** (Chen & Guestrin, 2016). Pirminiame eksperimente, taikant visus modelius yra naudojami tie patys duomenys, t.y. mokymosi duomenys visiems modeliam naudojami tie patys prisotinant skirtingus modelius, bei validavimas vyksta su tais pačiais testiniais duomenų rinkiniais. Pirminis eksperimentas yra atliekamas su pagal nutylėjimą duotais metodų hiperparametrų reikšmėmis.

3.3. Sumaišties matricos

Analizuojant 17 pav. yra matomos sumaišties matricos visiems keturiems bandomiems metodams, prognozuojant testiniam duomenų rinkiniui. Matosi, kad visi metodai prognozuoja geriau, nei atsitiktinis spėjimas (klaidingos ir teisingos prognozės nepasiskirsto santykiu 50/50), kas jau yra geras ženklas. Teisingai teigiamą rezultatą, kuris mums yra aktualiausias, prasčiausiai prognozavo KNN metodas, geriausiai – XGBoost, tačiau atsitiktinis miškas minimaliai lenkia XGBoost prognozuojant teisingą neigiamą rezultatą. Logistinė

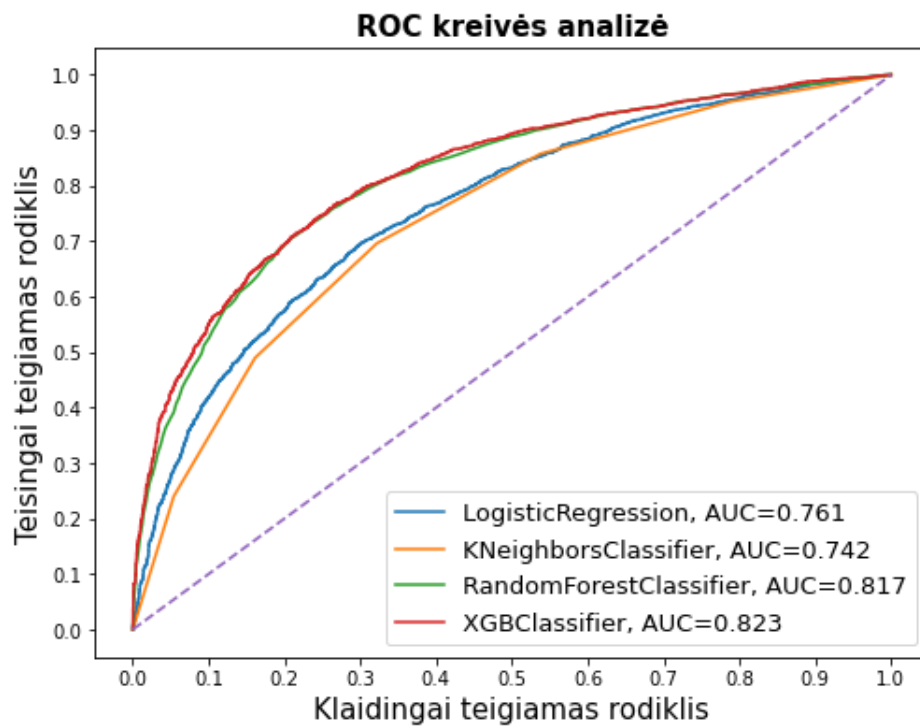


pav 17. Sumaišties matricos

regresija pasirodo geriau nei KNN, bet atsilieka nuo miškų metodu. Tai yra tik pirmas pažiūrėjimas sumaišties matricių, tolimesniame skyrelyje bus palyginimas palyginamos ir kitos metrikos.

3.4. ROC kreivė

ROC kreivė parodo jautrumą tarp teisingo teigiamo ir klaidingo teigiamo koeficientų, keičiant binarinio klasifikavimo atskyrimo slenkstį (kuris dažniausiai yra lygus 0.5). Geresnis modelis yra tas, kuris yra labiau nutolęs nuo pilkos punktyrinės linijos (18 pav.). Analizuojant šį pavyzdį matosi, kad labiausiai nutolus yra XGBoost ir atsitiktinio miško kreivės kurių AUC reikšmės, parodančios plotą po kreive yra artimos. Visgi XGBoost AUC reikšmė yra didžiausia. Pagal ROC-AUC kriterijų K artimiausių kaimynų metodas pasirodo prasčiausiai, jis yra arčiausiai pilkos kreivės, bet ne taip toli nutolęs nuo logistinės regresijos. Galiausiai, logistinė regresija yra tarp atsitiktinių miškų metodų ir KNN.



pav 18. ROC kreivės atskiriems modeliams

3.5. Modelių metrikos

Žiūrint į tikslumo metrikas 3 lentelėje matomos panašios tendencijos – beveik visose metrikose K artimiausių kaimynų metodas nusileidžia tiek logistinei regresijai, tiek atsitiktiniam miškui ir XGBoost, o logistinė regresija visose metrikose atsilieka abiemis metodams, grįžta medžiais. XGBoost metodas turi šiek tiek geresnę F1 metriką ir preciziškumą, tačiau skirtumas nėra ryškiai stipresnis. Jeigu reiktų rinktis tarp šių dviejų metodų, logika sako, kad reiktų rinktis XGBoost, tačiau, kadangi metrikų skirtumai nėra dideli, galima būtų padaryti gilesnę analizę - pvz. kuris modelis apsimoko ir apskaičiuoja prognozes greičiau.

3 lentelė. Apibendrintos tikslumo metrikos

Klasifikatorius	Preciziškumas	Jautrumas	F1 statistika	Tikslumas	AUC
Logistinė regresija	0.720	0.682	0.700	0.693	0.761
KNN	0.696	0.682	0.689	0.687	0.742
Atsitiktinis miškas	0.734	0.754	0.743	0.748	0.816
XGBoost	0.739	0.752	0.746	0.749	0.824

