Kauno technologijos universitetas

Matematikos ir gamtos mokslų fakultetas

Požymių inžinerijos automatizavimas mašininiam mokymuisi iš verslo duomenų bazių

Baigiamasis magistro studijų projektas

Aušrinė Zinkutė

Projekto autorė

Prof. Dr. Evaldas Vaičiukynas

Vadovas

Doc. Dr. Jurga Duobienė

Vadovė

Kaunas, 2023

Kauno technologijos universitetas

Matematikos ir gamtos mokslų fakultetas

Požymių inžinerijos automatizavimas mašininiam mokymuisi iš verslo duomenų bazių

Baigiamasis magistro studijų projektas

Didžiųjų verslo duomenų analitika (6213AX001)

Aušrinė Zinkutė

Projekto autorė

Prof. Dr. Evaldas Vaičiukynas

Vadovas

Doc. Dr. Jurga Duobienė

Vadovė

Dr. Paulius Danėnas

Recenzentas

Prof. Dr. Mantas Vilkas

Recenzentas

Kaunas, 2023

Kauno technologijos universitetas

Matematikos ir gamtos mokslų fakultetas

Aušrinė Zinkutė

Požymių inžinerijos automatizavimas mašininiam mokymuisi iš verslo duomenų bazių

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs (-usi);

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Aušrinė Zinkutė

Patvirtinta elektroniniu būdu

Zinkutė, Aušrinė. Požymių inžinerijos automatizavimas mašininiam mokymuisi iš verslo duomenų bazių. Magistro studijų baigiamasis projektas / vadovai Prof. Dr. Evaldas Vaičiukynas ir Doc. Dr. Jurga Duobienė; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Taikomoji matematika (Matematikos mokslai).

Reikšminiai žodžiai: verslo procesų automatizavimas, reliacinė duomenų bazė, mašininis mokymasis, požymių inžinerija, lentelių plokštinimas.

Kaunas, 2023. 66 p.

Santrauka

Tobulėjant informacinėms technologijoms ir didėjant verslo kuriamų ir saugomų duomenų kiekiui, pastebimas įmonių skaitmenizavimo ir procesų automatizavimo reikalingumas. Viena iš svarbiausių ir labiausiai verslo procesams įtaką darančių informacinių technologijų sistemų versle – duomenų bazė. Tinkamos duomenų bazės parinkimas gali turėti įtaką vėlesniems verslo sprendimams saugoti, analizuoti ir tvarkyti turimus duomenis. Vienos iš populiariausių duomenų bazių tipų versle – reliacinės duomenų bazės – įmonių veikloje dažniausiai pasirenkamos dėl struktūrizuotos duomenų schemos, stabilumo ir patogumo kasdienėms operacijoms vykdyti. Norint iš turimų duomenų išgauti verslui naudingą informaciją ir priimti sprendimus, reliacinėse duomenų bazėse saugomą informaciją galima panaudoti mašininio mokymosi modeliams, tačiau čia tenka atlikti papildomus pertvarkymus – mašininio mokymosi modeliams naudojami duomenys pateikiami lentelės forma, todėl požymių inžinerijos etape duomenys turi būti pertvarkomi iš reliacinės struktūros ir paruošiami modeliui apmokinti.

Šiame projekte atliekama požymių inžinerijos automatizavimo galimybių analizė reliacinėse duomenų bazėse. Siekiama išnagrinėti sistemas bei algoritmus, kurių pagalba duomenų paruošimas modeliui apmokinti gali būti automatizuojamas. Nagrinėjamas Python paketas Featuretools, kuris požymių kūrimui naudoja DFS algoritmą, bei paketas getML, kuriame yra galimybė požymių inžineriją automatizuoti algoritmais FastProp, Relboost, Multirel bei RelMT. Analizė atliekama nagrinėjant 6 reliacines duomenų bazes. Papildomi požymiai automatiškai sukuriami iš turimų duomenų bazių lentelių ir gauta požymių lentelė naudojama apmokinti XGBoost modelį.

Gauti modelių įvertinimų rezultatai rodo, jog automatizavus požymių inžineriją gauti modelio tikslumo rezultatai yra geri tiek klasifikacijos, tiek regresijos uždaviniams spręsti. Nagrinėti algoritmai matematiniais metodais gali sukurti nuo 17 iki 238 papildomų požymių per keletą minučių. Lyginant nagrinėtų duomenų bazių modelių tikslumą, geriausi rezultatai gauti su getML sistemos algoritmu FastProp.

Zinkutė, Aušrinė. Automating Feature Engineering for Machine Learning from Business Databases. Master's Final Degree Project / supervisors Prof. Dr. Evaldas Vaičiukynas and Assoc. Dr. Jurga Duobienė; Faculty of Mathematics and Natural Sciences, Kaunas University of Technology.

Study field and area (study field group): Applied Mathematics (Mathematical Sciences).

Keywords: business process automation, relational database, machine learning, feature engineering, propositionalisation.

Kaunas, 2023. 66 p.

Summary

With the development of information technologies and the amount of data created and stored by business, the need for digitization of companies and automation of processes is noticeable. One of the most important and most influential information technology systems in business is the database. Choosing the right database can have an impact on subsequent business decisions to store, analyze and manage the available data. One of the most popular types of databases in business – relational databases - are usually chosen in the activities of companies due to their structured data schema, stability and convenience for daily operations. In order to extract business-useful information from available data and make decisions, the information stored in relational databases can be used for machine learning models, however, additional transformations have to be done. Data used for machine learning models is usually presented in the form of a table, therefore relational structure must be transformed and prepared during the feature engineering stage to train the model.

This project analyzes the possibilities of feature engineering automation in relational databases. The aim is to examine systems and algorithms that can be used to automate data preparation for model training. Python package Featuretools, which uses the DFS algorithm for feature creation, and getML system, which has the possibility of automating feature engineering with FastProp, Relboost, Multirel and RelMT algorithms, are examined. The analysis is performed by examining 6 relational databases. Additional features are automatically generated from available database tables and the resulting feature table is used to train the XGBoost model.

The obtained model evaluation results show that the model accuracy results obtained after automating feature engineering are considered good for both classification and regression tasks. The considered algorithms can create from 17 to 238 additional features in a few minutes using mathematical methods. Comparing the accuracy of the analyzed database models, the best results were obtained with the FastProp algorithm of getML system.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

Santrumpos:

AUC – plotas po ROC kreive (angl. *Area Under Curve*);

CART – angl. *Classification And Regression Tree* algoritmas;

DBVS – duomenų bazių valdymo sistema (angl. *Database Management System, DBMS*)

DFS – angl. *Deep Feature Sythesis* algoritmas;

DI – dirbtinis intelektas;

MAE – vidutinė absoliutinė paklaida (angl. *Mean Absolute Error*);

RMSE – šaknis iš vidutinės kvadratinės paklaidos (angl. *Root Mean Squared Error*);

RPA – robotinių procesų automatizavimas;

SMOTE – mažumos įrašų sintetinis kūrimas (angl. *Synthetic Minority Over-sampling Technique*);

SQL – struktūrizuota užklausų kalba (angl. *structured query language*);

XGBoost – angl. *eXtreme Gradient Boosting* modelis.

Įvadas

Temos aktualumas. Augant duomenų kiekiui, didėja ir jų panaudojimo galimybės. Turimus duomenis keičiant informacija galima priimti verslui naudingus sprendimus bei siekti strateginių tikslų – didinti pardavimų skaičių, gerinti vartotojo patirtį, teikiamų paslaugų kokybę. Šiandieniniam verslui svarbu ne tik vykdyti kasdienę veiklą, bet ir tobulinti bei atnaujinti vykdomus procesus, siekiant palaikyti konkurencingumą rinkoje. Vis dėl to sudėtinga bei brangu ilgą laiką egzistuojančius procesus pakeisti automatizuotais, pritaikytais didiesiems duomenims tvarkyti: skaitmeninei transformacijai įmonė turi būti subrendusi ir pasiruošusi. Rutininius procesus, atliekamus žmogaus, pakeisti analizės automatizavimu keblu ne tik dėl techninių sprendimų pritaikymo, bet ir dėl pasiruošimo keisti ir atnaujinti metodus, kuriais remiamasi atliekant organizacijai svarbius tiek strateginius, tiek operatyvinius planus. Taip pat susiduriama ir su įvairiomis techninėmis kliūtimis: modelių analizė bei sudarymas gali tapti ilgais ir brangiais procesais, kuriems reikia aukšto lygio techninių žinių išmanymo bei aukštos kvalifikacijos žmogiškųjų išteklių. Kartu su analizuojamų duomenų kiekiu auga ir reikalavimai techninei aplinkai. Duomenis laikant vienoje iš populiariausių duomenų bazių sistemų versle – reliacinėje duomenų bazėje – projektui reikalingi įrašai gali būti laikomi skirtingose lentelėse, todėl duomenų paruošimas tampa laikui imliu ir brangiu procesu mašininio mokymosi uždaviniuose. Net ir naudojant naujausias technologijas bei turint kvalifikuotą personalą, duomenų tyryba yra ilgas procesas, todėl aktualu analizuoti kaip, panaudojant technologijas, galima automatizuoti atliekamus procesus ir apjungti techninius sprendimus taip, kad šiems reiktų kuo mažiau žmogiškųjų išteklių. Šiame darbe nagrinėjamas požymių kūrimo proceso automatizavimas reliacinėse duomenų bazėse padeda požymius, saugomus reliaciniais ryšiais apjungtose lentelėse, paruošti mašininio mokymosi uždaviniui.

Tyrimo objektas. Verslo procesų skaitmenizavimo technikos, skirtos automatizuoti požymių inžineriją.

Tyrimo tikslas. Automatizuotos požymių inžinerijos metodų, skirtų reliacinėms duomenų bazėms verslo ir finansų srityse, tyrimas ir panaudojimo rekomendacijos.

Tyrimo uždaviniai:

1. išnagrinėti verslo procesų skaitmenizavimo iššūkius;
2. atlikti plokštinimo metodų mokslinės literatūros analizę;
3. pasirinkti plokštinimo metodus ir reliacinių duomenų rinkinius iš verslo srities;
4. pritaikyti pasirinktus metodus ir mašininį mokymąsi realiose duomenų bazėse;
5. įvertinti ir aprašyti gautus tyrimo rezultatus;
6. pateikti rekomendacijas palyginant rezultatus tarp skirtingų duomenų bazių.

1. Literatūros apžvalga

Šioje darbo dalyje atliekama mokslinių šaltinių analizė. Pirmiausia atsižvelgiama į vadybinę problemos sprendimo paiešką, siekiant išnagrinėti organizacijos pasirengimą procesų automatizavimui bei su šia skaitmenine transformacija kylančiais iššūkiais. Kita literatūros apžvalgos dalis skirta techninių procesų analizei ir aprašymui, siekiant išnagrinėti siūlomus automatizavimo metodus.

1.1. Skaitmeninė transformacija organizacijoje

Pasak Liu et al. [1], skaitmeninė transformacija apibūdina skaitmeninių technologijų integraciją verslo procesuose. Imran et al. [2] teigia, jog ši ketvirtoji pramonės revoliucija (angl. *Industry 4.0*) apjungia eksponentiškai augantį moderniųjų technologijų, tokių kaip „daiktų internetas“ (angl. *Internet of Things*), debesų kompiuterija, dirbtinis intelektas, mašininis mokymasis, didieji duomenys ir kt., panaudojimą pramonėje ir kitose srityse. Autoriai pastebi, jog ši revoliucija stipriai padidina verslo objektų veiklos efektyvumą, produktyvumą, tačiau tuo pat metu kelia ir naujų iššūkių: kinta žinių ir kompetencijų poreikis, numatoma, jog atsiras poreikis kurti lankstesnes darbo vietas. Skaitmeninės transformacijos organizacijoje sąvoka svarbi aptariant siekį naudotis didžiųjų duomenų teikiama nauda: keičiant organizacijos procesus, svarbu išanalizuoti ar verslas yra pasiruošęs daugumą žmogiškųjų procesų, egzistavusių kaip rutina, perleisti kompiuteriui ir automatinėms programoms. Nors žmogaus priežiūros analizuojant duomenis reikia nemažai, tačiau didžiųjų duomenų analizė lenkia žmogaus skaičiavimų ribas tiek efektyvumu, tiek laiko bei klaidų minimizavimo galimybe.

Skaitmeninė transformacija suprantama ne tik verslo įmonių aspektu, bet yra ir viena sėkmingai vystomos valstybės ekonomikos dalių. Kasmet Europos Sąjungos šalių pažanga vertinama pagal įvairius kriterijus, vienas iš šių – skaitmeninės ekonomikos ir visuomenės indeksas DESI (angl. *Digital Economy and Society Index*) vertina šalis pagal 5 kriterijus: galimybę naudotis interneto ryšiu, žmogiškąjį kapitalą, naudojimąsi internetu teikiamomis paslaugomis, skaitmeninių technologijų integraciją ir skaitmenines viešąsias paslaugas. 2021 metų DESI ataskaitoje [3] teigiama, kad Lietuva 2021 metais užėmė 12-tą vietą tarp ES šalių pagal skaitmeninių technologijų integraciją – šioje srityje vertinamas naudojimasis elektroninėmis priemonėmis dalinantis informacija, aktyvumas socialiniuose tinkluose, didžiųjų duomenų naudojimas, debesijos bei DI technologijos, IRT (informacinių ir ryšių technologijų) panaudojimas žaliajai veiklai vykdyti, elektroninė prekyba bei šios apyvarta. Nors Lietuvos vidurkis žemesnis nei ES, tačiau pasižymi dvi nagrinėjamos sritys: socialiniai tinklai bei informacijos sklaida šiek tiek viršija ES vidurkį, o dirbtinio intelekto panaudojimas bei IRT darniai aplinkai vystyti viršija ES vidurkį apie 10 %. 2021 metų ataskaitoje minima, jog Lietuva yra viena iš pirmųjų šalių ES, parengusi nacionalinę dirbtinio intelekto strategiją: čia rekomenduojama kurti dirbtiniam intelektui naudoti palankią duomenų aplinką. Daug dėmesio skiriama viešajam sektoriui: siekiama, kad saugomi duomenys atitiktų tarptautinius standartus. Dėl šios priežasties pastebima pažanga skaitmeninių viešųjų paslaugų teikiamomis galimybėmis: ataskaitoje teigiama, jog 2020 metais elektroniniuose valdžios vartuose buvo integruota apie 600 elektroninių paslaugų: gyventojai ir organizacijos turėjo galimybę gauti iš anksto užpildytas elektronines formas. Didelė dalis vartotojų taip pat naudojasi teikiamomis paslaugomis, o interneto erdvėje taip pat pateikiami atvirieji duomenys.

Lietuvos Dirbtinio intelekto strategijos [4] ataskaitoje aprašomos ekspertų grupės išvados apie dabartinę Lietuvos DI integracijos situaciją bei pasiūlymai. Ataskaitoje teigiama, jog DI gali būti integruojamas tiek viešajame, tiek privačiame sektoriuje. Privatus sektorius pasižymi greitesne integracija naudojantis inovacijomis, o ataskaitoje teigiama, jog „nauda verslui akivaizdi: didesnis darbo produktyvumas, logistikos optimizavimas, įprastų procesų automatizavimas, greitesni verslo sprendimai ir tikslesnės vartojimo rinkos prognozės“. Viešajam sektoriui DI gali būti itin naudingas: DI panaudojimas gali gerinti piliečių gerovę, pvz., naudojant technologiją nusikaltimų prognozei ir telkiant teisėsaugos pareigūnus rizikos zonose; virtualių asistentų panaudojimas gali supaprastinti paslaugų teikimą; panaudojus išmaniąsias technologijas galima reguliuoti vyriausybės darbų srautą [4].

Vis dėlto dokumentuose, parengtuose Europos Komisijos (EK) jau teigiama, jog ES strateginiams tikslams pasiekti Pramonės 4.0 pagrindinių aspektų nepakanka. EK ataskaitoje [5] teigiama, jog Pramonė 4.0 neapima tokių svarbių temų kaip klimato krizė ar įtampa tarp socialinių ryšių – kitaip tariant, Pramonės 4.0 principai yra tiksliai nukreipti į verslo modelių optimizavimą ir teikiamą ekonominę naudą ir atsako į daugelį klausimų, kurie yra keliami šią dieną, tačiau neatsako į kylančias naujas problemas, kurios bus svarbios ateinantį dešimtmetį. Ataskaitoje [5] mokslininkai teigia, jog Pramonė 5.0 yra lankstus modelis, parodantis pasikeitusį mąstymą po COVID-19 pandemijos: teigiama, mokantis iš keleto praėjusių metų įvykių, reikalinga industrinė sistema, atsparesnė bet kada įvykti galintiems ekonomikos šokui, stresui ir kitiems *force majeure* įvykiams. Sistema taip pat turi integruotis su socialine ir globalia aplinka ir kuo labiau priartėti prie jau dabar teigiamų žaliųjų principų. Kitame Europos Komisijos išleistame straipsnyje [6] taip pat teigiama, jog Pramonės 5.0 nereikėtų laukti kaip Pramonės 4.0 tęsinio ar alternatyvos – tai yra požiūris kaip ateityje kartu egzistuos dabartiniai pramonės principai kartu su augančiais socialinių bei aplinkosaugos problemų trendais ir sprendimais. Straipsnyje [6] minimas ir naujas požiūris į visuomenę – Visuomenė 5.0 (angl. *Society 5.0*), pasirodęs Japonijoje dar 2016 metais. Ataskaitoje paaiškinama, jog visuomenės gyvenimo principai keitėsi nuo seniausių laikų: medžiotojų – rinkėjų visuomenė bei viduramžių visuomenės apima pirmuosius du visuomenės konceptus, trečiasis visuomenės etapas prasidėjo nuo trečiosios pramonės revoliucijos, o Visuomenė 4.0 atsirado su informacijos skaitmenizavimu iki šios dienos. Nagrinėjamoje politikoje [6] teigiama, kad Visuomenė 5.0 atsiranda ieškant balanso tarp ekonominio vystymosi ir socialinių bei aplinkosaugos problemų sprendimo. Mokslininkai teigia, kad Pramonė 5.0 apims platesnį požiūrį į pagrindinius tris konceptus: orientavimąsi į žmogų (angl. *human-centricity*), tvarumą bei atsparumą. Tai galima paaiškinti kaip organizacijų požiūrį į žmogų kaip svarbiausią vertybę: klausimą „Ką galime padaryti su technologijomis“ keistų klausimas „Ką technologijos gali padaryti mums“. Politikoje minimas ir naujas požiūris į darbuotojų – technologijų santykį: ne darbuotojas turi adaptuotis ir keisti įpročius, o sistema turi būti prisitaikanti prie darbuotojo poreikių ir padėti, mokyti naudotis technologijomis. Politikoje [6] kaip antroji Pramonės 5.0 dalis pateikiamas tvarumas ir teigiama, jog Pramonė 5.0 plėtos ciklinius procesus siekiant pakartotinai naudoti, panaudoti ir perdirbti žaliavas, taip sumažinant poveikį aplinkai. Atsparumas Pramonėje 5.0 atsiranda po COVID-19 pandemijos laikotarpio: poreikis pramonei dirbti nepertraukiamą darbą ir būti kuo atsparesnei bet kokioms įvyksiančioms krizėms. Moksliniame šaltinyje teigiama, kad COVID-19 pandemija išryškino dabartinės pramonės sistemos trapumą ir nepasiruošimą globalioms katastrofoms [6]. 2020 metų DESI ataskaitos apibendrinamajame EK pranešime spaudai [7] teigiama, jog 2020 metais sudarytoje ataskaitoje matomas pandemijos poveikis visoms DESI nagrinėjamoms sritims, todėl statistikoje siekiama įvertinti priemones, kurių ėmėsi valstybės narės siekdamos sumažinti

pandemijos pasekmes. Pranešime spaudai teigiama, jog stipriai remiama sveikatos apsaugos sistema: sistema koordinuota įdiegus taikomąsias programas bei platformas, sudaryta rekomendacija dėl duomenų rinkimo, kuris vyko naudojant mobiliąsias aplikacijas siekiant apriboti viruso plitimą. Apibendrintame pranešime spaudai teigiama, jog valstybės narės rodo pažangą visose 5-iose nagrinėjamose DESI nagrinėjamose srityse, jau minėtose anksčiau. Pastebėta, kad reikalinga pažanga skaitmeninių įgūdžių panaudojime: 42 % ES piliečių dar trūksta pagrindinių skaitmeninių įgūdžių, teigiama, kad informacinių ir ryšių technologijų (IRT) specialistų rasti vis dar yra sudėtinga. Ataskaitoje pastebimas ir šiek tiek išaugęs naudojimasis interneto paslaugomis, nors ši tendencija jau buvo matoma anksčiau (palyginimui – 2014 m. 75 % ES gyventojų bent kartą per savaitę naudojosi interneto paslaugomis, o 2020 m. duomenimis ši dalis išaugo iki 85 %). Ataskaitoje taip pat skiriamas dėmesys skaitmeninių technologijų diegimui įmonėse. Pastebima, jog didelės organizacijos naudojasi debesijos paslaugomis (38,5 % įmonių), 32,7 % taip pat taiko didžiųjų verslo duomenų analizės metodus. Tačiau mikro-įmonės, mažosios bei vidutinės įmonės šiomis technologijomis naudojasi rečiau: 17 % naudojasi debesijos paslaugomis ir tik 12 % naudojasi didžiųjų duomenų analitika.

Aukščiau minėti šaltiniai rodo, kad globaliame pasaulyje verslo organizacijų ir valstybinių struktūrų sistemos keičiasi ir turi keistis kartu, siekiant išlikti konkurencingoms pasauliniame kontekste. Vis dėlto „McKinsey & Company“ atliktame tyrime [8] pastebima, jog yra organizacijų, dar nepasiekusių Pramonės 4.0 lygio dėl įvairių priežasčių: sudėtinga koordinuoti veiklą didelėse organizacijose tarp skirtingų padalinių, trūksta pasiryžimo pertvarkymams, trūksta specialistų, atsiranda susirūpinimas dėl duomenų saugumo, trūksta informacijos tiek apie realią naudą, tiek apie trečiųjų šalių paslaugų teikėjus. Imran et al. [2] teigia, jog dėl sparčiai besikeičiančios globalios aplinkos verslas turi adaptuotis, visų pirma siekdamas išlikti konkurencingas. Mokslininkai teigia, jog čia svarbu siekti kompetencijomis paremtos darbuotojų struktūros: identifikuoti svarbiausias darbui vykdyti reikalingas žinias ir taip paskirstyti veiklą darbuotojams pagal jų turimą patirtį, o ne atvirkščiai – pagal tam tikrą darbo poziciją. Straipsnyje teigiama, jog taip paprasčiau prisitaikyti prie Pramonės 4.0 keliamų iššūkių. Tokia pozicija taip pat padeda identifikuoti kritiškai reikalingą darbuotojų patirtį. Straipsnyje [2] aptariama, jog svarbu identifikuoti ir atkreipti dėmesį į egzistuojančius socialinius faktorius pritaikant organizacijos vykdomą veiklą pokyčiams.

1.1.1. Skaitmeninės transformacijos keliami iššūkiai

Ataskaitose ir mokslo šaltiniuose pateikiamoje informacijoje pastebima, jog skaitmeninė transformacija tiek organizacijoje, tiek visoje valstybėje nėra paprastas žingsnis. Skaitmenizuojant rutininius procesus ar automatizuojant darbą atsiranda įvairių iššūkių.

Technologijos. Pasak Syed et al. [9], nauda gaunama tada, kai yra reali galimybė pritaikyti technologijas įmonės procesuose. Visoms įmonėms būdinga ieškoti sprendimų, padedančių gerinti įmonės procesų efektyvumą per operacijų ir procesų valdymą įmonės viduje. Tam puikiai tinka informacinių technologijų (IT) procesų gerinimas, tačiau inovatoriai ir sistemų naudotojai dažnai susiduria su aplikacijų programavimo sąsajos (angl. *application programming interface*, API) trūkumu, kas neleidžia įmonės informacinės sistemos keisti iš esmės [9]. Pakeisti informacinę sistemą yra sudėtingas uždavinys, praktiškai reikalaujantis didelių investicijų įmonės viduje. Čia galima paminėti ir robotinių procesų automatizavimo (RPA) sąvoką. Šis metodas apibūdina specifinę automatizavimo kryptį įmonės sistemose. Pasak daugelio mokslinių darbų autorių [9, 10, 11, 12], robotinių procesų automatizavimas siejamas su sistemomis, kuriamomis pasikartojančių

užduočių atlikimui. Užprogramuotas robotas atkartoja žmogaus atliekamus veiksmus iš eilės – sistema geba sekti tuo pačiu keliu, atidaryti reikalingas programas ir tai daryti reikalingą kiekį kartų. Robotiniai procesai dažniausiai automatizuoja itin gerai struktūrizuotus, griežtomis taisyklėmis aprašytus procesus, pavyzdžiui, kopijuoja failus, surenka ir sistemina informaciją, tvarko automatines elektroninio pašto dėžutes. Sėkmingas RPA panaudojimas priklauso nuo roboto išmanumo, reikalingo atliekant tam tikrą užduotį. Nagrinėtame moksliniame straipsnyje pateikiamas automatinis dokumentų anomalijų radimas naudojant dirbtinį intelektą ir mašininio mokymosi algoritmus [13]. Autoriai teigia, jog panaudojant dirbtinio intelekto algoritmus bei mašininį mokymąsi atsiranda galimybė nagrinėti ir išgauti informaciją, padedančią klasifikuoti, susieti, optimizuoti, grupuoti, prognozuoti ir atrasti egzistuojančius šablonus (angl. p*atterns*) turimuose duomenyse. Norint šiuos procesus efektyvinti, dažnai į pagalbą pasitelkiami RPA procesai, tuomet įmonės procesų automatizavimas leidžia pasiekti aukštą išmanumo lygį. Tai yra vienas iš daugelio įmonės skaitmeninės transformacijos aspektų. Tačiau „McKinsey & Company“ [8] atlikto tyrimo metu pastebima, kad vienas iš didžiausių rūpesčių organizacijai pereinant prie procesų skaitmenizavimo bei automatizavimo yra veiksmų koordinacija tarp skirtingų organizacijos padalinių: informacinių technologijų, finansų, pardavimų, gamybos bei kitų. Esamos funkcijos gali stipriai skirtis, todėl net tarpusavyje susijusių procesų automatizavimas gali tapti iššūkiu.

Žmogiškieji ištekliai. Lietuvos Dirbtinio intelekto strategijos ataskaitoje [4] minima, jog DI panaudojimas paveikia darbo jėgą, kadangi automatizuojamos užduotys, kurios anksčiau buvo daromos tik žmogaus. Ataskaitoje prognozuojama, jog DI integracijai organizacijose augant, darbo rinkoje bus pastebimi pokyčiai, kurie sumažins rutininių, nekūrybingų užduočių kiekį. „McKinsey & Company“ [8] teigia, jog vienas iš didžiausių barjerų pramonės įmonėms pereiti prie Pramonės 4.0 yra talentų trūkumas – darbo rinkoje yra mažai darbuotojų, turinčių reikalingas kompetencijas, pvz., duomenų mokslininkų (angl. *data scientists*). Tai lemia susirūpinimą pereinant prie naujų aplikacijų ir verslo procesų, kadangi žinių ir patirties trūkumas gali stabdyti kasdienes operacijas. Lietuvos Dirbtinio intelekto strategijos [4] ataskaitoje rekomenduojama ruoštis darbo rinkos pokyčiams peržiūrint mokymo programas: skatinti techninių įgūdžių lavinimą, tyrinėti technologijos, inžinerijos, matematikos sritis. Šių sričių specialistų trūkumas numatomas ruošiantis pokyčiams organizacijose.

