VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

Sprendimų palaikymo sistema gyvybės draudimo rizikos vertinimui

Decision Support System for Health Insurance Risk Evaluation

Baigiamasis bakalauro darbas

Atliko: Vladas Zvonkus

VU el. p.: vladas.zvonkus@mif.stud.vu.lt

Vadovas: docentė Asta Slotkienė

Vilnius
2023

TURINYS

SANTRAUKA

Pagrindinis baigiamojo darbo tikslas – palengvinti gyvybės draudimo rizikos vertintojų vykdomus procesus bei ateityje kuriamų mašininio mokymosi sprendimų įdiegimą sukuriant centralizuotą sprendimų palaikymo sistemą. Darbo metu buvo sprendžiami šie uždaviniai: išanalizuoti bei palyginti sprendimų palaikymo sistemų sprendimus draudimo rinkoje, suprojektuoti centralizuotą sprendimų palaikymo sistemos modulį, įgyvendinti pirmąja centralizuoto sprendimų priėmimo modulio versiją, realizuoti sprendimų palaikymo funkcionalumą rizikos vertinimo atvejui bei palyginti įvairių sprendimų palaikymo modelių efektyvumą konkrečiam verslo atvejui. Atlikti šie iškelti uždaviniai: išanalizuoti bei palyginti trys sprendimų palaikymo sistemų sprendimai draudimo rinkoje, suprojektuotas centralizuotas sprendimų palaikymo sistemos modulis, įgyvendinta pirmoji sprendimų priėmimo modulio versija su klasifikavimo bei naujų modelių registravimo funkcionalumu, sprendimų palaikymo funkcionalumas rizikos vertinimo atvejui realizuotas su žemu tikslumu mažumos klasėms.

Raktiniai žodžiai: sprendimų palaikymo sistema#1, rizikos vertinimas#2, mašininis mokymasis#3

SUMMARY

The main objective of the thesis is to ease the processes of life insurance risk assessors and future machine learning solution deployment by creating a centralized decision support system. The following tasks were addressed: analyse and compare existing decision support system solutions in the field of insurance, design a centralized decision support module, implement the first version of the decision support module, implement decision support functionality for the case of risk evaluation and compare the performance of different models for the specific business case. The following tasks were completed: completed analysis of three decision support systems in the insurance field, a centralized decision support module was designed and its first version with classification and new model registering functionality implemented, decision support functionality for the case of risk evaluation was realised with low performance for the minority class.

Keywords: decision support system#1, risk evaluation#2, machine learning#3

1. ĮVADAS

1.1. Temos aktualumas

Šio baigiamojo darbo ataskaitos tema bei problema buvo suformuluota autoriui dirbant sveikatos bei gyvybės draudimo įmonėje.

Svarbiausias draudimo įmonės subjektas – draudimo sutartis. Pasirašant gyvybės draudimo sutartį nustatoma draudėjo pageidaujama draudimo suma, kuri naudos gavėjams išmokama įvykus draudžiamajam įvykiui, bei draudimo įmoka, kurią draudėjas kiekvieną mėnesį moka draudimo įmonei. Tačiau šis procesas ne visada toks paprastas - kartais draudimo įmonei dėl aukšto kliento rizikingumo tenka atsisakyti suteikti norimą draudimo sumą ar įmoką ir tenka šias reikšmes pakoreguoti.

Specialistai, atliekantys šį vertinimo darbą vadinami rizikos vertintojais, o darbas kurį jie atlieka vadinamas rizikos vertinimu (angl. *underwriting*). Viena iš priemonių, kurias naudoja gyvybės draudimo įmonės apdraustojo(-ųjų) rizikai nustatyti yra sveikatos anketa, leidžianti draudimo rizikos vertintojui atsižvelgti į šiuos duomenis nustačius aukštą rizikingumą pakoreguoti draudimo sumą ir/arba įmokos dydžius.

1.2. Problema

Draudimo rizikos vertintojai skiria didelę dalį dėmesio nagrinėdami itin paprastus atvejus ir minėti procesai vykdomi neefektyviai.

1.3. Tikslas

Palengvinti rizikos vertintojų vykdomus procesus bei ateityje kuriamų mašininio mokymosi sprendimų įdiegimą sukuriant centralizuotą sprendimų palaikymo sistemą, teikiančią rekomendacinio pobūdžio informaciją.

1.4. Uždaviniai

1. Išanalizuoti bei palyginti sprendimų palaikymo sistemų sprendimus draudimo rinkoje;
2. Suprojektuoti centralizuotą sprendimų palaikymo sistemos modulį;
3. Įgyvendinti pirmąja centralizuoto sprendimų priėmimo modulio versiją;
4. Realizuoti sprendimų palaikymo funkcionalumą rizikos vertinimo atvejui;
5. Palyginti įvairių sprendimų palaikymo modelių efektyvumą konkrečiam verslo atvejui.

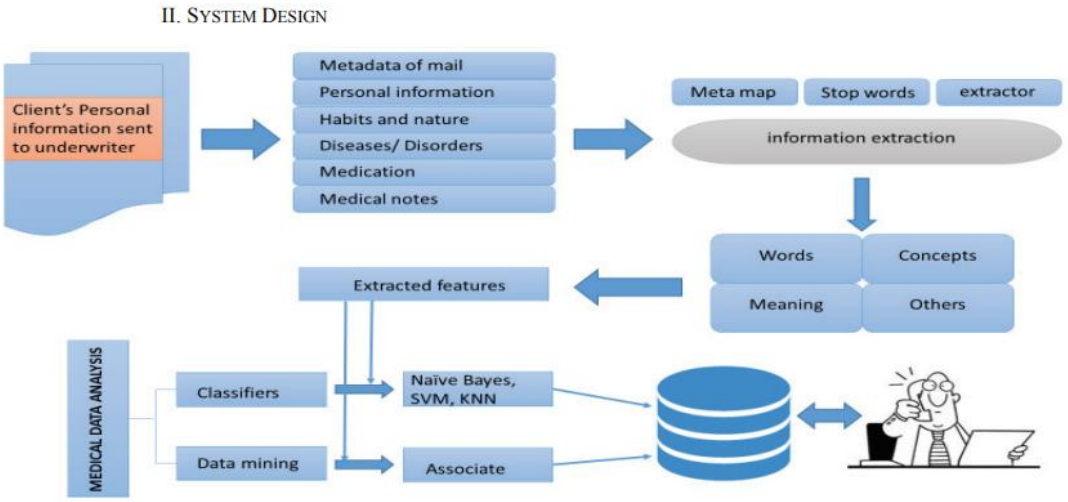
2. ANALITINĖ DALIS

2.1. Sprendimų palaikymo sistemos draudimo rinkoje

2.1.1. Išmaniosios rizikos vertinimo sistemos analizė

Pirmasis automatizuotos sprendimų priėmimų sistemos pavyzdys pasirinktas iš 2018-ais metais vykusios *International Conference for Convergence in Technology* konferencijos ištraukos [1]. Dėl milžiniško masto, įvairovės apdraustojo sveikatos duomenų, ligų bei vartojamų vaistų rizikos vertinimo procesas laikui bėgant tapo itin sudėtingas. Todėl poreikis sprendimus padedančioms priimti sistemoms (angl. *decision support systems*) išaugo.

Siūlomas sprendimus padedančios priimti sistemos modelis, remiantis prielaida, jog kliento informacija draudimo agentui ir/arba rizikos vertintojui siunčiama elektroniniu paštu (žr. pav. 2.1).



2.1 pav. „Išmanusis rizikos vertinimo procesas“ [1]

Esant nestruktūrizuotai kliento informacijai pagalbinei sprendimų priėmimo sistemai reikalingas atskiras savybių gavybos etapas (nuotraukoje *information extraction* bei *extracted features*). Iš elektroninio laiško(-ų) išgautos savybės perduodamos į klasifikatorių, rekomenduojantį tam tikrą draudimo planą, tiesiogiai susijusį su nustatyta kliento rizika. Modelio rekomenduojamą draudimo planą peržiūri draudimo rizikos vertintojas ir atitinkamai priima sprendimą.