3.6. Kintamųjų svarbiai

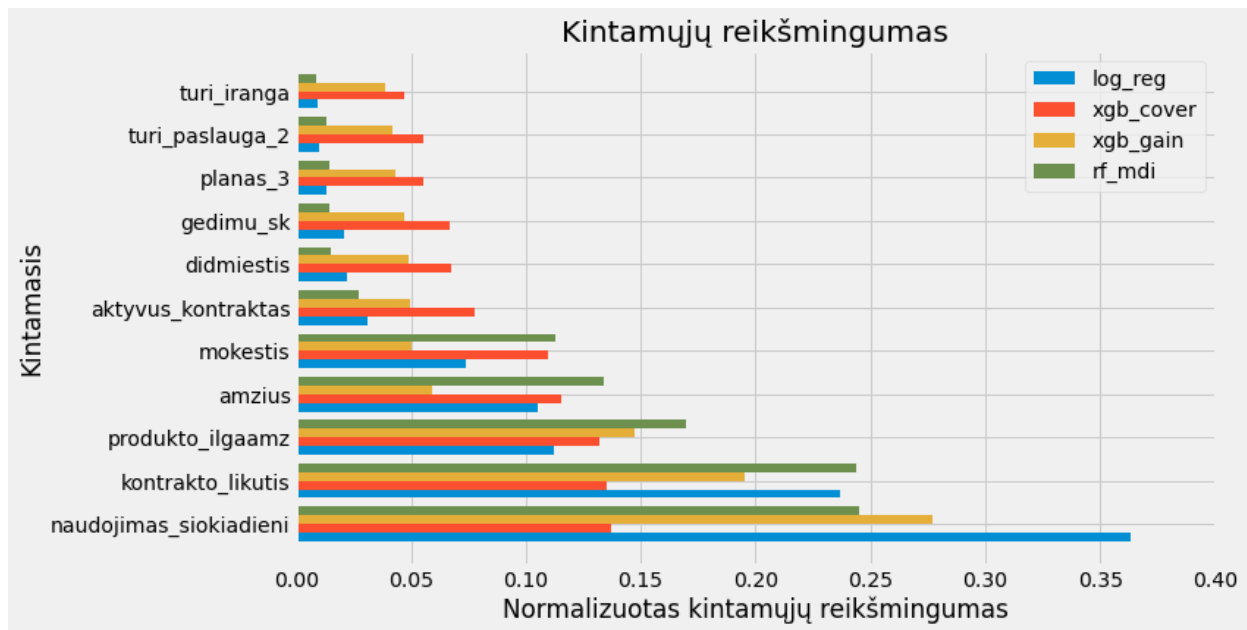
Norint suprasti, kurie faktoriai labiausiai lemia tam tikrą tikslo kintamojo prognozę, būtų padaryti kintamųjų svarbių analizę. 2.4 skyriuje buvo pristatyti naudotų metodų pagrindinės kintamųjų svarbumo metrikos. XGBoost Svarbio metrika yra jautresnė didesniems duo-

menų rinkiniams, kadangi ji yra grįsta kintamųjų pasirodymo dažnumu skirtingose mokymo dalinimo iteracijose. Reikšmingesni kintamieji įgaus ženkliai didesnę svorį ir metrikos tokiu būdu gali tapti paslinktos. Kadangi šiame darbe yra dirbama su ganėtinai nemažu duomenų rinkiniu, svorio nebus analizuojama. Taip pat reiktų paminėti, jog, kadangi k -artimiausių kaimynų metodas nėra parametrinis, jis neturi globalaus modelio, kuris galėtų būti naudojamas skaičiuojant kintamųjų svarbą tuo pačiu būdu, kaip tą galima daryti su parametriniais metodais. Todėl kintamųjų svarbumas naudojant KNN metodą nėra pateikiamas.

19 pav. yra pateikiamos pagrindinių kintamųjų reikšmingumo metrikų grafikas prieš tai pateiktiems mašininio mokymosi modeliams. Čia matomos keturios skirtingos metrikos, kurios buvo išskaičiuotos iš trijų modelių. Įdomu tai, kad šie rodikliai pateikė labai panašius kintamųjų svarbo reitingus, tik nežymiai skiriasi santykinės svarbos proporcijos tarp kintamųjų. Tačiau nepaisant šių skirtumų, visos metrikos išryškino nedidelį pagrindinių kintamųjų rinkinį, kuris turėjo didelės įtakos modelio veikimui. Apskritai, šie rezultatai suteikia tvirtų įrodymų apie šių kintamųjų tinkamumą ir rodo, kad pateikti modeliai nuo jų labai priklauso. Einant į detales, čia labiausiai išsiskiria trys kintamieji.

- Naudojimo kintamasis, kas yra natūralu - mažiau naudojantis paslaugą klientas yra labiau linkęs jos atsisakyti, ką jau matėme iš duomenų priešanalizės.
- Laikas iki kontrakto pabaigos. Telekomunikacijų srityje klientai dažnai įkontraktinami su tam tikra nuolaida, klientui norint ją nutraukti nepasibaigus kontraktui dažna praktika yra susimokėti netesybas, ko klientas ne visada nori daryti, todėl jis yra labiau linkęs nutraukti sutartį, kai kontraktas yra pasibaigęs
- Ilgaamžiškumas, kas taip pat sutampa su hipoteze iš duomenų priešanalizės ir bendrų prielaidų - ilgaamžiškas klientas yra labiau vertingas ir labiau linkęs likti su paslaugomis.

Taip pat reiktų pastebėti, kad skaitiniai kintamieji visur turi didesnę svarbumą lyginant su kategoriniais kintamaisiais. Tai gali būti dėl to, kas skaitiniai kintamieji įprastai suteikia daugiau informacijos ir kintamumo duomenyse ir todėl turi didesnę poveikį modelio prognozei. Reiktų atkreipti dėmesį, kad didmiesčio ir kategorinis kintamasis yra labiausiai neišbalansuotas kintamasis (13 pav.), todėl jis modeliui galėjo suteikti daugiau informacijos.



pav 19. Kintamųjų reikšmingumo normalizuotos reikšmės

3.7. Kryžminė patikra

Pabandykime atlikti kryžminę validaciją mūsų duomenų rinkiniui ir palyginkime su pirminiais rezultatais, kur metrikos buvo modelio tikslumo metrikos buvo skaičiuotos tik su viena iteracija. 4 pav. yra pateikiami rezultatai kuriuose matoma, kad logistinės regresijos ir KNN metrikos labai supanašėjo, o atsitiktinio miško ir XGBoost praktiškai susivienodino. Lyginant su pirminiais rezultatais, pateiktais 3 lentelėje, galima išvelgti ženklų pagerėjimą KNN metodui, kuris pasivijo logistinę regresiją, ir ženklų pagerėjimą atsitiktinių miškų metode, kuris prisivijo XGBoost metodą. Čia eksperimentas buvo atliktas su k-perlenkimų kryžminės validacijos parametru $k = 5$, įdomumo dėlei, siekiant didesnio tikslumo buvo atliktas tyrimas su didesniu $k = 10$, tačiau rezultatai buvo labai panašūs ir skirtumo nuo eksperimento su $k = 5$ nesudarė.

4 lentelė. Tikslumo metrikos atlikus k-perlenkimų kryžminę paiešką, kai $k = 5$

Klasifikatorius	Preciziškumas	Jautrumas	F1 statistika	Tikslumas	AUC
Logistinė regresija	0.694	0.694	0.694	0.694	0.765
KNN	0.695	0.695	0.695	0.695	0.746
Atsitiktinis miškas	0.750	0.749	0.749	0.749	0.818
XGBoost	0.750	0.750	0.750	0.750	0.824

3.8. Hiperparametrų paieška

Šiame skyriuje bus atliekami eskperimentai, ieškant geriausių hiperparametrų kombinacijų kiekvienam prieš tai naudotam modeliui gardelės ir atsitiktinės paieškos metodais.