Organizacijos požiūris: organizacija jau turi būti orientuota į naujas technologijas ir pokyčius. „McKinsey & Company“ [8] teigia, jog dar viena svarbi priežastis, dėl kurios vengiama automatizuoti procesus ir pereiti prie Pramonė 4.0 siūlomų pokyčių, yra pasirengimo trūkumas. Dauguma įmonių vengia procesų automatizavimo, kadangi šioms įmonėms tai būtų radikalus pokytis veikloje.

Duomenų kokybė: Lietuvos Dirbtinio intelekto strategijos [4] ataskaitoje teigiama, jog skirtingų administracijų surinkti duomenys gali būti fragmentuoti, kai kurie duomenų rinkiniai gali būti sunkiai pasiekiami. Šiuo atveju privatus sektorius gali nusistatyti duomenų standartus pagal poreikį, o viešasis sektorius dažnai neturi suvienodinto standarto, tinkamo DI sistemų panaudojimui. Tačiau remiantis „McKinsey & Company“ ataskaita [8], panaši problema gali kilti ir privataus sektoriaus organizacijose. Duomenys iš skirtingų šaltinių gali būti nepalyginami tarpusavyje, o tokių duomenų panaudojimas – viena iš svarbiausių Pramonės 4.0 dalių – gali tapti iššūkiu organizacijoms.

Saugumas: dar vienas iš „McKinsey & Company“ [8] atlikto tyrimo pastebėtų iššūkių yra nepasitikėjimas trečiųjų šalių paslaugų pardavėjais. Kai kurie automatizavimo procesai yra

atliekami ne įmonės ištekliais, o perkant paslaugas iš įvairių specialistų, tačiau tuomet organizacijai kyla rizika valdyti duomenis, kadangi ši informacija pateikiama tretiesiems asmenims. Tyrime pastebima, kad organizacijos nenoriai dalinasi vidiniais verslo procesų duomenimis. Čia kyla ir kitas iššūkis – nuspręsti, kada geriau įsigyti paslaugas iš trečiųjų šalių, o kada naudotis ištekliais, jau turimais organizacijos viduje.

1.1.2. Siūlomi technologijų integracijos sprendimai

Naudojamų technologijų tobulinimas. „McKinsey & Company“ tyrime [8] siūloma vietoje visos infrastruktūros reorganizavimo susikoncentruoti keletui technologinių pokyčių. Ataskaitos autoriai siūlo susitelkti į keletą naudingų skaitmenizuotų įrankių, padedančių pradėti gerinti verslo procesus. Siūloma atsižvelgti į skaitmeninį našumo valdymą, pavyzdžiui, švieslenčių naudojimą efektyvumui pristatyti ir matuoti – autoriai teigia, jog tai vienas greičiausių ir paprasčiausių įrankių, galinčių padėti pamatus tolimesnei įmonės skaitmenizacijai. Kitas minimas vertingas įrankis – numatomas sistemų palaikymas, kuris, nors jau egzistuojantis daugumoje įmonių, patobulintas naujais mašininio mokymosi algoritmais gali būti keletą kartų tikslesnis ir naudingesnis. Autorių teigimu, patobulinti sistemų priežiūros metodai gali padėti sumažinti priežiūros kaštus iki 15 %. Kitas svarbus įrankis – tobulinamas procesų automatizavimas. „McKinsey & Company“ ataskaitos [8] autoriai teigia, jog robotizavimas įmonėje vis augs, o robotizacijos kaštai pamažu kris. Taip pat minima, jog anksčiau minėti rutininių procesų robotizavimo metodai gali padėti tokiuose kasdieniuose procesuose, kaip paklausos planavimas bei užsakymų valdymas. Kitas svarbus patobulinimas – skaitmeninis kokybės valdymas. Gerinant įrankius, naudojamus kokybės užtikrinimui, pastebimas efektyvumo didinimas, klaidų sekimo užtikrinimas, kaštų mažinimas. Pažengusios organizacijos gali naudotis didžiųjų duomenų teikiamomis galimybėmis kokybės analizei, pvz. automatizuota kokybės analize problemų priežasčių identifikavimui [8].

Dėmesys žmogiškiesiems ištekliams. Remiantis 2020 metais „McKinsey & Company“ [14] atlikta apklausa, po COVID-19 pandemijos pradžios organizacijos atranda ir kitų būdų kovoje dėl talentingų darbuotojų. Esant didelei paklausai, atsiranda poreikis darbuotojus apmokinti įmonės viduje, taip siekiant įgauti trūkstamų žinių, reikalingų naujiems procesams organizacijoje vykdyti. Atliktoje apklausoje, kurioje nagrinėti beveik 900 respondentų atsakymai, į klausimą kaip efektyviausiai užpildyti spragas tarp turimų žmogiškųjų išteklių, 53 % respondentų nurodė, jog reikia apmokinti esamus darbuotojus ir plėsti šių žinias; 21 % nurodė išorinę darbuotojų samdą, 20 % - žmogiškųjų išteklių perskirstymą ir 6 % nurodė darbuotojų samdą pagal kontraktus [14]. Apklausos autoriai pastebi, jog gerinant darbuotojų žinias ir įgūdžius, atsiranda galimybė gamybos procesais gerinti kuriamą vertę vartotojui ir tai atlikti optimalesniais metodais. Tokia nauda išlieka ilgajame laikotarpyje.

Trečiųjų šalių paslaugų teikėjų analizė. McKinsey & Company [14] apklausos rezultatams rodant didelį nepasitikėjimą išoriniams paslaugų teikėjams, autoriai siūlo sudaryti ir analizuoti partnerių portfelį. Autoriai teigia, jog svarbu numatyti procesus, kurie gali būti palaikomi įmonės turimų išteklių – tai apima svarbiausius strateginius veiklos procesus, kurie turėtų būti palaikomi įmonės viduje siekiant palaikyti konkurencingumą rinkoje ir apsvarstyti paslaugas, kurios gali pagerinti procesų vykdymą bendradarbiaujant su trečiųjų šalių paslaugų teikėjais. Tam svarbu analizuoti ir bendradarbiauti su keletu patikimų paslaugų teikėjų. Autorių teigimu, Pramonė 4.0 rinkoje pateikė galimybę bendradarbiauti ir kooperuoti procesus net su keletu įvairių paslaugų teikėjų. Ataskaitoje

teigiama, jog portfelio sudarymas prasideda nuo rinkos išmanymo bei gerų partnerių pasirinkimo ir ryšių su jais palaikymo.

Atlikus pokyčių verslo organizacijų viduje analizę, nagrinėjamas dabartinis technologijų panaudojimas verslo įmonėse. Technologijos yra pakankamai opi sritis organizacijoms – tiek valstybinėms, tiek privačioms – kadangi technologijų kaita yra sparti, verslo procesus pritaikyti ir pertvarkyti yra vienas iš skaitmeninės transformacijos iššūkių. Toliau nagrinėjama viena iš svarbiausių ir organizacijose dažniausiai naudojamų informacinių technologijų sistemų – duomenų bazės.

1.2. Duomenų bazių sąvoka įmonių veikloje

Duomenų bazės sąvoka egzistuoja kiekvienoje organizacijoje – tiek viešojoje, tiek privačioje – todėl beveik kiekvienas automatizavimo sprendimas siejasi su šia informacinių technologijų dalimi. Pasak Butkienės et al. [9 p. 15], „duomenų bazė yra viena iš pagrindinių kompiuterizuotos informacinės sistemos komponenčių“. Knygoje autoriai pastebi, jog dar projektuojant duomenų bazę svarbi kompiuterizuojamos srities analizė: čia nagrinėjami duomenys, kurie bus saugojami duomenų bazėje, tikslinama, kaip šie duomenys turės būti apdorojami. Vėliau duomenų bazės kūrimas realizuojamas tam tikroje duomenų bazių valdymo sistemoje [15]. Vadinasi, siekiant automatizuoti net ir tam tikrą procesą reikia atsižvelgti į naudojamą duomenų saugojimo sistemą, o projektuojant naują duomenų bazę taip pat reikia išnagrinėti ir numatyti saugomų duomenų tipus, galimus pokyčius ateityje ir susipažinti su duomenų bazių sistema ir jos galimybėmis.

Butkienė ir kt. [15] duomenų bazių sąveiką su kitomis taikomosiomis programomis apibūdina kaip užklausų pateikimą duomenų bazei ir informacijos grąžinimą naudojamai programai: duomenų bazėms pasiekti dažniausiai naudojama struktūrizuota užklausų kalba SQL (angl. *structured query language*). Pasak autorių, taikomoji programa prisijungia prie duomenų bazės ir pateikia SQL kalba sudarytą užklausą, o užklausos rezultatai su informacija iš duomenų bazės pateikiami programai. Pasak Butlerio ir kt. [16], SQL yra ANSI (angl. *American National Standarts Institute*) standartas, sukurtas XX a. 8 dešimtmetyje. Autoriai teigia, jog standarto sukūrimo priežastis – siekis standartizuoti darbą su vienodos struktūros duomenimis skirtingose platformose. Galima teigti, jog standartinė duomenų bazių užklausų kalba supaprastina pokyčius organizacijos informacinių technologijų srityje, kadangi anksčiau naudojami procesai, paremti SQL užklausomis, gali būti panaudojami net ir keičiant duomenų bazių valdymo sistemą. Raasveldt [17] teigia, jog SQL kalba palaikoma visų didžiausių reliacinių duomenų bazių paslaugų teikėjų. Autoriaus teigimu, SQL kalba pasiekiama ir daugeliui ne reliacinių (angl. *No-SQL*) DBVS naudotojams – čia ši kalba palaikoma dėl paprastumo ir žinomumo.

Haigh [18] rašo, jog duomenų bazių valdymo sistemos konceptas atsirado 1963-aisiais metais, kai Bachman sukūrė pirmąją duomenų bazių valdymo sistemą siekdamas automatizuoti kompanijos „General Electric Company“ procesus. Straipsnio autorius teigia, jog duomenų bazių valdymo sistemos sukūrimas buvo būtinas skaitmenizuojant kasdienius procesus. Apie tai rašo ir anksčiau minėtas duomenų bazių valdymo sistemų kūrėjas Bachman [19]. Autorius teigia, jog duomenų bazių valdymo sistema buvo sukurta apjungiant daugelį elementų, jau egzistavusių to meto moksliniuose šaltiniuose bei sistemose. Automatizavimo metodai buvo pritaikyti prie nagrinėtos kompanijos „General Electric Company“ poreikių ir apėmė šiuos pagrindinius elementus:

- Galimybė tiesiogiai pasiekti duomenų bazę naudojant virtualią kompiuterio atmintį, efektyvumą padidinant įvairiais tuo metu naudojamais metodais (klasterizuojant įrašus, naudojant raktus ir kt.).
- Naudojamas tinklo modelis, kuriame loginiai įrašai apjungiami su fiziniais įrašais virtualioje atmintyje.
- Naudojama duomenų aprašymo kalba (angl. *DDL – Data Description Language*), kuri padeda nustatyti laikomų duomenų tipus, ryšius tarp įrašų bei apribojimus (angl. *constraints*).
- Naudojama duomenų laikymo ir išgavimo kalba (angl. *DML – Data Manipulation Language*), kuri buvo patogi to meto procedūroms kompiuteriu atlikti.
- Duomenų manipuliavimo formuluotės, egzistavusios tuo metu, buvo *store, retrieve, modify* bei *delete.*

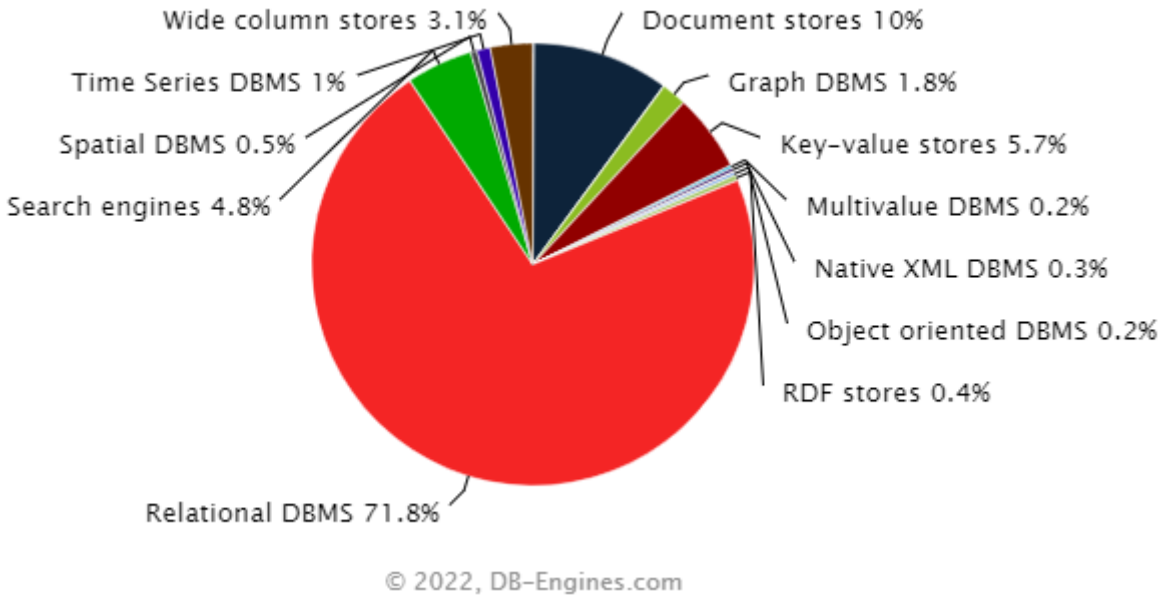
Tuo metu DBVS laikyta ypatinga sistema, veikianti kaip darbinė saugykla, išsauganti informaciją apie įrašus kompiuterio atmintyje bei užtikrinanti duomenų pateikimą naudojamoms programoms.

Apibendrinant galima teigti, jog duomenų bazei tinkamai veikti ir verslo procesams sklandžiai vykdyti reikalinga ir duomenų bazė, ir duomenų bazės valdymo sistema. Galima numanyti, jog pirmosioms duomenų bazių valdymo sistemoms atsiradus beveik prieš 60 metų, naudojama technologija jau patobulėjusi. Šiai dienai galima atrasti duomenų bazių valdymo sistemų, pritaikytų kiekvienai verslo vykdomai veiklai ir saugomiems duomenų tipams pasiekti bei tvarkyti.

1.2.1. Duomenų bazių valdymo sistemų tipai

Tobulėjant naudojamoms technologijoms, išsiplėtė ir duomenų bazių valdymo sistemų tipų sąrašas. Pasak Butkienės ir kt. [15] dar prieš projektuojant duomenų bazę reikia išanalizuoti laikomus duomenis. Išnagrinėjus saugojamus duomenis galima pasirinkti tam pritaikytą duomenų bazių bei duomenų bazių valdymo sistemas. Šiuo metu egzistuoja įvairių duomenų bazių tipų, kurie atsiranda siekiant patenkinti verslo poreikį saugoti įvairių tipų duomenis. Moksliniuose darbuose išskiriama keletas skirtingų DBVS tipų: reliacinės (angl. *relational*), ne reliacinės (angl. *NoSQL*), iš kurių populiariausios DBVS yra stulpelių (angl. *column-oriented*) [21, 22], *object-oriented* [21], *key-value* [21, 22], dokumentų [21, 22] (angl. *document store*), grafų (angl. *graph*) [21, 22] ir kt. Dauguma šių tipų atsirado dėl poreikių verslo duomenims saugoti ir taip pat siejasi su didžiųjų duomenų atsiradimu. DB-engines.com[1] puslapyje skaičiuoti 2022 metų kovo mėn. DBVS modelių populiarumo įvertinimai pateikiami paveiksle žemiau (žr. 1 pav.).

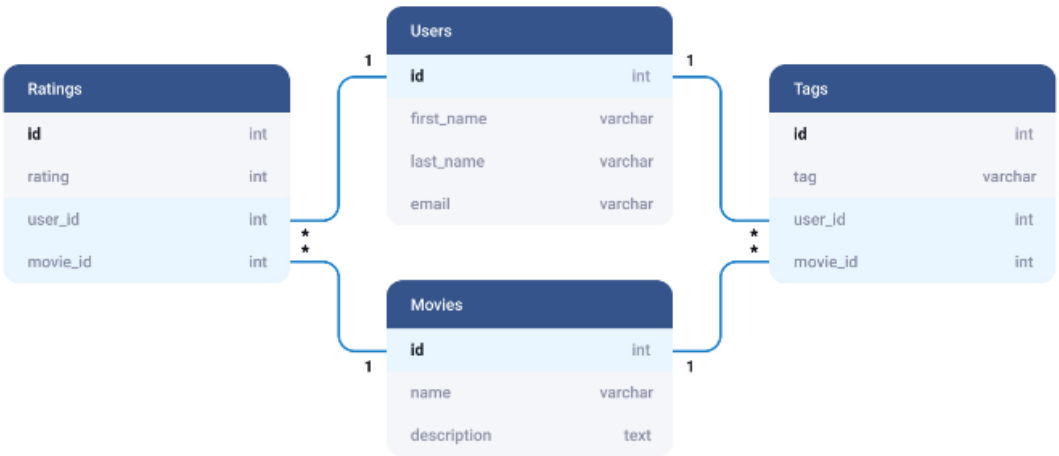
[1] https://db-engines.com/en/



1 pav. DB-engines.com sudarytas DBVS populiarumo grafikas (šaltinis: DB-engines.com)

Prieš aptariant populiariausias duomenų bazių valdymo sistemas, pravartu aptarti pagrindinį nagrinėjamų DBVS skirtumą – tai reliacinių ryšių DBVS ir ne reliacinių ryšių DBVS. Kaip matoma paveiksle viršuje, reliacinių ryšių DBVS populiarumas siekia kiek daugiau nei 70 %, o ne reliacinės DBVS sudaro likusius 30 %. Toliau apibūdinamos populiariausios grafike išvardintos DBVS.

Reliacinė DBVS. Pagal paveiksle pateiktus duomenis populiariausias duomenų bazių valdymo sistemų tipas yra reliacinė DBVS. Tai pastebima ir kituose moksliniuose šaltiniuose [17]. Ši duomenų bazių valdymo sistema sukurta Frank E. Codd 1970-aisiais metais [20]. Autoriaus aprašytas reliacinis modelis tapo pagrindu populiariausiai ir dažniausiai naudojamai DBVS. Pagrindinis šios sistemos elementas – reliaciniais ryšiais apjungta lentelė. Reliacinės duomenų bazės pavyzdys pateikiamas paveiksle žemiau.



2 pav. Reliacinė duomenų bazė (xbsoftware.com diagrama[2])

[2] https://xbsoftware.com/

Sistemoje naudojama reliacinė schema (angl. *relational schema*), kuri yra aprašoma nurodant lentelės pavadinimą bei tam tikrą skaičių atributų su fiksuotu duomenų tipu. Kiekvienas įrašas reliaciniais ryšiais paremtame modelyje egzistuoja eilutėje ir susideda iš atributų reikšmių. Raasveldt [17] teigia, jog reliacinės duomenų bazės nuo šių atsiradimo įrodė, jog yra vienas iš stabiliausių ir geriausiai kontroliuojamų įrankių verslo procesams atlikti. Šios DBVS tinkamos ir dideliems duomenų kiekiams laikyti ir dirbti. Anot Raasveldt [17], du didžiausi reliacinių DBVS privalumai yra normalizuotas duomenų saugojimas, kuris padeda išvengti duomenų dubliavimosi ir pagerina duomenų vientisumą, bei duomenų pasiekiamumas – atliekant duomenų analizę, gauti rezultatai yra atskiriami nuo kitų duomenų, saugojamų duomenų bazėje. Tai leidžia laisvai pasiekti ir tvarkyti duomenis. Kaip trūkumai įžvelgiamas sudėtingas duomenų saugojimas ir užklausų rašymas, kai turimi duomenys yra dideli. Ieškant tam tikros informacijos, reliacinė duomenų bazė atlieka paieškos ir sutapties (angl. *match*) operacijas. Esant dideliam kiekiui ryšių ir apjungtų duomenų, šis veiksmas tampa laikui imliu.

Dokumentų DBVS. Kitas populiarus duomenų bazių valdymo sistemų tipas yra dokumentų DBVS, dažniausiai apibūdinamos kaip neturinčios konkrečios schemos duomenų bazės. Tai yra ne reliacinė DBVS. Paveiksle viršuje (žr. pav. 1) šių DBVS populiarumas siekia 10 %. Pasak Nayak et al. [21], lyginant su reliaciniais ryšiais paremtu modeliu, schemos nebuvimas padeda visus įrašus duomenų bazėje laikyti neturint vienodos struktūros – vadinasi, skirtingi įrašai duomenų bazėje gali turėti skirtingus atributus. Atributų skaičius gali būti skirtingas kiekvienam įrašui, ir kiekvienas įrašas gali turėti keletą skirtingų reikšmių, išsaugotų atribute (stulpelyje). Dar vienas dokumentų DBVS privalumas yra tai, jog šios sistemos gali naudoti įvairius žymėjimus, kurie gali būti apdorojami kitose programose, taip informaciją duomenų bazėje tiesiogiai siejant su vykdomais įmonėje procesais, dažniausiai JSON tipo failais [21]. Tokio failo tipo pavyzdys nurodytas paveiksle žemiau (žr. 3 pav.).

```
{
  "firstName": "Vardenis",
  "lastName": "Pavardenis",
  "age": 27,
  "address": {
    "streetAddress": "Kauno g. 16",
    "city": "Vilnius",
    "country": "Lithuania",
    "postalCode": "LT-03212"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "123 456-1234"
    },
    {
      "type": "office",
      "number": "987 789-9876"
    }
  ],
  "children": [],
  "spouse": null
}
```

3 pav. JSON tipo faile saugoma įrašo informacija dokumentų DBVS

Paieškos variklio DBVS. *Search engine* yra kita populiari DBVS (4,8 %). Tai sistema, paremta duomenų paieška. Anot Wong [23], tokios DBVS sistemos architektūra primena dokumentų DBVS. Pagrindinės tokių DBVS charakteristikos yra supaprastinta išplėstinių sąvokų ir sakinių paieška,

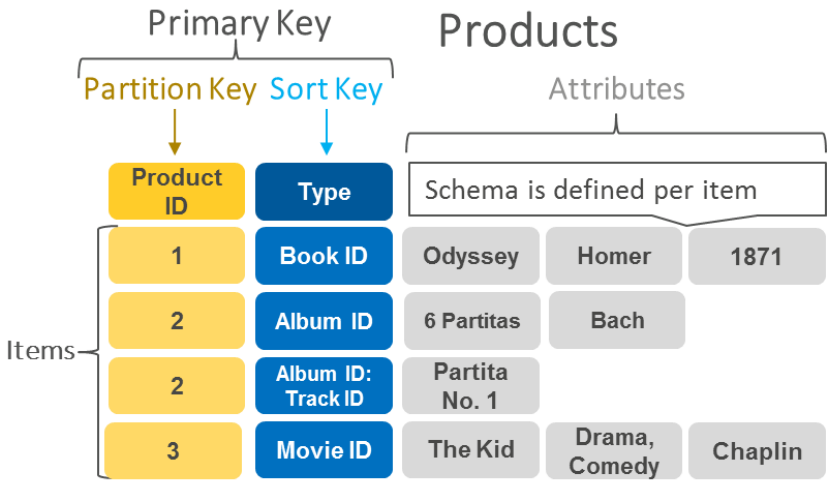
pilna teksto paieška, žodžių šaknies išskyrimas paieškai palengvinti (angl. *stemming*), paieškos rezultatų rangavimas ir grupavimas, paieškos distribucija. Žemiau pateikiamas *search-engines* paieškos užklausos rašymo pavyzdys.

```
POST employees/_search
{
  "query": {
    "match": {
      "phrase": {
        "query" : "heuristic"
      }
    }
  }
}
```

4 pav. Paieškos variklio DBVS užklausos pavyzdys

Pasak Wong [23], paieškos variklio DBVS gali būti naudojama pilnai teksto paieškai, struktūrizuotai paieškai bei analizei.

***Key-value* DBVS**. Pagal populiarumą šios DBVS apima beveik 6 %. Tai dar viena ne reliacinė DBVS. Teigiama, jog tai vienos paprasčiausiai interpretuojamų DBVS [21]. Šiose DBVS laikoma pora raktų ir šių raktų reikšmės, o informacija grąžinama, kai yra pateikiamos raktų reikšmės. Dėl paprastumo tokios DBVS nėra tinkamos sudėtingoms programoms ir verslo procesams apdoroti, tačiau paprastumas yra naudingas esant tam tikroms aplinkybėms. Tokios DBVS taupo išteklius ir gali gerinti sistemų našumą [21], tačiau nėra tinkamos dideliems duomenų kiekiams saugoti [22].

