FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Klasifikace typu zařízení ze síťového provozu ISP sítě pomocí shlukovacích metod
Student:	Karel Mudruňka
Vedoucí:	Ing. Josef Koumar
Studijní program:	Informatika
Obor / specializace:	Umělá inteligence 2021
Katedra:	Katedra aplikované matematiky
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

Nastudujte problematiku monitorování počítačových sítí a problematiku analýzy časových řad se zaměřením na jejich shlukování (angl. time series clustering) [1-2]. Prozkoumejte vedoucím vytvořený dataset časových řad obsahující volumetrické informace o IP adresách z vysokorychlostní ISP sítě a anotaci typu zařízení (server, end-device, network device,...). Na základě experimentů vyberte vhodnou shlukovací metodu či metody a vyhodnoťte použitelnost shlukovacích metod na časové řady volumetrických informací z vysokorychlostní ISP sítě. Vytvořte softwarový prototyp zvolené klasifikační metody na základě experimentů (například ve formě jupyter notebooku).

[1] Liao, T. Warren. "Clustering of time series data—a survey." Pattern recognition 38.11 (2005): 1857-1874.
[2] Aghabozorgi, Saeed, Ali Seyed Shirkhorshidi, and Teh Ying Wah. "Time-series clustering–a decade review." Information systems 53 (2015): 16-38.

Elektronicky schválil/a Ing. Magda Friedjungová, Ph.D. dne 1. listopadu 2023 v Praze.

Bakalářská práce

KLASIFIKACE TYPU ZAŘÍZENÍ ZE SÍŤOVÉHO PROVOZU ISP SÍTĚ POMOCÍ SHLUKOVACÍCH METOD

Karel Mudruňka

Fakulta informačních technologií
Katedra aplikované matematiky
Vedoucí: Ing. Josef Koumar
14. května 2024

Odkaz na tuto práci: Mudruňka Karel. *Klasifikace typu zařízení ze síťového provozu ISP sítě pomocí shlukovacích metod*. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2024.

Obsah

Seznam obrázků

Seznam tabulek

Chtěl bych poděkovat především Ing. Josefu Koumarovi za jeho čas strávený vedením této práce, nekonečnou ochotu a cenné rady, které mi při tvorbě práce poskytl. Dále děkuji své rodině za podporu po celou dobu studia. Také děkuji sdružení CESNET za možnost využití výpočetních zdrojů MetaCentra. Výpočetní zdroje poskytl projekt e-INFRA CZ (ID:90254), podpořilo Ministerstvo školství, mládeže a tělovýchovy ČR.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 14. května 2024

Abstrakt

Tato bakalářská práce se zabývá klasifikací typu zařízení ze síťového provozu na základě volumetrických informací o jejich síťové komunikaci pomocí shlukování. Datová sada poskytnutá k hledání vhodného modelu je tvořena časovými řadami informací o síťové komunikaci jednotlivých zařízení v síti CESNET3. Na základě literatury, struktury dat a experimentů je vybrána vhodná shlukovací metoda pro danou úlohu. Výstupem práce je shlukovací klasifikační model s průměrnou testovací přesností 90 % a makro F1 skórem 0,7. Hlavní předností navrženého modelu je jeho konzistence úspěšnosti predikcí v čase.

Klíčová slova shlukování, časové řady, K-means, klasifikace, síťové toky, makro F1 skóre, Python

Abstract

This bachelor's thesis deals with classification of device type based on volumetric information about their network communication using clustering. The provided dataset consists of time series data containing information about network traffic of individual devices in the CESNET3 network. Based on literature, structure of provided dataset and experiments, an appropriate clustering method is selected for the given task. The proposed method achieved classification accuracy of 90 % and macro F1 score of 0.7. The main advantage of the proposed model is consistency of its success rate of predictions over time.

Keywords clustering, time series, K-means, classification, IP flows, macro F1-score, Python

Seznam zkratek

CESNET3	Czech Education and Scientific Network
CSV	Comma-separated values
DBA	DTW barycenter averaging
DBSCAN	Density-based algorithm for discovering clusters in large spatial databases with noise
DTW	Dynamic time warping
FN	False Negative
FP	False Positive
GDBSCAN	Generalized DBSCAN
IoT	Internet of Things
IP	Internet protocol
IPFIX	IP Flow Information Export
ISP	Internet Service Provider
KNN	K-Nearest Neighbors
OPTICS	Ordering points to identify the clustering structure
P2P	Peer-to-peer
TCP	Transmission Control Protocol
TP	True Positive
TTL	Time To Live
UDP	User Datagram Protocol
WWW	World Wide Web

Úvod

Díky stálému vývoji informačních a telekomunikačních technologií se internet stává nedílnou součástí každodenního života čím dál tím více lidí. Na začátku roku 2024 mělo k internetu přístup přibližně 5,35 miliardy jedinců. Dostupnost internetu se celosvětově během posledních 5 let zvýšila o 28,9 % a od roku 2014 se téměř zdvojnásobila. [1]

Společně s rostoucím počtem uživatelů internetu roste i složitost monitorování a analýzy síťového provozu ze strany ISP (Internet Service Provider). Mezi hlavní důvody pro monitorování a následnou analýzu provozu patří detekce bezpečnostních hrozeb v síti, maximalizace efektivnosti struktury sítě, snaha o porozumění uživatelům a jejich chování, která vede k případnému zavedení úprav za účelem zvýšení kvality poskytovaných služeb.

Vzhledem k stále zvětšujícímu se množství dat a dalším důvodům, např. šifrování komunikace, kvůli kterému nelze z pohledu ISP přímo do komunikace nahlížet, se při analýze síťového provozu stále více a více využívají algoritmy strojového učení [2], které jsou s velkým množstvím dat schopny efektivně pracovat a dokáží predikovat čistě na základě statistických informací o šifrované komunikaci bez potřeby znát její obsah.

Strojové učení lze rozdělit na 2 třídy podle znalosti hodnot vysvětlované proměnné během učení. V případě, že tyto hodnoty známe, snažíme se vytvořit model, který je dokáže co nejlépe predikovat. Takové učení se nazývá supervizované. Mezi často používané modely při supervizovaném učení patří například rozhodovací stromy nebo naivní Bayesův klasifikátor. Naopak pokud hodnoty vysvětlované proměnné neznáme, jedná se o nesupervizované učení, jehož cílem bývá nalezení oblastí v prostoru příznaků, ve kterých je vysoká koncentrace dat a následná snaha pochopit, proč se data nachází zrovna v těchto oblastech. Typicky se jedná o úlohu pro shlukování. Shlukování se ale v kontextu analýzy síťového provozu často vyskytuje i jako model supervizovaného učení. Na základě požadované úlohy se při analýze síťového provozu používá široká škála modelů strojového učení, např. neuronové sítě nebo další modely supervizovaného i nesupervizovaného učení. [3]

Cílem této práce je otestovat použitelnost výše zmíněných shlukovacích algoritmů pro supervizovanou klasifikaci typu zařízení na základě jejich síťového provozu ve vysokorychlostní ISP síti. Na základě literatury a vedoucím práce poskytnuté datové sady bude vybrána shlukovací metoda/metody, které budou při klasifikaci využity. Na závěr bude na základě experimentů vytvořen jupyter notebook, ve kterém bude představen vybraný shlukovací model a jeho finální úspěšnost při predikci.

Kapitola 1

Teoretické základy práce

Na začátku této kapitoly budou v rychlosti představeny metody používané pro monitorování počítačových sítí. Následovat bude vysvětlení konceptů shlukovacích algoritmů a představení jejich využití pro analýzu časových řad. Nakonec budou popsány metody evaluace a předzpracovaní dat využité během experimentu.

1.1 Monitorování počítačových sítí

Monitorování a analýza síťového provozu se společně s rostoucí velikostí sítě stává čím dál důležitějším úkolem. Monitorování sítě například výrazně usnadňuje detekci a následné vyřešení problémů vzniklých v síti, čímž přispívá k zvýšené bezpečnosti a dostupnosti internetových služeb v dané síti. Z tohoto a mnoha dalších důvodů existuje velké množství automatických nástrojů určených k monitorování počítačových sítí. [4]

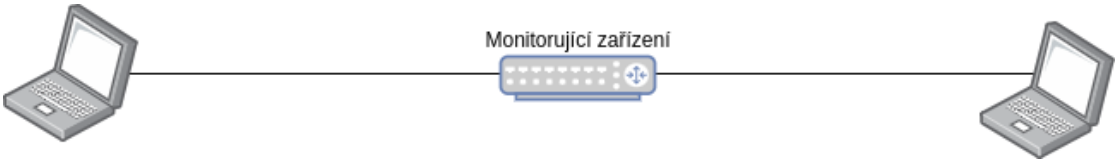
Monitorování počítačových sítí lze rozdělit do dvou následujících skupin:[5]

1. Pasivní monitorování
2. Aktivní monitorování

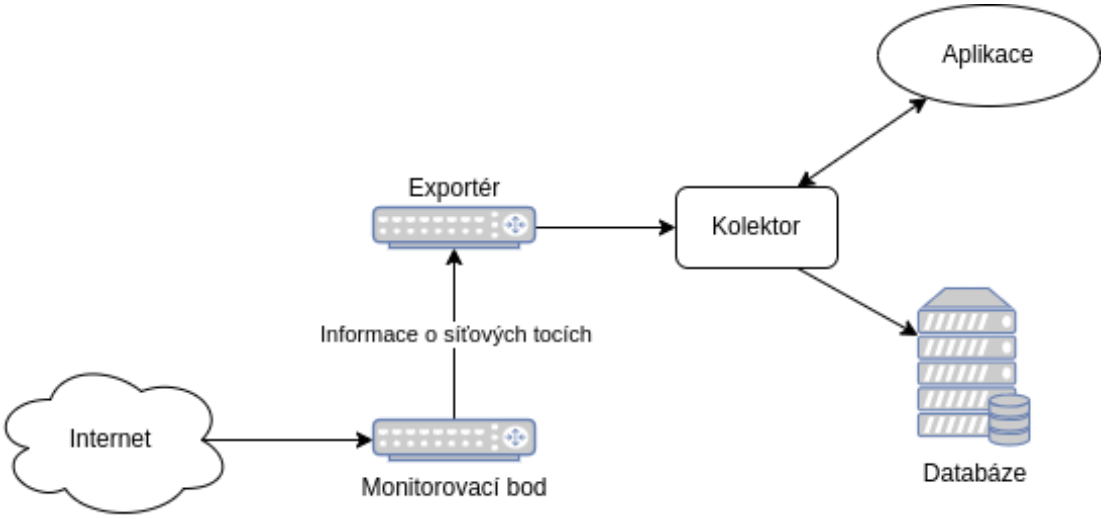
1.1.1 Pasivní monitorování

Techniky pasivního monitorování získávají data čtením síťového provozu, který prochází monitorujícím zařízením jak je znázorněno na obrázku 1.1. Nevytvářejí proto příliš nadbytečného síťového provozu.

Jednou z nejznámějších metod pasivního monitorování počítačových sítí je záchyt paketů (angl. packet capture), během kterého jsou jednotlivě zachyceny všechny pakety procházející monitorujícím zařízením a následně dochází k jejich analýze. V případě monitorování vysokorychlostních počítačových sítí má ale tento přístup problém, neboť se stane velmi náročným po paměťové i výpočetní stránce.



Obrázek 1.1 Architektura pasivního monitorování



Obrázek 1.2 Diagram architektury protokolu IPFIX

Výše zmíněný problém záchytu paketů řeší monitorování pomocí síťových toků (angl. flow monitoring), během kterého jsou zachycené pakety agregovány do síťových toků (angl. IP flows) a následně odesílány k analýze. Z tohoto důvodu je monitorování pomocí síťových toků nejčastěji využívaným řešením pro monitorování vysokorychlostních sítí. Podle Claise et al. [6] je síťový tok definován jako množina IP paketů procházejících monitorujícím zařízením během určitého časového intervalu takových, že všechny pakety patřící do stejného síťového toku mají soubor společných vlastností. Mezi tyto společné vlastnosti často patří obsah hlaviček paketu, například IP adresy zdroje a cíle nebo čísla portů využívaných při komunikaci atd.

I přes to, že existuje několik různých protokolů monitorování pomocí síťových toků, jejich architektura se většinou výrazně neliší. Dle Hofstede et al. [7] se architektura většiny těchto protokolů skládá z následujících síťových prvků:

- Monitorovací bod (angl. monitoring point) je zařízení, které monitoruje provoz síťový provoz, který ním prochází a provádí jeho následné zpracování, to je celé v literatuře označováno jako proces měření (angl. metering process), během kterého dochází k agregaci paketů do jednotlivých síťových toků. V případě, že monitorovací bod a exportér síťových toků netvoří jedno zařízení, jsou po dokončení agregace informace odeslány do nějakého exportéru síťových toků.
- Exportér síťových toků (angl. flow exporter) získává informace o jednotlivých síťových tocích od monitorujících zařízení a následně provádí jejich zapouzdření do zprávy ve formátu definovaném v daném protokolu. Nakonec odešle tyto zprávy nějakému z kolektorů síťových toků. Jak již bylo zmíněno, monitorovací bod a exportér síťových toků jsou běžně zkombinovány v jednom zařízení, poté se celé toto zařízení rovnou nazývá exportér síťových toků. V případě, že exportér síťových toků tvoří samostatné zařízení, je v architektuře označováno za sondu síťových toků (angl. flow probe).
- Kolektor síťových toků (angl. flow collector) přijímá zprávy v specifickém formátu od nějakého exportéru síťových toků, provádí nad získanými daty základní předzpracování a následně je poskytuje dalším aplikacím k jejich podrobné analýze.

Nejpoužívanějšími protokoly monitorování pomocí síťových toků jsou NetFlow a IPFIX (IP Flow Information Export), jehož architektura je pro ilustraci ukázána na obrázku 1.2.

1.1.2 Aktivní monitorování

Metody aktivního monitorování počítačových sítí vytvářejí vlastní testovací síťový provoz, čímž například simulují chování koncových uživatelů. To umožňuje správcům sítě testovat například latenci, míru ztráty paketů (angl. packet loss rate), propustnost a další parametry dané sítě. Snahou aktivního monitorování je zatěžovat samotnou síť co nejméně a zároveň vytvářet provoz, který co nejlépe simuluje běžný provoz v dané síti.

1.2 Shlukovaní

Cílem procesu shlukování je rozdělení datových bodů do jednotlivých tříd (shluků) na základně podobnosti hodnot jejich příznaků tak, aby podobnost hodnot příznaků 2 datových bodů v určitém shluku byla větší než podobnost 2 datových bodů z různých shluků. Kvůli tomu, že pro shlukování není potřeba existence vysvětlované proměnné, případně její znalost, je shlukování využíváno především jako metoda nesupervizovaného učení. Jednoduchým příkladem využití shlukování by mohlo být rozdělení zákazníků e-shopu do několika skupin na základě informací o jejich nákupech, jako jsou průměrná útrata za nákup, pravidelnost nakupování a další. Následně by e-shop na základě analýzy získaných shluků mohl například jednotlivým skupinám nabízet různé zboží za účelem maximalizace zisku. Díky své relativní jednoduchosti a úspěšnosti jsou algoritmy shlukování jednou z vůbec nejvyužívanějších metod strojového učení a nachází uplatnění kromě informatiky i v celé řadě dalších oblastí, kupříkladu v medicíně, geografii, sociálních vědách, ekonomii [8].

Nejdůležitějším pojmem pro shlukovací algoritmy je metrika, neboli vzdálenost 2 bodů. Dle Kumar et al. [9] se mezi nejčastěji používané metriky v shlukovacích algoritmech pro datové body x, y v R^n, kde n je počet příznaků datové sady řadí:

- Euklidovská metrika (angl. Euclidean Distance),

$$D_e(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- Manhattanská metrika (angl. Manhattan Distance),

$$D_{Ma}(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

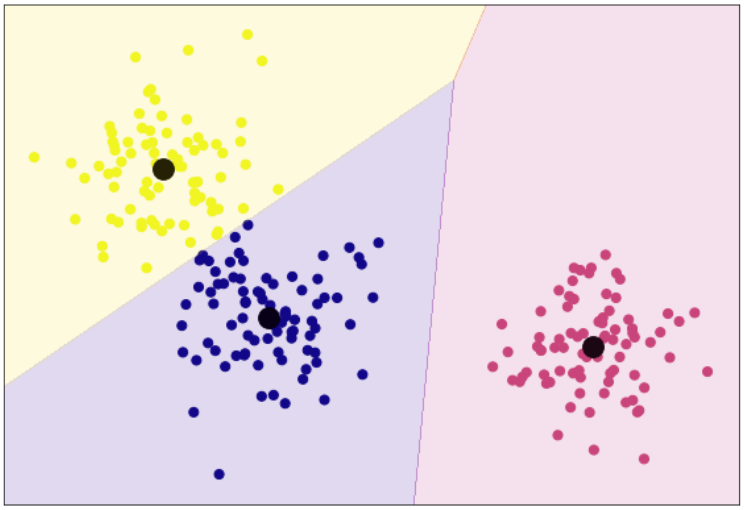
- Canberrská metrika (angl. Canberra Distance),

$$D_{Can}(x, y) = \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|}$$

Definice vzdáleností využívaných při shlukování časových řad, které byly použity během samotného experimentu jsou podrobně popsány v sekci Shlukování časových řad.