Gardelės metodas yra daug resursų kainuojantis metodas, vos tik padidinus hiperparametrų poaibį viena dimensija, ar keliomis reikšmėmis, apskaičiavimo greitis labai išauga, o optimalaus modelio radimas gali ir nepasiteisinti. Tas buvo patirta šiame darbe atliekant hiperparametrų paiešką XGBoost metodui, vienam eksperimente buvo paimamos $2^8 = 256$ galimos skirtingos kombinacijos, kitu atveju $2^4 * 3^4 = 1296$ kombinacijų. Šių parametrų prasmė ir suteikiamos reikšmės:

1. *learning_rate* reikšmė kontroliuoja žingsnio dydžio sumažėjimą kiekvienos didinimo iteracijos metu. Mažinant šį parametą galimai rezultatai bus geresni, tačiau yra padidinama skaičiavimų kaina. Pasirinktos reikšmės: [0.1, 0.3].
2. *max_depth* nurodo maksimalų medžio gylį. Gilesnis medis gali identifikuoti kompleksiškesnes priklausomybes tikslo kintamojo atžvilgiu, tačiau atsiranda rizika persotinimui. Pasirinktos reikšmės: [3, 5] + [6].
3. *min_child_weight* yra antrinės viršūnės stebinių minimalus skaičius, leidžiantis sukontroliuoti persotinimą pridėdant papildomų apribojimų. Pasirinktos reikšmės [1,3], didesniems duomenų rinkiniams būtų prasmingiau šį skaičių padidinti.
4. *gamma* reguliuoja minimalų nuostolių sumažinimą, kuris yra reikalingas norint padaryti kitą išskaidymą lapo viršūnėje. Pasirinktos reikšmės [0.1, 0.3] + [0]. Priskiriant šią reikšmę 0, medžiai gali būti didesni, kadangi šis parametras neberiboja medžių išsiplėtimo.
5. *subsample* nurodo duomenų rinkinio dalį, naudojamą kiekvienam medyje. Pasirinktos reikšmės [0.6, 0.8] + [1]. Pasirenkant šio hiperparametro reikšmę 1, kiekvienam medyje yra naudojamas visas duomenų rinkinys.
6. *colsample_by_tree* nurodo kintamųjų dalį, treniruojant kiekvieną medį. Pasirinktos reikšmės [0.6, 0.8, 1]. Vėlgi, pasirenkant šio hiperparametro reikšmę 1, kiekvienam medyje yra naudojami visi kintamieji, kas gali privesti prie persotinimo.
7. *n_estimators* yra medžių skaičius modelyje, didinant šį hiperparametrą ilgėja mokymo laikas. Pasirinktos reikšmės [100, 200].
8. *scale_pos_weight* yra kintamasis, kuris yra naudojamas su išbalansuotais duomenimis. Nors naudojamas rinkinys jau yra subalansuotas, paeksperimentuota su reikšmėmis 1, kuris nededa jokio svorio nei vienai klasei ir 2, kuris suteikia didesnę svorį teigiamai prognozei.

5 lentelėje yra matomas šio eksperimento rezultatas, kuriame yra matoma, kad antru atveju skaičiavimai truko 6 kartus ilgiau, tačiau geriausio modelio rezultatas tikslumo atžvilgiu pagerėjo vos per 0.0003. Kad ir kaip bebūtų, po šios gardelės paieškos yra rastas geresnis

modelis, lyginant su gautu po kryžminės patikros XGBoost modeliu, kadangi praeito modelio tikslumas siekė 0.75 (4 lent.).

5 lentelė. Gardelės paieškos rezultatai su dviem skirtingais kombinacijų skaičiais XGBoost metodui

Kombinacijų skaičius	256	1296
Laikas	4 min 45 s	29 min 55 s
Tikslumas	0.7617	0.7620

Šiame darbe bus siekiama pagerinti visus prieš tai skaičiuotus modelius, todėl bus atliekami tokie eksperimentai ir kitiems metodams. Identiškų parametrų, natūralu, kad tikrinti negalime, kadangi skirtingi modeliai naudoja skirtingus hiperparametrus, tačiau galime paeksperimentuoti su daugiau-mažiau parametrų kombinacijų ir patikrinti santykį, ar pavyksta išgauti geresnį tikslumą. Atsitiktiniam miškui eksperimentas atliekamas su $3^4 * 2 = 162$ ir $4^4 * 2 = 512$ kombinacijų. Šių parametrų prasmė atsitiktiniam miškui ir parinktos reikšmės:

1. *n_estimators* kaip ir XGBoost metode, nurodo medžių skaičių miške. Eksperimentui pasirinktos reikšmės [100, 200, 500] + [1000].
2. *max_features* nurodo kintamųjų skaičiaus dalį, iš kurių yra svarstoma, kai yra ieškomas naujas geriausias padalinimas kiekvienoje sprendimų medžio viršūnėje. Pasirinktos reikšmės ['sqrt', 'log2', 0.5] + [None]. *sqrt* reikšmė ištraukia šaknį iš kintamųjų skaičius ir suapvalina iki sveiko skaičiaus, analogiškai su *log2* reikšme, tik vietoj šaknies naudojama logaritmo funkcija, 0.5 atsitiktinai parenka pusę kintamųjų, o *None* - visada renkasi iš visų kintamųjų.
3. *min_samples_split* nurodo minimalų stebinių skaičių, kuris yra reikalingas, kad pirminė viršūnė būtų padalinta. Jei stebinių skaičius viršūnėje yra mažesnis nei šis parametras, viršūnė nebus padalinta ir taps sprendimų medžio lapu. Didinant šį parametą galima išvengti persotinimo, pasirinktos reikšmės [2, 5, 10] + [20].
4. *min_samples_leaf* nurodo minimalų stebinių skaičių, kuris yra reikalingas, kad viršūnė galėtų tapti sprendimų medžio lapo viršūne. Kaip ir *min_samples_split* parametre, didinant šio parametro vertę galima išvengti persotinimo. Pasirinktos reikšmės [1, 2, 4] + [8].
5. *bootstrap* nurodo, ar plėtomio imtys yra naudojamos, kai yra generuojamas sprendimų medis. Kai reikšmė yra True, kiekvienas medis atsitiktiniam miške yra apmokomas su atsitiktine duomenų imtimi, su užkeitimu (ang. bootstrap sampling). Tai reiškia, kad tam tikri stebiniai gali būti pakartojami imtyje, kai kiti stebiniai gali būti nepanaudoti. Kitu atveju, kai reikšmė nustatyta į False, kiekvienas medis bus apmokomas su visu duomenų rinkiniu. Naudojamos reikšmės paieškoje [True, False].

Analizuojant atsitiktinio miško rezultatus 6 lentelėje yra matoma, jog labai išaugo gardelės paieškos įgyvendinimo laikas. Chen and Guestrin (2016) savo darbe yra padaręs palyginimą, kad XGBoost metodas veikia greičiau nei atsitiktinis miškas, tačiau, remiantis šiuo eksperimentu su gardelės paieška, tokių išvadų reikėtų privengti, kadangi miškų modelių apmokymo laikas gali priklausyti nuo tam tikrų hiperparametrų verčių. Palyginus pasirinktas parametro $n_estimator$, reiškiančio medžių skaičių modelyje, vertes XGBoost ir atsitiktinio miško metoduose, yra matoma, kad atsitiktiniame miške yra naudotos ženkliai didesnės reikšmės, kurios daro įtaką modelio apsitreniravimo laikui, kadangi kuo daugiau sprendimų medžių modelis sugeneruoja, tuo labiau išauga treniravimo laikas. Apibendrinant šiuos du rezultatus, pastebima, jog XGBoost metodas rado geresnį sprendimą su neženkliai didesniu tikslumu (0.7620 prieš 0.7602), tačiau skirtumas yra labai menkas ir abu modelius praktiškai pavyksta pagerinti vienodai, lyginant su modeliais po kryžminės validacijos (4 lent.).