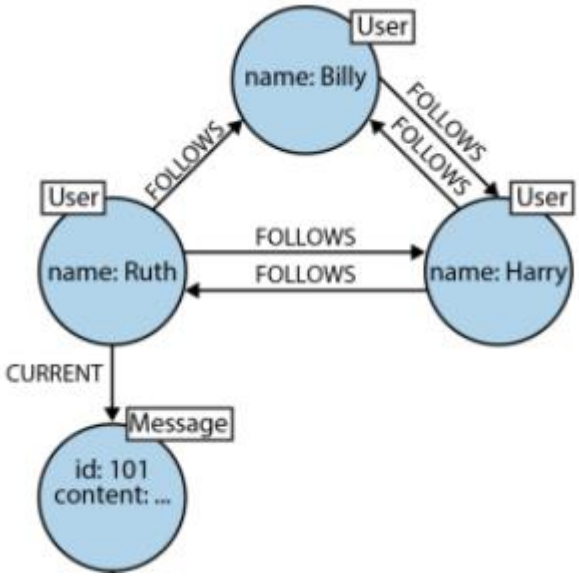


5 pav. *Key-value* DBVS sistemos pavyzdys. Šaltinis: AWS[3]

Grafų DBVS. Dar viena populiari DBVS yra grafų. Šio tipo DBVS populiarumas pastaraisiais metais sparčiai auga (žr. 7 pav.). Grafų duomenų bazės yra mazgų (angl. *nodes*) ir ryšių tarp jų rinkinys. Pasak Robinson ir Webber [24], toks paprastas informacijos pateikimas padeda saugoti labai įvairios struktūros informaciją. Autoriai kaip tinkamą pavyzdį nurodo platformos *Twitter* saugomus duomenis (žr. 6 pav.). Autorių teigimu, grafinė duomenų bazė labiau tinkama įvairiai

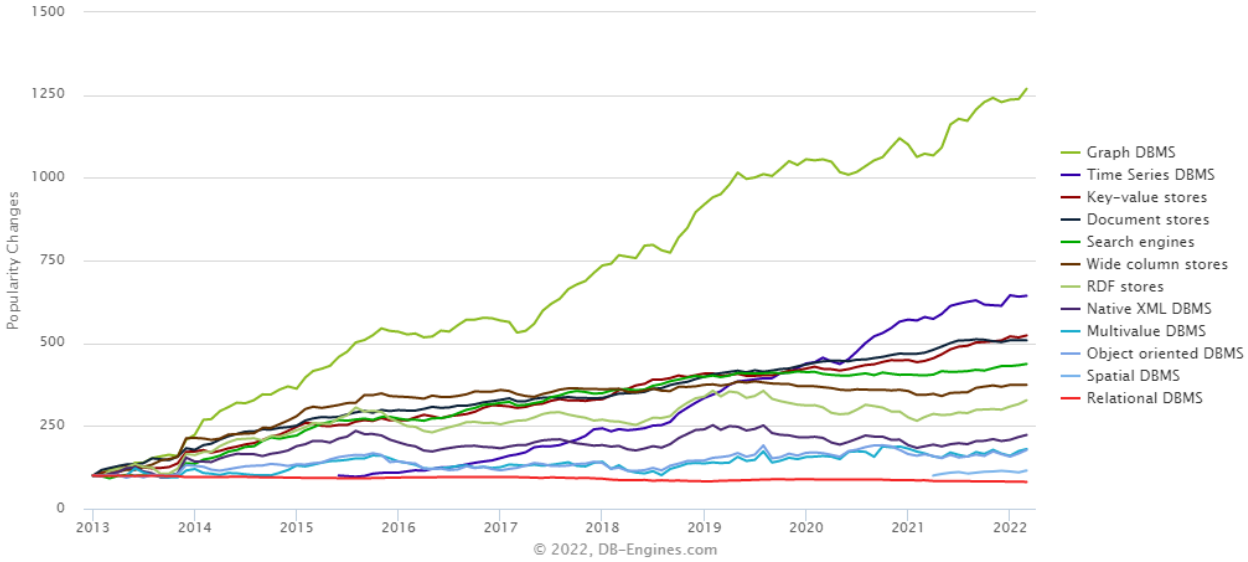
[3] https://aws.amazon.com

informacijai saugoti – kadangi saugomi duomenys gali būti labai turtingi informacijos ir tarpusavyje sietis. Tokius ryšius aprašyti griežtai struktūrizuotais reliaciniais ryšiais yra sudėtinga.



6 pav. Grafų DBVS pavyzdys – *Twitter* platforma (Robinson, 2015)

Toliau pateikiamas DB-engines.com atliktas tyrimas ir pateikiami DBVS populiarumo rezultatai. Pateikiami duomenys aprašo laikotarpį nuo 2013 m. iki 2022 m. kovo mėnesio (žr. 7 pav.).



7 pav. Skirtingų tipų DBVS populiarumo kitimas[4] 2013-2022 m.

Reliacinės duomenų bazės nagrinėjamu laikotarpiu išlieka sulaukiančios pastovaus susidomėjimo. Pateikiami duomenys yra normalizuojami siekiant kuo tiksliau palyginti skirtingą informaciją. Itin aiškiai matomas grafų DBVS populiarumo didėjimas, taip pat auga susidomėjimas laiko eilučių DBVS. Reliacinės duomenų bazės išlieka grafiko apačioje be didelių svyravimų. Grafike pateikiama analizė vykdoma internetinio puslapio DB-engines.com, kurio kūrėjai yra informacinių

[4] https://db-engines.com

sistemų konsultacijos įmonė „SolidIT“, įsikūrusi Austrijoje, konsultuojanti dėl DBVS pritaikymo verslo poreikiams. Atliekant populiarumo įvertinimus skaičiavimai atliekami remiantis žemiau pateikiamais kriterijais:

- **DBVS paminėjimų skaičius interneto svetainėse**: naudojami paieškos sistemų variklių *Google* ir *Bing* duomenys. Ieškomas sistemos pavadinimas kartu su terminu „duomenų bazė“ (angl. *database*), taip siekiant tikrinti tik reikšmingus rezultatus.
- **Bendras susidomėjimas sistema**: paieškų rezultatai *Google Trends*[5] sistemoje.
- **Techninių diskusijų apie sistemą dažnis**: nagrinėjami populiariausi informacinių sistemų klausimų – atsakymų (angl. *Q&A*) tipo puslapiai *Stack Overflow* bei *DBA Stack Exchange*. Čia nagrinėjami su DBVS susiję klausimai bei su šiomis temomis suinteresuotų asmenų skaičius.
- **Darbo pasiūlymų skaičius**: nagrinėjami skelbimai, kuriuose minima nagrinėjama sistema. Analizuojamos internetinės darbo skelbimų platformos *Indeed* bei *Simply Hired*.
- **Profiliai, kuriuose paminėta nagrinėjama DBVS**: informacijai rinkti naudojami *LinkedIn* platformos duomenys.
- **Aktualumas socialinėje medijoje**: nagrinėjami DBVS paminėjimai *Twitter* platformoje.

Atrinkti duomenys yra standartizuojami ir pritaikomi DBVS palyginimui tarpusavyje. Pateikiami populiarumo įvertinimai yra santykiniai, todėl turi būti interpretuojami tik lyginant sistemas tarpusavyje. Žemiau pateikiami 2022 metų kovo mėnesio populiariausių DBVS palyginimas (žr. 1 lent.).

1 lentelė. DB-engines.com sudaryta DBVS populiarumo vertinimo rezultatų lentelė

Populiarumas	**DBVS pavadinimas**	**DBVS tipas**	**DB-engines.com 2022 kovo mėn. vertinimas**
1	Oracle	Reliacinė	1251,32
2	MySQL	Reliacinė	1198,23
3	Microsoft SQL Server	Reliacinė	933,78
4	PostgreSQL	Reliacinė	616,93
5	MongoDB	Dokumentų	485,66
6	Redis	*Key-value*	176,76
7	IBM Db2	Reliacinė	162,15
8	Elasticsearch	Paieškos variklis	159,95
9	Microsoft Access	Reliacinė	135,43
10	SQLite	Reliacinė	132,18

Remiantis pateiktais vertinimo duomenimis, populiariausių DBVS reitinge vis dar dominuoja reliacinės DBVS. Kaip jau minėta anksčiau, vertinimas gali būti interpretuojamas tik lyginant DBVS tarpusavyje, todėl aiškiai pastebimas skirtumas tarp populiariausių sistemų (įvertinimas – daugiau nei 1000) ir dešimtoje vietoje esančios SQLite DBVS (įvertinimas 132). Tai gali būti paaiškinama didelių įmonių ir korporacijų naudojamų sistemų populiarumu: reliacinės DBVS naudojamos daugelį metų, yra patikimos, tinkamos struktūros daugumai verslo organizacijų, DBVS

[5] https://trends.google.com/trends/

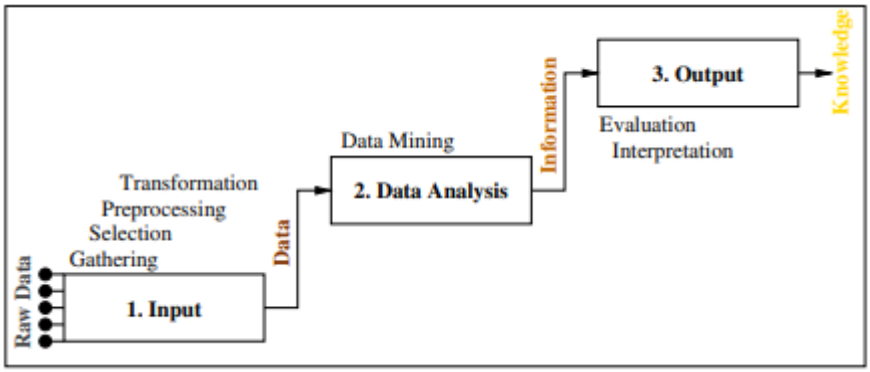
paslaugų teikėjai – taip pat patikimos informacinių technologijų milžinės, todėl naujos, anksčiau minėtos, ne reliacinės DBVS sistemos yra naudojamos tik tam tikriems verslo poreikiams tenkinti ir yra mažiau populiarios.

Toliau darbe nagrinėjama viena iš technologinių procesų tobulėjimo sričių, susijusi su duomenų bazėmis, DBVS bei duomenimis, laikomais bei tvarkomais duomenų bazės valdymo sistema – mašininis mokymasis ir galimybės automatizuoti duomenų paruošimą analizei.

1.3. Mašininis mokymasis ir duomenų inžinerija

Tsai et al. [25] pastebi, jog problemos nagrinėjant didelės apimties duomenis neatsirado staiga, tačiau egzistavo jau daugelį metų, kadangi duomenų sukūrimas ir saugojimas yra paprastesnis veiksmas nei naudingos informacijos juose atradimas. Straipsnyje minimos ir pagrindinės didžiuosius duomenis apibūdinančios „3V“ charakteristikos: didelė duomenų apimtis (angl. *volume*), duomenų įvairovė (angl. *variety*) bei duomenų apdorojimo greitis (angl. *velocity*). Šios charakteristikos nurodo, jog didieji duomenys kuriami ir įrašomi į duomenų bazes greitai, užima daug vietos. Tsai et al. taip pat teigia, kad duomenys gali būti fiksuojami iš skirtingų šaltinių, skirtingais tipais. Siekiant integruoti didžiųjų duomenų panaudojimą organizacijos veikloje ir iš neapdorotų fragmentų gauti informaciją kuo greičiau, reikia procesus pritaikyti anksčiau minėtiems skaitmenizavimo iššūkiams.

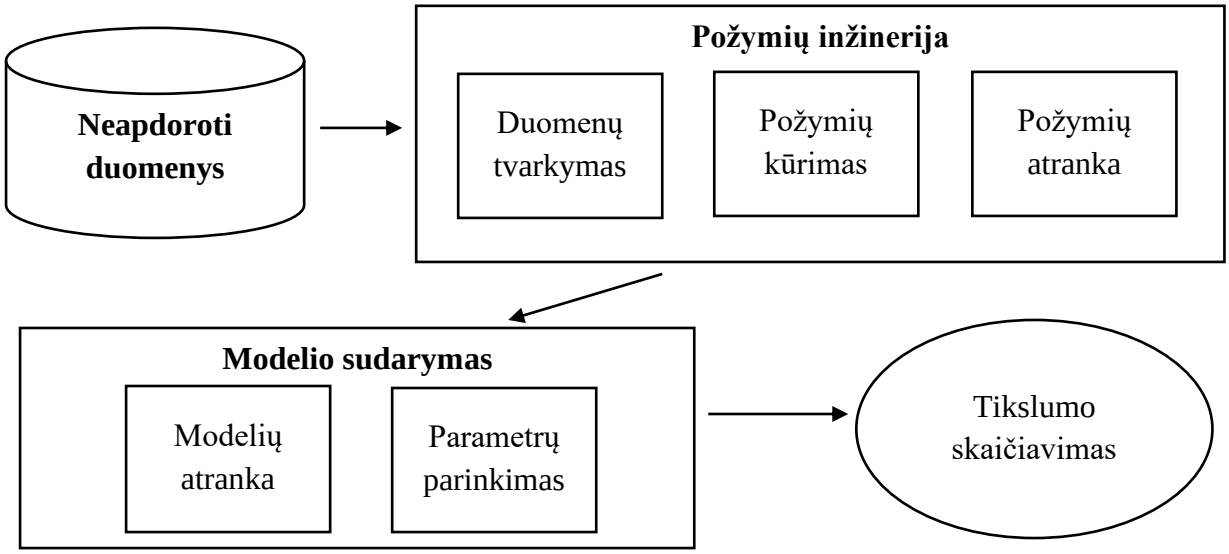
Tsai et al. [25] pateikiama diagrama supaprastintai nurodo informacijos išgavimo procesą naudojantis didžiųjų duomenų analitika (žr. 8 pav.)



8 pav. Informacijos išgavimas iš duomenų bazėse laikomų duomenų (Tsai et al., 2015)

Kaip matoma paveiksle viršuje, dar prieš analizę duomenų bazėse saugomi duomenys turi būti sutvarkomi: tai apima transformavimo, atrankos, agregavimo ir grupavimo procesus, kurie padeda įvesties duomenis panaudoti duomenų analizės procese, kuris naudoja matematinius algoritmus informacijai gauti. Pasitelkiant įvairius matematinius modelius, iš turimų įvesties duomenų galima gauti įžvalgas ir informaciją, kurią interpretuojant gali būti priimami verslo sprendimai. Būtent šie pagrindiniai žingsniai yra mašininio mokymosi (angl. *Machine Learning*) uždavinys. Šis informacijos išgavimo procesas yra pakankamai brangus, kadangi reikalauja aukšto lygio žinių ir patirties bei gali užimti daug laiko. Jordan ir Mitchell [26] teigia, jog mašininis mokymasis kelia klausimą: kaip aprašyti kompiuterio sistemą, kad ši galėtų automatiškai tobulėti iš savo patirties. Mašininis mokymasis apibūdina algoritmą, kuris išmoksta atpažinti duomenis pagal pateiktus pavyzdžius ir gali panaudoti išmoktą informaciją analizuojant naujus, nematytus įrašus. Chen, Song

bei Hu [27] kaip pavyzdį pateikia grafiką, detaliau apibūdinantį mašininio mokymo procesus paveiksle žemiau.



9 pav. Klasikinis mašininio mokymo procesas. Sudaryta remiantis Chen, Song ir Hu [27]

Remiantis pateikta proceso diagrama, požymių inžinerijos sritis yra viena iš pirmųjų mašininiame mokyme ir apima tris dideles šio proceso sritis. Moksliniuose šaltiniuose autoriai sutinka, jog požymių inžinerija yra daugiausiai laiko užtrunkantis procesas [27, 28, 29] mašininio mokymosi uždavinyje. Požymių inžinerijos etape sutvarkomi turimi duomenys, atrenkami svarbiausi kintamieji, atsisakoma nereikšmingos informacijos. Tai pirmoji sritis, kuri dalyvauja sąveikoje su turimais duomenimis. Tinkamai parinkti ir sutvarkyti kintamieji tiesiogiai veikia modelio pateikiamas prognozuojamas kintamųjų reikšmes [27].

Požymių inžinerijos procesas yra pasikartojantis, apimantis daug žingsnių: pateikti duomenys yra sutvarkomi, kuriami nauji požymiai, taip pat filtruojamos persidengiančios, pasikartojančios reikšmės. Siekiama palikti informaciją, reikalingą modeliui apmokinti [27]. Lavrač et al. [28] šias duomenų inžinerijos sritis pateikia kiek plačiau: duomenų tvarkymo žingsnis apima trūkstamų reikmių užpildymą, ribas peržengiančių kriterijų reikšmes, neegzistuojančias duomenų kombinacijas, triukšmą ir neteisingą informaciją duomenyse. Autoriai [28] šiuos procesus apibūdina kaip duomenų transformavimą. Darbe nagrinėjamos dvi duomenų transformavimo sritys: simbolinė (nagrinėjama informacija pateikiama ir pertvarkoma iš simbolinių reikšmių, pvz., žodžių) bei skaitinė (erdvinės duomenų transformacijos: duomenų panašumas nustatomas pagal atstumus tarp kintamųjų, o modeliavimo rezultatai skaičiuojami pagal nagrinėjamų objektų panašumus). Simbolinė analizė rezultatus nagrinėja bei pateikia lengviau suprantamais ir interpretuojamais rezultatais, tuo tarpu skaitinė analizė – sudėtingesnė, apimanti taisyklių mokymąsi, sprendimo medžių sudarymą, ryšiais paremtą loginį duomenų interpretavimą bei indukcinius programavimo algoritmus [28]. Svarbu paminėti, jog simbolinei analizei įprasta duomenis transformuoti į lentelės tipo struktūrą, kurioje viena eilutė atitinka vieną nagrinėjamą objektą, o stulpeliai – šio objekto savybes (atributus). Tuo tarpu duomenų transformavimas į skaitinę erdvę reikalingas tokiems plačiai naudojamiems matematiniams duomenų nagrinėjimo algoritmams kaip neuroniniai tinklai, atraminių vektorių metodas, atsitiktiniai miškai ir kt. Kai kurie iš šių metodų gali naudoti tiek simbolinius, tiek skaitinius duomenis [28]. Skaitiniai duomenys reikalingi statistinio tipo

matematiniams modeliams kurti. Įprastai tokiems modeliams naudojami lentelėje struktūrizuoti duomenys, kur viena eilutė taip pat atitinka vieną duomenų objektą, o stulpeliuose pateikiami nagrinėjamo objekto atributai pateikiami skaitinėmis reikšmėmis.

Kita svarbi nagrinėjamos požymių inžinerijos dalis yra požymių kūrimas. Ši mašininio mokymosi uždavinio dalis yra plačiai nagrinėjama akademinėje literatūroje, tačiau programinio kodo realizacijos yra pakankamai retos. Automatizuotas požymių kūrimas dar vadinamas automatine požymių inžinerija (angl. *automated feature engineering*). Šiame darbe daugiausiai dėmesio skiriama reliacinėms duomenų bazėms tinkamiems automatinės požymių inžinerijos metodams. Tokie metodai leidžia automatiškai apjungti duomenis iš atskirų lentelių duomenų bazėje, sukuriant papildomus požymius, kurie gali būti panaudojami modeliams kurti. Šio proceso automatizavimas padeda išvengti laikui imlaus bei brangaus agregavimo proceso, kuris dažniausiai atliekamas žmogaus. Nagrinėjamoje akademinėje literatūroje sprendžiama požymių kūrimo problema: kaip panaudoti duomenis, esančius reliacinės duomenų bazės lentelėse greitai bei efektyviai. Krogel ir Wrobel [30] teigia, jog paprastai verslo duomenų bazės gali turėti lentelę, kurioje saugoma tūkstančiai kiekvieno kliento transakcijų, ir tokių lentelių, apjungtų reliaciniais ryšiais, gali būti ne viena, todėl bendras analizuojamų duomenų kiekis auga nepaliaujamai ir gali siekti milijonus įrašų. Kanter ir Veeramachaneni [31] teigia, jog dažniausiai duomenų analizė atliekama turint struktūrizuotą, reliaciniais ryšiais paremtą duomenų rinkinį, kuris aprašo tam tikrą verslo procesą (tai gali būti ir pardavimų duomenys, banko sąskaitų transakcijos ir kt.). Turint šiuos duomenis, siekiama spręsti tam tikrą verslo problemą: siekiama analizuoti pardavimus per nuolaidų sezoną, numatyti, ar banko klientui verta duoti paskolą ir pan. Turint duomenis bei problemą, duomenų mokslininkas pirmiausia turi konstruoti ir atrinkti požymius, kurie daro įtaką problemai. Moksliniame darbe teigiama, jog paprastai pirmiausia gali būti panaudojami abstraktūs požymiai (pvz. lytis, amžius, miestas, išlaidų suma, prekių krepšelio dydis, pirkti produktai), tačiau vėliau pereinama prie labiau specializuotų ir intuityvių požymių kūrimo: autoriai teigia, jog požymiai gali būti kuriami ir nujaučiant, kas galimai daro įtaką nagrinėjamai problemai, pvz., pirkimo dažnis, sezoniškumas, dienos metas ir pan. Tokie požymiai gali būti konstruojami iš įvesties duomenų, naudojantis informacija, pasiekiama skirtingose lentelėse, susijusiose reliaciniais ryšiais. Autoriai [31] teigia, jog požymių kūrimas gali būti automatizuojamas analizuojant teksto, vaizdų bei signalų duomenų tipus, tačiau reliaciniais ryšiais apjungti duomenys, analizuojantys verslo procesus ir žmonių elgsenos principus, yra pakankamai sudėtingi, todėl požymių kūrimas tokiems duomenimis vis dar yra sudėtingas, intuityvus bei laikui imlus procesas, atliekamas žmogaus. Vis dėl to egzistuoja nemažai šią problemą nagrinėjančių mokslininkų, kurie taiko įvairius matematinius metodus, galinčius automatizuoti požymių inžinerijos dalį reliaciniais ryšiais apjungtuose duomenyse (žr. 2 lent.). Lavrač et al. [28] šį procesą įvardina kaip plokštinimą (angl. *propositionalisation*) – tai reliacinės duomenų bazės lentelių (įrašų bei jų atributų) pertvarkymas į vieną lentelę.

2 lentelė. Akademinėje literatūroje aptariami plokštinimo metodai

Metodas	**Metai**	**Citavimų skaičius Google Scholar[6]**	**Citavimų skaičius ResearchGate[7]**
Reinforcement learning [27]	2020	30	22
Wordification [31]	2015	35	294
AutoFeatures [29]	2020	-	-
RELAGGS [28, 32]	2003, 2020	30, 35	20, 26
RSD [33]	2003	184	127
SINUS [33]	2003	184	127
DAFEE [34]	2020	2	1
Cognito [35]	2016	146	129
DFS [31]	2015	401	294
Featuretools [36]	2018	-	-
ExploreKit [37]	2016	180	158
FDRL [38]	2020	3	4
CLAMF [39]	2007	42	34
MVC [40]	2008	5	6

Metodų gausa rodo, jog šioje srityje tobulėjama, kadangi, kaip jau minėta anksčiau, požymių inžinerijos proceso automatizavimas yra svarbi dalis mašininio mokymo uždavinio tobulinimo procese.

Išnagrinėjus mokslinius šaltinius, pastebima, kad šiuo metu skiriamas didelis dėmesys organizacijų skaitmenizavimui. Daugelyje nagrinėtų straipsnių teigiama, kad egzistuoja būtinybė keistis ir prisiderinti prie vykstančių technologinių pokyčių siekiant išlikti konkurencingiems rinkoje, o pandemijos patirtis parodė, kokios trapios verslo sistemos yra atsitikus nenumatytoms pasaulinio lygio katastrofoms. Vienas iš nagrinėtų informacinių technologijų komponentas – duomenų bazės – jau dabar turi didelę valdymo sistemų įvairovę, pritaikytą įvairiems verslo procesų poreikiams. Toliau darbe analizuojamos reliacinės duomenų bazių valdymo sistemos, kadangi šios technologijos yra populiariausias ir labiausiai naudojamas įrankis verslo organizacijose. Nagrinėtas didžiųjų duomenų analitikos konceptas parodė, jog nors didelė duomenų apimtis gali būti naudinga procesams, tačiau duomenų konvertavimas į naudingą informaciją yra sudėtingas ir daug patirties bei žinių reikalaujantis procesas. Nagrinėjant mašininio mokymo principų taikymą, atlikta literatūros analizė parodė, jog egzistuoja nemažai mašininio mokymo algoritmų, skirtų automatizuoti požymių inžinerijos žingsnius mašininio mokymo uždaviniams reliacinėse duomenų bazėse, taip palengvinant ir taupant organizacijos išteklius duomenų analizei atlikti ir verslui naudingai informacijai atrasti.

[6] https://scholar.google.com/schhp?hl=en&as_sdt=0,5

[7] https://www.researchgate.net

2. Tyrimų metodai

Šioje darbo dalyje aptariamos tyrimui naudojamos sistemos, algoritmai bei jų pavyzdžiai, palyginimui naudoti modeliai bei jų tikslumo vertinimo kriterijai, taip pat aprašomi metodai, taikyti duomenų tvarkymui ir paruošimui.

2.1. Algoritmų realizacija atviro kodo sistemose

Akademinėje literatūroje aptariami įvairūs algoritmai, galintys automatizuoti reliacinėse duomenų bazėse esančių duomenų požymių inžineriją. Šiame darbe aptariamos dvi nemokamos sistemos, kurios realizuoja keletą iš mokslinėje literatūroje aprašomų požymių inžinerijos algoritmų.

2.1.1. Featuretools

Featuretools Python paketas[8], automatizuojantis požymių kūrimą. Sistema veikia Python programavimo kalba, yra skirta laiko eilučių bei reliacinių duomenų rinkinių požymių kūrimui. Featuretools naudoja DFS (angl. *Deep Feature Synthesis*) algoritmą požymių kūrimui [31]. Sistemos internetiniame puslapyje teigiama, jog Featuretools gali būti naudojamas greta jau naudojamų įrankių mašininiam mokymuisi, kadangi Featuretools gali būti naudojamas įsidiegus reikalingą Python paketą. Papildomai Featuretools naudoja *Woodworks* Python paketą duomenų tipams parinkti, bei *Graphwiz* paketą, padedantį vizualizuoti lentelių ryšių struktūrą.

Šiame darbe analizuojama Featuretools sistema visiems tiriamiems duomenų rinkiniams naudota toliau pateiktais žingsniais:

1. Sukuriamas subjektų rinkinys (angl. *Entity set*), kuriame saugoma informacija apie lenteles, duomenų tipą bei reliacinius ryšius tarp lentelių (žr. 10 pav.).

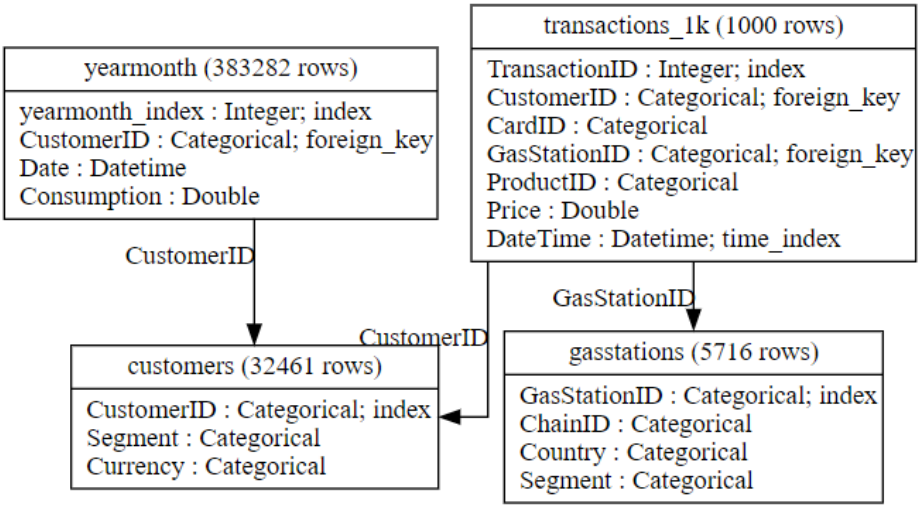
2. Subjektų rinkinyje detaliai nurodoma informacija apie kiekvieną jame egzistuojančią reliacinės duomenų bazės lentelę: indekso laukas, datos laukas (jei egzistuoja), atributų duomenų tipai.

3. Įvardinami reliaciniai ryšiai tarp visų lentelių, esančių subjektų rinkinio viduje. Nurodyti ryšiai įrašomi į subjektų rinkinį.

4. Pagal subjektų rinkinyje aprašytus duomenis atliekamas požymių kūrimas DFS algoritmu: tam naudojamos pasirinktos agregavimo ir transformavimo taisyklės, lentelė, kurioje yra priklausomas kintamasis, nereikalingi duomenų stulpeliai ir kt.