Straipsnio autorių nagrinėjami trys klasifikatoriai: naivus Bajeso metodas (angl. *naïve Bayes),* SVM (atraminių vektorių klasifikatorius, angl. *support vector machine*) bei KNN (K-artimiausių kaimynų, angl. *K-nearest neighbors*). Atlikus eksperimentą su dvejais skirtingais duomenų rinkiniais gauti toliau lentelėje pateikti tikslumo įverčiai (žr. lentelė 2.1).

2.1 lentelė. Gautų rezultatų palyginimas. Sudaryta autoriaus pagal [1]

Naudotas algoritmas	Gautas tikslumas		
	Pirmas duomenų rinkinys	Antras duomenų rinkinys	Vidurkis
Naivus Bajeso	58 %	54 %	56 %
K-NN	68 %	70 %	69 %
SVM	80 %	78 %	79 %

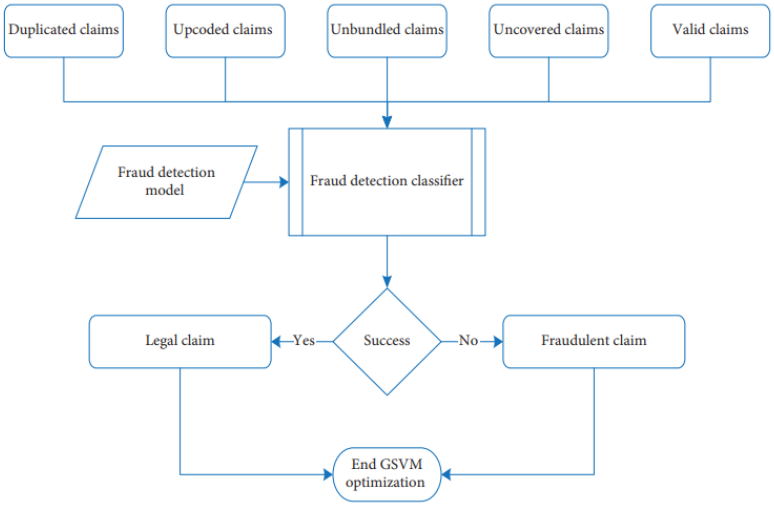
Pagal gautus kiekvieno iš trijų algoritmų tikslumo įverčius padaryta išvada, jog tiksliausiai duomenis klasifikavo *SVM* klasifikatorius – su apytiksliai 79 % vidutiniu klasifikavimo tikslumu, maždaug 10 % procentu daugiau nei antroje vietoje esantis *K-NN* klasifikatorius. Prasčiausias klasifikavimo tikslumo įvertis gautas su naiviu Bajeso algoritmu.

2.1.2. Sprendimų palaikymo sistemos draudimo žalų aptikimui naudojant genetinį atraminių vektorių klasifikatorių analizė

Antrasis pavyzdys siekia pateikti galimą sprendimą žalų vertinimo proceso automatizavimui pasinaudojus *GSVM* (angl. *genetic support vector machines*) klasifikatoriumi.

„*In the 1960s, John Holland invented genetic algorithms by involving a simulation of Darwinian survival of the fittest as well as the processes of crossover, mutation, and inversion that occurs in other genetics*“ [2], taigi GSVM iš esmės yra tas pats ankstesniame pavyzdyje minėtas SVM klasifikatorius, tik vietoj rankinio klasifikavimo parametrų parinkimo, naudojamas genetinis algoritmas, atkartojantis Darvinistinės evoliucijos „išgyvena stipriausi“ principą.

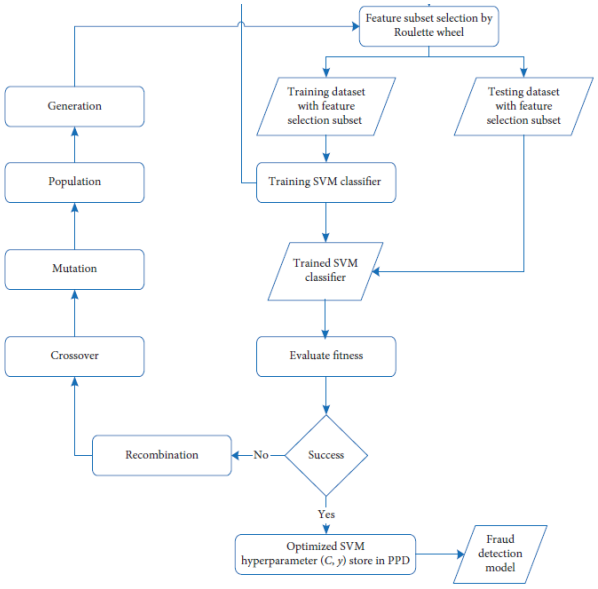
Toliau paveikslėlyje pateiktas koncepcinis žalų klasifikatoriaus modelis (žr. pav. 2.2).



2.2 pav. *DSS for Fraud Detection*, koncepcinis modelis [2]

Klasifikatorius priima žalos informaciją ir nustato jos klasę – ar tai yra teisėta (kurią galima patvirtinti), ar neteisėta (kurią reikėtų atmesti) žala.

Toliau esančioje GSVM apmokymo srauto diagramos iškarpoje (žr. pav. 2.3) matoma genetinio algoritmo dalis – šaka, prasidedanti nuo veiklos *Evaluate fitness* (gauto modelio tinkamumo įvertis) bei pasibaigianti su veikla *Generation*.



2.3 pav. *DSS for Fraud Detection*, *GSVM* apmokymo diagramos iškarpa [2]

Tinkamumo įvertis genetiniuose algoritmuose naudojamas nustatant, kurių natūralios atrankos dalyvių „genotipas“ pateks į sekančios kartos simuliaciją bei kurie individai išnyks. Šis procesas kartojamas N kartų, kol gaunamas optimizavimo algoritmo rezultatas - didžiausią tinkamumo įvertį turintis individas - *SVM* modelio parametrų rinkinys.

Straipsnio autoriai atliko eksperimentus su *GSVM*, sprendimų medžio bei naiviu Bajeso algoritmais, toliau lentelėje – jų gautų rezultatų palyginimas (žr. lentelė 2.2).

2.2 lentelė. Gautų rezultatų palyginimas. Sudaryta autoriaus pagal [2]

Naudotas algoritmas	**Duomenų rinkinys**	**Gautas tikslumas**	**Vidutinė vertė naudojant skirtingus duomenų rinkinius**
Genetinė vektoriaus palaikymo mašina su radialinės bazės funkcijos (RBF) branduoliu	100	71,43 %	87,906 %
	300	95,45 %	
	500	99,18 %	
	750	99,18 %	
	1000	82,56 %	
Sprendimų medis	100	62 %	74,44 %
	300	78 %	
	500	77,8 %	
	750	82,7 %	
	1000	71,7 %	
Naivus Bajeso	100	50 %	59,1 %
	300	61 %	
	500	56,8 %	
	750	60,7 %	
	1000	67 %	

Naudotas GSVM algoritmo tikslumas – 87,906 %, ~13 % aukštesnis nei antroje vietoje esančio pasirinkimų medžio algoritmo. Toliau formulėje pateiktas autorių naudotas tikslumo matas (žr.).

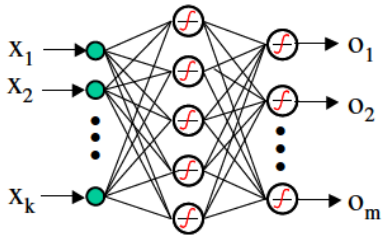
$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (2.1)$$

TP, *TN*, *FP*, *FN* – atitinkamai, tikrų teigiamų (angl. *true positive*), tikrų neigiamų (angl. *true negative*), netikrų teigiamų (angl. *false positive*) bei netikrų neigiamų (angl. *false negative*) rezultatų skaičius.

2.1.3. Automatizuotos draudimo rizikos vertinimo sistemos naudojant neuroninius tinklus analizė

Paskutinis pavyzdys automatinio sprendimų priėmimo draudimo rizikos vertinime susitelkia į neuroninių tinklų panaudojimą apdraustojo rizikos kategorijai nustatyti. Straipsnyje [3] pateikiamos keturios tiriamos neuroninio tinklo architektūros draudimo rizikos vertinimo uždaviniui spręsti.