6 lentelė. Gardelės paieškos rezultatai su dviem skirtingais kombinacijų skaičiais atsitiktinio miško metodui

Kombinacijų skaičius	162	512
Laikas	15min 17 s	1 h 42 min 25 s
Tikslumas	0.7586	0.7602

Eksperimentavimas su gardelės paieška bus pabaigiamas su paskutiniais dviem metodais, kurie iš pirminių rezultatų atliko ne tokį tikslų prognozavimą - KNN ir logistinė regresija. Kadangi šie metodai yra paprastesni, jų skirtingų hiperparametrų skaičius nėra toks didelis, todėl atlikti įvairesnį eksperimentą atliekant gardelės paiešką nėra tokių didelių galimybių, kaip buvo galima tą atlikti su pastaraisiais medžių metodais. KNN metode buvo atliekama paieška su $2^2 * 5 = 20$ ir $2^2 * 7 = 28$ kombinacijų skaičiumi. Eksperimento KNN metodui hiperparametrų reikšmės ir vertės:

1. $n_neighbors$ nurodo kaimynų skaičių, iš kurių yra renkamasi, darant prognozę (įvardija skaičių K). Pasirinktos reikšmės: [1, 3, 5, 7, 9] + [11, 13].
2. $weights$ nurodo svorį, kuris yra priskiriamas kiekvienam kaimynui, kai yra daroma prognozė. Pasirinktos reikšmės yra 'uniform', kuri reiškia, kad visi kaimynai turi vienodą svorį ir 'distance', su kuria kiekvieno kaimyno svoriai proporcingi atstumui nuo naujo, prognozuojamojo stebinio.
3. p įvardija atstumo metriką, pasirinktos reikšmės yra 1 (Manheteno atstumas) ir 2 (Euklidinis)

Rezultatai matomi 7 lentelėje, vėlgi, nors ir kombinacijų skaičių skirtumas nebuvo žymus, tačiau yra matomas skirtumas, kad tikslumas tarp eksperimentų padidėja labai neženkliai, o

vienai kombinacijai patikrinti KNN metodu truko apie 7 sekundes. Lyginant su 4 gautu tikslumu KNN metodu matome pagerėjimą, tikslumas pakilo per 0.02, kas iki šiol yra geriausias pokytis po gardelės paieškos eksperimento. Galiausiai, liko pristatyti logistinės regresijos tikrintus hiperparametrus:

1. *penalty* - šiame parametre yra nurodoma modeliui taikoma reguliacinė paklaida. Šis reguliavimas padeda išvengti persotinimo, neleidžiant suteikti per didelio svorio vienam iš kintamųjų. Pasirinktos reikšmės:
 - 'l1' įvardija Lasso regularizacija, kuri prideda koeficientų absoliučias reikšmes prie nuostolių funkcijos,
 - 'l2' įvardija Ridge regularizacija, čia prie nuostolių funkcijos yra pridedama kvadratinė koeficiento reikšmė.
2. *C* yra įvardintos regularizacijos stiprumo koeficientas. Mažesnė *C* reikšmė stiprina regularizacijos efektą. Pasirinktos reikšmės $[0.01, 0.1, 1, 10, 100] + [0.001]$.
3. *solver* nurodo optimizavimo algoritmą. Pasirinktos reikšmės yra 'liblinear', šis optimizatorius naudoja koordinačių nusileidimo ir nuoseklių minimalaus optimizavimo algoritmus (SMO), labiau naudojamas mažesniems duomenų rinkiniams, kita reikšmė 'SAGA' atitinka stochastinį vidutinį gradiento nusileidimą, kuris yra stochastinio gradiento nusileidimo plėtinys, labiau tinkamas didesniems duomenų rinkiniams.
4. *max_iter* nustato maksimalų optimizavimo algoritmo iteracijų skaičių, jeigu optimizavimas nesukonverguoja iki nurodyto skaičiaus, jis sustoja ir grąžina paskutinę reikšmę. Pasirinktos reikšmės $[100, 200] + [500, 1000]$.
5. *class_weight* apibrėžia svorius, susietus su tikslo kintamojo klasės žyma. Jis padeda susitvarkyti su nesubalansuotais duomenų rinkiniais, naudotos reikšmės yra 'None', kuri nekreipia dėmesio į balansavimą ir 'balanced' kuri padidina svorius mažumos klasei.

8 lentelėje yra matomi gardelės paieškos rezultatai, kuriuose yra matoma, kad logistinė regresija perbėga per skirtingas hiperparametrų kombinacijas greičiausiai, lyginant su kitais metodais. Tačiau po gardelės paieškos logistinė regresija buvo vienintėlis metodas, kuris nepagerėjo nei po parametrų kombinacijų skaičiaus padidinimo, nei lyginant su rezultatais atliekant kryžminę validaciją be gardelės paieškos - tikslumas išliko toks pat.

7 lentelė. Gardelės paieškos rezultatai su dviem skirtingais kombinacijų skaičiais KNN metodu

Kombinacijų skaičius	20	28
Laikas	2 min 14 s	3 min 13 s
Tikslumas	0.7178	0.7119

8 lentelė. Gardelės paieškos rezultatai su dviem skirtingais kombinacijų skaičiais Logistinės regresijos metodui

Kombinacijų skaičius	80	192
Laikas	1 min 22 s	3 min 4 s
Tikslumas	0.6941	0.6941

Atlikus šiuos eksperimentus pavyko reikšmingai pagerinti modelio tikslumą trim iš keturių modelių. Logistinė regresija vienintėlė nerado geresnio modelio, kuris pagerintų prognozės tikslumą, todėl galima teigti, kad modelis, su parametrais pagal nutylėjimą prognozuoja pakankamai gerai.