Duomenų agregavimui bei transformavimui dažniausiai naudojamos numatytosios (angl. *default*) funkcijos. Numatytosios agregavimo funkcijos yra suma, standartinis nuokrypis, maksimalios bei minimalios reikšmės, skaičius, unikalių reikšmių skaičius, moda; numatytosios duomenų transformavimo funkcijos yra diena, mėnesis, metai, savaitės diena, žodžių skaičius, ženklų skaičius. Tiek transformavimo, tiek agregavimo funkcijų yra įvairių, todėl galima pasirinkti ir kitas funkcijas, tinkamas tam tikram duomenų rinkiniui ar problemai nagrinėti.

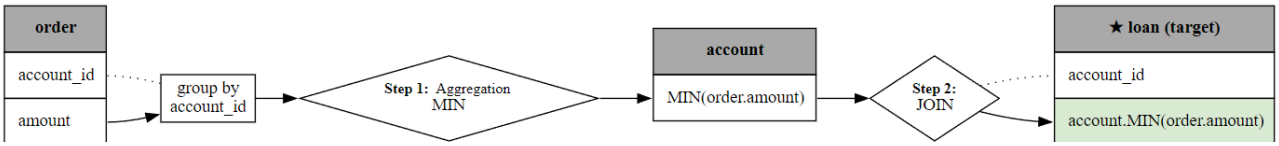
[8] https://www.featuretools.com



10 pav. Financial duomenų rinkinio subjektų rinkinys Featuretools

Remiantis sudarytų subjektų rinkiniu Featuretools sukuria lentelę su papildomais požymiais – požymių matricą. Šioje lentelėje yra visi įrašai iš lentelės su priklausomu kintamuoju (sukurtos lentelės įrašų skaičius lygus pagrindinės lentelės įrašų skaičiui) bei visi požymiai, sukurti panaudojus reliaciniais ryšiais apjungtas lenteles ir nurodytas agregavimo bei transformavimo funkcijas.

Siekiant geriau suprasti sukurtus požymius, gali būti panaudojamos Featuretools funkcijos *graph_feature* bei *describe_feature*. Grafinis vaizdavimas nurodo požymio kūrimo kelią. Paveiksle žemiau vaizduojama, kokie žingsniai reikalingi norint sukurti požymį *account.MIN(order.amount)* (žr. 11 pav.). Taip galima pamatyti panaudotas agregavimo funkcijas bei lenteles.



11 pav. Požymio kūrimo pavaizdavimas grafiniu būdu, panaudojus Featuretools funkciją *graph_feature*

Panaudojus požymio aprašymo funkciją (*describe_feature*), pateikiamas detalus požymio aprašymas:

'The minimum of the "amount" of all instances of "order" for each "account_id" in "account" for the instance of "account" associated with this instance of "loan".'

Lentelė su papildomais požymiais dar nėra paruošta modeliui kurti. Šiame darbe Featuretools sudaryta duomenų rinkinio požymių matrica paruošiama modelio apmokymui pagal žemiau pateiktą veiksmų planą:

- kategorinių kintamųjų kodavimas;
- trūkstamų reikšmių tikrinimas, požymių, turinčių didelę dalį trūkstamų reikšmių, šalinimas;
- koreliuojančių požymių šalinimas.

Taip pat pastebima, jog sukurti požymiai gali turėti ryšį su priklausomu kintamuoju (požymis sukurtas agreguojant priklausomo kintamojo reikšmes). Siekiant išvengti šios problemos, tokie

požymiai pašalinami iš duomenų lentelės. Featuretools pakete integruoti keletas pagrindinių įrankių, padedančių paruošti sukurtą požymių matricą:

- *encode_features* – funkcija, skirta kategorinių kintamųjų kodavimui;
- *selection.remove_highly_null_features* – funkcija, skirta pašalinti požymiams, kai tuščios reikšmės stulpelyje viršija nurodytą slenkstį;
- *selection.remove_low_information_features* – funkcija, kuri pašalina požymius, turinčius tik vieną unikalią reikšmę.

Gauta sutvarkyta požymių matrica yra duomenų lentelės tipo. Tokią lentelę paprasta pertvarkyti į norimą formą, reikalingą pasirinkto modelio kūrimui ir vertinimui. Sudaryta požymių lentelė tyrimo metu atsitiktinai dalinama į apmokymo ir testavimo imtis panaudojant Python *sklearn* paketo funkciją *train_test_split*. Paruošta testavimo imtis lygi 25 %, apmokymo – 75 % pradinės duomenų imties.

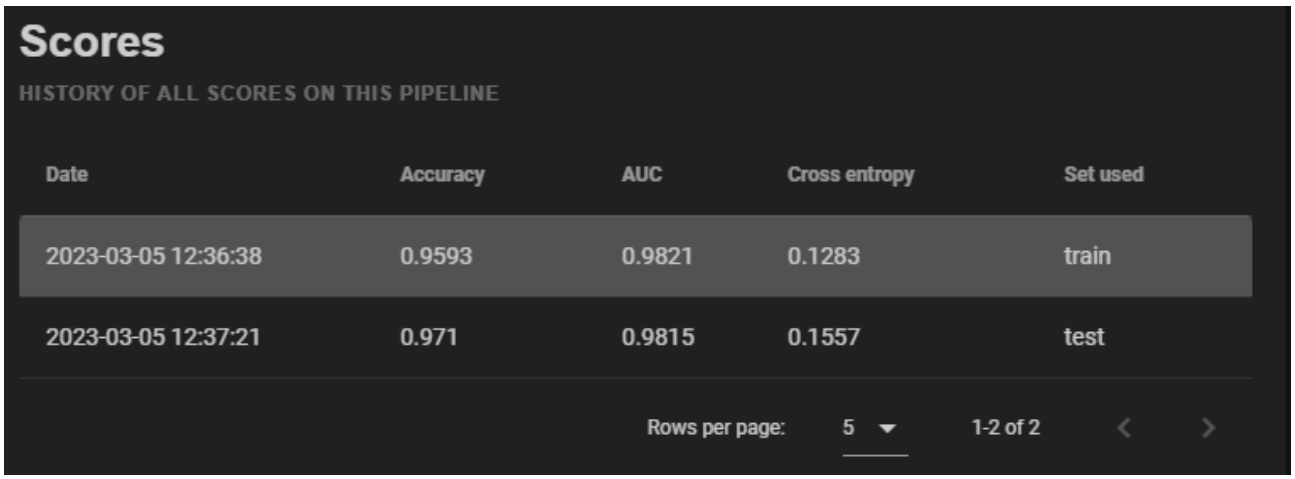
2.1.2. getML

Sistemos kūrėjų tinklapyje[9] getML apibūdinama kaip „inovatyvus įrankis“, skirtas automatizuoti duomenų mokslo projektams. Teigiama, jog sistema automatizuoja mašininio mokymosi procesą nuo duomenų įkėlimo iki apmokintų modelių panaudojimo verslo sistemose. Kūrėjai teigia, jog pagrindinis getML sistemos tikslas yra automatizuoti sudėtingą, laikui imlų požymių inžinerijos procesą. Šiam tikslui pasiekti sistemoje integruoti 4 skirtingi algoritmai, kurie, panaudojus įvairias matematines funkcijas, gali automatizuoti požymių inžineriją reliaciniuose duomenų rinkiniuose.

getML naudoja didelio našumo C++ programavimo kalba parašytą variklį, kuriame atliekami skaičiavimai. Kompiuteryje įdiegtoje sistemoje getML darbas vyksta naudojant Python paketą getML Jupyter užrašinėje (angl. *Jupyter notebook*). C++ bei Python programavimo kalbų apjungimas su sistema leidžia dirbti efektyviai, nereikalauja aukšto lygio žinių. Žemiau pateikiami pagrindiniai darbo su getML žingsniai, kurie pritaikomi šiame darbe vykdytiems tyrimams.

Projekto sukūrimas bei duomenų įkėlimas. getML sistemoje esanti vartotojo sąsaja leidžia pasirinkti, stebėti ir analizuoti keletą skirtingų projektų. Projekto informacija, duomenų lentelės, sukurti požymiai bei modeliai gali būti pasiekiami bet kuriuo metu. Dirbti galima pradėti nuo bet kurios projekto dalies: duomenų tvarkymo, požymių kūrimo, modelio parametrų derinimo ir kt. Žemiau paveiksle pateikiama vartotojo sąsajos dalis, kurioje galima peržiūrėti projektą Financial. Paveiksle pateikiami apmokyto modelio rezultatai, kai požymiai sukurti Multirel algoritmu (žr. 12 pav.).

[9] https://getml.com



12 pav. Multirel algoritmu sukurtais požymiais apmokinto modelio įverčiai getML vartotojo sąsajoje

Duomenų paruošimas getML sistemoje. getML sistema naudoja Python paketą getML, kurio pagalba atliekama duomenų transformacija, tvarkymas ir visi tolimesni skaičiavimai. Darbui su getML sistema svarbus duomenų paruošimo žingsnis. Norint kuo efektyviau naudotis sistemos galimybėmis, svarbu parinkti teisingus nagrinėjamų duomenų tipus: kategorinius, skaitinius, tekstinius laukus, laiko žymes ir kt. Viena svarbiausių rolių – raktai (getML sistemoje vadinami *join_key*), kurie žymi, kaip tarpusavyje jungiasi reliacinės duomenų bazės lentelės. Čia taip pat nurodoma priklausomo kintamojo rolė (angl. *target*). Skaičiavimams nereikalingi atributai gali būti įvardiname role „nenaudojamas“ (angl. *unused*) – *unused_float* ar *unused_string*. Tokia role pažymėti kintamieji nenaudojami požymių kūrimo procese. Raktų (angl. *join_key*) bei priklausomo kintamojo (angl. *target*) rolės taip pat nėra naudojamos požymių kūrime.

Nagrinėjamų duomenų sistemos aprašymas – duomenų modelis. Šiame žingsnyje getML sistemai nurodomas reliacinis duomenų modelis, kuriame aprašoma, kuri lentelė turi priklausomo kintamojo informaciją ir kurios lentelės yra papildomos. Anksčiau minėtieji rakto laukai nurodo, kaip lentelės jungiasi tarpusavyje, taip pat gali būti nurodomi ryšio tipai (vienas-su-vienu (angl. *one-to-one*), daug-su-daug (angl. *many-to-many*) ir kt.). Sukurtas duomenų modelis taip pat vaizduojamas getML vartotojo sąsajoje (žr. 13 pav.).

Apmokomi požymių kūrimo bei modelio algoritmai. Duomenys padalinami į apmokymo, testavimo bei validavimo imtis pasinaudojus funkcija *getml.data.split.random* ir nurodžius padalinimo proporcijas, kurios visiems nagrinėtiems duomenų rinkiniams pasirinktos vienodos: 80 % imties skiriama apmokymui, po 10 % - validavimui bei testavimui. Taip pat nurodomi parametrai tiek požymių kūrimo algoritmams, tiek XGBoost modeliui. Šiame darbe parametrų pasirinkimas netaikomas požymių kūrimo algoritmams – naudojami tik numatytieji parametrai. Šiame žingsnyje parenkamos ir papildomos duomenų tvarkymo operacijos (angl. *preprocessing*), kurios nėra susijusios su reliacine duomenų struktūra, tačiau yra pritaikytos atskiroms duomenų lentelėms tvarkyti. getML galima naudoti šias tikslumą gerinančias funkcijas: duomenų sezoniškumo nustatymą, kategorinių kintamųjų transformavimą į skaitines reikšmes, laisvos formos teksto tvarkymą, trūkstamų reikšmių užpildymą ir kt. Nustačius norimus parametrus, anksčiau sudarytas modelis yra tikrinamas (funkcija c*heck*) – teikiami siūlymai, kaip pagerinti sudarytą duomenų modelį bei pateikiamos

13 pav. Financial duomenų rinkinio reliacinių ryšių modelis getML vartotojo sąsajoje

klaidos, jei egzistuoja. Po tikrinimo sukuriami požymiai ir apmokomas modelis (funkcija *fit*). Apmokytam modeliui siūloma automatiškai derinti modelio parametrus pasinaudojant getML funkcija *tune_predictors*. Sistema naudoja Gauso Hiperparametrų paieškos metodą atrinkti geriausiems modelio parametrams ir taip pagerinti modelio tikslumą. Šis metodas buvo taikytas beveik visiems duomenų rinkinių modelių parametrams derinti.

Modelio įvertinimas. getML sistemoje šis procesas atliekamas panaudojus įvertinimo (funkcija s*core*) komandą. Čia apskaičiuojamas modelio tikslumas testavimo imties duomenims pagal getML sistemoje egzistuojančius modelių įvertinimo parametrus: klasifikacijos problemoms tai tikslumas (angl. *Accuracy*), AUC bei kryžminė entropija (angl. *Cross-entropy*), regresijos uždaviniams – determinacijos koeficientas (angl. *R-Squared*), MAE bei RMSE.

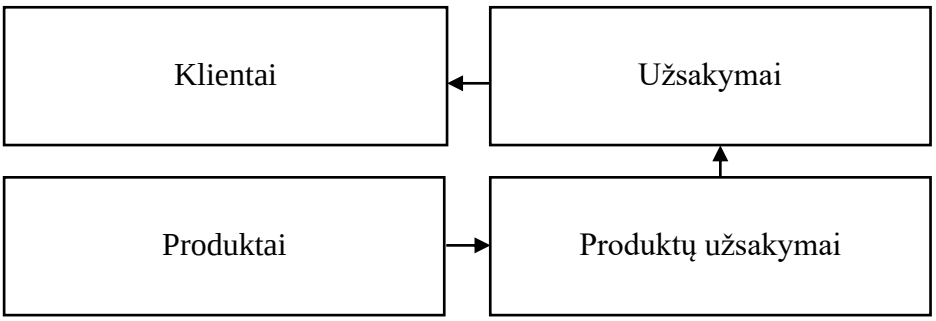
2.2. Automatinės požymių inžinerijos algoritmai

2.2.1. DFS algoritmas

DFS, arba *Deep Feature Synthesis* yra algoritmas, padedantis automatiškai generuoti požymius naudojantis reliacinės duomenų bazės duomenimis [31]. Algoritmas seka duomenų bazėje egzistuojančiais ryšiais iki pagrindinio atributo. Panaudojant matematines funkcijas pagal nagrinėtus ryšius sudaromas galutinis nagrinėjamo objekto požymis. Sudedant atliktus matematinius pertvarkymus iš eilės, galima teigti, jog kiekvienas gautas duomenų požymis turi tam tikrą gylį *d*. Būtent iš to kilo algoritmo pavadinimas, kuris nurodo, jog atliekama gili (angl. *deep*) požymių sintezė.

Įvesties duomenų rinkinys, pateikiamas algoritmui, yra lentelės ir duomenų rinkiniai, susiję tarpusavyje. Reikalingas unikalus raktas kiekvienai algoritmo naudojamai lentelei. Taip pat yra galimybė panaudoti susijusios lentelės unikalų ID raktą. Kiekvienas duomenų rinkinio objektas gali turėti šiuos duomenų tipus: skaitinis, kategorinis kintamasis, laiko žymės bei laisvojo teksto tipas. Toliau pagal turimų objektų bei lentelių skaičių, taip pat pagal ryšių kiekį apibrėžiamos naudojamos matematinės funkcijos. Šios funkcijos yra apibrėžiamos keletu lygių: objekto lygiu (angl. *entity*

level) bei ryšio lygiu (angl. *relational level)*. Objekto lygiu atliekami matematiniai pertvarkymai nurodo kiekvieno skaičiuojamo objekto informaciją: pvz., kategorinis kintamasis pakeičiamas į nustatytą skaitinę reikšmę, arba skaitinė reikšmė suapvalinama iki tam tikro lygio. Kiti pertvarkymai gali būti laiko žymės pakeitimas į savaitės dieną, metus, mėnesį ar valandą. Ryšio lygiu atliekami pertvarkymai atliekami analizuojant du susijusius objektus. Galimi du ryšio tipai – į priekį (angl. *forward*) bei atgal (angl. *backward*). Remiantis pateikiamu pavyzdžiu (žr. 14 pav.), į priekį einantis ryšys nurodo, jog analizuojant Klientų bei Užsakymų lenteles, objektas Klientų lentelėje turi *forward* ryšio tipą, kadangi kiekvienas Užsakymų lentelės įrašas gal turėti tik vieną įrašą Klientų lentelėje. Tokie požymiai vadinami tiesioginiais (angl. *direct*). Atvirkščias ryšio pavyzdys – *backwards* tipo – daug Užsakymų lentelės įrašų siejasi tik su vienu įrašu Klientų lentelėje. Tokie požymiai vadinami ryšių požymiais (angl. *relational*). Skaičiavimų funkcijos pavyzdžiai yra suma, minimali vertė, skaičius, maksimali vertė ir kt.



14 pav. Pavyzdinė reliacinės duomenų bazės struktūra (Kanter, Veeramachaneni, 2015)

DFS algoritmas, remdamasis reliaciniais ryšiais, egzistuojančiais tarp aukščiau pateiktų duomenų bazės lentelių, gali sukurti požymį, kuris nurodo kiekvieno kliento vidutinį užsakymo dydį. Pirmiausia apskaičiuojamas tiesioginis požymis (*dfeat*) ir produkto kaina pridedama prie „Produktų užsakymai“ lentelės. Tuomet sukuriamas reliacinis požymis (*rfeat*) lentelėje „Užsakymai“ – apskaičiuojama užsakymo suma panaudojant SUM funkciją: apskaičiuojama susijusių įrašų suma remiantis informacija iš „Produktų užsakymai“ lentelės. Galiausiai sukuriamas trečiasis požymis, kuris nurodo vidutinę kliento krepšelio sumą remiantis anksčiau sukurta informacija (žr. 3 lent.).

3 lentelė. DFS algoritmu sukurtas požymis

KlientoID	**Amžius**	**AVG(Užsakymai.SUM(Produktas.Kaina)**
1	...	...
2	37	127 Eur
3	...	...

DFS algoritmu sugeneruotas požymis, parašytas SQL kalba, pateikiamas žemiau. Sugeneruotas požymis čia nurodo donoro paaukotą sumą:

```
UPDATE Donors_1 target_table
   LEFT JOIN(SELECT donor_acctid, SUM(amount) as val
               FROM Donations rt
               GROUP BY donor_acctid) b
   ON b.donor_acctid = target_table.donor_acctid
SET target_table.Donors_1_100 = b.val
WHERE b.donor_accid = target_table.donor_acctid
```

2.2.2. FastProp algoritmas

Algoritmas, naudojamas getML sistemoje. Tai sistemos kūrėjų sukurtas algoritmas, kuris remiasi plokštinimo principais. Pasak getML kūrėjų, tai greitas bei efektyvus algoritmas, kuris panaudoja agregavimu paremtas operacijas ir transformuoja reliaciniais ryšiais sujungtas lenteles į vieną lentelę. FastProp algoritmas veikia greitai ir sudaro požymius pagal paprastas nesąlygines agregavimo operacijas.

FastProp algoritmo metodu sukurtas paprastas požymis pateikiamas žemiau:

```
CREATE TABLE FEATURE_1 AS
SELECT MAX(t2.column) AS feature_1, t1.rowid AS "rownum"
FROM "population" t1
   LEFT JOIN "peripheral" t2
   ON t1.join_key = t2.join_key
WHERE t2.time_stamp <= t1.time_stamp
ORDER BY t2.time_stamp
GROUP BY t1.rownum, t1.join_key, t1.time_stamp
```

Čia SQL kalba pažymėtas paprastas duomenų agregavimas: pirmosios lentelės duomenys apjungiami su antrosios lentelės duomenimis pagal laiko sąlygą, panaudojant GROUP BY agregavimo funkciją. Pasak getML sistemos kūrėjų, tai vienas greičiausių algoritmų požymių kūrimui.

2.2.3. Multirel algoritmas

Tai 2003 metais populiaraus akademinėje literatūroje *Multi-relational Decision Tree Learning* [41] algoritmo variacija, pritaikyta naudoti getML sistemoje. Algoritmas sudaro objektinę funkciją, kuri skaičiuoja sudaryto požymio kokybę pagal ryšį su nurodytu priklausomu kintamuoju, tuomet pasirenka agregavimo metodą ir stulpelius, kurie gali būti agreguojami, ir tikrina gautus rezultatus. Paliekami tik tie požymiai, kurie daro didžiausią įtaką priklausomam kintamajam; skaičiavimai kartojami tol, kol neberandamas funkcijos pagerėjimas ar pasiekiamas sustabdymo kriterijus.

getML kūrėjai teigia, jog algoritmas, nors ir populiarus akademinėje literatūroje, nebuvo pritaikytas versle dėl efektyvios realizacijos trūkumo. Akademinėje literatūroje nagrinėjama algoritmo realizacija pritaikoma reliacinių ryšių duomenų bazėje, pvz. MySQL, tačiau getML kūrėjai pastebi, jog tokie bandymai sukelia daug papildomų nereikalingų skaičiavimų duomenų bazės valdymo sistemoje. Kaip pavyzdys gali būti aptariamas požymis: paskutinių X dienų transakcijų skaičius. Kiekviena skirtinga X reikšmė turi būti perskaičiuojama iš naujo, kas kartą nuskaitant tuos pačius duomenis. Dėl šios priežasties požymių sudarymas duomenų bazės valdymo sistemoje daro didelę įtaką serverio pajėgumui ir gali paveikti kasdienes verslo operacijas. getML kūrėjai pateikia patobulintą Multirel algoritmo versiją: čia minimizuojamos nereikalingos iteracijos palaipsniui keičiant X reikšmę. Duomenų bazė atkuriama panaudojant C++ programavimo kalbą, ir anksčiau minėto X skaičiaus parinkimas atliekamas palaipsniui didinat reikšmę, todėl nereikia kas kartą perskaičiuoti visų duomenų. Pvz., skaičiaus (COUNT) agregavimo operacijose reikšmių keitimas yra tiesioginis: jei požymis anksčiau nebuvo reikšmingas, bet dabar yra – X reikšmę galima padidinti vienu vienetu. Jei padidinus X reikšmę, požymis tampa nebereikšmingu, tuomet X reikšmė mažinama vienu vienetu.

Algoritmas paremtas medžio tipo struktūra, todėl galima panaudoti *ensemble* algoritmus, kurie gerai veikia su ne reliaciniais ryšiais paremtais sprendimų medžiais – *bagging* bei *gradient boosting*.

Bagging čia padeda parinkti pavyzdžius iš pagrindinės duomenų lentelės; *gradient boosting* metodas naudojamas apskaičiuoti paklaidoms iš anksčiau sukurtų požymių. Naujai kuriami požymiai kompensuoja ir mažina ankstesnes paklaidas. Pavyzdinis Multirel algoritmu sukurtas požymis, parašytas SQL kalba, pateikiamas žemiau.

```
CREATE TABLE FEATURE_1 AS
SELECT COUNT(*) AS feature_1, t1.join_key, t1.time_stamp
FROM (SELECT *, ROW_NUMBER() OVER (ORDER BY join_key, time_stamp ASC) AS rownum
FROM POPULATION) t1
   LEFT JOIN PERIPHERAL t2
   ON t1.join_key = t2.join_key
WHERE (( t1.time_stamp - t2.time_stamp <= 0.499624))
AND t2.time_stamp <= t1.time_stamp
GROUP BY t1.rownum, t1.join_key, t1.time_stamp
```

Čia galima pastebėti kiek daugiau skirtingų skaičiavimų: tikrinamas eilučių skaičius pagal tam tikrą laikotarpį.

2.2.4. Relboost algoritmas

Relboost algoritmas, naudojamas getML programoje, yra XGBoost algoritmo versija, pritaikyta požymiams kurti. Pagrindinis Relboost bei Multirel algoritmų skirtumas yra tai, jog Multirel agreguoja stulpelius, o Relboost agreguoja mokymosi svorius.

Anksčiau aptartas Multirel algoritmas turi problemų dėl skaičiavimų gausos: požymių kūrimui Multirel gali agreguoti (ar skaičiavimams naudoti) visus atributus. Tai sukuria daugybę požymių, kurių tikrinimas užima dar daugiau laiko. Tai gali būti problema, jei turima duomenų bazė turi lenteles su daug skirtingų požymių (atributų). Tai nėra problema, specifiška vien Multirel algoritmui; su tokia pačia problema susidurti galima ir kuriant požymius ranka. Relboost algoritmas čia veikia kiek kitaip: stulpeliai čia panaudojami tik sąlygoms ir nėra agreguojami. Relboost algoritmas agreguoja mokymosi svorius, todėl procesas yra greitesnis. Tačiau šis metodas turi ir trūkumų: požymiai, sugeneruoti Relboost algoritmo, yra ne tokie intuityvūs ir labiau skiriasi nuo ranka generuotų, agregavimu paremtų požymių. Kitas trūkumas yra Relboost pritaikymas keletui skirtingų priklausomų kintamųjų klasių, kadangi algoritmas turi išmokti skirtingas taisykles ir svorius kiekvienai klasei.

Relboost algoritmo sukurto požymio pavyzdys, parašytas SQL kalba, pateikiamas žemiau.

```
CREATE TABLE FEATURE_1 AS
SELECT SUM(
CASE
     WHEN (t1.time_stamp - t2.time_stamp > 0.499624) THEN 0.0
     WHEN (t1.time_stamp - t2.time_stamp <= 0.499624 OR t1.time_stamp IS NULL OR
t2.time_stamp IS NULL) THEN 1.0
     ELSE NULL
END
) AS feature_1, t1.join_key, t1.time_stamp
FROM (SELECT *, ROW_NUMBER() OVER (ORDER BY join_key, time_stamp ASC) AS rownum FROM
POPULATION) t1
   LEFT JOIN PERIPHERAL t2
   ON t1.join_key = t2.join_key
WHERE t2.time_stamp <= t1.time_stamp
GROUP BY t1.rownum, t1.join_key, t1.time_stamp
```

2.2.5. RelMT algoritmas

getML RelMT algoritmas yra tiesinių modelių medžių realizacija reliaciniais ryšiais paremtam duomenų rinkiniui. Šis algoritmas apjungia tiesinio modelio privalumus (paprastas interpretavimas, tiesinio ryšio egzistavimas) su medžiais paremtais algoritmais (privalumai čia yra efektyvumas bei galimybė aptikti netiesinius ryšius). RelMT algoritmas tinkamas laiko eilučių problemoms, kadangi laiko eilutėse dažnai esanti autoregresija gali būti aprašoma tiesiniais modeliais. Tik medžiais paremti algoritmai dažniausiai tokių ryšių aptikti negali.

RelMT algoritmas gali būti panašus į Relboost: kiekviena medžio šaka šakojasi tuomet, kai yra tenkinama tam tikra sąlyga. Tačiau kai Relboost algoritmas apskaičiuoja šakos svorius, RelMT apmoko tiesinį modelį, kuris gali įvertinti tiesinius ryšius tarp pagrindinės duomenų lentelės ir papildomų duomenų lentelių.