Vzhledem k tomu, že pro většinu úloh řešených pomocí shlukovacích algoritmů neexistuje žádné obecné kritérium kvality modelu (jako např. přesnost modelu při klasifikaci), existuje velké množství shlukovacích algoritmů, které se snaží vytvářet shluky různými způsoby a zachytit při tom různé vlastnosti dat. Na základě procesu vytváření jednotlivých shluků a jejich požadovaných vlastností proto můžeme shlukovací algoritmy rozdělit do 3 hlavních skupin. [10]

1. Dělící shlukovací algoritmy (angl. Partitional clustering algorithms)



Obrázek 1.3 Příklad výsledku běhu algoritmu K-means na datech s 2 příznaky vygenerovaných pomocí sklearn.datasets.make_blobs [12], černé body představují centra shluků.

2. Hierarchické shlukování (angl. Hierarchical clustering)
3. Algoritmy shlukování pomocí hustoty (angl. Density-based clustering)

1.2.1 Dělící shlukovací algoritmy

Dělící shlukovací algoritmy, známé spíše pod anglickým názvem *Partitional clustering algorithms*, jsou z pravidla iterační shlukovací algoritmy, které mají na vstupu krom datové sady také maximální počet iterací a počet výsledných shluků. Jejich cílem je minimalizovat celkovou účelovou funkci, často v podobě součtu kvadrátů vzdáleností jednotlivých bodů od center shluku, do kterého spadají. [11]

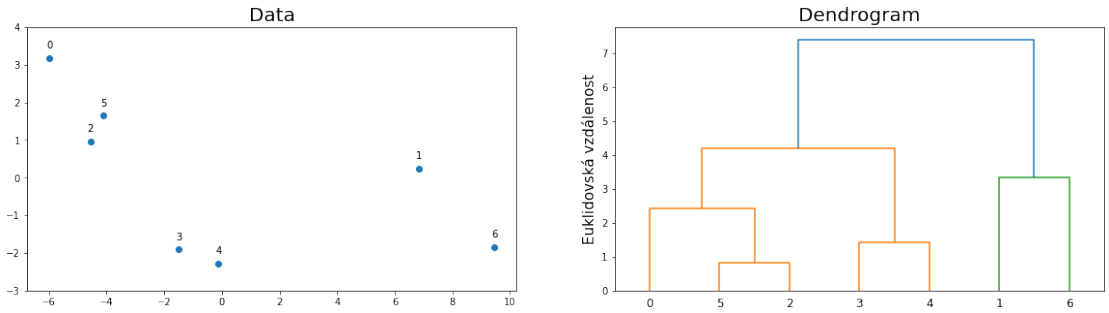
Jednotlivé body přísluší ke shluku, jehož centrum je k nim nejblíže, shluky, jak je vidět na obrázku 1.3, dělí prostor příznaků na k (vstupní parametr algoritmu udávající počet výsledných shluků) oblastí, které představují rozsah hodnot příznaků pro jednotlivé shluky.

Nejdříve se nějakou metodou zinicializují centra jednotlivých shluků, poté dojde v každém kroku iterace k napočítání příslušností jednotlivých datových bodů do shluků, vypočtení hodnoty účelové funkce a následnému přepočtení center clusterů. Iterace těchto algoritmů se zastaví v jedné z následujících situací:

a) Algoritmus dosáhl maximálního počtu iterací
b) Rozdíl hodnot účelové funkce po poslední provedené iteraci a před ní je menší než uživatelem zadaná konstanta na vstupu algoritmu

Hlavní výhodou těchto algoritmů v porovnání s algoritmy z ostatních skupin je, že jsou schopny relativně efektivně tvořit sférické shluky i pro rozsáhlé datové sady s velkým množství příznaků. Naopak jejich nevýhodou je, že vzhledem k procesu tvorby finálních shluků jsou citlivé na odlehlá měření (angl. outliers) a šum [13]. Další nevýhodou je problém počáteční inicializace center shluků. Pro různé počáteční inicializace budou výsledné shluky pravděpodobně diametrálně odlišné, protože algoritmus snadno zkonverguje do lokálního minima účelové funkce. Problematika počáteční inicializace shluků bude vysvětlena v dalších částech práce.

Mezi nejpoužívanější dělící shlukovací algoritmy se řadí:



Obrázek 1.4 Příklad dendrogramu pro hierarchické shlukování na náhodně vygenerovaných datech pomocí sklearn.datasets.make blobs [12]

- K-means
- K-medoids
- Fuzzy clustering

1.2.2 Hierarchické shlukování

Hierarchické shlukování probíhá tak, že na začátku svého běhu označí každý datový bod za samotný shluk. Následně v každé iteraci sloučí 2 sobě nejbližší shluky v jeden nový. Běh algoritmu si lze tedy představit jako tvorbu binárního stromu, jehož listy jsou samotné datové body a každý další vrchol je shluk vzniklý sloučením jeho potomků. Tento strom se tedy staví odspodu nahoru. V případě hierarchického shlukování se takto vzniklý strom běžně nazývá dendrogram. Dendrogram se využívá především k vizualizaci hierarchického shlukování, protože z něj lze snadno a přehledně vyčíst průběh celého procesu tvorby jednotlivých shluků a které body do nich spadají, viz. obrázek 1.4. Algoritmus skončí v jednom z následujících případů:

a) Počet shluků byl snížen na požadovaný počet (nepovinný parametr algoritmu)

b) Algoritmus sloučil všechny body do 1 shluku, čímž vygeneroval celý dendrogram

Celkem se tedy provede maximálně $n - 1$ iterací algoritmu, kde n představuje počet datových bodů v datové sadě.

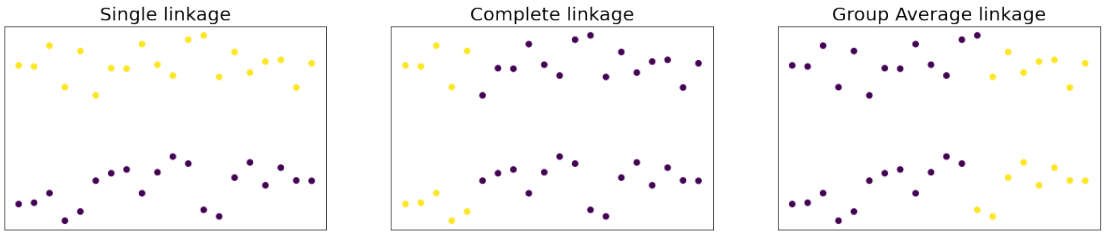
Pro hierarchické shlukování je nejdůležitější definovat vzdálenost 2 shluků. Na úplném počátku shlukování je to jednoduše vzdálenost 2 datových bodů. V pozdějších fázích, kdy je v jednotlivých shlucích větší množství datových bodů je zvolení správné definice vzdálenosti 2 shluků pro danou úlohu již složitější. Mezi nejčastěji používané definice vzdálenosti shluků X a Y, kde $D(x, y)$ představuje vzdálenost datových bodů $x \in X$ a $y \in Y$: [14]

- Metoda nejbližšího souseda (angl. Single linkage),

$$\Delta(X, Y) = \min_{x \in X, y \in Y} D(x, y)$$

- Metoda nejvzdálenějšího souseda (angl. Complete linkage),

$$\Delta(X, Y) = \max_{x \in X, y \in Y} D(x, y)$$



Obrázek 1.5 Výsledky hierarchického shlukování pro představené metody výpočtu vzdáleností 2 shluků na náhodně vygenerovaných datech

- Metoda průměrné vzdálenosti (angl. Group Average Linkage),

$$\Delta(X,Y) = \frac{1}{|X|+|Y|}\sum_{x\in X}\sum_{y\in Y} D(x,y)$$

Příklady výsledků shlukování s využitím těchto definic vzdáleností dvou shluků jsou na obrázku 1.5.

Největší předností hierarchického shlukování je již zmíněná snadná analýza průběhu a následná interpretovatelnost výsledků díky dendrogramu. Hierarchické shlukování také nevyžaduje na vstupu pevně stanovený počet výsledných shluků. Z dendrogramu lze totiž po dokončení běhu algoritmu snadno vyčíst, jak budou vypadat shluky pro jejich libovolný počet. Přední nevýhodou hierarchického shlukování je výpočetní složitost. Vzhledem k tomu, že pro vygenerování celého stromu potřebuje algoritmus $n-1$ iterací, je jeho celková složitost alespoň $\mathcal{O}(n^2)$. Z tohoto důvodu je nevhodný pro velké datové sady. Dalším problémem v případě velkých datových sad je, že dendrogram bude obsahovat velké množství hran a pravděpodobně kvůli tomu bude nepřehledný. Hierarchické shlukování také není vhodné pro data s velkým počtem příznaků kvůli tzv. prokletí dimenzionality [13]. V případě metody nejvzdálenějšího souseda je také relativně náchylné k problémům způsobeným odlehlými měřeními.

1.2.3 Algoritmy shlukování pomocí hustoty

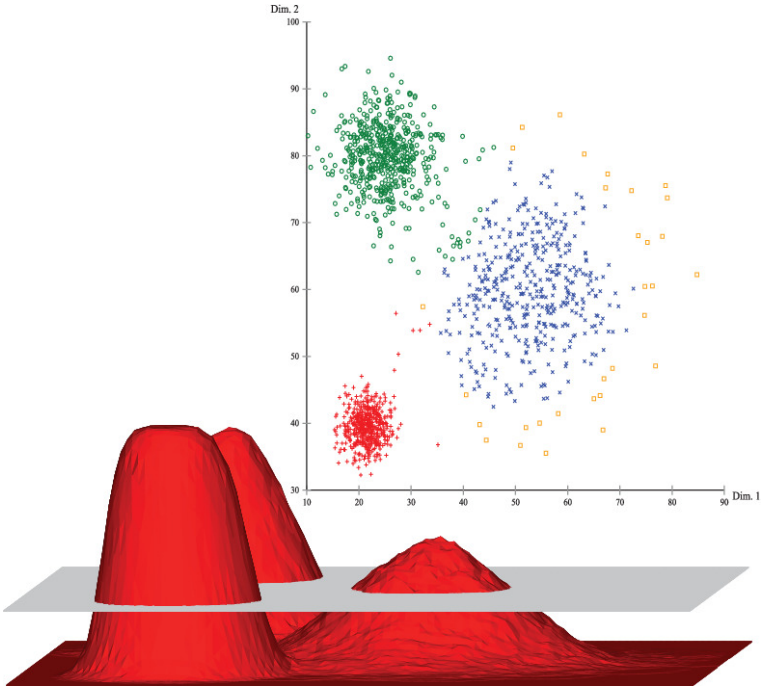
Algoritmy shlukování pomocí hustoty, jak už název vypovídá, jsou algoritmy, které v prostoru příznaků hledají souvislé oblasti s velkou hustotou datových bodů. Tyto oblasti jsou následně označovány za shluky, které jsou mezi sebou odděleny oblastmi s malou hustotou datových bodů, které jsou následně označeny za šum, jak je vidět na obrázku 1.6 [15]. Průběh algoritmů si také můžeme představit jako odhad hustoty pravděpodobnosti výskytu jednotlivých datových bodů v prostoru příznaků. Oblasti s odhadem pravděpodobnosti vyšším než nějaká prahová hodnota považujeme za shluky.

Prahovou hodnotu můžeme u algoritmů nastavit procentuálním množstvím bodů označených jako šum, které je například pro algoritmus DBSCAN doporučené 1 % až 30 % [16]. Jednotlivé algoritmy z této skupiny mají své další své parametry, kterými se dá regulovat prahová hodnota.

Mezi nejznámější algoritmy shlukování pomocí hustoty se řadí: [17]

- DBSCAN
- OPTICS
- GDBSCAN

Největší výhodou algoritmů shlukování pomocí hustoty oproti ostatním představeným shlukovacím algoritmům je, že jako jediné nutně nezařazují všechny body z datové sady do některého ze



Obrázek 1.6 Příklad výsledku běhu algoritmu shlukování pomocí hustoty společně s odhadnutou hustotou pravděpodobnosti. Dostupné z [15]

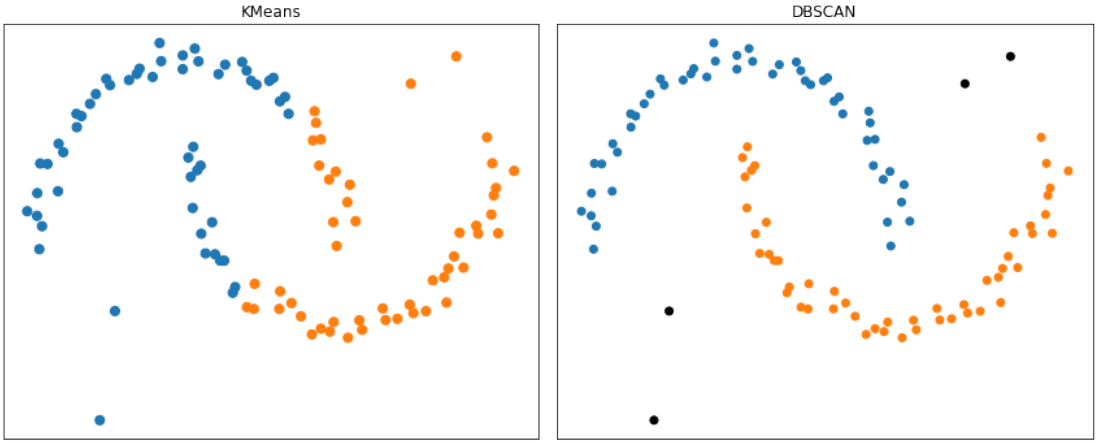
.

shluků a jsou díky tomu odolné vůči problémům způsobeným šumem a odlehlými měřeními. Dále jsou tyto algoritmy schopny úspěšně odhalit shluky libovolného tvaru, s čímž mají dělící shlukovací algoritmy i hierarchické shlukování problémy. Příklad datové sady, pro kterou si algoritmus shlukování pomocí hustoty vedl daleko lépe než algoritmus K-means lze vidět na obrázku 1.7. Tyto algoritmy také nevyžadují na vstupu pevný počet výsledných shluků, protože shluky jsou vytvářeny v průběhu běhu algoritmu na základě hustoty dat v prostoru příznaků. Vyžadují ale jiné parametry, které průběh algoritmu výrazně ovlivňují. Hlavním problémem algoritmů shlukování pomocí hustoty je, že při tvorbě shluků využívají okolí každého z bodů a počet dalších bodů, které se v něm nachází, a proto vyžadují poloměr tohoto okolí jako jeden z parametrů. S rostoucí dimenzionalitou dat roste i vzdálenost mezi jednotlivými datovými body. Proto pro data s velkým počtem příznaků může být velmi obtížné správně určit zmíněný poloměr okolí a shlukování pomocí hustoty se tedy pro ně stává nevhodným řešením. Ostatní dříve představené algoritmy tímto konkrétním problémem netrpí. [13]

1.3 Shlukování časových řad

Jedním z nejčastěji využívaných typů shlukování je shlukování časových řad. Časová řada je posloupnost jakýchkoliv pozorování, nejčastěji seřazených podle času. Časové řady jsou oblíbeným datovým formátem, protože umožňují uchovávat dynamické informace a jejich změnu jako funkci času, díky čemuž jsou využívány v celé řadě přírodních i sociálních věd. [18]

Na obrázku 1.8 je vyobrazen příklad časové řady počtu odeslaných paketů ze zařízení s agregací po 10 minutách. Časová řada na obrázku 1.8 má pouze jeden příznak, ale stejně jako statická data mají běžně časové řady několik příznaků, které mohou představovat kategorické i spojité hodnoty. Časové řady také běžně mívají různá časová měřítka a chybějící hodnoty měření v některých časech, často mají tedy různé délky. Před shlukovou analýzou je tedy běžné provést



Obrázek 1.7 Rozdíl výsledku běhů algoritmu K-means a DBSCAN na datech vygenerovaných pomocí sklearn.datasets.make_moons [12]. Černě označené body pro DBSCAN graf byly algoritmem označeny za šum.

předzpracování dat, které zahrnuje např. sjednocení časového měřítka pro všechny časové řady z datové sady. [19]

Dle Zolhavarieh et al. [20] lze shlukování časových řad rozdělit do 3 následujících kategorií:

1. Shlukování celých časových řad (angl. Whole time-series clustering) je shlukování, kdy v datové sadě časových řad považujeme časové řady za jednotlivé body. Shlukování tedy probíhá na základě podobnosti celých časových řad.

