VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

DATOVÉ SADY PRO SÍŤOVOU BEZPEČNOST

DATA SETS FOR NETWORK SECURITY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. JIŘÍ SETINSKÝ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. PETER TISOVČÍK**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

148469

Ústav:	Ústav počítačových systémů (UPSY)
Student:	**Setinský Jiří, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Kybernetická bezpečnost
Název:	**Datové sady pro síťovou bezpečnost**
Kategorie:	Data mining
Akademický rok:	2022/23

Zadání:

1. Nastudujte dostupnou literaturu rozebírající použití strojového učení v oblasti analýzy síťových dat.
2. Dle pokynů vedoucího analyzujte dostupná data o síťových tocích, o bezpečnostních událostech a data z externích zdrojů.
3. Navrhněte vhodný způsob tvorby datových sad s využitím metod strojového učení.
4. Navržený způsob implementujte.
5. Implementaci ověřte vytvořením několika datových sad.
6. Diskutujte dosažené výsledky a navrhněte další rozšíření.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Tisovčík Peter, Ing.**
Konzultant:	Ing. Martin Žádník, Ph.D.
Vedoucí ústavu:	Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání:	1.11.2022
Termín pro odevzdání:	17.5.2023
Datum schválení:	31.10.2022

Abstrakt

V oblasti síťové bezpečnosti se používají techniky strojového učení pro efektivní detekci anomálií a malwaru v síťovém provozu. Pro natrénování síťového klasifikátoru s vysokou úspěšností je potřeba kvalitní datová sada. Cílem práce je modifikace datové sady pomocí metod strojového učení za účelem zlepšení kvality datové sady, která povede na natrénování modelu s vyšší úspěšností. Datová sada je zanalyzována shlukovacím algoritmem a každý shluk je charakterizován statistickým popisem vyplývající z atributů vstupní datové sady. Statistický popis spolu s informacemi o původním klasifikátoru je použit pro výpočet skóre. Skóre slouží jako váha při modifikační fázi. Shluková analýza umožní vyfiltrovat data, která jsou důležitá pro natrénování výsledného modelu. Navržený přístup umožňuje zmírnit redundanci datové sady a nebo ji rozšířit o chybějící data. Výsledkem je modifikační framework, který je schopen redukovat datové sady nebo provádět jejich agregaci za účelem vytvoření kompaktní datové sady, která bude reflektovat aktuální síťový provoz. Na vytvořených datových sadách se podařilo natrénovat modely dosahující vyšší úspěšnosti v porovnání s existujícím řešením.

Abstract

In network security, machine learning techniques are used to effectively detect anomalies and malware in network traffic. A quality dataset is needed to train a network classifier with high accuracy. The aim of this paper is to modify the dataset using machine learning techniques to improve the quality of the dataset which will lead to training the model with a higher accuracy. The dataset is analyzed by a clustering algorithm and each cluster is characterized by a statistical description resulting from the attributes of the input dataset. The statistical description along with the information of the original classifier is used to compute the score. The score serves as a weight in the modification phase. Cluster analysis allows to filter out the data that are important for training the final model. The proposed approach allows us to mitigate the redundancy of the dataset or to augment it with missing data. The result is a modification framework that is able to reduce the datasets or perform their aggregation in order to create a compact dataset that reflects the actual network traffic. Models were trained on the created datasets and achieved higher accuracy compared to the existing solution.

Klíčová slova

Shlukování, Datové sady, Síťová bezpečnost, Strojové učení, Kvalita datové sady, Augumentace dat, Redukce dat, DGA, DoH, Modifikace dat, Úspěšnost modelu.

Keywords

Clustering, Datasets, Network security, Machine learning, Quality of dataset, Data augmentation, Data reduction, DGA, DoH, Data modification, Model accuracy.

Citace

SETINSKÝ, Jiří. *DATOVÉ SADY PRO SÍŤOVOU BEZPEČNOST*. Brno, 2023. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Peter Tisovčík

DATOVÉ SADY PRO SÍŤOVOU BEZPEČNOST

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Petra Tisovčíka. Další informace mi poskytl pan Ing. Martin Žádník, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jiří Setinský
16. května 2023

Poděkování

Rád bych chtěl poděkovat svému vedoucímu Ing. Petrovi Tisovčíkovi za jeho vstřícnost, trpělivost, ochotu a čas, který mi věnoval při vypracovávání diplomové práce. Dále chci poděkovat Ing. Martinovi Žádníkovi, Ph.D. za jeho prvotní vedení. Na závěr chci poděkovat své rodině a blízkým za veškerou podporu.

Obsah

Kapitola 1

Úvod

V dnešní době koluje po síti obrovské množství dat. Značnou část dat tvoří nelegitimní komunikace. Cílem je tento typ komunikace detekovat, nahlásit a vhodně zpracovat. Efektivním přístupem pro detekci se staly techniky strojového učení. Většinou se používají metody založené na učení s učitelem, jenž k natrénování klasifikátoru vyžadují trénovací datovou sadu. Důležitým faktorem pro dobře fungující klasifikátor je správná datová sada. V rámci této práce je kladen důraz na modifikaci datových sad. Pokud budeme mít k dispozici kvalitní datovou sadu, tak i výsledný klasifikátor bude vykazovat lepší výsledky a získáme tak přesnější nástroje pro detekci nelegitimního provozu na síti.

Práce se zabývá analýzou datových sad pomocí metod strojového učení a následnou modifikací konkrétní datové sady, která přispěje k lepším metrikám natrénovaného modelu. Chceme najít metodu, která odhalí nedostatky datové sady a na jejich základě provést modifikaci. Modifikovaná datová sada by měla následně sloužit jako vylepšená trénovací sada pro konkrétní klasifikátor. Kladen je důraz na doménu detekce anomálií v síťové oblasti.

Prezentované řešení je obecně použitelné na libovolnou datovou sadu, ale dosahuje specifických výsledků v aplikované doméně. Existující řešení můžeme rozdělit na dvě části. První část je obecná kvalita datové sady. Kvalitní datová sada musí splňovat určité aspekty na několika úrovních. Na nejvyšší úrovni musí datová sada splňovat obecné požadavky. Při zkoumání kvality datové sady na nižší úrovni už je nutností mít detailní informace o konkrétní doméně. Obecná kvalita datové sady je těžko určitelná, jelikož každá datová sada má specifický případ užití. Mnoho článků se zabývá spíše kvalitou samotného modelu místo datové sady, jelikož kvalita modelu je silně spjata s kvalitou datové sady. Druhá část se zabývá modifikací pro získání vyvážené datové sady, kdy je snahou zredukovat majoritní nebo rozšířit minoritní třídu. Metody pro vyvažování budou použity pro porovnání s navrženým řešením.

Datové sady budou podrobeny shlukové analýze. Přístup provádí shlukování vstupní datové sady na základě dodaných atributů. Počet shluků je určen automaticky na základě vhodné heuristiky. Jednotlivé shluky jsou interpretovány automaticky vygenerovanou zprávou. Zpráva je tvořena statistickým popisem jednotlivých atributů. Cílem je vytvořit obecnou zprávu, aby se dala získat na libovolných vstupních datech. Díky zprávě získáme přehled o datech v rámci jednotlivých shluků. Zpráva spolu s využitím původního klasifikátoru slouží pro výpočet skóre každého shluku. Skóre je normalizované a je použito jako váha při modifikaci. V modifikační fázi se na základě váhy shluků rozhodneme, která data ve vstupní sadě ponechat s jakým poměrem. Kromě redukce může dojít i ke spojení více datových sad. Nová data jsou opět analyzována a získané váhy umožní přidat pouze data, která umí klasifikátor predikovat s nižší úspěšností.

Přínosem uvedeného řešení je například udržování kompaktní datové sady. Datová sada po modifikaci nebude obsahovat velké množství redundantních dat. Redundance je odstraněna za pomocí redukce. Dále získáme možnost efektivně spojovat datové sady. Spojením obohatíme datovou sadu o data, která nemusí model správně klasifikovat. Tímto způsobem můžeme udržovat datovou sadu aktuální. Můžeme přidávat nová data, která reflektují aktuální provoz na síti a zároveň si ponechat stará data, která se na síti stále mohou objevovat. Současně díky redukci nebude velikost datové sady neustále růst.

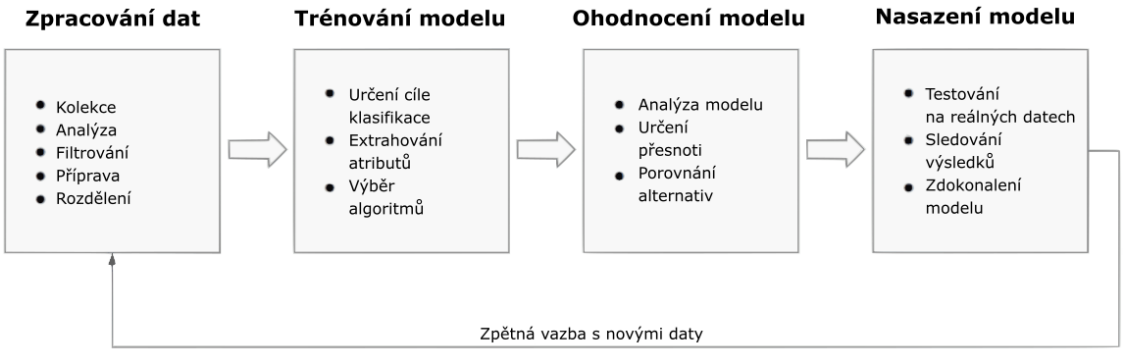
Výslednému cíli práce bude předcházet několik kroků. V kapitole 2 bude probrána teorie strojového učení v síťové oblasti, konkrétně metody shlukové analýzy. Kapitola 3 uvede problematiku určování kvality datové sady a nastíní způsoby modifikování a generování datových sad. Popis frameworku na modifikaci datových sad bude obsahovat kapitola 4. Následně bude popsána jeho implementace v kapitole 5. Předposlední kapitola 6 provede testování na reálných datech a rozebere výsledné řešení s možností dalšího rozšíření. V neposlední řadě kapitola 7 shrne celou práci.

Kapitola 2

Strojové učení

Strojové učení (Machine learning) nabývá na popularitě a velké množství odvětví ho začíná využívat. To stejné platí i pro oblast sítí. Díky nárůstu výpočetního výkonu a ukládání obrovského množství dat nastala doba, kdy je výhodné začít aplikovat strojové učení v síťovém prostředí. Strojové učení slouží k různým typům úloh, jenž jsou uvedeny v sekci rozdělení úloh 2.1. Všechny typy úloh jsou ve výsledku postaveny na stejných principech, jenž zahrnují sbíráni a předzpracování dat, výběr vhodné metody a její optimalizaci, vyhodnocení modelu spolu s vizualizací a finální nasazení modelu. Pomocí strojového učení se snažíme výpočetně reprezentovat libovolný jev, jenž slouží k vykonání požadované úlohy na základě okolního prostředí a předchozích zkušeností.

Typický proces strojového učení je zobrazen na obrázku 2.1. Proces je někdy označován jako získávání znalostí z databází. V první fázi máme na vstupu data z libovolného úložiště (např. databáze). Součástí předzpracování dat je detekce odlehlých hodnot, odstranění chybějících hodnot, filtrace nebo agregace atributů. Dále můžeme provést normalizaci dat. Následuje transformace dat, která má za úkol vybrat vhodnou podmnožinu atributů, odstranit nerelevantní informace a převést data do kompatibilního formátu s cílovou úlohou. Transformace dat je nepostradatelnou součástí, jelikož má velký dopad na celkovou kvalitu výsledného modelu. Připravená data jsou použita k natrénování zvoleného modelu. Proces trénování spočívá v ladění parametrů modelu za účelem získaní nejlepšího modelu v ohledu na úspěšnost, schopnost generalizace a výpočetní nároky [27]. V momentě výběru modelu vykazující nejlepší výsledky pro danou úlohu, tak přichází na řadu nasazení modelu do reálného prostředí. Nasazený model sledujeme a sbíráme zpětnou vazbu (např. nová data), která může být použita pro zdokonalení modelu v opakujících se cyklech.



Obrázek 2.1: Průběh cyklického vývoje modelu v oblasti strojového učení [2].

2.1 Rozdělení úloh

Nejčastěji se v procesu strojového učení používají tyto typy úloh [29]:

- **Klasifikace** – Klasifikace patří do kategorie učení s učitelem. Klasifikace řeší problém prediktivního modelování, kde se pro daná vstupní data předpovídá, do které kategorické třídy patří. Matematicky máme funkci **f**, jenž mapuje vstupní proměnné **X** na výstupní proměnné **Y** reprezentující výsledný cíl, anotaci nebo kategorii. Predikce třídy daného datového vektoru může být provedena na strukturovaných nebo nestrukturovaných datech. Příkladem klasifikace může být například detekce spamu u poskytovatelů e-mailových služeb, kde máme dvě třídy rozděleny na *je spam* a *není spam*.

- **Regrese** – Regresní analýza zahrnuje několik metod strojového učení, které umožňují předpovídat výslednou spojitou proměnnou **y** na základě hodnoty jedné nebo více proměnných **x**. Nejvýznamnější rozdíl mezi klasifikací a regresí spočívá v tom, že klasifikace předpovídá třídy kategorického charakteru, zatímco regrese umožňuje predikci spojité veličiny. Mezi těmito dvěma typy úloh strojového učení se často vyskytují určité překryvy. Regresní modely jsou v současnosti široce používány i v jiných oblastech než v IT, například ve finančnictví, marketingu či farmacii. Mezi známé typy regresních algoritmů patří například lineární nebo polynomiální metody.

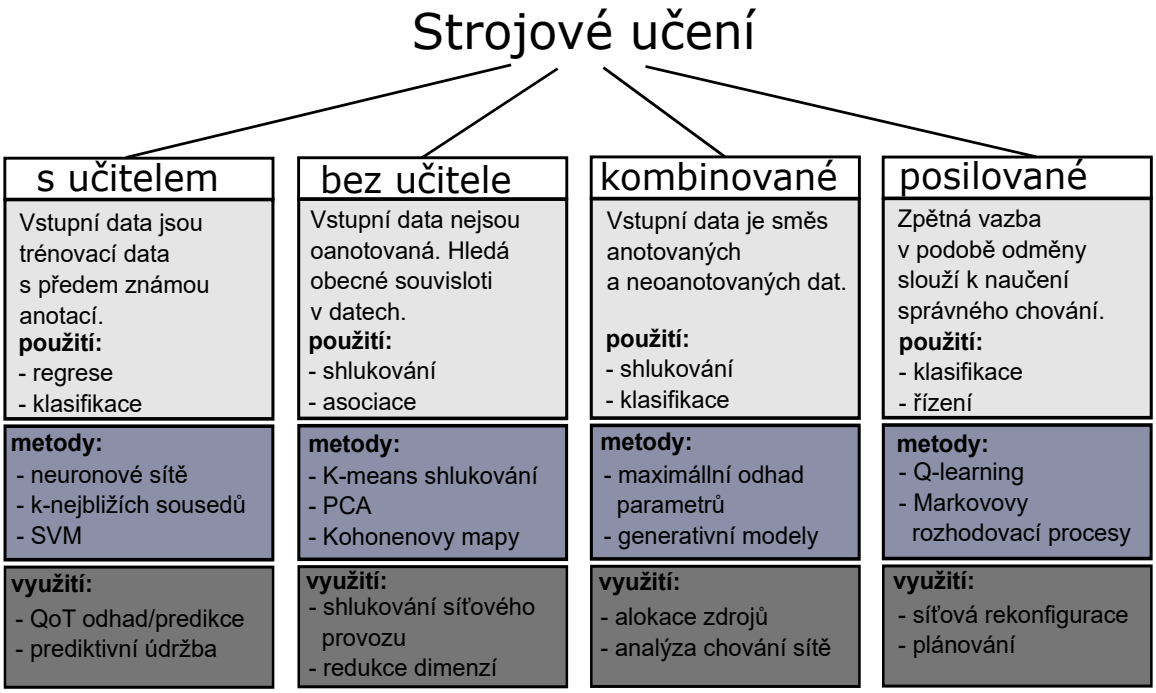
- **Shlukování** – Shluková analýza, také známa jako shlukování, je technika strojového učení bez učitele pro identifikaci a seskupování souvisejících datových bodů ve velkých datových sadách bez požadavku na konkrétní výsledek. Seskupuje objekty takovým způsobem, že objekty ve stejné kategorii, nazývané shluk, jsou si v určitém smyslu více podobné než objekty v jiných skupinách. Často se používá jako technika analýzy dat k odhalení zajímavých trendů nebo vzorů v datech, např. skupiny spotřebitelů na základě jejich chování. Shlukování lze využít V široké škále oblastí (obecně analýza rozsáhlých dat), jako je kybernetická bezpečnost, komerční sféra, zdravotní analýza nebo analýza chování. V sekci 2.2 budou podrobněji probrány různé metody shlukování.

- **Redukce dimenzí** – V oblasti dat a strojového učení je zpracování vysoko dimenzionálních dat náročným úkolem jak pro výzkumníky, tak pro vývojáře aplikací. Redukce dimenzí, což je technika učení bez učitele, je tedy důležitá, protože vede k lepší interpretaci, nižším výpočetním nárokům a snižuje míru přetrénování a redundanci zjednodušením modelů. Pro redukci dimenzí lze použít proces výběru příznaků nebo extrakci příznaků. Hlavním rozdílem mezi výběrem a extrakcí atributů spočívá v tom, že *výběr atributů* zachovává podmnožinu původních atributů, zatímco *extrakce atributů* vytváří zcela nové. Techniky redukce dimenzí budou detailněji popsány v sekci 2.3.

- **Asociační analýza** - Učení asociačních pravidel je přístup strojového učení založený na pravidlech k objevování zajímavých vztahů. Pravidla jsou v podobě příkazů `IF-THEN` mezi položkami ve velkých datových sadách. Jedním z příkladů je, že *pokud si zákazník koupí počítač nebo notebook (položku), pravděpodobně si zároveň koupí také antivirový software (jinou položku)*. Asociační pravidla se dnes používají v mnoha aplikačních doménách, včetně IoT služeb, lékařské diagnózy, analýzy chování uživatelů, vytížení webu, aplikací pro chytré telefony, aplikací kybernetické bezpečnosti a bioinformatiky. Ve srovnání s hledáním sekvencí, učení asociačních pravidel obvykle nebere v úvahu pořadí v rámci nebo napříč transakcemi.

Celkové rozdělení úloh strojového učení je znázorněno na obrázku 2.2, který zároveň uvádí kontext výše zmíněných úloh. Jsou popsány jednotlivé přístupy spolu s metodami, které se hojně používají. Přístupy jsou doplněny o praktické využití v oblasti sítí.

V rámci práce bude kladen důraz hlavně na přístup bez učitele, jenž má na vstupu neoanotovaná data, ve kterých se hledají souvislosti. Praktická část práce bude primárně experimentovat se shlukovacími algoritmy a s metodami na redukci dimenzí. Proto budou následovat sekce 2.2 a 2.3, které se budou zabývat teorií shlukové analýzy a metodami na redukci dimenzí.



Obrázek 2.2: Souhrnné rozdělení typů strojového učení do kategorií. První box popisuje obecnou charakteristiku daného přístupu. Druhý box uvádí příklady algoritmů, jenž jsou v dané kategorii využívány. Poslední box uvádí využití metod na konkrétních příkladech ze síťové oblasti [27].

2.2 Shluková analýza

Cílem následujícího textu je uvést metody shlukové analýzy a teoretické podklady pro jejich aplikování. Obecná definice shlukování byla uvedena v sekci 2.1. Hledání jednotlivých shluků je založeno na atributu vyjadřující vzájemnou podobnost objektů. Formálně je struktura shluků reprezentována jako množina podmnožin $C = C_1, ..., C_k \; z \; S \mid S = \bigcup_{i=1}^{k} = C_i \; a \; C_i \cap C_j = \emptyset \; pro \; i \neq j$. V důsledku toho každá instance v S patří přesně do jedné a pouze jedné podmnožiny [28].

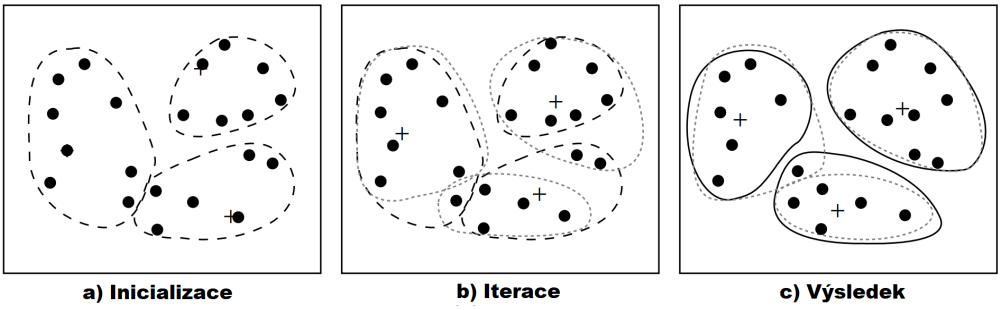
V rámci shluku chceme, aby objekty měly co největší podobnost, avšak co nejmenší podobnost napříč shluky. Veličina podobnosti je získána z atributů popisující porovnávané objekty, mezi kterými je obvykle spočítána vzdálenost v n-rozměrném prostoru [14]. Přehled metod shlukování bude užitečný v praktické části práce 4, kde budou aplikovány metody shlukování za účelem modifikování datové sady.

2.2.1 Metody založené na rozdělování

Nejjednodušší a nejzákladnější verzí shlukové analýzy je pomocí rozdělování, které organizuje položky datové sady do několika disjunktních shluků. Pro usnadnění můžeme předpokládat, že máme předem známý počet shluků. Tento parametr je výchozím bodem pro rozdělovací metody. Formálně máme datovou sadu D obsahující n objektů s počtem shluků k, které se mají vytvořit. Rozdělovací algoritmus organizuje objekty do k celků ($k \leq n$), kde každý celek představuje shluk. Shluky jsou vytvořeny tak, aby bylo optimalizováno rozdělovací kritérium, jenž může být reprezentováno jako funkce podobnosti na základě vzdálenosti. Objekty ve shluku jsou si navzájem podobné a současně nepodobné objektům v ostatních shlucích z hlediska atributů datové sady. K nalezení optimálního řešení bychom museli vyhodnotit všechna možná rozdělení datové sady D, což vede na NP těžký problém už při dvou shlucích. V praxi se využívá heuristik, jenž umožní snížení výpočetní náročnosti [14]. Nyní následuje uvedení nejznámějších a běžně používaných metod dělení – *k-means* a *k-medoids* (součástí budou i různé varianty algoritmů).

K-means

Technika využívající k reprezentaci shluku C_i středový bod. Středový bod je definován jako střední hodnota bodů uvnitř shluku. V první fázi algoritmus vybere náhodně k objektů z dat. Každý zvolený objekt je považován za střed shluku. Zbývající objekty se přiřadí do shluku, kterému se nejvíce podobá. Podobnost je spočítána jako euklidovská vzdálenost mezi daným bodem a středem shluku. *K-means* pracuje na iterativní bázi. Proces tvorby shluků je ilustrován na obrázku 2.3. V každé iteraci se pro každý shluk spočítá nová střední hodnota z bodů uvnitř shluku, jenž byly přiřazeny v předešlé iteraci. Nové střední hodnoty jsou následně použity k přerozdělení shluků. Iterace probíhají dokud se shluky neustálí, což znamená, že nový shluk se neliší od shluku z předešlé iterace.



Obrázek 2.3: Proces vytvoření shluků metodou *k-means* [14].

Metoda není optimální, výsledek často záleží na prvotní náhodné inicializaci středových bodů. Řešením může být spuštění algoritmů vícekrát s různou inicializací. Složitost metody je $O(nkt)$, kde n je počet objektů, k počet shluků a t počet iterací. Běžně platí podmínka $k < n \wedge t < n$. Proto je metoda celkem dobře škálovatelná a efektivní při zpracování velkých datových sad. Máme několik variant algoritmu, které se liší v inicializaci středů, počítání podobnosti nebo ve způsobu počítání střední hodnoty. Nevýhodou je, že uživatel musí dodat hodnotu k. Existují ale přístupy, které umožní automaticky odhadnout k. Mezi další nevýhody patří špatná shlukovatelnost objektů, které tvoří nekonvexní nebo nekonzistentně velké shluky. Metoda je citlivá na šum, či odlehlé hodnoty, jelikož podstatně ovlivní výslednou střední hodnotu [14].

K-medoids

Cílem metody k-medoids je odstínit problematiku odlehlých hodnot. Pokud do shluku přiřadíme odlehlou hodnotu, tak se rapidně změní těžiště shluku (střední hodnota) a následně se do shluku přiřadí i potenciálně nechtěné objekty. Pokud místo střední hodnoty zvolíme jako reprezentaci sluku přímo objekt shluku, tak se podaří snížit citlivost na odlehlé hodnoty.

Algoritmus *Partitioning Around Medoids (PAM)* je nejčastější realizací shlukování techniky k-medoids. Opět k problému přistupuje pomocí iterativního vylepšování shluků. Na začátku je zvoleno k objektů libovolným způsobem. Následně přichází na řadu nalezení takového objektu, jenž by jako nový reprezentant shluku zajistili lepší kvalitu rozdělení. Jsou vyzkoušeny všechny potenciální objekty. Proces výběru reprezentujícího objektu trvá tak dlouho, dokud výsledná kvalita rozdělení nemůže být zlepšena výměnou za jiného reprezentanta. Kvalita rozdělení je spočítána jako průměrná vzdálenost reprezentujícího objektů od zbylých objektů ve shluku.

Pro každý nereprezentující objekt p zjistíme, zda dojde k jeho přesunu následujícím způsobem [14]:

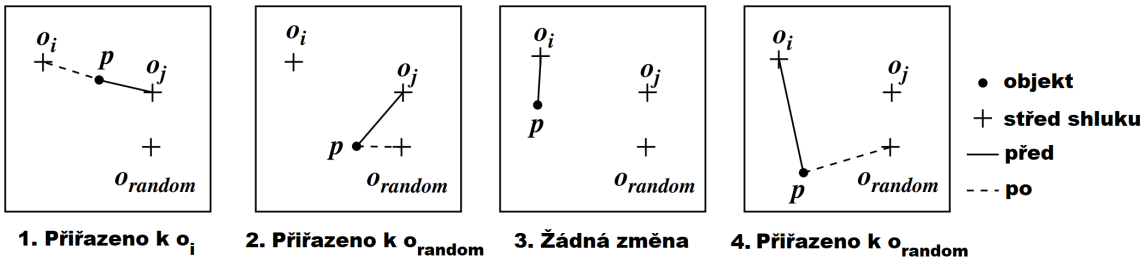
1) Objekt p patřil do třídy objektu o_j. Jestliže nahradíme objekt o_j objektem o_{random} a objekt p bude nejblíže k nějakému jinému centrálnímu objektu o_i, $i \neq j$, pak objekt p se přesune do třídy objektu o_i.

2) Objekt p patřil do třídy objektu o_j. Jestliže nahradíme objekt o_j objektem o_{random} a objekt p bude nejblíže k objektu o_{random}, pak objekt p se přesune do třídy objektu o_{random}.

3) Objekt p patřil do třídy objektu o_i. Jestliže nahradíme objekt o_j objektem o_{random} a objekt p bude stále nejblíže k objektu o_i, pak objekt p se nepřesune.