Toliau yra pritaikoma atsitiktinė paieška, taikant kryžminę validaciją. Parinkus skirtingų kombinacijų skaičių $n = 50$, pritaikoma atsitiktinė paieška visiems modeliams su tam tikromis parametrų reikšmėmis ar intervalais. Lentelėje 9 yra pateikiami eksperimento rezultatai, iš kurių matosi, kad XGBoost metodas pirmą kartą užleido atsitiktinį mišką į pirmą vietą tikslumo atžvilgiu. Tačiau ši paieška su $n = 50$ pasirodo geriau už gardelės paiešką tik KNN metodui, kadangi tiek XGBoost, tiek atsitiktinio miško metodai su gardelės paieška gavo geresnius rezultatus, tačiau ir laukimo laikas buvo didesnis, kadangi kombinacijų skaičius buvo pasirinktas didesnis. Taip pat reiktų atkreipti dėmesį, kad eksperimento įgyvendinimo laikas miškų metoduose nekoreliuoja kartu su gardelės paieškos metodu. Nors ir yra tikrinamas mažesnis kombinacijų skaičius, įgyvendinimo laikas yra ilgesnis. Tai atsitiko dėl panašios priežasties, kodėl labai skyrėsi XGBoost ir atsitiktinio miško treniravimo laikas gardelės paieškoje, t.y. parinkti hiperparametrai, ilginantys aptreniravimo laiką, buvo pasirenkami dažniau. Tarkim, gardelės paieškoje $n_estimator$, nurodantis medžių skaičių XGBoost metodui, buvo parinktos reikšmės $N_{XGBgrid} = [100, 200]$. Vadinasi, gardelės paieškoje šio hiperparametro reikšmių vidurkis yra $\mu_{N_{XGBgrid}} = 150$, kai atsitiktinėje paieškoje buvo renkamasi iš aibės $N_{XGBrand} = [100, 200, \dots, 900]$. Todėl, natūralu, kad atsitiktinai renkantis iš platesnios intervalo, šio parametro vidurkis atsitiktinėje paieškoje buvo $\mu_{N_{XGBrand}} = 520$. Tai yra tik vienas iš faktorių, galėjusių padidinti paieškos greitį. Norint atlikti tolimesnę analizę, reiktų paanalizuoti kitus hiperparametrus, kurie gali prailginti modelio apmokymo laiką. Pavyzdžiui, medžio gylio parametras max_depth , ar $subsample$, $colsample_bytree$ – didinant šiuos parametrus didės treniravimo laikas, bei $gamma$ hiperparametras su mažesne reikšme prailgins mokymo procesą. Atsitiktiniuose miškuose atitinkamai reikėtų palyginti $n_estimators$, max_depth , $max_features$ parametrus, juos didinant taip pat turėtų didėti mokymo laikas, o mažinant $min_samples_split$ ir $min_samples_leaf$ reikšmes, mokymo laikas taip pat ilgės.

9 lentelė. Tikslumo metrikos ir laikai su atsitiktine paieška, taikant kryžminę validaciją, kai $n = 50$

Metodas	KNN	Log. Regresija	Atsitiktinis miškas	XGBoost
Laikas	5 min 14 s	49 s	15 min 42 s	12 min 4 s
Tikslumas	0.72	0.694	0.759	0.751

Taigi, apibendrinant šį skyrių galima pasiimti tris hiperparametrų kombinacijas KNN, XGBoost ir atsitiktinio miško modeliams siekiant išgauti tikslesnį prognozavimą mūsų klientų prognozės modeliuose:

1. KNN po atsitiktinės paieškos parametrai:

- *weights* = 'uniform',
- *p* = 1,
- *n_neighbours* = 19.

2. XGBoost geriausi hiperparametrai po gardelės paieškos:

- *colsample_bytree* = 0.8,
- *gamma* = 0,
- *learning_rate* = 0.1,
- *max_depth* = 5,
- *min_child_weight* = 1,
- *n_estimators* = 100,
- *scale_pos_weight* = 1,
- *subsample* = 1.

3. Atsitiktinio miško geriausi parametrai po gardelės paieškos:

- *bootstrap* = True,
- *max_features* = 'sqrt',
- *min_samples_leaf* = 8,
- *min_samples_split* = 2,
- *n_estimators* = 100.

Reiktų pastebėti, kad išvardinti parametrai yra tie, iš kurių buvo rinktasi atsitiktinėje ir gardelės paieškoje, likę modelių parametrai, kurie nėra įvardinti yra palikti su reikšmėmis pagal nutylėjimą. Kadangi logistinė regresija nerado hiperparametrų kombinacijos, su kuria

būtų pagerėjus prognozės kokybė, yra paliekama išvada, kad modelio hiperparametrų reikšmės pagal nutylėjimas prognozuoja taip pat teisingai. Galiausiai, po hiperparametrų paieškos, tiksliausiai prognozuojanti modelis išlieka XGBoost, kadangi su šiuo modeliu buvo išgautas kol kas didžiausias tikslumas su reikšme 0.762.

3.9. Eksperimentas taikant klasių balansavimo metodus.

3.1.5 skyrelyje jau buvo paminėta tikslo kintamojo disbalanso problema ir kad pirminis eksperimentas buvo atliktas sumažinus daugumos klasės stebinius. Šiame skyriuje bus atliekamas eksperimentas, kaip skirsis modelių kokybės, kai tikslo kintamojo reikšmės bus subalansuotos, didinant trūkstamos klasės imtį arba mažinant perteklinės klasės imtį, bei nieko nekeičiant su duomenis, t.y. dirbant su išbalansuotu duomenų rinkiniu. Pirminis eksperimentas buvo atliktas mažinant duomenų klasės daugumos imtį atsitiktiniu būdu ir mokymo-validavimo duomenų rinkiniai buvo paskirstyti po duomenų balansavimo, dėl lengvesnio modelio metrikų interpretavimo, taikant kitus duomenų balansavimo metodus ir toliau bus tęsiama su šia praktika. Prognozei bus taikomas XGBoost modelis su parametrais, kuriuos pavyko rasti 3.8 skyrelyje naudojant gardelės paiešką ir nustatytas didžiausias tikslumas. Duomenų balansavimas buvo atliktas taikant funkcijas iš *imbalanced-learn* bibliotekos naudojant funkcijas **SMOTE**, **NearMiss**, **RandomOverSampler** (atsitiktinis trūkstamos klasės imties didinimas) ir **RandomUnderSampler** (atsitiktinis perteklinės klasės mažinimas) (Lemaître et al., 2017). 10 lentelėje yra pateikiami eksperimento rezultatai, iš kurių matosi, kad atsitiktinio trūkstamos klasės imties didinimo ir mažinimo metodai pasiekia panašius rezultatus, tik didinimo metodas turi šiek tiek geresnes metrikų vertes, tačiau abu metodai atsilieka nuo kitų dviejų imčių balansavimo metodų. NearMiss imties mažinimo metodas pralenkia atsitiktinius metodus, tačiau atsilieka nuo SMOTE imties didinimo metodo, kuris pasiekia geriausią tikslumą.