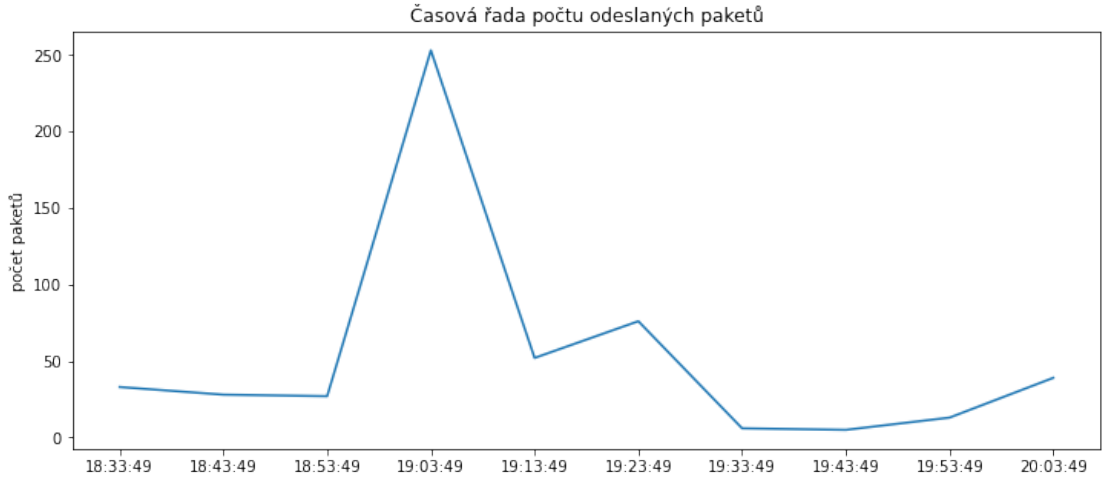
2. Shlukování podposloupností časových řad (angl. Subsequence time series clustering) požaduje na vstupu pouze 1 časovou řadu, ze které vybere několik podposloupností s využitím klouzavého okénka (angl. Sliding window) a následně provede shlukování na získaných podposloupnostech. Využívá se tedy především pro dlouhé časové řady.

3. Shlukování časových bodů (angl. Time point clustering) také využívá pouze jednu časovou řadu, nad kterou provede shlukování na základě kombinace diference časů dvou pozorování a podobnosti těchto dvou pozorování.

Vzhledem ke struktuře datové sady se pro shlukovou analýzu zpracovanou v této práci bude využívat shlukování celých časových řad. Proto bude nadále termín shlukování časových řad používán ve smyslu shlukování celých časových řad.

Jeden datový bod z datové sady časových řad si můžeme představit jako jednu datovou sadu statických dat, kde index představuje časové měřítko. Z tohoto důvodu bývá shluková analýza datových sad časových řad paměťově i časově náročná. I přes to je shlukování časových řad v praxi velmi využívaným nástrojem, protože dokáže v časových řadách objevit často opakující se vzory a další souvislosti, které by jinak pro člověka bylo nemožné objevit vzhledem k velikosti datových sad. Shlukování se také používá jako jeden z několika kroků složitějších algoritmů pro analýzu časových řad. [21]

Při shlukování časových řad se využívají algoritmy velmi podobné algoritmům pro shlukování statických dat. Buď se nějaký z algoritmů pro statická data nějakým způsobem upraví, aby byl schopen pracovat s časovými řadami, nebo se upraví datová sada tak, aby jednotlivé časové řady měly vlastnosti statických dat a následně se na datovou sadu použije jeden z algoritmů pro shlukování statických dat.



Obrázek 1.8 Příklad časové řady počtu odeslaných paketů ze zařízení

1.3.1 Míry podobnosti časových řad

Nejčastější úpravou je zavedení míry podobnosti, která představuje vzdálenost mezi dvěma časovými řadami a následně je v klasických shlukovacích algoritmech používána stejně jako vzdálenost dvou datových bodů v případě statických dat. Efektivně se tedy jedná spíše o míry rozdílnosti časových řad. Platí tedy, že čím menší je míra podobnosti, tím spíše budou dvě časové řady společně v jednom shluku. Existuje celá řada různých způsobů výpočtu vzdálenosti časových řad, které se zaměřují na jejich různé vlastnosti. Pro shlukování časových řad jsou nejčastěji využívanými metodami Euklidovská vzdálenost a DTW (Dynamic time warping). [21]

Euklidovská vzdálenost

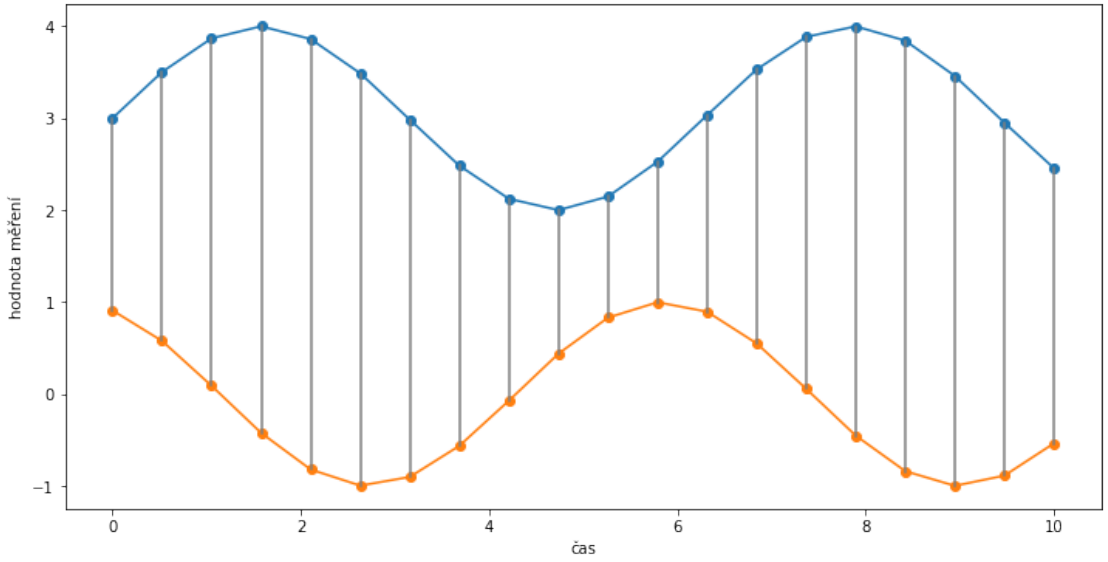
Euklidovská vzdálenost je nejjednodušší metoda výpočtu podobnosti časových řad. Pokud budeme mít dvě stejně dlouhé časové řady $X = (x_1, x_2, \ldots, x_T)$, $Y = (y_1, y_2, \ldots, y_T)$ o délce T s P příznaky, můžeme Euklidovskou vzdálenost časových řad X a Y vypočítat pomocí následujícího vzorce:

$$D_E(X, Y) = \sqrt{\sum_{i=1}^{T} \sum_{p=1}^{P} (x_{ip} - y_{ip})^2}$$

Tento výpočet je pro časové řady s jedním příznakem graficky znázorněn na obrázku 1.9

Největší výhodou Euklidovské vzdálenosti pro časové řady je výpočetní složitost. Ta je lineární vzhledem k délce zpracovávaných časových řad. Díky tomu je vhodná i pro shlukovou analýzu velkých datových sad. Další výhodou je snadná implementace. Nevýhodou Euklidovské vzdálenosti je, že na vstupu požaduje obě časové řady stejné délky. V praxi je ale běžné, že porovnávané časové řady mají chybějící hodnoty, a proto je často nutné před využitím Euklidovské vzdálenosti provést nějaké předzpracování datové sady, např. doplnění chybějících hodnot nebo časové řady agregovat po delších intervalech.

Další problém nastává v případě, kdy časové řady mají podobný průběh, ale jedna je vůči druhé posunuta o nějaký počet indexů. Protože Euklidovská vzdálenost vždy porovnává měření stejného indexu, vzdálenost takto posunutých časových řad bude velká a časové řady budou pravděpodobně v jiných shlucích, i přes to, že jsou až na posun vizuálně velmi podobné. Můžeme si např. všimnout, že pro minimalizaci Euklidovské vzdálenosti časových řad vyobrazených na 1.9



Obrázek 1.9 Grafické znázornění porovnávaných bodů při výpočtu Euklidovské vzdálenosti dvou časových řad, datové body modré časové řady byly vygenerovány funkcí $f(t) = sin(t) + 3$, $t \in \{0, 0.5, \ldots, 9.5, 10\}$ a datové body oranžové časové řady byly vygenerovány pomocí funkce $f(t) = sin(t + 2)$ pro t ze stejné množiny.

bychom museli modře zobrazenou časovou řadu posunout o dvě jednotky času zpět. Euklidovská vzdálenost má také problém v případě, že některé z měření v časové řadě má výrazně odlišné hodnoty oproti ostatním měřením, výsledná hodnota vzdálenosti tímto měřením bude hodně ovlivněna.

I přes výše zmíněné nedostatky se ale ukazuje, že při klasifikaci má Euklidovská vzdálenost podobnou úspěšnost jako další míry podobnosti, které těmito problémy netrpí. [21]

DTW

DTW je algoritmus pro výpočet podobnosti časových řad, který vznikl za účelem porovnání často opakujících se vzorů při strojovém zpracování řeči [22]. Díky tomu, že se dokáže efektivně vypořádat s posuny mezi časovými řadami, se DTW v posledních letech začalo využívat v celé řadě úloh strojového učení spojených s časovými řadami.

V knihovně tslearn [23] je implementován DTW pro dvě časové řady $X = (x_1, x_2, \ldots, x_M)$, $Y = (y_1, y_2, \ldots, y_N)$ s P příznaky dle pseudokódu 1, kde $d(x_i, y_j)$ představuje Euklidovskou vzdáleností měření x_i a y_j ve tvaru $d(x_i, y_j) = \sqrt{\sum_{p=1}^{P} (x_{ip} - y_{jp})^2}$.

Algoritmus nejdříve v krocích 1 až 5 vyplní cenovou matici C (Accumulated Cost Matrix) o rozměrech $(M + 1, N + 1)$ hodnotami nekonečno. Následně ji v krocích 6 až 10 znovu vyplní s využitím dynamického programování tak, že $C[i, j]$ obsahuje kvadrát minimální dosažitelné vzdálenosti mezi časovými řadami $(x_1, \ldots, x_i)$ a $(y_1, \ldots, y_j)$ pro všechny možné indexové posuvy jednotlivých měření takové, že do vzdálenosti musí být vždy zahrnuta $d(x_1, y_1)$ a $d(x_i, y_j)$. Po vyplnění celé matice bude tedy $\sqrt{C[M, N]}$ představovat optimální hodnotu pro vzdálenost X a Y za podmínky, že jsou v ní obsažené $d(x_1, y_1)$ a $d(x_M, y_N)$.

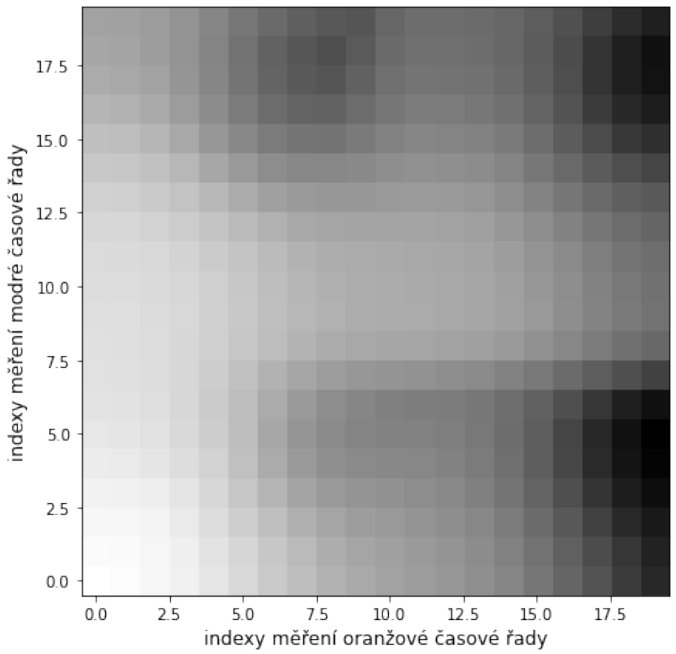
Příklad teplotní mapy (angl. heatmap) vypočtené cenové matice pro časové řady z obrázku 1.9 lze vidět na obrázku 1.10. Ze znalosti cenové matice pro dvě časové řady délek M a N lze snadno získat tzv. *optimal warping path*. [22]

Optimal warping path představuje posloupnost $p = (p_1, \ldots, p_L)$ dvojic $p_l = (n_l, m_l)$ indexů takových, že druhá mocnina vzdálenosti časových řad X a Y vypočtená pomocí DTW je rovna

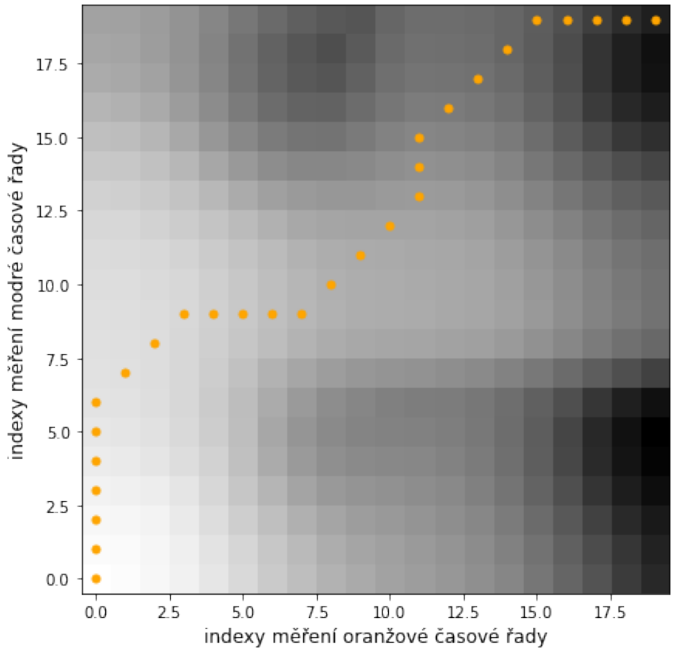
Algoritmus 1 DTW

Vstup: Časové řady X, Y o délce M, N

for $i = 0$ **to** M **do**
  for $j = 0$ **to** N **do**
    $C[i, j] \leftarrow \inf$
  end for
end for
for $i = 1$ **to** M **do**
  for $j = 1$ **to** N **do**
    $C[i, j] \leftarrow d(x_i, y_i)^2 + min(C[i-1, j], C[i, j-1], C[i-1, j-1])$
  end for
end for
return $\sqrt{C[i, j]}$



Obrázek 1.10 Teplotní mapa cenové matice při výpočtu podobnosti pomocí DTW pro časové řady z 1.9, tmavější barvy představují vyšší hodnotu



Obrázek 1.11 Znázornění *optimal warping path* na cenové matici z obrázku 1.10.

součtu vzdáleností n-tého měření z X a m-tého měření z Y přes všechny $l \in L$. To je graficky znázorněno na obrázku 1.12. Na obrázku 1.11 je pro porovnání zobrazena *optimal warping path* na cenové matici. [22]

DTW je velmi populárním algoritmem pro výpočet podobnosti časových řad především díky tomu, že při výpočtu na rozdíl od Euklidovské vzdálenosti uvažuje indexové posuny bodů časových řad, díky čemuž je tento algoritmus schopen mnohem lépe zachytit podobnost tvarů časových řad. Další výhodou DTW oproti Euklidovské vzdálenosti je schopnost zpracovat časové řady různých délek. To může například usnadnit proces předzpracování dat. Na druhou stranu největším nedostatkem DTW je časová složitost. [24]