Pavyzdinis RelMT sukurtas požymis, parašytas SQL kalba, pateikiamas žemiau.

```
CREATE TABLE FEATURE_1 AS
SELECT SUM(
CASE
    WHEN ( t1.time_stamp - t2.time_stamp > 0.499624) THEN
COALESCE (t1.time_stamp - julianday( '1970-01-01' ) - 17202.004, 0.0 ) * -122.121 +
COALESCE( t2.column - 3301.156, 0.0 ) * -0.003
    WHEN (t1.time_stamp - t2.time_stamp <= 0.499624 OR t1.time_stamp IS NULL OR
t2.time_stamp IS NULL) THEN
COALESCE( t1.time_stamp - julianday( '1970-01-01' ) - 17202.004, 0.0 ) * 3.654 +
COALESCE( t2.column - 3301.156, 0.0 ) * -1.824 + -8.720
     ELSE NULL
END) AS feature_1, t1.join_key, t1.time_stamp
FROM (SELECT *,ROW_NUMBER() OVER (ORDER BY join_key, time_stamp ASC) AS rownum FROM
POPULATION) t1
   LEFT JOIN PERIPHERAL t2
   ON t1.join_key = t2.join_key
WHERE t2.time_stamp <= t1.time_stamp
GROUP BY t1.rownum, t1.join_key, t1.time_stamp
```

2.3. Modelis

Siekiant palyginti tiek naudojamus algoritmus, tiek sistemų veikimą tarpusavyje, pasirinktas modelis XGBoost, kadangi šio modelio realizacija egzistuoja getML sistemoje kaip viena iš šios dalių.

XGBoost modelis laikomas itin greitu bei efektyviu modeliu, tinkamu tiek klasifikacijos, tiek regresijos problemoms spręsti. Modelio kūrėjų [42] teigimu, šis modelis tinkamas dideliems duomenų kiekiams apdoroti, yra spartus bei gerai apsaugotas nuo persimokymo.

Remiantis XGBoost modelio veikimo principais, pateikiamais XGBoost internetiniame tinklapyje[10], XGBoost modelis veikia sprendimų medžių pagrindu. Trumpai tariant, sukuriama sprendimų medžių eilė, kurioje kiekvienas medis sukuriamas siekiant ištaisyti prieš tai buvusio medžio klaidas. Modelis pradeda kurti sprendimų medį su viena šaka; tuomet apskaičiuojamos liekamosios paklaidos kiekvienam duomenų rinkinio įrašui. Toliau sukuriamas antras sprendimų medis, kurio tikslas – ištaisyti ankstesnio medžio klaidas. Tuomet dviejų turimų sprendimų medžių prognozės

[10] https://xgboost.readthedocs.io/en/latest/tutorials/model.html

apjungiamos, ir procesas kartojamas – kiekvienas tolimesnis sprendimų medis siekia taisyti klaidas. Galutinis modelio rezultatas – visų sprendimų medžių prognozių suma. XGBoost modelis gali būti užrašomas formule (1):

$$\hat{y}_1 = \sum_{k=1}^{K} f_k(x_1), f_k \in \mathrm{F}\,; \tag{1}$$

čia K yra medžių skaičius, f_k yra funkcija erdvėje F, o F yra visi galimi CART rinkiniai.

XGBoost naudoja gradientinio nusileidimo (angl. *Gradient descent*) metodą derinti kiekvieno sprendimų medžio parametrams. Čia apskaičiuojamas nuostolio funkcijos gradientas modelio parametrų atžvilgiu; parametrai atnaujinami neigiamo gradiento kryptimi.

XGBoost taip pat naudoja L1 (angl. *Lasso regularisation*) bei L2 (angl. *Ridge regression*) reguliarizacijos technikas minimizuoti modelio persimokymui ir modelio našumui gerinti. Formulėje (2) pateikiama modelio naudojama tikslo funkcija (angl. *objective function*) bei reguliarizacija:

$$obj(\theta) = L(\theta) + \Omega(\theta); \tag{2}$$

čia L yra apmokymo nuostolio funkcija (angl. *Loss function*), o Ω yra reguliarizacijos taisyklė (angl. *Regularisation term*).

Tikslo funkcija, kurią reikia optimizuoti, gali būti užrašoma taip (3):

$$obj(\theta) = \sum_{i}^{n} l(y_1, \hat{y}_1) + \sum_{k=1}^{K} \omega(f_k); \tag{3}$$

čia $\omega(f_k)$ yra medžio f_k kompleksiškumas.

Norint apskaičiuoti kiekvieno medžio parametrus, XGBoost algoritmas taiso anksčiau apmokintus medžius ir prideda naujų. Kiekvienos prognozės reikšmė gali būti užrašoma žingsnyje t kaip $\hat{y}_i^{(t)}$. Tuomet:

$$\begin{aligned}
&\hat{y}_i^{(0)} = 0;\\
&\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i);\\
&\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i);\\
&\dots\\
&\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i);
\end{aligned} \tag{4}$$

Atrinktas medis optimizuoja aukščiau pateiktą tikslo funkciją. Žemiau pateikiama išraiška (5), kai nuostolio funkcija yra MSE (angl. *Mean Squared Error*):

$$obj^{(t)} = \sum_{i=1}^{n} \left[2\left(\hat{y}_i^{(t-1)} - y_i\right) f_t(x_i) + f_t(x_i)^2 \right] + \omega(f_t) + constant; \tag{5}$$

XGBoost modelis gali optimizuoti bet kokią nuostolio funkciją.

Kitas svarbus modelio aspektas yra reguliarizacija, kuri padeda išvengti persimokymo (angl. *overfitting*) skiriant baudas. Anksčiau minėtas medžio kompleksiškumas $\omega(f_k)$ gali būti aprašomas formule (6):

$$\omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2; \quad (6)$$

čia w yra medžio lapų įverčių vektorius, o T yra lapų skaičius.

Tuomet modelis įvertina medžio struktūrą ir toliau naudojama funkcija padeda atrasti geriausią padalinimą, kuris atliekamas lapą perskiriant į du lapelius:

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma; \quad (7)$$

Formulė gali būti išskaidoma į: 1) naujo kairiojo lapelio įvertis, 2) naujo dešiniojo lapelio įvertis, 3) tikrojo (nepadalinto) lapo įvertis, 4) pridėtojo lapo reguliarizacija. Jei $Gain$ įvertis mažesnis nei γ, ši šaka naudojama nebus. Tai yra medžio genėjimo (angl. *pruning*) technika. Išdėstant visus galimus įrašus galima atlikti nuskaitymą ir apskaičiuoti struktūros įvertinimą visiems įmanomiems padalinimams; taip gali būti atrastas efektyviausias padalinimas.

XGBoost modelis naudoja skaitinius kintamuosius, todėl kategoriniai kintamieji turi būti pertvarkomi į skaitines reikšmes. Algoritmas turi daug parametrų, kuriuos patartina derinti pagal sprendžiamą mašininio mokymosi problemą. Remiantis XGBoost internetinio tinklapio žinynu[11], pagrindiniai derinami XGBoost parametrai pateikiami lentelėje žemiau.

4 lentelė. Derinami XGBoost modelio parametrai

Parametras	**Numatytoji reikšmė bei galimas reikšmių intervalas**	**Parametro reikšmė**	**Panaudojimas eksperimentuose**
objective	*reg:squarederror* galimos reikšmės: *reg:squarederror*, *reg:absoluteerror*, *binary:logistic* ir kt.	Tikslo funkcija.	Pasirinkta *reg:squarederror* arba *binary:logistic* priklausomai nuo sprendžiamo uždavinio
learning_rate	0,3 Galimas intervalas: [0, 1]	Žingsnio mažinimas siekiant išvengti persimokymo. Žema reikšmė gerina modelio rezultatus, tačiau lėtina skaičiavimą.	Optimalus įvertis naudotas pagerinti modelio rezultatams.
gamma	0 Galimas intervalas: [0, ∞]	Didesnė *gamma* reikšmė didina reguliarizaciją.	Optimalus įvertis naudotas pagerinti modelio rezultatams.
max_depth	6 Galimas intervalas: [0, ∞]	Maksimalus medžio gylis. Didesnė reikšmė gali pagerinti modelio rezultatus, tačiau turi didesnį šansą persimokymui.	Optimalus įvertis naudotas pagerinti modelio rezultatams.
subsample	1 Galimas intervalas: [0, 1]	Nurodo medžio stebinių dalį. Mažinant šią reikšmę galima išvengti persimokymo, tačiau kyla ribotumo problema.	Optimalus įvertis naudotas pagerinti modelio rezultatams.
colsample_bytree	1 Galimas intervalas: [0, 1]	Stulpelių atranka medžiui. Keičiant parametro reikšmes galima mažinti persimokymo problemą.	Optimalus įvertis naudotas pagerinti modelio rezultatams.

[11] https://xgboost.readthedocs.io/en/latest/parameter.html

Parametras	Numatytoji reikšmė bei galimas reikšmių intervalas	Parametro reikšmė	Panaudojimas eksperimentuose
min_child_weight	1 Galimas intervalas: $[0, \infty]$	Minimalus svoris medžio šakų kūrimui. Keičiant parametro reikšmes galima mažinti persimokymo problemą.	-
reg_lambda	1 Galimas intervalas: $[0, \infty]$	L2 svorių reguliarizacija. Keičiant parametro reikšmes galima mažinti persimokymo problemą.	-
reg_alpha	0 Galimas intervalas: $[0, \infty]$	L1 svorių reguliarizacija. Keičiant parametro reikšmes galima mažinti persimokymo problemą.	-
eval_metric	Pagal pasirinktą tikslo funckiją: *rmse*, *mae*, *logloss*, *auc* ir kt.	Matai modelio tikslumo vertinimui.	-

Eksperimentuose su sistema Featuretools derinti XGBoost parametrai pateikiami stulpelyje „Panaudojimas eksperimentuose“.

2.4. Metodai, naudoti duomenims tvarkyti bei įvertinti

Šiame skyriuje aptariami metodai, naudoti duomenų lentelėms tvarkyti bei modelių rezultatams įvertinti.

2.4.1. Klasių disbalansas

Keletas nagrinėjamų reliacinių duomenų rinkinių pasižymi itin dideliu priklausomo kintamojo disbalansu, todėl pasirinktas metodas SMOTE (angl. *Synthetic Minority Over-sampling Technique*) klasių disbalanso problemai spręsti.

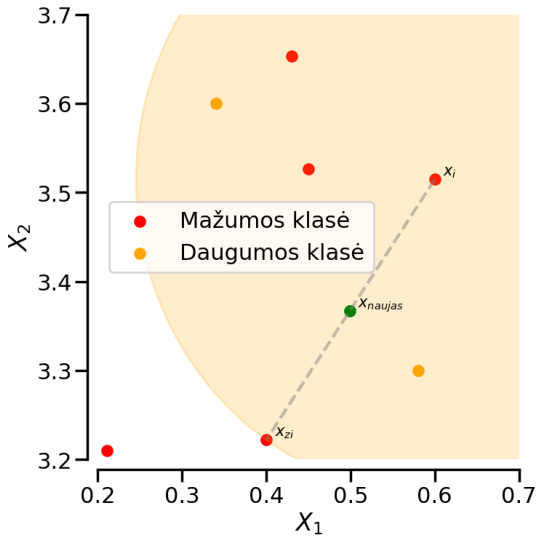
SMOTE metodas akademinėje literatūroje aprašomas 2002 metais [43], čia jis pateikiamas kaip mažumos įrašų sintetinis kūrimas. Metodas dirbtinai sukuria mažumos klasės įrašus tam, kad būtų pasiektas balansas nagrinėjamų klasių. Ši disbalanso problema aktuali tik klasifikacijos uždaviniams. SMOTE metodas pasirinktas siekiant neprarasti duomenų dirbtinai mažinant daugumos klasę.

Pirmiausia sukuriama atsitiktinė mažumos klasės stebinių imtis, kurios stebiniams sukuriami sintetiniai įrašai. Įrašui x_i sukuriamas x_{naujas} sintetinis įrašas panaudojant k-artimiausių kaimynų metodą (angl. *k-nearest neighbors*). Tarkime, iš trijų kaimyninių įrašų, egzistuojančių atrinktoje imtyje, pasirenkamas įrašas x_{zi} ir sintetinis įrašas sukuriamas pagal formulę (8):

$$x_{naujas} = x_i + \lambda \times (x_{zi} - x_i); \quad (8)$$

čia λ yra atsitiktinis numeris iš intervalo $[0, 1]$.

Tokia interpoliacija gali sukurti sintetinį įrašą, pavaizduotą paveiksle žemiau (žr. 15 pav.).



15 pav. Sintetinio įrašo kūrimas SMOTE metodu

SMOTE metodas naudojamas duomenų rinkiniams su dideliu disbalansu. Metodas taikytas modelio apmokymo imčiai.

2.4.2. Modelio tikslumo įvertinimo parametrai

Siekiant palyginti tiek naudojamus algoritmus, tiek atviro kodo sistemų veikimą tarpusavyje, modelio tikslumo įvertinimui pasirinkti šie matai:

- tikslumas (angl. a*ccuracy*);
- AUC – plotas po ROC kreive (angl. ROC – *Receiver Operating Characteristic,* AUC – *Area Under Curve*);
- kryžminė entropija (angl. *cross-entropy*);
- determinacijos koeficientas, arba R^2 (angl. *R Squared*);
- MAE – vidutinė absoliutinė paklaida (angl. *Mean Absolute Error*);
- RMSE – šaknis iš vidutinės kvadratinės paklaidos (angl. *Root Mean Squared Error*);

getML sistemoje papildomų integruotų tikslumo įvertinimo parametrų nėra. Toliau plačiau apibūdinami visi naudoti kriterijai.

Tikslumas. Tikslumas yra matas, nurodantis, kaip arti yra prognozuoti įrašai nuo realių reikšmių. Norint plačiau nagrinėti tikslumo sąvoką, reikalinga sumaišymo matrica (angl. *confusion matrix*) (žr. 5 lent.):

5 lentelė. Sumaišymo matrica

Imtis: P + N		**Prognozuojama reikšmė**	
		Teigiamas (P)	Neigiamas (N)
Reali reikšmė	Teigiamas (P)	*True positive* (TP)	*False negative* (FN)
	Neigiamas (N)	*False positive* (FP)	*True negative* (TN)

Remiantis pateikiama lentele, prognozuojamos klasifikacijos uždavinio reikšmės gali būti skirstomos į 4 skirtingus tipus. Teisingo priėmimo (angl. *True Positive*, TP) bei teisingo atmetimo (angl. *True negative*, TN) reikšmės rodo, kad prognozuota reikšmė buvo atspėta teisingai, o

klaidingo priėmimo (angl. *False Positive*, FP) bei klaidingo atmetimo (angl. *False Negative*, FN) reikšmės rodo, kai prognozuojama reikšmė buvo netiksli. Visa imtis klasifikacijos uždavinyje gali būti apibūdinama kaip teigiamų ir neigiamų reikšmių suma (P + N). Tikslumas nurodo, kokia dalis iš visų stebinių buvo prognozuota teisingai:

$$tikslumas = \frac{TP+TN}{P+N}; \quad (9)$$

AUC. Kitas su sumaišymo matrica susijęs matas yra AUC – plotas po ROC kreive. ROC kreivė nurodo klasifikatoriaus sąryšį tarp jautrumo (angl. *Sensitivity*) bei specifiškumo (angl. *Specificity*). Abu šie matai pateikiami formulėmis (10, 11):

$$sensitivity = \frac{TP}{TP+FN}; \quad (10)$$

$$specificity = \frac{TN}{TN+FP}; \quad (11)$$

Šie du matai aprašo ROC kreivę, kuri klasifikacijos uždaviniuose naudojama tikslumui įvertinti apskaičiavus plotą po kreive AUC (angl. *Area Under Curve*). Kuo didesnis plotas po ROC kreive, tuo geriau klasifikatorius atskiria klases. AUC įvertis matuojamas skalėje nuo 0 iki 1, čia 0,5 rodo, kad klasių skirstymas yra atsitiktinis, o geri klasifikatoriaus rezultatai yra kai AUC įvertis siekia nuo 0,8 iki 1.

Kryžminė entropija. Kitas naudotas modelio tikslumo vertinimo matas yra kryžminė entropija, kuri gali būti aprašoma formule (12):

$$H(P,Q) = -\sum p(x) log q(x); \quad (12)$$

čia $p(x)$ yra tikrasis priklausomo kintamojo tikimybių skirstinys, $q(x)$ yra numatomas priklausomo kintamojo tikimybių skirstinys.

Kuo numatomas (prognozuojamas) pasiskirstymas yra artimesnis tikrajam pasiskirstymui, tuo kryžminės entropijos nuostolis yra mažesnis ir modelio rezultatai yra geresni.

Determinacijos koeficientas, arba R^2 aprašomas formule (13):

$$R^2 = 1 - \frac{RSS}{TSS}; \quad (13)$$

čia RSS yra liekamųjų paklaidų kvadratų suma, o TSS – bendroji kvadratų suma.

R^2 nurodo kokią procentinę priklausomo kintamojo dalį paaiškina nepriklausomo kintamojo kitimas.

MAE, arba vidutinė absoliutinė paklaida apskaičiuojama pagal formulę žemiau (14):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i^r|; \quad (14)$$

čia $\hat{y}_i$ yra prognozuota reikšmė, y_i^r – reali reikšmė, n – duomenų imties dydis.

MAE nurodo gautų rezultatų artimumą tikrosioms reikšmėms.

RMSE, arba šaknis iš vidutinės kvadratinės paklaidos apskaičiuojama pagal formulę (15):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\widehat{y_i} - y_i^r)^2}; \quad (15)$$

čia $\widehat{y_i}$ yra prognozuota reikšmė, y_i^r – reali reikšmė, n – duomenų imties dydis.

RMSE, kaip ir MAE, taip pat nurodo prognozės artimumą tikrosioms reikšmėms, tačiau RSME teikia didesnį svorį didesnėms paklaidoms dėl kėlimo kvadratu, o MAE yra absoliutinis skirtumas. Abu paklaidų matavimo įverčiai rodo tikslesnius modelio rezultatus kai yra mažesni, t. y. kuo mažesnės MAE bei RSME paklaidos, tuo tikslesnės yra modelio prognozuojamos reikšmės.

2.4.3. Modelio parametrų derinimas

Derinant XGBoost modelio parametrus, naudojami keletas skirtingų metodų.

Tinklelio paieška (angl. g*rid search*) – tai nesudėtingas metodas modelio parametrų derinimui. Šis metodas sudaro visas galimas parametrų kombinacijas pagal pateiktą informaciją. Kiekviena sudaryta kombinacija yra tikrinama k-dalių kryžminio patikrinimo būdu, o geriausi modelio parametrai nustatomi pagal pasirinktą tikslumo matą. Geriausi modelio parametrai yra kombinacija, su kuria gautas geriausias modelio vertinimas. Šiame darbe tinklelio paieškos metodas naudotas Python paketo *sklearn* funkcija *GridSearchCV*[12]. Kadangi visi klasifikacijos uždavinio duomenų rinkiniai yra su klasių disbalansu, siekiama nustatyti geriausius XGBoost modelio parametrus pagal AUC įvertį; regresijos uždaviniams siekiama maksimizuoti determinacijos koeficientą. Visiems analizuojamiems modeliams parametrų derinimas atliekamas su 5 dalių kryžmine validacija.

Bajeso parametrų paieška (angl. *Bayes Search*) yra Python paketo *scikit-optimize* algoritmas[13], kuris remiasi Bajeso optimizavimo teorija. Teigiama, jog šis modelio parametrų derinimo algoritmas yra greitesnis ir efektyvesnis nei tinklelio paieška, ypač kai tiriamas duomenų rinkinys yra didelis arba nagrinėjama daug skirtingų parametrų. Bajeso parametrų paieškos metodas gali rasti tikslesnius parametrus su mažiau iteracijų, todėl yra greitesnis. Skirtingai nei tinklelio paieška, Bajeso parametrų derinimo algoritmas tikrina buvusias parametrų tikslumo reikšmes ir rezultatus, todėl analizei parenkami tie parametrai, kurie prieš tai vykdytose iteracijose pasirodė reikšmingesni. Šis algoritmas parametrams tikrinti naudoja k-dalių kryžminę validaciją. Šio algoritmo parametrų derinimui taip pat naudota 5 dalių kryžminė validacija.

Gauso hiperparametrų paieška (angl. *Gaussian Hyperparameter Search*) – getML sistemos kūrėjų algoritmas, kuris gali būti naudojamas parametrų derinimui. Algoritmas naudojamas getML funkcijoje *tune_predictors*, kuri padeda derinti parametrus jau apmokintam modeliui. Parametrų tikrinimas pradedamas nuo pagrindinio modelio, apmokinto su numatytaisiais parametrais. Tuomet vienu metu tikrinami 2 ar 3 skirtingi parametrai ir apmokomas naujas modelis, tikrinami rezultatai. Jei naujasis modelis yra geresnis nei buvęs, tuomet pagrindiniu modeliu tampa naujai sukurtasis, ir tikrinami kiti modelio parametrai. Taip atrandamas naujas tiksliausias modelis. Algoritmas tikrina parametrus lentelėje pateikta tvarka (žr. 6 lent.).

[12] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
[13] https://scikit-optimize.github.io/stable/auto_examples/sklearn-gridsearchcv-replacement.html

6 lentelė. getML funkcijos *tune_predictors* XGBoost parametrų paieškos algoritmo žingsniai

Parametrų derinimo etapas	**Parametras**	**Reikšmių erdvė**
1 – pagrindiniai parametrai	*learning_rate*	[0,05, 0.3]
2 – medžio parametrai	*max_depth*, *min_child_weights*, *gamma*	[1, 15], [1, 6], [0, 5]
3 – *sampling* parametrai	*colsample_bytree*, *subsample*	[0,75, 0,9], [0,75, 0,9]
4 – reguliarizavimo parametrai	*reg_alpha*, *reg_lambda*	[0, 5], [0, 10]

Pasak getML kūrėjų, daugeliu atvejų funkcijos *tune_predictors* pakanka suderinti XGBoost modelio parametrams.

3. Tiriamoji dalis

Šioje darbo dalyje aprašomi duomenų rinkiniai ir atlikti tyrimai. Žemiau pateikiamas 6 tirtų reliacinių duomenų rinkinių apibendrinimas. Visi nagrinėti rinkiniai publikuojami internete[14].

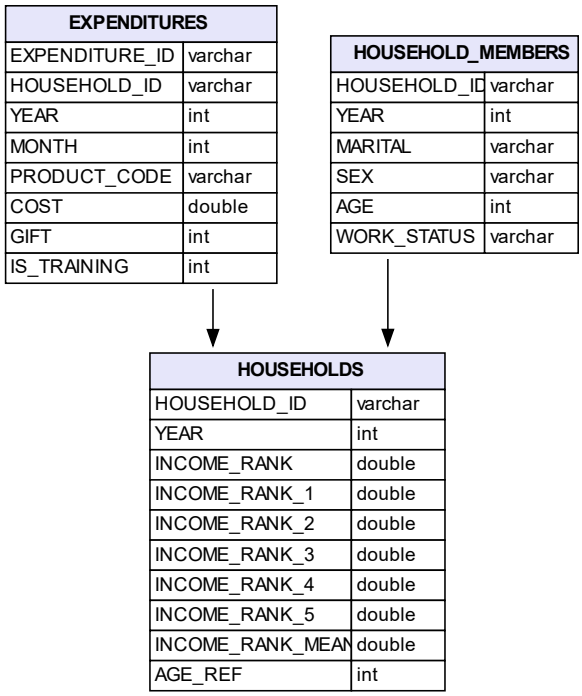
7 lentelė. Analizuojamų duomenų rinkinių apibendrinimas

Duomenų rinkinio pavadinimas	Dydis, MB	Lentelių skaičius	Įrašų skaičius	Atributų skaičius	Duomenų rinkinio tipas	Sprendžiama problema
ConsumerExpenditures	337,6	3	2.241.548	24	Prekyba	Klasifikacija
Financial	78,8	8	1.090.086	55	Finansai	Klasifikacija
VOC	2,7	8	29.067	89	Prekyba	Klasifikacija
CCS	15,9	6	422.868	29	Finansai	Regresija
Seznam	146,8	4	2.681.983	14	Prekyba	Regresija
Restbase	3	3	19.148	12	Prekyba	Regresija

Visi nagrinėjami rinkiniai turi ne mažiau tris reliaciniais ryšiais susietas lenteles. Įrašų skaičius lentelėje apibūdina įrašų kiekį visose duomenų rinkinio lentelėse, atributų skaičius – visų rinkinio lentelių atributų sumą. Rinkiniai sprendžia prekybos arba finansines problemas. Trys iš nagrinėtų rinkinių sprendžia detekcijos, trys – regresijos uždavinį.

3.1. ConsumerExpenditures duomenų rinkinys

Duomenų rinkinys, nagrinėjantis vartotojų pirkimo įpročius. Siekiama nustatyti, ar pirkta prekė yra



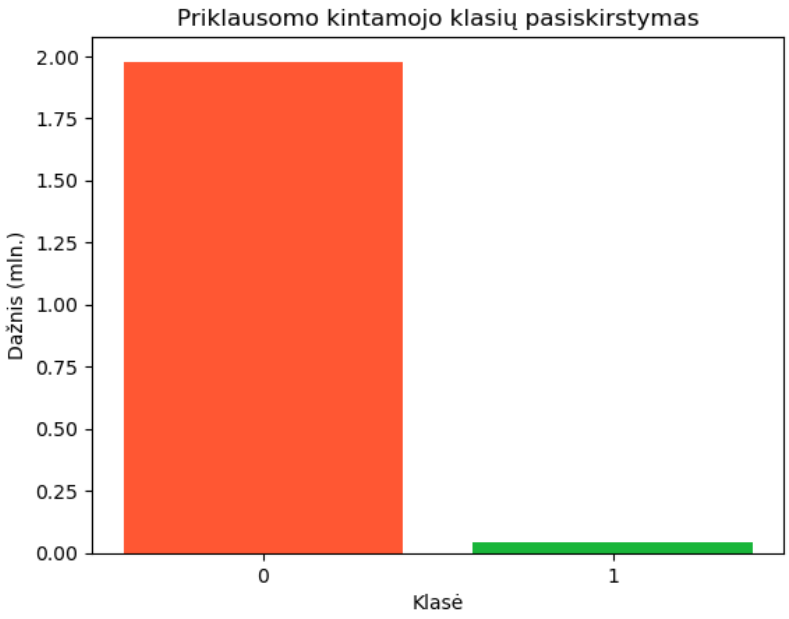
16 pav. ConsumerExpenditures duomenų rinkinys. Priklausomo kintamojo lentelė *EXPENDITURES* turi 2 mln. 20 tūkst. 634 įrašus bei 8 atributus. Priklausomas kintamasis – *GIFT*

[14] https://relational.fit.cvut.cz

dovana. Tai detekcijos uždavinys. Paveiksle pateikiama reliacinė duomenų schema (žr. pav. 16).

Priklausomas kintamasis *GIFT* turi dvi klases: 1 – pirkta prekė yra dovana, arba 0 – prekė nėra dovana. Kitos rinkinio lentelės detalizuoja nagrinėjamų namų ūkių pajamas, šeimos narių amžių, lytį, darbingumą.