10 lentelė. XGBoost modelio tikslumo metrikos taikant klasių balansavimo metodus su balansuotu validavimo rinkiniu

Metodas	Preciziškumas	Jautrumas	F1 statistika	Tikslumas	AUC
NearMiss	0.853	0.763	0.805	0.816	0.816
Atsitiktinis maž.	0.777	0.738	0.757	0.764	0.764
Atsitiktinis did.	0.787	0.747	0.766	0.773	0.773
SMOTE	0.897	0.854	0.875	0.878	0.878

Taigi, atrodo, kad SMOTE metodas suteikia geriausią tikslumą prognozei visoms metrikoms. Kyla natūralus klausimas - ar galima drąsiai teigti, kad šis modelis yra geriausias, ir jį reikėtų taikyti praktikoje, siekiant išvengti klientų atsisakymo? Ne visai, kadangi validacija buvo atliekama su subalansuotais duomenimis ir kaip ir buvo minėta 3.1.5 skyriuje, buvo

prisiimta rizika dėl galimų optimistiškesnių rezultatų ir neatitikimo su realios problemos scenarijumi. Pabandykime padaryti šį balansavimą tik mokymo duomenims ir atliekant prognozę su tuo pačiu XGBoost modeliu atlikti validaciją su išbalansuotu duomenų rinkiniu. Čia bus pridėdami du papildomi būdai - kadangi XGBoost metodas turi parametą *scale_pos_weight*, kuris padidina svorį mažumos klasei, taip tvarkantis su duomenų disbalansu, nustatykime šio parametro vertę į 14 (klasių disbalanso pasiskirstymo santykis apytiksliai siekia 1:14). Kitas atvejis bus pritaikyti XGBoost metodas, netaikant jokio klasių balansavimo. 11 lent. yra pateikiami rezultatai, kurie neatrodo labai džiuginantys, pabandykime juos paanalizuoti:

- XGBoost nenaudojant jokio balansavimo - preciziškumas neatrodo tragiškai, palyginus su kitomis metrikomis, tai reiškia, kad kai yra prognozuojamas kliento pasitraukimas, dažniau yra suprognozuojama teisingai, tačiau jautrumas yra labai žemas, o tai nurodo, kad labai maža dalis tikrai pasitraukusių klientų buvo suprognozuoti. Tą nurodo ir didelis tikslumas, įvardijantis, kad modelis buvo linkęs prognozuoti kliento nepasitraukimą, o šios klasės skaičius buvo daugumoje. AUC balą galima interpretuoti, kaip sugebėjimą atskirti dvi klases, čia šis balas yra pakankamai žemas, kas patvirtina prieš tai nusakytą išvadą, kad modelis yra linkęs prognozuoti į vieną pusę.
- XGBoost su svorio parametru, suteikiančiu klasės balansą, bei atsitiktiniai imties didinimo ir mažinimo metodai gauna panašius rezultatus - čia, atvirkščiai nei XGBoost su originaliais duomenimis. Preciziškumas yra labai mažas, tai reiškia, kad modelis apytiksliai neteisingai suprognozuoja 4 iš 5 pasitraukusių klientų, tačiau didesnis jautrumas parodo, kad paliekama mažesnė dalis nesuprognozuotų pasitraukusių klientų (randama apie 70 % pabėgusių klientų). Tikslumas atrodo neblogai - apie 80 %, tačiau vėl reiktų nepamiršti, kad duomenų rinkinys buvo išbalansuotas. AUC metrika parodo, kad modelis, palyginus su kitais geriausiai atskiria teigiamas ir neigiamas klases.
- SMOTE metodo nors ir F1 statistika atrodo neblogai, tačiau čia įtaką daro mažiausias skirtumas tarp preciziškumo ir jautrumo, kurių reikšmės bendrai nėra pačios geriausios. Situacija yra panaši, kaip ir su atsitiktiniais metodais - modelis suprognozuoja 3 iš 4 kartų neteisingai, o Jautrumas nurodo, kad tik 52 % pasitraukusių klientų buvo suprognozuoti teisingai. Į tikslumą kreipti didelio dėmesio nereiktų, dėl jau paminėtos išbalansavimo priežasties, o AUC balas atrodo šiek tiek blogesnis nei atsitiktinių ir XGBoost su svoriu metodų.
- NearMiss metodas bendrai turi blogiausias metrikas - Preciziškumas parodo, kad teisingai apytiksliai teisingai suprognozuojamas stebiny pasitaiko 1 kartą iš 20, nors ir suprognozuojama didžiausia dalis pabėgusių klientų (76 %), modelis akivaizdžiai yra labai linkęs prognozuoti į vieną pusę, t.y., kad klientas pabėgs. Tą nurodo ir labai mažas tikslumas, nes validavimo rinkinyje didžioji dalis yra neigiamų žymų (reiškiančių, kad klientas nepabėgs).

11 lentelė. XGBoost modelio tikslumo metrikos taikant klasių balansavimo metodus su nebalansuotu validavimo rinkiniu

Metodas	Preciziškumas	Jautrumas	F1 statistika	Tikslumas	AUC
XGBoost	0.707	0.144	0.240	0.944	0.570
XGBoost14	0.201	0.716	0.314	0.807	0.764
SMOTE	0.267	0.522	0.353	0.882	0.714
Atsitiktinis did.	0.196	0.726	0.309	0.800	0.765
Atsitiktinis maž.	0.188	0.738	0.300	0.787	0.764
NearMiss	0.056	0.766	0.105	0.196	0.462

Iš šio metrikų apibendrinimo sunku išrinkti geriausią modelį, kadangi visi jie turi aiškius trūkumus. Šioje vietoje tyrėjai turi nuspręsti, kas yra skausmingiau - ar teisingai suprognozuoti daugiau klientų dažniau pateikiant klaidingą prognozę ar teisingai suprognozuoti mažiau klientų, rečiau pateikiant klaidingą prognozę. Tačiau to išvengiant, reikėtų toliau eksperimentuoti su skirtingai metodais, modeliais, kintamaisiais siekiant geresnių metrikų.

Išvados

Telekomunikacijų industrijoje klientų išlaikymas yra labai svarbus, siekiant išlaikyti pajamas, rinkos dalį ir klientų pasitenkinimą. Šis tyrimas buvo motyvuotas, siekiant suprasti, kurie veiksniai skatina klientų sumažėjimą, bei kurie metodai yra veiksmingiausi šiam uždaviniui spręsti. Taip pat, pritaikius skirtingas mašininio mokymosi technologijas ir analizuojant skirtingų modelių veikimą, buvo gauta vertingų įžvalgų į šios problemos sprendimo metodologiją.

Atliekant literatūros apžvalgą, buvo pastebėta, kad daug šaltinių, sprendžiančių panašią problemą, renka medžiais grįstus mašininio mokymosi modelius, kurie pasirodo geriausiai, todėl šiame darbe taikyti metodai buvo pasirinkti XGBoost ir atsitiktinis miškas, o jų palyginimui - pasirinkti klasikiniai klasifikavimo uždavinį atliekantys metodai - logistinė regresija ir k -artimiausių kaimynų metodas.

Prieš atliekant modeliavimą, buvo atliktas duomenų valymas - taikant interkvartilių metodą buvo ištrintos išskirtys iš duomenų, o atliekant kolinearumo analizę atsisakoma panašių kintamųjų. Atlikus grafinę duomenų analizę buvo padarytos pirmosios prielaidos, apie faktorius, kurie galėjo lemti klientų tam tikrą elgesį, o šių prielaidų patikrinimas buvo atliktas po modeliavimo, tiriant kintamųjų reikšmingumą.

Lyginant skirtingų modelių tikslumo metrikas, dažniausiu atveju gaudavome labai panašų rezultatą XGBoost ir atsitiktinio miško metoduose, nors XGBoost metodas dažniausiai šiek tiek pralenkdavo atsitiktinį mišką, logistinė regresija ir KNN tarpusavyje turėjo panašius rezultatus, kurie buvo prastesni nei medžiais grįstų metodų.

