EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103006/I/2019/36086129771768068

VYUŽITIE JAZYKA R PRE APLIKÁCIU NIEKTORÝCH TECHNÍK STROJOVÉHO UČENIA

Diplomová práca

Bratislava 2019 **Bc. Filip Konečný**

EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

VYUŽITIE JAZYKA R PRE APLIKÁCIU NIEKTORÝCH TECHNÍK STROJOVÉHO UČENIA

Diplomová práca

Študijný program: Aktuárstvo

Študijný odbor: Kvantitatívne metódy v ekónomií

Školiace pracovisko: Katedra matematiky a aktuárstva

Vedúci záverečnej práce: Ing. Michal Páleš, PhD.

Bratislava 2019 **Bc. Filip Konečný**

Ekonomická univerzita v Bratislave
Fakulta hospodárskej informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Bc. Filip Konečný
Študijný program:	aktuárstvo (Jednoodborové štúdium, inžiniersky II. st., denná forma)
Študijný odbor:	3.3.24 Kvantitatívne metódy v ekonómii
Typ záverečnej práce:	Inžinierska záverečná práca
Jazyk záverečnej práce:	slovenský
Sekundárny jazyk:	anglický

Názov: Využitie jazyka R pre aplikáciu niektorých techník strojového učenia

Anotácia: Strojové učenie (machine learning) je podoblasťou umelej inteligencie, ktorá sa zaoberá algoritmami a technikami, ktoré umožňujú počítačovému systému "učiť sa". Učením v danom kontexte chápeme takú zmenu vnútorného stavu systému, ktorá má efektívnu schopnosť prispôsobiť sa zmenám okolitého prostredia. Strojové učenie sa značne prelína s oblasťami štatistiky a hĺbkovej analýzy údajov a má široké uplatnenie. Základným druhom úloh v rámci strojového učenia je klasifikácia, regresia a zhlukovanie. K modelom, ktoré sa tu používajú patria rozhodovacie stromy, neurónové siete, Bayesovské siete, lineárne modely, asociatívne pravidlá a pod. Táto práca bude opisovať základné informácie pre aktuárov o využití techník neurónových sietí a rozhodovacích stromov pre riešenie jednoduchých problémov v rámci strojového učenia. A to aj v kontexte požiadaviek Curriculum 2019 vydaných IFoA. Ako aplikačné prostredie sa využije jazyk R. Od študenta sa očakáva systematická práca, analytické myslenie, pokročilé matematické a IT znalosti.

Vedúci: Ing. Michal Páleš, PhD.
Katedra: KMA FHI - Katedra matematiky a aktuárstva FHI

Dátum zadania: 22.10.2017

Dátum schválenia: 03.11.2017

doc. RNDr. Jozef Fecenko, CSc.
vedúci katedry

Čestné prehlásenie

Čestne prehlasujem, že diplomovú prácu som vypracoval samostatne a všetky použité zdroje citujem.

Dátum: .....

podpis študenta

ABSTRAKT

KONEČNÝ, Filip: *Využitie jazyka R pre aplikáciu niektorých techník strojového učenia*. – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra matematiky a aktuárstva. – Vedúci záverečnej práce: Ing. Michal Páleš. PhD. Bratislava: FHI EU, 2019, 89s.

Cieľom záverečnej práce je prezentovať problematiku poisťovní spojenú so stúpajúcim množstvom dát nazývanú Big Data a potreby poisťovní na tento trend reagovať. Riešenie týchto problémov je možné použitím metód z oblasti strojového učenia ID3, CART a neurónové siete s dopredným šírením. Tieto postupy boli aplikované na dáta klasifikujúce rizikovú skupinu/prirážku poistencov a kreditné riziko vo forme schvaľovania žiadostí o úver. Použité bolo softvérové vybavenie jazyka R a jeho knižníc. Rozhodovacie stromy dokazujú svoju schopnosť riešiť interpretáciu dát s dostatočnou presnosťou v rámci svojich možnosti a závislosti na dostatočne významné atribúty. Neurónové siete s dopredným šírením sú schopné reagovať na interakciu v dátach, ale požadujú veľké množstvo dát s dostatočnou vnútornou interakciou.

Kľúčové slová: Big Data, strojové učenie, rozhodovacie stromy, ID3, CART, orezávanie rozhodovacích stromov, neurónové siete, algoritmus spätného šírenia chyby, ROC

ABSTRACT

KONEČNÝ, Filip: *Usage of language R for application some techniques of machine learning*. – University of Economics Bratislava. Faculty of economic informatics; Department of mathematics and actuarial science. – Thesis advisor: Ing. Michal Páleš. PhD. Bratislava: FHI EU, 2019, 89p.

Goal of the thesis is to present issues of insurance companies related to increasing amount of data called Big Data and needs of insurance companies to react to this trend. Solutions to these problems is possible by using machine learning methods such as ID3, CART and feed-forward neural networks. They were applied to data classifying risk group/surcharge of insured and credit risk in the form of loan approval. The used software was language R and its libraries. Decision trees prove their ability to solve data interpretation with satisfying degree of precision within its boundaries and its strong reliance on significant attributes. Feed-forward neural networks can react to interaction within data, but they demand huge amount of data with enough inner interaction.

Keywords: Big Data, machine learning, decision trees, ID3, CART, decision tree pruning, neural networks, algorithm of error backpropagation, ROC

Obsah

Úvod

Finančné inštitúcie ako poisťovne a banky zamestnávajú nespočetné množstvo špecialistov (medzi nimi aj aktuárov) na riešenie ich problémov. Ich náplňou je analyzovanie a manažovanie rizika úzkou spojeného s ich podnikateľkou činnosťou. Ich nenahraditeľnosť v poisťovniach je v oblasti tvorby rezerv alebo oceňovania poistných produktov. Podobné problémy musia riešiť aj banky, ktoré musia zaistiť rentabilitu ich produktov a ich konkurencieschopnosť.

Aktuári využívajú pri tom široké spektrum znalostí v oblasti matematiky a štatistiky, ktorými dokážu tieto riziká kvantifikovať. Sú spravidla konzervatívni vo voľbe nástrojov a volia overené postupy, ktoré vyhovujú aj regulátorom. Ich výsledky sú preto auditovateľné a vysvetliteľné jednotlivým užívateľom.

Tieto prostriedky ukazujú svoje limity, nakoľko informatizácia spoločnosti zvýšila informovanosť zákazníka a zvyšuje tlak na už aj tak vysoko konkurenčnom trhu. Vznikajúce tlaky tak vedú k znižovaniu marže a tvorbe lepších marketingových stratégií pre získanie si nových a udržanie existujúcich zákazníkov. Informatizácia tiež priniesla aj nové možnosti finančným inštitúciám v získavaní a spracovávaní dát o ich (potencionálnych) zákazníkoch. So zvyšujúcim sa objemom dát, ktoré je potrebné premeniť na informácie, vzniká motivácia na ich spracovávanie a vyhodnotenie aj medzi aktuármi pre tvorbu lepších a rentabilnejších modelov.

Riešenie týchto problémov ponúka mladá vedná disciplína nazývaná Data Science (dátová veda alebo veda o dátach). Jednou z významných činností pri analýze dát je aj využitie strojového učenia s nástrojmi umožňujúcimi spracovávať veľké množstvo dát a získať nový pohľad na dáta, ktoré nie sú dostupné pri využití klasických matematicko-štatistických metódach. Oblasť strojového učenia je rôznorodá a využíva postupy, ktoré sú odlišné od zaužívaných štatistických postupov využívanými aktuármi.

Strojové učenie preukázalo v posledných rokoch možnosti ako riešiť komplexné problémy s nevídanou presnosťou. Cieľom tejto oblasti je spravidla vysvetliť a predpovedať dáta a ich správanie. Ciele strojového učenia sú podobné cieľom aktuárov pri tvorbe modelov. Strojové učenie je oblasť veľmi široká a bohatá na postupy, ktoré nie vždy umožňujú interpretáciu vzniknutých modelov, a porozumenie ich fungovaniu je veľmi dôležité pre výber vhodnej metódy. Korektným využitím znalostí môžeme získať informačnú hodnotu s ďaleko vyššou presnosťou, na akú sme zvyknutí pri využití zaužívaných postupov.

V tejto práci sa budeme venovať vybraným metódam strojového učenia a vysvetleniu ich vnútorného fungovania a rovnako myšlienke postavenej za ich fungovaním a vznikom. Pri pochopení vnútorného fungovania modelov prezentujem budúcim užívateľom týchto metód širšie a hlbšie znalosti, ktoré im pomôžu z nich vyťažiť väčší potenciál.

Venovať sa budeme rozhodovacím stromom a neurónovým sieťam ako vybraným metódam strojového učenia. Opísaný bude teoretický základ potrebný pre pochopenie interného fungovania týchto modelov a ich limitov. Práca obsahuje fundamentálne znalosti pre aplikáciu prezentovaných metód na pokročilej úrovni.

1 Súčasný stav riešenej problematiky doma a v zahraničí

V súčasnosti profesia aktuára zahŕňa využívanie prediktívnych modelov pre pochopenie a formulovanie rizika primárne pre poisťovne, banky, iné finančné inštitúcie alebo aj iné spoločnosti zaoberajúce sa rizikom v merateľnej podobe [36]. Využíva pri tom poznatky zo štatistiky a matematiky pre pochopenie týchto rizík z kvantitatívneho hľadiska. Aktuárske povolanie je významnou súčasťou poisťovacieho a celého finančného sektora.

V súčasnosti prevládajú v praxi riešenia založené na metódach lineárnej regresie. Ich výhodou je priamočiare využitie a jednoduchosť interpretovania získaných výsledkoch. Tieto postupy sa stali de facto štandardom pri využívaní v oblasti oceňovania. Dôvod pre ich používanie môžeme hľadať aj v potrebe uspokojiť požiadavky regulátorov, ktorí kladú dôraz na spravodlivé poistné bez diskriminačných prvkov, ktoré by boli v rozpore s platnou reguláciou [24].

Dominujúcimi modelmi využívaným v oblasti oceňovania je GLM (generalized linear models, zovšeobecnené lineárne modely), ktorý dokáže vďaka svojim vlastnostiam pracovať s dátami, pri ktorých sú porušené predpoklady pre využitie klasickej lineárnej regresie. Ich obrovskou výhodou je priamočiara interpretovateľnosť získaných informácií, čo umožňuje manažmentu, regulátorom, vlastníkom a zákazníkom poskytnúť relevantné informácie v ľahko pochopiteľnej forme [13]. Obdobou týchto modelov pri klasifikácií je logistická regresia, ktorá je využiteľná pri klasifikačných problémoch. Ich obrovskou výhodou je triviálnosť ich prezentácie zúčastneným stranám s veľmi priamočiarym vysvetlením vplyvu jednotlivých ukazovateľov.

Pri konštrukcii týchto modelov sa využíva obmedzené množstvo informácií so štatistickou významnosťou alebo len informácie, ktoré sú predspracované a dostupné poisťovniam v dostatočnej kvalite. V posledných rokoch, ale technologický pokrok umožňuje ľubovoľnej spoločnosti získavať prakticky neobmedzené množstvo informácií a optimalizovať rozhodovacie procesy s nevídanou presnosťou. Pri využití klasických prostriedkov spracovanie takéhoto množstvo informácií narážame na prekážky v súčasne používanej metodike.

Viditeľným nedostatkom je schopnosť hľadať v obrovskom množstve informácií relevantné vzťahy a štruktúru, ktorú nemusí aktuár bez empirických znalostí nájsť. Je nerealistické očakávať vždy dostatok empirických znalostí od aktuára vzhľadom na rýchlosť inovácií a možností v súčasnom trhovom prostredí. Hľadanie potencionálneho vzťahu a štruktúry je možné aj hrubou silou, ale kvalita možných výsledkov je otázna. Problémy

spojené s veľkým množstvom dát a ich nepretržitým vznikom dostali označenie Big Data. Problémy Big Data nie sú posledné roky vlastné len poisťovniam a iným finančným inštitúciám, ale celému trhového prostrediu.

1.1 Big Data

Big Data býva charakterizované v angličtine piatimi V: *volume* (*objem*), *velocity* (*rýchlosť*), *variety* (*rôznorodosť*), *veracity* (*pravdivosť*), *value* (*hodnota*) [1, 38]. Pre lepšiu predstavu je vhodné vytvoriť určitú paralelu v poisťovníctve s týmito charakteristikami.

Objem (volume) dát je v súčasnom svete generovaný veľkým množstvom technických prostriedkov a v poisťovniach sa s nimi môžeme stretnúť pri tvorbe technických rezerv. Rozhodovanie na trhoch s cennými papiermi vyžaduje veľké množstvo kvalitatívnych a kvantitatívnych informácií získaných z rôznych zdrojov. Veľké množstvo dát je možné získavať aj pri poistencoch. Pri rozvoji technológií pod označením IoT (internet of things, internet vecí) je možné získať od poistencov alebo záujemcov o poistenie veľké množstvo telemetrických informácií, napríklad pri havarijnom, zdravotnom alebo inom poistení. Získavať nepretržitý objem informácií o návykoch a správaní poistencov umožňuje poisťovni správne vyhodnotenie rizikovosti klienta, čo môže byť významná konkurenčná výhoda. Nevyhnutným predpokladom je schopnosť využiť tieto dáta.

Rýchlosť (velocity) tvorby dát v súčasnom svete nepozná obdoby v histórii ľudstva a každým rokom má objem vytvorených dát narastať [32]. Súčasný nárast zariadení ako IoT a rozšírenie internetu umožňuje získavať obrovské množstvo dát a tie je potrebné analyzovať s čím ďalej väčšou rýchlosťou. Pri zväčšujúcej sa rýchlosti nadobúdania dát je použitie klasických štatistických modelov možné, ale je nutné vytvoriť dostatočné softvérové vybavenie pre spracovávanie a pripravovanie takýchto dát. Pri dostatočne veľkom objeme dát je možné uvažovať aj o distribuovaných systémoch na spracovávanie dát, čo výrazne komplikuje využitie klasických a zaužívaných modelov. Schopnosť dostatočne rýchlo reagovať na zmeny môže poskytovať firmám výhody pri oslovovaní a príprave produktov/služieb na mieru zákazníkovi.

Rôznorodosť (variety) vznikajúcich dát vytvára obrovské prekážky vo využívaní zaužívaných postupov v poisťovníctve a je nutné pristúpiť k spracovávaniu dát, ktoré vyžaduje väčšiu mieru automatizácie a sofistikovanejších metód práce so získanými dátami. Pri využívaní modelov založených na lineárnej regresii sme zvyknutí pracovať so spojitými alebo kategorickými (umelými) premennými. Dáta v praxi bežne nadobúdajú podobu rôzne podoby od voľného textu cez semi-štruktúrované formy (ako elektronické dokumenty XML)

až po štruktúrované dáta v tabuľkách na aké sme zvyknutí. Je preto nevyhnutné venovať pozornosť tejto problematike, ak poisťovňa kladie dôraz na nižšie náklady.

Pravdivosť (veracity) nie je problém cudzí poisťovniam. Ak sa pri poistení snaží budúci poistiteľ podhodnotiť svoje riziká a poisťovňa je nútená takéto správanie odhaliť pre minimalizáciu potencionálnych strát. V prípade Big Data problém nastáva pri použití nadobudnutých dát. Tie je nutné pre ďalšie spracovanie testovať s dôrazom na ich výpovednú hodnotu a potencionálne anomálie v ich štruktúre alebo výpovednej hodnote. Pri dostatočne veľkom množstve nepravdivých informácií môže byť nadobudnutá informácia zavádzajúca a viesť k stratám. V najhoršom prípade k bankrotu poisťovne.

Hodnota (value) získaných dát je kľúčová, nakoľko získané informácie majú rozdielny kvalitatívny charakter a je nutné vedieť ich ohodnotiť. Spravidla je dôležité venovať pozornosť, či daná informácia prispieva k zvýšeniu konkurencieschopnosti podniku alebo je možné túto informáciu speňažiť iným spôsobom (napr. predajom). Ak dáta nemajú žiadnu hodnotu, je zbytočné vynakladať prostriedky na ich získavanie a spracovávanie. Existujú však situácie kedy to môže byť preferovaná stratégia. Pri získaných dátach je možné, že vieme o ich vnútornej hodnote, ale nemáme dostupné prostriedky na konzistentné a predpovedateľné metódy na ich spracovávanie. Je relevantné takéto informácie odkladať pre budúce použitie.

Dáta sa považujú za „zlato“ 21. storočia a je nutné venovať pozornosť ich zbieraniu a spracovávaniu do dátových skladov. Firmám môžu poskytovať konkurenčnú výhodu, ktorá sa nemusí nutne využívať len interne, ale aj ako externé nástroje poskytované ďalším stranám. Niektoré interné nástroje využívané firmami sa často zverejňujú alebo speňažia vo forme licencií. Firma tak získava konkurenčnú výhodu, nakoľko dáta použité pri tvorbe takýchto nástrojov sú jej vlastníctvom a je veľmi komplikované a náročne vytvoriť alternatívy k takýmto riešeniam.

Pri práci s osobnými dátami v Európskej únii je nutné venovať aj pozornosť platným právnym predpisom a reguláciám. Napríklad v poisťovníctve nediskriminovanie na základe pohlavia alebo dať ľuďom možnosť byť zabudnutý na základe smernice GDPR [35]. Pri tvorbe dátových riešení je nevyhnutné tieto limitácie mať na pamäti. V praxi je však možné riešiť problémy agregovaním dát a ich anonymizovaním, čo môže viesť k zvýšeniu nepresnosti, ale uspokojí požiadavky kladené legislatívnym rámcom. Výhodou poisťovní v porovnaní s inými užívateľmi osobných dát sú požiadavky kladené na aktuárov. Ich prax v etike a používaných postupov im poskytuje legislatívnu výhodu.

1.2 Strojové učenie

Pri nástupe veľkého množstva dát je nevyhnutné nájsť spôsoby a postupy vhodné na ich spracovanie. Problematika spracovávania dát nie je nová pre aktuárov alebo dátových vedcov. Ale pri hľadaní vzťahov a informácií, ktoré je možné získať z dát, je nevyhnutné vynaložiť veľké množstvo času a energie. Širokým rozšírením výpočtovej techniky v podobe počítačov je možné využiť veľké množstvo výpočtovej sily pri hľadaní vzťahov a štruktúry v dátach, to umožňuje aplikáciu techník strojového učenia.

Oblasť strojového učenia vznikla v minulom storočí ako podoblasť umelej inteligencie. Jej primárnym cieľom môžeme označiť schopnosť učiť počítačové programy automaticky sa zlepšovať cez skúsenosti. Pri veľkom množstve dostupných dát je použitie strojového učenia preferovaná alternatíva ako dosiahnuť tento cieľ. Je možné tak pomocou strojového učenia vytvoriť počítačové programy, ktoré sú potom schopné budúce vzory spracovávať automaticky.

Oblasť strojového učenia je veľmi veľká a je možné ju rozdeliť do troch skupín podľa ich postupov a predpokladov použitých pri práci s dátami. Jedná sa *supervised learning* (*učenie pod dohľadom*), *unsupervised learning* (*učenie bez dohľadu*) a *reinforcement learning* (*učenie posilňovaním*) [12].

Učenie pod dohľadom (supervised learning) je využívané aj v aktuárskej praxi v podobe lineárnej regresie. Hlavným znakom je prítomnosť vstupných dát, pre ktoré poznáme požadované výsledky (výstupy). Cieľom tohto učenie je zistiť aký je vzťah medzi vstupom a výstupom. Získaný vzťah chceme typicky ako informáciu o štruktúre dát, povahe dát alebo ako prostriedok pre predikciu nových výstupov pre nové dáta. Lineárna regresia sa preto dá považovať za metódu spadajúcu do tejto skupiny metód. Veľkou výhodou tejto skupiny je, že v pestrej palete možných metód je možné nájsť vhodné metódy pre rôzne typy dát. Hlavná myšlienka týchto modelov je zníženie chyby pri hľadaní tohto vzťahu. V tejto práci sa budeme venovať len metódam z tejto skupiny. Konkrétne sa budeme venovať rozhodovacím stromom a neurónovým sieťam. Uvidíme tiež ako rôzne metódy dosahujú výsledky pomocou rozdielneho prístupu.

Využitie určitých modelov strojového učenia zo skupiny *učenia pod dohľadom* môže tiež viesť k získaniu nových znalostí o použitých dátach, ktoré nemuseli byť predtým známe. Ak lineárna regresia zisťuje akou mierou prispieva určitá premenná k požadovanému výsledku, tak rozhodovacie stromy poukazujú vždy v konkrétnej podmnožine alebo podpriestore dát, aký atribút prispieva k maximálnemu vysvetleniu. Lineárna regresia rieši

tento problém globálne pre všetky použité dát, ale rozhodovacie stromy objasňujú dáta len na konkrétnej podmnožine. Vzniknuté modely, tak môže slúžiť tiež ako podklady pre zlepšenie ostatných modelov. Neurónové siete dokážu reagovať na vnútornú interakciu medzi jednotlivými vstupnými premennými a dosiahnuť lepších výsledkov, ale ich konkrétnu interpretáciu nie je možné realizovať. Daný model však môže byť využitý na benchmarking[1] novo vznikajúcich modelov, ak je využitie neurónovej siete nevhodné.

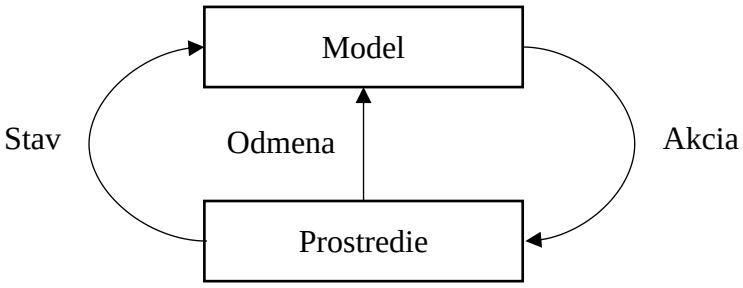
Učenie bez dohľadu (unsupervised learning) predstavuje formu, ktorá nutne nereprezentuje učenie ako v predchádzajúcich odstavcoch. Dáta využívané pri metódach z tejto skupiny nemávajú známe výstupy, resp. nie sú nejako označené ako v prípade *učenia s dohľadom*. Ak je nutné nájsť vzor správania alebo štruktúru dát bez predchádzajúcich znalostí o výsledku (alebo povahe dát), je na mieste voľba metód z tejto skupiny. Získané informácie sú opisného charakteru a pomáhajú užívateľom týchto informácií získať lepší pohľad na dáta. Získané vzory alebo odvodené pravidlá sú využiteľné pri ďalších analytických procesoch, čo môže z algoritmov v tejto oblasti spraviť veľmi atraktívnu voľbu pri konštruovaní akýchkoľvek modelov.

Typické pre túto skupinu je využitie *klástrovacích algoritmov* (*cluster algorithms*), ktoré využívajú stanovenú matematickú vzdialenosť medzi jednotlivými znakmi daných dát. Ak to samozrejme povaha dát umožňuje. Interpretácia *klástrovacích algoritmov* je veľmi intuitívna, čo uľahčuje ich použitie.

Učenie posilňovaním (reinforcement learning) zahŕňa metódy, ktoré optimalizujú správanie modelu vzhľadom na podnety z prostredia, v ktorom je trénované. Postupným učením tak tento proces zdokonaľuje svoje rozhodovanie. Tento proces spravidla pozná vstupný priestor a funkciu odmeny, ktorá v jednoduchej forme odmeňuje alebo trestá model za vyhodnotenie vstupu. Dostatočne dlhým trénovaním je možné získať funkčný model. Modely z tejto skupiny sú využívané v súčasnosti spravidla pri trénovaní samostatných rozhodovacích procesov ako hranie Go [3] alebo samo-jazdiace autá. Dokážu tak riešiť relatívne sofistikované úlohy, ale môžu byť aj veľmi zradné a viesť k nežiaducim výsledkom ako v prípade samo-jazdiacich áut k nehodám [20]. Ich využitie vo finančnej sfére si nájdú skôr pri automatizácií rutinných úloh s väčšou mierou variability, čím by priniesli zníženie nákladov. Využitie v aktuárskej činnosti má bariéry nakoľko povaha tejto skupiny strojového učenia sa zameriava na rozhodovacie procesy. Je však možné nazrieť na nich ako

[1] Testovanie a vyhodnocovanie kvality medzi dvomi a viacerými procesmi/modelmi.

alternatívu, ak dokáže poisťovňa zreplikovať pracovný proces aktuárov alebo iných zamestnancov a potencionálne straty by kompenzovala z ušetrených nákladov [10].