Pagrindinėje duomenų rinkinio lentelėje *EXPENDITURES* atlikti keletas pertvarkymų: eksperimente nenaudojamas atributas *IS_TRAINING*, kadangi apmokymo ir testavimo imtis buvo pasirinkta atsitiktinai. Pastebėta problema – priklausomo kintamojo klasių disbalansas. Iš turimų duomenų 1 977 953 įrašai priklauso klasei 0 (ne dovana) ir tik 42 681 įrašas priklauso klasei 1 (dovana). Žemiau pateikiama klasių pasiskirstymo diagrama (žr. 17 pav.).



17 pav. ConsumerExpenditures rinkinio priklausomo kintamojo *GIFT* klasių disbalansas

Išnagrinėjus duomenis, esančius kitose rinkinio lentelėse, trūkstamos atributo *WORK_STATUS* reikšmės pakeičiamos į 0. Pastebima kategorijų gausa atribute *PRODUCT_CODE*, kuris nurodo tam tikrą perkamą produktą. Šis atributas turi 514 unikalių reikšmių.

Tyrimo eiga Featuretools

Požymių kūrimas atliekamas įprasta Featuretools sistemos aprašyta eiga: aprašomas subjektų rinkinys, kuriame nurodomi duomenų tipai bei ryšiai tarp analizuojamų lentelių. Požymių kūrimui pasirinktos numatytosios agregavimo ir transformavimo funkcijos, ignoruojami identifikaciniai stulpeliai (*HOUSEHOLD_ID*, *EXPENDITURE_ID*). DFS algoritmo metodu sukuriami 55 papildomi požymiai įrašomi į naują lentelę, toliau vadinamą požymių matrica.

Požymių matricos sukūrimo trukmė – 2 min. 8 sek. Sukūrus požymių matricą, koduojami kategoriniai kintamieji, pašalinami požymiai, kur trūkstamos reikšmės apima daugiau nei 50 % visų reikšmių, pašalinami nereikšmingi požymiai. Patikrinus koreliaciją tarp sudarytų požymių pašalinami 26 koreliuojantys atributai; trūkstamos reikšmės užpildomos medianos reikšmėmis, taip pat pašalinami stulpeliai, turintys ryšį su priklausomu kintamuoju:

– households.NUM_UNIQUE(expenditures.GIFT);

– households.MODE(expenditures.GIFT) = 0.

Modeliui kurti paruošta požymių matrica turi 2 020 634 įrašus bei 233 požymius, paruošta testavimo imtis lygi 25 %, apmokymo – 75 % pradinės duomenų imties. Dėl klasių disbalanso apmokymo imčiai SMOTE metodu kuriami sintetiniai mažumos klasės įrašai. Modelio parametrams derinti pasirenkama GridSearchCV (angl. *Grid Search Cross-Validation*) funkcija; kadangi duomenų rinkinys yra didelis, derinami tik keletas XGBoost modelio parametrų: *max_depth* bei *learning_rate*. Apmokomas XGBoost modelis su geriausiais parametrais, tyrimo rezultatai nurodomi lentelėje žemiau (žr. 8 lent.).

Tyrimo eiga getML

Vienas iš svarbiausių žingsnių getML sistemoje – rolių (angl. *roles*) arba duomenų tipų priskyrimas visiems duomenų rinkinyje esantiems atributams. Visoms 3 duomenų rinkinio lentelėms priskiriami tipai; čia pasirenkama nenaudoti stulpelio *EXPENDITURE_ID*. Kitas svarbus žingsnis – duomenų modelio sukūrimas. Čia nurodomi ryšiai tarp lentelių, priklausomo kintamojo lentelė *expenditures_getml*, taip pat atliekamas duomenų padalijimas į apmokymo, testavimo bei validavimo imtis panaudojant funkciją *getml.data.split.random*. Toliau nurodomi papildomi duomenų paruošimo parametrai (angl. *preprocessing*). Šiame duomenų rinkinyje pasirinkta panaudoti sezoniškumo tikrinimą, kadangi tai gali turėti įtaką dovanų pirkimui, bei kategorinių kintamųjų kodavimą dėl didelio skaičiaus kategorinių kintamųjų. Pasirenkami numatytieji parametrai FastProp bei Relboost algoritmams. Multirel bei RelMT algoritmai dėl kompiuterio techninių resursų trūkumo su šiuo duomenų rinkiniu tirti nebuvo.

Tyrimo rezultatai

Gauti modelių įvertinimai pateikiami lentelėje žemiau.

8 lentelė. ConsumerExpenditures duomenų rinkinio tyrimo rezultatai. Geriausios reikšmės stulpeliuose pateikiamos pajuodintu šriftu

Sistema	**Algoritmas**	**Požymių skaičius**	**Tikslumas**	**AUC**	**Kryžminė entropija**
Featuretools	DFS	55	0,895	0,682	0,332
getML	FastProp	88	**0,981**	**0,923**	**0,064**
getML	Relboost	38	0,981	0,914	0,066

Pagal tyrimų rezultatus galima teigti, jog getML FastProp algoritmu sukūrus papildomus požymius, modelio rezultatai yra tiksliausi. Verta paminėti, jog klasių disbalansui tvarkyti papildomų veiksmų getML sistemoje imtasi nebuvo, tačiau aukštas AUC įvertis rodo, jog ši problema neturi įtakos getML algoritmų tikslumui. Šiame duomenų rinkinyje tiek FastProp, tiek Relboost rezultatai yra geresni nei DFS algoritmo.

3.2. Financial duomenų rinkinys

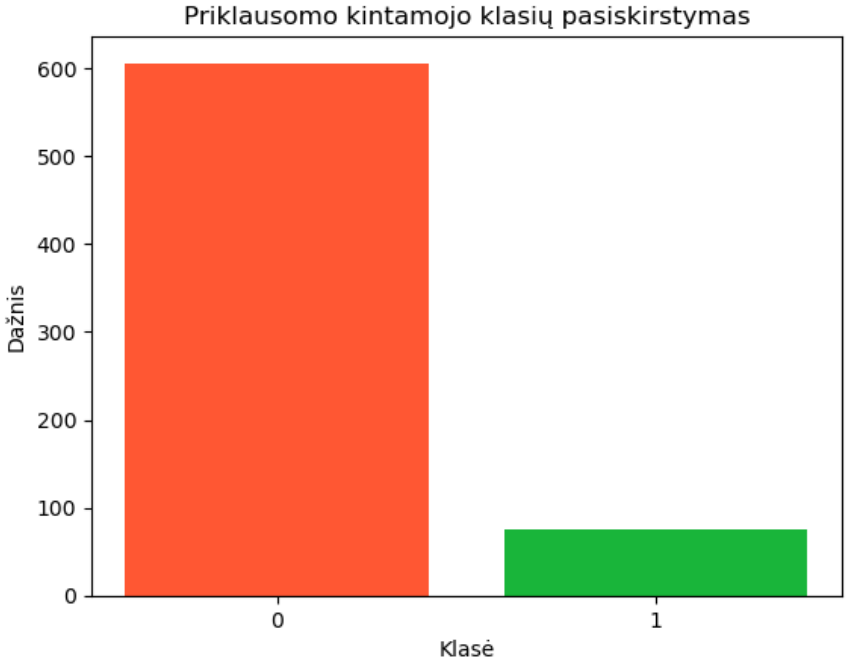
Reliacinis duomenų rinkinys, kuriame siekiama prognozuoti, ar teikiama paskola bus pelninga bankui. Sprendžiamas klasifikacijos uždavinys, kuriame siekiama nustatyti, ar klientas grąžins paskolos sumą (klasė 0) ar liks įsiskolinęs (klasė 1). Reliacinės duomenų lentelės pateikiamos priede (žr. 1 priedas), priklausomo kintamojo lentelė – paveiksle žemiau (žr. pav. 18).

loan	
loan_id	int
account_id	int
date	date
amount	int
duration	int
payments	decimal
status	varchar

18 pav. Financial duomenų rinkinio priklausomo kintamojo lentelė *Loan*. Šioje lentelėje yra 682 įrašai ir 7 atributai, priklausomas kintamasis – *status*

Pagrindinėje duomenų lentelėje saugoma informacija apie paskolas bei paskolos ėmimo data (*date*). Paskolos rizika banką domina dar prieš šią suteikiant, todėl atsižvelgiama į šią datą ir kitose lentelėse laikomi duomenys, egzistuojantys po paskolos paėmimo datos, pvz. piniginės perlaidos ir kita finansinė informacija, analizėje nėra naudojami siekiant išvengti informacijos apie žymas nutekėjimo (angl. *label leakage*). Analizei reikalinga informacija apie banko klientus saugoma papildomose reliaciniais ryšiais susietose lentelėse: tai informacija apie kliento asmens duomenis ir gyvenamąją vietą, kreditinę kortelę, perlaidas į kitus bankus bei kitas vykdytas operacijas, sąskaitų kiekį ir kt.

Pagrindinėje lentelėje saugoma informacija pertvarkoma siekiant supaprastinti nagrinėjamą problemą. Paskolos būsena (*status*) lentelėje pateikiama 4 skirtingais scenarijais: A – paskolos laikas baigėsi, klientas nėra įsiskolinęs, B – paskolos laikas baigėsi, klientas liko įsiskolinęs, C – paskolos terminas dar nesibaigė, klientas neturi įsiskolinimų bei D – paskolos terminas dar nesibaigė, tačiau kliento mokėjimai jau vėluoja. Šie 4 scenarijai pertvarkomi į detekcijos uždavinį, sprendžiantį ar klientas yra mokus: A bei C scenarijai žymi klasę 0 – klientas mokus, o scenarijai B ir D žymi klasę 1 – klientas nemokus, paskolos išdavimas yra rizikingas. Šis duomenų rinkinys taip pat turi disbalanso problemą (žr. pav. 19).



19 pav. Financial rinkinio priklausomo *status* kintamojo klasių disbalansas

Pagrindinė duomenų rinkinio lentelė aprašo 606 pelningas bei 76 nuostolingas paskolas. Kitų pertvarkymų pagrindinėje duomenų lentelėje nebuvo daryta. Žemiau pateikiami pertvarkymai kitose duomenų rinkinio lentelėse (žr. 9 lent.).

9 lentelė. Financial duomenų rinkinio lentelėse atlikti pertvarkymai

Problema	**Lentelė**	**Atributas**	**Duomenų tipas**	**Pertvarkymas**
Trūkstamos reikšmės	*district*	*A12, A15*	Skaičius	Užpildyta vidurkio reikšme.
Trūkstamos reikšmės	*trans*	*operation*	Kategorija	Pakeičiama reikšme „nežinomas“.
		k_symbol	Kategorija	Pakeičiama reikšme „nežinomas“.
Trūkstamos reikšmės	*order*	*k_symbol*	Kategorija	Pakeičiama reikšme „nežinomas“.

Kitų pertvarkymų duomenyse neatliekama.

Tyrimo eiga Featuretools

Duomenų rinkinio lenteles, egzistuojančius duomenų tipus bei reliacinius ryšius aprašius subjektų rinkinyje, algoritmui taip pat nurodomas laiko parametras (*cutoff_time*), kuris žymi paskolos paėmimo datą. Nurodžius šią informaciją, DFS algoritmas požymių kūrimui nenaudoja nežinomos informacijos, pvz. informacijos apie kliento transakcijas, vykdomas po paskolos paėmimo. Požymių kūrimui naudojamos numatytosios agregavimo bei transformavimo funkcijos, požymių inžinerijai nenaudojami identifikaciniai atributai: *account_id, card_id, client_id, trans_id, disp_id, district_id, loan_id, order_id* bei account_to. Per 48 sekundes DFS algoritmas sukuria 79 papildomus požymius.

Sukurta lentelė ruošiama modelio kūrimui įprastais metodais: koduojami kategoriniai kintamieji, šalinami nereikšmingi atributai, pašalinami 33 koreliuoti atributai, užpildomos trūkstamos reikšmės. Modeliui paruošta lentelė turi 682 įrašus bei 172 atributus. Lentelė atsitiktinai dalinama į apmokymo ir testavimo imtis, dėl didelio klasių disbalanso apmokymo imčiai taikomas SMOTE metodas. Kadangi šis duomenų rinkinys yra mažesnis, naudojamas *GridSearchCV* metodas XGBoost parametrų derinimui. Nagrinėjamas parametrų tinklelis: *max_depth, learning_rate, colsample_bytree, subsample, n_estimators* bei *gamma*. Pasirenkama 5 dalių kryžminė validacija ir apmokomas XGBoost modelis su geriausiais parametrais. Modelio rezultatai pateikiami lentelėje žemiau (žr. 10 lent.).

Tyrimo eiga getML

Į sistemą įkeliamos sutvarkytos duomenų rinkinio lentelės. Prieš paskiriant duomenų tipus (roles) atliekamas pertvarkymas duomenų lentelėse: getML sistemos dokumentacijoje teigiama, kad lentelėms, kurias sieja ryšys vienas-su-vienu (angl. *one-to-one*), požymių kūrimas nėra reikalingas, kadangi tokios lentelės gali būti tiesiog apjungiamos. Financial duomenų rinkinyje egzistuoja keletas lentelių, kurios jungiasi vienas-su-vienu ryšio tipu: *client, district, disp* bei *card*. Šios lentelės sujungiamos į vieną – gautoji lentelė turi 5 369 įrašus bei 24 atributus, kurie detalizuoja kliento asmeninę informaciją, gyvenamosios vietos detales bei informaciją apie kreditinę kortelę. Apjungiant lenteles gautos tuščios reikšmės atributuose (pvz. įrašai apie klientus, kurie neturi kreditinės kortelės) užpildomos 0. Kadangi naujai sukurtos lentelės duomenys jungiasi prie kitų lentelių per rakto lauką *account_id*, visi kiti raktų atributai (*disp_id, card_id, client_id, district_id*) pažymimi kaip nenaudojami požymių kūrime.

Kuriant duomenų modelį getML sistemoje taip pat yra galimybė išvengti informacijos apie žymas nutekėjimo (angl. *label leakage*): nurodoma kurti požymius pasinaudojant duomenimis, egzistuojančiais iki paskolos paėmimo datos. Sukurtas duomenų modelis turi 4 lenteles dėl anksčiau minėto lentelių apjungimo. Pagrindinė lentelė padalinama į apmokymo, testavimo bei validavimo imtis. Papildomi naudoti parametrai – sezoniškumas bei kategorinių kintamųjų kodavimas. XGBoost parametrų derinimui naudojama getML funkcija *tune_predictors*. Optimizuotas modelis įvertinamas su getML *score* funkcija. Šiame duomenų rinkinyje visų 4 nagrinėtų algoritmų požymių sudarymas, atranka bei modelio apmokymas atlikti greičiau nei per 1 min.

Rezultatai

10 lentelė. Financial duomenų rinkinio tyrimo rezultatai

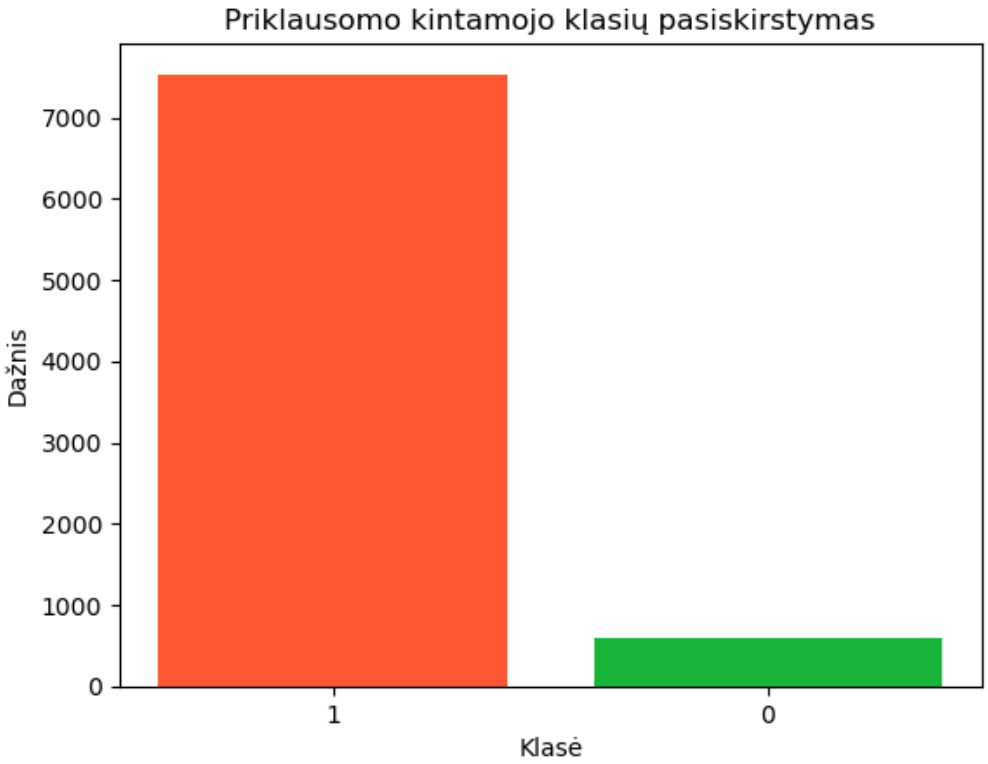
Sistema	Algoritmas	Požymių skaičius	Tikslumas	AUC	Kryžminė entropija
Featuretools	DFS	79	**0,971**	0,907	**0,121**
getML	FastProp	43	0,957	**0,982**	0,222
getML	Relboost	38	0,957	0,913	0,224
getML	Multirel	38	**0,971**	**0,982**	0,156
getML	RelMT	38	**0,971**	0,963	0,159

Visi nagrinėti algoritmai turi panašų tikslumo įvertį, žemiausias AUC įvertis DFS bei Relboost algoritmų rezultatuose. Šiuo atveju tiksliausias modelis apmokintas su Multirel algoritmu sukurtais požymiais. FastProp algoritmas turi įverčius, artimus RelMT bei Multirel algoritmams.

3.3. VOC duomenų rinkinys

Duomenų rinkinys, saugojantis vienos seniausių tarptautinių bendrovių – Rytų Indijos kompanijos (angl. *The East Indian Company*) prekybos kelionių bei administracinę informaciją. Sprendžiamas klasifikacijos uždavinys – siekiama numatyti, ar kelionė bus sėkminga (laivas pasieks galutinį uostą). Reliacinės duomenų rinkinio lentelės pateikiamos priede (žr. 2 priedas).

Pastebima, jog duomenų rinkinys turi daug trūkstamų reikšmių visose rinkinio lentelėse. Beveik visos duomenų lentelės laiko informaciją apie laivu plaukusius asmenis, detalizuojant tam tikro tipo keleivių skaičių išplaukiant, tam tikruose sustojimuose bei atplaukus į galutinį uostą. Pagrindinė duomenų lentelė (*voyages*) turi 8 126 įrašus bei 23 atributus. Dauguma atributų pašalinami dėl didelio skaičiaus trūkstamų reikšmių, taip pat pašalinami komentarai, tekstiniai atributai: *artificial_id, number_sup, trip_sup, master, particulars, next_voyage, bought, hired, type_of_boat*. Datų atributai – išvykimo, sustojimo, atvykimo – pašalinami dėl didelio trūkstamų reikšmių skaičiaus. Tai atributai *departure_date, cape_arrival, cape_departure, arrival_date*. Dauguma iš pašalintų požymių yra kategoriniai kintamieji, turintys tik keletą įrašų. Priklausomas kintamasis pertvarkomas į dviejų klasių klasifikacijos uždavinį, kuris siekia nustatyti, ar laivas suduš (klasė 0), ar sėkmingai atplauks į galutinį uostą (klasė 1). Prieš pakeitimą priklausomas kintamasis nurodė koks yra galutinis laivo uostas (jei šis buvo pasiektas). Kaip ir anksčiau nagrinėtuose duomenų rinkiniuose, taip ir VOC pastebėtas klasių disbalansas. 7 532 įrašai priklauso klasei 1 (laivas sėkmingai atplaukė į uostą), 594 įrašai priklauso klasei 0 (laivas neatplaukė į uostą). Skaičiavimui naudojami pagrindinės lentelės duomenys patikrinami ir sutvarkyti: pašalinti netinkami simboliai atributuose *tonnage, built, chamber*; kategorinių kintamųjų trūkstamos reikšmės keičiamos moda,



20 pav. VOC duomenų rinkinio priklausomo kintamojo *arrival_harbour* klasių pasiskirstymas

skaitinių – vidurkio reikšme. Toliau pateikiami duomenų pertvarkymai, atlikti kitose duomenų rinkinio lentelėse.

11 lentelė. VOC duomenų rinkinio lentelėse atlikti pertvarkymai

Problema	**Lentelė**	**Atributas**	**Duomenų tipas**	**Pertvarkymas**
Trūkstamos reikšmės	*craftsmen, impotenten, passengers, seafarers, soldiers*	*death_at_cape, left_at_cape, onboard_at_cape, death_during_voyage, onboard_at_arrival*	Skaičius	Užpildyta reikšme 0.
Trūkstamos reikšmės	*invoices*	*invoice, chamber*	Kategorija	Įrašai su trūkstamomis reikšmėmis pašalinami.
Atributai, neteikiantys informacijos	*craftsmen, impotenten, invoices, passengers, seafarers, soldiers,*	*number_sup, trip_sup*	Skaičius	Atributai pašalinami.
Trūkstamos reikšmės	*total*	Visi lentelės atributai	Skaičius	Lentelė nenaudojama tyrime.

Tyrimo eiga Featuretools

Subjektų rinkinyje aprašoma rinkinio schema, nurodomi ryšiai, duomenų tipai. Kuriant požymius pasirenkamos numatytosios agregavimo bei transformavimo funkcijos, DFS algoritmui pateikiami visi po duomenų pertvarkymo likę atributai. Per 11 sekundžių sukuriami 238 požymiai.

Sudaryta požymių matrica pertvarkoma modelio apmokymui: koduojami kategoriniai kintamieji, pašalinami 109 koreliuojantys požymiai, užpildomos trūkstamos reikšmės. Modelio apmokymui

paruošta duomenų lentelė turi 8 126 įrašus bei 136 atributus. Kadangi duomenų rinkinys taip pat turi disbalansą, apmokymo imčiai pritaikomas SMOTE metodas. Atlikus XGBoost modelio parametrų derinimą *GridSearchCV* metodu, apmokomas modelis ir gauti rezultatai pateikiami žemiau (žr. 12 lent.). Kadangi analizuojamas duomenų rinkinys yra mažesnis, modelio parametrų derinimui nagrinėjami šie parametrai: *max_depth*, *learning_rate*, *colsample_bytree*, *subsample*, *n_estimators* bei *gamma*.

Tyrimo eiga getML

Į sistemos atmintį įkeliamos jau sutvarkytos duomenų lentelės. Visiems duomenų rinkinio lentelėse egzistuojantiems atributams priskiriami duomenų tipai (rolės), kitų pertvarkymų neatliekama. Sukuriamas duomenų modelis, atliekamas pagrindinės lentelės *voyages* padalinimas į apmokymo, testavimo bei validavimo imtis. Papildomai duomenims apdoroti pasirenkamas trūkstamų reikšmių užpildymas bei kategorinių kintamųjų kodavimas. Naudojant numatytuosius parametrus, požymiai sukuriami su visais getML sistemos algoritmais, apmokomas XGBoost modelis vėliau derinamas getML sistemos funkcija *tune_predictors*. Modelio su numatytaisiais parametrais bei požymių kūrimas greičiausias su FastProp algoritmu – 4 sekundės, tačiau modelio parametrų derinimo trukmė – nuo 20 iki 40 min. Tyrimo rezultatai pateikiami lentelėje žemiau.

Tyrimo rezultatai

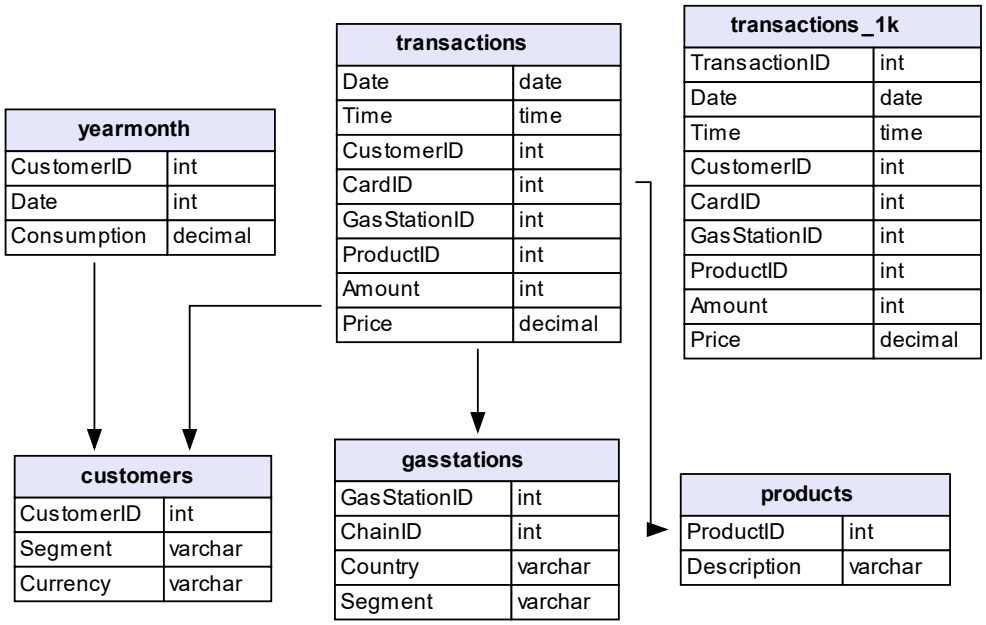
12 lentelė. VOC duomenų rinkinio tyrimo rezultatai

Sistema	Algoritmas	Požymių skaičius	Tikslumas	AUC	Kryžminė entropija
Featuretools	DFS	238	0,924	0,578	0,207
getML	FastProp	106	**0,937**	**0,838**	**0,200**
getML	Relboost	56	0,932	0,780	0,343
getML	Multirel	56	**0,937**	0,828	0,217
getML	RelMT	21	0,933	0,820	0,213

Pagal gautus tikslumo bei AUC rezultatus visi getML algoritmai yra tikslesni nei DFS. Tiksliausias modelis naudojo FastProp algoritmo sugeneruotą požymių lentelę. DFS algoritmo sukurti požymiai rodo gerą modelio tikslumą, tačiau AUC įvertis nėra geras.

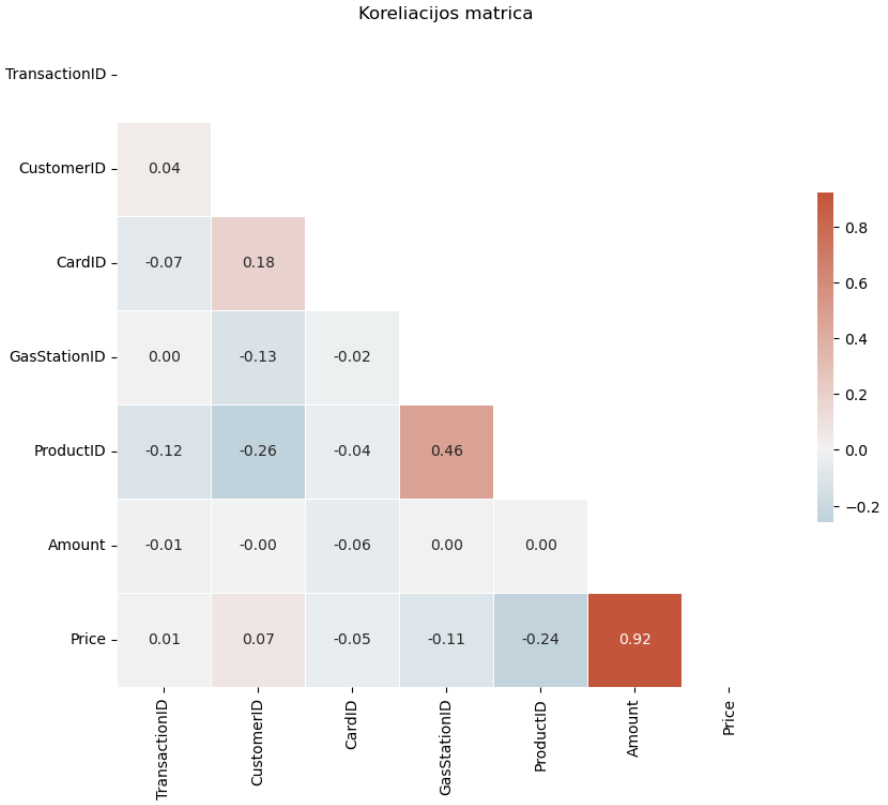
3.4. CCS duomenų rinkinys

Šiame duomenų rinkinyje pateikiama kreditinių kortelių transakcijų informacija apie mokėjimus, atliktus degalinėse. Tai – regresijos uždavinys. Siekiama nustatyti kokią sumą klientas išleis apsipirkdamas. Žemiau pateikiama reliacinė duomenų rinkinio schema (žr. 21 pav.).



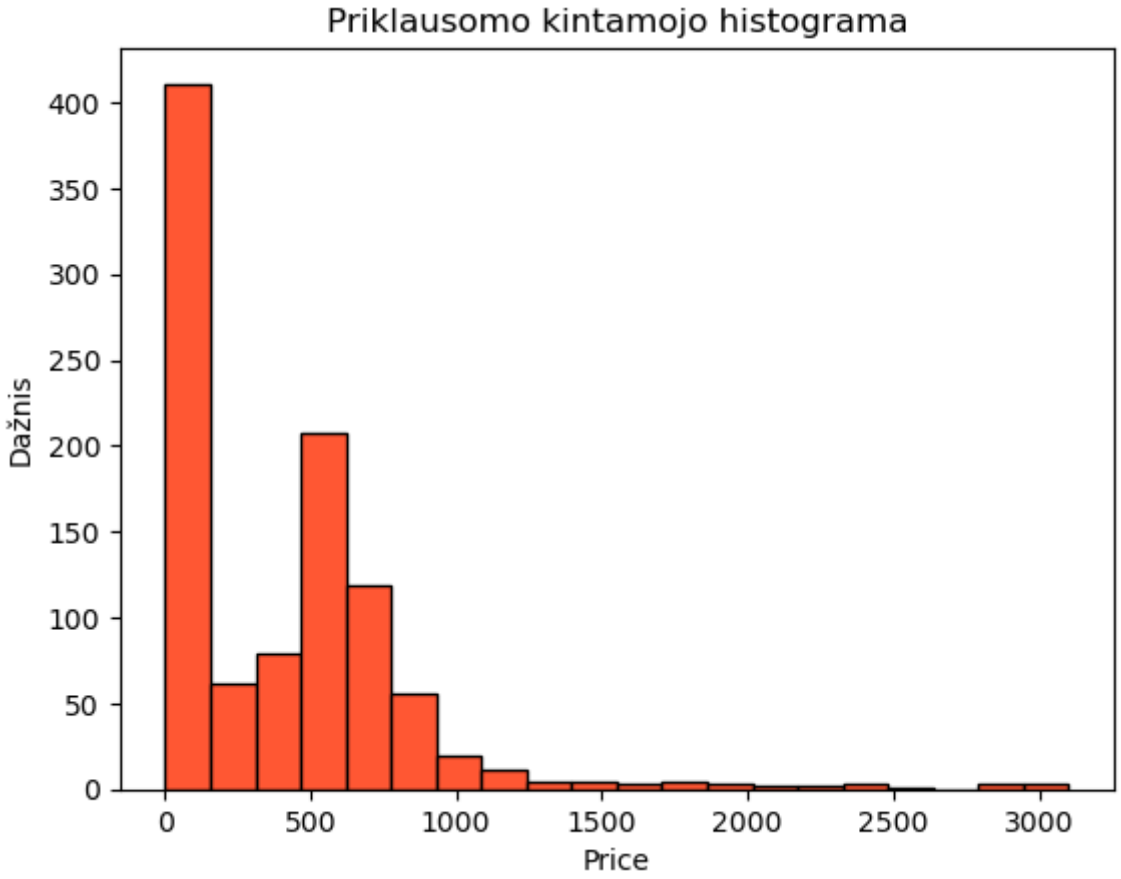
21 pav. CCS duomenų rinkinys. Priklausomo kintamojo lentelė *transactions_1k* turi 1000 įrašų bei 9 atributus. Priklausomas kintamasis – *price*

Priklausomo kintamojo lentelė – *transactions_1k* – laiko informaciją apie 1000 transakcijų, kurių metu įvairiose degalinėse klientai pirko tam tikras prekes ir išleido tam tikrą sumą (*price*). Datos (*date*) bei laiko (*time*) stulpeliai apjungiami į vieną. Kiekio (*amount*) atributas pašalinamas dėl aukštos koreliacijos (0,92) su priklausomu kintamuoju *price* (žr. 22 pav.).



22 pav. CCS duomenų rinkinio lentelės *transactions_1k* koreliacijos matrica

Tyrimui nenaudojamos lentelės *transactions* (lentelė neturi įrašų) bei *products* – ši lentelė turi tik du atributus: rakto lauką *ProductID* bei prekės aprašymą *Description*, kuris yra kategorinis kintamasis su 529 unikalių reikšmių. Pirkta prekė identifikuojama pagrindinėje lentelėje pagal lauką *ProductID*, todėl svarbi informacija nėra prarandama. Duomenų rinkinyje nėra trūkstamų reikšmių, daugiau duomenų pertvarkymų neatliekama. Žemiau pateikiama priklausomo kintamojo histograma.



23 pav. CCS duomenų rinkinio priklausomo kintamojo *price* histograma

Kitose duomenų lentelėse pertvarkymų neatliekama – duomenyse nėra trūkstamų reikšmių.

Tyrimo eiga *Featuretools*

Aprašius subjektų rinkinį, parenkamos numatytosios agregavimo bei transformavimo funkcijos. Visi atributai, esantys duomenų rinkinio lentelėse, yra naudojami požymių kūrimui. Raktų laukams *CustomerID, GasStationID, ChainID, CardID* bei *ProductID* parenkamas kategorinių duomenų tipas. Pastebima, jog didžioji dauguma įrašų – kategoriniai kintamieji. Per keletą sekundžių DFS algoritmas sukuria 71 požymį.

Sukūrus požymių matricą dėl didelio kategorinių duomenų kiekio kodavimui panaudojama Python *Pandas* paketo funkcija *get_dummies,* kuri koduoja visus kategorinius kintamuosius. Tai stipriai padidina sukurtos požymių matricos dydį. Modelio kūrimui paruošta lentelė turi 1000 įrašų ir 1 397 atributus. XGBoost modelio parametrų derinimui pasirenkamas BayesSearchCV metodas, pasirenkama 5 dalių kryžminė validacija. Dėl didelio atributų kiekio tikrinami modelio parametrai *max_depth, learning_rate* bei *n_estimators*. Siekiama maksimizuoti determinacijos koeficientą.

Suderinus modelio parametrus, atliekamas modelio apmokymas, gauti rezultatai pateikiami lentelėje žemiau (žr. 13 lent.).

Tyrimo eiga getML

Tyrimui naudojami anksčiau sutvarkyti duomenys. Nustatomi getML duomenų tipai kiekvienam lentelėse esančiam atributui. getML algoritmai analizei nenaudoja rakto laukų, todėl tokie atributai kaip *CustomerID, GasStationID* nėra naudojami požymių kūrime ir yra reikalingi tik reliaciniams ryšiams tarp lentelių aprašyti. Atliekamas pagrindinės lentelės *transactions_1k* padalinimas į apmokymo, testavimo bei validavimo imtis, aprašomas duomenų modelis, papildomam duomenų tvarkymui pasirenkamas kategorinių kintamųjų kodavimas, trūkstamų reikšmių užpildymas bei sezoniškumas. Požymių kūrimui pasirenkami numatytieji parametrai. Dėl naudojamo kompiuterio resursų trūkumo nagrinėjamam rinkiniui negali būti atliktas modelio parametrų derinimas getML funkcija *tune_predictors*. Tyrimo rezultatai pateikiami lentelėje žemiau.

Tyrimo rezultatai

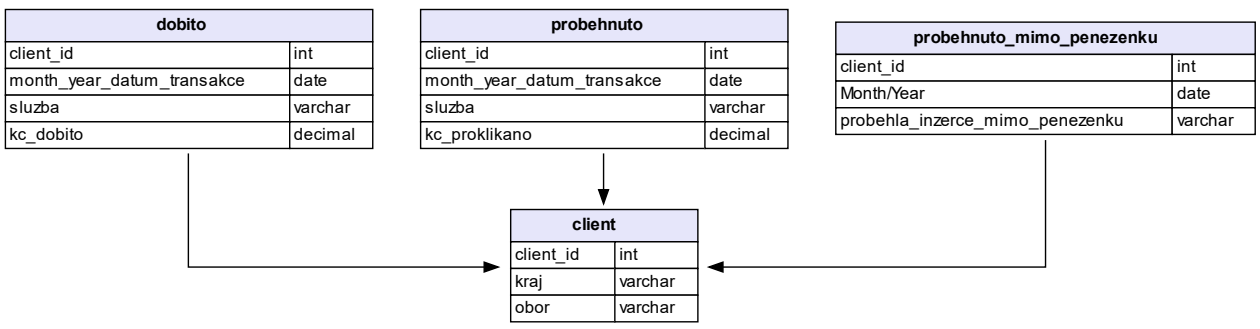
13 lentelė. CCS duomenų rinkinio tyrimo rezultatai

Sistema	**Algoritmas**	**Požymių skaičius**	**R^2**	**MAE**	**RMSE**
Featuretools	DFS	71	0,419	**154,336**	**323,962**
getML	FastProp	76	0,437	167,817	338,702
getML	Relboost	59	**0,475**	461,443	590,810
getML	Multirel	59	0,378	175,530	358,594
getML	RelMT	24	0,347	181,470	381,584

Aukščiausias R^2 gautas getML Relboost algoritmo sukurtiems požymiams, tačiau MAE bei RMSE paklaidos taip pat didžiausios. Tuo tarpu DFS algoritmo požymiams apmokytas modelis turi mažiausias paklaidas bei pakankamai aukštą R^2. Žemiausias R^2 gautas požymiams, sukurtiems Multirel bei RelMT algoritmų.

3.5. Seznam duomenų rinkinys

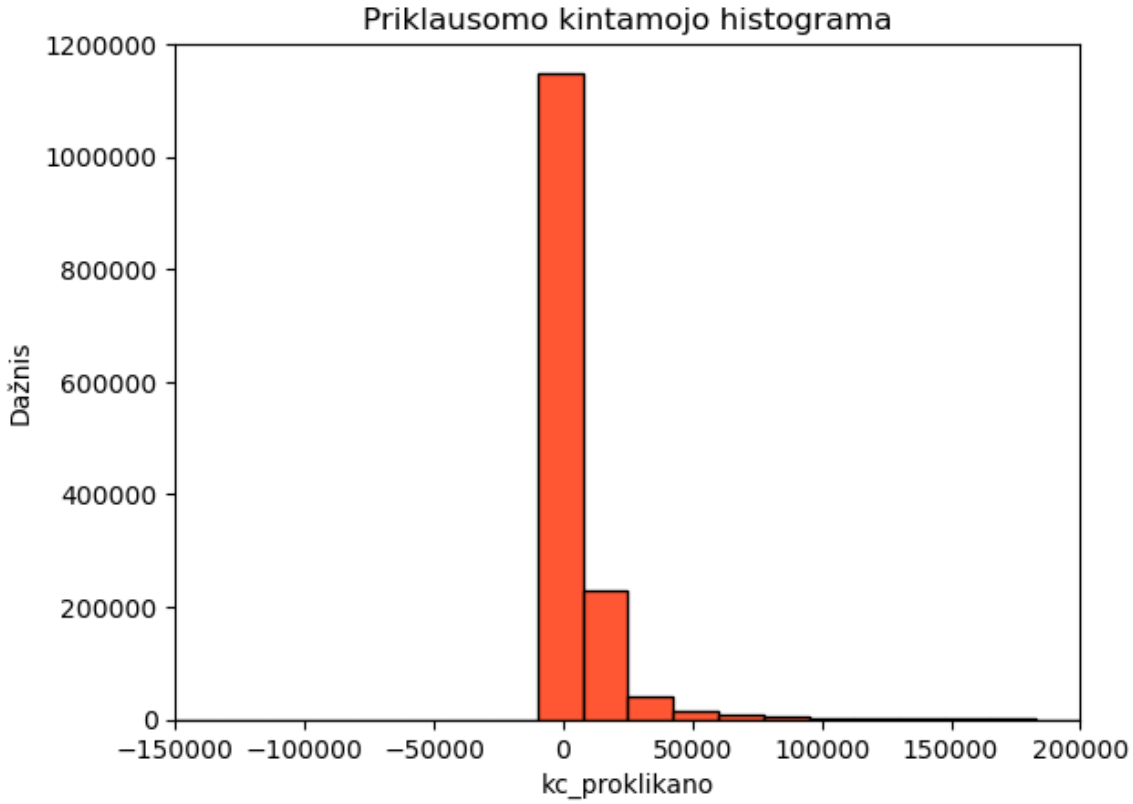
Duomenų rinkinys aprašo Čekijos internetinio tinklapio ir paieškos variklio www.seznam.cz kliento piniginės (angl. *wallet*) išlaidas internetinėms reklamoms ir skelbimams. Tyrimo tikslas – nustatyti



24 pav. Seznam duomenų rinkinys. Priklausomo kintamojo lentelė *probehnuto* turi 1 462 078 įrašų bei 4 atributus. Priklausomas kintamasis – *kc_proklikano*

kliento transakcijos sumą. Reliacinė duomenų rikinio schema pateikiama 24 paveiksle.

Seznam duomenų rinkinyje pagrindinė lentelė, saugojanti transakcijų informaciją, turi 4 atributus: kliento identifikacinį kodą (raktą *client_id*), transakcijos datą, nurodytą metais ir mėnesiu (dienos nėra detalizuojamos, transakcijos suma apima vieną mėnesį) atributą *sluzba*, nurodantį paslaugos tipą bei priklausomo kintamojo atributą *kc_proklikano*, nurodantį transakcijos sumą. Pastebima, jog pagrindinėje duomenų lentelėje beveik visi atributai turi nedidelį kiekį trūkstamų reikšmių. Paslaugos atributas (*sluzba*) užpildomas moda, įrašai su trūkstama *client_id* (rakto) reikšme yra pašalinami. Tikrinant priklausomą kintamąjį, pastebimos didelės išskirtys, todėl šios pašalinamos panaudojus z-įverčio metodą: pašalinami įrašai, kurie yra nutolę nuo vidurkio per tris standartinius nuokrypius. Pašalinus trūkstamas reikšmes bei didžiausias išskirtis, pertvarkyta pagrindinė lentelė turi 1 455 947 įrašus bei 4 atributus. Pastebima, jog lentelėje saugoma informacija apie 70 tūkst. 491 klientą, vadinasi, šis atributas nebus naudojamas požymių kūrime dėl pernelyg didelio kategorijų skaičiaus. Žemiau pateikiama priklausomo kintamojo histograma.



25 pav. Seznam duomenų rinkinio priklausomo kintamojo *kc_proklikano* histograma

Kitose duomenų rinkinio lentelėse saugoma ši informacija: *client* – kliento gyvenamoji vieta ir anonimiškas kliento darbo srities atributas, *dobito* – suma, sumokėta į Seznam piniginę Čekijos valiuta; *probehnuto* – suma, apmokėta iš lėšų, esančių Seznam piniginėje; *probehnuto_mimo_penezenku* – suma, apmokėta iš kitų lėšų (ne iš Seznam piniginės).

Kitose lentelėse atliekamas panašus duomenų tvarkymas, pateikiamas lentelėje žemiau (žr. 14 lent.).

14 lentelė. Seznam duomenų rinkinio lentelėse atlikti pertvarkymai

Problema	Lentelė	Atributas	Duomenų tipas	Pertvarkymas
Trūkstamos reikšmės	*dobito*	*client_id*	Rakto laukas	Įrašai su trūkstamomis reikšmėmis pašalinami.
Trūkstamos reikšmės	*client*	*kraj,* *obor*	Kategorija	Trūkstamos reikšmės užpildomos moda.
Atributai, neteikiantys informacijos	*probehnuto_mimo_penezenku*	*probehla_inzerce_mimo_penezenku*	Kategorija	Atributas pašalinamas

Tolimesnių pertvarkymų papildomose lentelėse neatliekama.

Tyrimo eiga Featuretools

Sudaromas subjektų rinkinys, kuriame nurodomi reliaciniai ryšiai, duomenų tipai, nagrinėjamos lentelės. Naudojamos numatytosios transformavimo bei agregavimo funkcijos, tačiau pasirenkama mažiau skaičiavimų: atsisakoma funkcijų *weekday* ir *day* kadangi duomenų rinkinyje turima informacija nenurodo mėnesio dienos. DFS algoritmui nurodoma ignoruoti visose lentelėse esantį rakto lauką *client_id* ir su šiuo atributu nekurti požymių dėl unikalių reikšmių gausos. Dėl duomenų rinkinio dydžio skaičiavimai trunka ilgiau: beveik per 2 minutes DFS algoritmu sukuriami 37 papildomi požymiai.

Algoritmo sudaryta lentelė paruošiama modelio apmokymui. Ši lentelė turi 1 455 947 įrašus bei 162 atributus. XGBoost modelio parametrai derinami *GridSearchCV* metodu, dėl duomenų gausos nagrinėjami tik keletas parametrų: *max_depth, learning_rate* bei *n_estimators*. Pasirenkama 5 dalių kryžminė validacija. Suderinto XGBoost modelio rezultatai pateikiami lentelėje žemiau (žr. 15 lent.).

Tyrimo eiga getML

Sutvarkytoms duomenų lentelėms priskiriami duomenų tipai (rolės). Aprašomas reliacinis duomenų modelis, atliekamas pagrindinės lentelės padalinimas į apmokymo, testavimo bei validavimo imtis. Pasirenkamas kategorinių kintamųjų kodavimas, sezoniškumas bei trūkstamų reikšmių užpildymas, požymių kūrimui bei modelio apmokymui nustatomi numatytieji parametrai. Dėl duomenų dydžio ir naudojamo kompiuterio resursų, šiam duomenų rinkiniui negali būti atliktas modelio parametrų derinimas, todėl funkcija *tune_predictors* nėra naudojama. Testavimo imties rezultatai su getML algoritmais nurodomi žemiau.

Tyrimo rezultatai

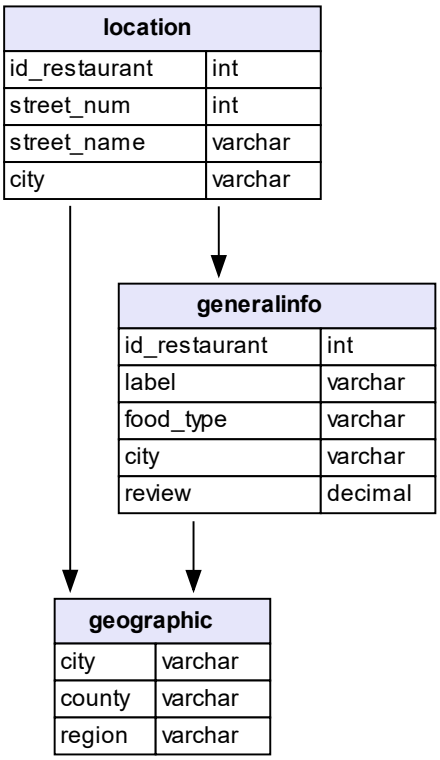
15 lentelė. Seznam duomenų rinkinio tyrimo rezultatai

Sistema	Algoritmas	Požymių skaičius	R^2	MAE	RMSE
Featuretools	DFS	37	**0,761**	**2884,150**	**7063,814**
getML	FastProp	102	0,623	3832,417	8920,005
getML	Relboost	53	0,618	3959,310	8960,768
getML	Multirel	53	0,617	4160,139	8971,057
getML	RelMT	18	0,641	3832,657	8685,147

Išsiskiria geri DFS algoritmo rezultatai – tiek determinacijos koeficientas, tiek paklaidos. Daugiausiai požymių sukurta FastProp algoritmu, šio algoritmo rezultatai iš getML sistemos algoritmų – vieni geriausių kartu su RelMT algoritmu su vos 18 papildomų požymių.

3.6. Restbase duomenų rinkinys

Nagrinėjamas duomenų rinkinys, kuriame saugomi San Francisko restoranų duomenys. Šio rinkinio analizės tikslas – prognozuoti restorano įvertinimą. Žemiau pateikiamas reliacinis duomenų rinkinys, atributai bei duomenų tipai, nurodomi ryšiai tarp lentelių:

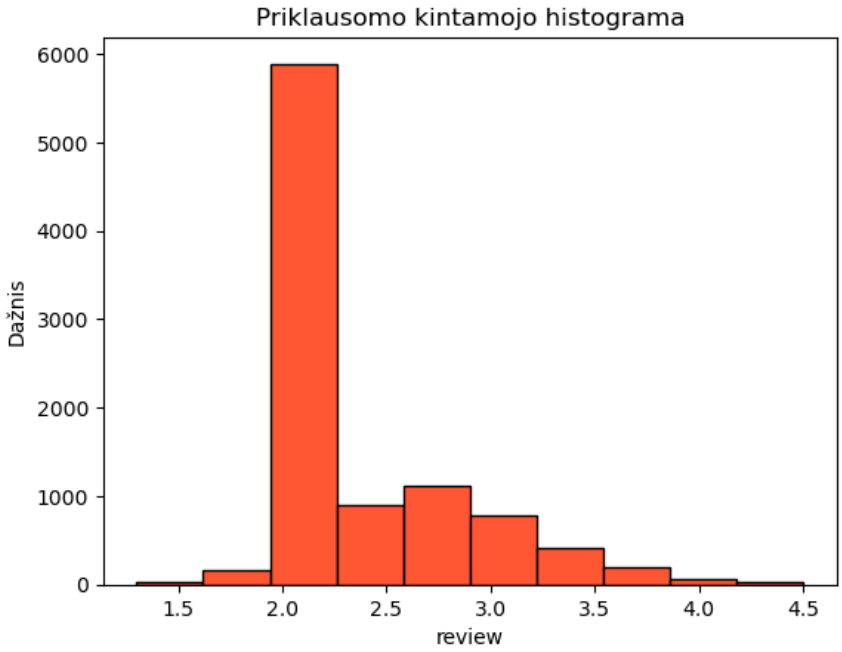


26 pav. Restbase duomenų rinkinys. Priklausomo kintamojo lentelė *generalinfo* turi 9 590 įrašų bei 5 atributus. Priklausomas kintamasis – *review*

Pagrindinė duomenų lentelė neturi trūkstamų reikšmių. Atributas *label* nurodo restorano pavadinimą, todėl analizėje nebus naudojamas. Likę kintamieji – kategoriniai, nurodantys maisto tipą bei restorano vietą mieste (*city*). Priklausomo kintamojo *review* reikšmės kinta nuo 1,3 iki 4,5. Priklausomo kintamojo histograma pateikiama paveiksle (žr. 27 pav.). Kita duomenų rinkinio lentelė *geographic* detalizuoja vietoves; pakeitimų čia neatlikta. Lentelė *location* nurodo restorano vietą – gatvę bei numerį, čia visi kintamieji turi trūkstamų reikšmių. Pateikiami ne visi restoranai (*id_restaurant*). Gatvės pavadinimo (*street_name*) trūkstamos reikšmės užpildomos populiariausia gatvės pavadinimo reikšme pagal miestą (*city*); gatvės numeris (*street_num*) – pagal gatvės numerio medianą mieste. Miesto reikšmės užpildomos reikšmėmis iš pagrindinės lentelės, kadangi šioje lentelėje nėra trūkstamų reikšmių (visa informacija yra žinoma). Kitų pakeitimų lentelėse neatliekama.

Tyrimo eiga Featuretools

Išsaugoma informacija apie duomenų rinkinio lenteles, ryšius bei duomenų tipą. Kadangi beveik visi duomenų rinkinio kintamieji – kategoriniai, naudojama daugiau agregavimo ir transformavimo



27 pav. Restbase duomenų rinkinio priklausomo kintamojo *review* histograma

funkcijų, padedančių kurti požymius iš tekstinių laukų. Požymių sudarymui nenaudojamos datos transformavimo funkcijos, kadangi čia nėra datos informacijos. Naudojamos numatytosios funkcijos ir papildomai naudojama *average_count_per_unique*, *number_of_common_words*, *number_of_unique_words*, *median_word_length*, *count_string*, *number_of_common_words*, bei *whitespace_count*. Per keletą sekundžių DFS algoritmu sukuriama 70 požymių.

Atliekamas sukurtos lentelės paruošimas modelio apmokymui: koduojamos kategorinės reikšmės, šalinamos nereikšmingos bei koreliuojančios reikšmės. Apmokymui paruošta duomenų lentelė turi 9 590 įrašų bei 362 atributus. Modelio parametrų derinimui naudojamas *BayesSearchCV* metodas. Gauti rezultatai pateikiami lentelėje žemiau (žr. 16 lent.).

Tyrimo eiga getML

Sutvarkyti duomenys importuojami į sistemą, duomenims priskiriamos rolės. Kadangi rakto laukas lentelėje *geographic* yra kategorinis kintamasis miestas (*city*), atributas dubliuojamas – getML sistemoje rakto laukas (rolė *join_key*) nėra naudojamas požymių kūrimui, tačiau tai svarbus kintamasis šiame duomenų rinkinyje. Sukuriamas duomenų modelis, pagrindinė lentelė dalinama į apmokymo, testavimo ir validavimo imtis. Funkcijos, naudojamos papildomam duomenų tvarkymui – kategorinių kintamųjų kodavimas bei trūkstamų reikšmių užpildymas. XGBoost parametrų derinimas atliekamas getML funkcija *tune_predictors*. Gauti rezultatai pateikiami lentelėje žemiau (žr. 16 lent).