Taip pat buvo susidurta su klasių disbalanso problema - pirminis eksperimentas buvo atliktas išbalansavus duomenis atsitiktiniu būdu, mažinant daugumos imtį, vėliau, darbo pabaigoje buvo pateikta analizė, atliekant kitus metodus. Teoriškai, geriausiai pasirodė SMOTE metodas, didinantis trūkstamos klasės imtį, tačiau toliau eksperimentuojant, buvo susidurta su priežastimi, kodėl drąsiai nereiktų šio metodo pripažinti kaip geriausio, kadangi atliekant modelio tikrinimą su išbalansuotais duomenimis rezultatai buvo kitokie. Pagrindinės išvados:

- Naudojamumo ir laiko, iki kontrakto pabaigos kintamieji buvo reikšmingiausi klientų atsisakymo prognozavime, kas buvo ir nuspėjama iš duomenų analizės.
- XGBoost metodas eksperimentuose turėjo geriausią tikslumą, labai arti jo buvo atsitiktinis miškas.
- Klasių disbalanso problema gali labai nusverti prognozes į vieną pusę, kas sukelia daug sunkumų, sprendžiant tokio tipo uždavinius. Teoriškai, SMOTE metodas turėjo geriausius rodiklius.

- Gerinant rezultatus, reikėtų daugiau paanalizuoti klasių balansavimo metodikas, taip pat praplėsti duomenis, ieškant daugiau reikšmingų kintamųjų, paeksperimentuoti su kitokiais kintamųjų pertvarkiais.

Literatūros sąrašas

- Ahmad, A. K., Jafar, A. & Aljoumaa, K. (2019). Customer churn prediction in telecom using machine learning in big data platform. *Journal of Big Data*, 6(1), 1–24.
- Alaliyat, S. (2023). Video -based fall detection in elderly’s houses.
- Bishop, C. M. & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*. Springer.
- Boehmke, B. & Greenwell, B. M. (2019). *Hands-on machine learning with r*. CRC press.
- Bogaert, M. & Delaere, L. (2023). Ensemble methods in customer churn prediction: A comparative analysis of the state-of-the-art. *Mathematics*, 11(5), 1137.
- Breiman, L. (2001). *Random forests*. doi:[10.1023/A:1010933404324](https://doi.org/10.1023/A:1010933404324)
- Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16, 321–357.
- Chen, T. & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, San Francisco, California, USA: ACM.
- Cohen, J., Cohen, P., West, S. G. & Aiken, L. S. (1983). *Applied multiple regression/correlation analysis for the behavioral sciences*. Routledge.
- Eurostat. (2022). Eurostat: Digital economy and society statistics - households and individuals. Prieita gegužės 9d., 2023.
- Freund, Y. & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1), 119–139.
- Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of statistics*, 1189–1232.
- Géron, A. (2022). *Hands-on machine learning with scikit-learn, keras, and tensorflow*. "O'Reilly Media, Inc."
- Gupta, S. & Zeithaml, V. (2006). Customer metrics and their impact on financial performance. *Marketing science*, 25(6), 718–739.
- Hastie, T., Tibshirani, R., Friedman, J. H. & Friedman, J. H. (2009). *The elements of statistical learning: Data mining, inference, and prediction*. Springer.
- Hauge, J. & Jamison, M. (2009). Analyzing telecommunications market competition: Foundations for best practices. *Public Utility Research Center, University of Florida*, 29.
- He, H. & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9), 1263–1284.
- Hosmer Jr, D. W., Lemeshow, S. & Sturdivant, R. X. (2013). *Applied logistic regression*. John Wiley & Sons.
- IBM. (2023). What is random forest? Retrieved from <https://www.ibm.com/topics/random-forest>

- Lemaître, G., Nogueira, F. & Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*.
- López Oliva, V., Egea, R., Museros, L. & Sanz, I. (2021). A customer churn detection model for the pay-tv sector.
- Malato, G. (2021). Hyperparameter tuning. grid search and random search your data teacher. <https://www.yourdatateacher.com/2021/05/19/hyperparameter-tuning-grid-search-and-random-search/>. Prieita gegužės 9d., 2023.
- Orellana, E. (2020). Smote. Retrieved from <https://emilia-orellana44.medium.com/smote-2acd5dd09948>
- Ozdemir, S. (2016). *Principles of data science*. Packt Publishing Ltd.
- Patro, R. (2021). Cross validation: K fold vs monte carlo. Retrieved from <https://towardsdatascience.com/cross-validation-k-fold-vs-monte-carlo-e54df2fc179b>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ..., Dubourg, V. et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12, 2825–2830.
- Powers, D. M. (2020). Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation.
- Quinlan, J. R. (1986). *Induction of decision trees*. San Francisco, CA: Morgan Kaufmann.
- Vafeiadis, T., Diamantaras, K. I., Sarigiannidis, G. & Chatzisavvas, K. C. (2015). A comparison of machine learning techniques for customer churn prediction. *Simulation Modelling Practice and Theory*, 55, 1–9.
- Zhou, Z.-H. (2012). *Ensemble methods: Foundations and algorithms*. doi:[10.1201/b11864](https://doi.org/10.1201/b11864)

Priedai

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
df = pd.read_csv('df_for_uni.csv')

iqr_cols = ['amzius', 'produkto_ilgaamz', 'kliento_ilgaamz', 'kontrakto_likutis',
            'gedimu_sk', 'naudojimas_savaitgali', 'naudojimas_siokiadieni', 'mokestis']
df_num = df[iqr_cols]
Q1 = df_num.quantile(0.25)
Q3 = df_num.quantile(0.75)
IQR = Q3-Q1
df = df[~((df_num < (Q1 - 2 * IQR)) | (df_num > (Q3 + 2 * IQR))).any(axis = 1)
        ]

scaler = MinMaxScaler()
df[['amzius', 'produkto_ilgaamz', 'kliento_ilgaamz', 'kontrakto_likutis',
    'gedimu_sk',
    'naudojimas_savaitgali', 'naudojimas_siokiadieni', 'mokestis']] = scaler.
    fit_transform(df[['amzius', 'produkto_ilgaamz', 'kliento_ilgaamz',
    'kontrakto_likutis', 'gedimu_sk', 'naudojimas_savaitgali',
    'naudojimas_siokiadieni', 'mokestis']])
```

Kodas 1. Duomenų nuskaitymas, normalizavimas ir išskirčių šalinimas IQR metodu

```
import seaborn as sns
from scipy.stats import chi2_contingency
obj_cols = ['turi_paslauga_1', 'turi_paslauga_2', 'turi_paslauga_3', '
            turi_paslauga_4',
            'turi_iranga', 'aktyvus_kontraktas', 'didmiestis', 'butas', '
            planas_1', 'planas_2', 'planas_3']
chi2_results = pd.DataFrame(index=obj_cols, columns=obj_cols)
for i in range(len(obj_cols)):
    for j in range(i, len(obj_cols)):
        contingency_table = pd.crosstab(df[obj_cols[i]], df[obj_cols[j]])
        chi2, pval, dof, expected = chi2_contingency(contingency_table)
        chi2_results.loc[obj_cols[i], obj_cols[j]] = pval
        chi2_results.loc[obj_cols[j], obj_cols[i]] = pval
chi2_results = chi2_results.apply(pd.to_numeric)
```