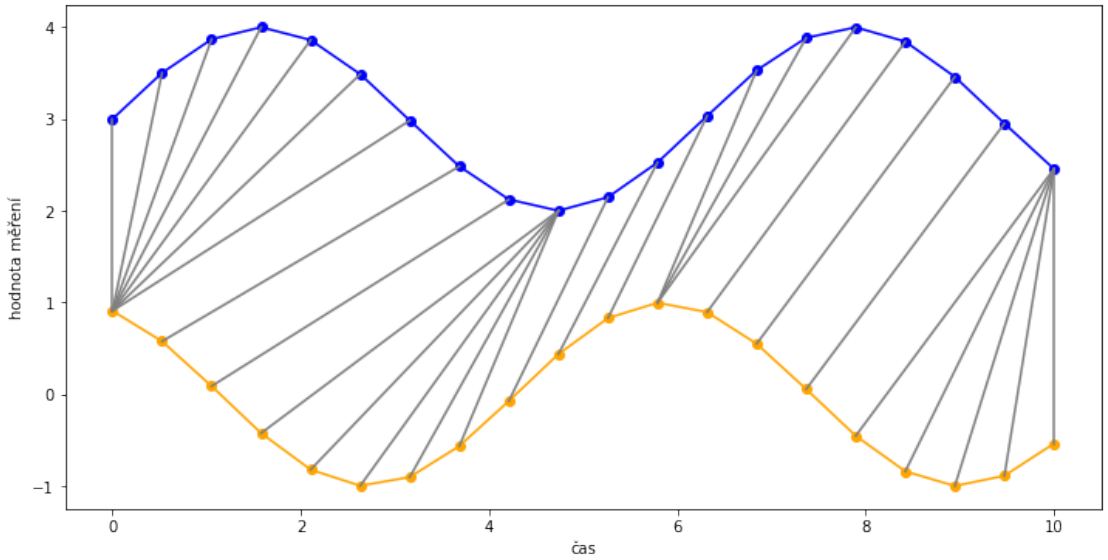
Pří výpočtu Euklidovské vzdálenosti je nutné napočítat jednotlivě vzdálenost N datových bodů, kde N představuje délku časových řad a následně tyto vzdálenosti sečíst. Časová složitost Euklidovské vzdálenosti je tedy vzhledem k délce časových řad $\mathcal{O}(N)$. Algoritmus DTW musí pro časové řady délek m a n vyplnit celou matici rozměrů (m, n). Pro každý prvek matice musí DTW vypočítat vzdálenost 2 datových bodů a sečíst ji s jedním ze tří už předpočítaných prvků matice. Časová složitost DTW je proto $\mathcal{O}(N \cdot M)$ [25]. Z tohoto důvodu může být DTW nevhodné pro velmi dlouhé časové řady a datové sady s velkým množstvím časových řad. Ve snaze snížit časovou složitost DTW bylo představeno několik omezení, dvěma nejpoužívanějšími jsou: [23]

- *Sakoe-Chiba band* je omezení charakterizované poloměrem r, který udává počet prvků matice nad a pod vedlejší diagonálou, které jsou algoritmem uvažovány.
- *Itakura parallelogram* vytváří omezení algoritmem uvažovaných prvků matice ve tvaru rovnoběžníku, jehož tvar a sklon lze regulovat parametrem s (itakura max slope).

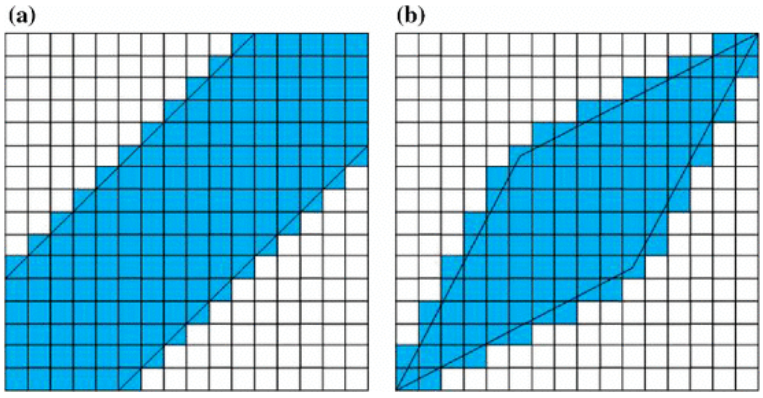
Představené omezení jsou graficky znázorněny na obrázku 1.13.

1.3.2 K-means

Algoritmus K-means je nejznámější shlukovací algoritmus ze třídy dělících shlukovacích algoritmů. Díky své jednoduchosti a v porovnání s ostatními algoritmy pro shlukování časových řad



Obrázek 1.12 Grafické znázornění porovnávání bodů při výpočtu podobnosti pomocí DTW na časových řadách z obrázku 1.9.



Obrázek 1.13 Představené omezení pro DTW, vlevo Sakoe-Chiba band (a), vpravo Itakura parallelogram (b). Dostupné z [26]

relativně malé časové složitosti se K-means uplatnil ve velkém množství experimentů zahrnujících shlukování časových řad. Mnohé experimenty ukazují, že K-means je vhodným řešením v případě datových řad s velkým množstvím časových řad. [21, 19, 27, 28]

Účelovou funkcí, kterou se algoritmus K-means snaží minimalizovat, je součet kvadrátů vzdáleností všech bodů od center shluků, do kterých jednotlivé body přísluší. Průběh samotného algoritmu lze shrnout do tří následujících kroků:

1. Přiřazení každého bodu do shluku, jehož centrum je nejblíže
2. Přepočítání center shluků na základě datových bodů v nich (nejčastěji průměrem hodnot)
3. Opakování kroků 1 a 2 dokud nedojde k překročení maximálního počtu iterací, nebo rozdíl hodnot účelové funkce v iteracích s indexy i a $i-1$ je menší než nějaká prahová hodnota.

Jednou z věcí, které výrazně algoritmus a jeho výsledek ovlivní, je počáteční inicializace center shluků. Častým přístupem je náhodná inicializace. Problémem náhodné inicializace ale bývá, že počet iterací potřebných k vytvoření kvalitních shluků může být velký. Dalším problémem náhodné inicializace je, že algoritmus snadno zkonverguje do lokálního minima a při opakovaném běhu bude vracet velmi rozdílné shluky. [29] Z tohoto důvodu bylo vytvořeno několik metod počáteční inicializace, které zmíněnými problémy netrpí.

Nejčastěji implementovaným a velmi spolehlivým inicializačním algoritmem je greedy K-means++, v některých knihovnách označený pouze jako K-means++ [12]. Tento algoritmus pro K shluků nejdřív na základě odhadu pravděpodobnosti výskytu dat v prostoru vybere $log(K)$ datových bodů jako centra shluků a jako další centra zbylých shluků označuje datové body, které jako centra shluků minimalizují účelovou funkci. [30]

1.4 Metody předzpracování dat

Při analýze časových řad je vzhledem k povaze dat většinou výhodné využít nějakou z metod předzpracování dat. Pro časové řady existuje velké množství různých technik předzpracování dat. Mezi nejčastěji používané se řadí například metody doplnění chybějících hodnot, detekce a odstranění šumu nebo vyhlazení časových řad. V této části budou popsány pouze metody vybrané na základě povahy datové sady, která bude v dalších částech práce představena a následně zpracována.

Normalizace

Normalizace je proces transformace dat za účelem změnit jejich měřítko. Často je transformace cílena na hodnoty z intervalu $[0, 1]$ nebo $[-1, 1]$ [31]. Normalizace je výhodná především v případě, kdy jednotlivé příznaky dat mají výrazně odlišná měřítka. V případě odlišných měřítek příznaků nastává pro shlukování a velké množství dalších modelů strojového učení problém, kdy jsou výsledky nejvíce ovlivněny příznaky s velkým měřítkem a váha ostatních příznaků je výrazně menší, i přes to, že pro danou úlohu mohou mít daleko větší informační hodnotu než příznaky s velkými měřítky.

Nejčastěji používanou metodou normalizace je min-max normalizace. Při min-max normalizaci je každá hodnota x zvoleného příznaku nahrazena hodnotou x' získanou jako:

$$x' = \frac{x - min}{max - min}$$

kde min představuje nejmenší hodnotu a max označuje největší hodnotu příznaku před transformací. Mezi další používané metody normalizace patří např. převedení dat na logaritmické měřítko.

Z-skóre standardizace

Z-skóre standardizace je stejně jako normalizace proces transformace měřítka dat. Jejích cílem je transformovat data tak, aby měla nulový průměr a rozptyl 1. Každá hodnota x ve sloupci tvořeným hodnotami $(x_1, \ldots, x_n)$ je v případě Z-score standardizace nahrazena hodnotou x' vzniklou následujícím způsobem:

$$x' = \frac{x - \bar{X}}{s_n}$$

kde $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} x_i$ je výběrový průměr všech hodnot sloupce a $s_n = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{X})}$ představuje výběrovou směrodatnou odchylku hodnot sloupce.

Nevýhodou Z-skóre standardizace oproti normalizaci je požadavek dat pocházejících z normálního rozdělení.

Undersampling

Anglický termín *Undersampling* označuje metodu předzpracování dat využívanou v případě, kdy v datové sadě výrazně převažují počty datových bodů jedné nebo několika tříd nad počty datových bodů z tříd ostatních. *Undersampling* postupuje tak, že z majoritních tříd odebírá datové body za cílem vyvážit počty datových bodů pro všechny třídy, čímž v některých případech napomáhá modelům při predikci původně minoritních tříd.

Oversampling

Anglický termín *Oversampling* označuje metodu předzpracování dat, která je založena na stejném principu jako *Undersampling*. Jediným rozdílem je, že *Oversampling* při vyvažování počtu datových bodů všech tříd duplikuje datové body z minoritních tříd, čímž vytváří novou datovou sadu s více záznamy pro minoritní třídy, než měla původní datová sada.

1.5 Metody evaluace modelů

Evaluační metriky mají velmi dobře známé a běžně používané anglické názvy, jejichž překlad může být matoucí pro čtenáře. Z tohoto důvodů jsou v textu ponechány jejich anglické názvy.

Accuracy

Nejběžněji používanou metrikou pro evaluaci popisuje anglický termín *Accuracy*, který značí celkovou přesnost definovanou jako podíl správně zařazených datových bodů a všech datových bodů z datové sady. Celková přesnost predikce na datové sadě s N datovými body a K predikovanými skupinami se tedy dá spočítat jako:

$$Accuracy = \frac{1}{N}\sum_{k \in K} TP_k$$

Kde TP_k (angl. True Positive) představuje počet správně zařazených bodů do třídy k.

Precision

Anglický termín *Precision* pro třídu k udává jak často model predikuje správně, pokud zařadí datový bod do třídy k. Vzorec výpočtu *Precision* pro třídu k je:

$$Precission_k = \frac{TP_k}{TP_k + FP_k}$$

Kde FP_k (angl. False Positive) představuje počet chybně zařazených bodů do třídy k.

Recall

Anglický termín *Recall* třídy k udává, jak často je model schopen predikovat správně vzhledem ke všem datovým bodům, které reálně patří do třídy k.

$$Recall_k = \frac{TP_k}{TP_k + FN_k}$$

Kde FN_k (angl. False Negative) udává počet špatně zařazených datových bodů, které v realitě patří do třídy k.

F1 skóre

F1 skóre (angl. F1-score) pro třídu k představuje harmonický průměr hodnot $Precission_k$ a $Recall_k$.

$$F1\text{-}score_k = 2 \cdot \frac{Precision_k \cdot Recall_k}{Recall_k + Precission_k}$$

F1 skóre se využívá především pro datové sady, ve kterých jsou počty datových bodů jednotlivých tříd výrazně odlišné.

Macro averaging

Macro averaging se používá v případě, kdy chceme vybranou metrikou evaluovat predikce pro celou datovou sadu místo pouze 1 klasifikační třídy.

Pro *Macro averaging* nejdříve sečteme danou metriku pro každou z tříd a následně výsledek podělíme počtem tříd. Tím získáme nevážený průměr metriky pro všechny třídy. Například *MacroAverageRecall* tedy získáme jako:

$$MacroAverageRecall = \frac{\sum_{k=1}^{K} Recall_k}{K}$$

Macro averaging je vhodný využít v případě datových sad s nevybalancovanými počty datových bodů pro jednotlivé třídy.

V některých případech evaluace predikce pro celou datovou sadu se také používá *Micro averaging*, který místo každé třídě přikládá stejnou váhu každému datovému bodu. [32]

Kapitola 2

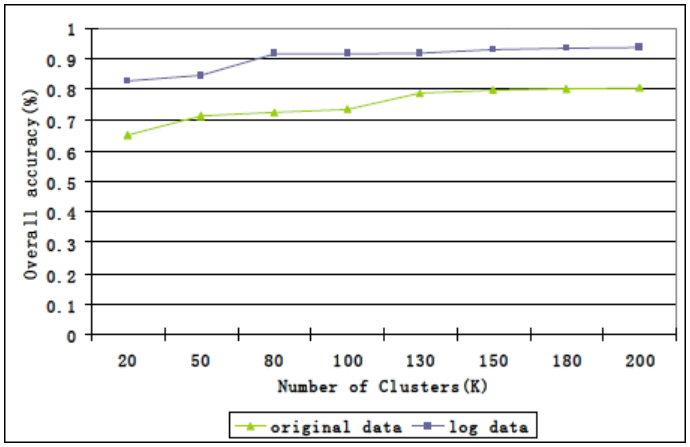
Existující relevantní práce

V této kapitole budou představeny již provedené experimenty související s touto prací. Kapitola je rozdělena do dvou částí. V první části budou představeny experimenty využití shlukování pro klasifikaci síťového provozu. V druhé části poté budou popsány některé práce zabývající se klasifikací zařízení v počítačových sítích pomocí metod strojového učení.

2.1 Využití shlukování pro klasifikaci při analýze síťového provozu

Liu et al. [27] představili experimentální použití shlukovacího algoritmu K-means pro klasifikaci síťového TCP (Transmission Control Protocol) provozu. Data pro shlukování byla získána ze sítě výzkumného zařízení, ve kterém bylo k internetu připojeno přibližně tisíc uživatelů. Síťový provoz byl klasifikován do 12 skupin, mezi které patřilo WWW, MAIL, P2P, atd. Jednotlivé datové body byly ve formě síťových toků, kde každý bod byl reprezentován 10 příznaky, jejichž příkladem může být počet bytů odeslaných v daném síťovém toku, nebo průměrná velikost paketu. Při analýze datové sady bylo zjištěno, že naprostá většina (83,7 %) získaného síťového provozu náležela do třídy WWW. Z tohoto důvodu se autoři rozhodli ke zmenšení počtu datových bodů v datové sadě. Po předzpracování dat a shlukové analýze se ukázalo, že algoritmus K-means pro klasifikaci dosáhl celkové přesnosti přibližně 90,5 %. Na obrázku 2.1 je vidět, že přesnost modelu výrazně vzrostla po využití logaritmické normalizace dat.

Další podobný experiment klasifikace síťového provozu s využitím shlukovacích algoritmů představili Erman et al. [28]. Podobně jako v předešlé představené práci se i tato práce zabývá klasifikací síťového provozu na základě síťových toků do 11 různých skupin. Autoři ke klasifikaci sítového provozu využili dvě datové sady. Jednu volně dostupnou z univerzity v Aucklandu z roku 2001 a druhá byla autory vytvořena na základě monitorování sítě univerzity v Calgary. Ke klasifikaci byly využity a porovnány tři shlukovací algoritmy, K-means, DBSCAN a AutoClass. Vzhledem k tomu, že obě datové sady byly velmi rozsáhlé a v jedné z datových sad výrazně převládal počet jedné z klasifikačních tříd, autoři se s ohledem na schopnost modelu predikovat všechny třídy co nejlépe rozhodli pro trénování vytvořit 2 nové datové sady. Jedna obsahovala 1 000 náhodných datových bodů z každé třídy z Aucklandské datové sady a druhá obsahovala 2 000 náhodných datových bodů z každé třídy z datové sady z univerzity v Calgary. Pro evaluaci bylo vygenerováno 10 dalších datových sad pro každou z původních dvou velkých datových sad. Pro obě skupiny bylo následně uloženo minimum, maximum a průměrná hodnota celkové přesnosti. Celková přesnost algoritmu DBSCAN byla menší než přesnost K-means a AutoClass. DBSCAN ale dokázal produkovat shluky, ve kterých výrazně převládali datové body nakonec predikované třídy a tudíž měl vysokou jistotu predikce. Jedním z důvodů, proč DBSCAN měl



Obrázek 2.1 Závislost přesnosti modelu na počtu shluků a normalizaci dat pro experiment představený v [27]

nižší celkovou přesnost než zbylé použité algoritmy je, že je jako jediný schopný odhalit šum, který byl označen za špatně klasifikovaný při evaluaci. Představený experiment také ukázal, že algoritmus AutoClass je pro tuto úlohu nevhodný, neboť trénování bylo oproti zbylým algoritmům velmi pomalé. I v tomto experimentu mělo zlogaritmování dat pozitivní vliv na výsledek.