16 lentelė. Restbase duomenų rinkinio tyrimo rezultatai

Sistema	Algoritmas	Požymių skaičius	R^2	MAE	RMSE
Featuretools	DFS	70	0,187	0,337	**0,453**
getML	FastProp	61	**0,204**	0,337	0,460
getML	Relboost	52	0,174	0,339	0,470
getML	Multirel	52	0,142	0,344	0,488
getML	RelMT	17	0,198	**0,333**	0,463

Su dideliu kiekiu kategorinių kintamųjų visais algoritmais sukurti požymiai rodo panašius rezultatus. Geriausias determinacijos koeficientas – FastProp algoritmo sukurtoje lentelėje. Didžiausios paklaidos – lentelėje, sukurtoje Multirel algoritmo, mažiausios – DFS, FastProp bei RelMT algoritmų.

3.7. Tyrimų įžvalgos ir rekomendacijos

Atlikus tyrimus su šešiais duomenų rinkiniais susiduriama su keletu skirtingų problemų kiekviename iš jų. Detekcijos uždaviniuose pastebėtas klasių disbalansas didžiausią įtaką turi su Featuretools vykdytiems eksperimentams. Trims nagrinėtiems detekcijos uždaviniams panaudotas SMOTE metodas tik iš dalies pagerina modelio įverčius, ir lyginant su getML sistemos algoritmais, AUC įvertinimas Featuretools tyrimuose yra mažiausias. getML sistemos algoritmai atliktuose klasifikacijos uždavinio eksperimentuose veikia geriau net ir esant žymiam klasių disbalansui, dviejuose iš trijų nagrinėtų detekcijos uždavinių geriausią įvertį turi FastProp algoritmas. Pastebėta, jog šis algoritmas veikia greičiausiai iš nagrinėtų ir taip pat sukuria daugiausiai papildomų požymių lyginant su kitais getML algoritmais.

Duomenų rinkinių su regresijos uždaviniu problemos skirtingos kiekviename duomenų rinkinyje – CCS rinkinyje yra vos 1000 įrašų modelio apmokymui, validavimui bei testavimui, Seznam duomenų rinkinio priklausomo kintamojo reikšmės pasiskirsčiusios itin netolygiai, o Restbase duomenų rinkinyje analizei gali būti panaudojami tik kategoriniai kintamieji. Regresijos uždaviniams analizuoti Featuretools naudojamas DFS algoritmas pateikia geresnius rezultatus nei detekcijos uždaviniams. DFS algoritmo sukurtais požymiais apmokintas modelis tiksliausiai prognozuoja Seznam duomenų rinkinio priklausomą kintamąjį – didžiausias ne tik determinacijos koeficientas, bet ir mažiausios paklaidos, nors DFS algoritmu sukurti tik 37 papildomi požymiai. Žemiausi MAE bei RMSE pastebimi ir Featuretools eksperimentuose CCS duomenų rinkinyje bei kategorinių kintamųjų duomenų bazėje Restbase. Tuo tarpu getML sistemoje skirtinguose duomenų rinkiniuose pasižymi skirtingi algoritmai, tačiau FastProp sukurti požymiai visuose atliktuose tyrimuose yra labai arti ar net vieni iš tiksliausių. CCS duomenų rinkinio tyrime geriausias determinacijos koeficientas gaunamas su Relboost algoritmo sukurtais požymiais, tačiau čia taip pat matomos ir didžiausios MAE bei RMSE paklaidos; tuo tarpu FastProp algoritmo determinacijos koeficientas yra geresnis nei DFS bei artimas Relboost ir taip pat turi vienas iš mažiausių paklaidų. Tiek Seznam, tiek Restbase atliktuose eksperimentuose FastProp algoritmo sukurti požymiai getML sistemoje modelio įvertinime turi artimas reikšmes tiksliausiems įverčiams.

Išnagrinėjus algoritmus galima teigti, jog automatinės požymių inžinerijos algoritmų kodo realizacijos nagrinėtose sistemose gali paspartinti verslo procesus nagrinėjant duomenis mašininio mokymosi uždaviniuose. Apžvelgus tiek detekcijos, tiek regresijos uždavinių eksperimentus galima

teigti, jog getML FastProp algoritmas – vienas stabiliausių dėl sukurtų požymių naudingumo modelio tikslumui. Šis algoritmas gali būti panaudojamas tiek detekcijos uždaviniuose, pvz. nagrinėjant vartotojų pirkimo įpročius ar analizuojant rizikas finansų sektoriuje, tiek regresijos uždaviniuose, pvz. prognozuojant pirkimo sumą ar pelningumą. Tuo tarpu DFS algoritmas išsiskiria mažiausiomis MAE bei RMSE paklaidomis, todėl gali būti naudojamas regresijos uždaviniams spręsti, kai siekiama minimizuoti prognozuojamų reikšmių paklaidas.

Išvados

1. Išanalizavus mokslinę literatūrą, pastebimas verslo įmonių poreikis skaitmenizuoti ir automatizuoti verslo procesus ne tik dėl procesų greičio bei tikslumo gerinimo, bet ir siekiant išlaikyti konkurencingumą. Raginimas skaitmenizuoti procesus pastebimas valstybiniu lygmeniu – raginimas automatizuoti procesus pastebimas tiek Lietuvos, tiek Europos Sąjungos mastu.
2. Nagrinėjant akademinę literatūrą pastebima plokštinimo metodų gausa. Nagrinėjama tema aktuali ir reikalinga verslo procesams gerinti, kadangi reliacinės duomenų bazės yra populiariausios bei stabiliausios versle naudojamos DBVS ir galimybė reliaciniais ryšiais apjungtus duomenis panaudoti mašininio mokymosi uždaviniuose yra itin aktuali.
3. Tyrimui atlikti pasirinkti algoritmai (realizuoti Python paketuose): DFS (Featuretools), FastProp (getML), Relboost (getML), Multirel (getML) ir RelMT (getML). Algoritmams pritaikyti naudojamos 6 reliacinės duomenų bazės iš prekybos bei finansų srities: ConsumerExpenditures, Financial ir VOC detekcijos uždaviniui, CCS, Seznam ir Restbase – regresijos uždaviniui spręsti.
4. Nagrinėjami duomenų rinkiniai sutvarkomi, atliekamas duomenų paruošimas plokštinimui Featuretools ir getML algoritmais. Nagrinėjamais algoritmais sukurti požymiai panaudojami XGBoost modelio apmokymui; suderinus modelio parametrus, vertinami testavimo imties rezultatai.
5. Detekcijos uždaviniuose geriausi rezultatai gauti su getML algoritmais. FastProp rezultatai yra geriausi arba labai arti geriausių rezultatų: FastProp AUC įverčiai detekcijos uždavinyje siekia 0,92, 0,98 ir 0,83. Klasių disbalansas getML algoritmams įtakos nedaro, tuo tarpu DFS algoritmo rezultatai detekcijos uždaviniuose su klasių disbalansu nėra geri. Regresijos uždaviniams spręsti tiek DFS, tiek getML algoritmai rodo gerus rezultatus. Geriausi R^2 įverčiai regresijos uždavinyje: 0,48 (Relboost), 0,76 (DFS), 0,20 (FastProp). Regresijos uždaviniuose DFS algoritmas išsiskiria mažiausiomis MAE ir RMSE paklaidomis lyginant su kitais nagrinėtais algoritmais.
6. Remiantis gautais rezultatais, geriausias algoritmas verslo procesams analizuoti yra getML FastProp. Šio algoritmo sukurti požymiai mašininio mokymosi uždavinyje rodo gerus rezultatus tiek kredito rizikos vertinimo, tiek pirkėjo išleidžiamos sumos prognozavimo uždaviniuose. Mažiausios MAE ir RMSE paklaidos gautos su DFS algoritmu, todėl regresijos uždaviniams versle, pvz., pardavimų prognozei, DFS algoritmas gali būti naudingas dėl mažesnių prognozės paklaidų. Vis dėlto DFS algoritmo rezultatai rodo, kad tai nėra tinkamas pasirinkimas detekcijos uždaviniui spręsti. Nagrinėjant duomenų rinkinį su kategorinių kintamųjų gausa, visi nagrinėti algoritmai rodo panašius rezultatus. Apibendrinant galima teigti, kad analizuoti algoritmai gali būti panaudojami požymių inžinerijai mašininio mokymosi uždavinyje automatizuoti. Tinkamą algoritmą reiktų pasirinkti pagal sprendžiamą verslo problemą.

Literatūros sąrašas

1. LIU, Day-Yang, Shou-Wei CHEN and Tzu-Chuan CHOU. Resource fit in digital transformation: Lessons learned from the CBC Bank global e-banking project. *Management Decision* [interaktyvus]. Emerald, 2011, vol. **49**(10), pp. 1728–1742 [žiūrėta 2022-02-28]. Doi: https://doi.org/10.1108/00251741111183852
2. IMRAN, Faisal and Jussi KANTOLA. Review of Industry 4.0 in the Light of Sociotechnical System Theory and Competence-Based View: A Future Research Agenda for the Evolute Approach. *Advances in Human Factors, Business Management and Society* [interaktyvus]. Springer, 2018, vol. **783**, pp. 118–128 [žiūrėta 2022-02-27]. Doi: 10.1007/978-3-319-94709-9_12
3. Informacinės visuomenės plėtros komitetas. 2021 m. skaitmeninės ekonomikos ir visuomenės indeksas (DESI) Lietuva. *Europos Komisija* [interaktyvus]. 2021 [žiūrėta 2022-03-02]. Prieiga per: https://ivpk.lrv.lt/uploads/ivpk/documents/files/DESI_2021__Lithuania__lt_2BDSuCQhAL89xoeMg7hbqZjNArY_80592.pdf
4. Lietuvos Respublikos ekonomikos ir inovacijų ministerija. *Lietuvos dirbtinio intelekto strategija*. Lietuvos Respublikos ekonomikos ir inovacijų ministerija [interaktyvus]. 2020 [žiūrėta 2022-03-02]. Prieiga per: https://eimin.lrv.lt/uploads/eimin/documents/files/DI_strategija_LT.pdf
5. RENDA, Andrea., Sylvia SCHWAAG SERGER, Daria TATAJ et al. Industry 5.0, a transformative vision for Europe: governing systemic transformations towards a sustainable industry. *Publications Office of the European Union* [interaktyvus]. 2022 [žiūrėta 2022-03-03]. Doi: https://data.europa.eu/doi/10.2777/17322
6. BREQUE, Maija, Lars DE NUL, Athanasios PETRIDIS. Industry 5.0: towards a sustainable, human-centric and resilient European industry. *Publications Office of the European Union* [interaktyvus]. 2021 [žiūrėta 2022-03-03]. Doi: 10.2777/308407
7. Pranešimas spaudai. Nauja Komisijos ataskaita atskleidė, koks svarbus skaitmeninis atsparumas ištikus krizei. *Europos Komisija* [interaktyvus]. Briuselis, 2020 [žiūrėta 2022-03-04]. Prieiga per: https://ec.europa.eu/commission/presscorner/detail/lt/ip_20_1025
8. McKinsey & Company. Industry 4.0 after the initial hype. Where manufacturers are finding value and how they can best capture it. *McKinsey Digital* [interaktyvus]. 2016 [žiūrėta 2022-03-05]. Prieiga per: https://www.mckinsey.com/~/media/mckinsey/business%20functions/mckinsey%20digital/our%20insights/getting%20the%20most%20out%20of%20industry%204%200/mckinsey_industry_40_2016.pdf
9. SYED, Rehan. et al. Robotic Process Automation: Contemporary themes and challenges. *Computers in Industry* [interaktyvus]. 2020, vol. **115**. [žiūrėta 2022-03-11]. ISSN 0166-3615. Doi: https://doi.org/10.1016/j.compind.2019.103162
10. LE, Quang Bon, Minh Dat NGUYEN, Van Can BUI, Thi Mai Huong DANG. The Determinants of Management Information Systems Effectiveness in Small- and Medium-Sized Enterprises. *Journal of Asian Finance, Economics and Business* [interaktyvus]. 2020 vol. **7**(8) pp. 567–576 [žiūrėta 2022-03-15]. Doi: 10.13106/jafeb.2020.vol7.no8.567
11. ANAGNOSTE, Sorin. Robotic Automation Process - The next major revolution in terms of back office operations improvement. *Proceedings of the International Conference on*

Business Excellence [intearaktyvus]. 2017, vol. **11**, pp. 676–686 [žiūrėta 2022-03-14]. Doi: https://doi.org/10.1515/picbe-2017-0072

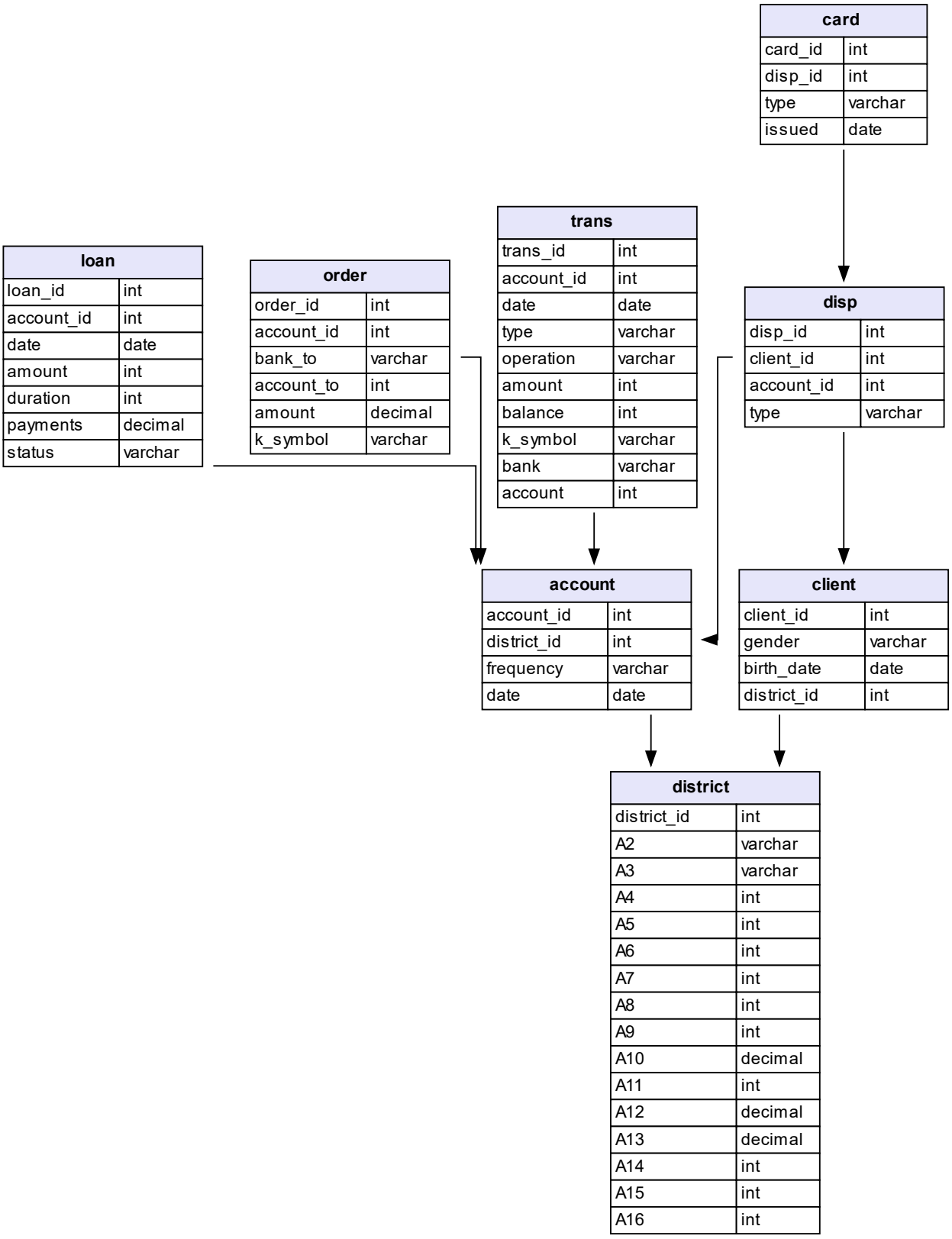
12. LAMBERTON, C., D. BRIGO, D. HOY. Impact of Robotics, RPA and AI on the Insurance Industry: Challenges and Opportunities. *Journal of Financial Perspectives* [interaktyvus]. 2017, vol. **4**(1) [žiūrėta 2022-03-15]. Prieiga per: https://ssrn.com/abstract=3079495
13. GUHA, Abhijit and Samanta DEBABRATA. Hybrid Approach to Document Anomaly Detection: An Application to Facilitate RPA in Title Insurance. *International Journal of Automation and Computing* [interaktyvus]. Springer, 2020, vol. **18**, pp. 55–72 [žiūrėta 2022-03-16]. Doi. https://doi.org/10.1007/s11633-020-1247-y
14. BENEDET, Paulo and Ivo NIKOLOV. Winning the war for talent in product development. *McKinsey & Company* [interaktyvus]. 2020 [žiūrėta 2022-03-18]. Prieiga per: https://www.mckinsey.com/capabilities/operations/our-insights/operations-blog/winning-the-war-for-talent-in-product-development
15. BUTKIENĖ, Rita ir kt. Duomenų bazės kūrimas. Nuo esybių-ryšių modelio iki duomenų įvedimo ir išvedimo formų: mokomoji knyga. *Technologija* [interaktyvus]. 2019 [žiūrėta 2022-03-20]. ISBN 9786090216118. Doi: https://doi.org/10.5755/e01.9786090215982
16. BUTLERIS, Rimantas ir kt. Duomenų bazių kūrimas Microsoft SQL Server priemonėmis: mokomoji knyga. *Technologija* [interaktyvus]. 2019 [žiūrėta 2022-03-20]. ISBN 9786090216316. Doi: https://doi.org/10.5755/e01.9786090216316
17. RAASVELDT, Mark. Integrating Analytics with Relational Databases. *Leiden Institute of Advanced Computer Science (LIACS)* [interaktyvus]. 2020 [žiūrėta 2022-03-26]. ISBN 9789061960003. Prieiga per: https://hdl.handle.net/1887/97593
18. HAIGH, Thomas. How Charles Bachman invented the DBMS, a foundation of our digital world. *Communications of the ACM* [interaktyvus]. 2016, vol. **59**(7), pp. 25–30 [žiūrėta 2022-03-26]. Doi: https://doi.org/10.1145/2935880
19. BACHMAN, Charles W. The Origin of the Integrated Data Store (IDS): The First Direct-Access DBMS. *IEEE Annals of the History of Computing* [interaktyvus]. 2009 vol. **31**(4), pp. 42–54 [žiūrėta 2022-03-26]. Doi: https://doi.org/10.1109/MAHC.2009.110
20. CODD, Edgar Frank. A relational model of data for large shared data banks. *Communications of the ACM* [interaktyvus]. 1970, vol. **13**(6), pp. 377–387 [žiūrėta 2022-03-27]. Doi: https://doi.org/10.1145/362384.362685
21. NAYAK, Ameya, Anil PORIYA and Dikshay POOJARY. Type of NOSQL databases and its comparison with relational databases. *International Journal of Applied Information Systems* [interaktyvus]. 2013, vol. **5**(4), pp. 16–19 [žiūrėta 2022-03-28]. Prieiga per: https://www.researchgate.net/publication/302557703_Article_Type_of_nosql_databases_and_its_comparison_with_relational_databases
22. ČEREŠŇÁK, Roman and Michal KVET. Comparison of query performance in relational a non-relation databases. *Transportation Research Procedia* [interaktyvus]. 2019, vol **40**, pp. 170–177 [žiūrėta 2022-03-28]. ISSN 2352-1465. Doi: 10.1016/j.trpro.2019.07.027
23. WONG, Wai Tak. Advanced Elasticsearch 7.0: A Practical Guide to Designing, Indexing, and Querying Advanced Distributed Search Engines. *Packt Publishing* [interaktyvus]. 2019 [žiūrėta 2022-03-28]. ISBN: 9781789957754. Prieiga per: https://search.ebscohost.com/login.aspx?direct=true&db=e000xww&AN=2238475&site=ehost-live

24. ROBINSON, Ian, Jim WEBBER and Emil EIFREM. Graph databases: new opportunities for connected data. *O'Reilly Media, Inc* [interaktyvus]. 2015 [žiūrėta 2022-03-29]. ISBN: 9781491930892. Prieiga per: https://books.google.lt/books?hl=en&lr=&id=RTvcCQAAQBAJ&oi=fnd&pg=PR2&dq=Graph+Databases+robinson&ots=fLeSIBl6lH&sig=mP68x1fij2hJViRPjC7LUiWbcwI&redir_esc=y#v=onepage&q=Graph%20Databases%20robinson&f=false
25. TSAI, Chun-Wei, Chin-Feng LAI, Han-Chieh CHAO and Athanasios V. VASILAKOS. Big data analytics: a survey. *Journal of Big Data 2* [interaktyvus]. 2015, vol. **21** [žiūrėta 2022-04-01]. Doi: https://doi.org/10.1186/s40537-015-0030-3
26. JORDAN, I. Michael and Tom. M. MITCHELL. Machine learning: Trends, perspectives, and prospects. *Science* [interaktyvus]. 2015 vol. **349**, pp. 255–260 [žiūrėta 2022-04-02]. Doi: 10.1126/science.aaa8415
27. CHEN, Yi-Wei, Qingquan SONG and Xia HU. Techniques for Automated Machine Learning. *ACM SIGKDD Explorations Newsletter* [interaktyvus]. 2021, vol. **22**(2), pp. 35–50 [žiūrėta 2022-04-03]. Doi: https://doi.org/10.1145/3447556.3447567
28. LAVRAČ, N., B. ŠKRLJ and M. ROBNIK-ŠIKONJA. Propositionalization and embeddings: two sides of the same coin. *Machine Learning* [interaktyvus]. 2020, vol. **109**(7), pp. 1465–1507 [žiūrėta 2022-04-05]. Doi: https://doi.org/10.1007/s10994-020-05890-8
29. COLLET, P. and A. BEAUGNON. AutoFeatures: Knowledge-Driven Automatic Feature Engineering for Detection Systems. *French Network and Information Security Agency (ANSSI)* [interaktyvus]. 2019 [žiūrėta 2022-04-05]. Prieiga per: https://www.cesar-conference.org/wp-content/uploads/2019/11/20191120_J2_240_P-COLLET_AutoFeatures_Knowledge-Driven_Automatic_Feature_Engineering_for_Detection_Systems.pdf
30. KROGEL, Mark-A. and Stefan WROBEL. Transformation-Based Learning Using Multirelational Aggregation. *Inductive Logic Programming* [interaktyvus]. 2001, vol. **2157** [žiūrėta 2022-04-06]. Doi: https://doi.org/10.1007/3-540-44797-0_12
31. KANTER, J. M. and K. VEERAMACHANENI. Deep feature synthesis: Towards automating data science endeavors. *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)* [interaktyvus]. 2015, pp. 1–10 [žiūrėta 2023-03-24]. Doi: https://doi.org/10.1109/DSAA.2015.7344858
32. PEROVŠEK Matic, Anže VAVPETIČ, Janez KRANJC, Bojan CESTNIK and Nada LAVRAČ. Wordification: Propositionalization by unfolding relational data into bags of words. *Expert Systems with Applications* [interaktyvus]. 2015, vol. **42**(17–18), pp. 6442-6456 [žiūrėta 2022-04-06]. ISSN 0957-4174. Doi: 10.1016/j.eswa.2015.04.017
33. KROGEL, Mark-A, Simon RAWLES, Filip ZELEZNÝ, Peter FLACH, Nada LAVRAČ and Stefan WROBEL. Comparative Evaluation of Approaches to Propositionalization. *Inductive Logic Programming* [interaktyvus]. 2003, vol. **2835**, pp. 197–214 [žiūrėta 2022-04-07]. Doi: https://doi.org/10.1007/978-3-540-39917-9_14
34. ZHAO, W., X. LI, G. RONG, M. LIN, C. LIN and Y. YANG. DAFEE: A Scalable Distributed Automatic Feature Engineering Algorithm for Relational Datasets. Iš: Algorithms and Architectures for Parallel Processing: 20th International Conference, ICA3PP 2020, New York City, NY, USA, October 2–4, 2020, Proceedings, Part II

[interaktyvus]. 2020, pp. 32–46 [žiūrėta 2022-04-06]. Doi: https://doi.org/10.1007/978-3-030-60239-0_3
35. KHURANA U., D. TURAGA, H. SAMULOWITZ and S. PARTHASRATHY. Cognito: Automated Feature Engineering for Supervised Learning. 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), Barcelona, Spain [interaktyvus]. 2016, pp. 1304–1307 [žiūrėta 2022-04-07]. Doi: https://doi.org/10.1109/ICDMW.2016.0190
36. NORDIN, A. End to end machine learning workflow using automation tools. *Massachusetts Institute of Technology* [interaktyvus]. 2018 [žiūrėta 2022-04-07]. Prieiga per: https://dspace.mit.edu/handle/1721.1/119776
37. KATZ, G., E. C. R. SHIN and D. SONG. ExploreKit: Automatic Feature Generation and Selection. 2016 IEEE 16th International Conference on Data Mining (ICDM), Barcelona, Spain [interaktyvus]. 2016, pp. 979–984 [žiūrėta 2022-04-07]. Doi: 10.1109/ICDM.2016.0123
38. SONG, M., J. WANG, T. ZHANG, G. ZHANG, R. ZHANG and S. SU. Effective Automated Feature Derivation via Reinforcement Learning for Microcredit Default Prediction. 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK [interaktyvus]. 2020, pp. 1–8 [žiūrėta 2022-04-07]. Doi: 10.1109/IJCNN48605.2020.9207410
39. FRANK, Richard, Flavia MOSER and Martin ESTER. A Method for Multi-relational Classification Using Single and Multi-feature Aggregation Functions. Knowledge Discovery in Databases: PKDD 2007, 11th European Conference on Principles and Practice of Knowledge Discovery in Databases [interaktyvus]. 2007, vol. **4702**, pp. 430–437 [žiūrėta 2022-04-07]. Doi: https://doi.org/10.1007/978-3-540-74976-9_43
40. HONGYU, Guo and Viktor HERNA. Learning from Skewed Class Multi-relational Databases. *Fundamenta Informaticae* [interaktyvus]. 2008, vol. **89**(1), pp. 69–94 [žiūrėta 2022-04-07]. Prieiga per: https://www.researchgate.net/publication/220444420_Learning_from_Skewed_Class_Multi-relational_Databases
41. ATRAMENTOV, A., H. LEIVA and V. HONAVAR. A Multi-relational Decision Tree Learning Algorithm – Implementation and Experiments. *Inductive Logic Programming. ILP 2003. Inductive Logic Programming* [interaktyvus]. 2003, vol. **2835**, pp. 38–56 [žiūrėta 2022-04-08]. Doi: https://doi.org/10.1007/978-3-540-39917-9_5
42. TIANQI, Chen and Carlos GUESTRIN. XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining [interaktyvus]. 2016, pp. 785–794 [žiūrėta 2023-03-28]. Doi: https://doi.org/10.1145/2939672.2939785
43. CHAWLA, N. V., K. W. BOWYER, L. O. HALL, W. P. KEGELMEYER. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research* [interaktyvus]. 2002, vol. **16**, pp. 321–357 [žiūrėta 2023-04-01]. Doi: https://doi.org/10.1613/jair.953

Priedai

1 priedas. Financial duomenų rinkinys.



2 priedas. VOC duomenų rinkinys.