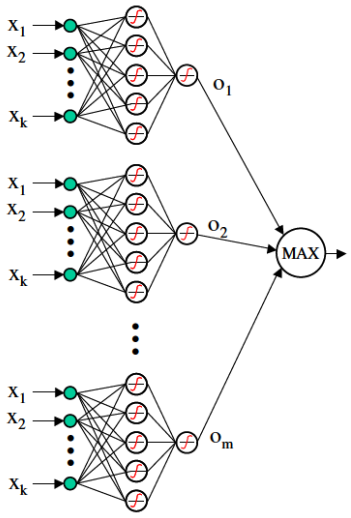
Pirmoji architektūra *D1* – paprastas vienetinis neuroninis tinklas su *m* išvesčių, kur *m* – klasių skaičius (žr. pav. 2.4).



2.4 pav. Architektūra D1, vienetinis neuroninis tinklas [3]

Kiekvienas tinklo išvesties mazgas naudoja sigmoidinę aktyvavimo funkciją, gauta klasė – jai atitinkančio išvesties mazgo aukščiausia vertė.

Antroji architektūra *D2* – kelių vienetinių neuroninių tinklų struktūra naudojant kategorines klases, apmokami *m* neuroniniai tinklai, kur *m* – klasių skaičius (žr. pav. 2.5).



2.5 pav. Architektūra D2, keli neuroniniai tinklai su kategorinėmis klasėmis [3]

Kiekvieno tinklo rezultatas – tikimybė, jog atvejis priklauso klasei m_i. Galutiniam rezultatui nustatyti naudojamas metodas *one-vs-other* – pasirenkama klasė su didžiausia tikimybės verte.

Trečioji architektūra *D3* – kelių neuroninių tinklų architektūra naudojant kelintines (angl. *ordinal*) klases, panaši į *D2*, tik šįkart naudojami *m-1* neuroniniai tinklai, kur neuroninio tinklo NN_i išvestis – tikimybė, jog įvestis priklauso aukštesnei klasei nei C_i (žr. lentelė 2.3).

2.3 lentelė. Architektūra D3, 5-ių klasių užkodavimas į 4-is dvejetainius klasifikatorius. Sudaryta autoriaus pagal [3]

		Klasė				
		C_1	**C_2**	**C_3**	**C_4**	**C_5**
Binarinis klasifikatorius	**NN_1**	0	1	1	1	1
	NN_2	0	0	1	1	1
	NN_3	0	0	0	1	1
	NN_4	0	0	0	0	1

Lentelėje raudono atskyrimo dešinėje pusėje – klasės, kurių tikimybė klasifikatoriaus yra nustatoma. Algoritmo išvestis – klasė su didžiausia tikimybe $P(C_i)$, kuri gaunama:

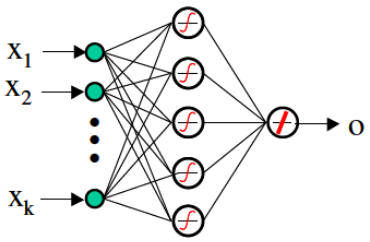
$$P(C_i) = O_{i-1} - O_i, \tag{2.2}$$

$$1 < i \ < m, \tag{2.3}$$

O_i - tinklo C_i išvestis, m – klasių skaičius.

Ši architektūra labiausiai tinka klasėms, kurias galima išrikiuoti tam tikra tvarka, rizikos kategorijos yra vienas tokių atvejų. Taip pat naudojamų tinklų skaičius yra vienu mažesnis, todėl teoriškai tokio tinklo architektūros apmokymas turėtų užtrukti mažesnį laiko tarpą.

Ketvirtoji architektūra *D4* – regresinis neuroninis tinklas. Pagrindinis skirtumas tarp architektūros *D1*, jog išvestis tik viena bei išvesties mazgui naudojama tiesinė funkcija vietoj sigmoidinės. Toliau pateiktas supaprastintas regresinio neuroninio tinklo modelis (žr. pav. 2.6).



2.6 pav. Gautų rezultatų palyginimas, klasifikavimo įverčiai [3]

Kiekvienai klasei priskiriama sveikojo skaičiaus reikšmė tarp 0 ir *m*, kur *m* – klasių skaičius. Regresinis neuroninis tinklas apmokamas siekiant sumažinti skirtumą tarp išvesties bei iš anksto žinomos klasės reikšmių. Algoritmo išvestis – klasė m_i su mažiausiu skirtumu tarp jai priskirto skaičiaus bei tinklo išvesties.

Su kiekviena neuroninio tinklo architektūra autoriai atliko eksperimentus naudojant sluoksninę kryžminę patikrą (angl. *stratified cross-validation*), kiekvienai tinklo architektūrai eksperimentą pakartojant 10 kartų ir išmatavo dvejas metrikas: vidutinį bendrą klaidų dažnį (angl. *overall error rate*, sutrumpintai *OER*), vidutinę neteisingo klasifikavimo kainą (angl. *misclassification cost*, sutrumpintai *MC*) bei abiejų šių metrikų standartinį nuokrypį. Toliau pateiktoje lentelėje matomi eksperimentiškai gauti rezultatai (žr. lentelė 2.4).

2.4 lentelė. Gautų rezultatų palyginimas. Sudaryta autoriaus pagal [3]

Projektas	**Bendras klaidų dažnis (OER)**		**Neteisingo klasifikavimo kaina (MC)**		**Klasifikavimo tikslumas (1 – OER)**
	μ	**σ**	**μ**	**σ**	**μ**
D1	15,125 %	5,394 %	205,353	132,283	84,875 %
D2	8,021 %	1,762 %	93,667	17,904	91,979 %
D3	7,408 %	0,448 %	83,458	6,799	92,592 %
D4	7,403 %	0,280 %	80,939	3,682	92,597 %

Pateikta lentelė yra pakeista pridedant papildomą stulpelį „Klasifikavimo tikslumas“ su tikslu patogiau palyginti buvusių pavyzdžių rezultatus. Iš gautų duomenų galima spręsti, jog architektūra *D4* buvo tiksliausia iš visų keturių rizikos vertinimo uždaviniui spręsti, taip pat su mažiausia neteisingo klasifikavimo kaina, iškart po jos su artimu klasifikavimo tikslumu yra architektūra *D3*.

Taip pat šaltinio autorių išmatuotas procesoriau darbo laikas atliekant kryžminę patikrą. Toliau lentelėje pateikti gauti procesoriau darbo laiko duomenys (žr. lentelė 2.5).

2.5 lentelė. Gautų rezultatų palyginimas, greitaveika. Sudaryta autoriaus pagal [3]

Projektas	**Procesoriaus darbo laikas, sek.**
D1	268,8
D2	37,2
D3	26,2
D4	9,1

Remiantis autorių gautai duomenimis galima teigti, jog regresinio neuroninio tinklo modelis yra ne tik tiksliausias bei ir sparčiausias rizikos vertinimo uždaviniams spręsti procesoriaus darbo laiko atžvilgiu.

2.1.4. Kitų autorių modelių palyginimas

Anksčiau pateiktų autorių modelių palyginimui nuspręsta naudoti šias metrikas:

- Uždavinys – kokį uždavinį autorius sprendė pasirinktu modeliu (rizikos vertinimo ar žalų sprendimo priėmimo);
- Realizavimo sudėtingumas – laipsnis, nusakantis modelio realizavimo sudėtingumą;
- Parametrų skaičius – kiek parametrų galima keisti norint gauti gerus modelio apmokymo rezultatus;
- Galimas klasių skaičius – kelių klasių nustatymo uždavinį galima spręsti su tam tikru modeliu (binarinis ir/arba daugiau nei 2-ų klasių uždavinys);
- Klasifikavimo tikslumas – modelį realizavusių autorių gautas klasifikavimo tikslumas.

Realizavimo sudėtingumui nustatyti autorius nusprendė naudoti taškais paremtą metriką. Taškų skaičius priklauso nuo modelio parametrų skaičiaus bei nuo to ar bent vienas iš trijų mašininio mokymosi karkasų leidžia realizuoti šį modelį bei jokių papildomų modifikacijų.