Obrázok 1.2.1: Životný cyklus modelu s posilňovaním

Zdroj: [12]

Pri využívaní metód z oblasti *učenia pod dohľadom* a *učenia posilňovaním* v umelom prostredí je treba brať ohľad na negatíva, ktoré môžu vzniknúť. Typicky môže prísť k výsledku, kedy algoritmus bude opakovať nežiaduci vzor, ktorý si nevšimli tvorcovia tohto modelu. Známy je prípad použitia strojového učenia pre vytvorenia nástroja v oblasti ľudských zdrojov spoločnosti Amazon. Vytvorený model zreplikoval správanie sa predchádzajúcich náborárov spoločnosti a výsledkom bola diskriminácia žien napriek tomu, že boli zo životopisov odobrané priame informácie o pohlaví uchádzača [5]. Je preto nutné venovať pozornosť tomu, že oblasť strojového učenia koná vždy optimálne vzhľadom na vstupné podmienky (dáta) a bude dosahovať nie vždy žiadané výsledky z pohľadu ľudskej morálky. Opatrné by pri tom mali byť obzvlášť poisťovne, ktoré sú regulátorom sledované pre potencionálne diskriminovanie na základe určitých znakov napriek tomu, že to legislatíva zakazuje.

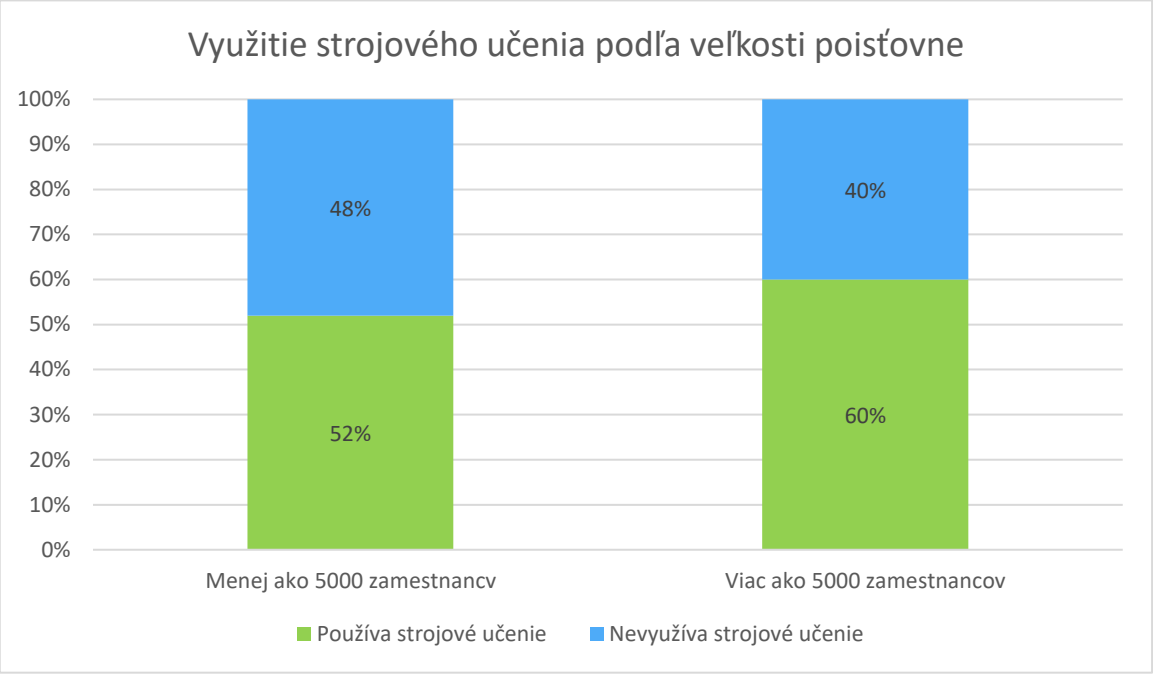
Pre lepšie pochopenie získaných informácií je výhodné poznať interné fungovanie každej použitej metódy strojového učenia. Pomáha to aj lepšiemu využitiu potenciálu zvolených metód. Je možné použiť ľubovoľnú metódu bez hĺbkovej znalosti jej fungovania, ale pri detailnom pochopení vnútorného fungovania je schopný autor modelu spracovať dáta pre zvolenú metódu do vhodnej formy. S tým ide ruka v ruke aj znalosť o vhodnosti voľby metódy pre daný typ dát alebo riešeného problému. Metódy strojového učenia sú schopné produkovať dostatočne presné výsledky, ale povaha zvolenej metódy nemusí byť vhodná a môže v určitých prípadoch produkovať zlé alebo zavádzajúce výsledky. Pri korektnom

použití jednotlivých metód tak môže získať minimálne autor týchto modelov informácie potrebné pre vyriešenie jeho problému.

1.3 Využitie strojového učenia a Big Data v poisťovniach

Strojové učenie si nachádzalo svoju funkciu vo všetkých spoločnostiach v posledných dvoch dekádach bez ohľadu na sektor. Jedinou podmienkou bola prítomnosť veľkého množstva dát a potreby nájsť v nich informácie pre zlepšenie rôznych cieľov ako predaja, lojality zákazníkov a vylepšovania produktov.

Prieskum vypracovaný spoločnosťou EARNIX [7] ukazuje významné využitie strojového učenia v poisťovníctve aj s kontextom. Údaje indikujú reálne využitie strojového učenia a jeho pozitívnom dopade na kvalitu a rýchlosť práce aktuárov. Využitie si nachádzajú už aj ako súčasť strategickej aktivity pár spoločností a nepracujú tak so strojovým učením ako s doplnkovým riešením, čo naznačuje jasný potenciál o strojovom učení v poisťovníctve. Tento potenciál nie je len v hypotetickej rovine.



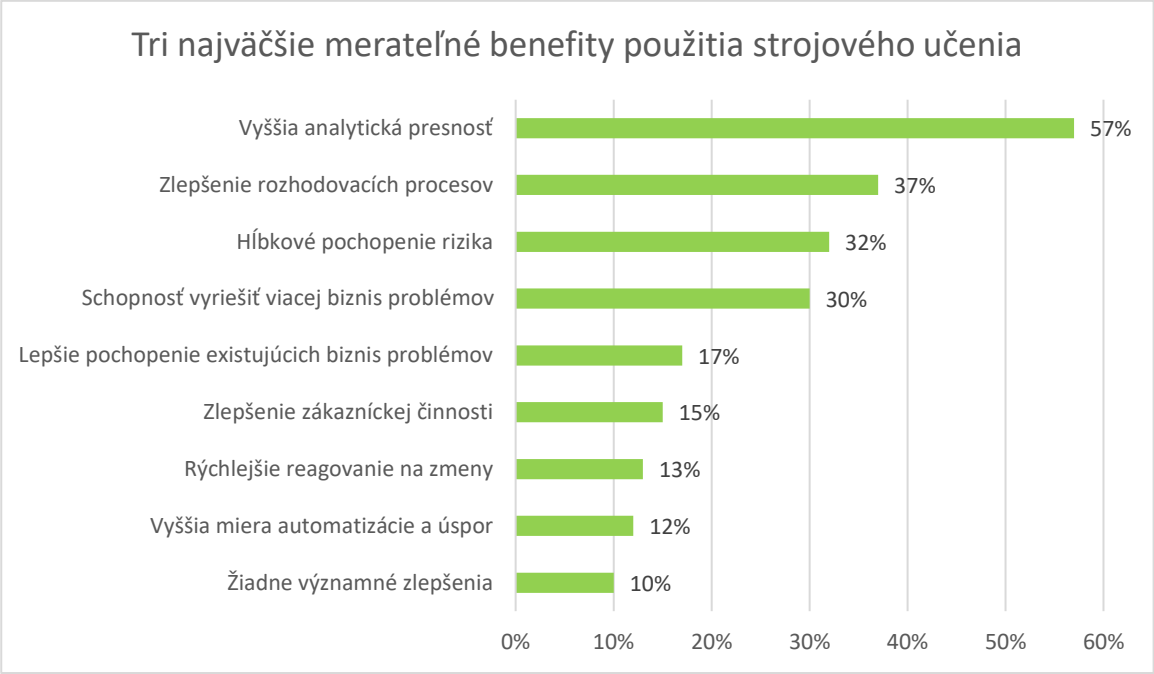
Graf 1.3.1: Podiel využitia strojového učenia v poisťovniach podľa ich veľkosti

Zdroj: [7]

Hlavné oblasti využitia v aktuárskej oblasti sú tvorba risk modelov (70%), dopytový modelov (45%), modely pre odhalenie podvodov (36%), analýza významnosti znaku (31%), odhalenie interakcie v dátach (31%) a analýza konkurenčného postavenia (28%). Použitie je prevažne pri riešení problémov spadajúcich pod *učenie pod dohľadom* alebo *bez dohľadu*.

Za špeciálnu pozornosť stojí tvorba risk modelov, ktoré zahŕňa prakticky základnú činnosť akejkoľvek poisťovni. Je preto vhodné mať v repertoári aktuára znalosti v strojovom učení pri testovaní alebo vývoji modelov už teraz.

Použitie strojového učenia získava aj praktický rozmer v ľubovoľných biznis procesoch spoločnosti ako poukazuje ďalšia otázka vybraná z prieskumu zisťujúca tri najväčšie benefity získané využitím strojového učenia.



Graf 1.3.2: Výsledky prieskumu pri označení troch najväčších benefitov využitia strojového učenia

Zdroj: [7]

Z výsledkov otázky troch najväčších benefitov získaných strojovým učením je možné vidieť, že strojové učenie prispieva pri korektnom použití lepšiemu pochopeniu spracovávaných dát. V tejto práci sa budeme venovať rozhodovacím stromom, u ktorých tento benefit je evidentný pri analýze výstupných modelov. Významné je aj hĺbkové pochopenie rizika, nakoľko riziko patrí k základnej podstate poisťovní a ak je schopná nasadiť vhodné metódy strojového učenia alebo ich kombináciu, je získaný benefit mocným nástrojom v konkurenčnom prostredí. Informačná výhoda sa tak môže pretaviť do zlepšenej marže, zníženia poistného alebo kombinácie oboch. Naprieč všetkými možnosťami je badateľné zlepšenie automatizácie a pochopenia dát hlavným prínosom už v súčasnosti.

Z celého prieskumu strojí za zmienku očakávania od strojového učenia v súčasnosti a budúcnosti. Poisťovne a aktuári očakávajú, že bude mať významný dopad naprieč

činnosťami celej poisťovne a nie len v špecifických prípadoch. Stretneme sa s ním určite pri oceňovaní, lepších analýzach rizika, automatizácie v procesoch poisťovne a zlepšenia zákazníckych a biznis procesov.

Využitie strojového učenia nájdeme na všetkých činnostiach poisťovne, z ktorých mnohé sú zdieľané aj s inými spoločnosťami a je možné si z nich zobrať inšpiráciu. Pre poisťovne budeme vychádzať z dokumentu *Big Data and role of the Actuary* [1].

Marketing má široké využitie pre Big Data a vedie k tvorbe stratégií a reklamných príležitostí pre ľahšie oslovenie zákazníka s poistným produktom. Pri súčasnom prevedení softvérových riešení je možné sledovať a zaznamenávať veľké množstvo údajov zo správania v jednotlivých softvérových riešeniach. Napríklad klient nie je schopný dokončiť objednávku spojenú so záujmom o poistku alebo je vidieť jeho zvýšený záujem o určité portfólio produktov pomocou analytických nástrojov sledujúcich správanie používateľov. Správanie (potencionálnych) zákazníkov je možné vo veľkom zaznamenávať a vytvárať tak presnejšie procesy pre ich konverziu alebo dodatočný krížový predaj ďalších riešení.

Ak sa z potencionálneho zákazníka stane skutočný zákazník, tak môže poisťovacia spoločnosť sledovať jeho správanie počas priebehu poistky a vytvárať podľa jeho zvyklostí ponuky, ktoré budú podporovať jeho lojalitu u poisťovne. Môže mu tak vedieť ponúknuť dodatočné produkty na základe jeho interakcie s poisťovňou, prípadne mu ponúknuť opcie, ktoré môžu byť zaujímavé pre poisťovňu a aj pre zákazníka. Pri dobre zvolenej stratégií môže poisťovňa vybudovať so zákazníkom silný vzťah, z ktorého budú profitovať obe strany. Modely v tejto oblasti by sa typicky sústredili na hľadanie neznámych vzorov správania zákazníkov a predikcie pravdepodobnosti nákupu.

Behaviorálna analýza zákazníkov môže priniesť nové porozumenie v správaní zákazníkov nie len pri krížovom predaji a cielení marketingových stratégií, ale aj pri predikcií jeho budúcich rozhodnutí. Poisťovňa tak vie lepšie segmentovať svojich zákazníkov a pripraviť si rôzne stratégie pri práci s nimi. *Behaviorálna analýza* vie tiež predpovedať budúce rozhodnutia ako kedy alebo či vôbec zákazník chce zrušiť poistnú zmluvu. Poisťovňa vie v dostatočnom predstihu reagovať na jeho konanie s cieľom jeho ďalšieho zotrvania. Pri životnom poistení je prevencia lapsov, resp. odchodov poistencov, veľmi atraktívna, keďže pri odchode poistenca je uvoľnenie rezerv v prípade rezervotvorného poistenia nákladný proces.

Zákaznícka spokojnosť sa dá tiež veľmi presne sledovať pomocou správnej metodológie a je známe, že udržať si existujúceho zákazníka je lacnejšie ako získanie nového. Použitie strojového učenia vedie nie len k lepšiemu pochopeniu zákazníka, ale vie

mu jednoduchšie poskytnúť dodatočné informácie. Správne namiešaný marketingový mix následne vedie k dodatočnému predaju (*upselling*).

Cielený marketing je v klasickom ponímaní veľmi náročný a nákladný proces, ktorý sa oplatil v minulosti len pri veľkých klientoch, ale s príchodom moderných technológií do každodenného života sa stáva dostupný aj pre malých zákazníkov. Tvorba a porozumenie profilu zákazníka je možné zautomatizovať. Výsledkom tak môže byť aj potencionálne predvídanie životných udalostí v živote poistenca a poskytnúť mu relevantné poistenie v prípade životných udalostí, ktoré sa dajú vyčítať z jeho správania a demografickej skupiny.

Upisovanie (underwriting) je oblasť, v ktorej môžu poisťovne ťažiť z mnohých prínosov strojového učenia a veľkého množstva dostupných dát. Jediná prekážka je typicky regulácia sektora, ktorá sleduje diskriminačné faktory použité pri posudzovaní a určovaní výšky poistného. Ale v prípade použitia rôznych techník ako redukcia dimenzionality nastáva tvorba agregátov, ktoré môžu maskovať použitie týchto faktorov. Tento proces môže poukázať aj na konkrétnejšie rizikové faktory. Optimalizácia môže byť aj vo forme rýchlejšieho a presnejšieho posúdenia klienta a jeho presnejšieho zaradenia v rámci rizikových faktorov. V strojovom učení by sa jednalo o problém klasifikácie.

Vývoj poistných produktov bude ťažiť z využitia z strojového učenia a Big Data možno ako prvá akturáska oblasť. Ich využitie nemusí zostať len pri lepšom pochopení dát a klientov so záujmom poistiť sa, ale aj pri skúmaní a rozširovaní sa na nové trhy. Poistné produkty sa však môžu stať veľmi komplikované pre klientov, ak budú veľmi fragmentované alebo stavané až príliš na mieru. Je preto vhodné prípadne vyvážiť mieru komplexnosti s jednoduchosťou, ktorú sú zákazníci schopní akceptovať pred prechodom ku konkurencií.

Dopomáha k tomu aj rast IoT zariadení, ktoré vykonávajú nejakú čiastkovú činnosť a ak jednou z nich je získavanie dát a ich odosielanie. Pri správnom použití produkujú veľké objemy relevantných dát. S ich použitím sa môžeme stretnúť na rôznych miestach aj v poisťovníctve ako na sledovanie stavu alebo zvyklostí (potencionálneho) poistenca. Získané dáta premeniť na informácie je úlohou odborníkov ako aktuárov, dátových vedcov a expertov na strojové učenie.

Využitie strojového učenia **v životnom poistení** môže výrazne uľahčiť analýzu a posúdenie klienta, ak máme dostupné široké množstvo dát s dôrazom na zdravotné. Posúdenie zdravotného stavu človeka je veľmi komplikované v čase uzatvárania poistenia, nie to ešte odhadnúť vývoj zdravotného stavu v budúcnosti. Pri priebežnom zbieraní dát o klientovi je možné odhadnúť jeho budúci vývoj a tým vedieť zlepšiť jeho odhad úmrtnosti.

Zákazníkovi je možné presnejšie stanoviť poistné s lepším porozumením zloženia rizikovej prirážky. Zaujímavý benefit spojený s Big Data a zákazníkmi poskytujúcimi dáta je odmeňovanie zákazníkov za ich zdravý životný štýl, ak je klient ochotný poskytnúť prístup k niektorým osobným dátam ako tým, ktoré sú generované inteligentnými hodinkami alebo inými zariadeniami počas dňa merajúce telesnú aktivitu [19].

Lepšia **tvorba opcií** môže viesť k tvorbe poistných produktov, kde sa opcie budú stanovovať nie len na základe zákazníckych preferencií, ale aj schopnosti rozlíšiť správanie klienta podľa jeho demografickej skupiny a jeho preferenciám. Stanovenie vhodných sadzieb a termínov môže viesť k ich väčšiemu využitiu.

Neživotné poistenie do určitej miery rozlišuje rozdielne znaky a problémy spojené s rozdielnymi príznakmi pre jednotlivé poistenia [9]. Do určitej miery sa tak aplikovali postupy, ktoré ponúka strojové učenie, ale v prípade použitia strojového učenia je možné získať lepší pohľad na povahu poistných zmlúv a možnú interakciu premenných v dátach. Získané informácie sa môžu použiť k vývoju cielenejších poistných produktov a dosiahnuť tak konkurenčnú výhodu. Jedinou prekážkou môže byť presvedčiť regulátora o použitých postupoch. Pri využití informácií z rozrastajúceho sa portfólia a možností IoT môže poisťovňa ponúknuť lepšie poistné (bonus). Potencionálne využitie týchto zariadení môžeme vidieť pri poistení automobilov, kde môžeme modelovať správanie vodiča na cestách a zistiť lokality, v ktorých sa pohybuje. Tieto dáta mávajú osobnú povahu a je preto nutné im venovať väčšiu pozornosť pri ich zabezpečení. Nespornou výhodou neživotného poistenia oproti životnému je väčšia voľnosť v experimentovaní, nakoľko v porovnaní so životným poistením je neživotné poistenie krátkodobé a nevyžaduje konzervatívny prístup v takej miere ako životné k tvorbe poistných produktov.

Automatizácia je pri dostatočné veľkom množstve dát najvýraznejšie cítiť v nákladoch naprieč celou poisťovňou. Analýza nahlásených poistných udalostí pre rýchlejšie posudzovanie alebo odhalenie podvodného správania môže byť lacnejšia a presnejšia. Urýchlenie alebo kompletné nahradenie ľudského rozhodovania je jedna z veľmi silných stránok typicky neurónových sietí alebo aj iných metód, ak to povaha dát a metóda umožňuje. Neurónové siete sú však schopné v súčasnosti komplexnejšieho rozhodovania ako typické metódy strojového učenia (*SVM – support vector machine, rozhodovacie stromy, klástrovanie*, atď.). Na vedomie je treba brať ako sú jednotlivé metódy používané a trénované, pretože pri trénovaní na dátach tvorených na základe historickej činnosti ľudí môže prísť iba k replikácií celého procesu, čo je síce žiadaný efekt, ale môže so sebou priniesť aj nepríjemné prekvapenia v replikovaní negatívnych predsudkov ako bolo

spomenuté. Námaha na vytvorenie automatu, ktorý bude spoľahlivý vo svojich záveroch je veľmi nákladné a vyžaduje odborný dohľad odborníkov v oblasti strojového učenia a tiež odborníkov riešenej problematiky.

V **zdravotnom a dôchodkovom poistení** je použitie strojového učenia a Big Data možné len v procesných oblastiach nakoľko legislatíva Slovenskej republiky nariaďuje striktne za akých podmienok je poistenie poskytnuté. Nie je tak možné využiť týchto prostriedkov na segmentáciu zákazníkov alebo individuálnu ponuku poistného produktu, ak zákon predpisuje presné postupy. Všetci občania Slovenskej republiky sú povinný odvádzať poistné podľa platných zákonov, ktoré stanovujú presné sadzby na základe veľkosti ich príjmu. Napríklad pri zdravotnom poistení osoby so zdravotným postihnutím je nemožné penalizovať, nakoľko zákon ich priamo zvýhodňuje polovičnou sadzbou [34].

Ako posledné je nutné poukázať na to ako sa použitie rôznych údajov a metód vytvára aj tlak na regulátorov, ktorí potrebujú trénovať a pripraviť ich zamestnancov na rastúcu komplexnosť v oblasti spracovávania dát. V porovnaní s technikami použitými len v strojovom učení sú klasické štatistické modely jednoduchšie na pochopenie a potreba nájsť špecialistov v strojovom učení alebo Data Science môže byť veľkým problémom pre súkromný sektor, nie to ešte pre verejný sektor. Sektory bez regulácie dokážu pružne zaraďovať nové riešenia do svojho portfólia a vyvíjať produkty, ktoré nemusia prísť do kontaktu so štátom. Ten potom reaguje až keď príde k nejakej udalosti, kde sú štátne inštitúcie povinné reagovať [20]. V poistnom a finančnom sektore sú aktivity štátnych regulátor bežnejšie.

2 Cieľ práce

Hlavný cieľ záverečnej práce je odprezentovať a vysvetliť do hĺbky širšiu problematiku, s ktorou sa stretávajú súčasné spoločnosti vo vzťahu k strojovému učeniu a jeho využití pri riešení existujúcich problémov s dôrazom na poisťovníctvo. Poskytnutý bude náhľad nad komplexným svetom Big Data a prehľad pre zorientovanie sa v problematike strojového učenia pri práci s ním.

Čiastkový cieľ je vysvetliť do hĺbky fungovanie vybraných metód strojového učenia, konkrétne sa jedná o rozhodovacie stromy a neurónové siete s dopredným šírením. Pochopením podstaty jednotlivých metód by mal čitateľ ako budúci užívateľ byť vybavený teoretickými znalosťami vnútorného fungovania metód ID3 a CART z rodiny rozhodovacích stromov a internému fungovaniu neurónových sietí s dopredným šírením. Teoretické poznatky budú vysvetľované na pôvodných matematických konceptoch jednotlivých metód doplnené o vlastné komentáre s pozorovaniami pri práci s nimi.

Pri posudzovaní vyprodukovaných modelov jednotlivými modelmi je nutné použiť metriku, ktorá sa dá využiť naprieč rôznymi typmi modelov a bude preto vysvetlené použitie ROC (receiver operating characteristics). Testovanie kvality modelov bude možné realizovať nezávisle na originálnych postupoch používaných interne pri vyhodnocovaní v jednotlivých metódach. ROC analýza nám umožní vidieť problémy spojené s výberom vhodného modelu presnejšie ako naivné spoliehanie sa na správne klasifikovanie dát. Následne túto techniku aplikujeme na vytvorené modely, aby sme vedeli porovnať ich kvalitu medzi sebou a stanoviť najvhodnejší.

V praktickej časti práci sa budeme zaoberať použitím pri klasifikačných problémov v oblasti zaradenia klienta do rizikových skupín, ktoré budú reprezentovať rizikovú prirážku. Ako ďalší klasifikačný problém zvolíme posúdenie žiadosti o úver. Bude sa tak jednať o problém kreditného rizika. Na týchto dátach aplikujeme techniky vysvetlené v metodike s komentárom k možným nedostatkom a potenciálnym problémom.