2.2 Klasifikace zařízení na základě síťového provozu

Klasifikací zařízení v síti na základě jejich provozu se již zabýval Zdeněk Kasner ve své bakalářské práci [33]. Cílem jeho práce byl návrh metody strojového učení, která bude na základě statistických informací o síťových tocích naměřených v síti schopna klasifikovat jednotlivá zařízení. Datová sada využitá při práci vznikla na základě analýzy síťového provozu 66 zařízení z 10 klasifikačních tříd v síti CESNET2. Výsledná datová sada pro klasifikaci obsahovala 6 příznaků. Jako klasifikační model byla vybrána metoda podpůrných vektorů (angl. Support Vector Machine). Na obrázku 2.2 je vidět, že proces klasifikace povoloval zařazení jednoho zařízení do několika klasifikačních tříd. Nad některými z příznaků byla před trénováním a následnou evaluací modelu provedena normalizace. Evaluace samotného modelu probíhala pomocí deseti-násobné křížové validace, pro kterou byla průměrná přesnost 97,5 %.

```
ip_address       labels
66.189.174.146   CLIENT
76.241.17.159    SERVER
92.155.236.218   SERVER,MAIL
107.41.42.63     SERVER,MAIL
128.135.183.34   SERVER,HTTP
140.231.116.196  SERVER,HTTP,FTP
145.22.147.241   SERVER,HTTP
```

Obrázek 2.2 CSV formát výstupu z modulu vytvořeného v práci [33]

Bai et al. [34] navrhli metodu pro automatickou klasifikaci IoT (Internet of Things) zařízení analýzou jejich síťového provozu do několika skupin s využitím neuronových sítí. Autoři při evaluaci své metody použili veřejně dostupnou datovou sadu s informacemi o zachycených paketech ze síťové komunikace skupiny IoT zařízení během 19 dní. Na základě této sady vznikly 4 třídy (Hubs, Electronics, Cameras, Switches&Triggers), do kterých byly IoT zařízení klasifikovány.

V rámci předzpracování dat byly pakety rozděleny do 2 skupin (user packet, control packet) na základě protokolů využitých při komunikaci. Následně byly napočítány příznaky, na kterých se v dalším kroku model neuronové sítě pro klasifikaci učil. Mezi vytvořené příznaky patří např. počet všech paketů komunikace, počet protokolů využitých při komunikaci zařízení, průměrná velikost paketu obou tříd a další. Předzpracovaná data byla využita pro naučení neuronové sítě obsahující konvoluční, maxpooling a další vrstvy. Po evaluaci byl model porovnán s dalšími 10 často využívanými metodami strojového učení. Na obrázku 2.3 je vidět, že autory navržený model neuronové sítě dopadl s výslednou evaluační přesností 74,8 % nejlépe ze všech testovaných metod. Při experimentech autoři začínali s agregací dat po 1 min. Ukázalo se, že s rostoucí délkou agregačního intervalu rostla i výsledná přesnost modelu.

Index	Methods	Accuracy(%)
1	SVM	58.5
2	RF	30.1
3	KNN(k=10)	27.6
4	Decision Tree	46.4
5	AdaBoost	48.5
6	LDA	49.4
7	QDA	52.4
8	MLP	52.1
9	CNN	56.3
10	LSTM	65.4
11	Ours	**74.8**

Obrázek 2.3 Porovnání modelu představeného v [34] s dalšími modely strojového učení. Copyright © 2018, IEEE

Pashamokhtari et al. [35] provedli experiment klasifikace IoT zařízení do 26 různých skupin, mezi kterými byl například Google Home, Amazon Echo a další, z pohledu ISP. Datovou sadu pro klasifikaci tvořila množina záznamů o síťových tocích nasbíraná pomocí protokolu IPFIX na routerech ISP, přes které procházela komunikace z IoT zařízení během ledna, března a dubna roku 2020. Po filtraci nasbírané datové sady vznikla finální sada pro klasifikaci o velikosti přibližně 3 milionů síťových toků, které byly rozděleny na trénovací množinu, na které probíhala evaluace modelu pomocí křížové validace a testovací množinu. Datová sada obsahovala celkem 18 příznaků, např. průměrný čas mezi 2 poslanými pakety, maximální velikost paketu, celkový počet paketů atd. Už při analýze dat bylo odhaleno několik souvislostí mezi hodnotami příznaků a predikovanými třídami, které naznačovali, že by následná predikce pomocí metod strojového učení mohla být úspěšná, například 95 % síťových toků, u kterých je počet paketů komunikace směrem k predikovanému zařízení větší než 500, spadala do třídy Sony speaker. Pro samotnou klasifikaci byl vybrán model náhodného lesa. Autoři také definovali novou metriku zvanou *Trust*, která byla při výsledné predikci spočítána a pokud byla menší než nastavená prahová hodnota, datový bod nebyl při evaluaci uvažován. Finální model měl po získání vhodných hyperparametrů klasifikační přesnost 96 %. Experiment také ukázal, že přesnost predikce se výrazně liší pro různé komunikační protokoly, model byl například schopen lépe predikovat komunikaci pomocí TCP než komunikaci přes UDP (User Datagram Protocol).

Kapitola 3
Datová sada

Datová sada pro klasifikaci představuje volumetrické informace v podobě časových řad o zařízeních připojených k síti CESNET3 (Czech Education and Scientific Network). Vysokorychlostní počítačová síť CESNET3 spojuje okruhy s vysokými přenosovými rychlostmi do největších univerzitních měst ČR a dalších oblastí. Krom standardního připojení k internetu a velkých přenosových kapacit pro vědecké a výzkumné účely nabízí svým uživatelům také mnoho dalších moderních služeb, např. videokonference, superpočítač MetaCentrum, IP telefonie atd. [36]

V této kapitole bude nejprve představen postup tvorby této datové sady společně s tvorbou anotace jednotlivých zařízení a následně budou ukázány některé vlastnosti datové sady.

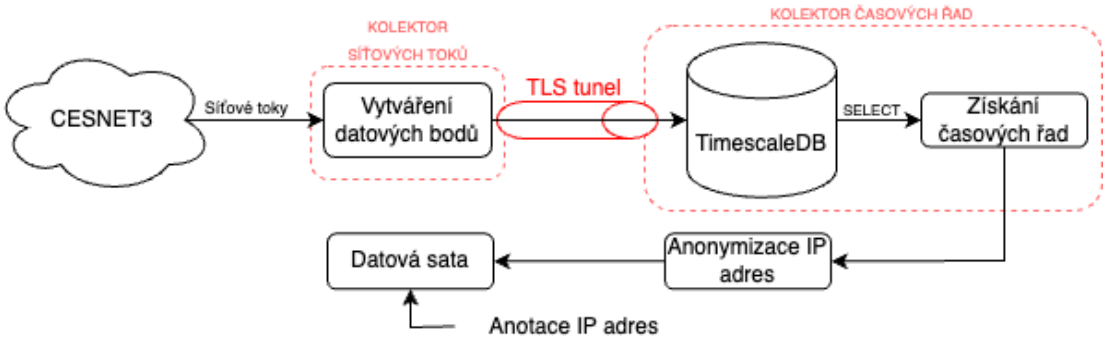
3.1 Metodologie tvorby datové sady

3.1.1 Sběr dat

Datová sada byla vytvořen monitorováním pomocí síťových toků využitím protokolu IPFIX. Samotnou tvorbu popisuje obrázek 3.1, tvorbu datové sady lze shrnout do následujících bodů:

1. Na CESNET3 síti se pomocí monitorovacích sond sbírá síťový provoz ve formě paketů
2. Pakety jsou na sondách pomocí exportéru síťových toků Ipfixprobe[1] agregovány do jednotlivých síťových toků

[1] `https://github.com/CESNET/ipfixprobe`



Obrázek 3.1 Proces tvorby datové sady

3. Síťové toky jsou sondami posílány na kolektor síťových toků IPFIXcol2[2]
4. Na kolektoru síťových toků je nasazen NEMEA[37] modul Agregátor, který síťové toky agreguje na intervaly, tzn. vytváří datové body. Agregátor pracuje s agregačním okénkem 10 minut. Každá časová řada je tvořena následujícími metrikami, které jsou dále v textu označovány jako příznaky:

   id_ip je unikátní identifikátor IP adresy identifikující do jaké časové řady datový bod patří

   time je čas datového bodu identifikující jeho pozici v časové řadě

   n_flows je celkový počet síťových toků vytvořených komunikací zařízení

   n_packets je počet paketů, které zařízení přijalo/odeslalo

   n_bytes je velikost síťové komunikace zařízení v bytech

   n_dest_asn je počet různých síťových autonomních systémů, do kterých byla ze zařízení odeslána nějaká data

   n_dest_ports je počet různých cílových portů využitých při komunikaci zařízení

   n_dest_ip je počet různých IP adres, se kterými zařízení komunikovalo

   tcp_udp_ratio_packets je počet paketů odeslaných zařízením při komunikaci pomocí TCP protokolu děleno počtem všech paketů (tzn. pakety protokolů TCP a UDP)

   tcp_udp_ratio_bytes je počet bytů odeslaných zařízením při komunikaci pomocí TCP protokolu děleno počtem všech bytů (tzn. pakety protokolů TCP a UDP)

   dir_ratio_packets je počet paketů odeslaných zařízením děleno počtem všech paketů komunikace

   dir_ratio_bytes je počet bytů odeslaných zařízením děleno počtem všech bytů komunikace

   avg_duration představuje průměrnou dobu komunikace zařízení

   avg_ttl je průměrné TTL (angl. Time To Live) paketů komunikace
5. Z agregátoru jsou metriky odesílány zabezpečeným tunelem na kolektor časových řad
6. Na kolektoru časových řad jsou datové body vloženy do databáze TimeScaleDB[3], čímž se v databázi postupně sestavují časové řady
7. Po zachycení dostatečného množství dat jsou časové řady získány z databáze
8. Vzniklá datová sada časových řad je poté anotována dle IP adres a následně jsou IP adresy anonymizovány z důvodu ochrany soukromí uživatelů sítě CESNET3

Cílem práce je navrhnout shlukovací model, který by běžel na kolektoru časových řad, čímž by usnadnil anotaci jednotlivých zařízení.

3.1.2 Anotace datové sady

Anotace probíhala poloautomatickou metodou, která probíhala nejprve na základě znalosti infrastruktury sítě CESNET3 a k ní připojených zařízení. Poté se prováděla pomocí reverzního DNS dotazu a dotazů do Shodan[4] anotace neznámých IP adres. Tímto způsobem lze oanotovat pouze zlomek všech komunikujících zařízení na síti CESNET3.

[2] `https://github.com/CESNET/ipfixcol2`
[3] `https://www.timescale.com/`
[4] https://www.shodan.io/

3.2 Popis datové sady

Samotná datová sada obsahuje celkem 178 081 časových řad agregovaných po 10 minutách, které představují informace o síťové komunikaci jednotlivých zařízení v síti mezi daty 2023-10-05 a 2023-11-13. Velikost celé datové sady je 31,8 GB.

Datová sada s anotacemi obsahuje anotace k celkem 107 892 zařízením, z těchto zařízení ale 25 388 není v datové sadě časových řad. Po filtraci datové sady časových řad, během které byly odebrány časové řady nenalezené pomocí datové sady s anotacemi, vznikla výsledná datová sada o velikosti 12,1 GB pro trénování a finální evaluaci obsahující 82 504 časových řad společně s jejich anotacemi.

Zastoupení jednotlivých tříd ve finální datové sadě ukazuje tabulka 3.1.

Tabulka 3.1 Tabulka zastoupení klasifikačních tříd v datové sadě

Typ zařízení	Počet zařízení
end-device	72 523
server	7 875
net-device	2 106

V datové sadě se původně nacházela i třída *nat*, ve které bylo pouze 8 časových řad. Vzhledem k způsobu tvorby anotace časových řad byla ale po diskusi s vedoucím sloučena s třídou *end-device*. Třída *nat* se totiž zvolenou anotační metodou nepříliš často dokáže přiřadit správný štítek.

Celkem nepřekvapivě je majoritní třídou *end-device* (koncové zařízení), i v ostatních pracích zabývající se analýzou síťového provozu často jedna z predikovaných tříd výrazně převládá nad ostatními. Toto pozorování je již v literatuře velice dobře známé a popsané [27, 28].

Jednotlivé časové řady jsou v datové sadě uloženy jako soubory v CSV (Comma-separated values) formátu, kde každý řádek obsahuje data nasbíraná za 10 minut. V případě, že zařízení během některých 10 minut nekomunikovalo, záznam z těchto 10 minut v CSV souboru chybí. Obrázek 3.2 představuje příklad 10 řádků časové řady informací o 7 příznacích.

	time	n_flows	n_packets	n_bytes	n_dest_asn	n_dest_ports	tcp_udp_ratio_packets	tcp_udp_ratio_bytes
0	2023-10-10 12:43:49+02	10	145	77475	9	3	1.00	1.0
1	2023-10-10 12:03:49+02	4	59	47479	4	3	1.00	1.0
2	2023-10-10 11:53:49+02	7	33	13287	7	6	0.97	1.0
3	2023-10-10 11:43:49+02	5	34	8520	5	4	1.00	1.0
4	2023-10-10 11:33:49+02	5	56	47019	5	4	1.00	1.0
5	2023-10-10 11:23:49+02	4	8	800	3	3	0.25	0.1
6	2023-10-10 11:13:49+02	1	14	4639	1	1	1.00	1.0
7	2023-10-10 11:03:49+02	9	151	71051	7	5	1.00	1.0
8	2023-10-10 10:53:49+02	4	30	12578	4	3	1.00	1.0
9	2023-10-10 10:33:49+02	4	32	8502	4	4	1.00	1.0

Obrázek 3.2 Příklad několika řádků časové řady komunikace jednoho ze zařízení

Vzhledem k tomu, že datová sada obsahuje velké množství časových řad s celkem 12 využitelnými příznaky pro učení modelů, je nemožné čistě na základě statické analýzy a pozorování jednotlivých časových řad vytvořit sadu pravidel pro predikci. I přes to lze ale v chování jednotlivých tříd nalézt pro některé z příznaků často opakující se vzory, které by v případě rozpoznání modelem mohly predikci výrazně pomoci.

Obrázek 3.3 Ukázka často opakujících se vzorů chování časových řad jednotlivých tříd pro příznak *n_flows*

Například pro příznak *n_flows* platí, že zařízení ze skupiny *end-device* mají při agregaci časových řad po 2 hodinách často nepravidelné hodnoty v rozmezí 0 až 500. Hodnoty třídy *net-device* jsou většinou také nezávislé na čase, dosahují ale vyšších hodnot než hodnoty pro *end-device*, většinou v rozmezí 200 až 2000. I přes to, že počet síťových toků třídy *net-device* dosahuje často vyšších hodnot, než počet síťových toků pro třídu *end-device*, počet bajtů síťové komunikace mají vyšší většinou časové řady třídy *end-device*. Pro servery platí, že jsou často vytíženy během denní doby pěti pracovních dnů v týdnu, hodnoty příznaky *n_flows* bývají pro servery výrazně vyšší, než pro zbylé třídy, běžně se pohybují v jednotkách desítek tisíců. Obrázek 3.3 uvádí příklady časových řad, které jsou agregovány po 2 hodinách a představují průběh 1 týdne příznaku *n_flows* pro jednotlivé třídy. Popsané vzory ale rozhodně nejsou pravidlem pro jednotlivé třídy, datová sada obsahuje velké množství časových řad a tudíž při statické analýze lze narazit na širokou škálu různých chování zařízení v čase.

Kapitola 4
Metodologie

V kapitole Metodologie bude nejdříve představen postup výběru vhodného shlukovacího algoritmu pro tuto práci. Dále bude vysvětleno použití jednotlivých metod v rámci předzpracování dat. Nakonec bude popsán průběh trénování jednotlivých modelů a výběru finálního modelu. Aby byly jednotlivé experimenty reprodukovatelné, rozhodl jsem se pro rozdělení datových sad a následné trénování modelů využít hodnotu 21 jako *random state*, i přes to se ale výsledky jednotlivých modelů natrénovaných na mém počítači a na MetaCentru na stejných hyperparametrech lehce lišily (maximálně v nízkých jednotkách procent).

4.1 Výběr vhodného modelu

Použitá shlukovací metoda byla vybrána především na základě literatury. V několika pracích zabývajících se shlukovou analýzou síťového provozu byl úspěšně využit algoritmus K-means [27, 28, 38]. Vzhledem k velikosti finální datové sady pro trénování a evaluaci modelů byla důležitým kritériem rychlost modelu. Erman et al. [28] porovnali při shlukování síťového provozu ve formě síťových toků rychlost algoritmů K-means, DBSCAN a AutoClass. Ukázalo se, že algoritmy K-means a DBSCAN jsou schopny v přijatelném čase (4h až 10h) zpracovat i datové sady o velikosti 100 000 a více datových bodů. Protože s rostoucí dimenzionalitou dat většinou pro DBSCAN klesá schopnost produkovat kvalitní shluky [13] a datová sada pro tuto práci obsahuje celkem 12 příznaků pro klasifikaci, jevil se jako vhodný shlukovací algoritmus ke klasifikaci síťových zařízení algoritmus K-means.