Karkasai pasirinkti pagal populiarumą (atsisiuntimų skaičių) bei įmonėje esančių programuotojų kiekį, turinčių patirties su programavimo kalba, su kuria karkasas yra realizuotas. Toliau pateiktos pasirinktos bibliotekos bei pasirinkimo paaiškinimas:

- *TensorFlow* – karkasas pasirinktas dėl nesunkiai išmokstamos *Python* programavimo kalbos bei apmokytų modelių įdiegimo funkcijų realaus naudojimo aplinkoje;
- *PyTorch* – pasirinktas taip pat dėl *Python* programavimo kalbos bei dėl spartaus modelių apmokymo greičio;
- *ML.NET* – mašininio mokymosi karkasas sukurta naudojantis *.NET* technologija. Darbovietėje plačiai naudojamos *.NET* technologijos, todėl šis karkasas puikiai tiktų dėl lengvesnio tolimesnio programinės įrangos palaikymo bei vystymo.

Toliau pateiktoje lentelėje nurodytos galimi realizavimo sudėtingumo įverčiai bei kokia kiekvieno įverčio reikšmė (žr. lentelė 2.6).

2.6 lentelė. Realizavimo sudėtingumo įverčio metrika

Realizavimo sudėtingumo įvertis	**Reikšmė**
0	Modelio parametrų skaičius <= 3 bei realizuojamas bent viename iš trijų karkasų
1	Modelio parametrų skaičius >3, <= 5 bei realizuojamas bent viename iš trijų karkasų
2	Modelio parametrų skaičius > 5 bei realizuojamas bent viename iš trijų karkasų
3	Modelis nerealizuotas nei viename iš trijų karkasų

Metrika sudaryta taip, kad galima būtų tarpusavyje palyginti modelius su besiskiriančiu parametrų skaičiumi, bei kad modelis, kurio neįmanoma realizuoti be rankinių pakeitimų viename iš trijų karkasų, turėtų maksimalų sudėtingumo lygį nepriklausomai nuo parametrų skaičiaus.

Toliau pateiktoje lentelėje matoma autoriaus sudaryta lentelė nagrinėtų pavyzdžių geriausių modelių palyginimui (žr. lentelė 2.7).

2.7 lentelė. Nagrinėtų modelių pavyzdžių palyginimas

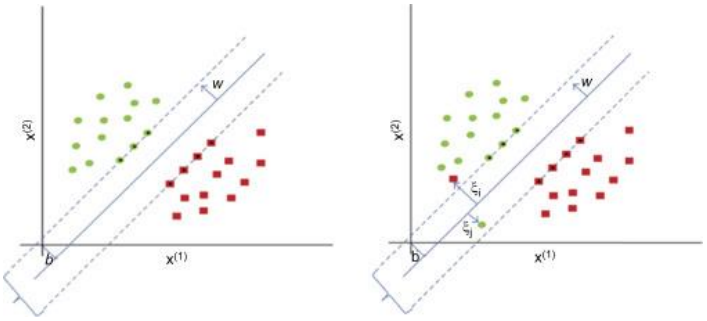
Modelis	**Uždavinys**	**Realizavimo sudėtingumas**	**Parametrų skaičius**	**Galimas klasių skaičius**	**Klasifikavimo tikslumas**
Vektoriaus palaikymo mašina (SVM)	Rizikos vertinimas	0	3	2	79 %
Genetinė vektoriaus palaikymo mašina (GSVM)	Atmestinų žalų aptikimas	3	>6	2	87,906 %
Regresinis neuroninis tinklas	Rizikos vertinimas	2	6	∞	**92,597 %**

Remiantis lentelės duomenimis galima teigti, jog tiksliausias tinklo modelis rizikos vertinimo uždaviniams spręsti yra regresinis neuroninis tinklas, turintis ~13 % aukštesnį klasifikavimo tikslumą nei vektoriaus palaikymo mašinos modelis tam pačiam uždaviniui spręsti. Paprasčiausia realizuoti vektoriaus palaikymo mašinos bei regresinio neuroninio tinklo modelius, kadangi abu šie modeliai yra palaikomi visų trijų pasirinktų mašininio mokymosi karkasų.

2.2. Sprendimo priėmimo metodai

2.2.1. Vektoriaus palaikymo mašina

Vektoriaus palaikymo mašina (angl. *support vector machine*) yra mašininio mokymosi modelis, kurio siekis yra nustatyti hiperplokštumą, skiriančią dvejas ar daugių klasių. Toliau pateiktame paveikslėlyje matomos galimų tokių plokštumų vizualizacija dvimatėje erdvėje (žr. pav. 2.7).



2.7 pav. *SVM* hiperplokštumos [4]

Vektoriaus palaikymo mašinos tikslumas po apmokymo priklauso nuo dviejų parametrų: naudojamo branduolio, dažniausiai tiesinis, *RBF* arba daugianaris, bei parametro *C*, nusakančio modelio griežtumą, kuo didesnė *C* reikšmė, tuo mažesnė galima marža tarp hiperplokštumos ir artimiausio duomens taško, tuo tikslesnis modelis apmokymo duomenims, tačiau galimai turintis mažesnį tikslumą su realiais duomenimis.

Sprendžiant netiesinius klasifikavimo uždavinius naudojamas netiesinis branduolys (*RBF*, daugianaris ar kitas), taikantis integralinę transformaciją, taip transformuodamas originalią požymių erdvę. Transformuotoje erdvėje vėlgi ieškoma optimalioji hiperplokštuma.

2.2.2. Regresiniai neuroniniai tinklai

Regresiniai neuroniniai tinklai – vienas paprasčiausių neuroninių tinklų tipų. Vietoje įprastos tikimybinės išvesties mazgo reikšmės (tarp 0 ir 1), naudojant tiesinę aktyvacijos funkciją galima gauti bet kokią realaus skaičiaus reikšmę.

Regresiniai tinklai dažniausiai naudojami prognozavimo uždaviniams spręsti, kadangi jų išvestį galima prilyginti realiam fizikiniam matavimui, pavyzdžiui oro temperatūrai, kraujospūdžiui ir t.t. Tačiau šį tinklą taip pat galime panaudoti kelių klasių klasifikavimo uždaviniui spręsti, kuomet yra galimybė kiekvienai klasei priskirti reikšmingą skaitinę reikšmę.

Darbe aprašytame kliento rizikos vertinimo atvejyje treji rizikos lygmenys – žemas, vidutinis bei aukštas rizikingumo lygmuo, kiekvienai iš klasių galima priskirti skaitinę reikšmę: atitinkamai 0, 1 ir 2.

2.2.3. *LightGBM*

Mašininio mokymosi algoritmas *LightGBM* yra populiaraus gradiento didinimo sprendimų medžio (toliau naudojamas angl. trumpinys *GBDT*) algoritmo [6] realizacija.

GBDT algoritmas veikia sukurdamas pasirinkimų medį bei apmokydamas jį duomenimis, tuomet peržiūrimos sudaryto pasirinkimų medžio spėjimo klaidos ir iš jų sukuriamas dar vienas pasirinkimų medis. Šis procesas kartojamas kol gaunamas galutinis sprendimų medis. Nuostolio funkcija (angl. *loss function*) *GBDT* algoritme optimizuojama naudojant gradientinio nusileidimo metodą.

LightGBM algoritmas siekia padidinti mokymosi efektyvumą bei plečiamumą (požymių skaičiaus atžvilgiu) [5]. Algoritmas tai pasiekia pašalindamas požymius mažomis gradiento reikšmėmis bei apjungdamas viena kitą išskiriančią požymių porą į vieną požymį. *LightGBM* algoritmu gaunamas iki virš 20 kartų greitesnis mokymasis, pasiekiant beveik tokį patį tikslumą [5].

2.3. Analitinės dalies išvados

Remiantis draudimo rinkoje esančių sprendimų palaikymo sistemų analize pasirinkta naudotis regresinio tinklo modeliu bei autoriaus pasirinktais modeliais palyginimui: atraminių vektorių klasifikatoriumi bei *LightGBM* algoritmu. Šiems metodams realizuoti pasirinkta naudotis *RStudio* programine įranga pradiniam duomenų apdorojimui bei *ML.NET* biblioteką modelių apmokymui bei testavimui.