```

sns.heatmap(chi2_results, annot=False, cmap='coolwarm')

corr_matrix = df.corr().round(2)
sns.heatmap(corr_matrix, cmap='coolwarm', annot=True)
plt.show()

```

Kodas 2. χ^2 ir koreliacinė analizė

```

import numpy as np
final_cols = ['turi_paslauga_2', 'aktyvus_kontraktas', 'turi_iranga', '
    didmiestis', 'planas_3', 'amzius', 'produkto_ilgaamz', 'kontrakto_likutis'
    , 'gedimu_sk',
    'naudojimas_siokiadieni', 'mokestis', 'atsisakymas']
from imblearn.under_sampling import RandomUnderSampler
convert_dict = {'turi_paslauga_2': int,
    'aktyvus_kontraktas': int,
    'turi_iranga': int,
    'didmiestis': int,
    'planas_3': int
    }

df = df.astype(convert_dict)
X = df[final_cols].drop('atsisakymas', axis = 1)
y = df['atsisakymas']
y = y.values.astype(np.int32)
rus = RandomUnderSampler(random_state=42)
X_rus, y_rus = rus.fit_resample(X, y)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_rus, y_rus, test_size =
    0.2, random_state = 42)

```

Kodas 3. Duomenų imties mažinimas atsitiktiniu metodu

```

# KNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)
y_pred_knn_prob = knn.predict_proba(X_test)

```

```

# Log. reg
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred_logreg = logreg.predict(X_test)
y_pred_logreg_prob = logreg.predict_proba(X_test)

# RF
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
y_pred_rf_prob = rf.predict_proba (X_test)

# XGB
import xgboost as xgb
xgb_model = xgb.XGBClassifier()
xgb_model.fit(X_train, y_train)
y_pred_xgb = xgb_model.predict(X_test)
y_pred_xgb_prob = xgb_model.predict_proba(X_test)

```

Kodas 4. Pirminis modeliavimas

```

importance = abs(logreg.coef_[0])
importance /= importance.sum()
results_log = pd.DataFrame({'feature': X_train.columns,
                           'log_reg': importance})

importance_scores = xgb_model.get_booster().get_score(importance_type='
gain')
total_score = sum(importance_scores.values())
normalized_scores = {feature: score / total_score for feature, score in
                      importance_scores.items()}
sorted_scores = sorted(normalized_scores.items(), key=lambda x: x[1],
                        reverse=True)
xgb_gain_scores = pd.DataFrame(sorted_scores, columns=['feature', '
xgb_gain'])

importance_scores = xgb_model.get_booster().get_score(importance_type='

```

```

        cover')
total_score = sum(importance_scores.values())
normalized_scores = {feature: score / total_score for feature, score in
    importance_scores.items()}
sorted_scores = sorted(normalized_scores.items(), key=lambda x: x[1],
    reverse=True)
xgb_cover_scores = pd.DataFrame(sorted_scores, columns=['feature', '
    xgb_cover'])

importance = rf.feature_importances_
importance /= importance.sum()
rf_mdi = pd.DataFrame({'feature': X.columns, 'rf_mdi': importance})

```

Kodas 5. Kintamųjų reikšmingumas

```

import time
from sklearn.model_selection import GridSearchCV

xgb_params = {
    'learning_rate': [0.1, 0.3],
    'max_depth': [3, 5],
    'min_child_weight': [1, 3],
    'gamma': [0.1, 0.3],
    'subsample': [0.6, 0.8],
    'colsample_bytree': [0.6, 0.8],
    'n_estimators': [100, 200],
    'scale_pos_weight': [1, 2]
}

xgb_model = XGBClassifier()
param_grid = [xgb_params]
models = {'xgboost': xgb_model}

for name, model in models.items():
    print(f"Running Grid Search with CV for {name}")
    start_time = time.time()
    grid_search = GridSearchCV(model, param_grid=param_grid, cv=5, scoring
        = 'accuracy', n_jobs=-1)
    grid_search.fit(X_rus, y_rus)
    end_time = time.time()
    print(f"Best Parameters: {grid_search.best_params_}")

```

```
print(f"Best Score: {grid_search.best_score_:.4f}")
print(f"Time taken: {np.round(end_time-start_time, 2)} seconds\n")
```

Kodas 6. Gardelės paieškos pvz. XGBoost metodui

```
from sklearn.model_selection import RandomizedSearchCV

xgb_params = {
    'learning_rate': [0.1, 0.3],
    'max_depth': range(3, 10),
    'min_child_weight': range(1, 6),
    'gamma': [0, 0.1, 0.2, 0.3],
    'subsample': [0.6, 0.7, 0.8, 0.9],
    'colsample_bytree': [0.6, 0.7, 0.8, 0.9],
    'n_estimators': range(100, 1000, 100),
    'scale_pos_weight': [1, 2, 3]
}

start_time = time.time()
xgb_random = RandomizedSearchCV(XGBClassifier(), xgb_params, n_iter=10,
    random_state=1111, cv=5, scoring='accuracy')
xgb_random.fit(X_rus, y_rus)
xgb_search_time = time.time() - start_time
best_xgb_params = xgb_random.best_params_
print("Best XGBoost parameters:", best_xgb_params)
```

Kodas 7. Atsitiktinės paieškos pvz. XGBoost metodui

```
from sklearn.metrics import precision_score, recall_score, f1_score,
    accuracy_score, roc_auc_score
from imblearn.over_sampling import SMOTE, RandomOverSampler
from imblearn.under_sampling import NearMiss

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)
resampling_techniques = {
    'Original': (X_train, y_train),
    'SMOTE': SMOTE(random_state=42).fit_resample(X_train, y_train),
    'Random OverSampling': RandomOverSampler(random_state=42).fit_resample(
        X_train, y_train),
```

```

    'Random UnderSampling': RandomUnderSampler(random_state=42).fit_resample(
        X_train, y_train),
    'NearMiss': NearMiss().fit_resample(X_train, y_train),
}
results_df = pd.DataFrame(columns=['Subset', 'Precision', 'Recall', 'F1-Score',
    'Accuracy', 'AUC Score'])
for subset, (X_resampled, y_resampled) in resampling_techniques.items():
    model = xgb.XGBClassifier(
        colsample_bytree=0.8,
        gamma=0,
        learning_rate=0.1,
        max_depth=5,
        min_child_weight=1,
        n_estimators=100,
        scale_pos_weight=1,
        subsample=1,
        random_state=42
    )

    model.fit(X_resampled, y_resampled)

    y_pred = model.predict(X_test)

    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    accuracy = accuracy_score(y_test, y_pred)
    auc = roc_auc_score(y_test, y_pred)

    results_df = results_df.append({
        'Subset': subset,
        'Precision': precision,
        'Recall': recall,
        'F1-Score': f1,
        'Accuracy': accuracy,
        'AUC Score': auc
    }, ignore_index=True)
print(results_df)

```

Kodas 8. Imties balansavimo metodų palyginimas