Pro výběr vhodného shlukovacího algoritmu mi byla také v listopadu roku 2023 vedoucím práce poskytnuta menší datová sada s přibližně 1 400 časovými řadami volumetrických informací o zařízeních v síti. Na této datové sadě byly porovnány shlukovací algoritmy K-means, K-medoids a K-shape. Jejich dosažené trénovací přesnosti obsahuje tabulka 4.1.

Tabulka 4.1 Tabulka trénovacích přesností jednotlivých modelů na menší datové sadě při výběru vhodného modelu pro tuto práci

shlukovací metoda	dosažená přesnost
K-means	0,78
K-medoids	0,75
K-shape	0,55

Výsledky experimentů na menší datové sadě potvrdily, že by K-means mohl být vhodným shlukovacím řešením pro klasifikaci zařízení nad poskytnutým datasetem. Na základě toho jsem

se v práci rozhodl zaměřit na K-means a vhodný výběr jeho hyperparametrů, především vhodné podmnožiny příznaků.

Další výhodou algoritmu K-means je snadná evaluace nových datových bodů. Při predikci nového datového bodu stačí napočítat jeho vzdálenost od všech center shluků a jako jeho anotaci predikovat anotaci shluku, jehož centrum je k němu nejblíže. To je z časového hlediska mnohem výhodnější než predikce pro některé z dalších shlukovacích algoritmů, u kterých by se v rámci predikce musela spočítat vzdálenost predikovaného datového bodu od všech datových bodů využitých při trénování.

4.2 Předzpracování dat

Již některé informace v kapitole zabývající se představením použité datové sady ukázaly, že před trénováním by bylo vhodné nějakým způsobem předzpracovat za účelem zvýšení schopnosti správné predikce modelu.

Chybějící hodnoty

Prvním problémem datové sady této práce byly chybějící hodnoty. Tím, že data byla sbírána v intervalech o délce 10 minut, bylo běžné, že zařízení v některých z intervalů v síti nekomunikovalo a tím pádem v datech informace o těchto 10 minutách chyběly. Časové řady v datové sadě měly proto velmi často různé délky, neboť se lišily počtem získaných měření. I přes to, že by se byl shlukovací model schopný s časovými řadami různých délek teoreticky schopný vypořádat za předpokladu využití DTW při výpočtu vzdálenosti 2 časových řad, rozhodl jsem se délky jednotlivých časových řad sjednotit. Jeden z důvodů také bylo to, že DTW má výrazně vyšší výpočetní složitost než Euklidovská vzdálenost, která vyžaduje časové řady stejných délek. Sjednocení délek jednotlivých časových řad probíhalo následovně:

1. Agregace časových intervalů po určených intervalech, konkrétně 1h, 2h a 4h. Během agregace byly jednotlivé příznaky dat agregovány následujícími způsoby:

   time byl přepsán počátečním časem nového intervalu

   n_flows, n_packets, n_bytes, n_dest_asn, n_dest_ports, n_dest_ip byly nahrazeny součtem jejich hodnot jednotlivých 10min intervalů

   tcp_udp_ratio_packets, tcp_udp_ratio_bytes dir_ratio_packets, byly nahrazeny průměry jejich hodnot z jednotlivých 10min intervalů

   dir_ratio_bytes, avg_duration, avg_ttl byly také nahrazeny průměry jejich hodnot z jednotlivých 10min intervalů

2. V případě, že i po agregaci byly v časové řadě chybějící hodnoty, byly doplněny měřením s nulovými hodnotami.

Příznaky *n_dest_asn*, *n_dest_ports* a *n_dest_ip* by se také teoreticky daly agregovat průměrem hodnot. Například hodnoty *n_dest_ports* totiž v jednotlivých 10min intervalech pravděpodobně neznamenají komunikaci na vždy různé cílové porty. Sečtení hodnot proto spíše poukazuje na to, jak často zařízení komunikovalo, než na reálný počet různých cílových portů komunikace.

Transformace dat

Dalším problémem byla rozdílná měřítka hodnot jednotlivých příznaků. Například hodnoty příznaku *tcp_udp_ratio_packets* byly v rozmezí 0 až 1, kdežto příznak *n_bytes* podle očekávání běžně dosahoval hodnot v měřítku desítek tisíců. Vzhledem k tomu, že hodnoty 4 příznaků byly už původně v rozmezí 0 až 1, rozhodl jsem se i zbylé příznaky transformovat pomocí min-max

normalizace do hodnot stejného měřítka, kdy za hodnoty max resp. min byly označeny největší resp. nejmenší hodnoty ze všech měření ve všech časových řad trénovací množiny.

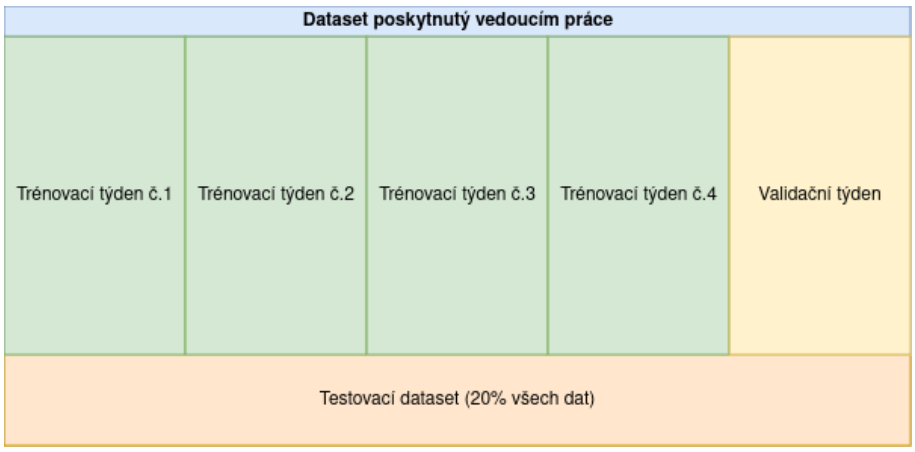
Undersampling

Při počátečních fázích trénování modelů se ukázalo, že nevyváženost hodnot jednotlivých tříd v datové sadě je velký problém. Model totiž označil většinu shluků při trénování za majoritní třídu *end-device* a následně při evaluaci predikoval téměř všechny body z evaluační množiny jako *end-device*. Ve snaze tomu zabránit a donutit model být schopný dobře predikovat i ostatní třídy jsem se rozhodl při učení odebrat z dat využitých k trénování modelů 65 % náhodných bodů třídy *end-device*.

4.3 Trénování a evaluace modelů

Po výběru vhodného modelu byla datová sada časových řad rozdělena na testovací a trénovací sady, kdy testovací sada obsahovala 20 % všech časových řad. Pro rozdělení byla použita funkce train_test_split z knihovny scikit-learn [12] s nastavením zachování stejného procentuálního zastoupení jednotlivých predikovaných tříd v obou nových datových sadách. Po rozdělení obsahovala trénovací datová sada 66 003 časových řad a v testovací sadě se nacházelo 16 501 časových řad.

Vzhledem k tomu, že v praxi by měl model běžet na kolektorech časových řad, kde by byl každý týden využit k predikci typu jednotlivých zařízení v síti na základě jejich týdenní síťové komunikace, byla trénovací datová sada po diskusi s vedoucím práce rozdělena na jednotlivé týdny, kdy každý týden tvořil v rámci jednoho zařízení 1 nový datový bod. Původní datová sada se skládala z celkem 5 týdnů a 4 dnů. Z důvodu zachování stejné struktury jednotlivých datových bodů byla data z posledních 4 dnů odebrána. Datové body získané z prvních 4 týdnů byly následně použity pro natrénování jednotlivých modelů a data z posledního týdne byla použita jako validační data. Po vybrání nejúspěšnějšího modelu na validačních datech byla přesnost finálního modelu odhadnuta na datech, se kterými se vybraný model během trénování ani evaluace nesetkal, které představovala testovací datová sada. Rozdělení původní datové sady je zobrazeno na diagramu 4.1.



Obrázek 4.1 Rozdělení původní datové sady na trénovací, validační a testovací množiny

Trénovací množina tedy před využitím metod předzpracování dat celkem obsahovala $4 \cdot 66003 = 264012$ datových bodů ve formě časových řad. Evaluační množina obsahovala 66 003

časových řad. Vstupem pro trénování a evaluaci jednotlivých modelů byl tedy python list časových řad představující data z jednoho týdne agregovaná na časové intervaly určené v rámci odstranění chybějících hodnot.

Samotné trénování jednotlivých modelů probíhalo v následujících krocích:

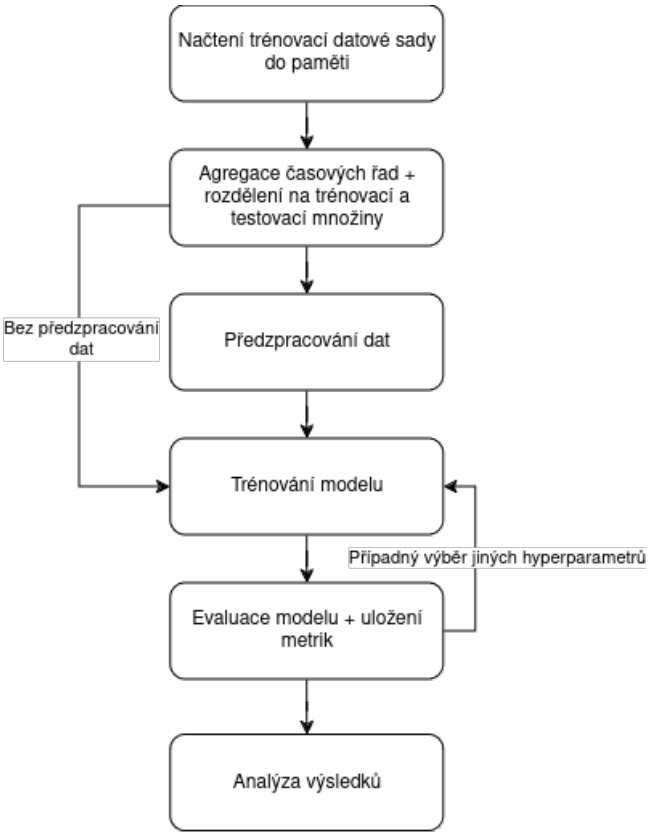
1. Načtení trénovací datové sady a její anotace do paměti
2. Agregace všech časových řad trénovací datové sady po vybraném intervalu, rozdělení na datové body po týdnech a následné označení množiny datových bodů z prvních 4 týdnů za trénovací množinu a datové body z posledního týdne za množinu validační.
3. Nepovinné předzpracování dat, pokud byla využita normalizace i *undersampling* třídy *enddevice*, byl nejdříve proveden *undersampling*. V případě provedení normalizace byly hodnoty min a max pro trénovací množinu uchovány pro následnou transformaci validační množiny pomocí nich.
4. Natrénování samotného modelu se zvolenými hyperparametry na trénovací množině. Uvažované hyperparametry při trénování modelů byly:
   - Počet shluků: byly uvažovány hodnoty 500, 750, 1000, 1250, 1500
   - Způsob výpočtu vzdálenosti 2 časových řad: Euklidovská vzdálenost nebo DTW
   - Agregační okénko pro jednotlivé časové řady: uvažovány byly hodnoty 1h, 2h a 4h.
   - Podmnožina příznaků využitá při trénování

   Pro počáteční inicializaci shluků byla vždy vybrána metoda K-means++. Vzhledem k tomu, že se po prvních pár iteracích účelová funkce algoritmu K-means výrazně neměnila, byl počet iterací pro algoritmus pevně nastaven na 30.
5. Evaluace modelu: Pro trénování jednotlivých modelů a jejich následnou evaluaci byla vytvořena třída *TsKmeansClassifier*, jejíž fungování je vysvětlené v následující kapitole. Při evaluaci byla spočítána přesnost, makro F1 skóre a makro *recall*. Vzhledem k tomu, že byly počty datových bodů v datové sadě pro jednotlivé klasifikační třídy výrazně nevyváženy, bylo cílem při trénování modelů maximalizovat makro F1 skóre. Pro jednotlivé modely byly také ukládány matice záměn predikcí na evaluační množině.

Celý proces trénování a evaluace je graficky znázorněn na obrázku 4.2.

Po vybrání nejlepšího modelu na základě metrik z validační množiny byla jeho finální přesnost odhadnuta na testovací datové sadě, která byla také rozdělena na časové řady z jednotlivých týdnů. Celkem tedy obsahovala $16501 \cdot 5 = 82505$ časových řad.

Kvůli velikosti datové sady a časové náročnosti trénování modelů (nejčastěji v jednotkách hodin) byly také k trénování a evaluaci modelů využity výpočetní služby organizace MetaCentrum. Díky MetaCentru bylo možné trénovat více modelů paralelně, což mělo silný pozitivní vliv na výsledek experimentu.



Obrázek 4.2 Diagram procesu trénování a evaluace modelů

Kapitola 5
Implementace

Vzhledem k dostupnosti velkého množství knihoven pro strojové učení v jazyce Python je implementace práce tvořena několika Python jupyter notebooky.

Jednotlivé časové řady byly v datové sadě uloženy jako soubory ve formátu CSV. Trénovací, validační a testovací datové sady proto po nahrání do paměti měly podobu python listu instancí třídy pandas.DataFrame [39].

Pro trénování modelů pomocí Euklidovské vzdálenosti byla využita třída *TimeSeriesKMeans* z knihovny tslearn [23]. V případě využití DTW jako vzdálenosti 2 bodů byla využita třída *TimeSeriesKMeans* z knihovny sktime [40, 41]. Obě využité třídy mají stejné uživatelské rozhraní. Pro finální analýzu nasbíraných výsledků byly použity knihovny Matplotlib [42] a seaborn [43].

Jak již bylo zmíněno, shlukovací metody jsou především metodami nesupervizovaného učení. V rámci práce proto bylo nutné upravit shlukovací model tak, aby byl schopný predikovat, k tomu slouží třída *TsKmeansClassifier*. Tato třída v konstruktoru přijímá trénovací množinu časových řad, list reálných anotací k jednotlivým časovým řadám a inicializovaný model se stejným uživatelským rozhraním jako *tslearn.clustering.TimeSeriesKMeans* [23]. Po zavolání konstruktoru dojde k běhu algoritmu K-means na trénovací množině dat, následně je pro každý vytvořený shluk zjištěna majoritní klasifikační třída v něm a danému shluku je na základě této třídy přidělena anotace. Shluky, které neobsahují žádné datové body jsou z množiny shluků po trénování odebrány.

Pro získání predikovaných anotací pro nová data slouží metoda *predict_annotations*, která přijímá list časových řad. Na základě vzdáleností center jednotlivých shluků jsou časovým řadám při predikci přiděleny stejné anotace jako je anotace shluku, jehož centrum je k predikované časové řadě nejblíže. Pro samotnou evaluaci shlukování slouží její metoda *evaluate_model*, která přijímá validační (případně testovací) množinu časových řad a příslušnou anotaci. Nejprve jsou pro validační množinu časových řad získány predikce pomocí metody *predict_annotations* a následně jsou na základě predikcí a reálných anotací vypočteny a vráceny evaluační metriky popsané v předešlé kapitole. Příklad použití popsané třídy k evaluaci modelu zobrazuje kód 5.1.

Třída také obsahuje serializační metody *to_pickle* a *from_pickle*, které přijímají název souboru, do kterého (resp. ze kterého) má být instance třídy uložena (resp. načtena) ve formátu Python pickle. Příklad použití metody *from_pickle* k načtení instance třídy ze souboru *Classifier.pkl* a následné získání predikcí pomocí načtené instance zobrazuje kód 5.2.

Dále byly vytvořena funkce pro *undersampling* vybrané klasifikační třídy a třída pro min-max normalizaci dat ve formě python listu časových řad ve formě pandas.DataFrame.

Po otestování funkčnosti kódu trénováním modelů na mém notebooku byl vzhledem k časovým a paměťovým nárokům, kdy data po nahrání a předzpracování zabírala přibližně 18 GB paměti, v rámci snahy o natrénování co největšího počtu modelů pro různé hyperparametry vytvořen python skript *run_clustering.py*, který byl s různými parametry pouštěn na výpočetních

Výpis kódu 5.1 Příklad použití třídy TsKmeansClassifier ke klasifikační evaluaci algoritmu K-means

```
model = TimeSeriesKMeans(n_clusters = 1500, metric = "euclidean",
max_iter = 30, n_jobs = -1, random_state = 21)

clf = TsKmeansClassifier(model, Train.data, Train.annotations)

model_accuracy, confusion_matrix, f1_macro, macro_recall =
clf.evaluate_model(Val.data, Val.annotations)
```

Výpis kódu 5.2 Příklad nahrání třídy TsKmeansClassifier ze souboru Classifier.pkl a následné získání predikcí pro vstupní data

```
import pickle

clf = TsKmeansClassifier.from_pickle("Classifier.pkl")

Val_pred = clf.predict_annotations(Val.data)
```

serverech MetaCentra a následně ukládal výsledky do složky, která byla poté využita k jejich analýze.