3. METODINĖ DALIS

3.1. Sprendimų priėmimo modulio projektavimas

Sprendimų priėmimo modulio projektavimo vyko toliau pateikta tvarka:

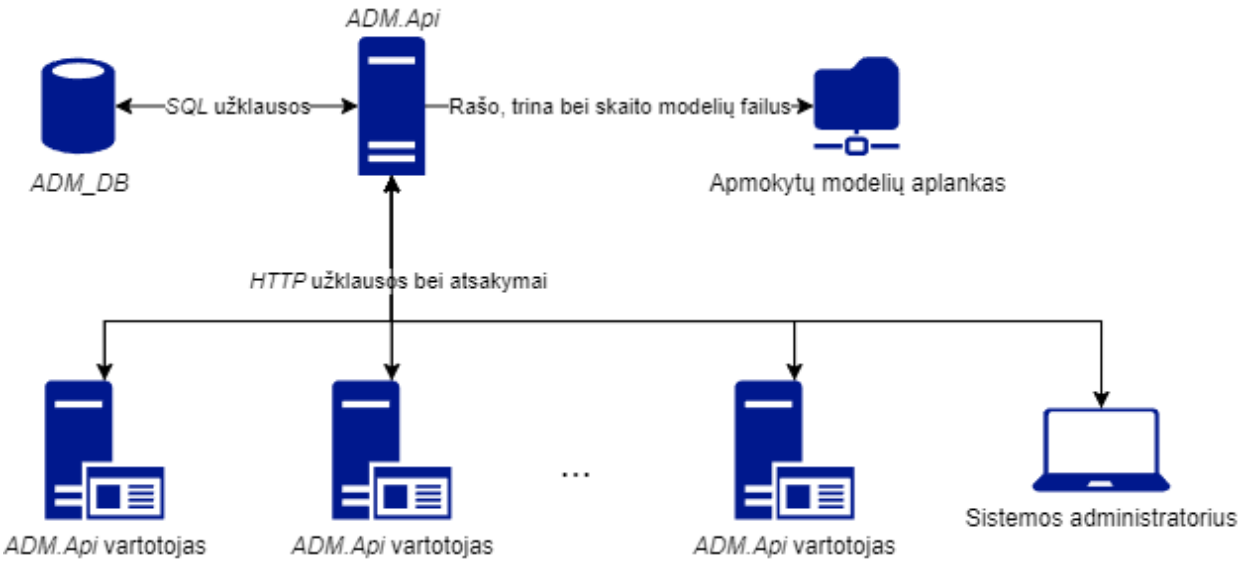
1. Aukšto lygio architektūros sukūrimas;
2. Galutinių taškų projektavimas;
3. Duomenų bazės schemos sudarymas.

3.1.1. Aukšto lygio architektūros sukūrimas

Kuriant aukšto lygio architektūrą buvo nuspręsta sprendimų priėmimo sistemą padalinti į tokias dalis:

- *ADM.Api* – programinė sąsaja, leidžianti gauti tam tikro apmokyto modelio rezultatą, prognozuojamą vertę, permokyti modelį su naujais duomenimis, jį ištrinti bei atnaujinti;
- *ADM_DB* - duomenų bazė saugojanti meta duomenis apie *ADM.Api* vartotojus, jiems pasiekiamus modelius bei modelių versijų istoriją;
- Apmokytų modelių aplankas – aplankas, kuriame talpinami visi apmokyti mašininio mokymosi modeliai bei ankstesnės jų versijos.
- *ADM.Api* vartotojas - sistema, naudojanti *ADM.Api* programinę sąsają;
- Sistemos administratorius – fizinis asmuo ar sistema, atliekanti(-is) modelių atnaujinimus ar kitus ne su prognozavimu susijusius administracinius veiksmus.

Toliau pateiktame paveikslėlyje matoma sukurta aukšto lygio diagrama (žr. pav. 3.1).



3.1 pav. *ADM.Api* aukšto lygio diagrama

Diagramoje daroma prielaida, jog *ADM.Api* sąsaja paremta *HTTP* protokolu, o *ADM_DB* – *SQL* duomenų bazė.

3.1.2. Galutinių taškų projektavimas

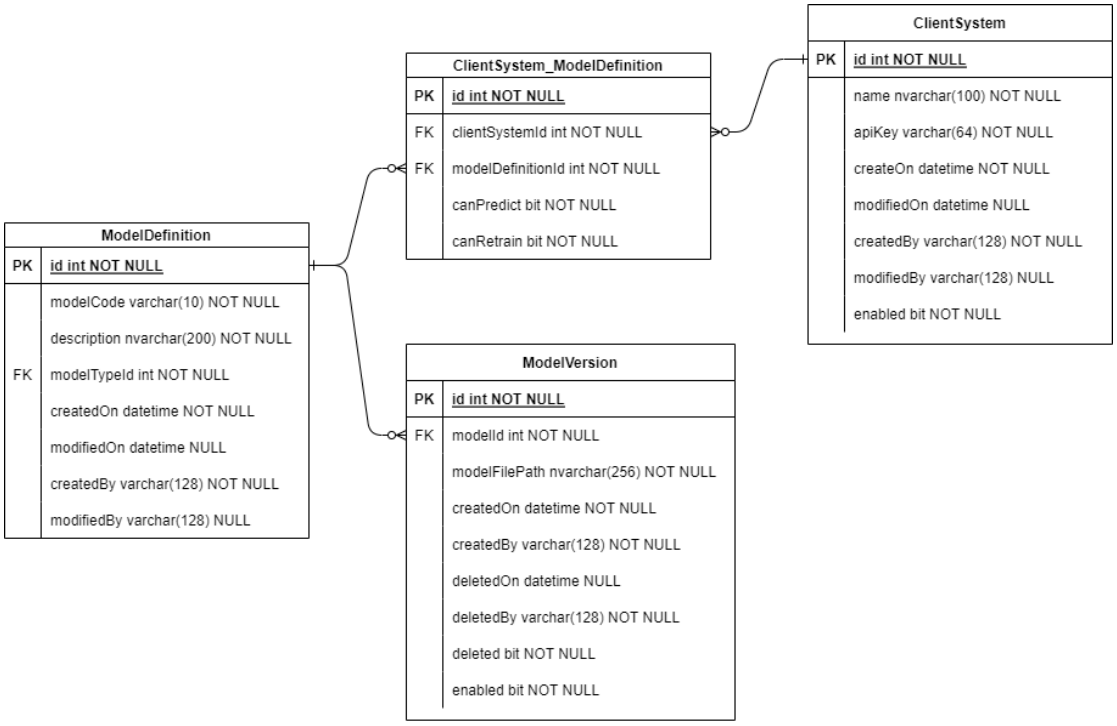
Galutinius taškus nuspręsta projektuoti vadovaujantis *REST* architektūriniais principais. Toliau pateikti suprojektuoti *ADM.Api* programinės sąsajos galutiniai taškai, kuriuos nuspręsta padalinti į tris grupes:

1. Priimto sprendimo gavimas
    a. *POST /api/models/{modelCode}/predict* – gaunama modelio apskaičiuota prognozuojama vertė ar klasė.
2. Modelių apibrėžčių valdymas
    a. *GET /api/models/definitions* – gaunamas visų modelio apibrėžčių sąrašas;
    b. *POST /api/models/definitions* – sukuriama nauja modelio apibrėžtis;
    c. *DELETE /api/models/definitions/{definitionId}* – ištrinama pasirinkta modelio apibrėžtis pagal unikalų identifikatorių.
3. Modelių versijų valdymas
    a. *GET /api/models/{modelCode}/versions* – gaunamas sąrašas egzistuojančių esamų modelio versijų;
    b. *POST /api/models/{modelCode}/versions/{versionId}/apply* – pasirinkta modelio versija tampa pagrindine darbine versija, kuria vykdomos visos tolimesnės prognozės ar klasifikavimai;
    c. *DELETE /api/models/{modelCode}/versions/{versionId}* - pasirinkta modelio versija ištrinama.

Galutiniuose taškuose minimas parametras *{modelCode}* – modelio kodas skirtas didesniam suprantamumui programinę sąsają naudojančiose sistemose.