4) Objekt p patřil do třídy objektu o_i. Jestliže nahradíme objekt o_j objektem o_{random} a objekt p bude nejblíže k objektu o_{random}, pak objekt p se přesune do třídy objektu o_{random}.

Obrázek 2.4 vizualizuje uvedené varianty přesunu objektu p. Pokud se přesunem objektu p zmenší vzdálenost od jeho reprezentujícího objektu, tak to přispěje ke zmenšení celkové sumě vzdáleností všech nereprezentujících objektů a vyšší kvalitě výsledného rozdělení, a proto může být objekt o_j nahrazen novým reprezentantem o_{random}.



Obrázek 2.4: Varianty přerozdělení objektů metodou *k-medoids* [14].

Uvedenou metodou získáme robustnější řešení, avšak složitost každé iterace algoritmu je $O(k(n-k)^2)$. Pro velké hodnoty n a k se stává metoda velmi výpočetně náročná, čímž přicházíme o dobrou škálovatelnost. Uživatel musí i nadále specifikovat hodnotu k. Řešením

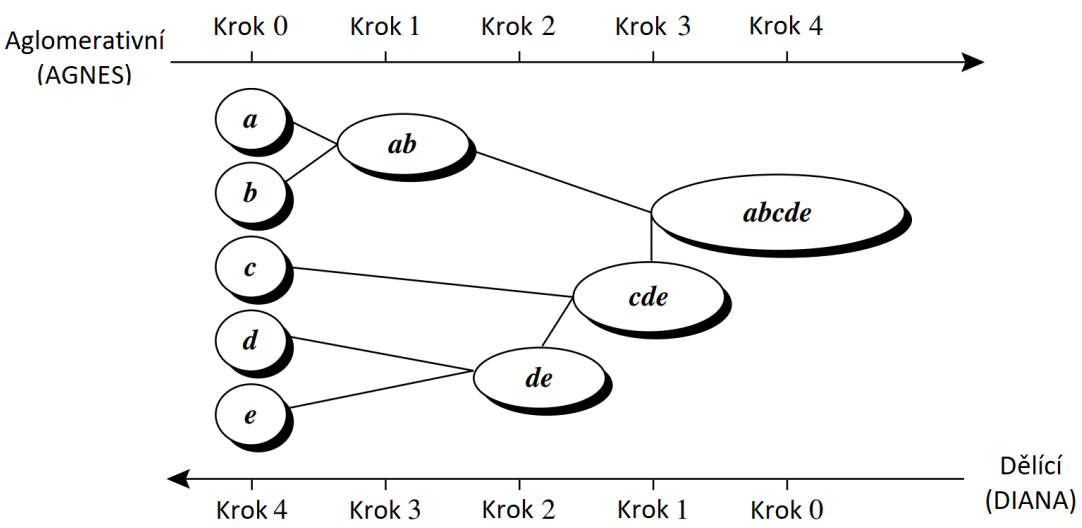
pro lepší škálovatelnost může být metoda *CLARA (Clustering LARge Applications)*. Je založena na stejném principu, ale provádí výpočty na náhodně vybraných vzorcích dat, což vede na nižší výpočetní složitost [14].

2.2.2 Hierarchické metody

Tyto metody vytvářejí hierarchické shluky rekurzivním rozdělením množiny dat buď shora dolů, nebo zdola nahoru. Rozdělení metod je uvedeno níže:

- **Aglomerativní shlukování** - Každý objekt zpočátku představuje samostatný shluk. Poté jsou podobné shluky postupně slučovány, dokud není dosažena požadovaná úroveň shlukování nebo jsou všechny shluky spojeny do jednoho shluku.
- **Dělící shlukování** – Všechny objekty zpočátku patří do jednoho shluku. Poté se shluk rozdělí na menší shluky, které se rekurzivně dělí na další podshluky. Tento proces pokračuje, dokud není získána požadovaná struktura shluků nebo každý objekt reprezentuje samostatný shluk.

Výsledkem hierarchických metod je dendrogram (Obrázek 2.5), reprezentující zanořené seskupení objektů a úrovně podobnosti, na kterých se seskupení mění. Shlukování datových objektů je získáno řezem dendrogramu v požadované úrovni podobnosti.



Obrázek 2.5: Dendrogram rerezentující dělící metodu (DIANA) a aglomerativní metodu (AGNES) nad pěti datovými objekty [14].

Slučování nebo dělení shluků se provádí podle určité metriky podobnosti zvolené tak, aby se optimalizovalo nějaké kritérium (např. suma čtverců). Metody hierarchického shlukování lze dále rozdělit podle způsobu, jakým se vypočítá míra podobnosti [28]:

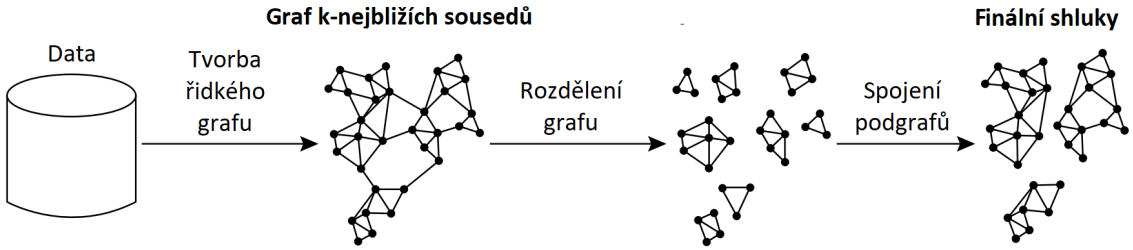
- **Minimální vzdálenost** - metody, které berou v úvahu vzdálenost mezi dvěma shluky jako nejkratší vzdálenost mezi libovolným členem jednoho shluku a libovolným členem druhého shluku. Pokud existuje mezi daty podobnost, tak podobnost dvou shluků je interpretována jako vzdálenost mezi dvěma nejpodobnějšími objekty z daných shluků.
- **Maximální vzdálenost** - metody, které reprezentují vzdálenost mezi dvěma shluky jako nejdelší vzdálenost mezi libovolnými objekty porovnávaných objektů.

- **Průměrná podobnost** - metody, které považují vzdálenost mezi dvěma shluky za zprůměrovanou vzdálenost mezi všemi objekty napříč shluky.

Nevýhodou metod využívající minimální vzdálenost je tendence spojovat dohromady shluky, jenž obsahují body, které tvoří tzv. přemostění mezi dvěma shluky. Na druhou stranu jsou metody univerzální a vykazují dobrou shlukovací schopnost nad různými topologiemi shluků.

Metody počítající průměrnou vzdálenost způsobují, že podlouhlé shluky se rozdělí a části podlouhlých sousedících shluků se naopak spojí. Oproti minimální vzdálenosti jsou schopny produkovat více kompaktní shluky a lépe využitelnou hierarchii.

Dalším hierarchickým algoritmem je **Chameleon**, který využívá dynamické modelování. Postupně spojuje malé shluky velmi podobných objektů. Je založen na tvorbě grafu k-nejbližších sousedů. Nejprve je každý objekt spojen váhovanou hranou s jeho k-nejbližšími sousedy. Poté se provede rozdělení grafu pomocí řezů s co nejmenší váhou. Na závěr dojde ke spojení grafů na základě relativního propojení a relativní blízkosti shluků. Proces tvorby shluků zobrazuje obrázek 2.6. Metoda je schopna identifikovat shluky různých tvarů.



Obrázek 2.6: Proces tvorby shluků algoritmem Chameleon pomocí dynamického modelování grafů nejbližších sousedů [14].

Obecnou výhodou hierarchických metod je schopnost nevytvářet pouze jedno rozdělení, ale více vnořených rozdělení, které umožňují různým uživatelům vybrat vhodné rozdělení podle požadované úrovně podobnosti. Mezi nevýhody hierarchických metod patří špatná škálovatelnost. Časová složitost hierarchických algoritmů je $O(m2)$, kde m je celkový počet instancí. Hierarchické metody nikdy nemohou vrátit zpět to, co bylo provedeno dříve. Konkrétně neexistuje žádná možnost zpětného navracení [28].

2.2.3 Metody založené na hustotě

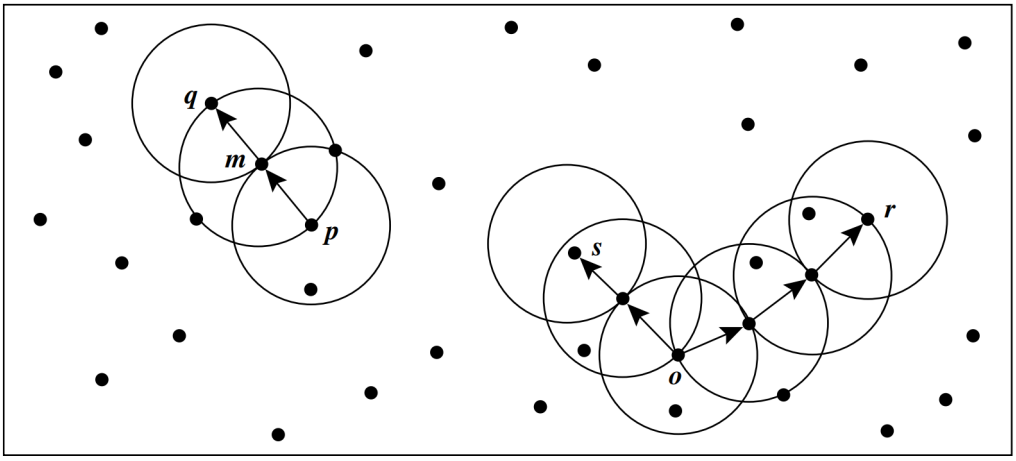
Doposud uvedené metody většinou vytváří shluky kulovitého tvaru. Pokud chceme identifikovat shluk, jenž je libovolného nebo nekonvexního tvaru, tak musíme přistoupit k metodám založených na hustotě. Principem je modelování shluku jako silně zahuštěného regionu v datovém prostoru odděleným regiony s nízkou hustotou. Nyní bude následovat popis známých metod využívající hustotu.

DBSCAN

DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) hledá datové objekty s vysoce zahuštěným okolím, jenž jsou označeny jako jádra. Hustota objektu o je vyjádřena jako počet objektů v jeho okolí. Vstupem metody je parametr velikosti okolí $\epsilon > 0$. Okolí ϵ objektu o je radius o poloměru ϵ se středem o.

Definováním okolí lze hustotu objektu vyjádřit jako počet objektů v daném okolí. Pro zjištění, zda má objekt vysokou hustotu je potřeba zadat další parametr *MinPts*, který označuje hranici pro okolí s velkou hustotou. Objekt je označen za jádro, pokud je ϵ okolí obsahuje alespoň *MinPts* objektů. Jádra reprezentují silně zahuštěné regiony.

Cílem shlukování je spojovat jádra a jejich okolí do větších regionů, jenž budou reprezentovat finální shluky. Ke spojování jader a okolí využívá metoda princip propojenosti na základě hustoty v prostoru datových objektů **D**. Dva objekty $\boldsymbol{p_1, p_2}$ jsou **propojeny na základě hustoty** s ohledem na parametry ϵ a *MinPts*, pokud existuje objekt $\boldsymbol{q}$, pro který platí, že $\boldsymbol{p}_1$ a $\boldsymbol{p}_2$ jsou dosažitelné na základě hustoty z $\boldsymbol{q}$. **Propojenost na základě hustoty** je relace ekvivalence, tudíž mohou být objekty propojovány pomocí tranzitivnosti. Dosažitelnost je definována následovně. Objekt $\boldsymbol{p}$ je **dosažitelný na základě hustoty** z objektu $\boldsymbol{q}$, jestliže existuje řetězec objektů $\boldsymbol{p_i, ..., p_n}$, kde $\boldsymbol{p_1 = q}$ a $\boldsymbol{p_n = p}$, takový, že objekt $\boldsymbol{p_{i+1}}$ je přímo dosažitelný na základě hustoty z objektu $\boldsymbol{p_i}$, pro $1 \leq i \leq n, p_i \in D$. Objekt $\boldsymbol{p}$ je **přímo dosažitelný na základě hustoty** z objektu $\boldsymbol{q}$, jestliže $\boldsymbol{p}$ se nachází v ϵ okolí objektu $\boldsymbol{q}$ a $\boldsymbol{q}$ je jádro. Propojenost na základě hustoty vytvoří uzavřené množiny, jenž tvoří výsledné shluky [14]. Obrázek 2.7 znázorňuje proces tvoření shluků.



Obrázek 2.7: Proces tvorby shluků metodou DBSCAN pomocí propojenosti objektů na základě hustoty. Metoda je schopna ignorovat šum a odlehlé hodnoty [14].

DENCLUE

Další metodou je DENCLUE (*DENsity-based CLUstEring*), která je založena na určení hustoty pomocí pravděpodobnostního rozdělení. Budeme se snažit modelovat hustotu pravděpodobnosti dle vstupních datových objektů. Vstupní data bereme jako náhodný vzorek, který byl vygenerován hledaným rozdělením pravděpodobnosti. Datový objekt považujeme za indikátor vyšší hustoty pravděpodobnosti v dané oblasti. Hustota pravděpodobnosti závisí na vzdálenostech ostatních bodů od zkoumaného bodu.

Formálně máme definovanou nezávislou náhodnou proměnnou f, jenž představuje datové objekty $x_1, ..., x_n$. Hustotu daného bodu aproximujeme pomocí rovnice 2.1

$$f_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{\boldsymbol{x - x_i}}{h}), \tag{2.1}$$

kde $K()$ je funkce vlivu a h je vyhlazovací parametr. Funkce vlivu slouží k vyjádření síly, jak daný bod ovlivňuje své okolí. Funkce musí být symetrická a integrovatelná s celkovým obsahem jedna přes všechny proměnné. Jako častá funkce vlivu se používá Gaussova funkce s rovnicí 2.2.

$$K(\frac{\boldsymbol{x} - \boldsymbol{x_i}}{h}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(\boldsymbol{x}-\boldsymbol{x_i})^2}{2h^2}} \tag{2.2}$$

Cílem je najít takové body (ohniska hustoty), které budou představovat lokální maxima s nejvyšší hustotu pravděpodobnosti. Triviální lokální minima (šum nebo odlehlé hodnoty) jsou řešeny prahováním. Aby byl bod označen jako ohnisko hustoty, tak musí splňovat podmínku $f(x) \geq \xi$, kde ξ je prahová hodnota definovaná uživatelem. Ohniska hustoty jsou spočítána pomocí *hill climbing* algoritmu, který hledá lokální maxima pomocí gradientů.

Shluk je následně reprezentován jako množina ohnisek a množina bodů, jenž spadají do daných ohnisek. A zároveň mezi všemi dvojicemi ohnisek existuje cesta, která vykazuje hustotu vetší než ξ. Díky propojování ohnisek může metoda identifikovat shluky libovolného tvaru. Metoda není citlivá na šum a odlehlé hodnoty, jelikož funkce vlivu (Gaussova funkce) vyhladí dané odchylky napříč všemi vstupními daty [14].

2.2.4 Metody založené na mřížce

Metody pracují s prostorem, který je reprezentován konečným počtem buněk tvořící mřížku. Mřížka je vytvořena nezávisle na distribuci vstupních objektů. Veškeré výpočty během shlukování probíhají pouze na mřížkové struktuře, což vede k efektivní časové složitosti. Složitost je často nezávislá na počtu vstupních hodnot, jelikož hlavním faktorem složitosti je počet buněk, na které je prostor kvantizován. Bude uvedena metoda STING využívajících statistických parametrů buněk a metoda CLIQUE, která kombinuje mřížku spolu s hustotou ke zpracování vysoce dimenzionálních dat.

STING

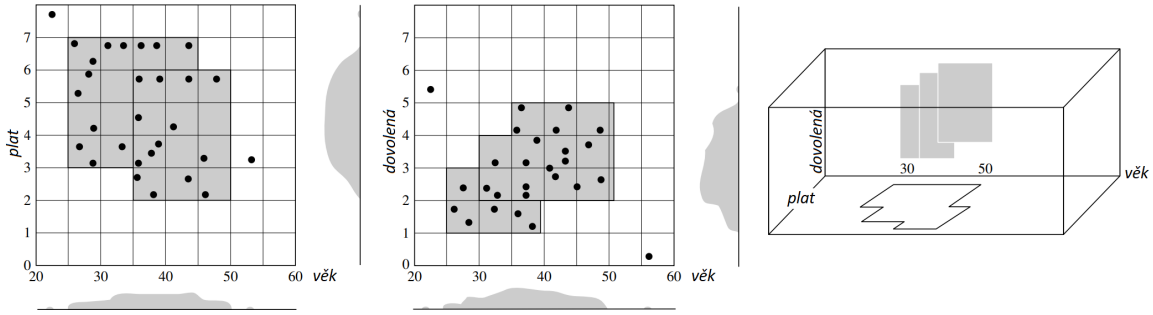
STING (*STatistical INformation Grid*) rozděluje vstupní data do buněk. Buňky se tvoří rekurzivním způsobem a dochází ke vzniku hierarchického uspořádání. Buňka na vyšší úrovní je rozdělena do buněk na nižší úrovni. Pro každou buňku jsou spočítány statistiky jako je průměr, minimun, maximum nebo směrodatná odchylka. Každé buňce je přiřazen typ rozdělení, který nejlépe charakterizuje data v dané buňce. Statistické parametry jsou počítány od spodních vrstev směrem k vyšším vrstvám využívajících distributivních výpočtů. Shluky jsou tvořeny přirozeně od nejvyšších vrstev díky mřížkové struktuře. Buňky na vyšších vrstvách sumarizují data v dané buňce. Nad mřížkou můžeme provádět různé dotazy. Můžeme ponechat pouze buňky, které mají vyšší pravděpodobnost zastoupení dat. Postupně můžeme získat jen buňky, které jsou pro nás relevantní a splňují předem definované požadavky.

Metoda využívá silně paralelního přístupu, výpočet je nezávislý na vstupních parametrech, protože využíváme už předem spočítaných statistik. Můžeme snadno rozšiřovat datový prostor. Tvorba shluků má časovou složitost $O(n)$, jelikož stačí vypočítat statistiky datových objektů o velikost n. Kvalita metody závisí na granularitě nejnižších vrstev. Pokud máme velkou granularitu, tak se zvýší nároky na výpočet a naopak při malé granularitě se snižuje kvalita shlukové analýzy. Omezením metody je také, že hranice mezi shluky budou vždy kolmé nebo rovnoběžné [14].

CLIQUE

V mnoha případech vstupní data obsahují vysoký počet atributů, což ztěžuje hledaní shluků. Metoda CLIQUE (*Clustering in QUEst*) je metoda založená na mřížce s tím rozdílem, že vyhledává shluky v různých podprostorech vstupních dat. Každá dimenze je rozdělena na intervaly a výsledkem je rozdělení prostoru dat na multidimenzionální buňky. Je stanoven práh hustoty l, který rozdělí buňky na silně nebo řídce zahuštěné. Metoda začíná na nejnižší úrovni a iterativně spojuje k-dimenzionální silně zahuštěné buňky c_1 a c_2. Vznikne nová $(k+1)$-dimenzionální buňka c, která je na základě prahu l označena za hustou nebo řídkou. Iterativní proces končí v momentě, kdy už nemáme k dispozici dimenze na slučování nebo nevznikají žádné silně zahuštěné buňky.

Druhým krokem je vytvoření samotných shluků v každém podprostoru na základě hustých buněk. Smyslem je najít co největší oblasti, které pokrývají všechny spojené husté buňky. Oblast je maximální hyperobdelník, jenž obsahuje pouze husté buňky a už nemůže být rozšířen v žádné dimenzi daného podprostoru. Nalezení shluku je obecně NP těžký problém. Metoda CLIQUE ale využívá *greedy* přístup, takže je složitost značně zefektivněna. Zvolí se libovolná hustá buňka, pro kterou se nalezne maximální oblast pokrývající tuto buňku a následně se oblast rozšiřuje o husté buňky, které ještě nebyly pokryty [14]. Ilustrativní příklad metody na třech dimenzí znázorňuje obrázek 2.8.



Obrázek 2.8: Proces tvorby shluku metodou CLIQUE pomocí hustých buněk v jednotlivých podprostorech vytvořením hyperobdelníku pro každý podprostor [14].

2.2.5 Metody založené na modelech

Metody v této kategorii se snaží namapovat vstupní data na matematický model. Oproti klasickému shlukování metody naleznou kromě identifikace shluků také charakteristiku shluku a samotný shluk představuje konceptuální třídu. Nejčastěji se používají modely rozhodovacích stromů a neuronových sítí.

Rozhodovací stromy

V rozhodovacích stromech jsou data reprezentována hierarchickým stromem, kde každý list představuje konceptuální třídu obsahující pravděpodobnostní charakteristiku dané třídy.

COWEB je nejznámějším algoritmem generující rozhodovací stromy pro neoanotovaná data. Algoritmus předpokládá, že všechny atributy jsou nezávislé (často nepravdivé). Cílem je dosáhnout vysoké předvídatelnosti hodnoty nominálního atributu pro daný shluk. Existuje i varianta pro numerické atributy. Algoritmus není vhodný pro velmi početná data [28].

Neuronové sítě

Přístup spočívá v reprezentování každého shluku neuronem nebo "prototypem". Vstupní data jsou také modelována jako množina neuronů, které jsou propojeny s prototypovými neurony. Každé spojení neuronu (datového objektu) s prototypovým neuronem (shlukem) má svou váhu, která je adaptivně optimalizována po čas trénování.

Populární neuronovým algoritmem jsou Kohonenovy mapy (SOM). Algoritmus vytváří jednovrstvou neuronovou síť. Trénovací proces spočívá v soupeření o jednotlivé datové objekty následujícím způsobem:

1) V soupeření o vstupní datový objekt se vítězem stává neuron, jehož vektor vah vykazuje nejbližší podobnost s konkrétním datovým objektem.

2) Vítězný neuron a jeho sousedi si následně upraví svoje vektory vah.

Algoritmus je užitečný také při vizualizaci vysoce dimenzionálních dat ve 2D nebo 3D prostoru. Na druhou stranu je citlivý na nastavení inicializačních vah a parametrů jako je učící koeficient nebo radius určující sousedy [28].

Pravděpodobnostní model

K datovým objektům se chováme jako k prvkům skrytých zatím neoanotovaných kategorií. Skrytá kategorie je matematicky reprezentována jako hustota rozdělení pravděpodobnosti nebo distribuční funkce. Každá skrytá kategorie je nazývána jako pravděpodobnostní shluk. Mějme pravděpodobnostní shluk C, jeho hustotu rozdělení f a bod o v datovém prostoru. Pravděpodobnost, že instance shluku C se nachází v bodě o, je vyjádřena jako $f(o)$. Příslušnost každého datového objektu ke shluku tedy není absolutní, ale je vyjádřena pomocí pravděpodobnosti. Běžným předpokladem je, že data daného shluku mají normální rozdělení. Množina shluků o velikosti k je následně modelována jako směs normálních rozdělení s parametry μ_j, σ_j $(1 \leq j \leq k)$ a vektor vah pro jednotlivé shluky $\omega_1, \omega_2, ..., \omega_k$. Aby uvedený model odpovídal co nejlépe vstupním datům, je proveden odhad maximální věrohodnosti pro zmíněné parametry [14].

Pro výpočet parametrů je použit algoritmus EM (*Expectation-Maximization*). Algoritmus má na vstupu inicializační vektor parametrů. Dále se opakují dva kroky tak dlouho, dokud se nepřestane zlepšovat věrohodnost modelu nebo zlepšení už je zanedbatelné. Zmíněné dva kroky vypadají obecně následovně:

- *Expectation step* - přiřazení objektů do shluků na základě parametrů pravděpodobnostních shluků.
- *Maximization step* - nalezení nových parametrů, které maximalizují věrohodnost daného modelu.

2.2.6 Určení počtu shluků

Důležitým krokem je zvolení počtu shluků, jenž je parametr, který musí být definován uživatelem u většiny shlukovacích metod. Proto budou uvedeny některé postupy, které pomůžou získat správný počet shluků. Metody jsou rozděleny do dvou kategorií. První kategorie se dívá pouze na kompaktnost uvnitř shluku (*intra-cluster*) a druhá kategorie bere v potaz i vzdálenost mezi shluky nebo-li separovatelnost (*inter-cluster*) [28].

Metriky založené na kompaktnosti shluku

Běžná metoda zkoumající kompaktnost uvnitř shluku je založena na poklesu metriky W_k (součet průměrných vzdáleností uvnitř shluku) pro jednotlivá K [28]. Zvolíme počet shluků K a pomocí vztahu 2.3 spočítáme metriku W_k

$$W_k = \sum_{k=1}^{K} \frac{1}{2N_k} D_k, \tag{2.3}$$

kde D_k 2.4 je vzdálenost mezi všemi dvojicemi daného shluku k.

$$D_k = \sum_{x_i, x_j \in C_k}^{K} \tag{2.4}$$

Následně danou metriku sledujeme s ohledem na měnící se K. Zpočátku metrika rapidně klesá a od určité hodnoty K se strmost křivky zmírní. Hodnota K v místě narovnání křivky je považována za vhodný počet shluků.

Existuje více metrik, u kterých můžeme sledovat jejich vývoj s ohledem na parametr K. Jednou z dalších metrik je suma čtverců (součet druhých mocnin euklidovských vzdáleností mezi datovým objektem a středem shluku), kterou se snažíme minimalizovat. Sumu čtverců využívá například metoda PRE (*Proportional Reduction in Error*), která je založena na poměru snížení sumy čtverců při porovnání shluků o velikosti K a $K + 1$. Počet shluků je navýšen jen v případě, že poměr snížení je větší nebo roven 0.4. Existují i složitější metody využívající sumu čtverců, které jsou postaveny na statistických základech [28].

Metriky založené na kompaktnosti i separovatelnosti shluků

Výše uvedené metody neberou v úvahu vzdálenost mezi shluky. Sledováním separovatelnosti shluků můžeme předejít rozdělení jednoho velkého shluku na další podshluky. Cílem je minimalizovat vzdálenost uvnitř shluku a naopak maximalizovat vzdálenost mezi shluky. Můžeme definovat metriku, která vyjadřuje poměr mezi rozptylem uvnitř shluku a rozptylem mezi shluky. Tento poměr se snažíme následně minimalizovat (minimalizovat čitatele a maximalizovat dělitele).

Dalším způsobem jak můžeme zjistit optimální K je vytvoření *validity* indexu [28], pro jehož výpočet potřebujeme metriky MD_k (průměrná vzdálenost uvnitř shluku k) a d_{min} (minimální vzdálenost mezi dvěma shluky). Rovnice 2.5 a 2.6 popisují jejich výpočet (μ_k je střední hodnota shluku k).

$$MD_k = \sum_{x_i \in C_k} \frac{||x_i - \mu_k||}{N_k} \tag{2.5}$$

$$d_{min} = min_{i \neq j} ||\mu_i - \mu_j|| \tag{2.6}$$

Pro vypočítání indexu je potřeba prozkoumat interval pravděpodobných hodnot K. Definujme K_{best} jako nejvhodnější počet shluků. Pokud mají data potenciál se dále dělit ($K < K_{best}$), tak alespoň jeden shluk bude mít vysoké MD_k. Na druhou stranu data rozdělená na příliš mnoho shluků ($K > K_{best}$) zapříčiní prudký pokles MD_k. Metrika d_{min} je vysoká pro ($K < K_{best}$) a naopak je velmi malá v situaci ($K > K_{best}$).