3 Metodika práce a metodika skúmania

3.1 Rozhodovacie stromy

Rozhodovacie stromy sú jednou zo starších metód strojového učenia, ktoré boli a stále sú oblasťou záujmu pre vedeckú oblasť a súkromný sektor. Ich veľkou výhodou je ľahká interpretovateľnosť expertmi pre oblasť, pre ktorú bol rozhodovací strom vytvorený. Dokážu tak posúdiť kvalitu stromu a prípadne zaviesť dodatočné pravidlá alebo pozmeniť existujúce. V praxi sa však väčšinou rozhodovacie stromy pretrénujú alebo sa vstupné dáta transformujú.

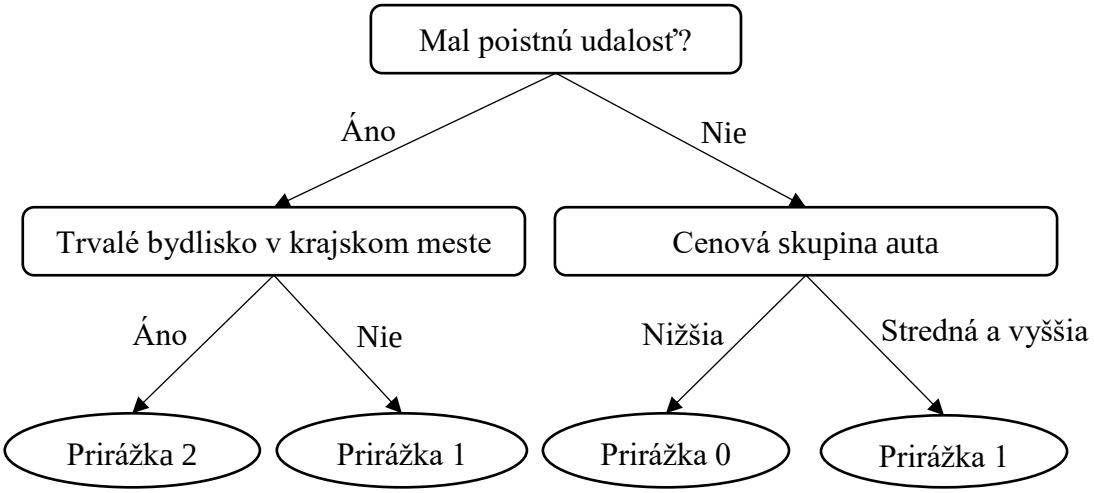
Praktickosť rozhodovacích stromov je výborná nad dátami, ktoré nedisponujú veľkým množstvom atribútov so spojitými číselnými dátami. Pokiaľ ich je väčšina alebo dostatočne veľké množstvo je na mieste zvážiť využitie inej metódy alebo kombináciu viacerých metód strojového učenia pre dosiahnutie vhodného klasifikátora.

Matematickú definícia stromu je detailne opísaná v iných publikáciách a použitá bude definícia bude podľa [4]. Rozhodovací strom sa vytvára za účelom klasifikácie dát, ktoré definujeme ako príznakový vektor (alebo tiež vzor) so sprievodnou triedou alebo numerickou hodnotu definujúcou vzor, pre ktoré rozhodovací strom tvoríme.

Rozlišujú sa dva druhy hodnôt a to numerické a kategorické (nominálne). Numerické hodnoty vyžadujú špeciálnu pozornosť pri klasifikácií v rozhodovacom strome, nakoľko majú spravidla spojitý charakter a je potrebné nájsť vhodné hodnoty (alebo hodnotu) pre rozdelenie intervalu. V prípade komplexnejších numerických dát (vektorov) by bolo vhodné aj použitie klastrovacích metód hľadajúcich zhluky jednotlivých príznakov. Kategorické hodnoty majú pre konkrétny rozhodovací strom konečný počet, sú známe a nezáleží na ich poradí.

Vzor (príznakový vektor) definujeme ako $o \in O$, kde O predstavuje množinu všetkých vzorov (príznakových vektorov). Trieda $c \in C$ predstavuje hodnotu z množiny všetkých tried C. Ako ďalšiu si definujeme množinu X, pre ktorú platí $X \subset O \times C$. X tak predstavuje podmnožinu všetkých kombinácií vzorov a tried.

Trénovací vzor zadefinujeme ako $x \in X$. Zadefinujeme si tiež funkciu $\pi(x) = c$, ktorá projektuje vzor x do triedy c. Každá pozícia vo vzore x predstavuje atribút okrem poslednej pozície, pre tú použijeme označenie c na rozlíšenie ako triedy pre klasifikáciu vzoru. Ak definujeme vzor ako $x = (o_1, o_2, o_3, ..., o_n, c) = (x_1, x_2, x_3, ..., x_n, x_{n+1})$, potom hodnotu atribútu na i-tej pozícií bude možné získať funkciou $a_i(x) = x_i$. Množina všetkých možných hodnôt pre atribút na i-tej pozícií zadefinujeme ako H_i.



Obrázok 3.1.1: Diagram jednoduchého rozhodovacieho stromu.

Zdroj: vlastné spracovanie, podľa [4]

Rozhodovací strom je matematický graf, ktorý spĺňa všetky predpoklady stromu podľa definície diskrétnej matematiky. Rozhodovací sa stáva tým, že nesie dodatočné informácie v uzloch a hranách, ktoré použijeme pre klasifikovanie vzorov *x*.

Rozhodovací strom môže mať rôzne množstvo hrán z každého uzlu a z každého môže pokračovať ďalší podstrom. Vždy má však jeden uzol, ktorý nazývame koreň, z ktorého sa vetví rozhodovací strom ďalej. Ak z uzla nevychádzajú hrany do ďalších uzlov, tak ich nazývame aj listy. Vnútorné uzly sú všetky uzly, ktoré nie sú koreň a listy, a sú použité pre vetvenie v rámci rozhodovania v strome.

Formálna definícia stromu je realizovaná pomocou teórie grafov [4], v ktorej platí $G = (V, E)$ je graf, kde V sú vrcholy (uzly) a E predstavujú hrany. My budeme v tejto práci označovať tento graf T podľa anglického *tree*.

Všetky referujúce uzly na uzol v môžeme zadefinovať nasledujúco:

$$F_{in}(v) = \{u \in V \mid (u, v) \in E\} \tag{3.1.1}$$

Všetky uzly, na ktoré sa dá pokračovať z uzlu v definujeme:

$$F_{out}(v) = \{u \in V \mid (v, u) \in E\} \tag{3.1.2}$$

Stromom sa stáva graf, ak sú splnené nasledujúce podmienky:

$$|F_{in}(v)| \leq 1, \forall v \in V \tag{3.1.3}$$

$$|F_{in}(r)| = 0, \exists! \, r \in V \tag{3.1.4}$$

A graf T je súvislý. Vzťah (3.1.3) predstavuje kritérium, že do každého uzlu vchádza aspoň jedna hrana. Vzťah (3.1.4) definuje jeden uzol r ako koreň stromu, do ktorého

nevstupuje ani jedna hrana a tým predstavuje vstupný uzol stromu. Obrázok 3.1.1 má ako koreň uzol s popisom „Mal poistnú udalosť?“.

Ďalej potrebujeme formálne zadefinovať listy, ktorými disponuje graf T podľa predchádzajúcej definície.

$$L(T) = \{v \in V \mid |F_{out}(v)| = 0\} \tag{3.1.5}$$

Vnútorné uzly zadefinujeme nasledujúco, ak r je koreň:

$$V_{inner}(T) = \{v \in V \mid |F_{out}(v)| \geq 1 \wedge v \neq r\} \tag{3.1.6}$$

Každý strom sa dá rozdeliť do podstromov. Podstrom $T|_v$ definujeme ako strom obsahujúci uzol v a takisto každý uzol, do ktorého uzol v odkazuje.

$$F_{out}(u) \subseteq T|_v, \forall u \in V(T|_v) \tag{3.1.7}$$

Označenie $T|^v$ reprezentuje strom T bez podstromu $T|_v$ a uzlu v. Ako $T|_v$ označíme vrcholy podstromu $T|_v$.

$$V|^v = V \setminus V|_v \cup \{v\} \tag{3.1.8}$$

Pre dokončenie definície rozhodovacieho stromu potrebujeme zadefinovať ešte schopnosť prechádzať stromom a vyhodnotenia jednotlivých vzorov. Pre prechod stromom použijeme zobrazenie $\delta : V \setminus L(T) \times O \rightarrow V$, ktoré môžeme opísať ako uzol v pre vzor o odkazuje na uzol u, tak že platí $(v, u) \in E(G)$. Získame tak potrebnú vlastnosť pre rozhodovanie v konkrétnom uzle. Pri vyhodnotení definujeme zobrazenie tried pre listy rozhodovacieho stromu $\kappa : L(T) \rightarrow C$, ktoré priradí listu triedu z množiny C. Následne tak vieme jednotlivý vzor o zaradiť do konkrétnej triedy c. Tento proces nazývame *klasifikácia*.

Formálne vyjadrenie procesu klasifikácie vzoru o:

$$\delta(v, o) = u, v \notin L(T) \wedge (v, u) \in E(G) \tag{3.1.9}$$

$$\kappa(v) = c, v \in L(T) \tag{3.1.10}$$

Funkcia δ vyjadruje prechod do nasledujúceho uzlu podľa hodnôt atribútov prítomných vo vzore o. Ak je ďalší uzol v množine listov, tak použitím funkcie κ vyjadríme triedu c, podľa toho aká je asociovaná s listom v.

Binárny strom predstavuje špecifickú podobu stromu, kde z každého uzlu vystupujú dve hrany, ak sa jedná o vnútorný uzol, alebo žiadna, ak sa jedná o list. Pri opise fungovania metód tvorby rozhodovacích stromov budeme využívať prevažne túto metódu.

$$|F_{out}(v)| = 2 \vee |F_{out}(v)| = 0 \tag{3.1.11}$$

3.1.1 Metóda ID3 pre konštrukciu rozhodovacích stromov

ID3 (Iterative Dichotomiser 3) je veľmi jednoduchý algoritmus na konštrukciu rozhodovacích stromov [25]. Pri svojej tvorbe rozhodovacích podstromov používa rekurzívne volanie tvoriacej funkcie, ktorá prechádza všetky dáta na trénovacej množine a hľadá najvhodnejší atribút na ďalšie vetvenie rozhodovacieho stromu. Táto metóda nepozerá „späť“ a ani „dopredu“ a realizuje svoje výpočty na práve dostupnej množine dát, kde hľadá najvhodnejší atribút na delenie vo vznikajúcom vrchole a podľa toho realizuje svoje rozhodnutia. Každé ďalšie volanie tvoriacej funkcie rozdelí trénovaciu množinu podľa zvoleného atribútu a zmenšuje sa tak množstvo dát, nad ktorými je nutné dané rozhodovanie vykonať. Postup tvorby stromu je tak relatívne rýchly ako pri hľadaní všetkých najlepších možných deleniach v celej množine dát. Poskytuje tak dostatočne dobré a presné výsledky za cenu preučenia sa.

Pri opise ID3 budeme tvoriť binárny strom, nakoľko je demonštrácia fungovania ID3 jednoduchšia a priamočiarejšia. V prípade potreby je veľmi ľahké modifikovať algoritmus pre prácu s viac ako dvomi vetveniami a je tak možné dostať strom s menšou hĺbkou.

Vytvorený strom pomocou metódy ID3 môže trpieť preučením a je až veľmi dobre prispôsobený na trénovacie dáta, ktoré používal pri tvorbe. To spôsobuje problémy pre jeho ďalšie použitie, ak by sme chceli výsledný strom použiť v inom systéme. Rozhodovací strom sa môže stať súčasťou iného riešenia. Napríklad v poisťovníctve na klasifikáciu poistencov do predpripravených rizikových skupín alebo na určenie rizikových poistencov alebo naopak, ak chceme nájsť menej rizikových poistencov a poskytnúť im zľavu.

Stromy vytvorené ID3 je takmer vždy nutné orezať a získame tak všeobecnejší strom, ktorý je dostatočne presný a nie príliš konkrétny pre trénovacie dáta. Alternatívou je určenie a použitie podmienky pre predčasné ukončenie budovania stromu, čo môže mať svoje vlastné nedostatky ako podučenie v danom uzle pre nedostatok trénovacích dát. V praxi sa stretneme typicky s dvomi variantami a to ukončením pri malej množine dostupných trenovacích dát pre vznikajúci vrchol, formálne $|T| < n$, alebo ak pravdepodobnosť určenia triedy pre uzol dosahuje požadovanú úroveň $1 - \alpha$, kde α reprezentuje akceptovanú chybu.

Určenie listu je priamočiare, ak má množina všetkých dostupných dát pre tvorbu daného uzlu výskyt len jedného znaku, tak sa určí za hodnotu listu. V prípade, že je splnená jedna z podmienok v predchádzajúcom odstavci je vybraná dominantná trieda.

Pozornosť by sa mala venovať numerickým dátam, ktoré vyžadujú špeciálnu starostlivosť pri použití ID3. Pri hľadaní informačnej hodnoty je nutné nájsť vhodné

rozdelenie číselných dát vhodnou hodnotou *s*, ktorá zabezpečí maximálny informačný zisk pre získané množiny dát.

Výber rozhodovacieho kritéria

Pri hľadaní najlepšieho atribútu na delenie potrebujeme definovať funkciu nečistoty $I(X)$ [16], ktorá bude mať nasledujúce vlastnosti:

- v prípade množiny dát, ktorá obsahuje vzory len s jednou triedou, tak získame hodnotu 0,
- v prípade množiny dát, ktorá obsahuje rovnomerné rozložené triedy získame hodnotu 1.

Účelom tejto funkcie je nájsť také rozdelenie množiny trénovacích dát, aby sme získali, čo najmenšiu hodnotu nečistoty.

Autor metódy ID3 [25] využíva vzťah entropie intuitívne, ale v iných prácach [16] je zadefinovaný formálne ako:

$$I(X) = -\sum_{c \in C} P(\pi(X) = c) \log_2 P(\pi(X) = c) \qquad (3.1.12)$$

$$0 \log_2 0 \equiv 0$$

Funkciu P definujeme ako funkciu pravdepodobnosti pre výskyt jednotlivých tried podľa danej podmienky. Napríklad v množine C sa nachádzajú dve triedy a to c_1 so 7 vzormi a c_2 s 3 vzormi, získame pravdepodobnosti $P(\pi(X) = c_1) = 0{,}7$ a $P(\pi(X) = c_2) = 0{,}3$. X reprezentujú trénovacie dáta použité pre konkrétny výpočet. Intuitívne tu používame funkciu π ako súčasť podmienky pre spomínanú funkciu pravdepodobnosti.

Pre výpočet entropie nad jednotlivými hodnotami atribútov rozdelíme množinu X do podmnožín podľa hodnôt atribútov, nad ktorými budeme realizovať delenie. Vyberieme len také vzory z množiny X spĺňajúce predpoklad, že vzory obsahujú atribút a_i z množiny A odpovedajúci konkrétnej hodnote α.

$$X_{a_i=\alpha} = \{x \mid a_i(x) = \alpha, \forall x \in X, \forall \alpha \in H_i\} \qquad (3.1.13)$$

Pričom používame funkciu $a_i(x)=x_i$, ktorá je zadefinovaná v prvej kapitole.

Výpočet hodnoty entropie pre atribút a_i získame ako vážený priemer všetkých možných delení podľa jednotlivých hodnôt atribútu H_i.

$$E(a_i) = \sum_{\alpha \in H_i} P(a_i(X) = \alpha) I(X_{a_i=\alpha}) \qquad (3.1.14)$$

Pre intuitívnejšie pochopenie si výpočet môžeme zobraziť.

Tabuľka 3.1.1: Príklad použitia entropie

a_i	Prirážka 1	Prirážka 2	$I(X_{a=\alpha})$
α	2	4	$-\left(\left(\frac{2}{6}\log_2\frac{2}{6}\right)+\left(\frac{4}{6}\log_2\frac{4}{6}\right)\right)\cong 0{,}918$
β	3	1	$-\left(\left(\frac{3}{4}\log_2\frac{3}{4}\right)+\left(\frac{1}{4}\log_2\frac{1}{4}\right)\right)\cong 0{,}811$

Zdroj: vlastné spracovanie, podľa [16]

Pre dokončenie výpočtu spočítame entropiu pre celý atribút pomocou (3.1.14):

$$E(a_i)\cong\frac{6}{10}0{,}918+\frac{4}{10}0{,}811\cong 0{,}875$$

Binárny strom dostaneme metódou entropie len vtedy, ak sú jednotlivé atribúty tiež binárne nakoľko vetvenie sa realizuje pomocou atribútov.

Pre dokončenie výpočtu je potrebné určiť hodnotu entropie pre trénovacie vzory prítomné v danom uzle bez rozlíšenia kategórií (3.1.12):

$$I(X)=-\left(\left(\frac{5}{10}\log_2\frac{5}{10}\right)+\left(\frac{5}{10}\log_2\frac{5}{10}\right)\right)=1$$

V prípade tohto atribútu je neusporiadanosť systému maximálna možná, keďže entropia dosiahla hodnotu 1. Najnižšia entropia je rovná 0 a predstavuje tak z pohľadu stromu uzol s dokonalou informačnou hodnotou pre určenie tvoreného uzlu.

Informačný zisk [16, 25] vypočítame podľa:

$$Z(a_i)=I(X)-E(a_i) \tag{3.1.15}$$

Pre konkrétne hodnoty, ktoré sme vypočítali:

$$Z(a_i)\cong 1-0{,}875\cong 0{,}125$$

Pre potreby vetvenia budeme preferovať informačný zisk, čo najväčší.

Najjednoduchšia forma rozhodovacieho kritéria, akú možno zvoliť, má podobu jednoduchej pravdepodobnosti výskytu jednotlivých javov v aktuálnej množine [29]:

$$I(X)=\sum_{c\in C}\min\{P(\pi(X)=c),P(\pi(X)\neq c)\} \tag{3.1.16}$$

Potom potrebujeme vyhľadať atribút, pre ktorý získame maximálny možný informačný zisk a na tom realizujeme delenie. Využijeme pri tom definíciu v (3.1.13), ktorú použijeme vo vzťahu (3.1.14) a získame tak informačný zisk vo vzťahu (3.1.15).

Populárny a tiež využívaný je Giniho index [2], ktorý je využívaný v ekonomických porovnania distribúcie bohatstva a tiež veľmi dobre slúži na porovnanie rovnomernej distribúcie informačnej hodnoty.

$$I(X) = \sum_{c_a}^{C} \sum_{c_b}^{C} P(\pi(X) = c_a) P(\pi(X) = c_b), a \neq b \qquad (3.1.17)$$

Analogicky sa počíta potom so vzťahmi (3.1.13), (3.1.14) a (3.1.15).

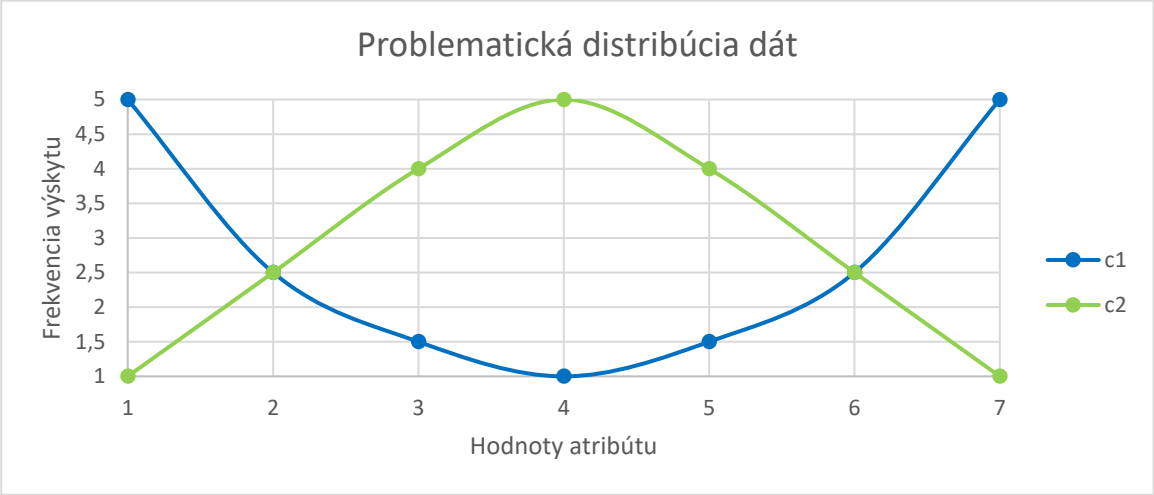
Každá metóda delenia má svoje opodstatnenie a záleží na povahe dát, s ktorými sa pracuje pri tvorbe stromu. V praxi sa ukazuje, že na reálnych dátach jednotlivé metódy produkujú podobné výsledky a nie je tak nutné vždy meniť funkciu pre získanie informačného pomeru [4].

Ďalšie funkcie pre použitie pri výbere rozhodovacieho kritéria sú Chí-kvadrát [16], ktorý sledujú nezávislosť premenných, a twoingovo kritérium [2], ale to sa nevzťahuje priamo na informačný zisk.

Najjednoduchší spôsob delenia nad numerickými hodnotami je nájsť hodnotu α, pre ktorú dosiahne funkcia $Z(a_i)$ z (3.1.15) najmenšiu hodnotu, pričom delenie trénovacej množiny dát X by nasledovala inú definíciu:

$$X_{a_i<\alpha} = \{x \mid a_i(x) < \alpha, \forall x \in X\} \qquad (3.1.18)$$

Rozdelenie týmto spôsobom, ale v ID3 môže spôsobiť nepresnosti pri ďalšom delení, ak by dáta boli distribuované rovnomerne na obe strany vzhľadom na znak c.



Graf 3.1.1: Hypotetická distribúcia dát spôsobujúca problémy pri voľbe rozhodovacieho kritéria

Zdroj: vlastné spracovanie

V prípade využitia predchádzajúceho postupu by sme pre takéto dáta nezískali dostatočne kvalitnú informáciu, ale ak by sme využili *klástrovacie algoritmy,* získame lepšiu informačnú hodnotu. Pri riešení podobného problému v niektorých publikáciách analyzovali povahu každého atribútu a vytvorili riešenie na mieru pre tieto atribúty, napr. delili

numerické dáta na intervaly [2]. Tejto problematike sa v tejto práci ďalej venovať nebudem, ale je veľmi dôležité brať ohľad na podobné problémy so spojitými dátami.

Algoritmus tvorby rozhodovacieho stromu

Algoritmus pre ID3 [29] je veľmi jednoduchý a tvorí ho rekurzívna funkcia tvoriaca vždy aktuálny uzol a rozhoduje či sa jedná o list alebo vnútorný uzol. Originálny algoritmus vytvára len binárny strom [25], ale je ľahko rozšíriteľný pre tvorbu viac ako dvoch vetiev. Z pohľadu užívateľa prebieha kontrola pre prítomnosť len jednej triedy a následne sa realizuje logika pre nájdenie najvhodnejšieho atribútu na rozdelenie dát v aktuálnom uzle. Po nájdení najvhodnejšieho uzlu sa testuje vhodnosť uzlu na použitie ako listu, pokiaľ tento test nie je úspešný, tak sa pokračuje rozdelením dát do ďalších volaní pre tvorbu rozhodovacieho stromu. Použitý atribút sa vyradí z množiny pre určovanie ďalšieho vetvenia a už nebude znovu použitý.

Algoritmus využíva funkciu $Z(a_i)$, ktorú sme si zadefinovali v (3.1.15). Samotný strom bude veľmi košatý a bude dokonale vyhovovať pre použité treňovacie dáta. Je preto potrebné ho zovšeobecniť orezaním.

Orezanie rozhodovacieho stromu

V minulosti existovali snahy vytvoriť pravidlá pre ukončenie generovania stromu pre vytvorenie jednoduchšieho stromu, ale neskôr sa zmenil prístup. Namiesto hľadania týchto pravidiel sa začalo pozerať na zjednodušenie vytvorených stromov a tým pádom na zovšeobecnenie rozhodovacieho stromu [2]. Niektoré z týchto pravidiel sú stále využívané, ale skôr ako doplnok pre zabránenie nadmerného vetvenia. Využité sú v podobe testu na minimálne dostupné množstvo trénovacích dát pri tvorbe uzlu alebo testu pre minimálny informačný zisk.

Rozhodovací strom vytvorený ľubovoľnou metódou by vykazoval vysokú mieru nepresnosti pri použití nad inými dátami, ako nad ktorými bol vytvorený. Tento problém bol vyriešený zmenšením stromu na dostatočne veľkú úroveň, aby dosahoval presné výsledky nad neznámou množinou dát. V praxi dominuje využitie dodatočných dát pre orezanie stromu, ale existujú aj metódy, ktoré realizujú orezanie pomocou dát z trénovacej množiny.

V prípade, že boli poskytnuté priamo tri množiny dát (nazvime ich treňovacie, testovacie a vyhodnocovacie), tak je možné využiť testovacie dáta pre túto operáciu. Tento scenár je v praxi skôr rarita. Ak sú poskytnuté len trénovacie a testovacie dáta, je vhodné rozdeliť trénovacie dáta na tretiny (alebo iný pomer). Jednu podmnožinu využiť pre

orezávanie a ostatné ako trénovacie. Pri poskytnutí len trénovacich dát, čo je najbežnejší prípad, postupujeme analogicky, ale netestujeme nad nezávislými dátami.

Ak poskytnuté množstvo dát je malé a nedá sa rozdeliť na tretiny bez dopadu na kvalitu výsledného rozhodovacieho stromu je vhodné rozdeliť strom na n častí a využiť vždy kombináciu n-1 množín ako celok na trénovanie a n-tú množinu na orezanie.

Všeobecný predpis pre orezanie stromu ID3 pochádza z [29] a nazýva sa generic tree pruning procedure (GPPT). Kľúčová myšlienka algoritmu spočíva v nájdení ľubovoľného podstromu $T|_v$ s najnepresnejšiu klasifikáciou na testovacích dátach v danom uzle v. Ten sa porovnáva s presnosťou klasifikácie podľa dominantnej triedy v jednotlivom uzle alebo s najpresnejšie klasifikujúcim podstromom $T|_u$, pre ktorý platí $(v, u) \in E(T)$. V strome sa následne podstrom $T|_v$ nahradí za nový list klasifikujúci podľa dominantnej triedy alebo podstrom $T|_u$.

Algoritmus pokračuje až dokým nedosiahne algoritmus len jeden uzol (koreň). Následne vyberieme ten, ktorý najpresnejšie klasifikuje dáta z testovacej množiny.

3.1.2 Metóda CART pre konštrukciu rozhodovacích stromov

Metóda CART [2] tvorí vždy binárne stromy, aj keď by napríklad ID3 zvolilo riešenie s viacerými vetveniami. Rozhodovací strom je tvorený rekurzívnymi volaniami tvoriacej funkcie, ale v porovnaní s metódou ID3 nikdy nevylučuje použitý atribút a ponecháva si ho v rozhodovacej množine z pohľadu ID3. Cieľom CART je dosiahnuť najpresnejší možný strom pre trénovacie dáta s minimom nepresností.

Ak definujeme funkciu Y ako funkciu, ktorá vracia 1 pre splnenie podmienky a 0 pre nesplnenie podmienky, tak môžeme nepresnosť rozhodovacieho stromu vyjadriť pomocou funkcie R, ktorá ohodnocuje nepresnosť klasifikácie stromu:

$$R(d) = \frac{1}{|X|} \sum_{x \in X} Y(d(x) \neq c), c = x_{n+1} \tag{3.1.19}$$

Integrálnou súčasťou funkcie $R(d)$ je funkcia d, ktorá sa konštruuje pomocou trénovacích dát a je formou abstrakcie pre vytvorenie a posúdenie presnosti klasifikátora. Funkcia R, ale v prípade trénovacích dát X bude dosahovať najnižšiu možnú hodnotu, nakoľko boli tieto isté dáta použité aj pre tvorbu stromu a tým aj funkcie d. Autori tak využívajú pri ohodnocovaní stromu iné ako trénovacie dáta. V prípade nezávislej trénovacej množine využívajú označenie $R^{ts}(d)$ a v prípade výberu n-tej časti pri delení na n častí popísanej v predchádzajúcej kapitole je použité označnie $R^{cv}(d)$. Ďalej budeme túto

kontrolnú funkciu v prípade akéhokoľvek postupu označovať $R^*(d)$. Funkcia $R^*(d)$ bude vyjadrovať skutočnú nepresnosť stromu a bude využívaná intenzívne pri orezávaní stromu. Cieľom pri tvorbe stromu je výsledok tejto funkcie minimalizovať.

Vzťah (3.1.19) si konkretizujeme pre priblíženie procesu rozhodovania [2]. Ak by sme stanovili podmienenú pravdepodobnosť nesprávnej klasifikácie nad uzlom $v \in L(T)$ ako:

$$r(v) = \sum_{c \in C \setminus \{\kappa(v)\}} p(c|v) \tag{3.1.20}$$

Môžeme potom stanoviť mieru zlej klasifikácie pre ten istý uzol v ako:

$$r(v) = 1 - p(\kappa(v)|v) \tag{3.1.21}$$

Z toho môžeme stanoviť celkovú pravdepodobnosť zlej klasifikácie pre uzol v:

$$R(v) = r(v)p(v) \tag{3.1.22}$$

Pravdepodobnostnú funkciu p vieme vyjadriť pomocou jednoduchej pravdepodobnosti:

$$p(v) = \frac{|X_v|}{|X|} \tag{3.1.23}$$

Funkcia $p(v)$ vyjadruje pravdepodobnosť, že daný vzor bude patriť do uzlu v. X je pôvodná trénovacia množinu a X_v trénovaciu množinu dostupnú pri tvorbe uzlu v.

Pre samotný strom potom funkcia R sa dá vyjadriť ako:

$$R(T) = \sum_{v \in L(T)} R(v) \tag{3.1.24}$$

Existujú klasifikačné prípady, kedy je nutné model penalizovať za zlú klasifikáciu iným ako rovnomerným spôsobom a je nutné vytvoriť funkciu ceny zlej klasifikácie:

$$c(c_i|c_j) \geq 0, c_i \neq c_j, \qquad c_i \in C, c_j \in C \tag{3.1.25}$$

$$c(c_i|c_j) = 0, c_i = c_j, \qquad c_i \in C, c_j \in C \tag{3.1.26}$$

Následne môžeme určiť funkciu, ktorá dosahuje najnižšiu hodnotu chybnej klasifikácie ako:

$$r(v) = \min_{c_i \in C} \sum_{c_j \in C} c(c_i|c_j)p(c_j|v) \tag{3.1.27}$$

Potom implicitne pre $c(c_i,c_j) = 1$ a $i \neq j$:

$$r(v) = \sum_{c_j \in C} c(c_i|c_j)p(c_j|v) = 1 - p(c_i|v) \tag{3.1.28}$$

Pre delenie získame vzťah, ktorý je veľmi dôležitý a je spomenutý nepriamo aj pri ID3, ak by sme uvažovali binárny strom, ale formálne vyjadrený pre CART:

$$R(v) \geq R(u_L) + R(u_R) \tag{3.1.29}$$

Uzol *v* je rodičovský uzol, ktorý odkazuje na dva uzly u_L a u_R. Pri tejto premise, tak jasne vyjadrujeme, že delením stromu získavame menšiu chybu klasifikácie vzoru z trénovacej množiny.

Výber rozhodovacieho kritéria

Pri tvorbe stromu sa uplatňuje princíp informačného zisku, ale hľadá sa jedno najlepšie delenie pre existujúce dáta v danom uzle. Výsledný algoritmus, tak pokračuje až do momentu, ak dosiahne malý informačný zisk alebo je v danom uzle málo dát (autori uvádzajú 5 vzorov). Autori algoritmu CART, ale predostierajú aj alternatívu v podobe nechať strom vyrásť úplne bez ohľadu na komplexnosť stromu.

Tvorba stromu CART sleduje pokles v nečistote uzlu (resp. maximálny informačný zisk) pri každom potencionálnom delení dát:

$$\Delta I(X) = I(X) - p_L I(X_L) - p_R I(X_R) \tag{3.1.30}$$

$$p_L = \frac{|X_L|}{|X|}, p_R = \frac{|X_R|}{|X|}, X_L \cup X_R = X$$

Určenie X_L (množina dát pre ľavý uzol) a tým pádom aj X_R (množina dát pre pravý uzol) nie je triviálny proces, nakoľko autori sa zamerali na tvorbu najlepšieho možného delenia pre jeden atribút a preto môže nastať ľubovoľné delenie. Pre lepšiu ilustráciu poslúži prípad, kde máme atribút a_i, ktorý má štyri hodnoty patriace do H_i. Nech sú tieto hodnoty definované ako $H_i = \{h_{i1}, h_{i2}, h_{i3}, h_{i4}\}$. Najlepšie delenie môže nastať pri CART aj pre prípad, kde platí:

$$X_L = \{x \mid a_i(x) \in \{h_{i1}, h_{i3}\}, \forall x \in X\} \tag{3.1.31}$$

Pri väčšom množstve kategorických atribútov a veľkej trénovacej množine dát *X*, tak získanie vhodného delenia je náročné. Ale s modernými informačnými technológiami je to realizovateľné. V prípade numerických dát autori prezentujú aj prístupy, ktoré hľadajú najlepšie intervaly numerických vhodných na delenie, kde môže výpočetná náročnosť narásť výraznejšie ako pri kombináciách hodnôt atribútov. To poukazuje aj na to ako je použitie rozhodovacích stromov bez moderných informačných technológií náročné.

Pre prísnejšie posúdenie informačného zisku môžeme využiť pravdepodobnosť, že daný uzol *v* bude využitý pre konkrétne dáta. Ak by sme pokles nečistoty v (3.1.30) označili ako $\Delta i(X)$, tak môžeme celkový informačný zisk vynásobiť pravdepodobnosťou použitia daného uzlu *v*.

$$\Delta I(X) = \Delta i(x) p(v) \tag{3.1.32}$$

Funkcia nečistoty môže byť zvolená ľubovoľná z (3.1.12), (3.1.16) a (3.1.17) a ďalších, ktoré boli spomenuté. Autori špecificky spomínajú kritérium entropie (3.1.12), Giniho index (3.1.16) a twoingovo kritérium.

Algoritmus tvorby rozhodovacieho stromu

Pre konštrukciu algoritmu na tvorbu rozhodovacieho si musíme jasne stanoviť ukončovacie podmienky:

- ak všetky trénovacie vzory majú rovnakú triedu,
- počet vzorov v trénovacej množine je malý (napríklad 5, môže byť aj 1),
- najlepšie možné delenie má pokles informačnej nečistoty menšiu ako hodnota β, ktorá je minimálne požadované zlepšenie, resp. max $\Delta I(X) < \beta$.

Pokiaľ je jedna z týchto podmienok splnená, tak bude novo-vytvorený uzol v označený ako list a bude pre neho platiť $\kappa(v) = c$, kde c bude trieda s najväčším počtom vzorov v dostupnej trénovacej množine.

Ďalej je potrebné jasná definícia delenia nakoľko potrebujeme zadefinovať deliacu funkciu s, ktorá môže nadobúdať rôzne správanie podľa potrieb užívateľa. V praxi väčšinou využijeme deliaci mechanizmus sprostredkovaný existujúcim riešením, ale jedna z výhod CART je v možnosti definovať túto funkciu podľa potreby. Autor originálne pracuje s delením s abstraktne, ale v tejto práci ju však zadefinujeme ako funkciu.

Funkcia s bude prijímať trénovacie vzory a bude vracať hodnotu 1 pre vzor spĺňajúci kritéria funkcie s a hodnotu 0 pre vzor nespĺňajúci tieto kritéria. Budeme preto hľadať funkciu zo všetkých prípustných funkcií v množine S, takú pre ktorú platí:

$$s = \underset{s \in S}{\operatorname{argmax}} \Delta I(X) \tag{3.1.33}$$

$$X_L = \{x \mid s(x) = 1, \forall x \in X\}$$

$$X_R = X \setminus X_L$$

Pri hľadaní tejto funkcie využívame vzťah (3.1.30) a zovšeobecňujeme vzťah (3.1.31).

Rozhodovací strom však bude mať rovnaké nedostatky ako strom vytvorený stromom ID3 a je nutné ho orezať.

Orezanie rozhodovacieho stromu

Autori CART vytvorili aj metódu pre orezávanie stromu minimal cost-complexity pruning (MCCP) [2], ktorá sleduje zníženie rozhodovacieho stromu podľa zmenšujúcej sa komplexnosti stromu. Pri orezávaní počiatočného stromu T_{max} budeme hľadať taký strom, ktorý spĺňa najlepšiu možnú presnosť na testovacej množine dát. Predpoklad pre vhodný

počiatočný strom T_{max} je, že musí byť dostatočne veľký, aby sme získali jeho orezávaním optimálny strom pre klasifikáciu. Pre T_{max} platí, že musí byť rozvetvený viac ako optimálny, potom vieme získať optimálny rozhodovací strom. Využívať pri tom budeme funkciu R z (3.1.24) pre posúdenie presnosti stromu.

Orezaný strom T' získame odstránením podstromu $T|_v$ zo stromu T a nahradíme uzol v novým uzlom u, pre túto operáciu definujeme vzťah $T' \preccurlyeq T$. Pre nový list u bude platiť $\kappa(u) = c$ pre takú triedu c, ktorá minimalizuje hodnotu R. Následne budeme hľadať taký strom, pre ktorý platí:

$$R(T') = \min_{T_k \preccurlyeq T} R(T_k) \tag{3.1.34}$$

Ak by sme zoradili orezané rozhodovacie stromy podľa počtu vnútorných uzlov, tak získame sekvenciu stromov: $T_{max} = T_1 > T_2 > T_3 > \ldots > T_n = \{r\}$. Pri orezávaní budeme hľadať taký strom T_k, ktorý bude dosahovať najnižšiu hodnotu $R(T_k)$, nad testovacími dátami, kde k nadobúda hodnoty 1, 2, ..., n. Proces orezávania, tak vytvorí sekvenciu orezaných stromov, ktoré im predchádzajú. Orezávanie je v porovnaní s vytvorením stromu výpočtovo jednoduchší a menej náročný proces.

Pre stanovenie hodnoty nepresnosti vzhľadom na veľkosť stromu sa využíva váha alebo cena za uzol v strome α. Pre cenu platí $\alpha \geq 0$ a môžeme definovať $R_\alpha(T)$ ako:

$$R_\alpha(T) = R(T) + \alpha|L(T)| \tag{3.1.35}$$

Hľadáme taký podstrom $T|_v$ pre odstránenie zo stromu T, pre ktorý platí:

$$R_\alpha(T|_v) = R_\alpha(T|_{v_L}) + R_\alpha(T|_{v_R}) \tag{3.1.36}$$

Pre hľadanie, tak potrebujeme nájsť vždy najmenšiu hodnotu α, ktorá spĺňa (3.1.36). $L(T|_v)$ pozostáva zo všetkých listov v podstrome $T|_v$:

$$R(T|_v) = \sum_{v \in L(T|_v)} R(v) \tag{3.1.37}$$

Následne platí $R_\alpha(T|_v)$ a $R_\alpha(\{u\})$:

$$R_\alpha(T|_v) = R(T|_v) + \alpha|L(T|_v)| \tag{3.1.38}$$

$$R_\alpha(\{v\}) = R_\alpha(v) + \alpha, v \in L(T|_v) \tag{3.1.39}$$

Potom pomocou týchto dvoch vzťahov (3.1.38) a (3.1.39) vieme vyjadriť najmenšiu možnú hodnotu α. Predpokladom je, že v nerovnosti $R_\alpha(T|_v) < R_\alpha(\{u\})$, reprezentuje u nový uzol náhradu podstromu $T|_v$. Vieme tak vyjadriť o α tvrdenie:

$$\alpha < \frac{R(\{u\}) - R(T|_v)}{|L(T|_v)| - 1} \tag{3.1.40}$$

Následne vieme definovať funkciu $g_k(v)$ pre ohodnotenie konkrétneho uzlu, $v \in T_k|_v$:

$$g_k(v) = \begin{cases} \dfrac{R(v) - R(T_t)}{|L(T_k|_v)| - 1}, v \notin L(T_k|_v) \\ \infty, v \in L(T_k|_v) \end{cases} \quad (3.1.41)$$

Hodnoty α_k vieme stanovať ako najmenšiu možná hodnotu funkciu g_k, čím získame aspoň jeden podstrom na orezanie. Formálne je hodnota α_k:

$$\alpha_k = \min_{v \in T_k} g_k(v) \quad (3.1.42)$$

Týmto procesom vieme vyjadrovať najmenšie možné hodnoty α_k pre konkrétny strom T_k a podľa toho vieme vždy nahradiť minimálne jeden vnútorný uzol jedným listom. Ak je viacej podstromov, pre ktoré vyhovujú α_k podľa (3.1.36), tak nahradíme všetky novými vhodnými listami. Pre α_k následne platí, že s každým orezaným stromom sa zväčšuje:

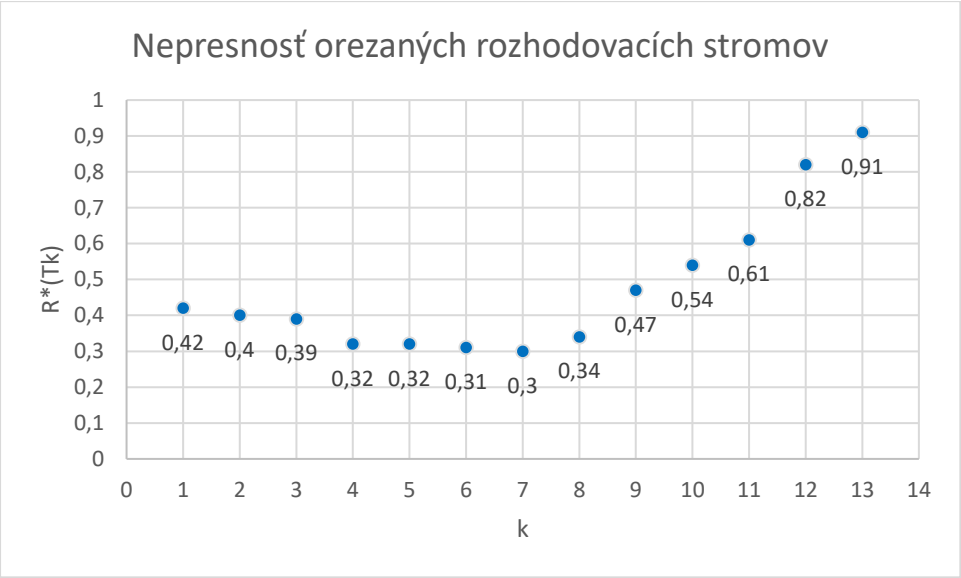
$$\alpha_{k-1} < \alpha_k < \alpha_{k+1} \quad (3.1.43)$$

Pre strom T_1, resp T_{max}, je hodnota $\alpha_1 = 0$, nakoľko sa jedná o neorezaný strom. Postupným hľadaním hodnôt α_k sa strom zmenšuje a orezáva, až kým sa dostaneme k jednému koreňovému uzlu.

Ak takto získame všetky orezané rozhodovacie stromy, tak je potrebné vybrať ten najvhodnejší. Výber sa realizuje podľa miery nepresnosti klasifikácie $R^*(d)$. Testovacie dáta sa použijú pre každý orezaný strom a vyberie sa ten najpresnejší. Formálne je možné vyjadriť tento vzťah ako:

$$R^*(T_{optimal}) = \min_k R^*(T_k) \quad (3.1.44)$$

Pre ilustráciu toho ako môžu hodnoty $R^*(T_k)$ vyzerať v praxi, môžeme použiť graf vytvorený na základe údajov autorov [2]. Na grafe uvidíme nepresnosť klasifikácie jednotlivých orezaných stromov a z nich môžeme na základe tejto informácie vybrať najvhodnejší z nich.



Graf 3.1.2: Znázornenie nepresnosti rozhodovacích stromov T_k nad testovacími dátami, kde k = 1, 2, ..., 13

Zdroj: vytvorené na základe údajov z [2]

Najmenšiu nepresnosť, resp. najvyššiu presnosť, dosahujú orezané stromy v intervale 4 až 7 a každý z nich môže byť vhodný kandidát. Pri originálnych dátach by bol najvhodnejší strom T_7. Pri štandardizovaní chyby vieme identifikovať vhodné rozhodovacie stromy podobne ako v štatistike.

$$\mathrm{R}^*(\mathrm{T}_k) \in \langle R^*(T_o) - SE(T_o); R^*(T_o) + SE(T_o)\rangle$$

$$T_o = T_{optimal}$$

Ľubovoľný strom s nepresnosťou spadajúcou do intervalu po zohľadnení chyby sa dá považovať za vhodný.

Štandardnú chybu SE (standard error) definujeme ako:

$$SE(T) = \sqrt{\frac{R^*(T)(1 - R^*(T))}{|X_{test}|}} \tag{3.1.45}$$

V niektorých prípadoch môže byť vhodné štandardnú chybu prenásobiť koeficientom b, ak tolerujeme napríklad väčšiu chybu za cenu jednoduchšieho stromu, ale je nutné dbať na to, že príliš jednoduchý strom nemusí byť dostatočne presný.

$$R^*(T_k) \in \langle R^*(T_o) - b \cdot SE(T_o);\ R^*(T_o) + b \cdot SE(T_o)\rangle \tag{3.1.46}$$

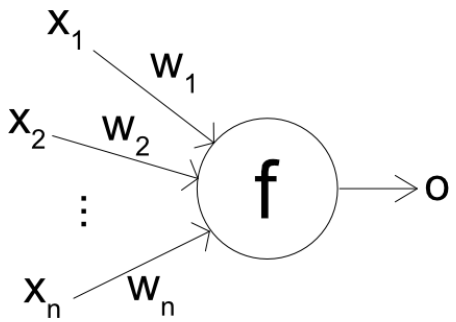
$$T_o = T_{optimal}$$

Metódy orezávania MCCP, GPPT alebo iné je možné aplikovať na rôzne rozhodovacie stromy, ak majú k dispozícii požadované informácie alebo ich je možné dopočítať.

3.2 Neurónové siete s dopredným šírením

Pôvod inšpirácie neurónových sietí nájdeme v biologických organizmoch a ich neurónových sieťach [27]. Neurónové siete nadobúdajú v matematických disciplínach rôzne podoby ako radiálne, kohenenove, rekurentné, konvolučné a mnohé ďalšie. V tejto práci sa budeme zaoberať len jedným druhom a to *neurónové siete s dopredným šírením* (*feed-forward neural networks*) [17] využívajúce algoritmus spätného šírenia chyby.

Základnou súčasťou neurónových sietí je neurón, ktorý plní základnú rozhodovaciu funkciu. Existujú mnohé definície neurónu, ale my budeme nasledovať definíciu podľa [22]. Každý neurón prijíma vstupy x_i a každý z týchto vstupov má priradenú váhu w_i. Neurón vracia na výstupe jednu hodnotu o. V prípade sigmoid neurónu, ktorý budem požívať v tejto práci sa bude hodnota nachádzať v intervale (0; 1).



Obrázok 3.2.1: Znázornenie matematického neurónu

Zdroj: [22]

V tejto práci však budeme pracovať len so sigmoid neurónmi, pre ktoré vieme povedať, že pri hodnote $z \rightarrow \infty$ platí limitne $f(z) = 1$ a podobne pri $z \rightarrow -\infty$ limitne platí $f(z) = 0$. Sigmoid funkcia má nasledujúci tvar:

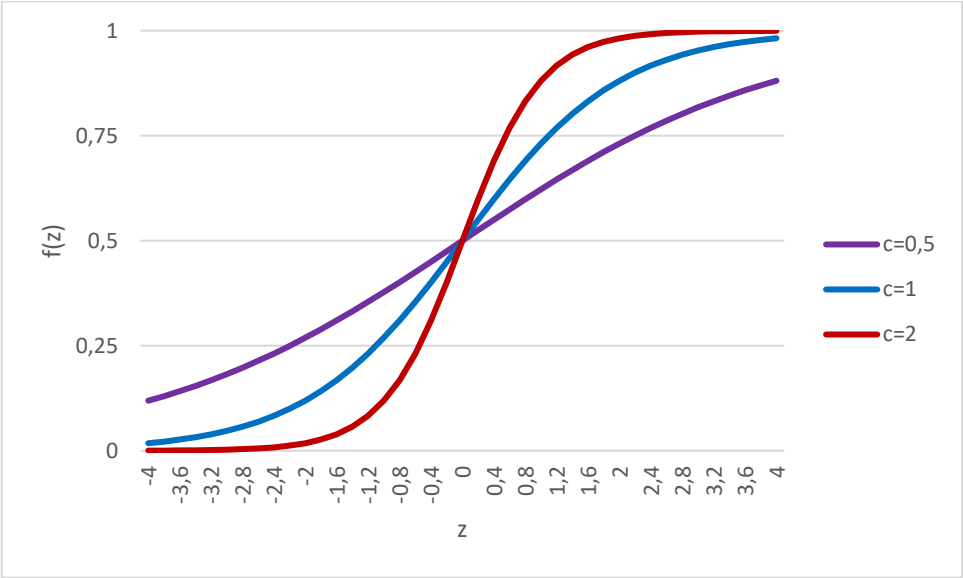
$$f(z) = \frac{1}{1 + e^{-cz}} \tag{3.2.1}$$

Konštanta c sa používa pre modifikáciu strmosti funkcie, ak chceme modifikovať vyhodnocovanie neurónu. V tejto práci budeme pracovať s $c = 1$. Sigmoid funkcia sa tiež označuje v iných prácach ako logistická funkcia.

Hodnotu z vypočítame ako lineárnu kombináciu zo všetkých vstupov a ich váh. Ďalšia dôležitá premenná je b alebo bias, ktorý sa správa podobne ako konštanta v lineárnej regresií a koriguje tak vstupnú hodnotu neurónu:

$$z = b + \sum_{i=1}^{n} w_i x_i \tag{3.2.2}$$

Sigmoid funkcia má požadované vlastnosti pre vyhodnotenie potenciálu vstupu na spojitom intervale a pri použití viacerých neurónov v neurónovej sieti tak vieme modifikovať jednotlivé váhy a biasy pre trénovacie dáta a získať tak internú logiku pre vyhodnocovanie vstupných dát.



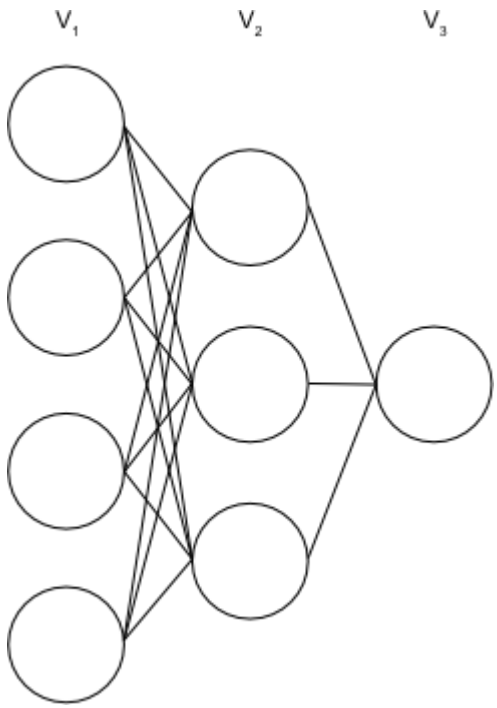
Graf 3.2.1: Hodnoty nadobúdane sigmoid neurónom/funkciou

Zdroj: vlastné spracovanie

Pre získanie požadovaných vlastností neurónových sietí potrebujeme neuróny zorganizovať do grafu, ktorý bude topológiou neurónovej siete. Neurónová sieť pri trénovaní bude hľadať najlepšie možné hodnoty pre váhy jednotlivých spojení medzi neurónmi a vhodných hodnôt biasov. Na základe toho môžu neuróny získať rôzne hodnoty a tým vedia aproximovať vhodnú stratégiu na vyhodnocovanie vstupných vzorov.

Problematickým aspektom neurónovej siete je vysvetlenie toho ako funguje nakoľko si hľadá vhodnú konfiguráciu pre dvojicu vstupného vzoru a výstupného vzoru. Vďaka svojej schopnosti poňať širokú škálu možných konfigurácií dokážu neurónové siete riešiť širokú paletu problémov (rozoznanie obrázkov, vyhodnocovať relatívne komplexné otázky), ale nevieme vysvetliť pri pohľade na konfiguráciu siete, prečo funguje tak ako funguje. Jednotlivé váhy a biasy sú výsledkom optimálnych (suboptimálnych) hodnôt pre trénovacie dáta. Neurónové siete sú v tejto rovine „čierne skrinky“.

Ak by sme chceli vytvoriť neurónovú sieť zo 4 vstupných neurónov, 3 vnútorných a jedného výstupného, tak získame graf ako na obrázku (3.2.2). Topológiu siete pri takejto konfigurácií zapisujeme ako 4 – 3 – 1, pričom platí, že každé číslo reprezentuje počet neurónov v danej vrstve.



Obrázok 3.2.2: Jednoduchá neurónová sieť s topológiou 4 – 3 – 1

Zdroj: vlastné spracovanie, podľa [18]

V prvej vrstve neurónovej siete sa nenachádzajú žiadne reálne matematické neuróny, ale vstupy z jednotlivých vzorov *X*. Neuróny v každej ďalšej vrstve predstavujú už skutočné neuróny, ktoré počítajú svoje hodnoty. V poslednej vrstve V_3 je uvedený len jeden výstupnú neurón, ale môže ich byť viacej. Pri neurónových sieťach rozlišujeme vstupnú vrstvu (V_1), vnútorné vrstvy (V_2), a výstupnú vrstvu V_3. Ak je vnútorných vrstiev viacej ako jedna, tak sa jedná o hlbokú neurónovú sieť (deep neural network) [22].

Neurónovú sieť môžeme formálne zapísať ako $NS = (N, C, I, O, w, b)$ [18]. N predstavuje množinu všetkých neurónov. C je množina všetkých hrán $N \times N$ medzi neurónmi. I je množina vstupných neurónov a platí, že $I \subseteq N$, resp. sú podmnožinou všetkých neurónov. O je množina výstupných neurónov a platí $O \subseteq N$, resp. sú podmnožinou všetkých neurónov. W je vektor váh jednotlivých hrán, $w : C \rightarrow R$, resp. nadobúdajú reálne hodnoty pre každú hranu. Vektor b sú hodnoty jednotlivých biasov neurónov, $b : N \rightarrow R \setminus I$, resp. nadobúdajú reálne hodnoty pre každý neurón okrem vstupných.

Samotné neuróny je možné formálne zatriediť do vrstiev V_i, ak $i \in \{1, 2, ..., n\}$ a $V_1 = X$ a $V_n = Y$. Pre jednotlivé vrstvy platí:

$$V_i \cap V_j = \emptyset, i \neq j \tag{3.2.3}$$

Samotné vyhodnotenie neurónovej siete je možné, ak dosadíme vstupný vzor *x* do vstupných „neurónov“ (ako bolo spomenuté, nejedná sa o skutočné neuróny) a prepočítame

hodnoty všetkých neurónov postupne od V_1 po V_n. Ak tento proces zadefinuje ako volanie funkcie $y(x)$, tak môžeme zadefinovať:

$$y(x) = a \tag{3.2.4}$$

Pričom za a považujeme požadovaný výstup pri vzore x. Pokiaľ vieme vyhodnotiť neurónovú sieť, tak môžeme zadefinovať jej chybu ako spriemerovanú štvorcovú chybu (MSE – mean squared error) [22]:

$$E(w,b) = \frac{1}{2n} \sum_{x \in X} \|y(x) - a\|^2 \tag{3.2.5}$$

Používa sa aj celková štvorcová chyba (SSE – sum of squared errors) [27]:

$$E(w,b) = \frac{1}{2} \sum_{x \in X} \|y(x) - a\|^2 \tag{3.2.6}$$

Pričom platí, že $y(x)$ reprezentuje nami vytvorenú sieť, ktorú sa snažíme optimalizovať, resp. vytvoriť. Ak poznáme chybu, tak z nej je odvodený algoritmus známy pod názvom spätné šírenie chyby (backpropagation), ktorý vieme použiť na odhad (optimalizáciu) hodnôt jednotlivých váh a biasov jednotlivých neurónov.

3.2.1 Algoritmus spätného šírenia chyby

Pri *sieťach s dopredným šírením* je tento algoritmus jednou z najefektívnejších volieb. Je možné ho použiť aj na rôznych akcelerátoroch ako grafických kartách, čo robí výrazne urýchľuje tvorbu neurónových sietí [22]. V tejto práci sa nebudeme zaoberať použitím takýchto prostriedkov, ale je veľmi dôležité si uvedomiť, že široké rozšírenie neurónových sietí bolo umožnené vďaka tejto technológií. Ako uvidíme pri definícií algoritmu spätného šírenia, tak proces optimalizácie je veľmi náročný na výpočtové prostriedky, čo bolo najväčšou bariérou pri praktickom použití neurónových sietí. V praxi sa totiž stretneme s neurónovými sieťami, ktoré majú aj 5 a viacej skrytých vrstiev s veľkým počtom neurónov v každej vrstve, čo vyžaduje veľké množstvo hrubej výpočtovej sily pri hľadaní správnych hodnôt váh a biasu.

Algoritmus spätného šírenia chyby mení nastavenia siete smerom od výstupu k vstupu a mení jednotlivé hodnoty váh na základe aktuálnej hodnoty chyby vzhľadom na konkrétnu váhu alebo bias [22, 27]. Výsledkom sú tak malé inkrementálne zmeny, ktoré sú udávané smernicou zmeny pri náraste hodnoty (prvá derivácia). Pri ďalších vzťahoch budeme uvažovať len jeden trénovací vzor. Zmena nastáva pri aplikácií každého vzoru cez

celý proces spätného šírenia. Pôvodné hodnoty váh a biasu sa pred použitím prvého vzoru pre trénovanie vyberú náhodne.

$$\Delta w_{i,j} = -\gamma \frac{\partial E}{\partial w_{i,j}} \tag{3.2.7}$$

Zmena ľubovoľnej *i*-tej váhy pre *j*-ty neurón je dosiahnutá ako parciálna derivácia funkcie chyby. Pre lepšie pochopenie je možné predstaviť si tento proces ako približovanie sa optimálnej hodnote váhy vzhľadom na vstup *x*, aby (3.2.5) alebo (3.2.6) dosahovalo najmenšie možné hodnoty. Konštanta γ určuje mieru rýchlosti učenia sa a slúži ako parameter, ktorý reguluje rýchlosť učenia sa siete. Ak je príliš malý, tak sa sieť učí pomaly, a ak je veľký, tak sa sieť učí rýchlo, ale je nepresnejšia vo výsledku. Podobný vzťah platí pre bias, ktorý sa pre uľahčenie výpočtu môže zadefinovať ako *n*+1-tá váha, ak *n* je počet všetkých váh pre neurón *j*, so vstupom rovným jedna [27, 29]. Zadefinovanie biasu ako váhy zjednodušuje vektorizáciu jednotlivých vstupov a výpočtov, čo zrýchľuje výpočty na súčasných počítačoch [22].

Ak vstup o_i do neurónu *j* z neurónu *i* chápeme ako konštantu pre konkrétnu váhu $w_{i,j}$, tak dostaneme:

$$\frac{\partial E}{\partial w_{i,j}} = -o_i \frac{\partial E}{\partial o_i w_{i,j}} \tag{3.2.8}$$

Smernicu chyby v neuróne *j* zadefinuje pod znakom δ:

$$\frac{\partial E}{\partial w_{i,j}} = o_i \delta_j \tag{3.2.9}$$

Následne získame vzťah, pre zmenu váhy v podobe:

$$\frac{\partial E}{\partial w_{i,j}} = -\gamma o_i \delta_j \tag{3.2.10}$$

Smernicu chyby vieme zadefinovať z derivácie funkcie sigmoid neurónu pre výstupnú vrstvu ako:

$$\delta_j = f'(z_j)(o_j - a_j) = o_j(1 - o_j)(o_j - a_j) \tag{3.2.11}$$

Pre schované alebo vnútorné vrstvy neurónu vo vrstve *i* musíme tento vzťah vypočítať z nasledujúcej vrstvy *i*+1:

$$\delta_j = f'(z_j) \sum_{q \in V_{i+1}} w_{jq}\delta_q = o_j(1 - o_j) \sum_{q \in V_{i+1}} w_{jq}\delta_q \tag{3.2.12}$$

Pod *q* označujeme neuróny z vrstvy V_{i+1}, do ktorých vstupuje výstup o_j z neurónu *j*.

Na záver nám stačí zmenu váhy aktualizovať:

$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j} \tag{3.2.13}$$

Pri algoritme spätného šírenia chyby je jediné kritérium nájsť deriváciu funkcie neurónu. Ak je tento predpoklad splnený, je možné túto funkciu použiť ako aktivačnú funkciu neurónu. Jedna z populárnych aktivačných funkcií je aj hyperbolický tangens:

$$f(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \tag{3.2.14}$$

Algoritmus spätného šírenia chyby pracuje potom v nasledujúcich krokoch:

1. prechod neurónovou sieťou pre konkrétny vzor a zapamätanie si jednotlivých výstupov z neurónov a smernice zmeny pre jednotlivé neuróny,
2. výpočet zmien váh a biasov (3.2.7),
3. aktualizácia jednotlivých váh a biasov (3.2.13),
4. opakuj dokým sú dostupné vzory v trénovacej množine.

V praxi sa učenie po jednom vzore nerealizuje vždy, nakoľko je prechod väčšou neurónovou sieťou veľmi náročný a zmeny sa realizujú v dávkach (X_d), kde sa potom zmena váh a biasu rieši kumulatívne pre ľubovoľnú váhu w:

$$w \leftarrow w + \sum_{m \in X_d} \Delta w_m \tag{3.2.15}$$

Zloženie dávok pre aktualizáciu siete by malo byť náhodné, aby dáta s podobným správaním neovplyvnili neurónovú sieť s veľmi veľkými zmenami špecifickými pre tieto dávky. Tento proces sa nazýva *stochastický gradientný zostup* (SGD – *stochastic gradient descent*).

3.2.2 Príprava dát pre neurónové siete

Neurónové siete dokážu vyriešiť veľmi širokú škálu problémov, ale je nevyhnutné pre ich trénovanie predspracovať dáta [28]. Typicky sa využívajú 2 prístupy a to normalizácia a škálovanie/štandardizácia. Pri použití kvalitatívnych charakteristík je vhodné vstupné dáta normalizovať.

Normalizácia štandardne prevedie všetky hodnoty do intervalu ⟨0;1⟩. Pre normalizáciu využijeme vzťah:

$$n_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{3.2.16}$$

Neurónové siete s dopredným šírením sú veľmi citlivé na vstupné dáta a ich výsledky budú priamo závisieť od vstupných dát. Najrýchlejšie trénovania sa nám osvedčilo na dátach

v intervale ⟨-1; 1⟩. Pre získanie hodnôt n_i na interval ⟨-1; 1⟩ využijeme modifikovaný vzťah (3.2.16):

$$n_i = 2\frac{x_i - \min(x)}{\max(x) - \min(x)} - 1 \tag{3.2.17}$$

Škálovanie/štandardizácia je dosiahnutá premietnutím hodnoty do normovaného normálneho rozdelenia:

$$s_i = \frac{x_i - \bar{x}}{\sigma} \tag{3.2.18}$$

Pri normalizácií je veľmi dôležité brať aj ohľad na význam danej premennej použitej pri trénovaní, pretože chyba (3.2.5) a (3.2.6) je počítaná z pôvodných hodnôt a napríklad menej významná hodnota na výstupnom neuróne môže mať rovnakú váhu ako významnejšia hodnota v inom výstupnom neuróne. Je preto vhodné zvážiť správny rozsah hodnôt. Ak je cieľom zachovať relatívnu distribúciu dát je vhodné využiť škálovanie.

Pri kategorických dátach budeme pracovať s každou potencionálnou hodnotou, ktorú bude nadobúdať ako samostatným neurónom. Premenná, ktorá bude nadobúdať 8 hodnôt, bude reprezentovaná 8 neurónmi. Tie sa budú spínať v stave 0 a 1 alebo -1 a 1 podľa spôsobu normalizácie.

3.3 Verifikácia modelov strojového učenia

Každá metóda strojového učenia využíva počas učenia nejakú metódu pre vyhodnotenie chyby interne pre konštrukciu alebo optimalizáciu modelu. Pri rozhodovacích stromoch sa sleduje maximalizácia informačného zisku, pri neurónových sieťach to býva typicky zníženie MSE (mean squared error) alebo SSE (sum of squared errors).

Pre správne ohodnotenie modelu sa v praxi využívajú normálne tri dátové množiny, ktoré sú rozdelené na trénovacie, testovacie a vyhodnocovacie. Dáta sú trénované na trénovacích a model sa optimalizuje pomocou testovacích, ale samotná presnosť a kvalita sa stanovuje podľa vyhodnocovacích. Pri obmedzenom množstve dát sú testovacie a vyhodnocovacie jedna a tá istá množina.

V tejto práci sa budeme venovať binárnym klasifikačným problémom a pre vyhodnocovanie klasifikačných problémov môžeme využiť veľmi jednoduchý a efektívny spôsob v podobe *confusion matrix* („matice zhody“) [8]. V matici sa porovnávajú pôvodné hodnoty z (testovacích) dát a hodnoty pochádzajúce z výsledného modelu.

Tabuľka 3.3.1: Predpis confusion matrix

Orig./Model	N	P
N	TN	FP
P	FN	TP

Zdroj: spracované podľa [8]

Vysvetlenie jednotlivých buniek:

- TN – true negatives, negatívne hodnoty klasifikované ako negatívne,
- TP – true positives, pozitívne hodnotu klasifikované ako pozitívne,
- FP – false positives, negatívne hodnoty klasifikované ako pozitívne,
- FN – false negatives, pozitívne hodnoty klasifikované ako negatívne.

Negatívne a pozitívne predstavujú dve triedy, pre ktoré sa snažíme dané hodnoty klasifikovať. Ich pomenovanie má dôvod v stratégií, ktorú chceme zvoliť alebo získať od modelu. V prípade rizikového poistenia budeme preferovať model, ktorý má vysokú hodnotu FP a nízku hodnotu FN, za predpokladu, že P predstavuje rizikovejšiu skupinu. Pri modeli predpokladajúci vhodnosť pacienta na podstúpenie vysoko rizikovej operácie srdca, pri ktorej by mohol zomrieť, budeme preferovať model s nízkym FP a vysokým FN.

Pre lepšiu prácu s vyhodnocovaním modelu zavedieme ďalšie indikátory ako *fp rate*, *tp rate*, *precision* (precíznosť), *recall (sensitivity)*, *specificity*, *accuracy* (presnosť) [8]:

$$fp\ rate = \frac{FP}{TN + FP} \tag{3.3.1}$$

$$tp\ rate = \frac{TP}{TP + FN} \tag{3.3.2}$$

$$precision = \frac{TP}{TP + FP} \tag{3.3.3}$$

$$recall = sensitivity = \frac{TP}{TP + FN} \tag{3.3.4}$$

$$specificity = \frac{TN}{TN + FP} \tag{3.3.5}$$

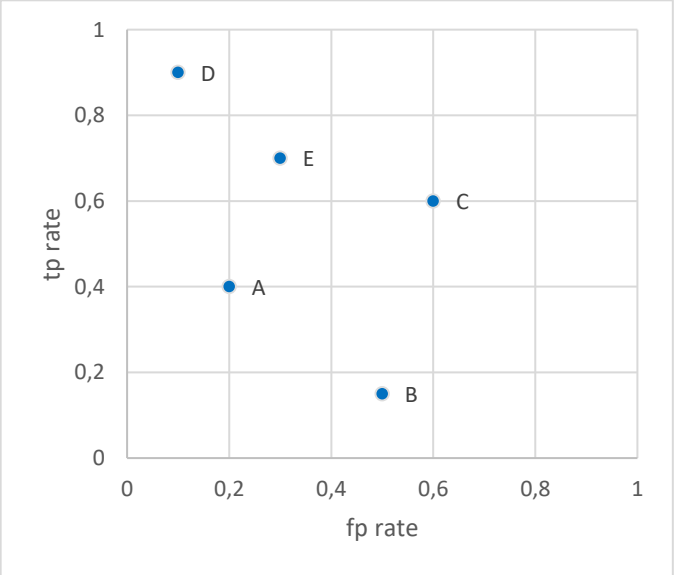
$$accuracy = \frac{TP + TN}{TN + FP + FN + TP} \tag{3.3.6}$$

Pre vyhodnocovanie kvality modelu je potrebné určiť si stratégiu ako bolo spomenuté a klásť dôraz na potrebný indikátor. Univerzálnym pravidlom je, že cieľom je získať hodnoty *sensitivity*, *specificity* a *accuracy* najvyššie možné. Niekedy však nie je možné takýto model získať. Napríklad model nevie rozlišovať hraničné hodnoty dostatočne presne a musíme

pristúpiť ku kompromisom alebo získať viacej dát. Alternatívou je aj iná forma predspracovania dát, ktoré zvýšia rozlišovacie schopnosti modelu.

Pre vizuálne znázornenie kvality modelu použijeme základný ROC graf. Na *x*-ovej osi premietneme *fp rate* a na *y*-ovej osi *tp rate* pre jednotlivé modely. Cieľom je získať model najbližší alebo rovný vektoru (*x*; *y*) = (0; 1).

Modely pri, ktorých *tp rate* a *fp rate* sú totožné hodnoty, sú nespoľahlivé, lebo majú 50 % pravdepodobnosť správnej klasifikácie, čo je totožné s náhodným určením výsledku podľa pomeru výskytu vzoru v množine dát. Modely, pri ktorých *fp rate* je väčšie ako *tp rate*, vieme vyhodnotiť ako modely s opačným učením. Ak by sme výsledky modelu s *tp rate* = 0 a *fp rate* = 1 negovali, tak vieme získať spoľahlivý model s *tp rate* = 1 a *fp rate* = 0.



Graf 3.3.1: Ukážka základného ROC grafu

Zdroj: vlastné spracovanie podľa [8]

Na grafe 3.3.1 si môžeme všimnúť, že bod D(0,1; 0,9) by reprezentoval model, ktorý dokáže správne identifikovať 90% všetkých pozitívnych znakov a 10% znakov by chybne identifikoval negatívnych znakov. Takýto model by bola najlepšia voľba. V porovnaní s ním model reprezentovaný bodom B(0,5; 0,15) by dosahoval lepšie výsledky, ak by všetky jeho výstupy negovali, v praxi je však nepoužiteľný. Bod C(0,6; 0,6) je neutrálny a dosahuje rovnakú mieru chybovosti ako keby sme náhodným výberom vybrali znak vzoru podľa distribúcie pravdepodobnosti v testovacích dátach. Je tak neutrálny a tiež v praxi nepoužiteľný.

V prípade väčšieho množstva znakov budeme využívať nasledujúci pohľad na získané výsledky v *confusion matrix*.

Tabuľka 3.3.2: Rozšírený predpis *confusion matrix*

Orig./Model	**A**	**B**	**C**
A	AA	AB	AC
B	BA	BB	BC
C	CA	CB	CC

Zdroj: vlastné spracovanie

4 Výsledky práce

Jednotlivé metódy uvedené v časti 3 budú v tejto kapitole aplikované na množiny dát v programovacom prostredí R. Pri tvorbe modelov strojového učenia bolo využité prostredie jazyka R [26]. Využité boli pri tom knižnice:

- *Weka* [33] a *Rweka* [14] pre ID3,
- *rpart* [30] pre CART a *rpart.plot* [21] pre vizualizáciu CART,
- *neuralnet* [11] pre trénovanie neurónových sietí,
- *caTools* [31] pre delenie dát na trénovacie a testovacie.

Použitá implementácia stromu ID3 nepodporuje orezávanie a prácu s numerickými hodnotami. Použité tak boli náhrady v podobe kategorických premenných, ktoré budú vysvetlené neskôr.

Pre tvorbu stromov ID3 je potrebné nainštalovať prostredie Java pre spustenie knižnice *Weka* na pozadí. Tvorba stromu ID3 bola následne realizovaná pomocou príkazov:

```
library(Rweka)

WPM(„refresh-cache“)
WPM(„install-package“, „simpleEducationalLearningSchemes“)
WPM(„load-package“, „simpleEducationalLearningSchemes“)

ID3 <- make_Weka_classifier(„weka/classifiers/trees/Id3“)
tree <- ID3(y ~ x1 + x2,
            data = data)
test_pred <- predict(tree, test_data, type = „class“)
```

Príkazy uvedené vo funkcií *WPM* pracujú priamo s rozhraním prostredia *Weka* a aktualizujú dostupnú knižnicu pre toto prostredie. Metóda ID3 sa nachádza v balíku prostredia *Weka* pomenovanom *simpleEducationalLearningSchemes*. Nasledovné príkazy vytvoria objekt pre tvorbu stromov ID3. Ten následne použijeme na vytvorenie stromu nasledovaným predikciou tried pre testovacie dáta.

Tvorba stromov CART je realizovaná jednoduchšie v porovnaní s ID3 a stačia na to tri príkazy a posledné dva sa venujú vizualizácií stromu a jeho orezaniu:

```
library(rpart)

tree <- rpart(y ~ x1 + x2,
              data = data,
              method = „class“)
test_pred <- predict(tree, test_data, type = „class“)
rpart.plot(tree) # vizualizacia
ptree <- prune(tree, cp=ZVOLENE_CP) # orezanie stromu
```

Pre tvorbu neurónových sietí využívame spomínanú knižnicu *neuralnet*, ktorú potrebujeme dodatočne nastaviť. Mieru učenia γ nastavíme na 0,01 pre všetky prípady. Použitá metóda pre odhad chyby bude SSE (3.2.6).

```
library(neuralnet)

nn <- neuralnet(y ~ x1 + x2,
                data = data,
                threshold = 0.01,
                hidden = POCET_NEURONOV_V_SKRYTEJ_VRSTVE,
                algorithm = „backprop“,
                act.fct = „logistic“,
                err.fct = „sse“,
                learningrate = 0.01,
                linear.output = FALSE)
test_pred <- compute(nn, test_data)$net.result
```

Pre odhad je nutné využiť funkciu *compute* poskytovanú balíkom *neuralnet* a následne výsledky spracovať. V prípade klasifikačného problému, kde je prítomný len jeden výstupný neurón, budeme výsledky transformovať pomocou jednoduchej nerovnosti. Ak bude hodnota výstupnej pravdepodobnosti väčšia ako 0,5, tak budeme výsledok považovať za pozitívny inak za negatívny. Je možné tiež túto hraničnú hodnotu optimalizovať pomocou ROC analýzy [8].

Použitie knižnice *caTools*, ktorá dokáže vzhľadom na zvolený znak rozdeliť rovnomerne originálnu množinu dát. V tejto práci bude rozdelená originálna množina dát na trénovaciu a testovaciu v pomere 80:20. Knižnicu použijeme pri delení originálnych dát týkajúcich sa úverových žiadostí pri posudzovaní kreditného rizika. Príklad použitia tohto nástroja:

```
library(caTools)

split = sample.split(data$y, SplitRatio = 0.8)
data = subset(data, split == TRUE)
test_data = subset(data, split == FALSE)
```

4.1 Klasifikovanie poistencov do rizikových skupín

Prvá množina dát bude vygenerovaná pre prípad klasifikácie poistencov v KASKO poistení. Ako inšpiráciu pre generovanie dát boli použité rôzne formuláre a dotazníky, ale hlavná inšpirácia bol dotazník od poisťovne Union [37]. Jednotlivé rizikové skupiny budú reprezentovať rozdielne rizikové prirážky, ktoré poisťovňa aplikuje na poistnom.

Generovaných bude 8 premenných: *značka automobilu*, *trieda automobilu* (podľa hodnoty), *lokalita poistiteľa*, *počet predchádzajúcich poistných udalostí poistiteľa*, *vek poistiteľa*, či je auto využívané na *rodinné účely*, *maximálna nosnosť automobilu*, *vek auta*, *chyba*. Tieto dáta následne skombinujeme a vyhodnotíme do rizikovej skupiny, ktorá bude reprezentovať prirážku na poistnom. Pri vygenerovaných dátach nebude existovať korelácia medzi dátami, ktorá existuje v reálnom svete, čo môže spôsobovať problémy. Jednotlivé riziká budú následne sčítané. Vygenerovaných bude 2000 trénovacích údajov a 2000 testovacích údajov.

Tabuľka 4.1.1: Charakteristika premenných pre zaradenie poistencov do rizikových skupín

Názov	Hodnoty	Zastúpenie/rozdelenie
Značka (z)	A, B, C, D	25%, 25%, 25%, 25%
Trieda (t)	N, S, V	50%, 40%, 10%
Lokalita (l)	L1, L2, L3	60%, 20%, 20%
Počet predch. PÚ (n)	0, 1, 2, ...	$N \sim P(0,1)$
Vek (v)	⟨18;100⟩	$V \sim \chi^2(17)$
Rodinné účely (u)	R, I	80%, 20%
Maximálna hmotnosť (h)	⟨1,1; 3,5⟩	$H \sim \chi^2(1,7)$
Vek auta (w)	⟨0; 15⟩	$W \sim \chi^2(4)$
Typ motora (m)	B, D	40%, 60%
Chyba €	⟨-∞; ∞⟩	$E \sim N(0, 1/32^2)$

Zdroj: vlastné spracovanie

Hodnoty premenných následne transformujeme pravdepodobnostnými funkciami a určíme tak ako mierou budú prispievať k rizikovosti.

$$p_{značka}(z) = \begin{cases} 0{,}02 & z = A \\ 0{,}05 & z = B \\ 0{,}03 & z = C \\ 0{,}08 & z = D \end{cases}$$

$$p_{trieda}(t) = \begin{cases} 0{,}02 & t = N \\ 0{,}04 & t = S \\ 0{,}01 & t = V \end{cases}$$

$$p_{lokalita}(l) = \begin{cases} 0{,}05 & l = L1 \\ 0{,}02 & l = L2 \\ 0{,}01 & l = L3 \end{cases}$$

$$p_{počet\ PÚ}(n) = 0{,}2n$$

$$p_{vek}(v) = e^{-0{,}077016(v\ -6{,}102647)}$$

$$p_{rod.účely}(u) = \begin{cases} 0{,}02 & u = R \\ 0{,}1 & u = I \end{cases}$$

$$p_{nax.hmotnosť}(h) = 0{,}02h$$

$$p_{vek\ auta}(w) = 0{,}01w$$

$$p_{typ\ motora}(m) = \begin{cases} 0{,}02 & m = B \\ 0{,}05 & m = D \end{cases}$$

Pravdepodobnostná funkcia veku je odvodená na základe úvahy, pri ktorej mladší vodiči sú rizikovejší ako starší. Riadi sa nasledujúcim predpisom:

$$y = e^{-a(x+b)}$$

$$[18; 0{,}4]: 0{,}4 = e^{-a(18+b)}$$

$$[27; 0{,}2]: 0{,}2 = e^{-a(27+b)}$$

Výsledná pravdepodobnosť rizikovosti je realizovaná pomocou lineárnej rovnice:

$$\begin{aligned} p_i = \min(&p_{značka}(z_i) + p_{trieda}(t_i) + p_{lokalita}(l_i) \\ &+ p_{počet\ PÚ}(n_i) + p_{vek}(v_i) \\ &+ p_{rod.účely}(u_i) + p_{max.hmotnosť}(h_i) \\ &+ p_{vek\ auta}(w_i) + p_{typ\ motora}(m_i) \\ &+ e_i, 1) \end{aligned} \tag{4.1.1}$$

Následne priradíme každému vzoru triedu c z množiny všetkých tried (*RP0*, *RP1*, *RP2*, *RP3*):

$$c(p) = \begin{cases} RP0 & 0 \leq p \leq 0{,}25 \\ RP1 & 0{,}25 < p \leq 0{,}5 \\ RP2 & 0{,}5 < p \leq 0{,}75 \\ RP3 & 0{,}75 < p \leq 1 \end{cases} \tag{4.1.2}$$

$$c_i = c(p_i)$$

4.1.1 Aplikácia metódy ID3

V prípade stromu ID3 bolo nutné premenné vek, počet poistných udalostí a vek auta previesť na kategorické premenné. Vek bol rozdelený v intervaloch ⟨15; 25), ⟨25; 35), atď.,

počet poistných udalostí bol prevedený na triedy v pomere 1:1, hmotnosť bola rozdelená do piatich intervalov odstupňovaných po 0,5 tony, vek auta bol odstupňovaný po 3 rokoch. Pri tomto strome nie je možné aplikovať orezania vzhľadom na chýbajúcu implementáciu [39].

Tabuľka 4.1.2: Výsledky predikcie neorezaného stromu ID3 pre klasifikáciu poistencov do rizikových skupín

		predikcia			
		RP0	**RP1**	**RP2**	**RP3**
originál	**RP0**	873	71	0	0
	RP1	290	533	24	0
	RP2	0	27	92	1
	RP3	0	0	1	0

Zdroj: vlastné spracovanie

Ak by sme spočítali všetky hodnoty, tak získame 1912 vzorov. To je o 88 menej ako sa nachádza v testovacej množine. Z výsledku a aj z poznámok v implementácii [39] je evidentné, že pre neznáme kombinácie hodnôt ID3 vyhodnotí vzor ako neznámu. Táto vlastnosť je evidentná pri hlbšom zamyslení sa ako funguje ID3. Vďaka tomuto javu neklasifikujeme neznáme vzory. Tento strom má tiež tendenciu preučiť sa alebo podučiť sa, čo si môžeme všimnúť v tomto výsledku, keď dosahuje 74,9 % presnosť podľa (3.3.6).

Hodnoty predikovaných tried boli spravidla podhodnocované z hľadiska zamýšľaného rizika. 290 + 27 + 1 = 318 vzorov sa dostalo do nižšej rizikovej skupiny v porovnaní s 71 + 24 + 1 = 96 vzormi, ktoré boli nadhodnotené. Ak sme averzní voči riziku budeme preferovať nadhodnotené riziká namiesto podhodnotených.

Vytvorený rozhodovací strom pre ID3 nebude zobrazený nakoľko sa jedná o veľmi veľký strom, ale je možné jeho náhľad vygenerovať zo skriptov priložených k tejto práci.

4.1.2 Aplikácia metódy CART

Dáta pre rozhodovací strom CART nebolo nutné upravovať nakoľko tento typ stromu dokáže rozpoznať veľké množstvo variability v dátach a nájsť v nich vzťah. Implementácia stromu CART interne delí dáta a využíva jednu časť na trénovanie a druhú na validáciu stromu pre prípad orezania.

Tabuľka 4.1.3: Výsledky predikcie neorezaného stromu CART pre klasifikáciu poistencov do rizikových skupín

		predikcia			
		RP0	**RP1**	**RP2**	**RP3**
originál	**RP0**	870	95	0	0
	RP1	260	629	1	0
	RP2	0	87	54	1
	RP3	0	1	3	0

Zdroj: vlastné spracovanie

Výsledné hodnoty obsahujú všetky vzory použité pre testovanie v originálnom počte 2000. Výsledná presnosť stromu je 77,65 %, čo je relatívne dobrý výsledok vzhľadom na povahu dát a distribúciu chyby ovplyvňujúcu hraničné hodnoty v originálnych dátach.

Jednotlivý strom sa dá orezať a získame tak zovšeobecnený strom, ktorý môže byť zaujímavejší. Pre potreby tejto práce si vystačíme s celkovou presnosťou jednotlivých orezaných stromov. Jednotlivé stromy si zobrazíme s počtom vnútorných uzloch pre predstavu o ich zložitosti.

Tabuľka 4.1.4: Výsledky presnosti orezaných stromov vytvorených metódou CART

Počet vnútorných uzlov	**Presnosť**
0	48,25 %
1	63,40 %
2	72,10 %
3	74,75 %
8	76,96 %
10	77,65 %

Zdroj: vlastné spracovanie

4.1.3 Aplikácia neurónových sietí

Vytvorenie neurónovej siete so 4 výstupnými neurónmi alebo len jedným pre výsledné riziko sa nepodarilo v dostatočne kvalitnej podobe. Tento jav je možné prisúdiť tomu, že dané dáta nemajú vnútornú koreláciu (sú to len sčítané pravdepodobnosti) a model v jednotlivých neurónoch má problém nájsť globálne minimum a zasekne sa v lokálnom minime bez ohľadu na nastavenie hodnoty miery učenia [27]. V niektorých prípadoch našiel

minimum, ktoré vyhovovalo podmienkam pre ukočenie trénovania, ale klasifikácia bola buď podľa väčšinovej triedy alebo s presnosťou okolo 50 %., čo je tiež nežiadúci výsledok.

4.1.4 Zhodnotenie výsledných modelov

ROC analýza je vhodná pre posudzovanie binárnych klasifikačných problémov, čo nie je možné v tomto prípade, preto využijeme možnosť vyhodnotenia na základe jednoduchej presnosti. Nakoľko pre vygenerované dáta je možné posúdiť len jeden strom ID3 s presnosťou 74,9% a šesť stromov vytvorených metódou CART.

Pri jednotlivých výsledkoch je nutné si uvedomiť, že nie každý strom klasifikuje pre všetky skupiny dát. Napríklad len neorezaný strom CART s 10 vnútornými uzlami dokáže klasifikovať skupinu RP3. Už rozhodovací strom CART s 2 vnútornými uzlami dokáže rozoznať len RP0 a RP1, preto je nutné venovať takýmto situáciám pozornosť. Rozhodovací strom s 3 vnútornými uzlami dokáže rozoznať 3 skupiny s vysokou presnosťou 74,75%, ktorá je veľmi blízka presnosti metódy ID3. Voľba výsledného modelu pre tento problém by bol rozhodovací strom typu CART s 3 vnútornými uzlami.

Je treba si uvedomiť, že jednotlivé dáta sú generované a výsledky, tak budú odpovedať ich povahe. Ak by pri generovaní dát bola odstránená chyba, tak by modely dosahovali väčšiu presnosť (ale nie 100 %). Pri ďalšej množine dát budeme pracovať s reálnymi dátami a uvidíme tak väčšiu variabilitu v riešení problémov.

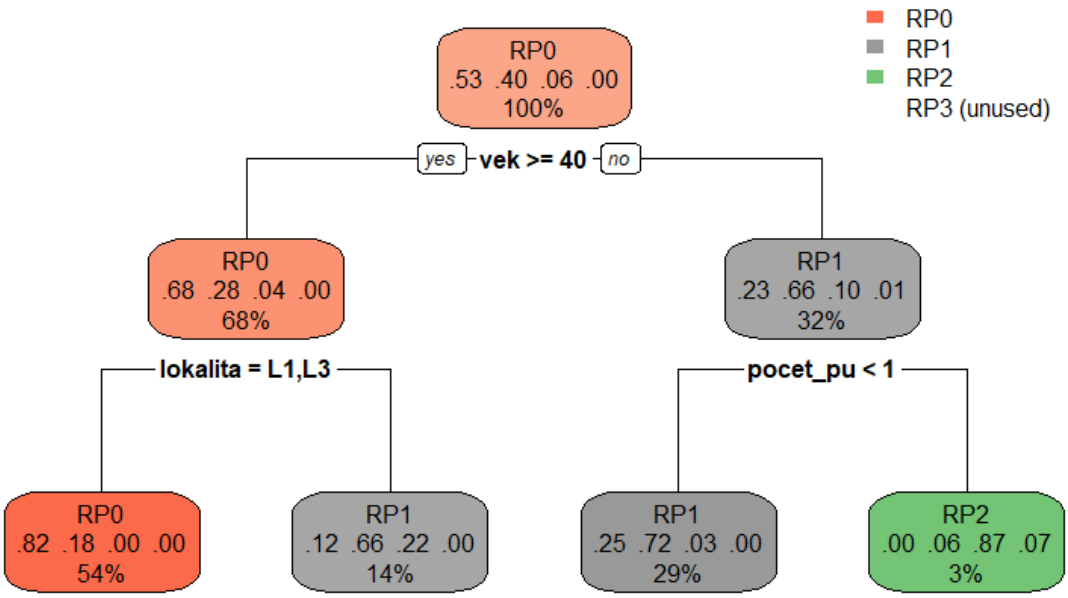
V praxi vieme použiť ešte jeden relevantný prístup opísaný aj pri internom fungovaní CART a to je ohodnotenie chyby klasifikácie pre každú zlú kombináciu predikcie. Pre každý prípad, kedy by bola poistná skupina nadhodnotená by bola ponechaná hodnota 1 nakoľko by potencionálny poistenec prešiel ku konkurencii. Ale pre podhodnotenie by bola zvolená 1,5 nakoľko poisťovňa riskuje, že jej nevystačia technické rezervy na poistné plnenia. Následne by stačilo tieto hodnoty sčítať pre vyhodnotenie. Metóda CART dokáže dokonca s takouto penalizáciou interne pracovať pokiaľ je to nutné. Neurónové siete dokážu na takúto situáciu reagovať vytvorením funkciou chyby reprezentujúcou túto požiadavkou.

Pri interpretácií najlepšieho modelu na obrázku 4.1.1 sú skupiny poistencov rozdelené hneď na začiatku na základe veku, to jasne symbolizuje, že ľudia s vekom nižším ako 40 rokov sú rizikovejšia skupina. Ak by sme pokračovali dátami s vekom nižším ako 40 rokov, tak rizikovejšie správanie v minulosti prispieva k vyššiemu poistnému. Dokonca zelený uzol s *RP2* zastrešuje aj všetky prípady *RP3*, čo indikuje že *RP3* zdieľa podobné znaky s *RP2*. Takúto znalosť možno využiť pri snahe odhaliť anomálie v dátach pre minimalizáciu budúcich poistných strát.

Namiesto toho môžeme pozrieť na dáta podľa listov stromu a rozdeliť testovacie dáta (a tým dáta poistencov) na 4 skupiny zákazníkov:

- poistenec s vekom 40 a viac bývajúci v lokalite L1 a L3,
- poistenec s vekom 40 a viac bývajúci v lokalite L2,
- poistenec s vekom menej ako 40 bez predchádzajúcich poistných udalostí,
- poistenec s vekom menej ako 40 s predchádzajúcou poistnou udalosťou.

Aktuár tak vidí, že jeho zákaznícke portfólio je dominované 4 skupinami zákazníkov. Pokiaľ by sa jednalo len o neznáme dáta, s ktorými pracuje prvýkrát, tak by aktuár hneď intuitívne vedel, akí zákazníci sa nachádzajú v portfóliu. Túto informáciu môže použiť aj pre lepšiu tvorbu opcií. Použiť ju môžeme v marketingu pre lepšie pochopenie zákazníckej bázy spoločnosti a tvorbu potrebných stratégií pre oslovenie potencionálnych zákazníkov. Cieliť tak môžeme na také skupiny, o ktorých vieme, že prispievajú viac na technické rezervy ako z nich môžu čerpať. Ťažiť môžeme z toho, že tento produkt je už zabehnutý a má reálnu trhovú pozíciu.



Obrázok 4.1.1: Vybraný rozhodovací strom CART s 3 vnútornými uzlami

Zdroj: výstup z *rpart.plot*

4.2 Klasifikovanie dát žiadateľov o úver

Druhá množina dát bude pochádzať z [15]. Konkrétne sa bude jednať o dáta pre pre určenie kreditného rizika. Nájsť ich môžeme v balíčku *CASdatasets* pod názvom „credit“ [6]. Dáta pozostávajú z 21 premenných a 1000 vzorov, ktoré rozdelíme v pomere 80:20 ako trénovacie a testovacie. Dáta predstavujú profily žiadateľov o úver a vhodnosť kandidáta na schválenie úveru. Výsledkom týchto dát bude na základe nezávislých premenných zamietnutie alebo schválenie úveru. Pri práci s dátami bude cieľom získať model s najlepšou mierou zamietania žiadostí.

Tabuľka 4.2.1: Charakteristika premenných pre dáta žiadateľov o úver

Premenná	Typ premennej	Variabilita hodnoty
Stav kontrolného účtu	kategorická	4 hodnoty
Dĺžka úveru	numerická	dĺžka trvania v mesiacoch
Kreditná história	kategorická	5 hodnôt
Účel	kategorická	11 hodnôt
Veľkosť úveru	numerická	veľkosť úveru
Úspory	kategorická	5 hodnôt
Zamestnanie	kategorická	5 hodnôt
Pomer splátky	kategorická	4 hodnoty
Rodinný stav	kategorická	5 hodnôt
Spoluručiteľ	kategorická	3 hodnoty
Rezident od	kategorická	5 hodnôt
Veľkosť majetku	kategorická	4 hodnoty
Vek	numerická	roky
Iné splátky	kategorická	3 hodnoty
Obydlie	kategorická	3 hodnoty
Počet existujúcich úverov	kategorická	4 hodnoty
Práca	kategorická	4 hodnoty
Vyživuje osôb	kategorická	2 hodnoty
Telefónne číslo	kategorická	2 hodnoty
Cudzinec	kategorická	2 hodnoty
Vyhodnotenie žiadosti	kategorická	2 hodnoty

Zdroj: [6]

Tieto dáta spolu vytvárajú v prípade neurónovej siete 72 premenných. Premenná „vyhodnotenie žiadosti“ je ponechaná v binárnom stave (0 pre schválenie a 1 pre zamietnutie) a sledujeme v nej koľko žiadostí je úspešne zamietnutých.

Pre tieto dáta vieme vypočítať všetky indikátory ROC analýzy, nakoľko sa jedná o binárny klasifikačný problém. Cieľom je byť averzný k riziku, preto budeme chcieť modely s najvyššou možnou hodnotou *tp rate* podľa (3.3.2).

4.2.1 Aplikácia metódy ID3

Pre tvorbu rozhodovacieho strom ID3 bolo nutné znovu transformovať všetky numerické atribúty na kategoriálne. Transformované boli premenné trvanie úveru na intervaly po 6 mesiacoch, veľkosť úveru v intervaloch po 500, vek v intervaloch po 5. Tento hendikep môže spôsobovať problémy pri tvorbe stromov vo všeobecnosti, nakoľko tak schovávame veľké množstvo informácií pred interným algoritmom stromu. Pre výsledný strom je potom nutné všetky ďalšie vstupy transformovať rovnako.

Tabuľka 4.2.2: Výsledok predikcie neorezaného stromu ID3

		predikcia	
		N	**P**
originál	**N**	73	27
	P	27	21

Zdroj: vlastné spracovanie

Znovu si môžeme všimnúť chýbajúce vzory z 200 testovacích vzorov a prítomných je len 154, čo znovu poukazuje na povahu algoritmu ID3 a jeho neschopnosť pri práci s neznámymi kombináciami vzorov.

Z klasifikovaných vzorov môžeme vypočítať všetky vlastnosti ROC analýzy a získame tak lepší pohľad na povahu výsledkov:

Tabuľka 4.2.3: Vlastnosti ROC analýzy pre strom metódy ID3

fp rate	tp rate	precision	sensitivity	specificity	acccuracy
0,2547	0,4375	0,4375	0,4375	0,7453	0,6494

Zdroj: vlastné spracovanie

Grafický výstup stromu je veľmi veľký pre jeho čitateľnosť, ale je možné ho vygenerovať z priložených skriptov.

Vzhľadom na neschopnosť klasifikovať všetky neznáme vzory a neschopnosť určiť zamietnuté žiadosti (dokonca podpriemerne; jedná sa o indikátor *tp rate*) pre žiadateľov o úver je tento model nepoužiteľný v praxi a nebudeme sa mu venovať ďalej.

4.2.2 Aplikácia metódy CART

V porovnaní so stromom ID3 metóda CART dokáže klasifikovať všetky prípady a variácie v dátach. V prípade tohto stromu budeme posudzovať neskôr presnosť na základe testovacích dátach v porovnaní s internými dátami zvolenými metódou CART.

Tabuľka: 4.2.4: Výsledok predikcie neorezaného stromu CART

		predikcia	
		N	**P**
originál	**N**	123	17
	P	29	31

Zdroj: vlastné spracovanie

V porovnaní s metódou ID3 boli klasifikované všetky testovacie vzory. Z nich môžeme vypočítať jednotlivé indikátory pre posúdenie kvality modelu.

Tabuľka 4.2.5: Vlastnosti ROC analýzy pre neorezaný strom metódy CART

fp rate	tp rate	precision	sensitivity	specificity	acccuracy
0,1214	0,5167	0,6458	0,5167	0,8786	0,77

Zdroj: vlastné spracovanie

Pokiaľ je cieľom modelu vyhľadávať a úspešne zamietať žiadateľov o úver, tak dosahuje model priemerné výsledky (*tp rate*) a jeho voľba nemusí byť vhodná. Pozitívnym aspektom modelu je, že dokáže rozoznať relatívne dobre žiadosti vhodné na schválenie (nízka hodnota *fp rate*).

Strom vygenerovaný metódou CART, ktorý odhaľuje najlepšie zamietnuté žiadosti by bol prakticky nečitateľný, ale je možné si ho vygenerovať z priložených skriptov.

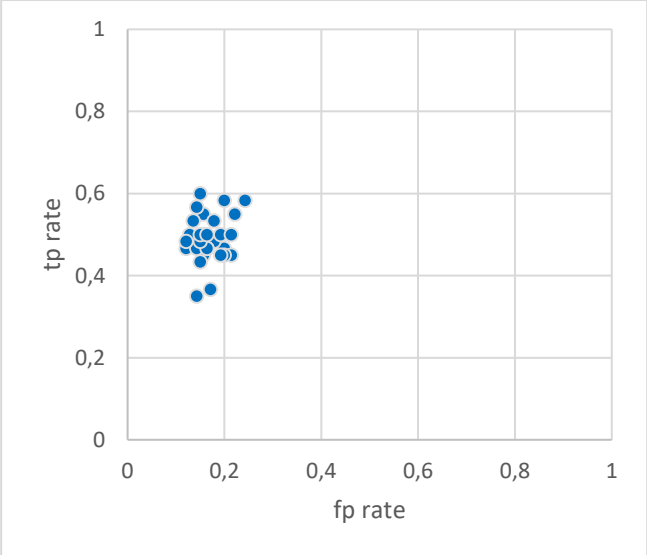
Tabuľka 4.2.6: Všetky orezané rozhodovacie stromy s vybranými vlastnosťami ROC analýzy zoradené podľa počtu vnútorných uzlov

Vn. Uzly	TN	FP	FN	TP	fp rate	tp rate	accuracy
0	140	0	60	0	0	0	0,7
2	132	8	47	13	0,0571	0,2167	0,725
3	129	11	37	23	0,0786	0,3833	0,76
4	128	12	36	24	0,0857	0,4	0,76
5	132	8	40	20	0,0571	0,3333	0,76
13	123	17	29	31	0,1214	0,5167	0,77

Zdroj: vlastné spracovanie

4.2.3 Aplikácia neurónových sietí

Pri použití neurónovej siete bola zvolená jedna skrytá vrstva a výstupná vrstva o jednom neuróne predstavujúcom pravdepodobnosť zamietnutia úveru. Pri tvorbe neurónovej siete boli vytvorené rôzne neurónové siete s rôznym počtom neurónov v skrytej vrstve od 2 až po 30 vrátane. Miera učenia bola 0,01. Výsledky si zobrazíme v podobe základného ROC grafu namiesto tabuľky všetkých sietí nakoľko sa jedná o veľké množstvo sietí a nie všetky siete sú relevantné. Bližšie si pozrieme len údaje ROC analýzy o najlepšej sieti s 21 neurónmi v skrytej vrstve.



Graf 4.2.1: Základný ROC graf pre všetky neurónové siete

Zdroj: vlastné spracovanie

Tabuľka 4.2.7: Výsledok predikcie neurónovej siete s 21 neurónmi v skrytej vsrtve

		predikcia	
		N	**P**
originál	**N**	119	21
	P	24	36

Zdroj: vlastné spracovanie

Tabuľka 4.2.8: Vlastnosti ROC analýzy pre neurónovú sieť s 21 neurónmi v skrytej vrstve

fp rate	tp rate	precision	sensitivity	specificity	acccuracy
0,15	0,6	0,6316	0,6	0,85	0,775

Zdroj: vlastné spracovanie

Z vlastností ROC analýzy možno vidieť, že model je relatívne presný, ale ukazovateľ *sensitivity*/*tp rate* dosahuje len 60 % a preto nemusí byť v praxi vhodná voľba. Avšak dosahuje lepšie výsledky ako neorezaný rozhodovací strom CART.

Lepšie výsledky by sme mohli získať aj použitím väčšieho množstva dát. Neurónové siete sú vo všeobecnosti najlepšie trénované na veľkom množstve dát, ale pre hľadanie vhodného modelu je najlepšie využiť nejakú formu akcelerátoru.

Pri práci s dátami žiadateľov o úver vďaka nepozornosti pri kontrole dát boli použité v poskytnutej forme a neboli zohľadnené prípady, keď niektoré kategorické dáta boli použité ako numerické. Konkrétne sa jednalo o *pomer splátky*, *obydlie*, *existujúce úvery* a *vyživované osoby*. Dáta v tomto stave získali marginálne lepšie modely, z ktorých jeden vyčnieva a stojí za zmienku, aj z pohľadu ako má predspracovanie dát dopad na kvalitu neurónových sietí.

Tabuľka 4.2.9: Výsledok zle predspracovaných dát pre neurónové siete

Neuróny	TN	FP	FN	TP	fp rate	tp rate	accuracy
22	56	14	9	21	0,2	0,7	0,77

Zdroj: vlastné spracovanie

V prípade tejto siete stojí za pozornosť vysoká hodnota *tp rate*, ktorá ukazuje zvýšenú presnosť pri zamietaní žiadostí o úver v porovnaní s predchádzajúcimi sieťami. Nakoľko neboli dostupné podrobné informácie o dátach a ich vnútornej povahe je tento model ukážkou toho ako ich potencionálna znalosť môže viesť k lepším výsledkom. Vo

všeobecnosti dosahovali tieto dáta lepšie výsledky zamietnutí v porovnaní s riadnou reprezentáciou podľa definície. Neurónové siete sú veľmi citlivé na predspracovanie dát a je potrebné mať dostupných, čo najviac informácií o dostupných dátach.

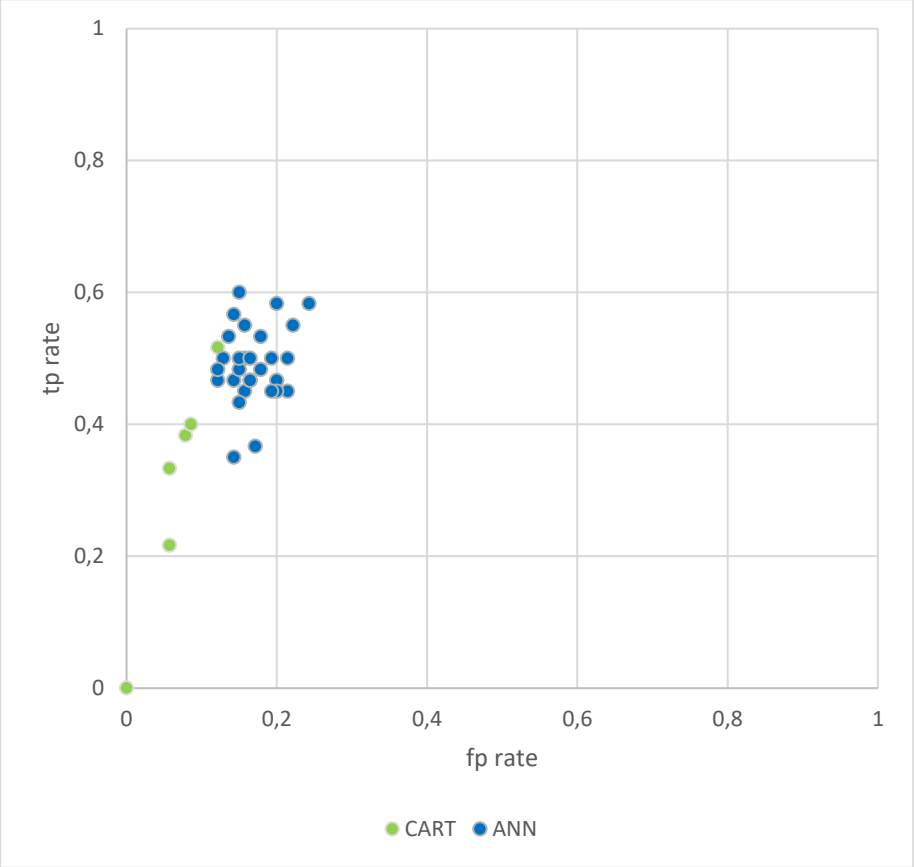
Tabuľka 4.2.10: Všetky neurónové siete s počtom neurónom v skrytej vrstve v intervale od 2 do 30

Neuróny	TN	FP	FN	TP	fp rate	tp rate	accuracy
2	106	34	25	35	0,2429	0,5833	0,7050
3	122	18	30	30	0,1286	0,5000	0,7600
4	112	28	32	28	0,2000	0,4667	0,7000
5	118	22	33	27	0,1571	0,4500	0,7250
6	110	30	33	27	0,2143	0,4500	0,6850
7	115	25	31	29	0,1786	0,4833	0,7200
8	112	28	33	27	0,2000	0,4500	0,6950
9	112	28	25	35	0,2000	0,5833	0,7350
10	119	21	34	26	0,1500	0,4333	0,7250
11	109	31	27	33	0,2214	0,5500	0,7100
12	113	27	30	30	0,1929	0,5000	0,7150
13	120	20	39	21	0,1429	0,3500	0,7050
14	123	17	32	28	0,1214	0,4667	0,7550
15	116	24	38	22	0,1714	0,3667	0,6900
16	117	23	32	28	0,1643	0,4667	0,7250
17	120	20	32	28	0,1429	0,4667	0,7400
18	118	22	27	33	0,1571	0,5500	0,7550
19	110	30	30	30	0,2143	0,5000	0,7000
20	117	23	32	28	0,1643	0,4667	0,7250
21	119	21	24	36	0,1500	0,6000	0,7750
22	118	22	30	30	0,1571	0,5000	0,7400
23	113	27	33	27	0,1929	0,4500	0,7000
24	115	25	28	32	0,1786	0,5333	0,7350
25	119	21	31	29	0,1500	0,4833	0,7400
26	121	19	28	32	0,1357	0,5333	0,7650
27	123	17	31	29	0,1214	0,4833	0,7600
28	120	20	26	34	0,1429	0,5667	0,7700
29	119	21	30	30	0,1500	0,5000	0,7450
30	117	23	30	30	0,1643	0,5000	0,7350

Zdroj: vlastné spracovanie

4.2.4 *Komparácia výsledných modelov*

Pre vyhodnotenie budú použité všetky modely získané metódou CART a neurónové siete, ktoré boli natrénované v tabuľke 4.2.10. Na dátach o zamietnutí žiadostí o úver bude možné vidieť ROC analýzu na porovnanie modelov medzi sebou, nakoľko sa jedná o binárny klasifikačný problém.



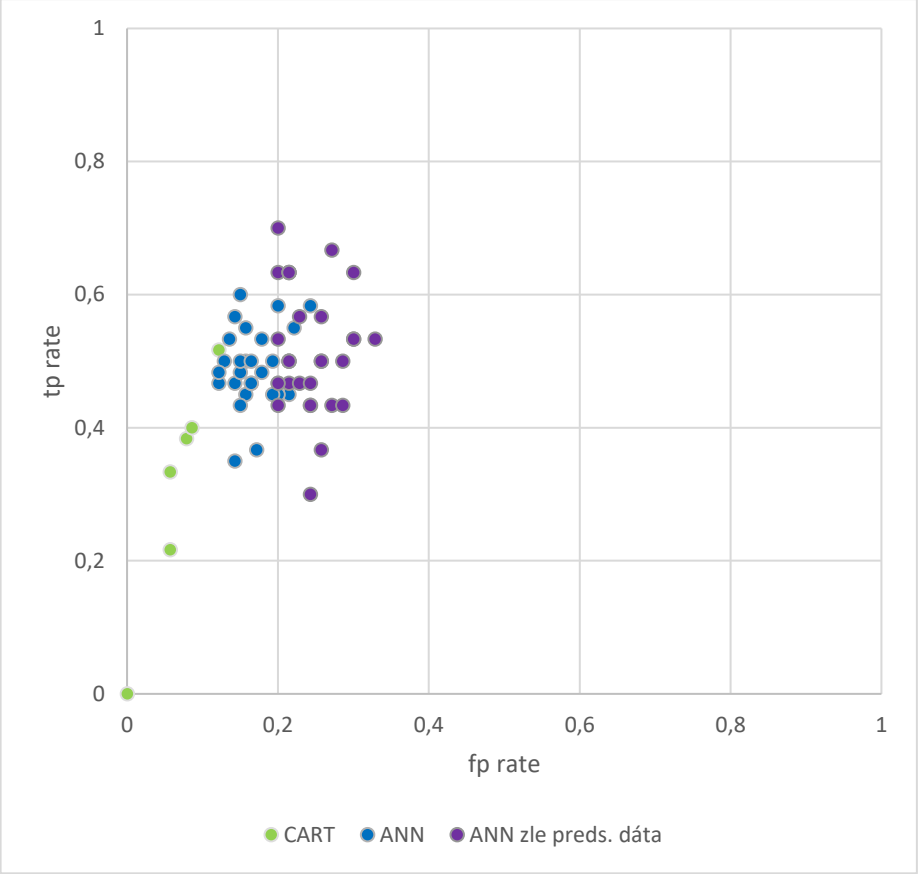
Graf 4.2.2: Základný ROC graf pre modely CART a neurónové siete

Zdroj: vlastné spracovanie

Ako najlepší výsledok si môžeme všimnúť neurónovú sieť s 21 neurónmi v skrytej vrstve, pričom najlepší strom získaný metódou CART bol prekonaný ôsmymi neurónovými sieťami v *tp rate*.

Pri stromoch metódy CART stojí za zmienku klesajúca schopnosť klasifikovať zamietnutia jednotlivých žiadostí o úver. Z tohto môžeme usúdiť, že jednotlivé dáta majú veľmi silnú interakciu a neurónové siete budú lepšia voľba. Pri použití iných metód ako lineárnej regresie by bolo zaujímavé preskúmať kombinácie jednotlivých nezávislých znakov ako nezávislých premenných. Vidíme tak, že použitie rôznych techník strojového učenia môže viesť k intuitívnym informáciám ohľadom dát, s ktorými pracujeme.

Pre zaujímavosť ešte uvedieme ROC graf doplnený o neurónové siete získané nad zle predspracovanými dátami. Tieto výsledky je možné získať pri použití priložených skriptov na CD.



Graf 4.2.3: Základný ROC graf doplnený o neurónové siete s zle predspracovanými dátami

Zdroj: vlastné spracovanie

Pri dostatku informácií o použitých dátach by bolo možné získať neurónové siete s vysokou presnosťou. Ak by sme predpokladali, že prístup zvolený pre predspracované dáta bol korektný, tak by sme pre tento problém zvolili neurónovú sieť s 22 neurónmi v skrytej vrstve. Pripomeňme si, že cieľom týchto modelov je zamietať žiadosti s čo najväčšou presnosťou.

4.2.5 Zhodnotenie výsledných modelov

Kreditné riziko predstavuje riziko pre veriteľa, že daný úver nebude splatený a stratí tak požičané zdroje. Je preto na mieste byť obozretný pri interpretácií výsledkov. Pri analýze dát kreditného rizika žiadosti o úver je problém odhaliť žiadosti, ktoré by mali byť zamietnuté a je na mieste zvážiť, či je model vhodný na použitie. Ideálny postup by bol získať ďalšie dáta. Ak by to nebolo možné vyskúšať iné techniky strojového učenia pre nájdenie vhodné vzťahu. Vhodná voľba by boli *klástrovacie algoritmy a Bayesovské klasifikovanie* pre nájdenie zhlukov alebo kombinácií hodnôt, ktoré ovplyvňujú riziko defaultu. Pre porovnanie by bolo zaujímavé zvoliť typicky používanú *logistickú regresiu* ako klasický prostriedok pre analýzu kreditného rizika. Z nej by sme vedeli, ktoré premenné prispievajú najviac k defaultu [23].

Interpretácia súčasných modelov CART a neurónových sietí nám ukazuje významnú interakciu v dátach a bolo by vhodné preskúmať daný problém najprv pohľadom na neorezaný strom metódy CART. Videli by sme na ňom, že vytvorený strom má pomerne veľa vnútorných uzlov kým dokáže identifikovať zamietnuté žiadosti o úver. Z toho môžeme usúdiť, že veľa premenných prispieva k zamietnutiu žiadosti o úver. Nie je tak možné spraviť vierohodný záver zo stromu CART.

Pri pohľade na výsledky neurónových sietí je možné vidieť výrazne zlepšenie klasifikovania zamietnutých žiadostí, ale stále existuje významné množstvo žiadostí, ktoré uniknú pozornosti a sú schválené (40 %). Ak by sme využili model najlepšej neurónovej siete, tak by sme potrebovali kompenzovať potencionálne straty očakávané z požičiavania týmto 40 % a podľa toho prispôsobiť podmienky poskytovaného úveru.

5 Diskusia

Pri riešení problematiky strojového učenia sme sa zamerali na použitie rozhodovacích stromov a neurónových sietí s dopredným šírením, ktoré sú jedny z mnohých metód zameraných na hľadanie vzťahu medzi závislou premennou a nezávislými premennými. Pri pohľade na riešenie problémov môžeme vidieť odlišný prístup k objasňovaniu vzťahov medzi jednotlivými metódami.

Rozhodovacie stromy hľadajú v danej podmnožine najvhodnejší atribút, ktorý má najväčší podiel na rozdelení závislej premennej. Ich nedostatok je možné vidieť na prvej množine dát s klasifikáciu do rizikových skupín. Povaha týchto dát dokáže nájsť najvýznamnejšie premenné, ale nedokáže objasniť dokonale dáta, ak prínos každej premennej je malý. Musíme si uvedomiť, že rozhodovacie stromy delia dáta vzhľadom na výslednú triedu/kategóriu. Ak je šum príliš veľký a informačný zisk malý, tak nie je schopný spraviť kvalitné delenia. Napriek tomu dokáže poodhaliť významné premenné nachádzajúce sa v jednotlivých podmnožinách dát. Tieto informácie je možné použiť pri delení dát do špecifických skupín alebo ich označovaním pre presnejšiu klasifikáciu v ďalších dátach. Pri druhej množine dát s žiadosťami o úver je možné vidieť, že určité typy problémov vyžadujú iný pohľad na problém. Ak interakcia medzi jednotlivými dátami je príliš veľká, je vidieť nedostatky rozhodovacích stromov. Tie sa snažia vždy nájsť najlepšie riešenie len na lokálnej úrovni len pomocou jedného atribútu a ľubovoľnej kombinácie (alebo deliacej hodnoty) jednotlivých hodnôt. Pokiaľ dáta preukazujú rozdielne vlastnosti na rozdielnych množinách, tak je táto metóda jedna z najlepších volieb.

Neurónové siete s dopredným šírením ukazujú svoje nedostatky pri riešení problémov, ak pracujú s dátami bez hlbšej interakcie. Jednotlivé váhy nie sú potom schopné skonvergovať k suboptimálnym hodnotám a zasekávajú sa v lokálnych minimách. Tento problém je všeobecne známy problém neurónových sietí a je nutné byť na neho pripravený. Nesporná výhoda neurónových je vidieť pri riešení problémov s výraznou vnútornou interakciou. Pri dátach o žiadostiach o úver vidíme výrazné zlepšenie presnosti v porovnaní s rozhodovacími stromami. Dokáže častokrát poukázať na vnútornú interakciu, ktorá je často neviditeľná ľudskému pozorovaniu. Problémom neurónových sietí sú ich nároky na množstvo dát potrebných na ich trénovanie. Pri malom množstve dát je veľmi komplikované vytvoriť neurónovú sieť s dostatočnou presnosťou. Pri veľkých objemoch dát je možné vytvoriť neurónové siete s nevídanou presnosťou aj v porovnaní s najlepšie optimalizovanými modelmi z inej rodiny metód strojového učenia. Najväčší nedostatok

neurónových sietí je evidentný len z ich vnútornej reprezentácie a keby sme otvorili ľubovoľnú sieť, tak je nemožné pre užívateľa pochopiť jej interné fungovanie. Z praktického pohľadu je neurónová sieť „čierna skrinka“, ktorá je definovaná viacrozmerným priestorom

Aktuár môže rozhodovacie stromy využiť pre hľadanie kombinácií premenných najlepšie prospievajúcich k určitému výsledku a na základe toho stanoviť svoj ďalší postup, ak rozhodovací strom nie je vhodná voľba. Výborne sa osvedčia aj pri pohľade na neznáme dáta, ak chce získať intuitívnu predstavu aké podmnožiny dát sa v nich nachádzajú. Ak sú neurónové siete vhodná voľba a je dostupné veľké množstvo dát, tak dokážu ušetriť veľké množstvo energie pri hľadaní vhodného modelu. Ich využitie si tiež nájde miesto v benchmarkingu a hľadaní anomálií za predpokladu, že neurónové siete by boli trénované priamo na testovanie pre výskyt anomálií. Vo všeobecnosti sú použiteľné pri širokú paletu problémov a sú limitované kreativitou, množstvom dát a niekedy ich praktickosťou.

Pre ďalšie pokračovanie tejto práce by bolo zaujímavé zahrnúť ďalšie metódy strojového učenia. Metóda C4.5 je priamym pokračovateľom metódy ID3 a je to relevantná voľba. Tejto práci je blízky aj *náhodný les* (*random forest*), ktorý dokáže agregovať rozhodovanie niekoľkých rozhodovacích stromov. Ten dokáže veľmi presne identifikovať aj relatívne komplikované problémy, pre ktoré sa normálne používajú neurónové siete. *Klástrovacie algoritmy* by boli výborná voľba pre problém s odhalením aké zhluky sú najtypickejšie pre zamietnuté žiadosti. Podobne by dokázalo odhaliť interakciu medzi dátami *Bayesovské klasifikovanie,* z ktorého dokážeme vyčítať za akých podmienok sú najčastejšie zamietnuté žiadosti.

Záver

Diplomová práca prezentuje širšiu problematiku spracovávanie narastajúceho množstva dát spoločností (s dôrazom na poisťovne). Tento trend dostal označenie Big Data a považuje sa za kľúčové vedieť z týchto dát získať ekonomický úžitok. V prípade poisťovní je výhodou lepšia schopnosť pracovať s týmito dátami a zvýšiť konkurencieschopnosť už aj tak na saturovaných trhoch.

Získané dáta je možné spracovávať pomocou metód strojového učenia, z ktorých si musí užívateľ zvoliť požadovaný cieľ a z toho usúdiť vhodný výber metód z jednotlivých skupín práce so strojovým učením. Je dôležité vedieť, čo je cieľom práce s dátami, nakoľko sa od toho odvíja voľba metódy strojového učenia a možného ďalšieho postupu s výsledkami a ich interpretácií.

Širšou aplikáciou poznatkov o dátach a spôsoboch ich spracovania strojovým učením môže získať poisťovňa benefity v jej rôznych oddeleniach. Výhody pre marketingové oddelenie môžu viesť k lepšiemu cieleniu a pochopeniu potencionálnych zákazníkov. Lepšie pochopenie správania existujúcich vedie k lepším vzťahom medzi poistencom a poisťovňou, čo je veľmi výhodné pre obe strany. Poisťovňa vie ponúknuť poistencovi lepšie produkty a včasne reagovať na zmeny v jeho správaní. Z takéhoto vzťahu profitujú obe strany, čo je veľmi dôležité pri súčasných trendoch s dôrazom na spoločensky zodpovedné podnikanie. Upisovanie a vývoj poistných produktov je oblasť, ktorá je kľúčovou činnosťou poisťovne a vedie k lepšej konkurencieschopnosti a tiež k nižším nákladom pri zvýšenej presnosti odhadov.

Aktuári môžu tieto poznatky využiť pri lepšom pochopení existujúcich dát a benchmarking iných modelov. Výsledné informácie môžu použiť pre lepšie rozdelenie portfólia poistencov na hypotetické subportfólia a vytvoriť lepšie opcie. Pri detekcií určitých skupín poistencov, ktoré zdieľajú podobné znaky môžu využiť dodatočné bonusy alebo malusy, ak to zmluva umožňuje. Nastavenie finančných podmienok poistných a iných zmlúv je možné lepšie, ak vidíme určité komplikácie v stanovovaní vierohodných rizikových hodnôt s akými sme sa stretli pri dátach o žiadostiach o úver.

Prezentované boli poznatky vnútorného fungovania modelov ID3, CART a neurónových sietí. Z ich vnútorného fungovania sú implikované aj ich potencionálne nedostatky. Výsledné poznatky je potom pri hlbšom pochopení možné pretaviť do praxe a získať z nich okamžitý úžitok pri interpretácií dát. Špecifická ROC analýza je zaujímavá pri posudzovaní klasifikačných problémov a poukazuje na nedostatky pri vyhodnocovaní

dát len na základe presnosti. Detailnejší pohľad umožňuje lepšie pochopenie správania sa modelu na daných dátach. ROC analýzou je možné modifikovať pre ľubovoľný počet, je tak možné individuálne sledovať presnosť pre každú kategóriu. Je nevyhnutné mať, ale dopredu informácie o požadovaných cieľoch pri vyhodnocovaní.

Z výsledkov prác je možné odsledovať proces tvorby a posudzovania jednotlivých výsledných modelov a rozhodovania sa medzi nimi. Výber jednotlivých modelov je sprevádzaný určitými kompromismi ako voľba medzi všeobecnejšími modelmi v prípade rozhodovacích stromov za cenu nižšej presnosti. Podobne je nutné venovať pozornosť spracovaniu dát, ako sme mohli vidieť pri neurónových sieťach. Tvorba vhodných modelov je dlhodobá činnosť od spracovania dát po optimalizáciu jednotlivých parametrov metód strojového učenia.

Zoznam použitej literatúry

[1] AMERICAN ACADEMY OF ACTUARIES. *Big Data and the Role of the Actuary*. [online]. Washington : American academy of Actuaries, 6. 2018. Dostupné na: https://www.actuary.org/sites/default/files/files/publications/BigDataAndTheRoleOfTheActuary.pdf

[2] BREIMAN, L. a kol. *Classification and regression trees*. s.l. : Chapman and Hall/CRC, 1984. ISBN 0-412-04841-8.

[3] BYFORD, S. *Google's AlphaGo AI beats Lee Se-dol again to win Go series 4-1*. s.l. : The Verge. 15.3. 2016. Dostupné na : https://www.theverge.com/2016/3/15/11213518/alphago-deepmind-go-match-5-result

[4] ČECHMÁNEK, J. *Rozhodovací stromy pro klasifikaci dat*. [diplomová práca]. Praha : Univerzita Karlova v Praze, 2008.

[5] DASTIN, J. *Amazon scraps secret AI recruiting tool that showed bias against women*. [online]. New York : Reuters. 10.10. 2018. Dostupné na: https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G

[6] DUTANG, CH. *CASdatasets: Insurance datasets (Official website)*. [online] 28.5. 2016. [cit. 2019-03-26]. Dostupné na: http://cas.uqam.ca/

[7] EARNIX. *Machine Learning – Growing, Promising, Challenging*. [elektronický zdroj]. London : Earnix, 2017. Dostupné na: http://earnix.com/wp-content/uploads/2017/03/2017_Machine_learning_survey_report.pdf

[8] FAWCETT, T. *An introduction to ROC analysis*. Palo Alto : Institute for the Study of Learning and Expertise, 19.12. 2005. 14s. Dostupné na: https://people.inf.elte.hu/kiss/11dwhdm/roc.pdf

[9] FECENKO, J. Neživotné poistenie. Bratislava : Ekonóm, 2012. ISBN 978-80-225-3400-0.

[10] FRANKLIN, W. *Machine learning algorithm vs. Actuarial science…who will win?*. [online]. s.l. : Medium, 5.8. 2016. Dostupné na: https://towardsdatascience.com/machine-learning-algorithm-vs-actuarial-science-who-will-win-b203f31145ce

[11] FRITSCH, S. - GUENTHER, F. *neuralnet: Training of Neural Networks*. [online]. 7.2. 2019. Dostupné na: https://CRAN.R-project.org/package=neuralnet

[12] FUMO, D. *Types of Machine Learning Algorithms You Should Know*. [online]. s.l. : Medium, 15.6. 2017. Dostupné na: https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861

[13] GOLDBURD, M. - KHARE, A. - TEVET, D. *Generalized Linear Models for Insurance Rating*. [elektronický zdroj]. s.l. : Casualty Actuarial Society, 2016. [cit. 2019-03-22]. ISBN 978-0-9968897-3-5. Dostupné na: https://www.casact.org/pubs/monographs/papers/05-Goldburd-Khare-Tevet.pdf

[14] HORNIK, K. - BUCHTA, C. - ZEILEIS, A. *Open-Source Machine Learning: R Meets Weka*. [online]. 2009. Dostupné na: http://doi.org/10.1007/s00180-008-0119-7

[15] CHARPENTIER, A. *Computational Actuarial Science with R*. s.l. : Chapman and Hall/CRC, 2014. ISBN 978-1466592599.

[16] JAHODA, M. *Rozhodovací stromy*. [diplomová práca]. Praha : Univerzita Karlova v Praze, 2009.

[17] KVASNIČKA, J. a kol. *Úvod do teórie neurónových sietí.* s.l. : Iris, 1997. ISBN 978-80-88778-30-1.

[18] MACEK, K. *Umělé neuronóvé sítě a jejich použití v oblasti pojistných rizik*. [diplomová práca]. Praha : Univerzita Karlova v Praze, 2006.

[19] MARR, B. *How Big Data Is Changing Insurance Forever*. [online]. .s.l. : Forbes Media LLC, 16.12. 2015. Dostupné na: https://www.forbes.com/sites/bernardmarr/2015/12/16/how-big-data-is-changing-the-insurance-industry-forever/

[20] MCCLOSKEY, J. *Self-driving Uber car hits and kills cyclist as she crosses road*. [online]. London : Metro, 19.3. 2018. Dostupné na: https://metro.co.uk/2018/03/19/self-driving-uber-car-hits-kills-cyclist-crossed-road-7400145/

[21] MILLBORROW, S. *rpart.plot: Plot 'rpart' Models: An Enhanced Version of 'plot.rpart'*. [online]. 12.4. 2019. Dostupné na: https://cran.r-project.org/package=rpart.plot

[22] NIELSEN, M. *Neural Networks and Deep Learning*. [elektronický zdroj]. 2015. [cit. 2019-04-13]. Dostupné na: http://neuralnetworksanddeeplearning.com/

[23] PEUSSA, A. *Credit risk scorecard estimation by logistic regression*. Helsinki : University of Helsinki. 2016.

[24] PRADIER, P. - CHNEWIESS, A. *The evolution of insurance regulation in the EU since 2005*. s.l : Documents de travail du Centre d'Economie de la Sorbonne, 2016. ISSN 1955-611X.

[25] QUINLAN, J.R. *Induction of Decision Trees*. In : Machine Learning 1, 1986. Boston : Kluwer Academic Publishers.

[26] R CORE TEAM. R: *A language and environment for statistical computing*. [online]. Viedeň : R Foundation for Statistical Computing [cit. 2019-04-14]. Dostupné na: https://www.R-project.org/

[27] ROJAS, R. *Neural Networks - A Systematic Introduction*. Berlin, New York : Springer-Verlag, 1996.

[28] SARLE, S. W. *comp.ai.neural-nets FAQ, Part 2 of 7: Learning*. Cary : faqs.org, 11.10. 2002. Dostupné na: http://www.faqs.org/faqs/ai-faq/neural-nets/part2/

[29] SHALEV-SHWARTZ, S. - BEN-DAVID, S. *Understanding Machine Learning: From Theory to Algorithms*. New York : Cambridge University Press, 2014. ISBN 978-1-107-05713-5.

[30] THERNEAU, T. - ATKINSON, B. *rpart: Recursive Partitioning and Regression Trees*. [online]. 12.4. 2019. Dostupné na: https://CRAN.R-project.org/package=rpart

[31] TUSZYNSKI, J. *caTools: Tools: moving window statistics*, GIF, Base64, ROC AUC, etc. [online] 6.3. 2019. Dostupné na: https://CRAN.R-project.org/package=caTools

[32] WAAL-MONTGOMERY, M. *World's data volume to grow 40% per year & 50 times by 2020: Aureus*. [online]. Singapure : e27, 15.1. 2015. Dostupné na: https://e27.co/worlds-data-volume-to-grow-40-per-year-50-times-by-2020-aureus-20150115-2/

[33] WITTEN, IH. - FRANK, E. *Data Mining: Practical Machine Learning Tools and Techniques*. 2. vyd. San Francisco : Morgan Kaufmann, 2005. ISBN 978-0120884070.

[34] Zákon o zdravotnom poistení a o zmene a doplnení zákona č. 95/2002 Z. z. o poisťovníctve a o zmene a doplnení niektorých zákonov : účinnosť od 1.3.2019.

[35] Regulation (EU) 2016/679 : 2016 : General Data Protection Regulation.

[36] <https://www.investopedia.com/terms/a/actuary.asp> stav k 18.2.2019

[37] <https://www.onlia.sk/online-poistenie-vozidiel-vypocet-ceny> stav k 16.3.2019

[38] <https://www.sas.com/en_us/insights/big-data/what-is-big-data.html> stav k 22.2.2019.

[39] <http://weka.sourceforge.net/doc.packages/simpleEducationalLearningSchemes/weka/classifiers/trees/Id3.html> stav k 29.4.2019

Prílohy

Príloha č.1 – generovanie dát pre rizikovú skupinu/prirážku

```
n <- 2000
seed <- 1234 # 40 / 1234
dataname <- "test" # "train" / "test"

set.seed(seed)
znacka <- factor(sample(1:4, n, replace=TRUE), labels = c("A",
"B", "C", "D"))

set.seed(seed)
trieda <- factor(sample(1:3, n, replace = TRUE, prob = c(0.5, 0.4,
0.1)), labels = c("N", "S", "V"))

set.seed(seed)
lokalita <- factor(sample(1:3, n, replace = TRUE, prob = c(0.6,
0.2, 0.2)), labels = c("L1", "L2", "L3"))

set.seed(seed)
pocet_pu <- rpois(n, 0.1)

set.seed(seed)
a <- rchisq(n, 17)
vek <- 18 + (a - min(a)) / (max(a) - min(a)) * 82

set.seed(seed)
rodinne_ucely <- factor(sample(1:2, n, replace = TRUE, prob =
c(0.8, 0.2)), labels = c("R", "I"))

set.seed(seed)
max_hmotnost <- 1.1 + sapply(rchisq(n, 1.7), function(h) min(h,
2.4))

set.seed(seed)
vek_auta <- sapply(rchisq(n, 4), function(v) min(v, 15))

set.seed(seed)
typ_motora <- factor(sample(1:2, n, replace = TRUE, prob = c(0.4,
0.6)), labels = c("B", "D"))

set.seed(seed)
chyba <- rnorm(n, 0, 1 / 32)

f_znacka <- function(z) switch(as.character(z), A=0.02, B=0.05,
C=0.03, D=0.08)
f_trieda <- function(t) switch(as.character(t), N=0.02, S=0.04,
V=0.01)
f_lokalita <- function(l) switch(as.character(l), L1=0.05,
L2=0.02, L3=0.01)
f_rodinne_ucely <- function(u) switch(as.character(u), R=0.02,
I=0.1)
```

```
f_typ_motora <- function(m) switch(as.character(m), B=0.02,
D=0.05)

poistne_riziko <- sapply(znacka, f_znacka) + sapply(trieda,
f_trieda) + sapply(lokalita, f_lokalita) +
    0.2 * pocet_pu + exp(-0.077016*(vek-6.102647)) +
sapply(rodinne_ucely, f_rodinne_ucely) +
    0.02 * max_hmotnost + vek_auta * 0.01 + sapply(typ_motora,
f_typ_motora) + chyba
poistne_riziko <- sapply(poistne_riziko, function(r)
returnValue(min(r, 1)))
poistne_riziko <- (poistne_riziko - min(poistne_riziko)) /
(max(poistne_riziko) - min(poistne_riziko))

f_poistna_skupina <- function (p) if (p > 0.75) returnValue(3)
else if (p > 0.50) returnValue(2) else if (p > 0.25)
returnValue(1) else returnValue(0)
poistna_skupina <- factor(sapply(poistne_riziko,
f_poistna_skupina), labels = c("RP0", "RP1", "RP2", "RP3"))

data <- data.frame(znacka, trieda, lokalita, pocet_pu, vek,
rodinne_ucely, max_hmotnost, vek_auta, typ_motora, chyba,
poistne_riziko, poistna_skupina)

write.csv2(data, file=paste(dataname, "data.csv", sep = "_"))
```

Príloha č.2 – použitie ID3 na dátach s rizikovou skupinu/prirážkou

```
library(RWeka)

WPM("refresh-cache")
WPM("install-package", "simpleEducationalLearningSchemes")
WPM("load-package", "simpleEducationalLearningSchemes")

ID3 <- make_Weka_classifier("weka/classifiers/trees/Id3")

data <- read.csv2("train_data.csv")

data$vek <- as.factor(round(data$vek / 10))
data$pocet_pu <- as.factor(data$pocet_pu)
data$max_hmotnost <- as.factor(ceiling(data$max_hmotnost - 1) / 2)
data$vek_auta <- as.factor(ceiling(data$vek_auta) / 3)

tree <- ID3(poistna_skupina ~ znacka + trieda + lokalita +
pocet_pu + vek + rodinne_ucely + max_hmotnost + vek_auta +
typ_motora,
            data = data)

test_data <- read.csv2("test_data.csv")
test_data$vek <- as.factor(round(test_data$vek / 10))
test_data$pocet_pu <- as.factor(test_data$pocet_pu)
test_data$max_hmotnost <- as.factor(ceiling(test_data$max_hmotnost
- 1) / 2)
test_data$vek_auta <- as.factor(ceiling(test_data$vek_auta) / 3)

test_pred <- predict(tree, test_data, type = "class")

table(test_data[, 13], test_pred)
```

Príloha č.3 – použitie CART na dátach s rizikovou skupinu/prirážkou

```
library(rpart)

data <- read.csv2("train_data.csv")

tree <- rpart(poistna_skupina ~ znacka + trieda + lokalita +
pocet_pu + vek + rodinne_ucely + max_hmotnost + vek_auta +
typ_motora,
              data = data,
              method = "class")

par(mar = rep(0.2, 4))
plot(tree, uniform = TRUE)
text(tree)

test_data <- read.csv2("test_data.csv")
test_pred <- predict(tree, test_data, type = "class")

table(test_data[, 13], test_pred)

for (i in 1:length(tree$cptable[, "CP"])) {
    ptree <- prune(tree, cp=tree$cptable[i, "CP"])
    test_pred <- predict(ptree, test_data, type = "class")
    acc <- sum(test_data[, 13] == test_pred) / nrow(test_data)
    print(acc)
}
```

Príloha č.4 – použitie neurónových sietí na dátach s rizikovou skupinu/prirážkou

```
# nacitanie trenovacich dat
d1 <- read.csv2("train_data.csv")
d1$X <- NULL

# transformacia dat do umelych premennych
n <- names(d1)
fac_n <- names(Filter(is.factor, d1))
cs <- sapply(fac_n, function(n) contrasts(d1[[n]], contrasts =
FALSE), USE.NAMES = TRUE)
d2 <- model.matrix(~ 0 + .,
                   data = d1,
                   contrasts.arg = cs)

# konverzia na data frame
dn <- data.frame(d2)

# priprava dat
n2 <- names(dn)
for (col in n2[!n2 %in% list("chyba", "poistne_riziko",
"poistna_skupinaRP0", "poistna_skupinaRP1", "poistna_skupinaRP2",
"poistna_skupinaRP3")])
{
    # standardizacia
    #dn[,col] <- scale(dn[,col])

    # normalizacia <-1, 1>
    dn[,col] <- (dn[,col] - min(dn[,col])) /
                (max(dn[,col]) - min(dn[,col])) * 2 - 1
}

# priprava testovacich dat
td1 <- read.csv2("test_data.csv")
td1$X <- NULL

# transformacia dat do umelych premennych
tn <- names(td1)
fac_tn <- names(Filter(is.factor, td1))
tcs <- sapply(fac_tn, function(n) contrasts(td1[[n]], contrasts =
FALSE), USE.NAMES = TRUE)
td2 <- model.matrix(~ 0 + .,
                    data = td1,
                    contrasts.arg = tcs)

# konverzia na data frame
tdn <- data.frame(td2)

# priprava dat
tn2 <- names(tdn)
for (col in n2[!n2 %in% list("chyba", "poistne_riziko",
"poistna_skupinaRP0", "poistna_skupinaRP1", "poistna_skupinaRP2",
"poistna_skupinaRP3")])
{
    # standardizacia
```

```
    #dn[,col] <- scale(dn[,col])

    # normalizacia <-1, 1>
    tdn[,col] <- (tdn[,col] - min(dn[,col])) / (max(dn[,col]) -
min(dn[,col])) * 2 - 1
}

library(neuralnet)
find_net <- function (neurons, training_set, test_set,
expected_result) {
    set.seed(42)
    nn <- neuralnet(paste("poistna_skupinaRP0 + poistna_skupinaRP1
+ poistna_skupinaRP2 + poistna_skupinaRP3 ~",
                          paste(n2[!n2 %in% list("chyba",
"poistne_riziko", "poistna_skupinaRP0", "poistna_skupinaRP1",
"poistna_skupinaRP2", "poistna_skupinaRP3")], collapse = " + ")),
                    data = training_set,
                    threshold = 0.01,
                    hidden = neurons,
                    stepmax = 10^5,
                    rep = 1,
                    algorithm = "backprop",
                    act.fct = "logistic",
                    err.fct = "sse",
                    learningrate = 0.01,
                    linear.output = FALSE)

    if (length(attributes(nn)$names) == 8) {
        return(NULL)
    }

    raw_result <- compute(nn,test_set[1:(ncol(test_set)-
6)])$net.result > 0.5
    raw_result_dim <- dim(raw_result)
    result <- list()

    for (i in 1:raw_result_dim[1]) {
        row_r <- sapply(0:3, function (j) return(if (raw_result[i,
j+1]) j else -1))
        result[[i]] <- max(row_r)
    }



    result2 <- factor(result,
                      as.character(0:3),
                      labels = c("RP0", "RP1", "RP2", "RP3"))

    cv_matrix <- table(orig = expected_result, pred = result2)
    accuracy <- sum(sapply(1:4, function(i) cv_matrix[i, i])) /
                nrow(test_set)

    return(data.frame(accuracy, groups = unique(result2)))
}

results <- list()
```

```
for (i in 2:30) {
    results[[i]] <- find_net(i, dn, tdn, td1$poistna_skupina)
}
```

Príloha č.5 – použitie ID3 na dáta úverových žiadostí

```
library(CASdatasets)
data("credit")

data <- credit
data$duration <- as.factor(ceiling(data$duration / 6))
data$credit_amount <- as.factor(ceiling(data$credit_amount/ 500))
data$installment_rate <- as.factor(data$installment_rate)
data$residence_since <- as.factor(data$residence_since)
data$age <- as.factor(ceiling(data$age / 5))
data$existing_credits <- as.factor(data$existing_credits)
data$num_dependents <- as.factor(data$num_dependents)
data$result <- as.factor(data$class)
data$class <- NULL

library(caTools)
set.seed(123)
split = sample.split(data$result, SplitRatio = 0.8)
training_set = subset(data, split == TRUE)
test_set = subset(data, split == FALSE)

library(RWeka)

WPM("refresh-cache")
WPM("install-package", "simpleEducationalLearningSchemes")
WPM("load-package", "simpleEducationalLearningSchemes")

ID3 <- make_Weka_classifier("weka/classifiers/trees/Id3")

tree <- ID3(result ~ .,
            data = training_set)

test_pred <- predict(tree, test_set, type = "class")

table(test_set[, 21], test_pred)
```

Príloha č.6 – použitie CART na dáta úverových žiadostí

```
library(CASdatasets)

data("credit")

d0 <- credit

d0$installment_rate <- as.factor(d0$installment_rate)
d0$residence_since <- as.factor(d0$residence_since)
d0$existing_credits <- as.factor(d0$existing_credits)
d0$num_dependents <- as.factor(d0$num_dependents)

library(caTools)
set.seed(42)
split = sample.split(credit$class, SplitRatio = 0.8)
training_set = subset(d0, split == TRUE)
test_set = subset(d0, split == FALSE)

library(rpart)

tree <- rpart(class ~ .,
              data = training_set,
              method = "class")

par(mar = rep(0.2, 4))
plot(tree, uniform = TRUE)
text(tree)

test_pred <- predict(tree, test_set, type = "class")

table(test_set[, 21], test_pred)

results <- list()
for (i in 1:length(tree$cptable[, "CP"])) {
    ptree <- prune(tree, cp=tree$cptable[i, "CP"])
    test_pred <- predict(ptree, test_set, type = "class")
    cv_matrix <- table(test_set[, 21], test_pred)

    if (dim(cv_matrix)[2] == 2) {
        res <- data.frame(nsplit=tree$cptable[i, "nsplit"],
                          TN=cv_matrix[1,1],
                          FP=cv_matrix[1,2],
                          FN=cv_matrix[2,1],
                          TP=cv_matrix[2,2])
    } else {
        res <- data.frame(nsplit=tree$cptable[i, "nsplit"],
                          TN=cv_matrix[1,1],
                          FP=0,
                          FN=cv_matrix[2,1],
                          TP=0)
    }

    results[[i]] <- res
}
```

```
output <- Reduce(function(x, y) merge(x, y, all=TRUE), results)
output$fp_rate <- output$FP / (output$FP + output$TN)
output$tp_rate <- output$TP / (output$TP + output$FN)
output$precision <- output$TP / (output$TP + output$FP)
output$sensitivity <- output$TP / (output$TP + output$FN)
output$specificity <- 1 - output$fp_rate
output$accuracy <- (output$TP + output$TN) /
                   (output$TN + output$FP + output$FN + output$TP)
```

Príloha č.7 – použitie neurónových sietí na dáta úverových žiadostí

```
library(CASdatasets)

data("credit")

d0 <- credit

d0$installment_rate <- as.factor(d0$installment_rate)
d0$residence_since <- as.factor(d0$residence_since)
d0$existing_credits <- as.factor(d0$existing_credits)
d0$num_dependents <- as.factor(d0$num_dependents)

n <- names(d0)
fac_n <- names(Filter(is.factor, d0))
cs <- sapply(fac_n, function(n) contrasts(d0[[n]], contrasts =
FALSE), USE.NAMES = TRUE)

d1 <- model.matrix(~ 0 + .,
                   data = d0,
                   contrasts.arg = cs)
d2 <- data.frame(d1)
dn <- d2
n2 <- names(dn)

for (col in n2[!n2 %in% "class"])
{
    # standardizacia
    #dn[,col] <- scale(dn[,col])

    # normalizacia <-1, 1>
    dn[,col] <- (dn[,col] - min(dn[,col])) / (max(dn[,col]) -
min(dn[,col])) * 2 - 1
}

library(caTools)
set.seed(42)
split = sample.split(credit$class, SplitRatio = 0.8)
training_set = subset(dn, split == TRUE)
test_set = subset(dn, split == FALSE)

library(neuralnet)

find_net <- function (neurons, training_set, test_set) {
    set.seed(42)
    nn <- neuralnet(paste("class ~", paste(n2[!n2 %in% "class"],
collapse = " + ")),
                    data = training_set,
                    threshold = 0.01,
                    hidden = 21,
                    stepmax = 10^5,
                    rep = 1,
                    algorithm = "backprop",
                    act.fct = "logistic",
                    err.fct = "sse",
```

```
                    learningrate = 0.01,
                    linear.output = FALSE,
                    lifesign = "full")

    result <- compute(nn, test_set[1:ncol(test_set)-1])$net.result
> 0.5
    cv_matrix <- table(test_set[, ncol(training_set)] > 0.5,
result)

    if (dim(cv_matrix)[2] == 2) {
        returnValue(data.frame(neurons=neurons,
                               TN=cv_matrix[1,1],
                               FP=cv_matrix[1,2],
                               FN=cv_matrix[2,1],
                               TP=cv_matrix[2,2]))
    } else {
        returnValue(data.frame(neurons=neurons,
                               TN=cv_matrix[1,1],
                               FP=0,
                               FN=cv_matrix[2,1],
                               TP=0))
    }
}

results <- list()
for (i in 2:30) {
    results[[i-1]] <- find_net(i, training_set, test_set)
}

output <- Reduce(function(x, y) merge(x, y, all=TRUE), results)
output$fp_rate <- output$FP / (output$FP + output$TN)
output$tp_rate <- output$TP / (output$TP + output$FN)
output$precision <- output$TP / (output$TP + output$FP)
output$sensitivity <- output$TP / (output$TP + output$FN)
output$specificity <- 1 - output$fp_rate
output$accuracy <- (output$TP + output$TN) / (output$TN +
output$FP + output$FN + output$TP)
```