3.1.3. Duomenų bazės schemos sudarymas

Suprojektuotų galutinių taškų funkcionalumui įgyvendinti sudaryta toliau paveikslėlyje pateikta duomenų bazės schema (žr. pav. 3.2).



3.2 pav. *ADM_DB duomenų bazės schema*

Pagrindiniai duomenų bazės schemos objektai: modelis (schemoje *Model*) bei vartotojas (schemoje *ClientSystem*). Vartotojui priskiriami jam/jai pasiekiami modeliai naudojant lentelę *ClientSystem_ModelDefinition*, tuo pačiu nustatomi ir galimi veiksmai (*predict* bei *retrain*).

3.2. Automatizuotas sprendimų priėmimas rizikos vertinimo atvejui

3.2.1. Mokymuisi naudojamų duomenų pasirinkimas

Duomenys, kurie bus naudojami modelio apmokymui yra sąjunga šių duomenų: užpildyto kliento sveikatos klausimyno, gyvybės draudimo sumos, periodinės įmokos dydžio, įmokos dažnio bei pradinio įnašo sumų eurais prieš sveikatos duomenų įvertinimą jau galutinai pasirašyto sutarties fakto metu.

Toliau esančioje lentelėje pateikta duomenų rinkinio, kuris po pradinio apdirbimo bus vėliau naudojamas modelio apmokymui, požymių dalis (žr. lentelė 3.1).

3.1 lentelė. Pradinio duomenų rinkinio požymių aibės dalis

Įrašo pavadinimas	Paaiškinimas	Duomenų tipas
Age	Amžius, metai	Sveikasis skaičius
Height	Ūgis, centimetrai	Sveikasis skaičius
...	...	...
Diabetes	Ar sergama diabetu	Dvejetainis skaičius
...	...	...
LeftEye	Nešiojamų arba paskirtų akinių ar lęšių stiprumas kairei akiai, dioptrijos	Tekstas

...	...	...
LifeSumReq	Pageidautina gyvybės draudimo suma	Sveikasis skaičius
...	...	...
PayAct	Faktinė mokėjimo suma	Skaičius

Pasirinkta didžioji dalis kliento sveikatos duomenų, išskyrus tekstinius ligų, žalingų įpročių ar kt. paaiškinimus, kadangi atvejų dalis su pažymėtais (t. y. klientas serga tam tikra liga ar turi žalingą įprotį) apklausos variantais yra maža (mažiau nei 5 % atvejų).

3.2.2. Prognozuojamos klasės sudarymas

Kadangi pradiniai duomenys neturi konkrečios klasės ar nuspėjamos reikšmės (rizikos vertintojai vertinimo metu klientui nepriskiria rizikos kategorijos), šią kategoriją iš turimų duomenų reikia sukonstruoti. Pasirinkta sukurti dvi klases: klasę A (žymima dvejetaine reikšme 0), kuriai priskiriami klientui kurių draudimo sumos bei įmokos dėl pakankamai gerų sveikatos duomenų keisti nereikia, bei klasę B (žymima dvejetaine reikšme 1), kurią pamatęs rizikos vertintojas turėtų atidžiau peržiūrėti kliento sveikatos anketą.

Kiekvienas duomenų įrašas priskiriamas prie klasės A arba klasės B pagal toliau pateiktus žingsnius:

1. Sukonstruojamas naujas stulpelis pavadinimu *Multiplier* pagal toliau pateiktas bei autoriaus sukurtas formules:

$$MonthlyPayReq = \frac{InitPayReq}{PeriodReq} + PayReq, \tag{3.1}$$

$$MonthlyPayAct = \frac{InitPayAct}{PeriodAct} + PayAct, \tag{3.2}$$

$$Multiplier = MAX\left(\frac{MonthlyPayAct}{MonthyPayReq} \times \frac{LifeSumReq}{LifeSumAct}, 1\right), \tag{3.3}$$

2. Įrašui sukuriamas naujas stulpelis *Class* ir jam priskiriama reikšmė:

$$Class = \begin{cases} 0, & Multiplier = 1 \\ 1, & Multiplier < 1 \end{cases}, \tag{3.4}$$

Į *Multiplier* stulpelio skaičiavimą įtraukta tiek draudimo suma, tiek vidutinė mėnesinė įmoka *MonthlyPay* kadangi dėl aukšto kliento rizikingumo draudimo suma gali būti sumažinta arba mėnesinės įmokos dydis padidintas. *Multiplier* formulė sudaryta taip, kad reikšmė liktų lygi 1-am, jeigu draudimo suma sumažėtų ar padidėtų tiek pat kiek mėnesinė įmoka, kadangi draudimo bendrovės atžvilgiu šis pokytis būtų nereikšmingas.

3.2.3. Pradinis duomenų apdorojimas

Pradinio duomenų apdorojimo metu bus atliekami šie žingsniai:

- Įrašų su tuščiomis reikšmėmis šalinimas arba naujų reikšmių sintezė;
- Išskirčių bei uždaviniui neaktualių reikšmių šalinimas;

- Prognozuojamos reikšmės sukūrimas;
- Reikšmių normalizavimas (išskyrus modelius, naudojančius medžio algoritmą);
- Įrašų padalinimas į testavimo bei apmokymo duomenų rinkinius atitinkamai santykiu 30:70;
- Mažumos klasės atveju, naujų įrašų sukūrimas apmokymo duomenų rinkinyje;

3.2.4. Apmokyto modelio testavimas

Apmokyto modelio testavimui pasirinktos metrikos dvejetainius klasifikavimo uždavinius uždavinius sprendžiantiems modeliams įvertinti. Toliau pateiktos pasirinktos metrikos:

- Tikslumas – santykis teisingų spėjimų bei visų spėjimų (bendras modelio tikslumas vienam atsitiktiniam įrašui);
- *AUC* (angl. *area under the curve*) – metrika nurodanti tikimybę, jog modelis įvertins teigiamos klasės (šiuo atveju klasę *B*) įrašą aukštesne užtikrintumo verte nei neigiamos klasės įrašą (klasę *A*).
- *AUPRC* (angl. *area under the precision-recall curve*) – metrika, panaši į *AUC* ploto kreivės skaičiavimo principu, tačiau taip pat naudojanti tikslumo (angl. precision) bei *recall* metrikas kreivei apskaičiuoti. Ši metrika pasirinkta, nes tikimasi, jog duomenų rinkinys nebus subalansuotas;
- *F1* - metrika, gaunama apskaičiuojant harmoninį vidurkį modelio tikslumo bei *recall* įverčių.

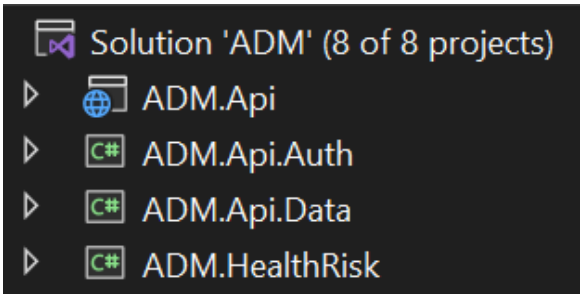
4. ĮGYVENDINIMO DALIS

4.1. Sprendimų priėmimo modulio programavimas

ADM.Api projektas padalintas į keturis subprojektus:

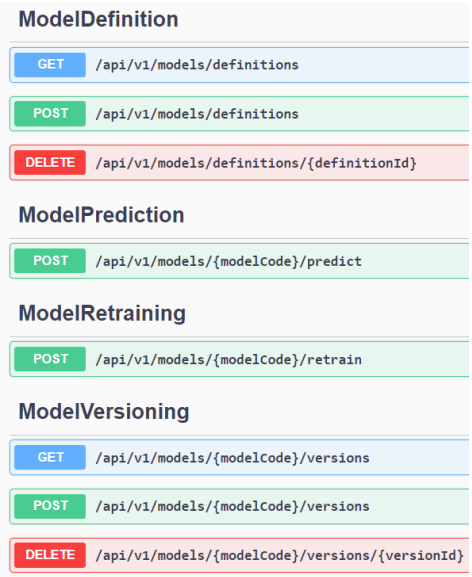
- *ADM.Api* – programinės sąsajos kodas: galutiniai taškai, programinės sąsajos raktų filtrai, modelių atvaizdavimas (angl. *mapping*) bei pagrindiniai servisai klasifikavimo funkcionalumui įgyvendinti;
- *ADM.Api.Auth* – servisai susiję su autorizacijos funkcionalumu, tikrinama ar užklausos siuntėjas turi prieigą vykdyti klasifikavimą norimame modelyje;
- *ADM.Api.Data* – servisai, suteikiantys sąsają su duomenų baze, duomenų gavimo bei trynimo operacijos;
- *ADM.HealthRisk* – realizuoto sprendimų palaikymo modelio įvesčių bei išvesčių modelių aprašymas.

Toliau paveikslėlyje pateiktas vaizdas aprašytų modelių sąrašas Visual Studio programoje (žr. pav. 4.1).



4.1 pav. *ADM.Api* subprojektų sąrašas *Visual Studio* programoje

Visi įgyvendinti galutiniai taškai pasiekiami per naršyklėje pasiekiamą *Swagger* programos naudotojo sąsaja (žr. pav. 4.2). Galutinių taškų pavadinimai sukurti laikantis *REST* programinės sąsajos standartų.



4.2 pav. *ADM.Api* galutinių taškų sąrašas *Swagger* naudotojo sąsajoje

Konkretus modelis prie ADM.Api priregistruojamas tiek per galutinį tašką - */models/definitions*, tiek programiškai sukuriant įvesties ir išvesties klases (pavyzdyje

HealthRisk.ModelInput bei *HealthRisk.ModelOutput*) pridedant kodo eilutę *Startup* klasės faile (žr. pav. 4.3).

```
services.AddPredictionEnginePool<HealthRisk.ModelInput, HealthRisk.ModelOutput>()
    .FromFile(aiModelsOptions.HealthRiskModelPath, true);
```

4.3 pav. *ADM.Api* naujo modelio registravimas programos kode

Tuomet šis modelis tampa pasiekiamas pagrindinėje klasifikavimo klasėje – *MultiModelPredictionEngine* ir susiejamas su konkrečia modelio apibrėžtimi.

Dabartinėje programinės sąsajos versijoje nauji sistemos naudotojai registruojami tiesiogiai duomenų bazėje, o teisės priskiriamos sukuriant atitinkamus įrašus lentelėje *ClientSystem_ModelDefinition*.

Kartu su paprastais programinės sąsajos naudotojais realizuota ir administratoriaus rolė. Dabartinėje versijoje administratoriaus programinės sąsajos raktas laikomas programos konfigūracijų faile, tik turint administratoriaus raktą galima kurti, trinti bei peržiūrėti esamus modelių apibrėžimus bei įkelti ir trinti naujas modelio versijas.

4.2. Automatizuoto sprendimų priėmimo realizavimas rizikos vertinimo atvejui

4.2.1. Duomenų paruošimas

Duomenys gaunami vykdant *SQL* užklausas į dvejas skirtingas duomenų bazes: vienoje informacija apie sveikatos duomenis bei pageidaujamą draudimo suma, įmoką bei pradinį įnašą, kitoje – po galimos rizikos vertintojo korekcijos galutinės draudimo sumos, įmokos bei pradinio įrašo reikšmės.

Toliau pateikta *SQL* užklausos pradžios bei pabaigos iškarpos (žr. pav. 4.4).

```
SELECT
    [Age] = Ins.[Age]
    ,[Height] = InsHS.[Height]
    ,[Weight] = InsHS.[Weight]
    ,[RespiratoryDiseases] = ISNULL(InsHS.[RespiratoryDiseases], 0)
    ,[CardiovascularDisease] = ISNULL(InsHS.[CardiovascularDisease], 0)
    …
…
WHERE
    Ins.[Age] IS NOT NULL AND
    InsHS.[Height] IS NOT NULL AND
    InsHS.[Weight] IS NOT NULL AND
    Req.[LifeSum] > 0 AND
    Req.[Deposit] > 0
```

4.4 pav. Rizikos vertinimo SQL užklausa

Kode matoma kaip kiekvienam požymiui, galinčiam įgyti *NULL* reikšmę, yra taikoma sąlyginė transformacija naudojant funkciją *ISNULL* – kiekvienai tuščiai reikšmei priskiriama reikšmė *0*. Nuspręsta šią dalį pradinio duomenų apdorojimo palikti *SQL* užklausai kadangi scenarijų kalboje tai yra nesudėtingai įgyvendinama. Scenarijaus pabaigoje (pradedant nuo žodžio *WHERE*) taikomas *IS NOT NULL* filtras *Age*, *Height* bei *Weight* požymiams, kad būtų praleistos visos anketos, kuriose šie duomenys dėl tam tikrų priežasčių nebuvo supildyti.

Užklausa įvykdyta naudojant įrankį *Microsoft SQL Server Management Studio*, o užklausos rezultatas išsaugotas *CSV* (angl. *comma separated values*) failo formatu, šio failo išvesties pavyzdys pateiktas toliau esančioje lentelėje (žr. lentelė 4.1).

4.1 lentelė. Rizikos vertinimo SQL užklausos rezultatas

Eilės nr.	***Age***	**...**	***Epilepsy***	**...**	***LifeSumReq***	***LifeSumAct***	**...**	***PayReq***	***PayAct***
1	2	...	0	...	10000	10000	...	10	10
2	43	...	1	...	35000	25000	...	25	18
3	36	...	0	...	40000	40000	...	40	40
4	33	...	0	...	50000	50000	...	50	50

Bendras duomenų rinkinio dydis – ~8000 įrašų. Gautas CSV failas įkeliamas į *RStudio* aplinką naudojant standartines *R* programavimo kalbos bibliotekas ir vykdomas pradinis duomenų apdorojimas.

Visų pirma sutvarkomos *LeftEye* bei *RightEye* požymių reikšmės, kadangi jų tipas duomenų bazėje – tekstinė eilutė, o reikšmę klientas įveda rankiniu būdu. Esant teigiamoms dioptrijų reikšmėms pašalinamas „+“ simbolis, o tuomet pašalinamos visos reikšmės, kurių neįmanoma interpretuoti kaip skaičių naudojant paieškos reiškinį (angl. *regex*).

Toliau pagal prognozuojamos klasės sudarymo poskyryje pateiktas formules (žr. formulės 3.1, 3.2, 3.3) atitinkamai sukuriami nauji stulpeliai: *MonthlyPayReq, MonthlyPayAct, Multiplier* bei *Class*. Visiems įrašams su *Multiplier* reikšme lygia „1” priskiriama *Class* reikšmė „0”, o visiems likusiems – reikšmė „1”.

Sekantis apdorojimo žingsnis – išskirčių šalinimas. Išskirtys šalinamos tik 9-iems iš 93-ijų požymių, kadangi likę požymiai – dvejetainiai skaičiai. Toliau pateiktos dvi išskirčių šalinimo strategijos bei požymiai, kurioms šios strategijos pritaikytos:

- Išskirčių šalinimas naudojant tarpkvartilinį intervalą – *Age, Height, Weight, LifeSumReq, LifeSumAct, MonthlyPayReq, MonthlyPayAct*;
- Verčių, ties arba mažiau 0,01 procentilės arba daugiau 0,99 procentilės šalinimas – *LeftEye, RightEye*.

Pasirinkta mažiau jautri išskirtims strategija regos požymiams, kadangi buvo pastebėta, jog taikant tarpkvartilinę išskirčių šalinimo strategiją šalinamų įrašų kiekis buvo 10-20 kartų didesnis nei su visais kitais požymiais.

Pašalinus išskirtis duomenų rinkinys padalinamas į testavimo bei apmokymo duomenų rinkinius atitinkamai santykiu 30:70. Kadangi B klasės įrašų kiekis sudaro tik 0,46 % visų duomenų rinkinio taikomas *SMOTE* (angl. *Synthetic Minority Over-sampling Technique*) algoritmas apmokymo duomenų rinkiniui ir pasiekiamas apytiksliai 1:1 abiejų klasių santykis. Naudojami *SMOTE* algoritmo parametras artimiausių kaimynų skaičiui K = 3. Pasirinktas mažas kaimynų skaičius su siekiu kuo mažiau modifikuoti originalų duomenų rinkinį.

Apmokymo bei testavimo duomenų rinkiniai išsaugomi failų pavadinimais *healthsurvey_train.csv* bei *healthsurvey_test.csv* atitinkamai.

4.2.2. Modelio apmokymas bei testavimas

Modelio apmokymas bei testavimas vykdomas naudojant toliau pateiktas *ML.NET* bibliotekas:

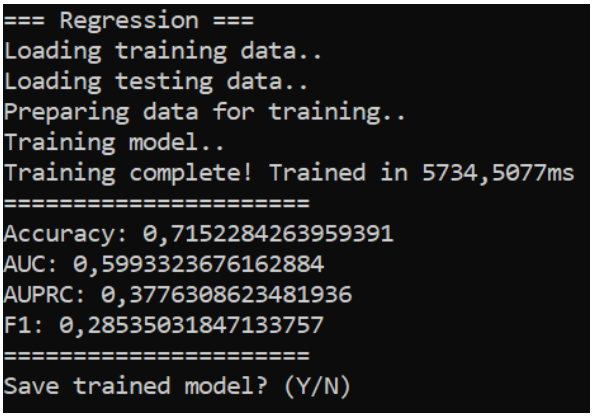
- *Microsoft.ML* – bazinės klasės ir metodai modelio apmokymui bei testavimui;
- *Microsoft.ML.Data* – apmokyto modelio testavimo rezultatų klasės;
- *Microsoft.ML.StandardTrainers* – regresiniam bei *SVM* modelių apmokymui;
- *Microsoft.ML.LightGbm* – *LightGBM* modelio apmokymui.

Sukurti keturi projektai modelių apmokymui, testavimui bei rezultatų palyginimui:

- *ADM.HealthRisk.Trainers.LightGbm* – komandinės eilutės programa, *LightGBM* modelio apmokymas bei testavimas;
- *ADM.HealthRisk.Trainers.Regression* – komandinės eilutės programa, regresinio neuroninio tinklo modelio apmokymas bei testavimas;
- *ADM.HealthRisk.Trainers.Svm* – komandinės eilutės programa, *SVM* modelio apmokymas bei testavimas;
- *ADM.HealthRisk.Trainers.Common* – su anksčiau paminėtais projektais bendrinamas projektas, kuriame aprašyta skaitomų *CSV* failų duomenų struktūra bei aprašytos pastovios reikšmės su apmokymo bei testavimo duomenų failų vieta diske.

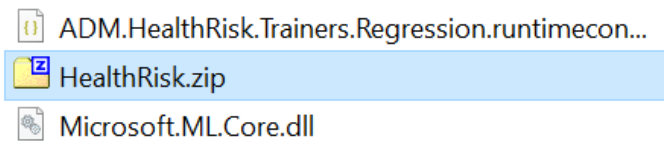
Didelių reikšmių įtaka apmokymui sumažinama *SVM* bei regresinio tinklo modeliams pritaikant *min-max* (mažiausios-didžiausios reikšmės) normalizavimą.

Toliau paveikslėlyje pateiktas pavyzdinė ADM.HealthRisk.Trainers.Regression išvestis (žr. pav. 4.5)



4.5 pav. *ADM.HealthRisk.Trainers.Regression* programos išvestis

Pateikiama paprasta informacija apie modelio apmokymo procesą, apmokymo trukmę bei gautas metrikas po modelio testavimo. Naudotojas taip pat turi pasirinkimą po modelio apmokymo išsaugoti modelį į diską (žr. pav. 4.6).



4.6 pav. *ADM.HealthRisk.Trainers.Regression* išsaugotas modelio failas

Išsaugotą modelį galima įkelti į *ADM.Api* programinės sąsajos naudojamą modelių aplanką (aplankas pavadinimu *Data/*) ir naudoti klasifikavimo uždaviniams spręsti.

4.2.3. Apmokytų modelių rezultatų palyginimas

Atlikti bandymai su regresiniu, *SVM* bei *LightGBM* modeliai, toliau lentelėje pateikti gauti rezultatai (žr. lentelė 4.2). Rezultatai pateikiami tūkstantųjų tikslumu.

4.2 lentelė. Rizikos vertinimo SQL užklausos rezultatas

Modelio pavadinimas	**Tikslumas**	***AUC***	***AUPRC***	***F1***	**Apmokymo laikas, sekundės**
LightGBM	0,994	0,741	0,362	0,421	0,682
Regression	0,919	0,802	0,290	0,070	5,735
SVM	0,995	0,235	0,003	0,000	0,328

Gautas didžiausias bendras modelio tikslumas 99,5 % su regresijos bei *SVM* modeliais, tačiau žvelgiant į metrikas geriau apibūdinančias modelio tikslumą klasifikuojant mažumos klasės įrašus – *AUPRC* bei *F1* - rezultatai parodo prastą modelio gebėjimą atrasti ryšį tarp sveikatos anketos duomenų bei gyvybės draudimo sumos ir vidutinės mėnesinės įmokos dydžių pokyčio. *AUPRC* metrikos dydis nežymiai viršija klasės santykį su bendru įrašų skaičiumi testavimo duomenyse (0,46 %) modeliuose *LightGBM* bei *Regression*, tad galima teigti, jog koreliacija tarp požymių bei nustatytos klasės egzistuoja.

5. REZULTATAI IR APIBENDRINIMAS

Išanalizuoti trys sprendimų palaikymo sistemų sprendimai draudimo rinkoje, palyginti autorių gauti rezultatai bei pasirinktas regresijos modelis rizikos vertinimo atvejui tirti. Suprojektuotas centralizuotų sprendimų palaikymo sistemos modulis bei realizuota pradinė jo versija, turinti klasifikavimo bei naujų modelių registravimo funkcionalumą. Sukurti trys modeliai su tikslu spręsti apdraustojo sveikatos anketos rizikos vertinimo uždavinį. Sukurtų ir ištestuotų modelių rezultatai palyginti, tačiau rezultatai nebuvo patenkinami - koreliacija tarp sveikatos anketos duomenų bei draudimo sumos ir įmokos dydžių pokyčio silpna.

Darbo tikslas pasiektas ne iki galo – sukurta pradinė centralizuotos sprendimų priėmimo sistemos versija, leidžianti mašininio mokymosi specialistams ateityje kuriamus bei tobulinamus modelius patalpinti bei atnaujinti.

Autoriaus rekomendacijos panašaus pobūdžio darbams:

- Realizuojant centralizuotą sprendimų palaikymo sistemą pasirinkti metodus, leidžiančius įvestį pateikti ne per lankstesnes duomenų struktūras, pavyzdžiui *JSON*, išvengiant modelių registravimo programiniame kode;
- Įtraukti apdraustojo tekstinius sveikatos duomenų aprašymus bei taikyti natūraliosios kalbos apdorojimo metodus;
- Norint pasiekti geresnių modelio rezultatų paanalizuoti kitų papildomų kintamųjų įtaką apdraustojo rizikai ir atlikti eksperimentus su dviejų etapų modeliais, kurie galėtų atskleisti sveikatos anketos duomenų įtaką draudimo sumos bei įmokos pokyčiui;

LITERATŪRA

[1] - Dubey, A., Parida, T., Birajdar, A., Prajapati, A. K., & Rane, S. (2018, April). Smart Underwriting System: An Intelligent Decision Support System for Insurance Approval & Risk Assessment. In 2018 3rd International Conference for Convergence in Technology (I2CT) (pp. 1-6). IEEE.

[2] - Sowah, R. A., Kuuboore, M., Ofoli, A., Kwofie, S., Asiedu, L., Koumadi, K. M., & Apeadu, K. O. (2019). Decision support system (DSS) for fraud detection in health insurance claims using genetic support vector machines (GSVMs). Journal of Engineering, 2019.

[3] - Yan, W., & Bonissone, P. P. (2006, July). Designing a Neural Network Decision System for Automated Insurance Underwriting. In The 2006 IEEE International Joint Conference on Neural Network Proceedings (pp. 2106-2113). IEEE.

[4] - Pisner, D. A., & Schnyer, D. M. (2020). Support vector machine. In Machine learning (pp. 101-121). Academic Press.

[5] - Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems, 30.

[6] - Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of statistics, 1189-1232.