Následně pro každou hodnotu $K(2 \leq K \leq K_{max})$ spočítáme vektory hodnot pro metriky MD_k a d_{min}.

$$V_{MDK} = [v_{mdk}(2), ..., v_{mdk}(K_{max})]$$
$$V_{DMIN} = [v_{dmin}(2), ..., v_{dmin}(K_{max})]$$

Jednotlivé prvky vektoru V_{MDK} spočítáme vztahem 2.7 a prvky V_{DMIN} vztahem 2.8.

$$v_{mdk}(K) = \frac{\sum_{k=1}^{K} MD_K}{K} \tag{2.7}$$

$$v_{dmin}(K) = \frac{K}{d_{min}} \tag{2.8}$$

Finální *validity* index je získán jako součet normalizovaných vektorů V_{MDK} a V_{DMIN}. Využívá se znalosti, že vektory mají malé hodnoty pouze při $K = K_{best}$. Jako optimální počet shluků je zvoleno K, které se nachází jako parametr u minimálního prvku vektoru *validity* indexu [28].

2.3 Redukce dimenzí

Mezi hlavní důvody redukce dimenzionality patří snížení výpočetní a paměťové náročnosti při tvorbě modelu. Metody pro trénování modelu jsou více robustní na méně dimenzionální data, s redukovaným prostorem atributů lze lépe pochopit podstatu vstupních dat, což vede na snadnější získání znalostí nebo lepší možnost vizualizace dat. Redukcí ale přijdeme o interpretovatelnost původních dat [1].

Existují dva přístupy redukce. První spočívá pouze ve výběru podmnožiny atributů. Vyberme pouze k dimenzí z celkového počtu d, které dávají největší informaci a zbylé $(d - k)$ dimenze nepoužijeme [1].

Druhý přístup je založený na extrakci nové množiny atributů s dimenzí k, která je vytvořena kombinací původních d dimenzí. Existuje více metod na extrakci atributů. Nejznámější jsou metody založené na lineární projekci prostoru. PCA (*principal component analysis*) je metoda bez učitele a LDA (*linear discriminant analysis*) je metoda s učitelem [1]. Existují i nelineární metody redukce dimenzí, ale ty nebudou součástí této práce. Pro naše účely bude popsána pouze metoda PCA, jelikož zkoumaná problematika vyžaduje učení bez učitele.

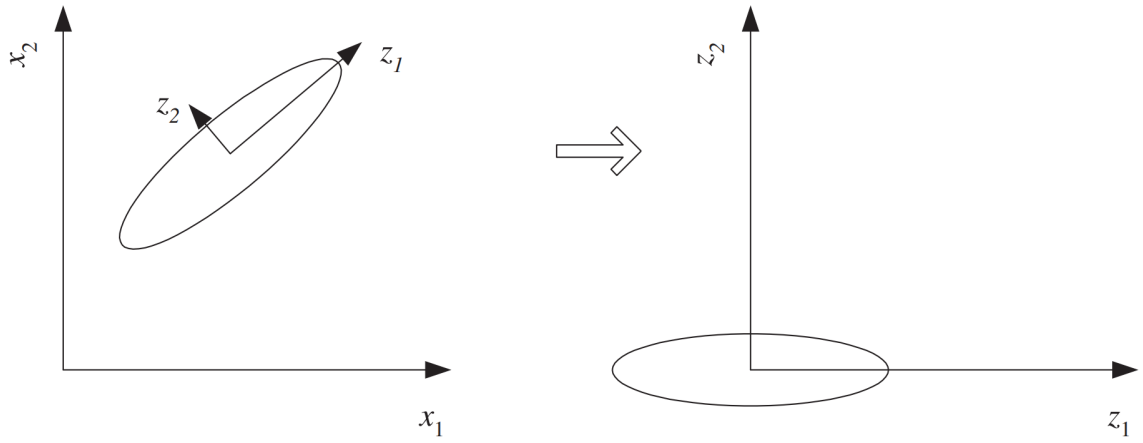
2.3.1 PCA

Metoda se snaží najít mapování z d-dimenzionálního prostoru nového $(k < d)$-dimenzionálního prostoru s minimální ztrátou informace. Snahou je maximalizovat rozptyl. Hlavní komponenta w_1 vykazuje při projekci dat na danou komponentu největší rozptyl a datové objekty jsou potom dobře separovatelné. Hlavní komponenty budou tvořit zmenšený k-dimenzionální prostor. Hlavní komponenta w_1 s největším rozptylem je reprezentována jako vlastní vektor kovarianční matice vstupních dat s nejvyšší vlastní hodnotou (vlastní hodnota odpovídá rozptylu). Jako další hlavní komponenty jsou vybrány vlastní vektory s postupně snižujícími se vlastními hodnotami (rozptylem).

Samotné mapování do menšího prostoru dimenzí je provedeno za pomocí rovnice 2.9

$$z = W^T(x - m), \tag{2.9}$$

kde k sloupců matice W jsou jednotlivé vlastní vektory kovarianční matice vstupních dat seřazené podle velikosti vlastních hodnot. Každý datový objekt je vycentrován odečtením od datového objektu x střední hodnotu m. Po provedení lineární transformace získáme k-dimenzionální prostor (viz obrázek 2.9) s dimenzemi vlastních vektorů a rozptylem odpovídajícím vlastním hodnotám [1].



Obrázek 2.9: Proces redukce dimenzí metodou PCA vycentruje vstupní data (odečtením střední hodnoty) a následně provede rotaci os ve směru největšího rozptylu. Dimenze s nízkým rozptylem můžeme odstranit, a docílíme tak redukce dimenzí [1].

Kapitola 3

Kvalita a modifikace datových sad

Úkolem kapitoly je rozebrat existující řešení společně s teoretickými podklady pro úspěšné vybudování modifikátoru datových sad. Datové sady jsou běžně modifikovány, ale v rámci redukce dimenzí atributů (například [31]). Článek [6] představil různé typy redundance dat v oblasti neuronových sítí. Redundantní data mohou být nalezena za pomocí vzdálenosti v prostoru atributů. Nejčastější příčinou modifikace datových sad v kontextu datových objektů je problém nevyvážených datových sad [3]. Existující modifikační přístupy 3.1 vznikly primárně za účelem vyvážení poměrů jednotlivých tříd v datové sadě. Vyvážení datové sady probíhá formou pod-vzorkování majoritní třídy nebo nad-vzorkování minoritní třídy. V kapitole je dále popsána problematika ohodnocování datových sad (viz sekce 3.2) a v neposlední řadě jsou rozepsány možné způsoby, jak získat nové data pro modifikaci datové sady. Zdrojem nových dat můžou být reálná data ze sítě 3.3 nebo se vygenerují vhodným generativním způsobem 3.4.

3.1 Vyvažování datových sad

K modifikování datových sad dochází často v souvislosti s nevyváženou datovou sadou. Nevyvážené datové sady vedou na klasifikátor, který se často není schopen naučit rozpoznávat minoritní třídu. Dochází i ke zkreslení některých metrik (např. celková úspěšnost), které při výpočtu neberou ohled na nevyváženost tříd [3]. Navržený přístup v rámci práce se liší oproti vyvažování v tom, že datová sada je modifikována jako celek a neřeší prioritně nevyváženost dat. Snahou je získat minimální datovou sadu, která nebude obsahovat velké množství redundantních data a případně přidat nová data, která přinesou nové informace do trénovacího procesu.

V rámci knihovny pro strojové učení a datovou analýzu `Scikit-learn` byly implementovány metody pro vyvážení datových sad [21]. Framework `Imbalanced-learn`[1] implementuje nejpoužívanější metody, které byly představeny pro účely vyvážení dat. Metody jsou rozděleny do 4 kategorií (pod-vzorkování, nad-vzorkování, kombinované, skupinové učení). Důvodem pro vytvoření frameworku byl přirozený výskyt datových sad, které mají jednu třídu zastoupenou méně než ostatní třídy. Problém nevyváženosti dat se vyskytuje v mnoha odvětvích jako je telekomunikace, bioinformatika, detekce hrozeb nebo zdravotnictví [21]. Implementace je založená kromě `Scikit-learn` na knihovnách `Numpy` a `Scipy`. Nevyvážená

[1] https://imbalanced-learn.org/stable/index.html

datová sada má poměr vyváženosti R_D [21] definován pomocí vzorce 3.1

$$R_D = \frac{|D_{min}|}{|D_{maj}|}, \tag{3.1}$$

kde D_{min} je množina zastupující minoritní třídu a D_{maj} naopak zastupuje majoritní třídu datové sady D. Cílem je převzorkovat datovou sadu D, tak aby výsledná datová sada D_S měla vyšší poměr vyváženosti ($R_D < R_{D_S}$).

Nyní budou rozepsány zmíněné 4 kategorie [21] a ke každé skupině bude uveden základní princip spolu s referencemi na konkrétní metody.

- **Pod-vzorkování** - Úkolem je zredukovat četnost majoritní množiny D_{maj}. Metody můžeme rozdělit na dva přístupy. První přístup umožňuje definovat výsledný poměr vyváženosti R_{D_S}. Na druhou stranu existují metody, u kterých není možné poměr vyváženosti stanovit. Pod-vzorkování probíhá na základě čištění prostoru atributů podle empirických metrik. Tabulka 3.1 uvádí konkrétní metody se stručným popisem.
- **Nad-vzorkování** - Druhým případem je naopak snahou generovat data do minoritní množiny D_{min} pro dosažení vyššího koeficientu vyváženosti R_{D_S}. V tabulce 3.3 jsou popsány různé varianty nad-vzorkovacích metod.
- **Kombinované** - Nad-vzorkováním můžeme dosáhnout špatné generalizace modelu, proto je vhodné následně aplikovat čistící pod-vzorkovací metody.
- **Skupinové učení** - V momentě, kdy pod-vzorkujeme majoritní množinu D_{maj}, tak přijdeme o podstatnou část trénovacích dat. Skupinové učení přichází s alternativou, jak při trénování využít maximum vzorků. Výsledkem je množina vyvážených datových sad, které jsou následně využity k finálnímu natrénování modelu.

Tabulka 3.1: Metody na pod-vzorkování datové sady

Metoda	**Popis**
ClusterCentroids [8] (lze definovat R_{D_S})	Každá třída je reprezentována synteticky vygenerovanými vzorky, které byly vytvořeny aplikováním shlukovacího algoritmu. Původní data jsou nahrazena novými vzorky, které vznikly jako středy jednotlivých shluků v dané třídě.
RandomUnderSampler [8] (lze definovat R_{D_S})	Náhodně je vybrána podmnožina redukované třídy. Varianta s navracením nebo bez navracení.

NearMiss [23] (lze definovat R_{D_S})	Vzorky dat jsou vybírány na základě heuristik využívající k-nejbližších sousedů. Existují tři druhy heuristik. První heuristika vybírá vzorky s nejmenší průměrnou vzdáleností k-nejbližších sousedů druhé třídy. Druhá heuristika porovnává naopak k-nejvzdálenější sousedy a opět vybírá vzorky s nejmenší průměrnou vzdáleností. Třetí metoda vyfiltruje vzorky na hranici s druhou třídou a z nich následně vybírá vzorky s největší průměrnou vzdáleností k-nejbližších sousedů.
TomekLinks [18] (nelze definovat R_{D_S})	*Tomek link* existuje mezi dvěma vzorky z různých tříd, pokud jsou navzájem nejbližšími sousedy. Nadále se odstraňují vzorky mající tuto vazbu.
ENN [37] (nelze definovat R_{D_S})	V metodě *EditedNearestNeighbours* musí každý vzorek splňovat kritérium, aby byl ponechán v datové sadě. První varianta požaduje, aby všech k-nejbližších sousedů patřilo do stejné třídy. Druhé variantě pak k ponechání vzorku stačí většina.
RepeatedENN [35] (nelze definovat R_{D_S})	Metoda provádí opakovaně *EditedNearestNeighbours*, což vyústí ve vyšší redukci dat.
AllKNN [35] (nelze definovat R_{D_S})	Jedná se o rozšíření předchozí metody, kdy se iterativně zvětšuje počet sousedů, pro které musí platit zvolené kritérium.
CondensedNearestNeighbour [15] (nelze definovat R_{D_S})	Třída určená k redukci se prochází iterativně. Prvek je vybrán, pokud jeho nejbližší soused patří do druhé třídy. Vybraný prvek se z redukované množiny odstraní a proces se opakuje do doby, dokud existuje prvek, který může být vybrán. Jinými slovy se metoda snaží zanechat vzorky, které se překrývají s protichůdnou třídou.
OneSidedSelection [15] (nelze definovat R_{D_S})	Předchozí metoda je citlivá na šum, a proto *OneSidedSelection* aplikuje *Tomek link* pro odstranění šumu. Navíc se provádí pouze jeden průchod redukovanou třídou.
NeighbourhoodCleaningRule [20] (nelze definovat R_{D_S})	Snahou je vyčistit prostor atributů provedením sjednocením dvou množin. První množina jsou prvky, které odmítla metoda *EditedNearestNeighbours* a druhá množina jsou špatně klasifikované prvky 3-NN modelem. Vybrané prvky se nacházejí na hranici s jinými třídami.

InstanceHardnessThreshold [32] (lze definovat R_{D_S})	Přístup využívá klasifikátoru natrénovaného na datech, který je schopen predikovat vzorek s určitou pravděpodobností. Odstraněny jsou vzorky s nižšími pravděpodobnostmi.

Tabulka 3.3: Metody pro nad-vzorkování datové sady

Metoda	**Popis**
RandomOverSampler [24]	Nejprimitivnějším způsobem jak nad-vzorkovat data je využít náhodného vzorkování s navracením, což vede na duplicitu vzorků. Existuje varianta, která umožní odstranit duplicitu tím, že pro nové vzorky provede rozptýlení hodnot atributů, které ovšem musí být numerické.
SMOTE [4]	Nové vzorky jsou vytvářeny za pomocí interpolace. Při interpolaci se berou do úvahy všechny dostupné vzorky v rozšiřované třídě.
ADASYN [16]	Na rozdíl od *SMOTE* se provádí interpolace pouze ze vzorků, které jsou špatně klasifikovány modelem k-nejbližších sousedů.
BorderlineSMOTE [13]	Cílem rozšiřujících metod je získání optimální klasifikační hranice mezi třídami. *BorderlineSMOTE* provádí interpolaci jen na vzorcích, které se nacházejí v oblasti sousedící s jinou třídou.
SVMSMOTE [25]	Pro výběr vzorků, které budou použity k interpolování, je použit algoritmus *Support Vector Machine*.
KMeansSMOTE [10]	Před samotným nad-vzorkováním se na minoritní třídu aplikuje Kmeans algoritmus, která rozdělí data na shluky. Interpolace následně probíhá ze středů jednotlivých shluků. Nově vytvořené vzorky lépe zachovávají distribuci dat.
SMOTENC [5]	Pro aplikování nad-vzorkování na data, která obsahují jak numerické, tak kategorické atributy. Interpolují se numerické atributy a kategorické zůstávají stejné.

SMOTEN [5]	*SMOTEN* metoda je použita v případě, že data se skládají jen z kategorických atributů. Při hledání nejbližšího souseda není použita Euklidova vzdálenost, ale VDM (*Value Difference Metric*). Metrika pracuje s rozdílem pravděpodobností výskytu porovnávaných hodnot v kategorickém atributu. Interpolovaná hodnota nového vzorku je dána majoritní hodnotou nejbližších sousedů ve stejné třídě.

3.2 Ohodnocování datových sad

Ohodnocování datových sad je složitá problematika, ke které je přistupováno z více úhlů pohledu. Obecně se u dat sleduje množina dimenzí, pro které je vyhodnoceno, zda data splňují požadavky jednotlivých dimenzí. Mezi dimenze patří například úplnost dat, aktuálnost, redukce duplicit, výběr atributů a mnoho dalších [11]. Dimenze definují jak s daty zacházet, aby byla dosažena kvalitní množina dat. Uvedený proces ohodnocování dat je chápán na vyšší úrovni.

Na nižší úrovni by bylo ideální mít metodu, která by vyčíslila kvalitu datové sady. Snahou je nalézt indikátor kvality, podle kterého by se porovnávaly datové sady. Porovnáním datových sad by se určilo, zda je datová sada úplná nebo je vhodné ji rozšířit. Dalším přínosem by bylo vyhodnocení, která ze dvou datových sad je lepší a případně by se provedlo spojení sad za účelem získání lepší trénovací množiny. Článek [33] se snaží docílit navržení frameworku, který by umožnil získat indikátor kvality datové sady a na jeho základě provádět zmíněné operace. V následující sekci bude nastíněno fungování a princip navržené metody.

V první fázi je důležité provést definici pojmů, se kterými se bude pracovat při samotném ohodnocování. Metoda je obecná a použitelná na libovolná vstupní data, tudíž i samotné pojmy jsou obecně definované a slouží výhradně pro sjednocení terminologie.

1. **Datová sada** - Nutnou součástí trénování modelu. Datová sada je kolekcí vstupních dat. Je složena z datových položek udávající formát. Formát je specifický pro každou doménu. Datová položka se skládá z jednotlivých atributů a v případě učení s učitelem je k dispozici i její anotace. Formát datové položky není stanoven.

   Proces trénování modelu zahrnuje rozdělení datové sady na trénovací a testovací část. Obě části mají velký dopad na kvalitu výsledného modelu. Jelikož existují různé typy algoritmů [26], které vyžadují jiné postupy i různá data na vstupu. Výstupem transformace dat je atributová sada.

2. **Dobrá datová sada** - Jelikož zatím neexistuje univerzální metrika pro ohodnocení sady [19], tak se kvalita datové sady většinou odvíjí od reputace autorů. V síťové oblasti se vše rychle mění a je složité recyklovat veřejnou datovou sadu a aplikovat ji v jiném prostředí. Cílem je zjistit, zda datová sada je vhodná pro konkrétní případ užití. Dobrá datová sada obsahuje anotace u všech datových položek. Množství dat je přijatelné pro natrénování modelu. Data uspokojují požadavky dimenzí jako

je správnost, úplnost, konzistence, věrohodnost a čerstvost. Pokud data nejsou úplná a spolehlivá, tak nemůžeme považovat danou datovou sadu za dobrou. Splnění uvedených požadavků často vyžaduje subjektivní pohled, z důvodu nepřesných definic. Autoři [33] zavedli přesnější pravděpodobnostní definici.

Pro konkrétní doménu existuje definovaná datová sada D. Díky monitorovací sondě je možné zachytit požadované atributy a anotace, a tím zaručit jejich věrohodnost. Kvalita datové sady je spočítána jako pravděpodobnostní vektor úplnosti C_k a spolehlivosti R_k:

$$C_k = P(x \in D \mid x = L_k); \\ \forall k \; in \; \{0, 1, ..., dim(L)\}) \tag{3.2}$$

$$R_k = P(x = L_k \wedge \; d(x) = L_k \mid x \in D); \\ \forall k \; in \; \{0, 1, ..., dim(L)\}) \tag{3.3}$$

kde L je vektor anotací, $dim(L)$ je počet anotací, x je datová položka a $d(x)$ je anotace položky x v datové sadě D. Vektor C ukazuje pravděpodobnost, jestli je datová položka zahrnuta v datové sadě, když se vyskytuje v cílové doméně. Kvůli nerovnoměrnému zastoupení jednotlivých anotací se pravděpodobnost počítá pro každou anotaci zvlášť. Obdobně vektor R reprezentuje pravděpodobnost chybně oanotované datové položky. Vektor C zahrnuje požadavky úplnosti a čerstvosti a vektor R zase pokrývá správnost a konzistenci. Finální indikátor kvality datové sady je vyčíslen jako harmonický průměr mezi dvěma vektory:

$$D_q = \frac{2 \cdot mean(C) \cdot mean(R)}{mean(C) + mean(R)} \tag{3.4}$$

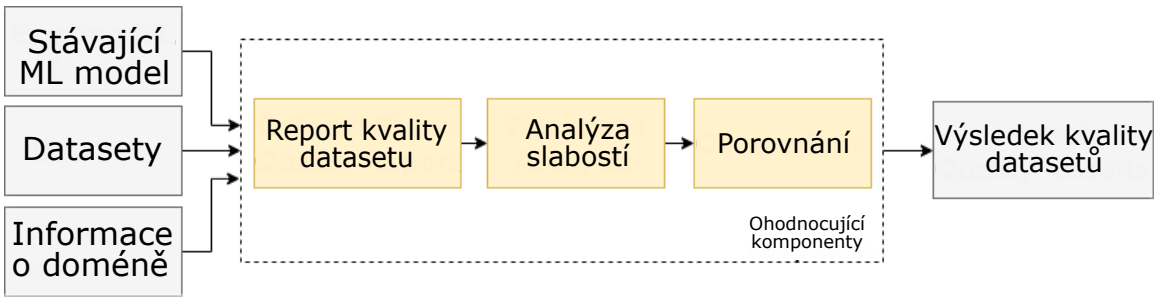
Přestože se podařilo získat přesnější definici kvality datové sady, tak provést vyčíslení je složité. Pro správné vyčíslení je nutné neustále monitorovat aplikační doménu a mít spolehlivý prostředek na anotaci dat. V reálném prostředí nemůžeme takové podmínky splnit, a proto definici pouze aproximujeme s pomocí menších datových dat v offline prostředí nebo specificky vygenerovaných dat.

3. **Lepší datová sada** - Pro žádný problém řešený strojovým učením neexistuje pouze jedna perfektní datová sada. Důvodem, proč chceme porovnávat datové sady, je zjištění, která se pro daný problém více hodí nebo zda je výhodné sady částečně nebo úplně spojit. Aby bylo možné rozhodnout, která datová sada je lepší, je potřeba cílovou doménu jednoznačně identifikovat. K identifikaci je potřeba datová sada, informace o doméně a klasifikační model. Datová sada A je označena za lepší, pokud je označena za dobrou datovou sadu a zároveň dosahuje většího skóre kvality než datová sada B pro danou doménu.

4. **Minimální datová sada** - Každá datová sada zahrnuje minimální datovou sadu rozšířenou o další datové položky. Položky navíc přidávají už zahrnuté skupiny dat. Minimální datová sada neobsahuje žádné duplicity a odstraněním jakéhokoliv záznamu bychom ztratili úplnost. Pojem minimální datové sady není nový a je použit například v článku [9].

3.2.1 Ohodnocující framework

Cílem je najít slabiny a ohodnotit kvalitu datové sady pro dané prostředí a doménu. Návrh frameworku je zobrazen na obrázku 3.1. Vstupem frameworku jsou datové sady, informace o doméně a cílový klasifikační model. V první fázi jsou datové sady staticky ověřeny na formát a zda splňují podmínky dobré datové sady. Fáze dva (*Weakness analysis*) zahrnuje dynamické ohodnocení datové sady. Na základě informace o vstupní doméně se vygenerují různé datové sady. Na závěr se provede porovnání mezi vstupními i vygenerovanými datovými sadami za účelem identifikovat lepší datovou sadu. Výsledkem je report kvality a další doporučení.



Obrázek 3.1: Struktura ohodnocujícího frameworku [33].

V rámci experimentů byly analyzovány datové sady v DGA (*Domain generation algortihm*) doméně. Nad datovou sadou bylo zjištěno zastoupení jednotlivých tříd, což pomohlo k identifikaci dobré datové sady. V rámci dynamické fáze byly vygenerovány datové sady pomocí genetického algoritmu způsobem uvedeným zde 3.4. Cílem bylo najít minimální nebo lepší datovou sadu. Minimální datová sada je tvořena záznamy, na kterých klasifikátor dosahuje nejlepší úspěšnosti. Lepší datová sada je vytvořena za pomocí sloučení původní datové sady a sady, která dosahovala nejhorší úspěšnosti.

Nalezení lepší datové sady umožní natrénovat model, který dosahuje vyšší úspěšností než původní model. Minimální sada zkrátí trénovací čas a nároky na paměť. Celý proces ohodnocování datových sad je stále v rané fázi vývoje a vyžaduje další zkoumání a experimenty [33].

3.3 Zdroj síťových dat

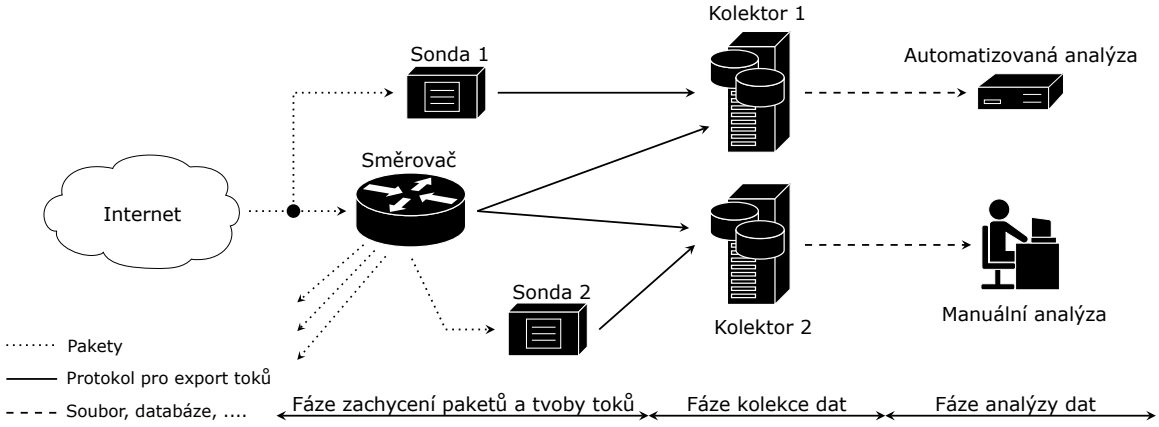
Po síti proudí obrovské množství dat a cílem je tyto data sbírat za účelem další analýzy. Abychom mohli modifikovat datovou sadu, je zapotřebí mít dobrý zdroj dat. Jedním z dobrých zdrojů dat může být právě kolekce reálných dat. Za poslední roky bylo představeno velké množství způsobů monitorování s různým záměrem. Metody můžeme rozdělit na aktivní a pasivní.

Aktivní přístup přímo zasahuje do síťového provozu různými dotazy (například nástroj *Ping*) za účelem získání informací. Pasivní přístup nevytváří žádný síťový provoz, ale pouze sleduje provoz uživatelů v měřícím místě (sondě). Za pasivní přístup můžeme považovat zachytávání paketů. Sledování konkrétních paketů sice umožní detailní pohled na provoz, ale ve vysokorychlostních sítích je tento způsob značně nepoužitelný z důvodu drahého hardwaru a složité infrastruktury. Dostupnějším a více škálovatelným přístupem je agregování paketů do toků, které se následně ukládají. Tok je definován jako množina IP paketů procházející měřícím bodem sítě v časovém intervalu mající stejné vlastnosti [7]. Za stejné vlastnosti lze

považovat hodnoty v hlavičce paketu, jako je cílová a zdrojová IP adresa, čísla portů, obsah paketu nebo meta informace. Monitorování na základě exportování toků vedlo k masovému rozšíření a ke vzniku současných protokolů jako je *NetFlow* a IPFIX (*IP Flow Information eXport*) [17]. Následující část bude popisovat obecné principy exportu toků.

3.3.1 Architektura monitorování toků

Architektura monitorování toků je rozdělena do několika fází. Grafické znázornění architektury s jednotlivými fázemi je zobrazeno na obrázku 3.2. První fáze je zpracování paketu, kdy je paket zachycen a následně podroben předzpracování. Měřící body jsou nasazeny jako dedikované karty na lince nebo jsou součástí rozhraní směrovače. Druhá fáze obnáší tvorbu samotného toku agregací paketů se stejnými vlastnostmi. V momentě, kdy je tok považovaný za ukončený, je exportovaný za pomocí zvoleného protokolu. Tok je popsán společnými vlastnostmi toku. Třetí fáze má za úkol předzpracovat a uložit obdržené toky z předchozí fáze. Poslední fáze provádí analýzu daných toků z hlediska korelace, agregace, profilování nebo detekce anomálií. Často může být součástí kolekce dat [17]. Fáze analýzy v kontextu této práce bude obnášet shlukování nasbíraných dat.



Obrázek 3.2: Architektura monitorování toku [17].

Zpracování paketu

Zpracování paketu je klíčová část při monitorování toků. Paket je zachycen síťovou kartou, kde je provedena kontrola chyb a následně je paketu přiřazeno časové razítko. Velmi důležitá je synchronizace času napříč měřícími body, jelikož se při tvorbě datové sady řadí data právě podle časového razítka. Kontrola chyb a tvorba časových razítek jsou povinné kroky. Mezi volitelné kroky patří ořezání paketu pro nižší výpočetní složitost. Často se ponechávají pouze hlavičky paketů a datová část je ořezána. Poslední krokem předzpracování paketu je vzorkování a filtrování. Cílem vzorkování je nalézt takovou podmnožinu paketů a zachovat vlastnosti původní množiny. Pakety můžou být vzorkovány systematicky, kdy zachytíme každý n-tý paket, nebo můžeme využít náhodného vzorkování. Náhodné vzorkování je obecně lepší řešení, jelikož není ovlivněno periodickým chováním. Filtrováním můžeme odstínit pakety s nepožadovanými vlastnostmi. Pakety jsou porovnávány buď na základě rozsahu hodnot polí v paketu, nebo pomocí hodnoty *hashe* paketu. Vzorkování a filtrování slouží také k redukci množství zpracovávaných dat [17].

Tvorba toků

Cílem je sestavit tok, jehož hodnoty jsou definovány pomocí IE (Informačních Elementů), které jsou standardizovány autoritou IANA[2]. Každý element má svoje unikátní ID. Nejčastěji se pro popis toku používají informace ze síťové a transportní vrstvy (IP adresy, čísla portů, ...), ale IE může být definován pro libovolnou vrstvu (např. aplikační). Množina informačních elementů exportována do kolektorů je vždy stanovena pomocí šablon. Informace o všech aktivních tocích jsou ukládány ve vyrovnávací paměti toků.

Vyrovnávací paměť ukládá informační elementy. IE mohou být klíčové nebo neklíčové. Klíčové elementy (např. IP adresa) rozlišují, zda daný paket patří k danému toku nebo je nutné vytvořit nový tok. Tok je běžně zaznamenáván bez ohledu na směr komunikace, ale existuje i varianta, kdy je tok rozdělen podle směru komunikace. Máme více druhů pamětí pro zaznamenávání toků. Když je tok reprezentován jedním paketem (při řídkém vzorkování), není nutné kontrolovat ukončení toku a tok může být hned exportován. Druhým případem je permanentní paměť, kde se ukládají toky, které nikdy nevyprší a neexportují se. Permanentní toky slouží k účtování komunikace.

Tok je považován za ukončený (připravený k exportu), pokud dojde k vypršení časovače nebo nastane konkrétní událost (FIN paket, plná paměť, ...). Aktivní časovač sleduje periodické a dlouho trvající toky. Po jeho vypršení nedojde k odstranění toku z paměti, ale pouze k jeho obnovení. Pasivní časovač odstraní po vypršení tok z paměti, pokud nebyl zachycen žádný paket, který by patřil k danému toku. Toky mohou být před exportem podrobeny opět vzorkování a filtrování obdobným způsobem jako u zpracování paketu. Toky připravené k exportaci jsou zakódovány do zvoleného protokolu (např. IPFIX) [7] a následně jsou libovolným transportním protokolem (SCTP, TCP nebo UDP) odeslány na kolektor [17].

Kolekce dat

Jakmile obdrží kolektor data od exportérů, provede nad nimi případnou kompresi dat, agregaci, anonymizaci, filtrování či vytvoří sumarizaci dat. Samotné předzpracování většinou probíhá ještě v operační paměti a až výsledná data jsou zapsána do permanentní paměti. Používá se několik způsobů, jak uložit data do permanentní paměti.

Jedním ze způsobů je souborové úložiště (např. *nfdump*), které je velmi rychlé, ale neefektivní pro dotazování. Data jsou ukládány v textové nebo binární podobě. Dále se používají řádkové databáze (*MySQL*), které umí snáze zpracovávat dotazy, ale stále čteme celé řádky, což není vždy optimální. Poslední možnost je použít sloupcové databáze (*FastBit*), které umí při dotazování efektivně načíst pouze relevantní sloupec, což vede na vysokou efektivitu [17]. Z kolektoru si následně můžeme vytáhnout potřebná data pro další analýzu. V rámci práce budou nasbíraná data sloužit pro modifikaci datových sad na základě shlukové analýzy.

3.4 Generování datových sad

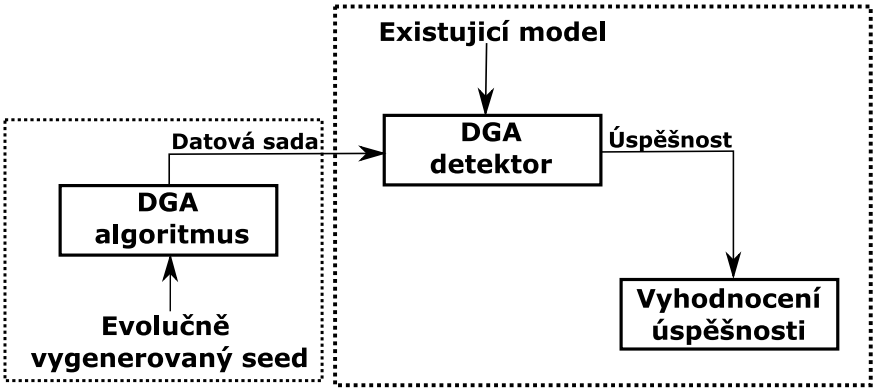
Hlavním důvodem pro generovaní datové sady algoritmem je získání nových datových objektů, které potenciálně přispějí k vyšší kvalitě datové sady. V momentě, kdy máme k dispozici kvalitní trénovací datovou sadu, tak na ní můžeme natrénovat přesný klasifikátor. Často se můžeme setkat s klasifikátory, které nemají přesné výsledky jen kvůli tomu, že byly natrénovány na nevhodné datové sadě. Generování dat je specifické pro každou do-

[2] https://www.iana.org/assignments/ipfix/ipfix.xhtml

ménu a často to může být problematický proces. Obecně není problémem vygenerovat data, ale najít k ním správnou anotaci. Pomocí algoritmů pro generování se snažíme vytvářet datové sady, na kterých dosahuje klasifikátor nízké úspěšnosti a na jejich základě vybudovat přesnější klasifikátor. Sekce je zaměřena konkrétně na generování dat pomocí genetického algoritmu v oblasti DGA (*Domain generation algorithm*), jelikož se bude pracovat s datovými sadami, které byly vygenerovány zmíněným způsobem. Další metody pro generování dat byly uvedeny formou nad-vzorkovacích metod v tabulce 3.3.

3.4.1 Generování DGA

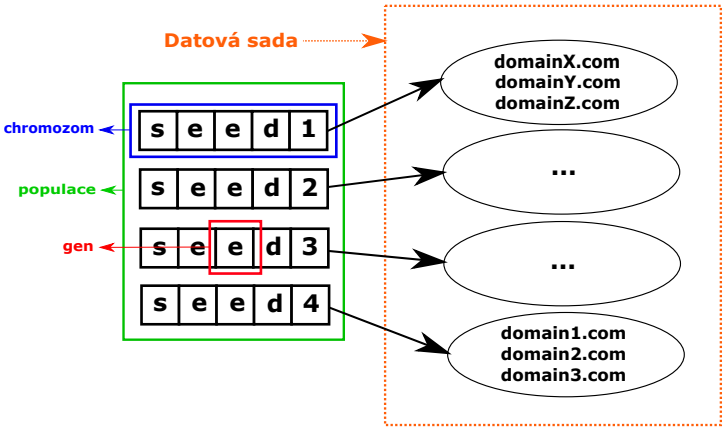
V našem případě při práci s DGA lze využitím umělých sad získat kvalitní datovou sadu zahrnující širokou škálu DGA algoritmů, a získat tak přesnější klasifikátor. Při generování datových sad genetickým algoritmem využíváme takzvaných seedů, podle kterých vytváří DGA algoritmus konkrétní doménová jména. Seed určuje jak bude výsledná datová sada vypadat. Zde přichází na řadu genetický algoritmus, pomocí kterého můžeme získat nejlepší seed a vytvořit tak dobrou datovou sadu. Obrázek 3.3 ukazuje proces, jakým je vytvářena datová sada pomocí DGA algoritmu a její následné ohodnocení. V kontextu genetického algoritmu má DGA detektor roli fitness funkce, která určuje úspěšnost vygenerované sady na referenčním klasifikátoru.



Obrázek 3.3: Generování DGA sad a jejich ohodnocení [34].

V následujícím textu bude stručně popsán postup, jak jsou tvořeny datové sady za pomocí genetického algoritmu. Na počátku je náhodně vygenerovaná populace, která nastartuje evoluční vývoj. V první fázi přichází na řadu ohodnocení populace. Jak bylo uvedeno výše, tak z populace složené ze seedů je vygenerovaná datová sada a ta je klasifikována. Klasifikaci provádí zmiňovaná fitness funkce, která určí kvalitu populace. Poté následuje selekce nejkvalitnějších jedinců (seedů). Existují různé přístupy jak vybrat nejlepšího jedince například *Roulette Wheel Selection*, *Tournament Selection*, *Rank Selection* nebo *Random Selection*. Po získání nejlepších jedinců je prováděno vzájemné křížení, za účelem vzniku dalších kvalitních jedinců. Opět máme různé možnosti křížení, které jsou například *One Point Crossover*, *Multi Point Crossover*, *Without Crossover*. Na závěr je provedena mutace (například náhodným prohozením znaků v seedu) jedinců, díky které můžeme s určitou pravděpodobností dosáhnout dalšího kvalitního jedince. Výsledná populace (nová generace) je poté složena jen z jedinců vzniklých křížením a mutací. Výše uvedený popis se nadále opakuje, dokud nejsme spokojeni s kvalitou populace [34].

Pro uchopení věcí do souvislostí je na obrázku 3.4 uvedena analogie mezi pojmy DGA terminologie a terminologie genetického algoritmu. Populace se skládá z chromozomů (seedů) a chromozomy obsahují gen, jenž reprezentují znaky. Výsledná datová sada se potom skládá ze skupin domén, které byly vygenerovány pomocí stejného seedu. Každý seed může být na vstupu jiného DGA algoritmu a rozdílné skupiny doménových jmen umožní diverzifikovat a zkvalitnit cílovou datovou sadu.



Obrázek 3.4: Analogie mezi DGA a genetickým algoritmem [34].

Kapitola 4

Návrh modifikátoru datových sad

Cílem práce je navrhnout framework, který umožní modifikovat datové sady. Fungování frameworku je rozděleno na několik částí. V první řadě je vyžadována schopnost shlukovat data v rámci jedné datové sady. Shlukování bude probíhat na základě vstupních atributů datové sady. Výsledkem shlukování je nalezení stejných nebo podobných skupin dat. Skupiny dat jsou následně statisticky popsány bez ohledu na konkrétní případ užití. Shluk je popsán a vyhodnocen pomocí vstupních charakteristik a navíc lze využít nástroje pro ohodnocování dat 3.2 nebo vyhodnotit shluk s přiloženým klasifikátorem a získat úspěšnost nad konkrétní shlukem dat. Získaný statistický popis a informace o původním klasifikátoru se stanou podkladem pro automatizovaný výpočet skóre pro jednotlivé shluky. Skóre se poté znormalizuje a při modifikační fázi bude sloužit jako váha, která říká v jakém poměru daný shluk upravit. Modifikace datové sady může nabývat redukční nebo selektivní fáze. V případě, že máme velkou datovou sadu, tak je snahou data redukovat za účelem získání kompaktnější datové sady, na které bude natrénovaný klasifikátor. Nově natrénovaný klasifikátor by měl dosahovat minimálně stejné úspěšnosti na referenčních datech jako původní a zároveň se zrychlí doba trénování a sníží se nároky na paměť. Další variantou je rozšířit datovou sadu o chybějící data. Nová data jsou opět analyzována a získané váhy jsou použity k selekci potřebných dat. Interním výstupem frameworku jsou váhy pro jednotlivé shluky, které jsou předány zvolenému modifikačnímu algoritmu, jenž provede redukci nebo selekci nových dat.

Strukturu modifikačního mechanismu můžeme vidět na obrázku 4.1. Navržený framework bude fungovat následujícím způsobem:

1. **Vstup** - Na vstupu bude analyzovaná datová sada, která se modifikuje za účelem získání minimální datové sady nebo lepší datové sady. Datová sada se na začátku rozdělí na jednotlivé třídy, které jsou postupně předány shlukovacímu algoritmu.

2. **Shlukovací algoritmus** - Uvnitř frameworku pracuje vhodný shlukovací algoritmus, který má za úkol rozdělit vstupní datovou sadu na jednotlivé shluky podobných vlastností.

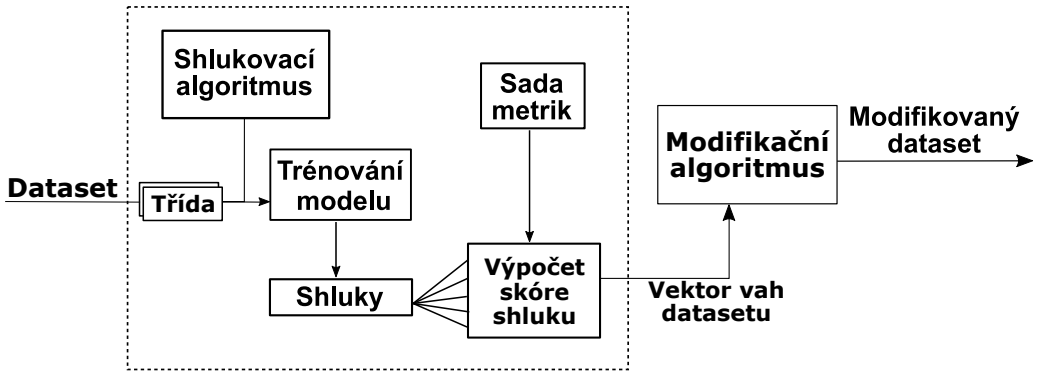
3. **Trénování modelu** - Po inicializaci a nastavení parametrů shlukovacího algoritmu se vytvoří model, který se natrénuje na jednotlivých třídách vstupní datové sady.

4. **Tvorba shluků** - Pomocí natrénovaného modelu rozdělíme vstupní data na jednotlivé shluky.

5. **Výpočet skóre** - Pro každý shluk se spočítá sada statistických metrik, jenž může být libovolně definována. Výsledné skóre shluku je odvozeno z definovaných metrik. Normalizované skóre všech shluků tvoří výsledný vektor vah a slouží jako podklad pro další modifikaci datové sady.

6. **Modifikační algoritmus** - Část modifikování je velice obecná a rozmanitá. V této fázi může dojít k libovolné transformaci datové sady. Lze sbírat data ze sítě pomocí IPFIX nebo Netflow (viz sekce 3.3) a filtrovat data na základě vygenerovaného vektoru vah. Další možností je využít syntetické tvorby datové sady např. pomocí metody genetického algoritmu (viz sekce 3.4). V neposlední řadě se dá provést křížové shlukování napříč více datovými sadami, jinými slovy se jedná o vyfiltrování požadovaných dat z různých datových sad.

7. **Výstup** - Na výstupu je modifikovaná datová sada, která má potenciálně pozitivní dopad na trénování výsledného klasifikačního modelu.



Obrázek 4.1: Struktura frameworku pro modifikaci datové sady.

4.1 Generování vektoru vah

Při generování vektoru vah půjde o identifikaci shluků, které jsou vhodné k rozšíření či redukci a v jaké míře. Při tvorbě vektoru vah se musí nejprve stanovit množina statistických metrik 4.1.1, které budou popisovat jednotlivé shluky. Metriky jsou následně analyzovány v sekci 4.1.2 a jsou vstupem pro automatizovaný výpočet skóre 4.1.3, jenž zahrnuje několik kritérií. Dle spočítaného skóre je možné přiřadit jednotlivým shlukům váhy, které budou určovat důležitost shluku pro trénování modelu. Na základě normalizovaného skóre je sestaven výsledný vektor vah, který číselně vytyčí, jaká skupina dat je vhodná pro rozšíření či naopak zredukování.

4.1.1 Množina metrik

Pro každý shluk se spočítá množina metrik, které mají charakterizovat daný shluk dat. Množina metrik se skládá ze statistických charakteristik, aby byly obecně aplikovatelné na všechny typy dat. Každá metrika je spočítána pro všechny vstupní atributy v rámci jednoho shluku. Během návrhu byly zvoleny následující metriky:

- **Průměr** - Aritmetický průměr bude sloužit při výpočtu dalších atributů. Střední hodnotu atributu a pro shluk s spočítáme pomocí vztahu 4.1:

$$\mu_{s,a} = \frac{\sum x_{a,s}}{N_s}, \tag{4.1}$$

kde $x_{a,s}$ je prvek atributu a ve shluku s a N_s je celkový počet prvků shluku.

- **Rozptyl** - Získáme přehled, jak jsou jednotlivé hodnoty atributů rozptýleny. Rozptyl shluku s a atributu a spočítáme pomocí vztahu 4.2:

$$\sigma^2_{s,a} = \frac{\sum x^2_{a,s}}{N} - \mu_{s,a}a^2, \tag{4.2}$$

kde $x_{a,s}$ je prvek atributu a ve shluku s, N_s je celkový počet prvků shluku a $\mu_{s,a}$ je průměr atributu a v daném shluku s.

- **Diference shluku** - Značí odchylku daného shluku od vstupní datové sady. Je vyjádřena procentuálně a říká o kolik se liší atribut a ve shluku s oproti hodnotám v původní datové sadě. Metrika je spočítána na základě průměru atributů původní datové sady a průměru vytvořeného shluku. Diference shluku s pro atribut a je spočítána rovnicí 4.3:

$$diff_{s,a} = \frac{\mu_a - \mu_{s,a}}{\mu_a - ext_{s,a}} \cdot 100, \tag{4.3}$$

kde μ_a je průměr atributu a v původní datové sadě, $\mu_{s,a}$ je průměr atributu a ve shluku s a $ext_{s,a}$ je minimální (pro menší hodnoty) nebo maximální (pro vetší hodnoty) hodnota atributu a ve shluku s.

- **Korelace** - Počítáme Spearmanův korelační koeficient ρ (4.4) pro libovolnou dvojici atributů v rámci shluku s. Získáme informaci, které atributy mají mezi sebou monotonní vztah (nejenom lineární). Koeficient se používá jak pro numerické či ordinální kategorické atributy. Vztah pro výpočet:

$$\rho = 1 - \frac{6\sum d^2}{n(n^2-1)}, \tag{4.4}$$

kde d je rozdíl pořadí dvou zkoumaných atributů a n je počet dat ve shluku s.

- **Velikost shluku** - Pouhá četnost dat v konkrétním shluku. Může identifikovat málo početné shluky nebo naopak velké shluky, které umožní vyšší míru redukce.

4.1.2 Analýza a vizualizace metrik

Po úspěšném natrénování shlukovacího modelu aplikujeme model na vstupní datovou sadu. Výsledkem je množina shluků, jenž rozdělují datovou sadu. Pro každý shluk je následně spočítána uvedená sada metrik. Množina metrik popisující jednotlivé shluky má textovou podobu uvedenou výpisem 4.1. Každá metrika je seřazena vzestupně podle hodnot. U metriky korelace atributů jsou uvedeny pouze jednosměrné korelační koeficienty (bez inverzní dvojice). Metriky jednotlivých shluků jsou podrobeny detailnější analýze pro určité případy užití za účelem vytvoření finálního reportu. V rámci návrhu byla provedena vizualizace metriky *diference shluku*, která vykreslí graf teplotní mapy napříč všemi shluky.

Výpis 4.1: Podoba výsledných metrik pro jeden shluk.

```
cluster <number>
Variance of features:
    feature_1 <value>
    feature_2 <value>
    ...
    feature_n <value>
Average of features:
    feature_1 <value>
    feature_2 <value>
    ...
    feature_n <value>
Average deviation of features (%):
    feature_1 <value>
    feature_2 <value>
    ...
    feature_n <value>
Correlation of features:
    feature_1 feature_2 <value>
    feature_2 feature_3 <value>
    ...
    feature_n−1 feature_n <value>
Size of cluster:
    <value>
```

Analýza statistických metrik byla provedena pro dva experimenty. První experiment zahrnuje analýzu datové sady obsahující jednotlivá DGA doménová jména. Pro experimenty byla zvolena záměrně problematika DGA, jelikož byla předmětem dřívějšího zkoumání, což přináší více zkušeností v dané oblasti. Druhý experiment je založen na analýze datové sady, jenž reprezentuje agregované hodnoty dílčích datových sad. Zatím se jedná čistě o experimenty na shlukování a analýzu konkrétních datových sad. Experimenty slouží jako stavební kameny pro automatizaci výpočtu výsledného skóre a následného sestavení vektoru vah (viz sekce 4.1.3).

Experiment 1

Vstupem byla datová sada, jejíž struktura je v tabulce 4.1. Jednotlivé položky jsou doménová jména vygenerovaná pomocí DGA. Atributy jednotlivých jmen byly spočítány na základě bakalářské práce [30]. Ke každému jménu jsou přepočítané atributy charakterizující konkrétní jméno. Celková velikost datové sady je 1 482 251 záznamů.

Tabulka 4.1: Ukázka datové sady obsahující doménová jména vygenerované pomocí DGA se vstupními atributy.

domain	dictionary	alexa	entropy	num	cons	len	dga	diff	met_ent
lcwbbhpudkajknyv.eu	0.375	0.462	3.750	0.000	0.875	16	0.75	1.288	0.234
pmdkikqswlxvlkov.eu	0.000	0.462	3.453	0.000	0.875	16	0.72	1.261	0.216
fqythgcoolts.pw	0.333	0.606	3.252	0.000	0.833	12	1.00	1.394	0.271
yomkanerraticallyqozaw.com	0.636	0.767	3.732	0.000	0.636	22	1.00	1.233	0.170
wigaardenslavetusul.com	0.737	0.956	3.577	0.000	0.579	19	1.00	1.044	0.188
llymupjmfulxnhyh.eu	0.188	0.464	3.203	0.000	0.875	16	0.75	1.286	0.200
b3eo7ob2g6lqdvvind.com	0.167	0.315	3.725	0.222	0.556	18	0.61	1.291	0.207
xqqtagwpxtjpesyb.eu	0.375	0.487	3.500	0.000	0.875	16	0.78	1.291	0.219
fjrksagabardinedazyx.com	0.550	0.837	3.722	0.000	0.700	20	1.00	1.163	0.186

Po provedení shlukové analýzy byly získány výsledné metriky pro jednotlivé shluky. Bylo vytvořeno 5 shluků. Ukázka spočtených metrik pro jeden konkrétní shluk 4 je uvedena na výpisu 4.2 (z důvodu přehlednosti byly některé méně významné atributy vynechány).

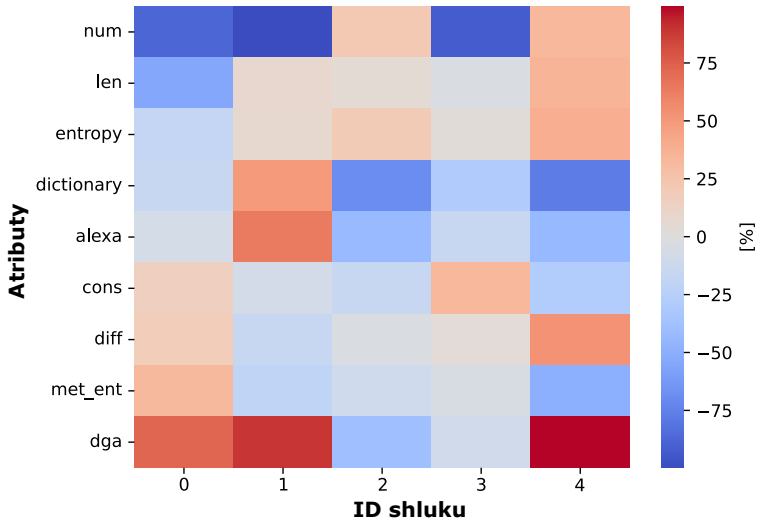
Výpis 4.2: Podoba statistických metrik pro shluk s ID 4.

```
cluster4
Rozptyl atributu:
    dga 0.000127
    metric_entropy 0.000725
    alexa 0.005195
    diff 0.005452
    dictionary 0.008860
    num 0.015981
    cons 0.016633
    entropy 0.061227
    len 17.690538

Prumer atributu:
    dictionary 0.083050
    metric_entropy 0.153770
    alexa 0.320295
    num 0.361999
    cons 0.508749
    dga 0.999356
    diff 1.679061
    len 26.519653

Diference shluku (%):
    dictionary -77.572480
    metric_entropy -49.845366
    alexa -44.024367
    cons -28.940243
    num 32.791800
    len 35.283658
    diff 53.524105
    dga 99.456082

Korelace atributu:
    alexa diff -0.988394
    len metric_entropy -0.938647
    num cons -0.865917
    dictionary num -0.369973
    alexa cons -0.340386
    len dga -0.042909
    dictionary dga 0.015315
    dga metric_entropy 0.052620
    cons dga 0.072466
    num len 0.342260
    cons diff 0.343314

Velikost shluku:
    80165
```

Charakteristika velikost shluku odhaluje početné shluky, které mají velký potenciál k redukci nebo naopak malé shluky, které mohou být predikovány s nízkou úspěšností. Diference shluku identifikuje, zda některé atributy ve shluku nejsou velmi odchýlené, což indukuje formu anomálie, u které může dojít k chybné klasifikaci. Korelace atributů může být použita k označení atributů, které jsou záporně korelovány, i když podporují stejnou třídu. Rozptyl atributů shluku oproti celkovému rozptylu napovídá, zda je daný shluk kompaktní či rozptýlený. Tabulka 4.2 zachycuje velikosti jednotlivých shluků, kde nejpočetnějším shlukem je shluk 3 (492 015 položek) a nejmenším je shluk s ID 4 (80 165 položek). Diference shluků jsou zobrazeny teplotní mapou na obrázku 4.2. Teplotní znázorňuje relativní odchýlení hodnot atributů od průměru původní datové sady. Červená barva značí nadprůměrné odchýlení hodnot a naopak modrá barva podprůměrné odchýlení hodnot. Z uvedených metrik je snahou odvodit shluky, které mohou být důležité při trénování modelu. Vizualizovaná teplotní mapa slouží k usnadnění analýzy jednotlivých atributů.

Tabulka 4.2: Velikost jednotlivých shluků pro experiment 1.

ID shluku	**0**	**1**	**2**	**3**	**4**
Velikost	302 862	440 220	166 988	492 015	80 165



Obrázek 4.2: Analýza metriky diference shluků nad jednotlivými daty.

Například shluk `4` obsahuje větší zastoupení číslic (atribut `num` 32.8 %) s nadprůměrnou délkou adres (atribut `len` 35.3 %) a vykazuje velkou podobnost s doménovými adresami vygenerovanými za pomocí DGA (atribut `dga` 99.5 %), zároveň je nejméně početným shlukem. Shluk `4` zastupuje delší DGA adresy, které obsahují vyšší množství číslic. Z hlediska klasifikace představuje dobře oddělitelný shluk a je vhodné mu přiřadit nižší váhu. Ověření může proběhnout i samotnou klasifikací. Je důležité taky poukázat na shluk `1`, jenž je druhým nejpočetnějším shlukem. Od původní datové sady se odchyluje nízkým zastoupením číslic (atribut `num` -95.4 %). Atributy `alexa` (61.2 %) a `dictionary` (45.3 %) podporují legitimní doménové jména a jsou hodnotově nadprůměrné, což naznačuje podobnost DGA jmen s legitimními adresami. Shluk `1` se hodnotami atributů přibližuje k druhé třídě a vede to k potencionální problémovější klasifikaci, a proto je vhodné mu přiřadit vyšší váhu.

Experiment 2

Druhá část experimentu se zabývala možností analyzovat datové sady "druhého řádu". Smyslem je vzít více datových sad, provést agregaci atributů a následně vytvořit datovou sadu, jež popisuje jednotlivé datové sady. V tabulce 4.3 je ukázka výsledné datové sady. Každý datový objekt v sadě reprezentuje datovou sadu, kde atribut `Dataset` je identifikátorem datové sady, která obsahuje DGA doménová jména vygenerované evolučním algoritmem (viz sekce 3.4). Zbylé atributy datového objektu jsou vytvořeny průměrováním původních atributů v dílčích datových sadách. Zkoumaná datová sada obsahuje 6 091 položek, jež každá představuje konkrétní datovou sadu. Provedením shlukové analýzy získáme skupiny datových sad, jež vykazují podobné rysy. Pro každý shluk je opět spočítaná sada statistických metrik jako v předešlém experimentu.

Shluková analýza odhalila 6 shluků, jejichž velikosti jsou uvedeny v tabulce 4.4. Nejméně početným shlukem je shluk `0`, který obsahuje pouze 179 položek. Mezi početné shluky (přes 1 500 položek) patří shluky `2` a `3`. Způsob vypočtu a formát výstupu (viz výpis 4.2) jednotlivých metrik je totožný s předchozím experimentem. Pro analýzu diference shluků byla provedena opět vizualizace pomocí teplotní mapy, která je zobrazena na obrázku 4.3. Nad teplotní mapou je provedena analýza, která povede k identifikaci zajímavých shluků.

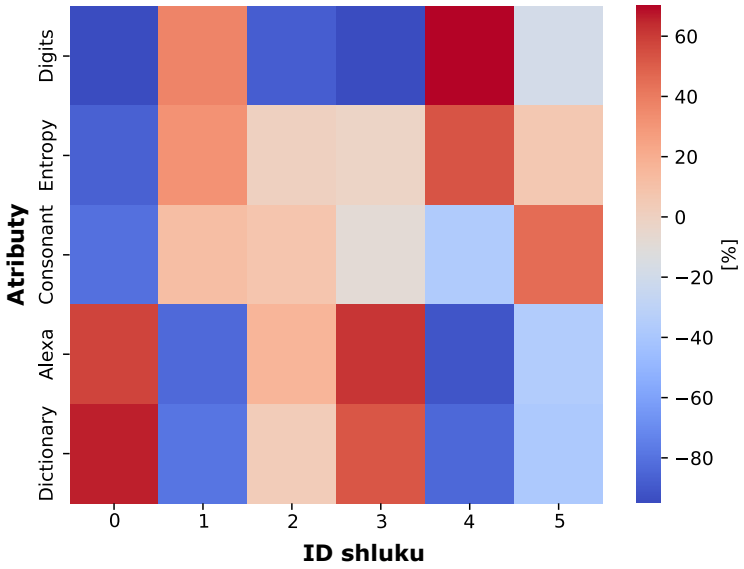
Tabulka 4.3: Ukázka datové sady obsahující agregovaná data. Atribut *Dataset* obsahuje název souboru datové sady a ostatní atributy jsou agregovány průměrem.

Alexa	Consonant	Dataset	Dictionary	Digits	Entropy
0.144	0.783	acc063.88_433.csv	0.272	0.004	2.404
0.149	0.714	acc053.00_86e.csv	0.431	0.002	2.426
0.033	0.724	acc093.12_dd5.csv	0.071	0.172	2.403
0.197	0.679	acc032.12_155.csv	0.421	0.004	2.406
0.225	0.668	acc008.62_264.csv	0.493	0.003	2.392
0.165	0.712	acc037.12_234.csv	0.295	0.004	2.408
0.160	0.702	acc049.12_257.csv	0.278	0.004	2.407
0.209	0.661	acc019.00_f5a.csv	0.486	0.005	2.393
0.184	0.721	acc033.25_fe8.csv	0.466	0.000	2.403

Tabulka 4.4: Velikost jednotlivých shluků pro experiment 2.

ID shluku	0	1	2	3	4	5
Velikost	179	897	1 623	2 007	454	931

Zajímavý je nejmenší shluk `0`, kde mají atributy `Dictionary` (60.8 %) a `Alexa` (54.1 %) nadprůměrné zastoupení a naopak atributy `Consonant` (-81.4 %) a `Entropy` (-86.9 %) vykazují podprůměrnější hodnoty. Zmíněné atributy o daném shluku říkají, že se jedná o datové sady obsahující DGA jména, která vykazují vyšší podobnost s validními doménovými jmény. Shluk `0` díky podobnosti s legitimními adresami má předpoklady způsobovat problematickou klasifikaci, a proto je vhodné mu nastavit velkou váhu. Na druhou stranu například shluk `1` má atributy `Dictionary` (-80.2 %) a `Alexa` (-83.7 %) podprůměrné. Jelikož uvedené atributy podporují legitimní adresy, tak jejich podprůměrné hodnoty značí dobrou oddělitelnost shluku. Shluk `1` je tedy kandidátem na nižší váhu.



Obrázek 4.3: Analýza metriky diference shluků nad agregovanými daty.

4.1.3 Výpočet skóre shluku

Po provedení manuální analýzy statistických charakteristik bylo cílem zadefinovat výpočet skóre, který zautomatizuje dříve manuálně provedený postup. Vektor vah je složen ze skóre pro jednotlivé shluky. Skóre by mělo udávat koeficient složitosti modelu klasifikovat daný shluk. Hodnota skóre je v rámci vektoru vah normalizována a bude sloužit jako podklad při modifikační fázi. Bude následovat popis jednotlivých komponent, které budou na vstupu pro výpočet finálního skóre a následného vektoru vah.

- **Průměrná diference** ($mean_k$) - Každý shluk obsahuje spočítanou metriku *diference shluku*. Daná metrika se zprůměruje pro jednotlivé atributy. Zjistí se tak, jak moc se daný shluk obecně odlišuje od ostatních.
- **Zastoupení extrémních diferencí** ($extreme_k$) - Spočítáme kolik atributů nabývá diference větší než 50 %. Následně získáme poměr atributů s extrémní diferencí vůči celkovému počtu atributů. Metrika odhalí, v kolika atributech se jednotlivé shluky hodnotově odchylují od zbytku dat.
- **Odchýlení úspěšnosti klasifikátoru** (ad_k) - Další ukazatelem při výpočtu skóre je ohodnocení shluku klasifikátorem (pokud je k dispozici). Na začátku se spočítá celková úspěšnost modelu nad celou datovou sadou. Následně se vypočítá relativní odchylka celkové úspěšnosti vůči úspěšnosti pro daný shluk. Relativní odchylka ad_k (viz rovnice 4.5) shluku k je spočítána jako poměr rozdílů úspěšností:

$$ad_k = \frac{(acc_{total} - acc_k -) - acc_{min}}{acc_{max} - acc_{min}}, \tag{4.5}$$

  kde acc_k je úspěšnost shluku k v původní datové sadě, acc_{total} je celková úspěšnost datové sady a $acc_{max,min}$ je minimální nebo maximální rozdíl úspěšností napříč všemi shluky. Pro zajištění vyšší míry fluktuace úspěšnosti jednotlivých shluků se klasifikátor použitý k analýze natrénuje ideálně na jiné podmnožině dat nebo proběhne trénování na velmi malé části analyzovaných dat.
- **Diference kontradiktorních atributů** ($contra_k$) - Dva atributy lze považovat za protichůdné, pokud vykazují zápornou korelaci. Pro ilustraci: atribut f_1, který je korelovaný s pozitivní třídou a druhý atribut f_2, který koreluje s negativní třídou. Vztah mezi těmito atributy by měl být záporně korelovaný. Pokud se zjistí, že dané dva atributy jsou záporně korelované, tak se kontroluje jejich diference. Za normálních podmínek by měly být jejich diference protichůdné, ale pokud jsou diference podobné, naznačuje to, že daný shluk může být klasifikován s nižší úspěšností.
- **Podobnost jiné třídě** (sim_k) - Najdeme atributy, které vykazují vyšší korelaci s jednotlivými třídami datové sady. Následně jsou tyto atributy zkoumány pro jednotlivé shluky. Cílem je najít shluky, které se mohou hodnotami atributů přibližovat jiné třídě. Atribut korelovaný s pozitivní třídou, bude mít ve shluku negativní třídy (překrývající se s pozitivní třídou) nadprůměrnou diferenci shluku. Hodnota metriky může následně identifikovat shluky, které mají tendenci se překrývat s jinou třídou.

Celkové skóre $score_k$ klasifikovatelnosti shluku je spočítáno jako průměr všech uvedených metrik pomocí rovnice 4.6. Všechny metriky nabývají hodnot v intervalu $\langle 0, 1 \rangle$. Čím vyšší hodnota, tím vyšší pravděpodobnost, že shluk bude mít nižší úspěšnost klasifikace.

$$score_k = \frac{mean_k + extreme_k + ad_k + contra_k + sim_k}{5} \tag{4.6}$$

V momentě, kdy je metrika konstantní nebo dosahuje záporné korelace vůči metrice ad_k, tak je z výpočtu vynechána. Důvodem konstantních hodnot jednotlivých prvků skóre (např. $contra_k$ nebo sim_k) může být neexistence dvojice atributů, které mají opačné korelační koeficienty vůči predikované třídě. V některých případech užití může tato situace nastat, a proto je nutné ji ošetřit. Tabulka 4.5 ukazuje konkrétní případ, kdy jsou dané metriky zahrnuty do výpočtu. Je provedena korelace s metrikou ad_k vůči ostatním metrikám. Případ z tabulky ukazuje, že metriky $extreme_k$ a $contra_k$ jsou konstantní (hodnota NaN), jelikož pro konstantní hodnoty není korelační koeficient definován. Metrika sim_k dosahuje záporného korelačního koeficientu vůči ad_k. Metrikou s kladným korelačním koeficientem je $mean_k$, která by spolu s ad_k byly jako jediné zahrnuty do výpočtu pro uvedený případ.

Tabulka 4.5: Vyhodnocení validity metrik pro výpočet skóre za pomocí korelace s metrikou úspěšnosti.

Metrika	**Korelace**
$mean_k$	0.70
$extreme_k$	NaN
$contra_k$	NaN
sim_k	-0.73

Po provedení výběru vhodným metrik do výpočtu je výsledné skóre normalizováno do intervalu $\langle 0.5, 1\rangle$ za pomocí rovnice 4.7:

$$weight_k = \frac{score_k - score_{min}}{score_{max} - score_{min}} \cdot 0.5 + 0.5, \tag{4.7}$$

kde $score_k$ je skóre shluku k a $(score_{max}, score_{min})$ je maximální a minimální hodnota skóre.

Spodní hranice 0.5 byla zvolena z důvodu, aby daný shluk při modifikaci nezanikl. Pokud shluk dosáhne normalizovaného skóre 1, tak představuje data, na kterých dosahuje klasifikátor relativní nejnižší úspěšnosti. Důsledkem je, že při redukci či selekci jsou data zachována v největším poměru.

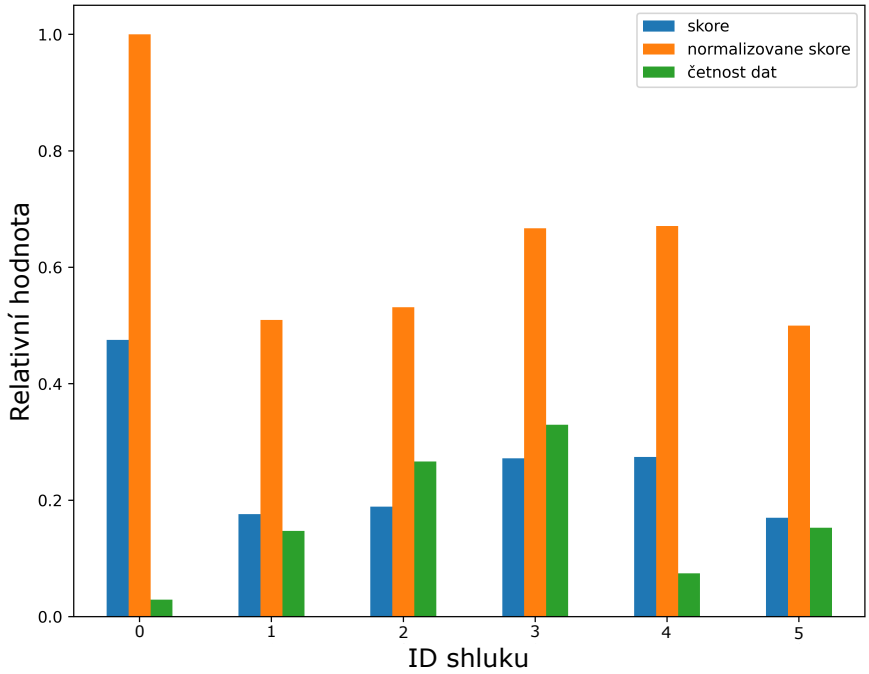
V některých případech může nastat, že jeden shluk se chová jako anomálie a dosahuje abnormálně vysokého skóre. Důsledkem anomálie mezi shluky jsou nízké rozdíly v normalizovaném skóre, tudíž dochází k nežádoucímu rovnoměrnému rozložení skóre. Anomálie je odhalena pomocí 2σ pravidla. Shluk se považuje za anomálii, pokud jeho skóre převyšuje práh $score_{mean} + 2 \cdot score_{std}$. Pokud je shluk identifikován jako anomálie, tak je při normalizaci skóre shluk vynechán a jeho váha je nastavena na 1. Při výpočtu 4.7 je za hodnotu $score_{max}$ dosazena druhá největší hodnota skóre. Váha pro shluk s druhým největším skórem se nerovná 1, ale nastaví se pomocí vztahu $(weight_3 + 1)/2$, kde $weight_3$ je normalizované skóre shluku s třetím nejvyšším skóre.

Podoba výsledného vektoru vah je znázorněna tabulkou 4.6. V tabulce dosahuje nejvyšší váhy shluk `1`, jež následují Shluky `2` a `4`. Naopak shluk `6` spolu se shluky `0`, `3` a `5` dosahují nejnižších vah. Z uvedeného vektoru vah plyne, že v největším poměru budou zachovány shluky `1`, `2` a `4` a na druhou stranu shluky `0`, `3`, `5` a `6` budou ve větší míře zredukovány. Získaný vektor vah následně slouží jako podklad pro modifikační fázi. Způsoby modifikace budou součástí navazující sekce 4.2.

Tabulka 4.6: Podoba výsledného vektoru vah pro datovou sadu.

ID shluku	0	1	2	3	4	5	6
Váha shluku	0.587	1.000	0.932	0.533	0.919	0.541	0.500

Pro ověření, že skóre a váhy odpovídají manuální analýze v provedených experimentech, se provedla vizualizace spočítaného skóre pro druhý experiment. Součástí výpočtu skóre nebyla zahrnuta odchylka úspěšnosti klasifikátoru. Obrázek 4.4 zobrazuje skóre, normalizované skóre a proporční velikost pro jednotlivé shluky.



Obrázek 4.4: Vizualizace skóre, normalizovaného skóre (váhy) a poměru velikostí jednotlivých shluků druhého experimentu.

Sloupcový graf z obrázku 4.4 ukazuje, že skóre shluku `0` dosahuje nejvyšší hodnoty (váha 1), a proto je vhodným kandidátem na rozšíření a při redukci bude zachován v plném rozsahu. Na druhou stranu ostatní shluky (včetně shluku `1`) mají nižší váhy, což povede na vyšší míru redukce s ohledem na četnost dat. Můžeme říct, že jsme pomocí automatizovaného přístupu získali stejné výsledky jako při manuální analýze.

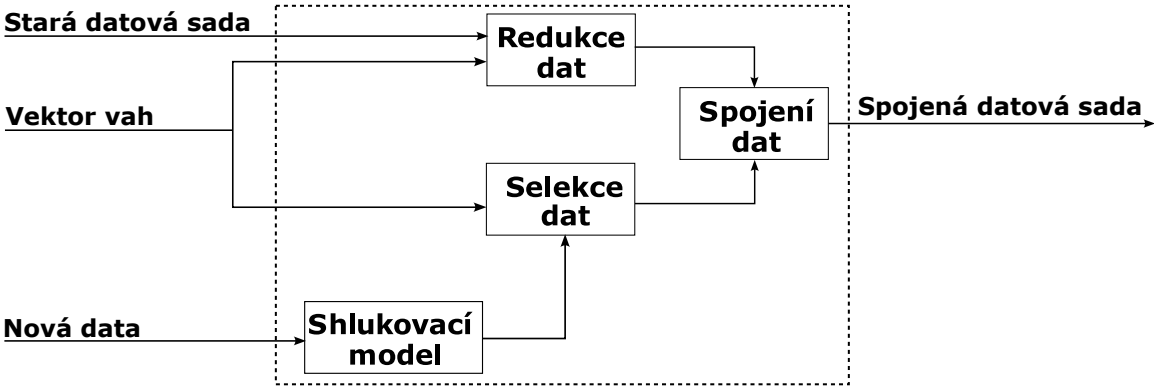
4.2 Možnosti modifikace

Na základě automatizovaně vygenerovaného vektoru vah bude upravena datová sada, aby byla vhodnější pro natrénování konkrétního modelu. Vstupem modifikační fáze bude analyzovaná datová sada, vektor vah a nová data. Je možné provádět dva modifikační úkony:

1. **Redukce staré datové sady** - Stará datová sada je podrobena shlukové analýze. Klasifikátor pro výpočet ad_k je natrénován na malé podmnožině dat pro zajištění vyšší fluktuace úspěšností pro jednotlivé shluky. Výsledný vektor vah je použit k redukci dat.

2. **Selekce nových dat pro rozšíření** - Nová data jsou opět podrobena shlukové analýze, kde se pro výpočet ad_k použije klasifikátor natrénovaný na starých datech. Získaný vektor vah identifikuje shluky, které původní klasifikátor predikuje s relativně nízkou úspěšností. Díky vektoru vah je možné vyselektovat data, která jsou vhodná pro rozšíření staré datové sady.

Oba zmíněné úkony mohou být prováděny současně nebo nezávisle na sobě. Je možné provést redukci staré datové sady nebo ji rozšířit o nově vybraná data. Redukovaná data jsou na závěr spojena s novými daty a vznikne tak nová modifikovaná datová sada, která by měla dosahovat lepších výsledků. Obrázek 4.5 graficky znázorňuje celý proces modifikace.



Obrázek 4.5: Struktura procesu při modifikaci původní datové sady na základě vygenerovaného reportu. Na vstupu jsou původní data, nová data a vygenerovaný vektor vah pro obě datové sady. Nová data budou sloužit k rozšíření původní datové sady.

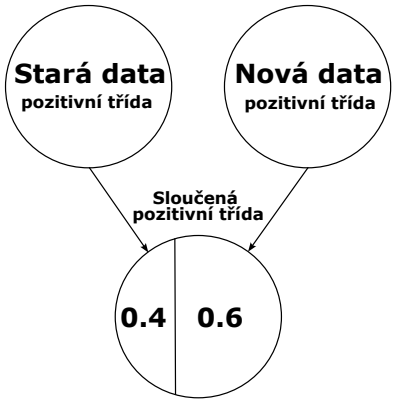
Redukce i rozšíření (selekce nových dat) dat probíhá proporčně k jednotlivým vahám reportu. Vstupním parametrem modifikace datové sady je výsledný počet prvků v dané třídě nebo poměr dat, které chceme redukovat či vybrat pro rozšíření. Proces redukce a rozšíření se ve své podstatě neliší. V obou případech dochází k váženému pod-vzorkování podle vektoru vah pro modifikovanou datovou sadu. Při redukci dochází k odstranění dat, které mají minimální dopad na výsledný klasifikátor. Naopak při rozšíření se snažíme extrahovat data, které klasifikátor predikuje s nižší úspěšností. Jednotlivé shluky jsou náhodně pod-vzorkovány na základě koeficientu $coeff_k$, který je spočítán rovnicí 4.8:

$$coeff_k = \frac{demanded}{potential} \cdot weight_k, \tag{4.8}$$

kde *demanded* je počet dat, které chceme extrahovat, *potential* je celková velikost po vynásobení jednotlivými váhami shluku $weight_k$.

V situaci kdy požadované množství dat převyšuje potencionální množství dané váhami $weight_k$, tak se implicitně jednotlivé váhy proporčně zvětší, aby bylo zajištěno požadovaný počet dat. V momentě, kdy získáme zredukovanou množinu starých dat a vyfiltrujeme nová data pro rozšíření, tak nastane sloučení zmíněných množin dat. Modifikační proces probíhá pro každou třídu zvlášť a můžeme určit poměr zredukovaných a nových dat za účelem získání váženého spojení výsledné datové sady, díky které může po natrénování model dosahovat lepších výsledků. Obrázek 4.6 ilustruje konkrétní případ, kdy dojde k váženému spojení starých a nových dat pro pozitivní třídu. Poměr mezi daty může být nastaven manuálně nebo se odvodí z poměru úspěšností pro jednotlivé třídy starých a nových dat. Například při analýze zjistíme, že původní model klasifikuje negativní třídu nové datové sady stále

s vysokou úspěšností a naopak pozitivní třídu s nižší úspěšností. Důsledkem je, že by měla mít pozitivní třída pro nová data nastavena vyšší poměr, zatím co u negativní třídy může poměr zůstat vyvážený.



Obrázek 4.6: Vážené spojení starých dat a nových dat pozitivní třídy. Data jsou spojena v poměru 2:3. Poměr může být libovolně nastaven. Stejné váhování ve zvoleném poměru probíhá i pro negativní třídu.

Cílem nových dat je přinést do datové sady nové objekty, které přinesou novou informaci pro následné trénování modelu. Nyní bude uvedeno několik způsobů, jak dodat nová data do modifikačního procesu.

- **Generování dat** - Pokud v konkrétní problematice existuje možnost vygenerovat data z dané domény, tak je určitě efektivní tuto variantu využít. Například v doméně DGA existuje možnost generovat podvržené domény za pomocí evolučního algoritmu (viz sekce 3.4). Jako fitness funkce by byl zvolen poměr domén, které spadají do potřebného shluku na základě vektoru vah.
- **Kolekce dat ze sítě** - Dalším zdrojem jsou reálná data zachycená ze síťového provozu (viz sekce 3.3). Nevýhodou je ale dodatečná anotace dat. V doméně DGA by se získaly z kolektoru doménové adresy a následně se může ověřit, že jsou validní například nástrojem *Whois*. Tento způsob ale zajistí data jen z jedné třídy, a proto je vhodné způsoby získání nových dat kombinovat.
- **Sjednocení více sad** - Neposledním způsobem je vytvoření jedné datové sady sjednocením napříč množinou více datových sad. Původní datová sada je rozšířena o data z ostatních datových sad a získá se efektivně spojená datová sada, která může být použita pro trénování. Kromě samotného rozšíření je možnost zredukovat původní data za účelem obohacení o aktuálnější data při zachování původní velikosti datové sady.

Kapitola 5

Implementace modifikátoru datových sad

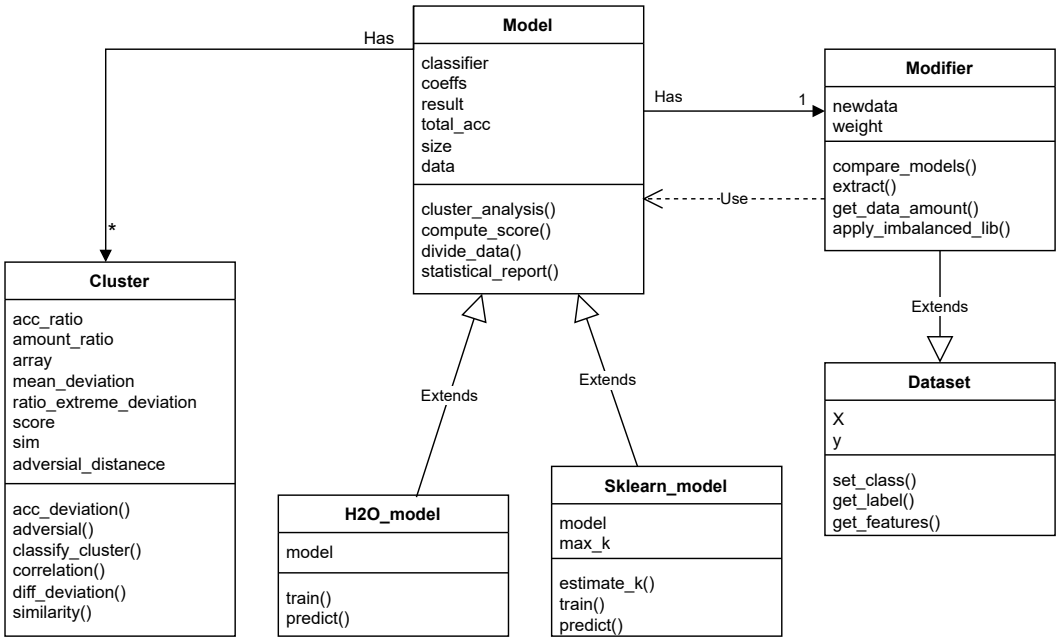
Implementace frameworku probíhala v prostředí jazyka `Python`. Implementace probíhala iterativním způsobem. Postupně byly přidávány jednotlivé funkcionality a byla snaha vytvořit finální podobu architektury. Je využito třídního přístupu pro lepší udržovatelnost a přehlednost. Celý framework je interaktivní a umožňuje nastavování různých parametrů modifikace. Interaktivnímu přístupu napomáhá využití interaktivního `Python` notebooku. Notebook umožňuje přehledně sledovat průběh modifikace a efektivně modifikovat jednotlivé části kódu. Framework se celkově skládá z několika tříd (viz diagram tříd 5.1) a kódu, jenž dané třídy využívá. Framework poskytuje dva způsoby modifikace datové sady. V prvním případě dochází k redukci datové sady a v druhém případě po redukci dochází k rozšíření o nová data. Implementace je zatím uzpůsobena pro zpracování datových sad pro binární klasifikaci. V síťovém prostředí se často setkáme s binární klasifikací.

5.1 Architektura

Navržený způsob modifikace byl transformován do podoby třídního diagramu, který je zobrazen na obrázku 5.1. Diagram tříd je pojat abstraktně a obsahuje nejzákladnější atributy a metody pro nastínění základních vazeb. Datová sada je reprezentovaná třídou `Dataset`. Třída poskytuje základní funkcionalitu pro načtení a extrakci atributů vstupních dat. Třídu `Dataset` rozšiřuje třída `Modifier`, která doplňuje základní funkcionalitu o metody, které provádějí již modifikaci dat a jejich následné vyhodnocení. S třídou `Modifier` pracuje třída `Model`, která má za úkol datovou sadu rozdělit na shluky vhodným algoritmem. Uvnitř třídy jsou vytvořeny jednotlivé shluky, které zastupuje třída `Cluster`. Třída `Cluster` má za úkol spočítat jednotlivé statistické metriky a následně i komponenty, ze kterých je spočítáno skóre shluku. Vektor vah je zpracován ve třídě `Model` a je použit třídou `Modifier` při modifikační fázi pro vážené vzorkování dat. Detailnější vazby a principy jsou popsány v následujících částech.

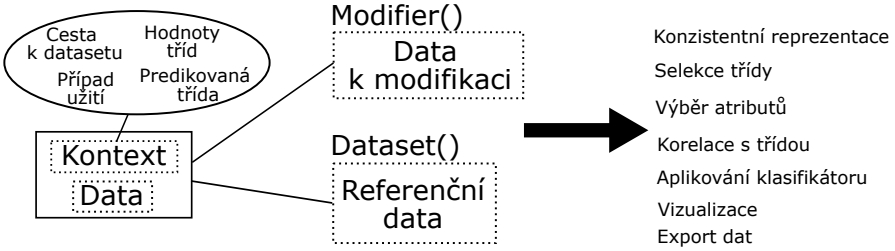
5.1.1 Manipulace s daty

Prvním krokem modifikace je zpracování vstupních datových sad. V případě redukce byla na vstupu použita jedna datová sada. Pro situaci spojování datových sad jsou na vstupu dvě datové sady. Proces zpracování vstupních dat je konceptuálně zobrazen na obrázku 5.2. Spolu s daty jsou na vstupu metadata, která definují kontext datové sady. Kontext definuje



Obrázek 5.1: Diagram tříd pro modifikační framework datových sad.

případ užití, cestu k datům, jméno klasifikovaného atributu a jeho hodnoty. Vstupní data jsou rozdělena dle zvoleného poměru na část pro modifikaci (`Data k modifikaci`) a část pro testování (`Referenční data`). `Data k modifikaci` jsou zpracována třídou `Modifier`, jelikož třída umožní jejich pozdější modifikaci 5.1.4 podle vektoru vah. `Referenční data` jsou zpracovány třídou `Dataset` a slouží k ověření, zda model natrénovaný na modifikovaných datech dosahuje lepších metrik než při trénování na datech bez modifikace. Mezi základní operace, které poskytují obě třídy patří vytvoření konzistentní reprezentace predikovaného atributu bez ohledu na vstupní data. Predikovaná třída nabývá hodnoty 1 pro pozitivní třídu a hodnoty 0 pro negativní třídu. Další operací je možnost nastavit třídu, se kterou se má pracovat při výpočtu vektoru vah. Dále je možné vybírat atributy pro shlukovou analýzu, najít korelační koeficienty s predikovanou třídou, aplikovat na data klasifikátor (za účelem získání celkové úspěšnosti), vizualizovat rozložení dat nebo modifikovaná data exportovat. Data jsou načtena v rámci třídy ve formátu `csv` pomocí knihovny `Pandas`[1]. Vnitřní reprezentace datové sady je uložena v heterogenní 2D struktuře `Dataframe`.

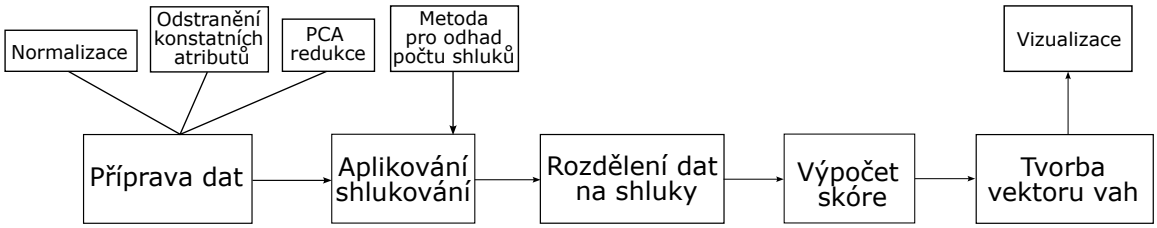


Obrázek 5.2: Schéma pro zpracovaní a rozdělení vstupních dat se základní funkcionalitou.

[1] https://pandas.pydata.org/

5.1.2 Shlukovací proces

Hlavní komponentou frameworku je shlukovací fáze, jenž je reprezentována abstraktní třídou `Model`. Třída definuje abstraktní metody, jež budou implementovány různými typy knihoven pro aplikování shlukovacích algoritmů. Kromě abstraktních metod jsou implementovány metody pro řízení celého shlukovacího procesu k získání výsledného vektoru vah. Shluková analýza musí být provedena na datech stejné třídy, jinak by navržený postup nefungoval. Tok dat analytického procesu pro zvolenou třídu je znázorněn na obrázku 5.3. Analytický proces, který probíhá uvnitř třídy `Model`, se dělí na několik kroků. V první fázi se vstupní data podrobí předzpracování formou normalizace, odstranění konstantních hodnot a případné redukce dimenzí metodou PCA. Na předzpracovaných datech se provede natrénování zvoleného shlukovacího modelu a zobrazí se jeho parametry (zvolený počet shluků). Vstupem shlukovacího modelu je i metoda pro automatický odhad počtu shluků. Po aplikování shlukování se data rozdělí do jednotlivých shluků (instanciace třídy `Cluster`) a pro každý shluk jsou spočítány všechny potřebné hodnoty pro výpočet skóre. Množina hodnot skóre je následně normalizována pomocí vzorce 4.7 a spočítané hodnoty tvoří finální vektor vah. Vektor vah je na závěr uložen do interní podoby a může být případně vizualizován (viz obrázek 4.4). Bylo implementováno více shlukovacích algoritmů a více metod pro odhad počtu shluků. Pro jednotlivé algoritmy byly využity dvě knihovny, které usnadní jejich implementaci. Jedná se o knihovny `H2O` a `Sklearn`. Následuje konkrétní popis integrace použitých knihoven.



Obrázek 5.3: Tok vstupních dat při analýze frameworkem.

H2O

První knihovna je implementována prostřednictvím cloudové platformy `H2O`[2]. `H2O` poskytuje modul pro `Python` a umožňuje externě aplikovat metody strojového učení a přenést výpočetní nároky na stranu serveru, jenž disponuje distribuovanými jednotkami pro efektivní paralelizaci. Na serveru běží *Java virtual machine*, se kterým komunikujeme pomocí REST rozhraní. Třída `H2O_model` zajišťuje propojení knihovny do implementovaného frameworku. Abychom mohli využít `H2O` modul, musíme provést konverzi dat na kompatibilní strukturu `H2OFrame`. V rámci knihovny je primárně používán shlukovací algortimus *K-means*. Model je inicializován s maximálním počtem shluků 10. Maximální počet iterací trénování je nastaven na 100. Model umožňuje dynamicky určit počet shluků k. Metoda pro výpočet ideálního počtu shluků je uvedena níže [12]:

1. Začneme s velikostí shluků k=*1*, následně je pro shluk spočítán střed.
2. Najdeme atribut s největším rozsahem a rozdělíme data podle jeho střední hodnoty.

[2] https://docs.h2o.ai/h2o/latest-stable/h2o-docs/index.html

3. Spustíme *k-means* metodu nad dvěma vzniklými shluky.
4. Najdeme shluk, jenž obsahuje atribut s největším rozsahem a rozdělíme shluk podle jeho střední hodnoty.
5. Proces opakujeme až do doby, kdy je splněna terminální podmínka.

`H2O` používá jako terminální podmínku *proportional reduction in error* (PRE) 5.1. Hodnota PRE je spočítána na základě sumy čtverců (SSW) následovně:

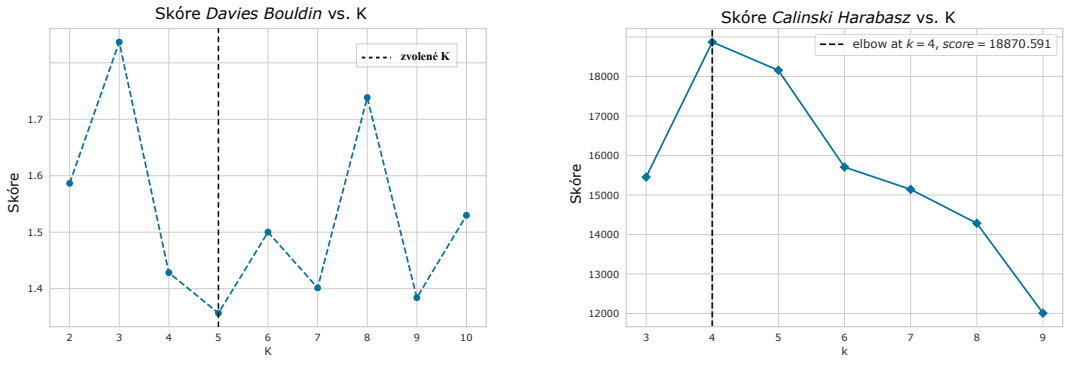
$$PRE = \frac{(SSW[before\ split] - SSW[after\ split]))}{SSW[before\ split]} \tag{5.1}$$

Vytváření nových shluků skončí, když PRE bude menší než stanovený práh. Cílový práh je stanoven jako menší hodnota z dvojice:

$$\begin{aligned} prah = \ & 0.8 \\ & \mid 0.02 + \frac{10}{pocet\_dat} + \frac{2.5}{pocet\_atributu^2} \end{aligned} \tag{5.2}$$

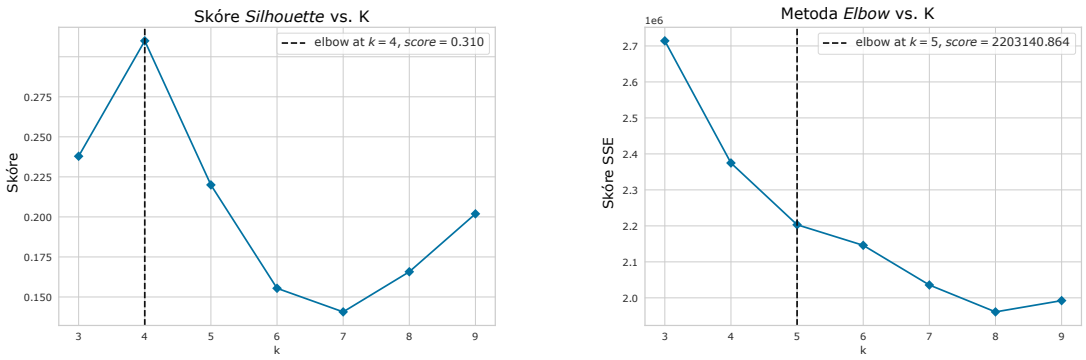
Sklearn

Druhou využitou knihovnou je `Sklearn`[3], jenž je běžně používána pro aplikování datové analýzy a strojového učení. Třída `Sklearn_model` implementuje rozhraní knihovny. V rámci třídy můžeme definovat libovolný shlukovací algoritmus, který poskytuje `Sklearn`. Konkrétně jsou definovány metody *K-means* a aglomerativní hierarchické shlukování. Pro automatizovaný odhad počtu shluků je na výběr z více metod. Opět dochází k iterativnímu navyšování shluků, dokud není splněna terminální podmínka. Implementované metody pro určení terminální podmínky jsou založeny na výpočtu různých skóre, včetně *Calinski Harabasz*, *Silhouette*, *Davies Bouldin* a také na úbytku sumy čtverců středů shluků (*Elbow*) [22]. Uvedené metody jsou vždy založené na metrikách uvedených v sekci 2.2.6. Implementace je provedena za pomocí vizualizační knihovny `Yellowbrick`[4], která rozšiřuje `Sklearn`. Knihovna umožňuje odhadnout optimální počet shluků konkrétní metodou a následně ji vizualizovat. Obrázek 5.4 ilustruje výpočet parametru k z pohledu různých metod. V rámci frameworku je lepší pracovat s metodami, které odhadují vyšší počet shluků, což umožní identifikovat více skupin dat pro modifikaci.



Obrázek 5.4: Vizualizace implementovaných metod pro automatický výpočet počtu shluků.

[3] https://scikit-learn.org/stable/
[4] https://www.scikit-yb.org/en/latest/index.html



Obrázek 5.4: Vizualizace implementovaných metod pro automatický výpočet počtu shluků.

Metody *Calinski Harabasz* a *Silhouette* volí optimální k při maximální hodnotě skóre. Naopak pro metodu *Davies Bouldin* je optimální k při minimální hodnotě. Metoda *Elbow* sleduje poměr úbytku chybové metriky sumy čtverců středů jednotlivých shluků a volí k při nízké změně chybové metriky. Nejvyšší počet shluků (5) byl odhadnut metodami *Elbow* a *Davies Bouldin*.

5.1.3 Skóre shluku

Jednotlivé shluky jsou reprezentovány třídou `Cluster`. Třída implementuje všechny operace, které se provádí nad daty patřící do jednoho shluku. Data jsou překonvertována na `Numpy`[5] pole z důvodu výkonnosti. Díky konverzi dochází k efektivnějšímu výpočtu všech metrik v rámci shluku. Jsou zde implementovány jak statistické metriky 4.1.1, tak metriky pro výpočet skóre 4.1.3. Každá metrika je spočítána příslušnou metodou třídy `Cluster`. Pro výpočet skóre pro jednotlivé shluky potřebujeme kromě spočítaných metrik i globální informace o celé datové sadě. Mezi globální informace patří korelace atributů s klasifikovanou třídou. Korelace s predikovanou třídou je spočítána pomocí metody, která je zahrnuta v třídě `Dataset` a je součástí základních operací nad datovou sadou (viz obrázek 5.2). Na základě spočítaných korelací jsou dodatečně vytvořeny dvojice atributů, které mají protichůdné korelační koeficienty vůči predikované třídě. Po spočítání všech potřebných informací pro výpočet skóre je podle rovnice 4.6 získána finální hodnota skóre pro daný shluk.

Zde bude uvedena konkrétnější implementace metrik $contra_k$ a sim_k, jelikož jejich implementace jednoznačně nevyplývá z návrhu. Pro jednotlivé implementace budou uvedeny pseudokódy pro vyjádření algoritmického postupu. Pseudokód metriky $contra_k$ je uveden v algoritmu 1. Pro výpočet metriky $contra_k$ se pro každou dvojici záporně korelovaných atributů ve shluku zjistí, zda tyto atributy vykazují obracený korelační koeficient vůči predikované třídě (budou se nacházet v množině dvojic, které vykazují opačné korelační koeficienty s predikovanou třídou). Pokud ano, tak se provede rozdíl metriky diference shluku (viz vzorec 4.3) pro dané atributy. Rozdíly jsou akumulovány a na závěr se hodnota zprůměruje, provede se normalizace do intervalu $\langle 0, 1\rangle$ a získá se doplněk do jedničky.

Výpočet metriky sim_k se provádí ve dvou fázích. Pseudokód 2 naznačuje algoritmický výpočet. V první fázi se vezmou záporně korelované atributy s predikovanou třídou a spočítá se průměr jejich diferencí (viz vzorec 4.3), který je převeden do intervalu $\langle 0, 1\rangle$. Průměr se počítá jen v situaci, kdy množina korelovaných atributů s predikovanou třídou není prázdná.

[5] https://numpy.org/doc/stable/index.html

To stejné se provede pro kladně korelované atributy. Druhá fáze provede doplněk do jedničky podle právě analyzované třídy, jelikož pro každou třídu má metrika opačný význam. Finální metrika se spočítá jako průměr hodnot pro kladně a záporně korelované atributy.

Algoritmus 1 Výpočet metriky $contra_k$

Vstup: Záporně korelované dvojice atributů neg_pairs a dvojice atributů s opačným korelačním koeficientem vůči predikované třídě $corr\_pairs$.

let $contra = 0$
let $cnt = 0$
for $neg\_pair \in neg\_pairs$ **do**
    if $neg\_pair \in corr\_pairs$ **then**
        $cnt = cnt + 1$
        Získej diferenci shluku $diff_1$ pro první atribut z dvojice neg_pair
        Získej diferenci shluku $diff_2$ pro druhý atribut z dvojice neg_pair
        let difference = $|diff_1$ - $diff_2|$
        contra = contra + difference
    end if
end for
if $cnt \neq 0$ **then**
    $contra = 1 - contra/(2 \cdot cnt)$
end if

Výstup: Metrika $contra$ pro daný shluk k.

Algoritmus 2 Výpočet metriky sim_k

Vstup: Pozitivně korelované atributy pos_attrs a negativně korelované atributy neg_attrs vůči predikované třídě.

let $sim\_neg = 0$
for $neg\_attr \in neg\_attrs$ **do**
    Získej diferenci shluku $diff$ pro první atribut neg_attr
    $sim\_neg = sim\_neg + diff$
end for
if $length(sim\_neg) \neq 0$ **then**
    $sim\_neg = (1 + sim\_neg/length(sim\_neg))/2$
end if

let $sim\_pos = 0$
for $pos\_attr \in pos\_attrs$ **do**
    Získej diferenci shluku $diff$ pro první atribut pos_attr
    $sim\_pos = sim\_pos + diff$
end for
if $length(sim\_pos) \neq 0$ **then**
    $sim\_pos = (1 + sim\_pos/length(sim\_pos))/2$
end if

if *je analyzována pozitivní třída* **then**
    $sim\_pos = 1 - sim\_pos$
else
    $sim\_neg = 1 - sim\_neg$
end if

let $sim = (sim\_neg + sim\_pos)/2$

Výstup: Metrika sim pro daný shluk k.

5.1.4 Modifikace dat

Třída `Modifier` implementuje metody, které už vedou na modifikaci datové sady nebo slouží k vizualizaci a porovnání výsledků. Mezi modifikační metody patří hlavně `extract()` a `apply_imbalanced_lib()`. Pro aplikování konkurenčních metod pro redukci dat (viz sekce 3.1) je použita metoda `apply_imbalanced_lib()`. Framework využívá k provádění modifikace funkci `extract()`. Funkcionalita, která stojí za metodou `extract` je znázorněna pseudokódem v algoritmu 3. Na vstupu funkce je shlukovací model, vektor vah, požadované množství dat k extrakci a samotná data pro modifikaci. Data analyzované třídy jsou rozdělena na jednotlivé shluky shlukovacím modelem. Následně se každý shluk iterativně podrobí extrakci dat. Množství dat, které má být vybráno pro každý shluk zahrnuje dva faktory. Prvním faktorem je váha shluku a druhý faktor je požadovaná velikost extrahovaných dat. Vynásobením velikosti shluku s jeho váhou se získá potencionální velikost extrahovaných dat ve shluku. Pro zajištění celkové požadované velikosti dat pro všechny shluky, jsou váhy pro každý shluk upraveny, aby součet extrahovaných dat jednotlivých shluků dával dohromady celkovou požadovanou velikost. Úprava vah probíhá proporčně, aby bylo zachováno jejich původní rozložení (viz rovnice 4.8). Upravená váha ($coeff_i$) slouží jako koeficient pro náhodně pod-vzorkování daného shluku. Pod-vzorkované shluky jsou postupně přidávány do množiny, která bude tvořit finální výstup extrakčního procesu.

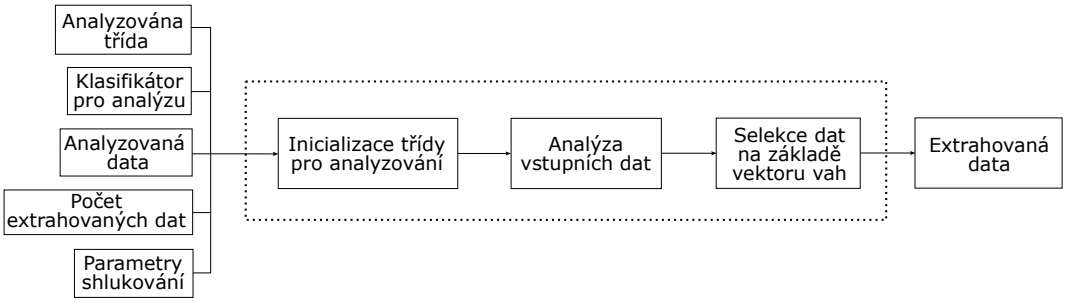
Algoritmus 3 Extrakce dat na základě vektoru vah

Vstup: Shlukovací model *cluster_model*, vektor vah *weights*, požadovaný počet dat k extrakci *demanded_amount* a data pro modifikaci *data*.

1: Rozděl vstupní *data* na jednotlivé shluky pomocí modelu *cluster_model*
2: Získej celkový počet shluků K
3: **let** $R = \emptyset$
4: **for** $i \in K$ **do**
5: Vyber $data_i$ patřící do shluku i
6: Vyber váhu $weight_i$ pro shluk i
7: Na základě $weight_i$ a *demanded_amount* spočítej vzorkovací koeficient $coeff_i$
8: Náhodně pod-vzorkuj $data_i$ pomocí $coeff_i$
9: Přidej pod-vzorkovaná data do množiny R
10: **end for**

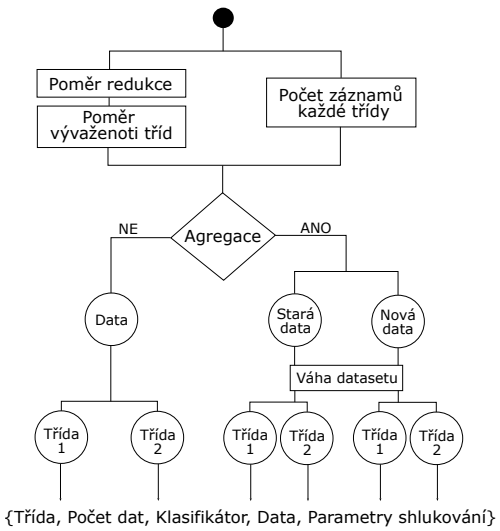
Výstup: Množina R obsahující extrahovaná data.

Celý modifikační proces konkrétní třídy a konkretní datové sady implementuje nezávisle definovaná funkce `modification_one()`. Funkce využívá metody výše uvedených tříd. Schématická podoba modifikačního procesu je uvedena na výpisu 5.5. Do procesu vstupuje několik parametrů. První parametr určuje, jaká třída má být podrobena analýze a modifikaci. Dalším parametrem je klasifikátor, který bude při analýze sloužit k výpočtu metriky ad_k. Dále jsou na vstupu samotná data pro analýzu. Parametr počtu extrahovaných dat říká, kolik dat se má extrahovat při selekci dat. Parametry shlukování definují použitou knihovnu, která aplikuje shlukovací algoritmus, a také metodu pro odhad počtu shluků. Uvnitř procesu dochází k inicializaci zvoleného rozhraní knihovny pro aplikovaní shlukovacího algoritmu. Výběr knihovny probíhá formou vytvoření vhodného potomka třídy `Model`. Následně dochází k analýze vstupních dat (viz obrázek 5.3), jejíž výsledkem je vektor vah. Vektor vah se uplatní při selekci požadovaných dat (viz algoritmus 3) a extrahována data jsou předána na výstup modifikačního procesu.



Obrázek 5.5: Proces k zajištění analýzy a extrakce dat pro konkrétní data a třídu.

Celá funkce `modification_one()`, která zajišťuje proces analýzy a extrakce dat je navržena a implementována za účelem snadného zakomponování paralelního přístupu. Paralelní řešení je zavedeno pomocí knihovny `Joblib`[6]. Vstupní parametry pro modifikační funkci jsou vypočítány hromadně pro všechny třídy a analyzovaná data. Výsledkem je vstupní vektor parametrů pro funkci `modification_one()`, která je prováděna paralelně. Počet paralelních procesů je dán typem modifikace. Pokud provádíme pouze redukci, tak stačí dvě jádra pro každou třídu, ale v momentě slučování dvou datových sad lze využít až čtyři jádra. Výpočet vektoru parametrů pro paralelní extrakční proces (viz obrázek 5.5) je zobrazen diagramem 5.6. Počet dat, který se má extrahovat je dán několika komponenty. Jednou z možností je zadat počet výsledných záznamů pro danou třídu nebo definovat poměr redukce a poměr vyvážení tříd. Je zvoleno, zda se provede redukce nebo agregace dat. V případě, že se provádí redukce, tak se pracuje s jednou datovou sadou. Při agregaci se pracuje s dvěma datovými sadami. Pro dva datasety je navíc definován jejich slučovací poměr pro každou třídu (viz obrázek 4.6). Dále se stanoví pro každá data klasifikátor, shlukovací parametry a určíme extrahovaný počet dat. Vznikne několik inicializačních vektorů pro různé třídy, které jsou předány pro paralelní zpracování funkcí `modification_one()`. Na závěr modifikační fáze se provede spojení extrahovaných dat pro jednotlivé třídy a data pomocí funkce `concat()`. Vzniklou modifikovanou datovou sadu uložíme pro následné testovaní.



Obrázek 5.6: Proces k vytvoření inicializačního vektoru pro modifikační fázi.

[6] https://joblib.readthedocs.io/en/latest/parallel.html

Kapitola 6

Výsledky a diskuze

Pro účely ověření funkčnosti navrženého řešení byly zvoleny dva případy užití. V prvním případě půjde o redukování datové sady 6.2. Datová sada bude rozdělena na trénovací a testovací část. Trénovací část bude předložena frameworku, který provede analyzování dat a následné vytvoření reportu. Report bude použit pro redukční fázi. Na zredukovaných datech bude natrénovaný nový model, který bude sloužit k porovnání s původním modelem.

Druhý případ užití bude vytvářet datovou sadu, která vznikne sloučením dvou datových sad 6.3. Z obou datových sad je vyčleněna část dat, která bude sloužit k testování. Testovací sada obsahuje části obou datových sad. Zbylé části datových sad jsou připraveny na modifikaci. V prvním kroku se zredukuje původní datová sada, která bude v dalším kroku doplněna o data z nové datové sady. Nová data jsou opět vybrána na základě vytvořeného reportu na nových datech.

Aplikování modifikačního frameworku vyžaduje referenční klasifikátor. Klasifikátor se externě načte nebo se přímo natrénuje na analyzovaných datech. V rámci frameworku je použito několik klasifikátorů. Bude zavedeno pojmenování pro jednotlivé klasifikátory pro pochopení kontextu, jelikož názvy se budou vyskytovat v průběhu testování. Klasifikátor `Old` slouží jako referenční klasifikátor pro porovnání výsledků a je natrénován na původní trénovací datové sadě. Klasifikátor `Dumb` je použit k výpočtu deviace úspěšnosti ve shluku ad_k. Je natrénován na velmi malé podmnožině trénovacích dat pro zajištění fluktuace úspěšnosti. Na datech vytvořených modifikačním frameworkem konkrétní metodou je natrénován klasifikátor `Clustered`, což je sekundární výstup frameworku hned po modifikované datové sadě. Poslední klasifikátor je `Compare`, který je výstupem konkurenční metody, který bude porovnán s `Clustered` klasifikátorem. Všechny testovací případy využívají napříč všemi datovými sadami model rozhodovacího stromu z knihovny `Sklearn` s volitelně zakomponovanou *Adaboost* technikou, která umožňuje natrénovat více klasifikátorů s různými váhami pro trénovací sadu.

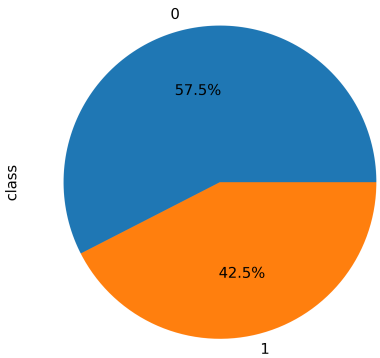
6.1 Datové sady

V rámci práce byly využity dva typy datových sad. První typ dat se zaměřuje na detekci doménových jmen vygenerovaných pomocí DGA. DGA (*Domain generation algorithm*) slouží ke generování podvržených doménových jmen k těžko detekovatelné komunikaci mezi útočníkem a obětí. Princip generování evolučním algoritmem byl uveden v sekci 3.4.1. Datová sada s DGA adresami slouží k binární klasifikaci, zda se jedná o legitimní doménovou adresu či byla adresa vygenerována pomocí DGA algoritmu. Datová sada byla vytvořena v rámci

bakalářské práce, která se touto problematikou zabývala [30]. Datová sada s vypočítanými atributy je uvedena v tabulce 6.1. Predikovaná třída je reprezentována atributem `class`. Dále jsou obsaženy atributy, které měří podobnost s legitimními adresami pomocí atributu `alexa`, shodu ve slovníku pomocí atributu `dictionary` a podobnost s DGA adresami pomocí atributu `dga`. Délka adresy je vyjádřena atributem `len`, entropie atributem `entropy`, zastoupení souhlásek atributem `cons`. DGA adresy mají například vyšší hodnoty pro atributy `dga`, `cons` nebo `entropy`. Naopak nižších hodnot pro DGA adresy dosahují atributy `alexa` a `dictionary`. Vyšší počet subdomén, který je reprezentován atributem `suffix_num`, naznačuje legitimní adresu. Neutrálním atributem je zastoupení číslic `num`. Celková velikost s procentuálním zastoupením tříd je vizualizována grafem 6.1. Datová sada je nevyvážená, majoritní třídou (57.5 %) jsou legitimní adresy. Celková velikost datové sady je 2 093 827 záznamů.

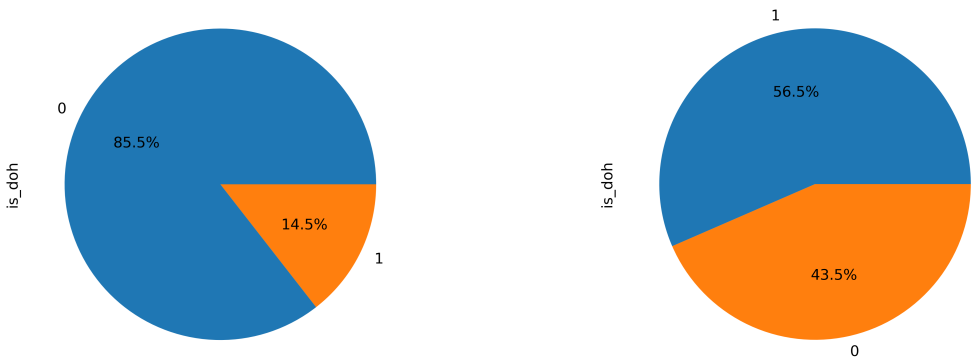
Tabulka 6.1: Ukázka DGA datové sady s doménovými adresami.

	0	1	2
domain	lcwbbhpudkajknyv.eu	pmdkikqswlxvlkov.eu	fqythgcoolts.pw
class	dga	dga	dga
dictionary	0.4	0.0	0.3
alexa	0.5	0.5	0.6
alexa3	1.0	1.0	0.9
alexa4	0.4	0.4	0.7
alexa5	0.0	0.0	0.2
entropy	3.8	3.5	3.3
num	0.0	0.0	0.0
cons	0.9	0.9	0.8
len	16	16	12
suffix	1	1	1
suffix_num	1	1	1
www	0	0	0
dga	0.8	0.7	1.0
dga3	1.0	1.0	1.0
dga4	1.0	1.0	1.0
dga5	0.2	0.2	1.0
diff	1.3	1.3	1.4
metric_entropy	0.2	0.2	0.3
spec_char	0.0	0.0	0.0



Obrázek 6.1: Zastoupení jednotlivých tříd DGA datové sady s celkovou velikostí 2 093 827.

Druhým typem datové sady je detekce DoH (*DNS over HTTPS*). DoH přispívá ke zvýšení soukromí uživatele přidáním šifrování do překladu DNS adres. Cílem je detekovat DoH oproti normálnímu HTTPS provozu. Detekce probíhá nad šifrovanými daty, z čehož plynou i zvolené atributy datových sad. V oblasti DoH se pracovalo s dvěma datovými sadami. První obsahovala uměle vygenerovaná data za pomocí virtualizovaného prostředí s aktivovaným DoH v různých prohlížečích. Druhý dataset byl vytvořen na základě reálného provozu, kdy se všechny DNS dotazy přesměrovaly na proxy server, který aplikoval DoH. Podrobnější informace o tvorbě DoH dat jsou uvedeny v článku [36]. Tabulka 6.2 ilustruje ukázku dat s konkrétními atributy. Množina atributů je pro obě datové sady totožná. Mezi atributy, které dobře rozlišují DoH provoz patří počet paketů `packets_sum`. Požadavky a odpovědi DoH DNS se obvykle skládají z nejméně pěti paketů, což umožňuje rozlišit kratší připojení jako klasické HTTPS. Nejvýznamnějším rozdílem mezi DoH a klasickým HTTPS je doba trvání toku `time`. Se serverem DoH je navázáno spojení, které se používá po delší dobu. Delší spojení může být vytvořeno také stahováním nebo sledováním videa, ale zde se přenese mnohem víc dat za krátký interval (`bytes`). Komunikaci DoH lze identifikovat menším rozptylem velikostí paketů v požadavku (atribut `var_pkt_size`) a méně rozlišitelný je rozptyl velikostí v odpovědi `var_pkt_size_rev`. DoH má tendenci mít symetrické množství příchozích a odchozích dat po celou dobu spojení (atribut `bytes_ration`). Atributy `stSum`, `ndSum`, `rdSum` sledují symetričnost pro každou třetinu spojení nezávisle, aby se mohla rozlišit prvotní symetričnost HTTPS spojení, která se později stane nesymetrickou. DoH implementované v prohlížečích vykazuje specifické vzory chování s dávkami paketů a pauzami ovlivněnými interakcí uživatele. Dochází k nepravidelnému zpoždění mezi pakety (atributy `minDelay`, `avgDelay`, `maxDelay`), které dobře rozlišuje DoH komunikaci [36]. Obrázek 6.2 zachycuje jednotlivé sady, kde lze vidět procentuální zastoupení tříd a celkovou velikost datasetu. Generovaný dataset obsahuje 1 581 271 záznamů a je velmi nevyvážený, DoH provoz zahrnuje 14.5 % celkových dat. Reálný dataset zahrnuje 2 055 955 položek, z toho 43.5 % tvoří DoH provoz.



Obrázek 6.2: Zastoupení predikovaných tříd pro generovaný DoH dataset o velikosti 1 581 271 (vlevo) a reálný DoH dataset o velikosti 2 055 955 (vpravo).

Tabulka 6.2: Ukázka atributů DoH datové sady.

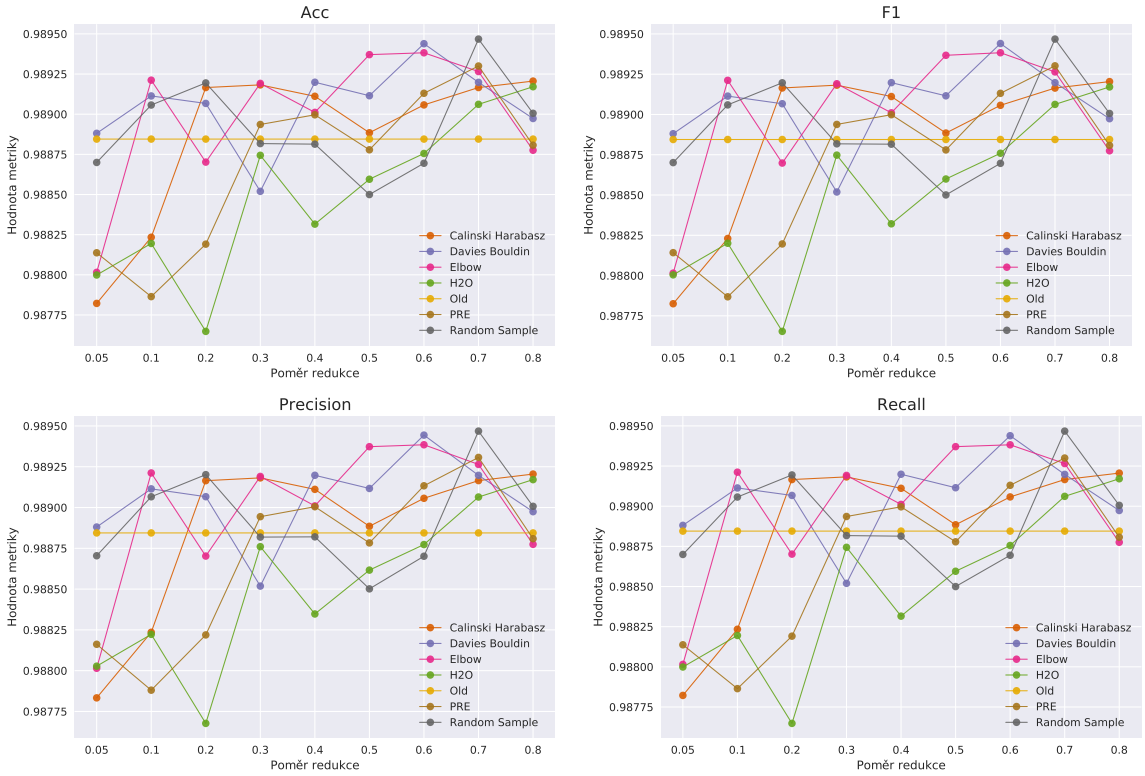
	0	1	2	3	4
is_doh	True	True	True	False	False
bytes_rev	6 505	4 024	4 507	5 851	2 934
bytes	3 678	1 554	1 848	873	873
packets	35	14	16	7	7
packets_rev	40	15	20	10	7
packets_sum	75	29	36	17	14
bytes_ration	0.6	0.4	0.4	0.1	0.3
num_pkts_ration	0.9	0.9	0.8	0.7	1.0
time	5 024.8	5 065.8	4 485.3	882.2	59.1
av_pkt_size	105.1	111.0	115.5	124.7	124.7
av_pkt_size_re	162.6	268.3	225.3	585.1	419.1
var_pkt_size	13 726.6	26 588.6	21 219.5	51 302.2	51 302.2
var_pkt_size_rev	214 048.6	314 756.0	253 736.3	292 217.8	170 459.6
median_pkt_size	105.5	105.5	105.5	290.5	290.5
median_pkt_size_rev	58.5	48.0	55.0	1416.0	654.0
mindelay	9.1	9.7	9.8	280.6	15.1
avgdelay	1 479.1	540.7	873.9	341.2	28.7
maxdelay	3 311.9	5 056.5	4 476.2	602.7	46.0
bursts	0	0	0	0	0
fazzel	0	0	0	0	0
time_leap_ration	1.0	1.0	1.0	1.0	1.0
autocorr	0.0	0.6	0.0	0.0	0.0
stSum	0	-1	1	0	0
ndSum	-3	0	-4	-1	0
rdSum	1	-3	-2	0	0

6.2 Redukce dat

Prvním případem testování byla redukce dat. Snahou bylo zredukovat analyzovanou datovou sadu, aniž by se zásadně zhoršila úspěšnost zkoumaného klasifikátoru. Datová sada byla vždy podrobena shlukové analýze a na jejím základě byl vytvořen report. V rámci shlukové analýzy byly aplikovány různé metody pro automatizovaný odhad počtu shluků (viz rovnice 5.1 a obrázek 5.4). Z testování byla vyřazena metoda `Silhouette` z důvodu špatné škálovatelnosti. Jako shlukovací algoritmus byl použit *K-means*, jelikož vykazuje dobrou škálovatelnost. Modifikační fáze probíhala za účelem vytvoření vyvážené datové sady. Cílem bylo porovnat metriky klasifikátoru (*Accuracy,F1,Precision,Recall*) pro jednotlivé metody a zároveň je porovnat pro různé poměry redukce. Metriky dosahují v některých případech minimálních rozdílů a na grafech vypadají podobně. Metody byly porovnány s referenčním modelem `Old` a dále s konkurenčním modelem `Compared - Random Sample`, který využíval náhodného pod-vzorkování. Datová sada se postupně zmenšovala po 10 procentech.

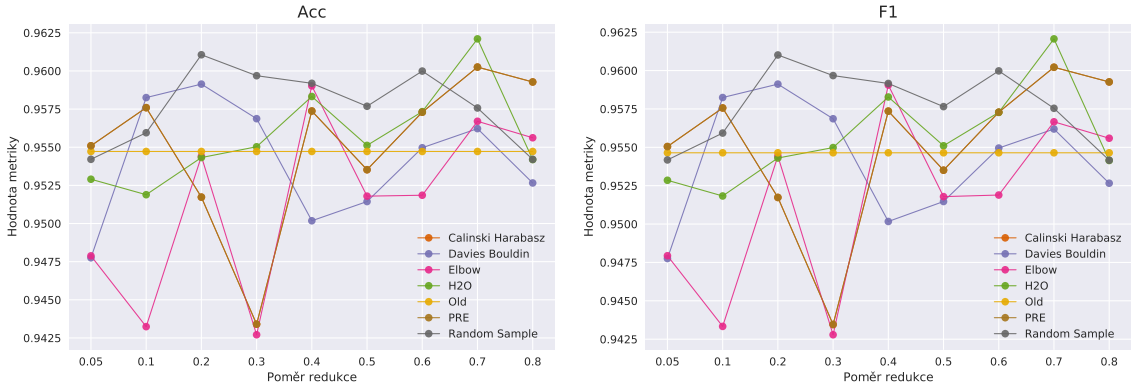
První datovou sadou pro redukování byl DGA dataset 6.1. Obrázek 6.3 zobrazuje jednotlivé metriky DGA klasifikátoru. V grafech je zanesena hodnota referenčního klasifikátoru `Old`. Z grafů lze pozorovat, že při menší míře redukce dosahují skoro všechny metody lepších metrik než u referenčního modelu. Podařilo se překonat `Compared` model pro náhodné vzorkování `Random Sample`. Shlukovací metody dosahují lepších metrik než náhodné pod-vzorkování skoro pro všechny poměry redukce (kromě míry redukce 0.7 a 0.2). Nejhůře si vedla metoda `H2O` a naopak metoda `Davies Bouldin` dosahuje nejlepších výsledků i ve velkých mírách redukce. Shlukovací metody při nízké míře redukce převážně vykazují lepší

výsledky než referenční modely. Při vyšší míře redukce dosahují lepších výsledků už jen některé metody (`Elbow`,`Davies Bouldin`), konkretně pro míru redukce 0.1.

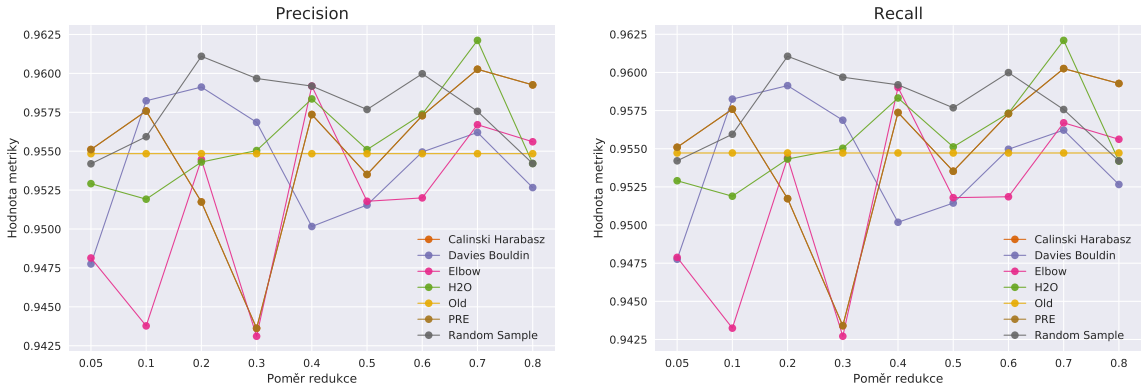


Obrázek 6.3: Vizualizace jednotlivých metrik porovnávaných modelů pro různé poměry redukce pro redukovanou DGA datovou sadu příslušnými metodami.

Další zredukování proběhlo pro datovou sadu obsahující reálná data pro detekci DoH 6.2. V případě DoH nedošlo k rapidnímu zlepšení oproti náhodnému pod-vzorkování `Random Sample`. Výsledky postupného redukování pro různé modely jsou zobrazeny grafy na obrázku 6.4.



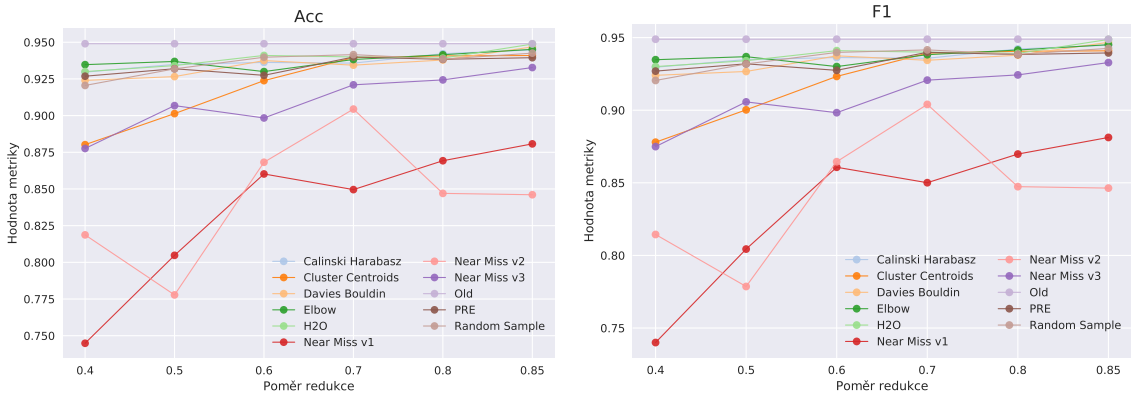
Obrázek 6.4: Vizualizace jednotlivých metrik porovnávaných modelů pro různé poměry redukce v rámci redukce DoH datové sady (reálná data) příslušnými metodami.



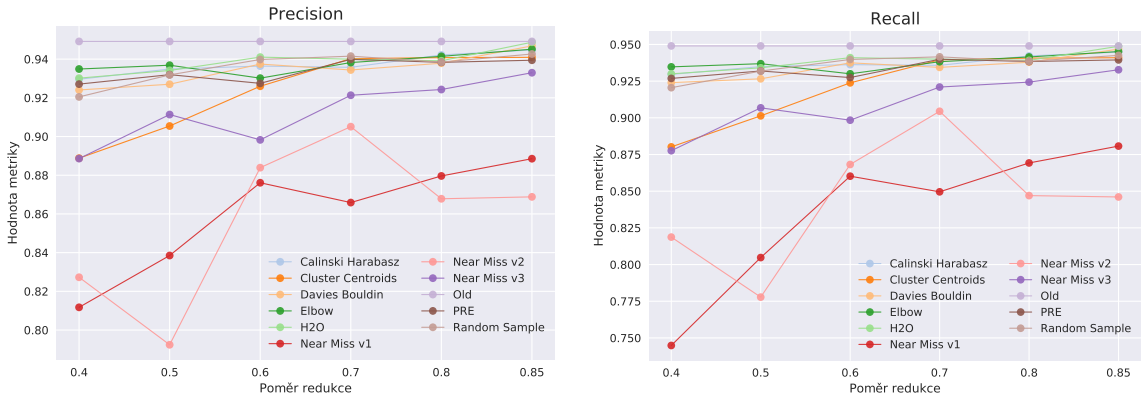
Obrázek 6.4: Vizualizace jednotlivých metrik porovnávaných modelů pro různé poměry redukce v rámci redukce DoH datové sady (reálná data) příslušnými metodami.

Při nízké míře redukce (0.8 a 0.7) dosahují některé metody lepších výsledků než náhodné pod-vzorkování `Random Sample` i referenční model `Old`. Naopak pro střední míry redukce náhodné pod-vzorkování dosahuje lepších metrik. Ve vysoké míře redukce (0.1 a 0.05) bylo opět dosaženo lepších výsledků než u referenčních modelů. Lepší metriky byly získány jen v případě nízké míry redukce metodami `PRE` a `H2O`. Dalšího zlepšení bylo dosaženo v případě vysoké míry redukce metodou `PRE` a částečně metodou `Davies Bouldin`.

V posledním případě v rámci redukce byly srovnány metody pro vyvažování datových sad 3.1. Zredukování proběhlo na malé podmnožině (17 132) původní DoH reálné datové sady 6.2. Důvodem testování na malém množství dat je časová náročnost některých porovnávaných metod pro vyvažování. Časové porovnání bude provedeno v sekci 6.4. Byly srovnány metody, u kterých lze definovat faktor vyváženosti tříd R_{D_S}, aby mohl být dosažen cílený poměr redukce. Konkrétně se jedná o metody `Near miss v1,v2,v3`, `Clustered Centroids` a již porovnávaný náhodný `RandomUnderSampler`. Obrázek 6.5 znázorňuje jednotlivé metriky výsledných klasifikátorů pro modifikované datové sady s různými poměry redukce. Všechny modely generované na základě frameworku dosahují relativně konstantních výsledků. Porovnávané metody existujících přístupů dosahují horších výsledků skoro ve všech redukčních poměrech. V několika případech byla překonána i metoda `RandomUnderSampler`.



Obrázek 6.5: Vizualizace jednotlivých metrik porovnávaných modelů s existujícími řešeními pro různé poměry redukce v rámci malé podmnožiny DoH datové sady (reálná data).



Obrázek 6.5: Vizualizace jednotlivých metrik porovnávaných modelů s existujícími řešeními pro různé poměry redukce v rámci malé podmnožiny DoH datové sady (reálná data).

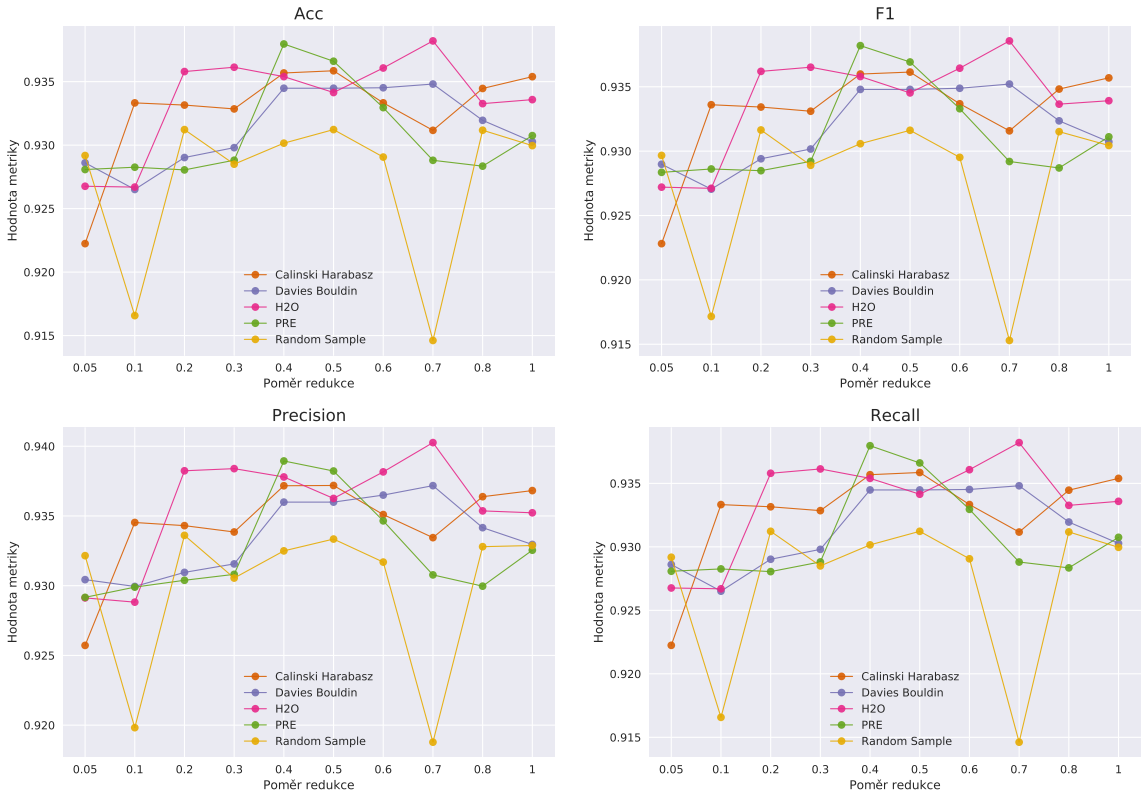
Z porovnávaných metod si nejlépe vedla metoda `Cluster centroids`, která dokázala konkurovat metodám frameworku v nižších mírách redukce (0.7 a 0.8), ale při vyšších poměrech redukce je metoda vždy horší. Metody `Near miss v1,v2,v3` nedokázaly v porovnání uspět v žádném poměru redukce. Žádná metoda nepřekonala referenční model `Old`, jelikož se pracuje s nízkým množstvím dat a sebemenší odebrání dat vede na horší model. Zmíněný scénář redukce primárně slouží pro porovnání s existujícími řešeními.

6.3 Slučování datasetů

Dalším a více přínosnějším případem užití pro uvedený framework je slučování neboli agregace datasetů. Principem agregace je spojení dvou datových sad za účelem nalezení efektivní podmnožiny. Hledá se podmnožina dat, jelikož v síťovém prostředí se nachází velké množství dat. Pro zajištění, aby model byl stále aktuální a uměl predikovat aktuální provoz na síti, tak je nutné periodicky doplňovat nová data do trénovací sady, aby odrážela aktuální situaci na síti. Zároveň je vhodné, aby v původní trénovací datové sadě zůstala i stará data, protože i ty se mohou na síti objevit. Pouhým přidáváním nových dat by trénovací datová sada neustále rostla a v určitý moment by se trénovací sada stala neakceptovatelná pro trénování z důvodu nadměrné velikosti. Framework přichází s konkurenčním řešením vůči pouhému sjednocení a následnému náhodnému pod-vzorkování.

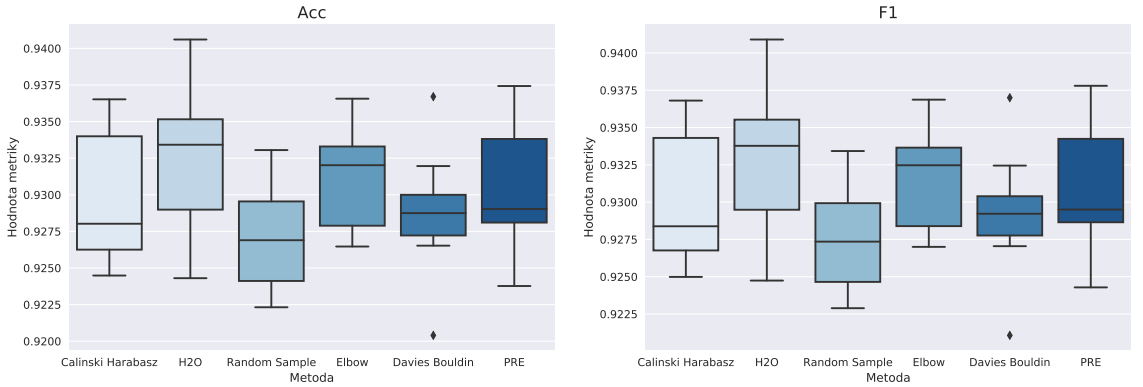
Pro testování byl použit demonstrativní případ užití, kdy se datová sada s generovanými DoH daty zredukuje (vyfiltrují se relevantní data pro trénování) a následně je doplněna o nová data z datové sady s reálnými DoH záznamy 6.2. V rámci spojení je zachována původní velikost datové sady nebo dojde k redukci datové sady ve vyšší míře. Uvedená testovací situace je zrealizována algoritmem *K-means* spolu s metodami na odhad počtu shluků, které jsou porovnány napříč různými úrovněmi redukce. Obrázek 6.6 ilustruje zvolené metriky testovacího scénáře pro různé metody a jednotlivé poměry redukce. Metody frameworku jsou dobře oddělitelné od porovnávané metody. Skoro ve všech případech implementované metody dosahují lepších výsledků než referenční metoda náhodného pod-vzorkování `Random Sample`. Nejlépe si vedou metody `H2O` a `Calinski Harabasz`.

Jelikož navržený framework i porovnávaná metoda náhodného pod-vzorkování pracují s faktorem náhodnosti. Pro vyloučení náhodnosti z výsledků bylo provedeno iterativně více opakování stejného úkonu. Modifikace datové sady proběhla 10x za sebou se stejnými

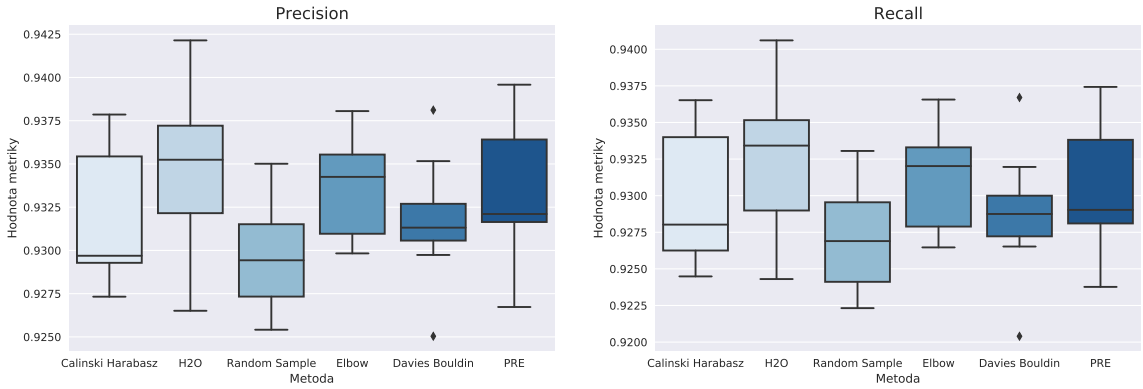


Obrázek 6.6: Vizualizace metrik modelů při tvorbě agregované podmnožině dvou DoH datových sad pro konkrétní metody s různými poměry redukce.

podmínkami (redukční poměr 0.5). Výsledkem byl vzorek populace pro jednotlivé metriky obsahující faktor náhody. Distribuce jednotlivých metrik pro zmíněné metody jsou porovnány pomocí krabicových grafů na obrázku 6.7. Obrázek zachycuje rozdíly distribucí pro jednotlivé metody. Střední hodnoty metrik všech implementovaných metod jsou vyšší než konkurenční metoda náhodného pod-vzorkování `Random Sample`. Mezi horší metody patří `Calinksi Harabasz` díky nejnižší střední hodnotě a `Davies Bouldin` lze považovat za horší kvůli výskytu odchýlené hodnoty s nízkou hodnotou.



Obrázek 6.7: Vizualizace statistického porovnání distribucí metrik pro jednotlivé metody pro ověření pozitivního přínosu frameworku.



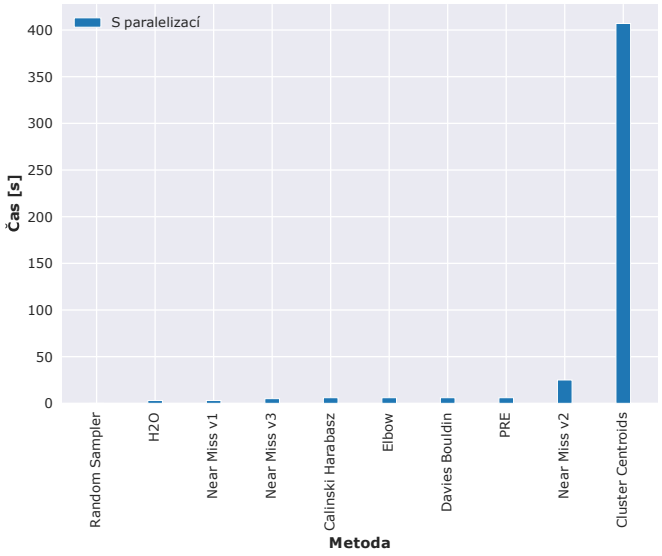
Obrázek 6.7: Vizualizace statistického porovnání distribucí metrik pro jednotlivé metody pro ověření pozitivního přínosu frameworku.

Průměrných výsledků dosahuje metoda `PRE`. Mezi nejlepší metody se díky svým relativně vyšším středním hodnotám řadí `H2O` a `Elbow`. Metoda `H2O` dosahovala nejvyšší maximální úspěšnosti (94.1 %) a minimální hodnota (92.4 %) je vyšší než u porovnávané metody `Random Sample` (92.2 %). Z uvedených grafů je možné vyvodit, že navržený modifikační framework pozitivně přispívá na výsledné metriky natrénovaného modelu. Z dlouhodobějšího pohledu framework dokáže produkovat modely, které vykazují vyšší metriky než při pouhém sloučení a náhodném pod-vzorkování.

6.4 Časová komplexita

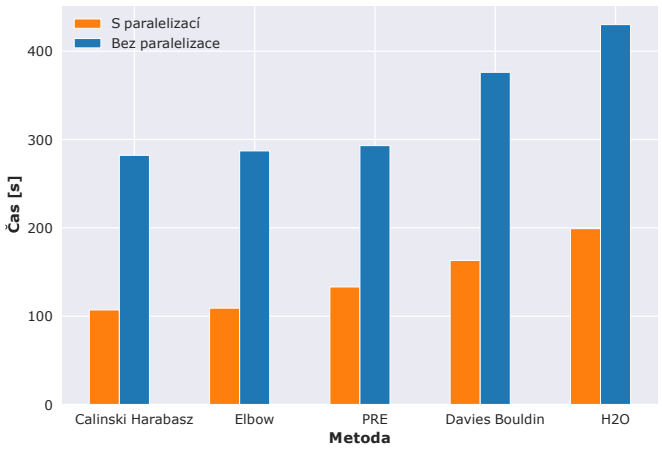
Přichází na řadu získání časové náročnosti frameworku zpracovat vstupní datové sady, které jsou podstoupeny modifikaci. Referenční stroj, na kterém probíhalo výkonové testování disponuje dvanácti procesory *Intel(R) Xeon(R) CPU E5-2609 v3 @ 1.90GHz* s velikostí RAM 270 GB. Bude provedeno časové srovnání pro oba případy užití (redukce i sloučení), jelikož v každém případě je provedena jiná míra paralelizace. V případě redukce jsou vytvořeny dva procesy pro zpracování dvou tříd. Naopak při sloučení dochází k inicializaci čtyř procesů pro každé dvě třídy dvou datových sad.

Pro redukci bylo provedeno časové srovnání s metodami pro vyvažování datových sad 3.1. Srovnání proběhlo pro paralelizované varianty všech metod. Graf 6.8 zobrazuje trvání porovnávaných metod pro malou podmnožinu dat (17 132 vzorků). Nejlépe si vede metoda náhodného pod-vzorkování `Random Sample`, která zpracovala vstupní dat za 36 ms. Metoda `Random Sample` je primitivní, a proto dosahuje nejrychlejšího zpracování. Z metod frameworku je nejrychlejší `H2O` metoda (3 s). Ostatní implementované metody dosahují podobné časové náročnosti, kdy zpracují vstupní data za 6 s. Existující metody z `imbalanced-learn` je možné rozdělit na dvě části. Metoda `Near Miss v1` zpracuje data za 3 s a metoda `Near Miss v3` má dobu zpracování 5 s. Nedochází tak k velkému časovému rozdílu. Naopak časové náročně jsou metody `Near Miss v2` (25 s) a speciálně metoda `Cluster Centroids`, které trvá přes 6 minut zpracovat malé množství dat. Metoda `Cluster Centroids` je důvodem proč testování probíhalo na malé podmnožině dat.



Obrázek 6.8: Porovnání časové náročnosti redukce metodami frameworku a existujícími řešeními na celkovém počtu 17 132 záznamů.

Při slučování datasetů došlo ke srovnání mezi sekvenčními a paralelními verzemi metod frameworku. Graf 6.9 srovnává uvedené přístupy. Celková velikost zpracovávaných datových sad činí 3 637 226. Mezi nejrychlejší metody patří `Calinski Harabasz` (107 s), `Elbow` (109 s) a `PRE` (133 s). Na druhou stranu nejpomalejší metoda je `H2O` (199 s). Důvodem zpomalení oproti časovému prvenství při redukci 6.8 může být horší škálovatelnost, jelikož na pozadí knihovny běží virtuální stroj jazyka `Java`. Tabulka 6.3 rozšiřuje časovou náročnost na počet zpracovaných datových objektů za sekundu. V nejlepším sekvenčním případě pro metodu `Calinski Harabasz` je framework schopen zpracovat 12 897 datových objektů za sekundu a v paralelním případě se jedná o 33 992 datových objektů za sekundu. Dochází k 2.64násobnému zrychlení. Naopak v nejhorším sekvenčním případě metoda `H2O` zpracuje 8 458 položek za sekundu a v paralelní verzi 18 277 položek za sekundu. Dosaženo je 2.16násobného zrychlení.



Obrázek 6.9: Časová náročnost sloučení dvou datasetů pro jednotlivé metody v rámci frameworku na celkovém počtu 3 637 226 záznamů.

Tabulka 6.3: Srovnání časové náročnosti s ohledem na velikost zpracovávaných dat.

Metoda	Počet datových objektů za sekundu bez paralelizace	Počet datových objektů za sekundu s paralelizací	Zrychlení
Calinski Harabasz	12 897.96	33 992.77	2.64
Elbow	12 673.26	33 369.05	2.63
Davies Bouldin	9 673.47	22 314.27	2.31
PRE	12 413.74	27 347.56	2.20
H2O	8 458.67	18 277.52	2.16

6.5 Diskuze

Bylo provedeno několik experimentů s vytvořeným frameworkem. Framework byl aplikován na dva případy užití. Na začátku se prováděla redukce datové sady a poté se uskutečnilo sloučení dvou datových sad. Při redukování byl framework aplikován na dva typy dat (DGA a DoH data). Redukce DGA datové sady zlepšila metriky natrénovaného modelu na datové sadě vytvořené pomocí frameworku. Zlepšení úspěšnosti nebylo výrazné (méně než o 0,1 %), jelikož klasifikace DGA dosahovala ve výchozím stavu velmi vysokých metrik s úspěšností přes 98 %. Důležité je, že se podařilo překonat metriky původního modelu, ale i modelu, který byl natrénován na datové sadě, která byla náhodně pod-vzorkována. Zlepšení nastalo ve všech redukčních poměrech pro většinu metod implementovaných frameworkem. Při řešení byl použit primárně algoritmus *K-means* z důvodu dobré škálovatelnosti. Současně byl aplikován algoritmus hierarchického shlukování, který dosahoval podobných výsledků. Do budoucna by bylo možné ještě zkusit algoritmy založené na hustotě.

V problematice DoH došlo k překonání původního modelu při menších redukčních poměrech. Došlo ke zlepšení úspěšnosti o 0.4 % při redukčním poměru 0.7. Některé metody frameworku dokázaly zlepšit metriky i pro vyšší míru redukce. Metoda `Davies Bouldin` byla lepší o 0.2 % při poměru redukce 0.1. Model postavený na náhodném pod-vzorkování byl horší jen v několika případech. Důvodem horších metrik v některých situacích je faktor náhodnosti a také struktura DoH atributů (horší analýza atributů šifrovaného provozu).

Redukce byla dále provedena pro malou množinu DoH dat za účelem porovnání s dalšími existujícími metodami pro modifikaci. Metody frameworku dosahovaly stabilních výsledků pro všechny míry redukce a vždy dosáhly lepších metrik než u existujících metod. Existující metody dosahovaly horších metrik, které byly kolísavé, speciálně ve vyšších mírách redukce. Při míře redukce 0.5 v porovnání s metodami frameworku byl zaznamenán nejvyšší pokles úspěšnosti (15 %) pro metodu `Near Miss v2` a nejmenší (2 %) pro metodu `Near Miss v3`. Z pohledu časové náročnosti jsou implementované metody konkurenceschopné s existujícími přístupy (kromě náhodného vzorkování), v některých případech jsou dokonce i časově efektivnější.

Druhým případem užití bylo sloučení dvou datových sad. Většina metod frameworku dosahovala výrazně lepších metrik než v případě sloučení a náhodného pod-vzorkování. Bylo dosaženo lepších metrik pro všechny míry redukce kromě jedné (0.05). V nejlepším případě došlo metodou `H2O` ke zlepšení úspěšnosti o 2 % při redukčním poměru 0.7. Modifikační framework v sobě nese faktor náhodnosti (při modifikaci se provádí náhodná redukce s váhou shluku). Pro potlačení náhodného vlivu bylo frameworkem vytvořeno 10 datových sad se stejnými podmínkami pro redukční poměr 0.5. Natrénováním modelu na datových sadách byly zjištěny distribuce metrik napříč všemi datovými sadami pro porovnávané metody. Krabicovým grafem byly porovnány jednotlivé metody. Všechny metody frameworku do-

sahovaly vyšších středních hodnot než přístup náhodného pod-vzorkování. Nejlépe si vedla metoda `H2O`, která měla o 0.7 % vyšší střední hodnotu úspěšnosti než porovnávaná metoda `Random Sample`. Porovnáním distribucí lze tvrdit, že metody frameworku dosahují z dlouhodobějšího hlediska vyšších metrik než náhodný přístup.

Míra zlepšení se odvíjí od aplikovaného případu užití. Problematika, ve které se dosahuje úspěšnosti přes 90 %, obvykle neprokáže tak výrazné zlepšení jako v oblasti s nižšími úspěšnostmi. Největším problémem frameworku je stanovit vhodné metriky, které stojí za výpočtem vektoru vah. Vektor vah má zásadní dopad na výsledné metriky modelu, jelikož ovlivňuje modifikaci datové sady, na které je model natrénovaný. Dalším úskalím je nedeterministický výběr dat při modifikaci. Do budoucna by bylo vhodné provést robustnější experimenty, které důkladněji vyvrátí faktor náhodnosti. Dalším řešením náhodnosti je zakomponování deterministického výběru dat, například odstraňováním dat od středu shluku a zachovat jeho hranice. V neposlední řadě by bylo vhodné provést aplikování frameworku na více případech užití a podložit tak pozitivní dopad navrženého řešení.

Kapitola 7

Závěr

Cílem práce bylo vytvořit framework, který modifikuje datové sady. Modifikace probíhá s využitím strojového učení. Přístup se snaží efektivně odstraňovat redundanci dat a nebo naopak doplnit chybějící data. Provádí se cílená modifikace za pomocí shlukování. Framework je nezávislý na případu užití. Výstupem frameworku je nová datová sada, která má pozitivní dopad na trénování modelu.

Řešení předcházelo nastudování teorie na aplikování strojového učení se zaměřením na shlukovací algoritmy. Byla rozebrána existující řešení pro modifikaci datových sad a uvedena problematika určování kvality datových sad. Součástí teoretické části byla sumarizace zdrojů dat v síťové oblasti.

Byl navržen přístup, který analyzuje datovou sadu pomocí shlukovacího algoritmu. Datová sada je rozdělena na shluky s použitím vstupních atributů. Počet shluků je určen automaticky zvolenou heuristikou. Každý shluk je popsán statistickými metrikami, které charakterizují daný shluk. Statistické metriky společně s aplikováním původního modelu slouží pro automatizovaný výpočet skóre shluku. Normalizované skóre udává váhu shluku, tedy jaký význam má daný shluk pro trénování. Čím vyšší váha shluku, tím větší důraz se klade na zachování dat v tomto shluku. Datová sada je v modifikační fázi upravována na základě vektoru vah. Pod-vzorkování jednotlivých shluků je prováděno náhodně s ohledem na váhu každého shluku.

Vytvořený framework umožňuje redukci datové sady, s minimálním dopadem na kvalitu výsledného modelu. Redukce datové sady zmírní výpočetní nároky na trénování a vytvoří se prostor pro nová data. Další funkcionalitou modelu je efektivní slučování datových sad. Při spojování dojde k redukci redundantních dat a následně jsou přidána nová data, která například reflektují aktuální provoz na sítí. Tento přístup zajistí tvorbu kompaktní datové sady, která obsahuje jak aktuální, tak i stará data, aniž by velikost datové sady rostla. Experimenty prokázaly, že aplikováním frameworku získáme datovou sadu, která umožňuje natrénovat model s vyššími metrikami na referenčních datech, než je tomu u porovnávaných metod. Při redukci byl v porovnání s existujícími řešeními zaznamenán rozdíl v úspěšnosti, který v některých případech přesahoval 10 %. V rámci slučování dosáhl framework průběžného zlepšení úspěšnosti o 0.7 % v problematice, kde byla dosahována úspěšnost přes 90 %.

Do budoucna by bylo vhodné aplikovat framework na více případech užití a provést robustnější porovnání s existujícími metodami. Stále existuje prostor pro zlepšení metrik pro výpočet skóre, které jsou pro framework zásadní. Jako další krok vývoje by bylo přínosné zavést více deterministický způsob modifikace, namísto použití váženého náhodného vzorkování.

Literatura

[1] Alpaydin, E. a Bach, F. *Introduction to Machine Learning*. 3rd ed. Cambridge: MIT Press, 2014. Adaptive computation and machine learning. ISBN 0262028182.

[2] Chang, M. *4 stages of the Machine Learning (ML) modeling cycle*. Mar 2022. Dostupné z: https://www.linkedin.com/pulse/4-stages-machine-learning-ml-modeling-cycle-maurice-chang.

[3] Chawla, N. V. Data Mining for Imbalanced Datasets: An Overview. In: Maimon, O. a Rokach, L., ed. *Data Mining and Knowledge Discovery Handbook*. Boston, MA: Springer US, 2010, s. 875–886. DOI: 10.1007/978-0-387-09823-4_45. ISBN 978-0-387-09823-4. Dostupné z: https://doi.org/10.1007/978-0-387-09823-4_45.

[4] Chawla, N. V., Bowyer, K. W., Hall, L. O. a Kegelmeyer, W. P. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*. 2002, sv. 16, s. 321–357.

[5] Chawla, N. V., Bowyer, K. W., Hall, L. O. a Kegelmeyer, W. P. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*. 2002, sv. 16, s. 321–357.

[6] Chen, J.-A., Niu, W., Ren, B., Wang, Y. a Shen, X. Survey: Exploiting Data Redundancy for Optimization of Deep Learning. *ACM Comput. Surv.* New York, NY, USA: Association for Computing Machinery. feb 2023, sv. 55, č. 10. DOI: 10.1145/3564663. ISSN 0360-0300. Dostupné z: https://doi.org/10.1145/3564663.

[7] Claise, B., Trammell, B. a Aitken, P. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. Září 2013.

[8] developers imbalanced-learn. *3. under-sampling*. 2014. Dostupné z: https://imbalanced-learn.org/stable/under_sampling.html.

[9] Domensino, A.-F., Winkens, I., Haastregt, J., Van Bennekom, C. a Heugten, C. Defining the content of a minimal dataset for acquired brain injury using a Delphi procedure. *Health and Quality of Life Outcomes*. Únor 2020, sv. 18. DOI: 10.1186/s12955-020-01286-3.

[10] Douzas, G., Bacao, F. a Last, F. Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE. *Information Sciences*. Elsevier. 2018, sv. 465, s. 1–20.

[11] Gudivada, V., Apon, A. a Ding, J. Data Quality Considerations for Big Data and Machine Learning: Going Beyond Data Cleaning and Transformations. *International Journal on Advances in Software*. Červenec 2017, sv. 10, s. 1–20.

[12] H2O. *K-means clustering*. H2O.ai, 2016. Dostupné z: https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/k-means.html.

[13] Han, H., Wang, W.-Y. a Mao, B.-H. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: Springer. *Advances in Intelligent Computing: International Conference on Intelligent Computing, ICIC 2005, Hefei, China, August 23-26, 2005, Proceedings, Part I 1*. 2005, s. 878–887.

[14] Han, J. a Kamber, M. *Data mining: concepts and techniques*. 3rd ed. Burlington, MA: Elsevier, 2012. ISBN 9780123814791.

[15] Hart, P. The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory*. Citeseer. 1968, sv. 14, č. 3, s. 515–516.

[16] He, H., Bai, Y., Garcia, E. A. a Li, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In: IEEE. *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. 2008, s. 1322–1328.

[17] Hofstede, R., Čeleda, P., Trammell, B., Drago, I., Sadre, R. et al. Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE COMMUNICATIONS SURVEYS AND TUTORIALS*. 2014, sv. 16. DOI: http://dx.doi.org/10.1109/COMST.2014.2321898. ISSN 1553-877X. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6814316.

[18] Ivan, T. Two modifications of CNN. *IEEE transactions on Systems, Man and Communications, SMC*. 1976, sv. 6, s. 769–772.

[19] Kenyon, A., Deka, L. a Elizondo, D. Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets. *Computers & Security*. Prosinec 2020, sv. 99, s. 102022. DOI: 10.1016/j.cose.2020.102022.

[20] Laurikkala, J. Improving identification of difficult small classes by balancing class distribution. In: Springer. *Artificial Intelligence in Medicine: 8th Conference on Artificial Intelligence in Medicine in Europe, AIME 2001 Cascais, Portugal, July 1–4, 2001, Proceedings 8*. 2001, s. 63–66.

[21] Lemaître, G., Nogueira, F. a Aridas, C. K. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research*. 2017, sv. 18, č. 17, s. 1–5. Dostupné z: http://jmlr.org/papers/v18/16-365.html.

[22] Mahendru, K. *How to determine the optimal K for K-means?* Analytics Vidhya, Jun 2019. Dostupné z: https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb.

[23] Mani, I. a Zhang, I. KNN approach to unbalanced data distributions: a case study involving information extraction. In: ICML. *Proceedings of workshop on learning from imbalanced datasets*. 2003, sv. 126, s. 1–7.

[24] Menardi, G. a Torelli, N. Training and assessing classification rules with imbalanced data. *Data mining and knowledge discovery*. Springer. 2014, sv. 28, s. 92–122.

[25] Nguyen, H. M., Cooper, E. W. a Kamei, K. Borderline over-sampling for imbalanced data classification. *International Journal of Knowledge Engineering and Soft Data Paradigms*. Inderscience Publishers. 2011, sv. 3, č. 1, s. 4–21.

[26] Nilsson, N. J. *Introduction machine learning - stanford university*. 1998. Dostupné z: https://ai.stanford.edu/~nilsson/MLBOOK.pdf.

[27] Rafique, D. a Velasco, L. Machine learning for network automation: overview, architecture, and applications [Invited Tutorial]. *Journal of Optical Communications and Networking*. 2018, sv. 10, č. 10, s. D126–D143. DOI: 10.1364/JOCN.10.00D126.

[28] Rokach, L. a Maimon, O. Clustering Methods. In: Maimon, O. a Rokach, L., ed. *Data Mining and Knowledge Discovery Handbook*. Boston, MA: Springer US, 2005, s. 321–352. DOI: 10.1007/0-387-25465-X_15. ISBN 9780387254654. Dostupné z: https://doi.org/10.1007/0-387-25465-X_15.

[29] Sarker, I. H. Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*. Mar 2021, sv. 2, č. 3, s. 160. DOI: 10.1007/s42979-021-00592-x. ISSN 2661-8907. Dostupné z: https://doi.org/10.1007/s42979-021-00592-x.

[30] Setinský, J. *Detekce škodlivých doménových jmen*. Brno, CZ, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/23737/.

[31] Sharma, N. a Saroha, K. Study of dimension reduction methodologies in data mining. In: *International Conference on Computing, Communication & Automation*. 2015, s. 133–137. DOI: 10.1109/CCAA.2015.7148359.

[32] Smith, M. R., Martinez, T. a Giraud Carrier, C. An instance level analysis of data complexity. *Machine learning*. Springer. 2014, sv. 95, s. 225–256.

[33] Soukup, D., Tisovčík, P., Hynek, K. a Čejka, T. Towards Evaluating Quality of Datasets for Network Traffic Domain. In: *2021 17th International Conference on Network and Service Management (CNSM)*. 2021, s. 264–268. DOI: 10.23919/CNSM52442.2021.9615601.

[34] Tisovčík, P. *Odhalování kybernetických útoků s využitím technik strojového učení (Dátové sady)* [online]. Říjen 2020 [cit. 2021-01-27]. https://docs.google.com/presentation/d/1s0REtKg5xJDsqds_scf4Io_k-m5k2kCiBPQtwD3_W2k/edit?us=sharing". Dostupné z: https://docs.google.com/presentation/d/1s0REtKg5xJDsqds_scf4Io_k-m5k2kCiBPQtwD3_W2k/edit?ussharing.

[35] Tomek, I. AN EXPERIMENT WITH THE EDITED NEAREST-NIEGHBOR RULE. 1976.

[36] Vekshin, D., Hynek, K. a Cejka, T. DoH Insight: Detecting DNS over HTTPS by Machine Learning. In: *Proceedings of the 15th International Conference on Availability, Reliability and Security*. New York, NY, USA: Association for Computing Machinery, 2020. ARES '20. DOI: 10.1145/3407023.3409192. ISBN 9781450388337. Dostupné z: https://doi.org/10.1145/3407023.3409192.

[37] Wilson, D. L. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*. IEEE. 1972, č. 3, s. 408–421.

Příloha A

Obsah přiloženého paměťového média - CD

Na paměťovém médiu se nacházejí následující adresáře:

- **/Text DP** - Zdrojové soubory práce pro LATEX a práce ve formátu PDF.
- **/Modifikační framework** - Zdrojové soubory modifikačního frameworku a manuál.