V rámci práce s výpočetními službami MetaCentra byly vedoucím práce vytvořeny dva shellové skripty. Skript *run_clustering.sh*, který definoval prostředí, počet využívaných jader a velikost paměti pro výpočty v rámci trénování modelů a následně pustil výše popsaný python skript. Skript *run_all_clustering.sh* obsahoval možné hodnoty trénovacích hyperparametrů společně s cyklem, který prošel možnosti hyperparametrů a pro každou z nich zařadil pomocí příkazu *qsub* nový proces běhu *run_clustering.sh* s vybranými hyperparametry do fronty procesů běžících na výpočetních serverech MetaCentra.

Kapitola 6
Výsledky a ponaučení

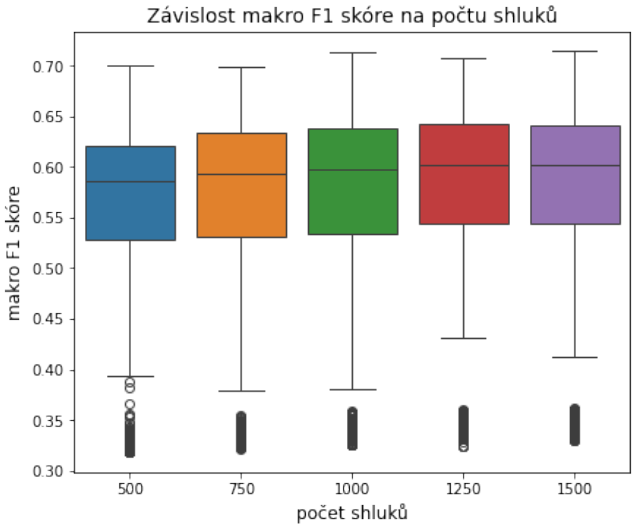
Díky možnosti využití služeb MetaCentra bylo v rámci práce natrénováno přes 30 000 modelů algoritmu K-means.

Min-max normalizace dat měla vzhledem k měřítkům jednotlivých příznaků celkem nepřekvapivě výrazný pozitivní vliv na makro F1 skóre modelů. Stejný efekt měl i *undersampling*. Po získání výsledků z prvních přibližně sedmi tisíc naučených modelů jsem se proto rozhodl dále trénovat už pouze modely na datech, na které byla použita normalizace i *undersampling*. Tabulka 6.1 zobrazuje závislost průměru makro F1 skóre všech natrénovaných modelů na použití těchto metod v rámci předzpracování dat.

Tabulka 6.1 Tabulka závislosti průměru hodnot makro F1 skóre všech natrénovaných modelů na využití představených metod předzpracování dat

		normalizace	
		ano	ne
undersampling	ano	0.61	0.46
	ne	0.52	0.44

Jako vzdálenost časových řad byla uvažována Euklidovská vzdálenost a DTW. V případě Euklidovské vzdálenosti bylo v přijatelném čase možné bez problémů natrénovat modely s jakýmkoliv počtem časových řad. Naopak pro DTW se už na datové sadě o velikosti přibližně 1 400 časových řad ukázalo, že by mohl být v případě výrazně větší datové sady časově nepřijatelný. I přes to jsem chtěl DTW použít, neboť modely natrénované pomocí DTW dosahovaly na menší datové sadě o několik jednotek procent lepší trénovací přesnosti, než v případě Euklidovské vzdálenosti. Bohužel se ale algoritmus DTW ukázal být časově nepřijatelným, ani s využitím služeb MetaCentra se totiž nepovedlo natrénovat modely s využitím DTW za týden, po kterém bylo trénování automaticky vypnuto. Později se ukázalo, že hlavním problémem z hlediska výpočetní složitosti DTW pro K-means byl způsob aktualizace center jednotlivých shluků na konci iterace. Pro aktualizaci center jednotlivých shluků se totiž v případě DTW běžně využívá metoda DBA (DTW barycenter averaging), která zabrala většinu výpočetního času. Po změně způsobu aktualizace center shluků na průměr hodnot (jako v případě Euklidovské vzdálenosti) se nakonec podařilo s pomocí DTW natrénovat několik modelů. Toto omezení se ale výrazně negativně projevilo na jejich úspěšnosti, neboť aktualizace center shluků jako průměr hodnot jednotlivých časových řad je vzhledem k vlastnostem DTW nevhodným řešením. Snahu o úspěšné zahrnutí DTW do trénování jednotlivých modelů pro takto velký dataset hodnotím jako chybu, neboť jsem snahou o optimalizaci DTW strávil mnoho času, který mohl být využit výrazně efektiv-



Obrázek 6.1 Boxplot závislosti makro F1 skóre na počtu shluků natrénovaných modelů

něji, například místo modelů trénovaných pomocí DTW mohlo být natrénováno několik modelů pomocí jiného shlukovacího algoritmu než K-means. Dalším vhodným algoritmem se totiž jevil K-medoids, jehož implementace v dostupných knihovnách ale vyžaduje na začátku vypočítat párovou matici vzdáleností jednotlivých bodů datasetu. Kvůli celkovým paměťovým nárokům tohoto modelu jsem se na začátku experimentu rozhodl strávit čas snahou o časovou optimalizaci DTW.

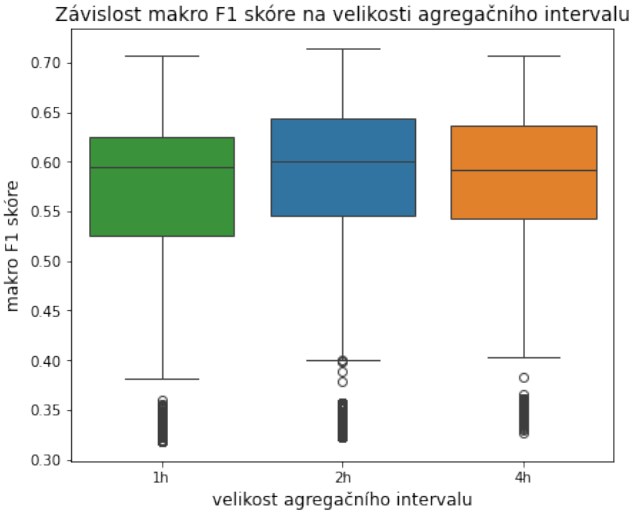
Vzhledem k počtu časových řad bylo nutné pro úspěšnost modelu zvolit vhodný počet shluků. Během trénování byly uvažovány hodnoty 500, 750, 1000, 1250 a 1500. Graf 6.1 zobrazuje závislost makro F1 skóre na počtu shluků pro množinu modelů natrénovaných na těchto hodnotách. Vzhledem k tomu, že se po 1000 shlucích makro F1 skóre výrazně neměnilo, rozhodl jsem se pro větší počet shluků než 1500 modely netrénovat, neboť by krom zvýšené časové náročnosti také hrozilo, že dojde k přetrénování.

V rámci předzpracování dat byly časové řady z několika důvodů agregovány po větších časových intervalech. Při trénování byla vyzkoušena agregace po jedné, dvou a čtyřech hodinách. Z grafu 6.2 lze vyčíst, že obecně byla vzhledem k makro F1 skóre nejúspěšnější agregace po 2 hodinách.

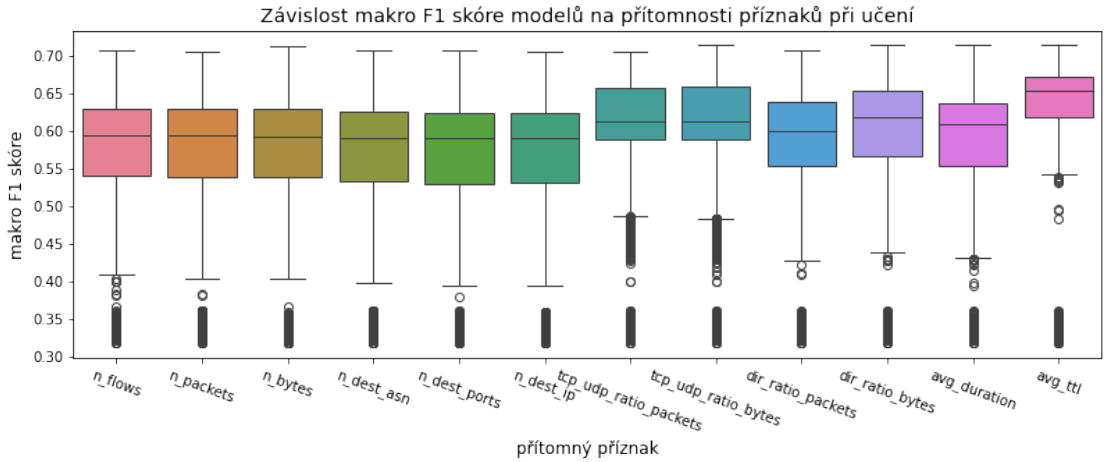
Nejdůležitějším faktorem při učení modelů byl výběr vhodných příznaků. Znalost signifikantních příznaků při trénování nám totiž dává informaci o tom, ve kterých aspektech síťové komunikace se jednotlivé typy zařízení liší. Před analýzou výsledků bylo očekáváno, že mezi nejúspěšnější příznaky při predikci budou patřit *n_bytes*, *n_flows* a *n_dest_ip*. Samotné zkoumání výsledků ale ukázalo, že ani jeden z těchto příznaků není klíčový pro maximalizaci makro F1 skóre modelu.

Graf 6.3 zobrazuje závislost makro F1 skóre na jednotlivých příznacích tak, že pokud byl příznak obsažen v množině příznaků, na kterých byl model natrénován, je výsledek modelu na validační množině zahrnut v tomto grafu v souvislosti s daným příznakem. Je vidět, že nejúspěšnější byly modely, které v množině příznaků využitých pro trénování obsahovaly příznak *avg_ttl* (průměrné TTL (Time To Live) paketu komunikace zařízení).

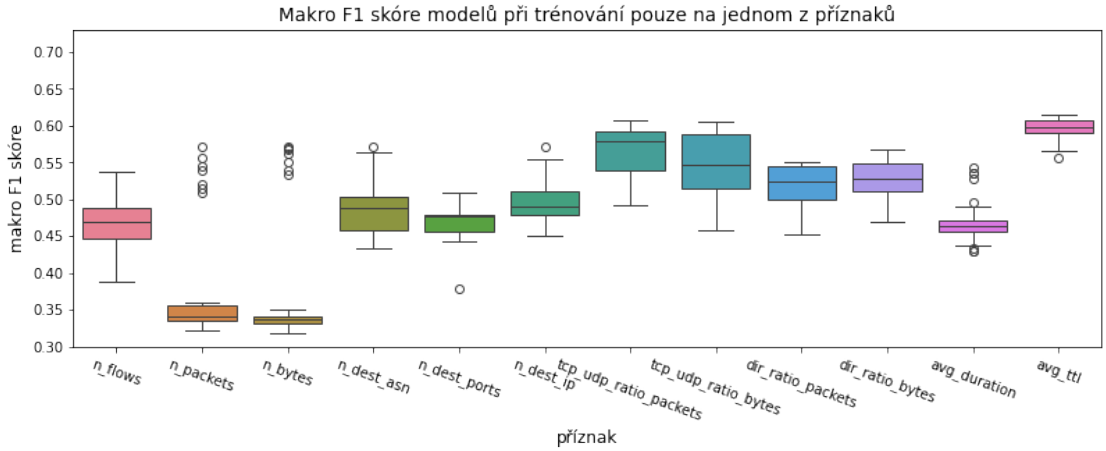
Úspěšnost příznaku *avg_ttl* ukazuje i graf 6.4, který zobrazuje makro F1 skóre modelů v situaci, kdy byl model natrénován pouze na jednom z vybraných příznaků. Výrazná neúspěšnost příznaků *n_packets* a *n_bytes* by se dala vysvětlit tím, že v datové sadě se nachází relativně hodně časových řad, které pro tyto příznaky nabývají výrazně vyšších hodnot než zbylá většina časových řad. Tato odlehlá měření byla algoritmem K-means pravděpodobně označena za samo-



Obrázek 6.2 Boxplot závislosti makro F1 skóre na agregačním intervalu časových řad



Obrázek 6.3 Boxplot závislosti makro F1 skóre na příznaku, který byl obsažen v množině využité pro trénování



Obrázek 6.4 Boxplot makro F1 skóre modelů natrénovaných pouze na jednom vybraném příznaku

statné shluky a proto byla většina datových bodů obsažena pouze v malém množství shluků, ve kterých proto převládala majoritní třída, tzn. štítek *end-device*. Při predikci byl tedy téměř každý bod označen za *end-device*. *Recall* jednotlivých tříd byl tedy velmi nízký a proto bylo i makro F1 skóre nízké. Tento problém by byl teoreticky řešitelný volbou jiného způsobu normalizace dat. Požadovaný způsob normalizace dat by musel znevýhodňovat vysoké hodnoty v tom smyslu, že by se po transformaci v poměru s průměrnou hodnotou nejevily jako tolik odlišné, vhodnou volbou by například byla logaritmická transformace dat a následné využití min-max normalizace. Dalším problémem min-max normalizace vzhledem k účelu této práce je, že v případě nasazení modelu do praxe může dojít k situaci, kdy nějaký z příznaků bude mít mezi časovými řadami získanými v posledním týdnu vyšší hodnotu, než je maximální hodnota v natrénovaných datech. Při predikci bude tento příznak mít po transformaci hodnotu vyšší než 1, což je nežádané.

Celkově pro natrénované modely platilo, že trojice příznaků *tcp_udp_ratio_bytes*, *avg_ttl* a *dir_ratio_bytes* dosahovala nejlepších výsledků vzhledem k makro F1 skóre. Naopak příznaky *n_packets*, *n_dest_ports* a *n_dest_asn* dosahovaly největší celkové přesnosti, což v tomto experimentu nebylo úplně žádané, neboť to znamená, že model dokáže dobře predikovat majoritní třídu na úkor ostatních tříd. To by v praxi znamenalo označení téměř každého zařízení za koncové (třída *end-device*), což by nepřineslo prakticky žádnou informaci o typu jednotlivých zařízení.

Jako celkově nejúspěšnější model se ukázal být model natrénovaný s hyperparametry popsanými v tabulce 6.2. Hodnoty jednotlivých evaluačních metrik nejlepšího modelu zobrazuje

Tabulka 6.2 Tabulka využitých hyperparametrů při učení nejúspěšnějšího modelu z hlediska makro F1 skóre

použité příznaky	tcp_udp_ratio_bytes, dir_ratio_bytes, avg_duration, avg_ttl
výpočet vzdálenosti časových řad	Euklidovská vzdálenost
normalizace dat	ano
undersampling	ano
počet shluků	1500
velikost agregačního intervalu	2h

tabulka 6.3

Po vybrání nejlepšího modelu na základě validačního týdne byla odhadnuta jeho celkovou přesnost, makro F1 skóre a makro *recall* na testovací množině, která představovala data, se kterými se model během trénování a evaluace vůbec nesetkal. Testovací množina byla opět roz-

Tabulka 6.3 Tabulka hodnot evaluačních metrik modelu nejúspěšnějšího z hlediska makro F1 skóre

evaluační metrika	hodnota
makro F1 skóre	0.71
celková přesnost	0.90
makro recall	0.73

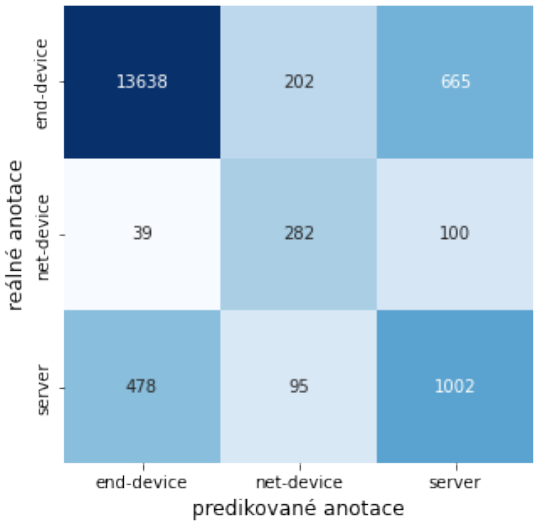
dělena do pěti týdnů. Tabulka 6.4 ukazuje úspěšnost modelu vzhledem k evaluačním metrikám na jednotlivých týdnech.

Tabulka 6.4 Tabulka hodnot evaluačních metrik nejúspěšnějšího modelu pro predikci na testovacích týdnech

evaluační metrika	1. týden	2. týden	3. týden	4. týden	5. týden	průměr
makro F1 skóre	0.71	0.70	0.70	0.70	0.68	0.70
celková přesnost	0.90	0.90	0.90	0.90	0.89	0.90
makro recall	0.75	0.74	0.75	0.74	0.72	0.74

Ani jedna z evaluačnch metrik se v žádném z pěti testovacích týdnů výrazně nelišila od hodnot jednotlivých metrik pro evaluační týden. Podařilo se tedy získat model, který je v čase konstantní. To je v případě klasifikace síťových zařízení na základě jejich týdenní komunikace výrazně pozitivní jev. Mnohdy se totiž stává, že modely s velmi vysokou úspěšností na trénovací a validační množině nejsou následně z důvodu přeučení a zaměření se na často měnící se vlastnosti dat schopny efektivně generalizovat problém a pro nová data je jejich úspěšnost predikce výrazně nižší, než bylo původně očekáváno. K tomuto jevu v případě zvoleného modelu nedošlo, což znamená, že byl nalezen model, který dokáže s konstantně vysokou přesností odhalit typ zařízení. Za předpokladu, že jednotlivé metriky úspěšnosti modelu jsou pro úlohu klasifikace síťových zařízení postačující, mohl by model být úspěšně využit v praxi.

Díky téměř konstantním hodnotám evaluačních metrik v testovacích týdnech se matice záměň predikcí modelu pro jednotlivé týdny výrazně nelišila. Matice záměň predikcí modelu na prvním testovacím týdnu je zobrazena na obrázku 6.5.



Obrázek 6.5 Matice záměn modelu na prvním testovacím týdnu

Vzhledem k počtu časových řad s reálnou anotací *end-device* není překvapivé, že tuto třídu dokáže model predikovat nejpřesněji. Z hlediska použitelnosti v praxi je však mnohem důležitější úloha predikce zbylých dvou tříd. Model je obecně schopen relativně dobře odlišit třídy *net-device* a server. Hlavním problémem získaného modelu je to, že vzhledem k počtu zařízení v třídách server a *net-device* je do nich zařazeno i relativně velké množství časových řad s reálnou anotací *end-device*. Toto se z pravidla opakovalo u všech natrénovaných modelů při evaluaci na validační množině. Model také relativně často predikuje server jako *end-device*. Jako pozitivní výsledek vzhledem k počtu zařízení ve třídě *end-device* považuji, že se tento jev neopakuje i pro třídu *net-device*.

Kapitola 7
Závěr

Práce se zabývala klasifikací typu zařízení ze síťového provozu na základě volumetrických informací o jeho síťové komunikaci ve formě časových řad pomocí shlukovacích metod, které nejčastěji nacházejí uplatnění jako modely nesupervizovaného strojového učení. Hlavním cílem této práce bylo vybrání vhodné shlukovací metody a následné otestování její použitelnosti pro klasifikaci.

Na základě literatury, experimentů a povahy poskytnuté datové sady, která obsahovala 3 klasifikační třídy, byla vybrána shlukovací metoda K-means, pro kterou byla provedena podrobná analýza úspěšnosti predikce na poskytnuté datové sadě.

Díky výpočetním službám MetaCentra bylo v rámci práce natrénováno více než 30 000 modelů s různými hyperparametry, z nichž byl vybrán finální shlukovací model, jehož průměrná přesnost klasifikace typu zařízení na základě jejich síťové komunikace v 5 testovacích týdnech byla 90 %. Průměrné testovací makro F1 skóre bylo 0,70. Tyto výsledky dokazují, že shlukovací algoritmus K-means je použitelný pro úlohu klasifikace typu síťových zařízení. Ukázalo se, že nejdůležitějším příznakem vzhledem k úspěšnosti predikce tohoto modelu bylo průměrné TTL paketů komunikace zařízení. Nakonec byl vytvořen softwarový prototyp ve formě jupyter notebooků, který pro predikci typu zařízení používá získaný model.

Hlavním pozitivem získaného modelu se ukázala být jeho schopnost dosahovat konzistentních výsledků v delším časovém intervalu. To je pro úspěšný model klasifikace typu síťového zařízení na základě jeho komunikace v praxi nezbytným kritériem. Na základě své konzistence a úspěšnosti se model jeví jako vhodné řešení pro detekci typu síťového zařízení v rámci modulu pro detekci anomálií v síti CESNET3.

Budoucí práce bude nejprve cílena na evaluaci vytvořeného modelu na datech získaných z nasbírané síťové komunikace jednotlivých zařízeních z ročního časového intervalu. Poté bude získaný model porovnán s klasickými přístupy klasifikace časových řad, mezi které patří např. neuronové sítě nebo KNN (K-Nearest Neighbors).

Bibliografie

1. *Digital Around the World — DataReportal – Global Digital Insights — datareportal.com* [online]. 2024. [cit. 2024-03-05]. Dostupné z: https://datareportal.com/global-digital-overview.
2. SINGH, Hardeep. Performance Analysis of Unsupervised Machine Learning Techniques for Network Traffic Classification. In: *2015 Fifth International Conference on Advanced Computing Communication Technologies*. 2015, s. 401–404. Dostupné z DOI: 10.1109/ACCT.2015.54.
3. ALQUDAH, Nour; YASEEN, Qussai. Machine Learning for Traffic Analysis: A Review. *Procedia Computer Science*. 2020, roč. 170, s. 911–916. Dostupné z DOI: https://doi.org/10.1016/j.procs.2020.03.111.
4. CECIL, Alisha. A summary of network traffic monitoring and analysis techniques. *Computer systems analysis*. 2006, s. 4–7.
5. SVOBODA, Jakub; GHAFIR, Ibrahim; PRENOSIL, Vaclav. Network Monitoring Approaches: An Overview. *International Journal of Advances in Computer Networks and Its Security– IJCNS*. 2015, roč. 5, s. 88–93. Dostupné také z: https://www.researchgate.net/publication/305957483_Network_Monitoring_Approaches_An_Overview.
6. CLAISE, Benoît; AITKEN, Paul; TRAMMELL, Brian. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information*. RFC Editor, 2013. Dostupné z DOI: 10.17487/RFC7011.
7. HOFSTEDE, Rick; ČELEDA, Pavel; TRAMMELL, Brian; DRAGO, Idilio; SADRE, Ramin; SPEROTTO, Anna; PRAS, Aiko. Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE Communications Surveys Tutorials*. 2014, roč. 16, č. 4, s. 2037–2064. Dostupné z DOI: 10.1109/COMST.2014.2321898.
8. XU, Rui; WUNSCH, D. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*. 2005, roč. 16, č. 3, s. 645–678. Dostupné z DOI: 10.1109/TNN.2005.845141.
9. KUMAR, Vijay; CHHABRA, Jitender Kumar; KUMAR, Dinesh. Performance evaluation of distance metrics in the clustering algorithms. *INFOCOMP Journal of Computer Science*. 2014, roč. 13, č. 1, s. 38–52. Dostupné také z: https://infocomp.dcc.ufla.br/index.php/infocomp/article/view/21.
10. GUNOPULOS, Dimitrios. Clustering Overview and Applications. In: *Encyclopedia of Database Systems*. Boston, MA: Springer US, 2009, s. 383–387. Dostupné z DOI: 10.1007/978-0-387-39940-9_602.
11. PITAFI, Shahneela; ANWAR, Toni; SHARIF, Zubair. A Taxonomy of Machine Learning Clustering Algorithms, Challenges, and Future Realms. *Applied Sciences*. 2023, roč. 13, č. 6. Dostupné z DOI: 10.3390/app13063529.

12. PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011, roč. 12, s. 2825–2830. Dostupné také z: https://scikit-learn.org/stable/.

13. BINDRA, Kamalpreet; MISHRA, Anuranjan. A detailed study of clustering algorithms. In: *2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*. 2017, s. 371–376. Dostupné z DOI: 10.1109/ICRITO.2017.8342454.

14. NIELSEN, Frank. Hierarchical Clustering. In: *Introduction to HPC with MPI for Data Science*. Cham: Springer International Publishing, 2016, s. 195–211. ISBN 978-3-319-21903-5. Dostupné z DOI: 10.1007/978-3-319-21903-5_8.

15. CAMPELLO, Ricardo J. G. B.; KRÖGER, Peer; SANDER, Jörg; ZIMEK, Arthur. Density-based clustering. *WIREs Data Mining and Knowledge Discovery*. 2020, roč. 10, č. 2, e1343. Dostupné z DOI: https://doi.org/10.1002/widm.1343.

16. SCHUBERT, Erich; SANDER, Jörg; ESTER, Martin; KRIEGEL, Hans Peter; XU, Xiaowei. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Trans. Database Syst.* 2017, roč. 42, č. 3, s. 1–21. Dostupné z DOI: 10.1145/3068335.

17. BHATTACHARJEE, Panthadeep; MITRA, Pinaki. A survey of density based clustering algorithms. *Frontiers of Computer Science*. 2021, roč. 15, s. 1–27. Dostupné z DOI: 10.1007/s11704-019-9059-3.

18. MASTRANGELO, Christina M.; SIMPSON, James R.; MONTGOMERY, Douglas C. Time Series Analysis. In: *Encyclopedia of Operations Research and Management Science*. Boston, MA: Springer US, 2013, s. 1546–1552. ISBN 978-1-4419-1153-7. Dostupné z DOI: 10.1007/978-1-4419-1153-7_1045.

19. WARREN LIAO, T. Clustering of time series data—a survey. *Pattern Recognition*. 2005, roč. 38, č. 11, s. 1857–1874. Dostupné z DOI: https://doi.org/10.1016/j.patcog.2005.01.025.

20. ZOLHAVARIEH, Seyedjamal; AGHABOZORGI, Saeed; TEH, Ying Wah et al. A review of subsequence time series clustering. *The Scientific World Journal*. 2014, roč. 2014, s. 1–19. Dostupné také z: https://doi.org/10.1155/2014/312521.

21. AGHABOZORGI, Saeed; SEYED SHIRKHORSHIDI, Ali; YING WAH, Teh. Time-series clustering – A decade review. *Information Systems*. 2015, roč. 53, s. 16–38. Dostupné z DOI: https://doi.org/10.1016/j.is.2015.04.007.

22. MÜLLER, Meinard. Dynamic Time Warping. In: *Information Retrieval for Music and Motion*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, s. 69–84. ISBN 978-3-540-74048-3. Dostupné z DOI: 10.1007/978-3-540-74048-3_4.

23. TAVENARD, Romain; FAOUZI, Johann; VANDEWIELE, Gilles; DIVO, Felix; ANDROZ, Guillaume; HOLTZ, Chester; PAYNE, Marie; YURCHAK, Roman; RUSSWURM, Marc; KOLAR, Kushal; WOODS, Eli. Tslearn, A Machine Learning Toolkit for Time Series Data. *Journal of Machine Learning Research*. 2020, roč. 21, č. 118, s. 1–6. Dostupné také z: http://jmlr.org/papers/v21/20-091.html.

24. KATE, Rohit J. Using dynamic time warping distances as features for improved time series classification. *Data mining and knowledge discovery*. 2016, roč. 30, s. 283–312. Dostupné z DOI: 10.1007/s10618-015-0418-x.

25. KEOGH, Eamonn J.; PAZZANI, Michael J. Scaling up dynamic time warping for data-mining applications. In: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Boston, Massachusetts, USA: Association for Computing Machinery, 2000, s. 285–289. Dostupné z DOI: 10.1145/347090.347153.

26. GONG, Xueyuan; FONG, Simon; CHAN, Jonathan H; MOHAMMED, Sabah. NSPRING: the SPRING extension for subsequence matching of time series supporting normalization. *The Journal of Supercomputing*. 2016, roč. 72, s. 3801–3825. Dostupné z DOI: 10.1007/s11227-015-1525-6.

27. LIU, Yingqiu; LI, Wei; LI, Yunchun. Network Traffic Classification Using K-means Clustering. In: *Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 2007)*. IEEE, 2007, s. 360–365. Dostupné z DOI: 10.1109/IMSCCS.2007.52.

28. ERMAN, Jeffrey; ARLITT, Martin; MAHANTI, Anirban. Traffic classification using clustering algorithms. In: Association for Computing Machinery, 2006, s. 281–286. Dostupné z DOI: 10.1145/1162678.1162679.

29. KHAN, Shehroz S.; AHMAD, Amir. Cluster center initialization algorithm for K-means clustering. *Pattern Recognition Letters*. 2004, roč. 25, č. 11, s. 1293–1302. Dostupné z DOI: https://doi.org/10.1016/j.patrec.2004.04.007.

30. CELEBI, M. Emre; KINGRAVI, Hassan A.; VELA, Patricio A. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*. 2013, roč. 40, č. 1, s. 200–210. Dostupné z DOI: https://doi.org/10.1016/j.eswa.2012.07.021.

31. ALI, Peshawa Jamal Muhammad; FARAJ, Rezhna Hassan; KOYA, Erbil; ALI, Peshawa J Muhammad; FARAJ, Rezhna H. Data normalization and standardization: a technical report. *Mach Learn Tech Rep*. 2014, s. 1–6. Dostupné z DOI: 10.13140/RG.2.2.28948.04489.

32. GRANDINI, Margherita; BAGLI, Enrico; VISANI, Giorgio. Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*. 2020. Dostupné z DOI: 10.48550/arXiv.2008.05756.

33. KASNER, Zdeněk. *Flow-Based Classification of Devices in Computer Networks*. 2016. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology.

34. BAI, Lei; YAO, Lina; KANHERE, Salil S.; WANG, Xianzhi; YANG, Zheng. Automatic Device Classification from Network Traffic Streams of Internet of Things. In: *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. IEEE, 2018, s. 1–9. Dostupné z DOI: 10.1109/LCN.2018.8638232.

35. PASHAMOKHTARI, Arman; OKUI, Norihiro; MIYAKE, Yutaka; NAKAHARA, Masataka; GHARAKHEILI, Hassan Habibi. Inferring Connected IoT Devices from IPFIX Records in Residential ISP Networks. In: *2021 IEEE 46th Conference on Local Computer Networks (LCN)*. IEEE, 2021, s. 57–64. Dostupné z DOI: 10.1109/LCN52139.2021.9524954.

36. CESNET, z.s.p.o. *CESNET 3 network* [online]. April 2024. Dostupné také z: https://www.cesnet.cz/en/sit-cesnet3-eng.

37. CEJKA, Tomas; BARTOS, Vaclav; SVEPES, Marek; ROSA, Zdenek; KUBATOVA, Hana. NEMEA: a framework for network traffic analysis. In: *2016 12th International Conference on Network and Service Management (CNSM)*. IEEE, 2016, s. 195–201. Dostupné z DOI: 10.1109/CNSM.2016.7818417.

38. LIU, Duo; LUNG, Chung-Horng; LAMBADARIS, Ioannis; SEDDIGH, Nabil. Network traffic anomaly detection using clustering techniques and performance comparison. In: *2013 26th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 2013, s. 1–4. Dostupné z DOI: 10.1109/CCECE.2013.6567739.

39. MCKINNEY, Wes et al. Data structures for statistical computing in python. In: *Proceedings of the 9th Python in Science Conference*. Austin, TX, 2010, sv. 445, s. 51–56.

40. LÖNING, Markus; BAGNALL, Anthony; GANESH, Sajaysurya; KAZAKOV, Viktor; LINES, Jason; KIRÁLY, Franz J. sktime: A unified interface for machine learning with time series. *arXiv preprint arXiv:1909.07872*. 2019. Dostupné z DOI: 10.48550/arXiv.1909.07872.

41. LÖNING, M; KIRÁLY, F; BAGNALL, T; MIDDLEHURST, M; GANESH, S; OASTLER, G; LINES, J; WALTER, M; VIKTORKAZ, F; MENTEL, L et al. sktime/sktime: v0. 13.4. *Zenodo, doi*. 2022, roč. 10. Dostupné z DOI: 10.5281/zenodo.3749000.

42. HUNTER, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*. 2007, roč. 9, č. 3, s. 90–95. Dostupné z DOI: 10.1109/MCSE.2007.55.

43. WASKOM, Michael L. seaborn: statistical data visualization. *Journal of Open Source Software*. 2021, roč. 6, č. 60, s. 3021. Dostupné z DOI: 10.21105/joss.03021.

Obsah příloh

- `src`
  - `readme.txt` ..... stručný popis zdrojových kódů
  - `impl` ..... zdrojové kódy implementace společně s výsledky
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF