FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Klasifikace internetového provozu
Student:	Bc. Jana Mašková
Vedoucí:	Ing. Karel Klouda, Ph.D.
Studijní program:	Informatika
Studijní obor:	Počítačová bezpečnost
Katedra:	Katedra informační bezpečnosti
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Cílem práce je studium relevantních postupů strojového učení, které mohou být užitečné pro klasifikaci internetového provozu. Autor bude zkoumat, které atributy by měly být extrahovány ze síťového provozu (PCAP), aby bylo možné identifikovat různé typy komunikace (například video stream, audio stream, XMPP atd.). Hlavní pozornost by měla být zaměřena na konkrétní použití v úlohách souvisejících s bezpečností, jako např. detekce malwaru, detekce botnetů, detekce úniku soukromých informací. Autor by měl nastudovat aktuální techniky v této oblasti a porovnat jejich vlastnosti. Rovněž je vhodné pokusit se vylepšit některý z vybraných postupů. Srovnání bude provedeno na veřejně dostupných datových sadách (např. CTU-13).

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 14. února 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Klasifikace internetového provozu

Bc. Jana Mašková

Katedra informační bezpečnosti
Vedoucí práce: Ing. Karel Klouda, Ph.D.

9. ledna 2020

Poděkování

Hluboce děkuji panu doktoru Karlu Kloudovy za kontrolu psaného textu a úžasné Adéle Bryan za lidský přístup. V neposlední řadě bych chtěla moc poděkovat rodičům a jejich několikaleté podpoře ve studium.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické 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. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 9. ledna 2020 .....

Odkaz na tuto práci

Mašková, Jana. *Klasifikace internetového provozu*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Práce se zabývá celým procesem strojového učení pro klasifikaci internetového provozu a určení škodlivé komunikace. Proces je popsán od získání dat, jejich zpracování až po výběr vhodných příznaků a algoritmů, společně s jejich výsledky. Pro řešení této úlohy byly vybrány supervizované klasifikační algoritmy a algoritmy pro detekci anomálií. Při klasifikaci internetového provozu bylo dosaženo vysoké úspěšnosti pro všechny zvolené datasety pomocí stromových algoritmů. U detekce anomálií bylo dosaženo uspokojivé přesnosti pouze u dvou datasetů ze sedmi.

Klíčová slova Strojové učení, klasifikace, detekce anomálií, počítačová síť, síťová bezpečnost, flow-based klasifikace, předzpracování dat.

Abstract

This thesis delves into the topic of machine learning for the classification of internet traffic and the determination of harmful traffic. All steps of machine learning are considered as data collection and data preprocessing. Suitable classification algorithms and anomaly detection algorithms were chosen to accomplish the main task of the thesis. With regards to the classification of internet traffic, a high success rate was achieved for all selected datasets using

supervised algorithms based on decision tree. For harmful traffic detection, only two of the seven datasets achieved a satisfactory score with used anomaly detection algorithms.

Keywords Machine learning, classification, anomaly detection, network, security network, flow-based classification, data preprocessing.

Obsah

Úvod **1**

1 Přehled teorie k počítačovým sítím a Internetu **3**
1.1 Paket 3
1.2 Síťový tok 4

2 Strojové učení pro klasifikaci internetového provozu **5**
2.1 Proces strojového učení 6
2.2 Základní druhy úloh 6
2.3 Overfitting a underfitting problém 7
2.4 Klasifikační supervizované algoritmy 8
2.5 Shlukovací nesupervizované algoritmy 17
2.6 Ensemble klasifikační modely 18
2.7 Modely pro detekci anomálií 22
2.8 Vyhodnocení výsledků klasifikačních algoritmů 29
2.9 Pojmy používané při klasifikaci internetového provozu 39
2.10 Přístupy ke klasifikaci síťové komunikace 40
2.11 Výzvy na poli klasifikace internetového provozu a detekce anomálií 43

3 Rešerše existujících prací **47**
3.1 Rešerse pro klasifikaci provozu 47
3.2 Rešerse pro detekcí anomálií 50

4 Předzpracování dat **53**
4.1 Typy dat 53
4.2 Šumu, odlehlé hodnoty a anomálie 54
4.3 Čištění dat 57
4.4 Transformace příznaků 60
4.5 Volba a výběr příznaků 67

5 Návrh a implementace **71**
5.1 Volba jazyka 71
5.2 Flow 71
5.3 Vybrané soubory pro tvorbu datasetů 73
5.4 Získání příznaků a tvorba datasetů 77
5.5 Implementace předzpracování dat 80
5.6 Implementace volby a výběru příznaků 92
5.7 Algoritmy strojového učení 94

6 Výsledky **109**
6.1 Výpočetní sestava 109
6.2 Souhrn labelů pro aplikace a kategorie 109
6.3 Použité hodnotící metriky 109
6.4 Čištění dat 111
6.5 Volba a výběr příznaků 112
6.6 Výsledky selekčních algoritmů 122
6.7 Výsledky algoritmů pro extrakci příznaků 124
6.8 Normalizace a transformace vstupních dat 126
6.9 Algoritmy strojového učení 135
6.10 Algoritmy pro detekci anomálií 139
6.11 Kaskádní řešení 141

Závěr **145**

Literatura **147**

A Seznam použitých zkratek **161**

B Obsah přiloženého CD **163**

Seznam obrázků

1.1 Ukázka Uni-flow a bi-flow 4

2.1 Základní druhy úloh strojového učení. 7
2.2 Grafiká ukázka overfitting a underfitting problému [1] 8
2.3 Proces vytvoření modelu pro predikci výstupních hodnot nových záznamů [2] 9
2.4 Rozhodovací strom pro úlohu zda hráč půjde hrát tennis dle počasí. 10
2.5 Mapování vstupního prostoru do prostoru transformovaných příznaků o vyšší dimenzi, ve kterém lze body rozdělit pomocí nadroviny [3] . 14
2.6 Ukázka tří možností rozdělení prostoru nadrovinou. SVM však hledá takové, které maximalizuje možnou vzdálenost od podpůrných vektorů. [4] 15
2.7 Rozdělení závislých ensemble metod podle způsoby využití výsledků předešlé iterace v další iteraci [5] 20
2.8 Detekce anomálií pro dva příznaky X a Y 23
2.9 Isolation forest a vztah hodnoty skóre s vzhledem k průměrné délce cesty záznamu $E(h(x))$ z [6] 26
2.10 Uspořádání neuronů do vrstev v dopředné neuronové síti [7] 29
2.11 Architektura autoencoderu [8] 30
2.12 Zobrazení ROC a AUC metody klasifikace [9] 32
2.13 Grafické zobrazení distribuce dvou tříd s AUC hodnotou rovno jedné a k tomu příslušející ROC křivka [10] 33
2.14 Grafické zobrazení distribuce dvou tříd s AUC hodnotou rovno $0,7$ a k tomu příslušející ROC křivka [10]. 33
2.15 Grafické zobrazení distribuce dvou tříd s AUC hodnotou rovno $0,5$ a k tomu příslušející ROC křivka [10] 34
2.16 Ukázka dvou precision-recall křivek [11] 35
2.17 Ukázka rozdělení trénovacího datasetu při použití křížové validace [12] 35

2.18 Vzniklé modely vytvořené algoritmem strojového učení s vysokým biasem na různých vstupních datasetech charakterující stejný problém [13] . . . 36
2.19 Vzniklé modely vytvořené algoritmem strojového učení s nízkým biasem na různých vstupních datasetech charakterující stejný problém [13] . . . 37
2.20 Učící křivky znázorňující algoritmus strojového učení s vysokým a nízkým biasem [13] . . . 37
2.21 Učící křivky znázorňující algoritmus strojového učení s vysokým a nízkým biasem [13] . . . 38
2.22 Dva krajní případy učících křivek: model s vysokým biasem a nízkou variancí a model s nízkým biasem a vysokou variancí [13] . . . 38
2.23 Nejvhodnější komplexita modelu v závislosti na hodnotě biasu a variance [13] . . . 39

4.1 Rozdělení základních typů dat [14] . . . 54
4.2 Ukázka různých typů šumu [15] . . . 55
4.3 Bodová anomálie [16] . . . 56
4.4 Kontextová anomálie [16] . . . 56
4.5 Graf zobrazující kolektivní anomálii zvýrazněnou červenou barvou [16] . . . 57
4.6 Graf hustoty Weibullova rozdělení pro různé hodnoty parametru c 59
4.7 Vliv influential bodu na sklon regresní křivky [17] . . . 60
4.8 Logaritmická transformace rozdělení vstupního příznaku x [18] . . 63
4.9 Ukázka krabicového grafu původních příznaků (vlevo) a příznaků normalizovaných směrodatnou odchylkou (vpravo) [19] . . . 66
4.10 Ukázka krabicového grafu původních příznaků (vlevo) a příznaků normalizovaných rozpětím (vpravo) [19] . . . 66
4.11 Ukázka tří příznaků a jejich relevance vůči labelu. Příznak $f1$ je relavantní, zatímco příznaky $f2$ a $f3$ jsou irelevantní. Pomocí příznaku $f1$ lze rozlišit mezi dvěma skupinami shluků. Odstranění příznaků $f2$ a $f3$ nebude mít žádný negativní efekt na přesnost algoritmu [20] . . . 68
4.12 Proces klasifikace při výběru příznaků pomocí filtrovací metody. . 68
4.13 Proces klasifikace při výběru příznaků pomocí wrapper metody. . . 69

5.1 Histogram pro délku trvání full-flow pro CICIDS2017 dataset . . . 75
5.2 Histogram pro délku trvání full-flow s více jak jednou transakcí pro CICIDS2017 dataset . . . 76
5.3 Histogram pro délku trvání full-flow s více jak 25 pakety pro CICIDS2017 dataset . . . 77
5.4 Histogram DIntPktIdl a čtyři rozdělení: Exponenciální, Gamma, Paretovo a Weibullovo . . . 85

5.5 Empirická distribuční funkce DIntPktIdl a čtyři rozdělení: Exponenciální, Gamma, Paretovo a Weibullovo 86
5.6 Rozdělení TOS bajtu a význam jednotlivých bitů 88
5.7 TcpOpt příznak je 12-místný řetězec s pevně definovanými znaky na každém místě 89
5.8 Histogram pro TCP sloupec DIntPkt bez transformace 91
5.9 Histogramy pro TCP sloupec DIntPkt po transformaci 103
5.10 Histogram pro TCP sloupec DIntPktMax bez transformace 104
5.11 Histogramy pro TCP sloupec DIntPktMax po transformaci 105
5.12 Dvojice příznaků s vyším korelačním koeficientem než $\pm 0,95$ pro dataset CTU07 a Ordinal kódováním 106
5.13 Třicet nejvíce důležitých příznaků podle algoritmu *LGBMClassifier* pro dataset IDS17 při Argus kódování. 107
5.14 Kumulativní důležitost příznaků s tresholdem na $0,85$ pro dataset IDS17 při Argus kódování. 107
5.15 Graf znázorňující vztah průměrného skóre AUC ROC vůči počtu příznaků pomocí metody *RFECV* pro dataset WIDE. 108

Seznam tabulek

2.1 Matice záměn pro binární klasifikaci 31
2.2 Rozdíl mezi ROC a precision-recall křivkou 34
2.3 Souhrnná tabulka přístupů klasifikace internetového provozu a jejich vlastností [21] 44

4.1 Ukázka Ordinal kódování kategoriálních dat pro kód země zdroje (sCo) 62
4.2 Ukázka OneHot kódování kategoriálních dat pro kód země zdroje (sCo) 62
4.3 Ukázka binárního kódování kategoriálních dat pro kód země zdroje (sCo) 63

5.1 Přehled zkrácených jmen datasetů 74
5.2 Početní a procentuální zastoupení full-flow v datasetu CICIDS2017 75
5.3 Horní kvartil a medián délek trvání full-flow pro zbylé datasety. . . 78
5.7 Důležitost příznaků v algoritmu CART pro dataset WIDE. 84
5.8 Vysvětlení významu znaků v Dir příznaku označujícího směr komunikace 88
5.4 Přehled příznaků spolu s popisem, první část. První tři řádky představují identifikátor komunikace pro flow. 100
5.5 Přehled příznaků spolu s popisem, druhá část. 101
5.6 Přehled příznaků spolu s popisem, třetí část. 102

6.1 Přehled labelů pro aplikace a kategorie 110
6.2 Procentuální vyjádření všech chybějících hodnot v datasetech . . . 115
6.3 Datasety a počet flow po předzpracování 116
6.4 Dvacet nejvíce důležitých příznaků podle algoritmu *LGBMClassifier* 117
6.5 Dvacet nejvíce důležitých příznaků podle algoritmu *DecisionTreeClassifier* 119
6.6 Dvacet nejvíce důležitých příznaků podle algoritmu *LassoCV* . . . 121
6.7 Selekce příznaků pro všechny datasety a obě kódování 123

6.8 Extrakce příznaků pro všechny datasety a obě kódování 125
6.9 Normalizace a její dopad na CTU06 dataset 127
6.10 Normalizace a její dopad na CTU07 dataset při Argus kódování a vlastním výběru příznaků 128
6.11 Normalizace a její dopad na CTU08 dataset při Argus kódování a vlastním výběru příznaků 129
6.12 Normalizace a její dopad na CTU91 dataset při Argus kódování a vlastním výběru příznaků 130
6.13 Normalizace a její dopad na CTU92 dataset při Argus kódování a vlastním výběru příznaků 131
6.14 Normalizace a její dopad na IDS17 dataset při Argus kódování a vlastním výběru příznaků 132
6.15 Normalizace a její dopad na WIDE dataset při Argus kódování a vlastním výběru příznaků 133
6.16 Transformace a její dopad na CTU06 dataset 134
6.17 Transformace a její dopad na WIDE dataset při Argus kódování a vlastním výběru příznaků 135
6.18 Výsledky vybraných algoritmů pro klasifikaci internetového provozu, část první 137
6.19 Výsledky vybraných algoritmů pro klasifikaci internetového provozu, část druhá 138
6.20 Výsledky vybraných algoritmů pro detekci anomálií 140
6.21 Výsledky vybraných algoritmů a datasetů při přidaném příznaků určující kategorii 142
6.22 Výsledky kaskádního algoritmu s klasifikátorem Random Forest . . 144

Úvod

Klasifikace internetového provozu je jedno ze stále aktuálních témat na poli počítačové a síťové bezpečnosti. Její použití je klíčové pro účely správy sítě a detekce anomálií, ale rychle se rozšiřuje do dalších odvětví: například z obchodního hlediska poskytuje cenné marketingové informace prostřednictvím profilování zákazníka [22], zatímco další vědci a vládní společnosti využívají těchto informací především k identifikování globálních internetových trendů.

Klasifikace internetového provozu zastupuje při správě sítě stejně důležitou roli jako například kvalita poskytovaných služeb (Quality of Service, zkráceně QoS) nebo detekce narušení (Intrusion Detection). Analýza síťového provozu poskytuje lepší náhled na chování sítě a může být použita pro optimalizaci a údržbu sítě, analýzu a simulaci normálního provozu a škodlivého provozu, detekci anomálií, filtrování nežádoucí komunikace, vylepšení bezpečnostní politiky sítě, zjištění dosavadních a budoucích požadavků na výkon sítě a další, o tom svědčí i řada studií [23] [24] [25] [26].

Pochopení a znalost distribuce síťového provozu z pohledu aplikací nebo typu provozu je další věc, která značně napomáhá při správě sítě. Pro určení distribuce provozu je zapotřebí jeho správná identifikace, kterou mohou zajistit klasifikační algoritmy. Populární metody klasifikace jako identifikace aplikací na základě čísla portu a nebo obsahu uživatelských dat obsahují nedostatky, které je činí v určitých případech nepoužitelnými, například při peer-to-peer (P2P) komunikaci nebo šifrované komunikaci. Naopak přístup klasifikace využívající strojového učení a statistických údajů o toku dat se zdá být úspěšným směrem.

V posledních letech se klasifikace internetového provozu pomocí strojového učení stává čím dál více probíraným tématem nejen díky zvyšujícímu se množství P2P komunikace a internetovému provozu všeobecně. Prudký rozvoj informačních technologií zapříčinil rychlý rozvoj Internetu a služeb přes něj nabízených. Ze sítě, která byla založena pro vědecké účely se stala celosvětová síť využívána denně miliardou lidí pro osobní i pracovní záležitosti [27] [28].

Podle Cisco studie [29] mělo v roce 2017 přístup k Internetu více jak 45 % lidí na světě, navíc se předpokládá, že do roku 2022 se toto procento celosvětového počtu lidí s přístupem na Internet zvedne ze 45 % na 60 %. S přibývajícím počtem lidí a možnostmi, jak Internet využívat, se také pojí zvýšení množství síťového provozu, to se má do roku 2022 zvýšit až trojnásobně.

Úkolem této diplomové práce je určit vhodné algoritmy strojového učení pro klasifikaci internetového provozu a porovnat jejich úspěšnost. Klasifikace bude prováděna na zvolených datasetech, pro které je potřeba provést předzpracování dat a nalézt nejvhodnější příznaky.

Kapitola 1

Přehled teorie k počítačovým sítím a Internetu

Internet je celosvětová síť vzájemně propojených počítačových sítí. Mezi jednu jeho vlastnot spadá rozdělení aplikačních dat na protokolové datové jednotky nižší vrstvy, které jsou nakonec směřovány po síti k cíli nezávisle na sobě. To jak se komunikace bude rozdělovat, adresovat, přenášet, směřovat a přijímat definuje sada internetových protokolů TCP/IP (Transmission Control Protocol/Internet Protocol).

Teoretický model TCP/IP je rodělen do čtyř vrstev. Každá vrstva poskytuje své služby sousední vyšší vrstvě a využívá služeb sousední nižší vrstvy. Jmenovitě od nejnižší vrstvy k nejvyšší:

- fyzická vrstva,
- síťová vrstva,
- transportní vrstva,
- aplikační vrstva.

1.1 Paket

Teoretický pojem paket slouží v informatice pro označení shluku dat, která jsou přenášená přes síť založené na přepojování paketů. V této práci pojem paket bude využíván právě pro záznamy v PCAP (Packet Capture) souboru zachycené některým protokolovým anlyzérem, jako například Wireshark [30], tcdump [31], a podobně. Záznamy obsahují uživatelská data (payload) a hlavičky dostupných vrstev podle umístění protokolového analyzéru v síti na konkrétní TCP/IP vrstvě. Hlavičky obsahují potřebná data k doručení paketu.

1.2 Síťový tok

Síťový tok, dále v práci uváděn pod anglickým pojmem flow, je označení sekvence jednoho a více po sobě jdoucích paketů cestujících mezi dvěma koncovými uzly počítačové sítě, které sdílejí stejnou pětici údajů: IP adresa zdroje, IP adresa cíle, číslo portu zdroje, číslo portu cíle a použitý síťový protokol [32].

Flow lze dále dělit na uni-flow a bi-flow. Uni-flow obsahuje pouze pakety, které cestují jedním směrem, to odpovídá i výše zmíněné definici. Bi-flow obsahuje pakety cestující oběma směry, jak od zdroje k cíli, tak od cíle k zdroji, za předpokladu, že jsou na obou stranách dodržena i stejná čísla portů a protokol. Dále v této diplomové práci bude pod pojmem flow myšleno vždy bi-flow.

Obrázek 1.1 zobrazuje příklad shluknutí paketů z proběhlé komunikace do uni-flow a bi-flow.

Celá komunikace

Zdrojová IP	Zdrojový port	Cílová IP	Cílový port	Velikost paketu [B]
192.168.1.101	56911	191.168.1.103	80	300
191.168.1.103	80	192.168.1.101	56911	2060
192.168.1.101	56911	191.168.1.103	80	1240

Uni-flow

Zdrojová IP	Zdrojový port	Cílová IP	Cílový port	Počet paketů	Celková velikost paketů [B]
192.168.1.101	56911	191.168.1.103	80	2	1540
191.168.1.103	80	192.168.1.101	56911	1	2060

Bi-flow

Zdrojová IP	Zdrojový port	Cílová IP	Cílový port	Počet paketů zdroje	Počet paketů cíle	Velikost odchozích paketů [B]	Velikost příchozích paketů [B]
192.168.1.101	56911	191.168.1.103	80	2	1	1540	2060

Obrázek 1.1: Ukázka Uni-flow a bi-flow

Kapitola 2

Strojové učení pro klasifikaci internetového provozu

Strojové učení se rozděluje podle dat použitých pro fázi učení na dva základní přístupy a to strojové učení s učitelem (supervizované) a strojové učení bez učitele (nesupervizované). Postupem času vznikl z těchto dvou základních přístupů i přístup kombinující jejich vlastnosti, přístup zvaný semi-supervizované učení.

Tabulková data, se kterými pracují algoritmy strojového učení, se nazývají souhrnně dataset. Dataset může sloužit k natrénování, verifikaci a testování modelu strojového učení. V neposlední řadě pro nová data natrénovaný model strojového učení odvozuje nové skutečnosti, například typ provozu.

Jeden řádek datasetu, který reprezentuje jeden pozorovaný objekt, je označován jako záznam. Každý záznam v datasetu je popsán stejnou množinou statistických a charakteristických vlastností pozorovaného objektu, takzvanými příznaky. Dataset obsahuje vstupní příznaky, které mohou být rozšířeny i o hodnoty výstupních příznaků, labelů. Labelům, které označují správnou výstupní hodnotu záznamu, se říká ground truth labely a jsou brány jako „učitel“. Výstup algoritmu strojového učení je označován jako predikovaná data, labely nebo pouze jako výstupní hodnoty algoritmu.

V případě klasifikace internetového provozu pomocí strojového učení jsou příznaky hodnoty, jež charakterizují internetový provoz a ground truth label udává jméno aplikace, protokolu nebo typu komunikace, která je asociována s daným internetovým provozem. V případě detekce anomálií označuje ground truth label pozitivní třídou všechnu škodlivou komunikaci.

Úkolem supervizovaných algoritmů je najít v datech souvislosti mezi vstupními a výstupními příznaky. Cílem nesupervizovaných algoritmů je seskupit vstupní data do shluků na základě jejich podobností.

2.1 Proces strojového učení

Proces strojového učení probíhá v několika fázích:

- **Předzpracování dat**: Přípravná fáze, během které je dataset připraven do potřebné podoby pro algoritmus strojového učení. Dataset lze před vstupem do algoritmu rozdělit na trénovací, testovací a validační data.
- **Proces učení**:
  - **Trénovací fáze**: Vybranému algoritmu strojového učení jsou nej dříve poskytnuta trénovací vstupní data, na kterých se naučí požad ovanou činnost a k té vytvoří patřičný rozhodovací model.
  - **Testování / validace**: Testovací data jsou použita k určení úspěš nosti a robustnosti naučeného modelu. Úspěšnost predikce lze určit pomocí některé z hodnotících metrik, pokud testovací data obsahují i ground truth labely.
- **Aplikace modelu na další data**: Je-li výsledek naučeného modelu uspokojující, lze naučený model dále použít na predikci dalších dat. V opačném případě lze upravit algoritmus (jeho parametry, jiná volba trénovacích dat, a podobně) a celý proces opakovat.

2.2 Základní druhy úloh

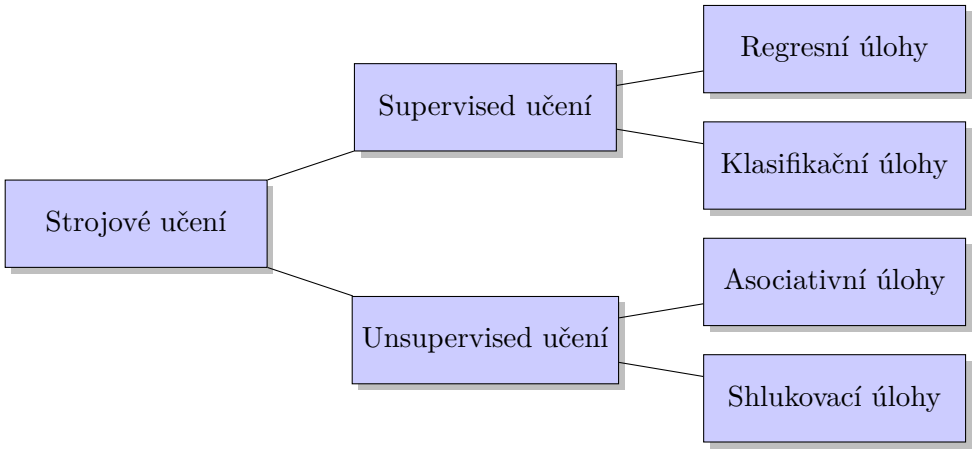
Algoritmy supervizovaného i nesupervizovaného strojového učení se dále dělí na základě typu úloh, které řeší a jejich výstupu.

Supervizované učení lze podle výstupu dělit na regresi a klasifikaci. Hlavní rozdíl mezi regresním a klasifikačním algoritmem je výstup. Výstupem regresního algoritmu jsou spojité hodnoty, nejčastějším příkladem je predikce ceny bytu na základě známých údajů jako počet pokojů, podlahová plocha bytu, lokalita a další. Výstup klasifikačního algoritmu je nejčastěji mapován na pevně danou konečnou množinu hodnot, zvané třídy, a nejčastěji se používá pro přiřazení pozorovaného objektu do právě jedné třídy. V případě klasifikace internetováho provozu se jedná o přiřazení flow ke konkrétnímu jménu aplikace či obecnější vlastnosti provozu, například videostream.

Výstup klasifikačního algoritmu tedy označuje třídu, do které byl záznam přiřazen. Podle počtu unikátních hodnot v labelech se rozlišuje druh klasifikace, je-li výsledek klasifikován pouze do dvou tříd, pak se jedná o binární klasifikaci, jinak vícetřídní. Při binární klasifikaci je výsledek predikován jako pozitivní nebo negativní třída. Pozitivní třída značí třídu zájmu, tu je snaha klasifikovat co nejpřesněji. Záznamy, které neodpovídají charakteristickým vlastnostem pozitivní třídy, spadají do negativní třídy. Datasetu, který obsahuje stejné množství záznamů od každé třídy se říká vybalancovaný dataset.

Nesupervizované učení lze podle typu úlohy dělit na shlukování (clustering) a asociování / přidružování (association). Shlukování sdružuje záznamy na základě jejich podobnosti, příkladem je spam filtr, který porovnává obsah hlavičky, těla a odesílatele s již příjmutou poštou a na základě analogických vlastností přiřadí email do spamu a nebo jako doručenou poštu. Oproti tomu asociování se snaží najít vztahy mezi odlišnými záznamy, které se ale vyskytují pospolu. Ukázkovým příkladem asociování jsou doporučovací systémy, například v internetovém obchodě si zákazníci často kupují máslo a chleba dohromady, proto lze odvodit asociaci mezi máslem a chlebem a při koupi chleba je jako další produkt doporučeno máslo.

Tato diplomová práce se zabývá pouze klasifikačními supervizovanými algoritmy, které klasifikují do předem pevně dané výstupní množiny tříd a shlukovacími nesupervizovanými algoritmy, které hledají podobnosti mezi daty.



Obrázek 2.1: Základní druhy úloh strojového učení.

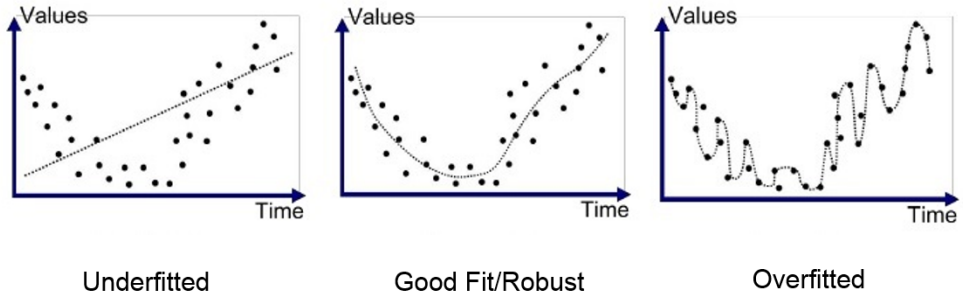
2.3 Overfitting a underfitting problém

Vstupem do algoritmu strojového učení jsou záznamy, v případě supervizovaného učení i ground truth labely. Každý záznam x je reprezentován jako řádkový vektor, který je složen ze vstupních příznaků $x = (x_1, \ldots, x_n)$. Množina záznamů je uspořádaná do matice X, jejíž řádky reprezentují záznamy a sloupce příznaky. Příslušející ground truth labely Y k matici X jsou reprezentovány ve sloupcovém vektoru.

Generalizace dat je proces, při kterém model vyhodnocuje data ze širšího pohledu a tudíž je schopen nalézt i všeobecnější vlastnosti, ty mu umožňují predikovat vhodné labely i pro nová data, která nebyla přítomna při vytváření modelu.

Úkolem strojového učení je nalézt takovou aproximační funkci f, která zobrazuje vstupní data X na hodnoty jejich ground truth labelů Y:

$$Y = f(X).$$



Obrázek 2.2: Grafiká ukázka overfitting a underfitting problému [1]

Obrázek 4.1 znázorňuje dva případy problémů, ke kterým může dojít během procesu strojového učení, a to underfitting a overfitting.

Underfitting problém, nastává pokud vzniklý statistický model nedokáže přiměřeně popsat strukturu vstupních dat ani predikovat nová data. Underfitted model je takový model, který je příliš jednoduchý a je nutné navýšit jeho složitost, například dodat některé parametry, zvýšit stupeň polynomu či přidat další příznaky popisující záznam [33].

Overfitting problém naopak nastává, pokud je model příliž složitý a obsahuje více neznámých parametrů, než data dokážou pokrýt. Overfitted model je takový model, který popisuje vstupní data až moc dobře a výsledný model zahrnuje i odlehlé hodnoty a šum. To má za následek, že model umí perfektně predikovat labely pro data, ze kterých byl model vytvořen, ale neumí generalizovat nová data a to vede k neuspokojivé přesnosti jejich predikce [34].

Ideálním případem je nalezení takového robustního modelu, který je odolný vůči odlehlým hodnotám a šumu a zároveň dokáže predikovat hodnoty nových dat, která jsou generovaná stejným procesem, co vstupní data X.

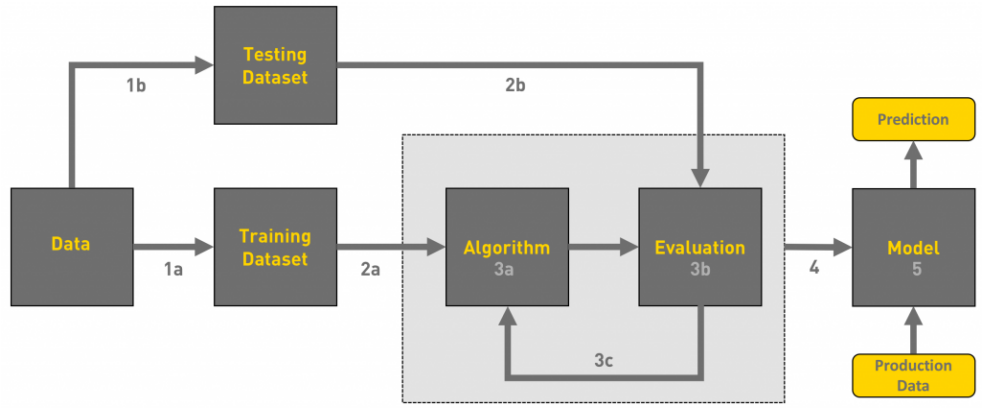
2.4 Klasifikační supervizované algoritmy

Natrénovaný model klasifikačního supervizovaného algoritmu je funkce f (na obrázku 2.3 číslo 5), zvaná též klasifikátor, která přiřazuje novému záznamu x_{new} výstupní hodnotu y_i, label. V případě klasifikačních algoritmů label nabývá hodnoty z předem určené konečné množiny hodnot $\{y_1, \ldots, y_k\}$.

K vytvoření funkce f je zapotřebí klasifikačního supervizovaného algoritmu, matice trénovacích záznamů X_{train} a sloupcového vektoru příslušejících ground truth labelů Y_{train}. Dvojice (X_{train}, Y_{train}) se nazývá trénovací dataset a algoritmus se na něm „učí“ určitému vstupu přiřadit určitý výstup

na základě hodnot příznaků. Natrénovaný model umí následně predikovat labely dle naučených vlastností pro další vstupní data. Správnost natrénovaného modelu lze ověřit na testovacím datasetu, někdy nazývaným validačním, který je reprezentován novou dvojicí (X_{test}, Y_{test}), matice X_{test} obsahuje data, která nebyla přítomna v matici X_{train}. Do naučenéo modelu vstupuje pouze matice X_{test} a ten vrací predikované hodnoty labelů Y_{pred}, které slouží k určení úspěšnosti modelu pomocí metriky využívající znalost hodnot Y_{pred} a ground truth labelů Y_{test}, více o hodnotících metrikách v kapitole 6.3.

Na obrázku 2.3 zobrazuje $1a$ a $1b$ rozdělení datasetu na trénovací a testovací data. Čerchovaný box 3 popisuje proces učení a validaci modelu a jeho případné úpravy podle dosažených výsledků, box 5 reprezentuje finální model, jenž může být použit na nová vstupní data, pro které predikuje labely.

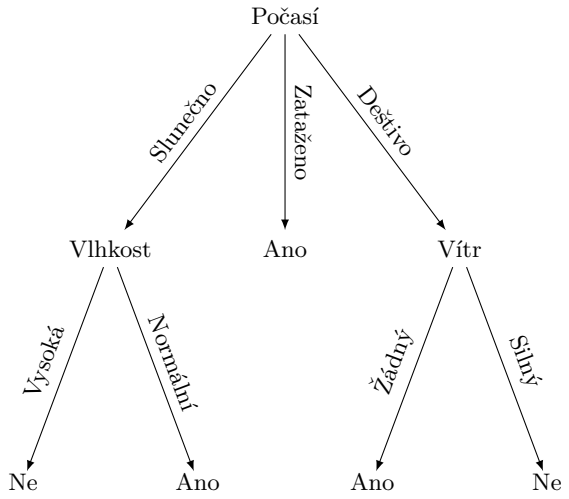


Obrázek 2.3: Proces vytvoření modelu pro predikci výstupních hodnot nových záznamů [2]

2.4.1 Rozhodovací strom

Rozhodovací strom (decision tree), je sada hierarchicky uspořádaných rozhodovacích pravidel tvořících stromovou strukturu. Uzly stromu představují větvení na základě hodnoty právě jednoho příznaku, větvení je obvykle implementováno pomocí if-then-else pravidla. Z uzlu vede konečný počet hran (nejčastěji se používá binární strom s právě dvěma hranami vedoucí z každého vnitřního uzlu) a každá hrana reprezentuje výsledek rozhodnutí. Klasifikační proces spočívá v přechodu mezi uzly od kořene k listům na základě jednotlivých rozhodnutí a jejich výsledků. Listy stromu obsahují výslednou hodnotu klasifikace. Rozhodovací stromy lze využít na binární predikci i vícetřídní predikci.

Výhoda této metody je její přehlednost a snadná interpretace, která umožňuje rychle a jednoduše pochopit dané výsledky a identifikovat klíčové položky. Další výhoda stromů je, že vstupní data nemusí procházet fází přadzpracování



Obrázek 2.4: Rozhodovací strom pro úlohu zda hráč půjde hrát tennis dle počasí.

dat, přesněji transformací a normalizací, více informací v kapitolách 4.4.2 a 4.4.3.

Rozhodovací stromy dokáží samy rozpoznat, které příznaky jsou opravdu klíčové a které naopak lze vypustit z modelu. Tato vlastnost je velmi přínosná pokud dataset obsahuje velké množstvím příznaků v řádu desítek a více. Ostatní typy algoritmů často nedokáží určit klíčové a neklíčové příznaky, a proto pro dosažení lepších výsledků potřebují na vstupu dostat již vybrané klíčové příznaky.

Trénování rozhodovacích stromů spočívá v postupném větvení stromu od kořene na základě trénovacích dat. V každém uzlu se extrahují pravidla tak, aby výsledné rozložení co nejlépe separovalo data ve smyslu zvolené kriteriální statistiky. Pro každý příznak se najde nejlepší rozdělovací pravidlo a následně je vybrán příznak, který má nejlepší hodnotu kritéria.

Kriteriální statistika: Kriteriální statistika zajišťuje volbu vhodného příznaku do uzlu a určení větvení. Literatura o rozhodovacích stromech obsahuje řadu algoritmů, které se různí právě ve volbě kriteriální statistiky. V případě klasifikačního problému nejpoužívanějšími kritérii jsou Informační zisk (neboli relativní entropie, též zvané Kullbackova–Leiblerova vzdálenost, anglicky Information gain) a Gini index.

Strom vybírá do rozhodovacího uzlu ten příznak, který dosahuje nejlepšího skóre dle kriteriální statistiky. Tím se příznaky, které nejvíce určují správnou hodnotu labelu, ocitávají na vrcholu rozhodovacího stromu a směrem dolů do uzlů se vysytují příznaky nesoucí méně informace o hodnotě labelu.

Entropie: Nechť překvapení při pozorování hodnoty y_i diskrétní náhodné veličiny X, je dáno vzorcem:

$$I(y_i) = I(X = y_i) = -\log P(X = y_i),$$

kde $P(X = y_i)$ je pravděpodobnostní distribuce matice záznamů X, které spadají do třídy y_i, $i \in (1, \ldots, k)$. Pak vzorec entropie diskrétní náhodné veličiny X s počtem k tříd je následující:

$$H(X) = \mathbb{E}I(X) = -\sum_{i=1}^{k} P(X = y_i) \log(P(X = y_i)).$$

Entropie v případě klasifikace záznamů udává zastoupenost klasifikačních tříd v datasetu, neboli jak je vyvážený. V případě je-li entropie rovna 0, pak všechny záznamy spadají pouze pod jednu klasifikační třídu v opačném případě, je-li entropie rovna 1, jsou všechny klasifikační třídy zastoupeny stejným počtem záznamů. Nízká hodnota entropie udává malou překvapenost při určování třídy a též vysokou nevyváženost datasetu.

Information gain: Je metrika z teorie informace, která udává kolik informací o labelu nese každý z příznaků nebo také lze zjednodušeně říci, udává předpokládaný úbytek entropie způsobený rozdělením záznamů na základě vybraného příznaku, neboli jak dobře vybraný příznak rozděluje záznamy do tříd. V uzlu je vybrán příznak, který maximalizuje tuto hodnotu.

K vypočtení Information gain je potřeba definovat podmíněnou entropii pro dataset X vzhledem k vybranému příznaku a a všem jeho hodnotám, které nabývá a_j.

$$H(X \mid a) = -\sum_{j} P(a = a_j) \sum_{i} P(X = y_i \mid a = a_j) \log_2(P(X = y_i \mid a = a_j))$$

Information gain $IG(X, a)$ se vypočte jako redukce entropie datasetu X za předpokladu volby příznaku a:

$$IG(X, a) = H(X) - H(X \mid a).$$

Gini index: Nebo také Gini impurity měří, jak často by se klasifikátor spletl, pokud by záznamům X z jednoho podstromu byla přiřazena náhodná třída klasifikace y_i, kde $i \in 1, \ldots, k$. Gini index měří pravděpodobnost, že záznamy patří do třídy y_i s pravděpodobností $P(X = y_i)$, zkráceně psanou jako p_i, ale klasifikuje je jinou třídou $(1 - p_i)$ pro každou ze tříd. Ve výsledku se hledá podmínka, která minimalizuje Gini index $GI(X)$:

$$GI(X) = \sum_{i=1}^{k} p_i(1 - p_i) = 1 - \sum_{i=1}^{k} p_i^2.s$$

Algorithm 1 Pseudokód algoritmu rozhodovací strom *DecisionTree(X, Y, attr)* s využitím Gini indexu.

Funkce: *exNode(X, l_j)* značí list stromu a všem vstupním datům X je přiřazena hodnota labelu l_j. *inNode* značí uzel stromu, ve kterém jsou data rozdělena na základě příznaku *SplitAtt* a jeho hodnoty *SplitValue*. Funkce $gini\_index(X, Y, a)$ vrací hodnotu Gini indexu pro příznak a a vstupní data X s ground truth labely Y.

Input: Množina vektorů trénovacích dat $X = \{x_1, \ldots, x_m\}$, množina ground truth labelů $Y = \{y_1, \ldots, y_m\}$ pro data X, která nabývají hodnot z $L = \{l_1, \ldots, l_t\}$. Množina jmen příznaků $attr$.

Output: Naučený rozhodovací strom *DecisionTree*.

if X obsahuje záznamy patřící pouze do jedné třídy $l_j \in L$ **then**
    return $exNode(X, l_j)$ ▷ List stromu
else if splněná, některá z ukončujících podmínek **then**
    $l \leftarrow mode(Y)$ ▷ Nejčastěji se vyskytující label
    return $exNode(X, l)$
else
    $values \leftarrow$ inicalizace množiny
    for $a \in attr$ **do**
        $values \leftarrow values \cup gini\_index(X, Y, a)$
    # Vyběr nejlepšího příznaku na základě kriteriální statistiky Gini index
    $a_{best} = min(values)$
    $p \leftarrow$ rozdělující bod pro příznak a_{best}
    $X_l, Y_l \leftarrow filter(X, Y, a_{best} < p)$
    $X_r, Y_r \leftarrow filter(X, Y, a_{best} \geq p)$
    return $inNode\{Left \leftarrow DecisionTree(X_l, Y_l, attr \setminus a_{best}),$
        $Right \leftarrow DecisionTree(X_r, Y_r, attr \setminus a_{best}),$
        $SplitAtt \leftarrow a_{best},$
        $SplitValue \leftarrow p\}$ ▷ Uzel stromu

2.4.1.1 C4.5

C4.5 patří mezi nejznámější a nejvíce využívané implementace rozhodovacího stromu [35]. C4.5 zakládá výběr příznaku do rozhodovacího uzlu na hodnotě Information gain nebo poměrného informačního zisku (information gain ratio, IGR), což je Information gain podělený hodnotou entropie zkoumaného příznaku:

$$IGR(Y;X) = \frac{IG(Y;X)}{H(X)}.$$

Výhody C4.5 algoritmu jsou, že umí pracovat s chybějícími daty a po sestavení stromu jsou navíc ořežány větve stromu, které mají slabý vliv na výslednou klasifikaci. Ořezáním stromu (pruning) se docílí zmírnění overfitting problému [36].

C4.5 umí pracovat s chybějícími daty jak v trénovacích tak testovacích datech. Pro testovací záznam s chybějícím příznakem je vráceno pravdě podobnostní rozdělení labelů, do kterých záznam spadá. To je vypočteno pro všechny labely, které se vyskytují v podstromu daného uzlu, který provádí větvení na základě chybějícího příznaku. V trénovacím fázi je upraven vzoreček pro výpočet Information gain nebo Information gain ratio tak, aby dokázal pracovat i s chybějícími příznaky tím, že je nezapočítává. Neboli výsledná hodnota zisku je vynásobena poměrem nechybějících dat vůči celkovému počtu a entropie se počítají pouze pro nechybějící data [37]. Více informací v knize [38].

2.4.1.2 CART

Classification and Regression Tree, CART, je rozhodovací strom, který vybírá příznak do uzlu stromu na základě Gini indexu nebo nejvyšší hodnoty Information gain. Neumí pracovat s chybějícími hodnotami a nejčastěji je implementován jako binární strom [39].

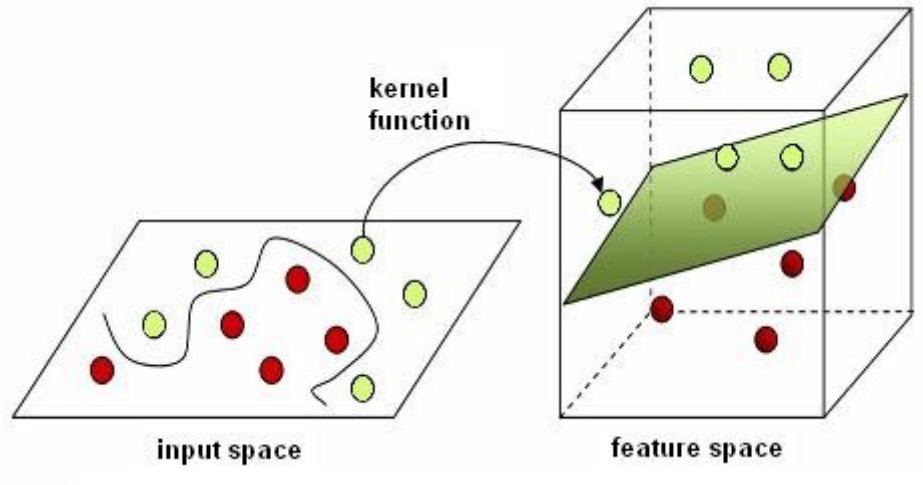
2.4.1.3 Reduced Error Pruning Tree

Reduced Error Pruning Tree klasifikátor, zkráceně REPTree, je jeden z nejrychlejších implementací rozhodovacího stromu. Stromová struktura je implementována za pomoci informačního zisku a nebo pomocí metody nejmenších čtverců. REPTree využívá stromovou logiku k sestavení několika podstromů pro daný rozhodovací uzel, ze kterých je následně vybrán ten nejlepší. Vytvořený model je na konci učící fáze ještě prořezán metodou reduced error pruning s backfitting [40]. Stejně jako C4.5 algoritmus umí pracovat s datasetem, který obsahuje chybějící data.

2.4.2 Support Vector Machine

Algoritmus Support Vector Machine (SVM) byl prvně představen [41]. Hlavní princip SVM je najít optimální nadrovinu ve vícedimenzionálním vektorovém prostoru vstupních dat, která rozděluje body spadající do různých tříd a zároveň maximalizuje svou vzdálenost od nejbližších bodů k nadrovině. Velká výhoda SVM je možnost transformování nelineární klasifikačního problému na lineární. Pro komplexnější popis, jak binární SVM funguje, lze odkázat na práci [42] a [43].

Transformace nelineárního klasifikačního problému na lineární je klíčová idea SVM algoritmu. Při řešení nelineárního problému nastává, že nelze najít takovou nadrovinu, která lineárně oddělí data. Proto SVM zobrazuje vstupní prostor trénovacích dat použitím jádrového triku (kernel trick) do prostoru vyšší dimenze až nekonečné dimenze, ve které mohou být body jednodušeji rozděleny nadrovinou, viz obrázek 2.5.

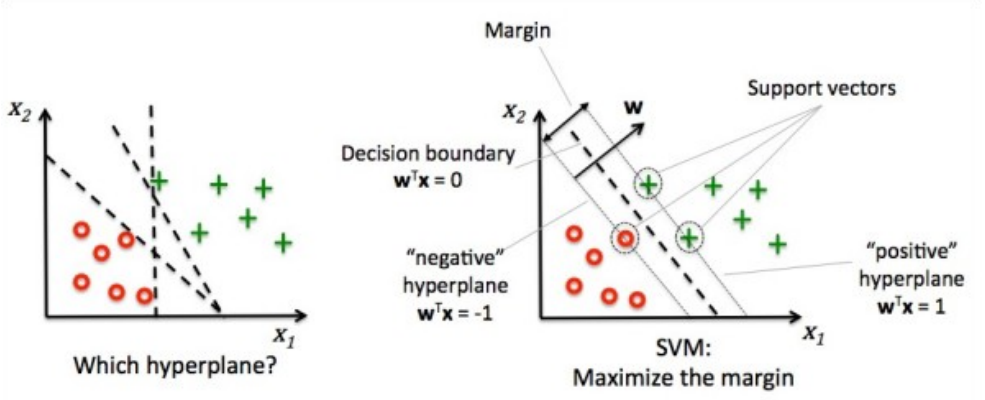


Obrázek 2.5: Mapování vstupního prostoru do prostoru transformovaných příznaků o vyšší dimenzi, ve kterém lze body rozdělit pomocí nadroviny [3]

Body, které se vyskytují na okraji své třídy a určují klasifikační nadrovinu, se nazývají podpůrné vektory (support vectors), viz obrázek 2.6.

Během klasifikační fáze SVM algoritmus klasifikuje nový bod pomocí podpůrných vektorů podle toho, do které části transformovaném prostoru nový bod spadá.

Transformace do vícedimenzionálního prostoru je obvykle popisována pomocí jádrové (kernel) funkce, která definuje skalární součin vektorů v novém prostoru. Jako kernel funkce SVM se nejčastěji používá Gaussian kernel i díky jeho dobrým dosahovaným výsledkům, avšak SVM nabízí širší výběr kernel funkcí, nejčastější jsou: lineární, polynomiální, sigmoid a již zmíněný Gaus-



Obrázek 2.6: Ukázka tří možností rozdělení prostoru nadrovinou. SVM však hledá takové, které maximalizuje možnou vzdálenost od podpůrných vektorů. [4]

sian kernel. SVM algoritmus poskytuje obvykle dobrý výkon a výsledky i bez nutnosti vylepšování algoritmu a to i na poli klasifikace internetového provozu.

Níže jsou uvedeny definice kernel funkcí z Python knihovny *scikit-learn* [44], kde: x_i a x_j jsou dva záznamy (řádkové vektory příznaků), $\langle x_i, x_j \rangle$ skalární součin vektrů x_i a x_j, d stupeň polynomu, $r \geq 0$ a $\gamma > 0$ konstanty.

$$\begin{aligned}
\text{Lineární kernel: } K(x_i, x_j) &= \langle x_i, x_j \rangle \\
\text{Polynomiální kernel: } K(x_i, x_j) &= (\gamma \cdot \langle x_i, x_j \rangle + r)^d \\
\text{Sigmoid kernel: } K(x_i, x_j) &= \tanh(\gamma \cdot \langle x_i, x_j \rangle + r) \\
\text{Gaussian kernel: } K(x_i, x_j) &= exp(-\gamma ||x_i - x_j||^2)
\end{aligned}$$

SVM algoritmus lze po úpravě také použít na vícetřídní klasifikaci, která obvykle využívá principu jeden-vs-jeden (one-vs-one) nebo jeden-vs-zbytek (one-vs-rest). Jeden-vs-jeden vícetřídní SVM klasifikátor vytvoří pro každou kombinaci dvojic tříd jeden binární SVM algoritmus a bod je klasifikován na základě hlasování binárních SVM klasifikátorů do třídy, která obdrží nejvíce hlasů. Jeden-vs-zbytek vícetřídní klasifikátor vytváří jeden SVM klasifikátor pro každou výstupní třídu, kde záznamy ze zvolené výstupní třídy označí jako jednu skupinu a zbylé záznamy z ostatních tříd označí jako druhou skupinu. Nový bod je klasifikován na základě SVM klasifikátoru, který vrací nejvyšší skóre pro spolehlivost (confidence score).

Nevýhody SVM spočívají v jeho nemožnosti srozumitelné interpretace a tedy nemožnosti porozumět, co je příčinou dobrých nebo špatných výsledků. Spolu s tímto problémem souvisí i obtížná volba kernel parametrů a kernel

funkce samotné. Další nevýhoda spočívá ve výpočetní složitosti, jelikož počet operací roste lineárně s počtem přibývajících support vectors.

2.4.3 K-Nearest Neighbours

V češtině k-nejbližších sousedů, zkráceně označován jako k-NN či kNN algoritmus. Tento algoritmus je možné použít jako supervizované i nesupervizované učení. Princip k-NN spočívá ve výpočtu vzdáleností všech záznamů od nového záznamu (nejčastěji se používá Euklidova vzdálenost), ze kterých je vybráno k-nejbližších sousedů. V případě supervizovaného učení je novému záznamu přiřazen takový label, který má největší zastoupení ve vybrané množině k-nejbližších sousedů.

Algorithm 2 Pseudokód klasifikace nového záznamu pomocí k-NN modelu *kNN(*X*,* Y*,* $x_{unknown}$*,* k*)*

Input: Množina vektorů trénovacích dat $X = \{x_1, \ldots, x_m\}$, množina ground truth labelů $Y = \{y_1, \ldots, y_m\}$ pro data X, nový záznam bez labelu $x_{unknown} \notin X$, počet nejbližších sousedů $k \in \mathbb{N}$.

Output: Label pro nový záznam $x_{unknown}$.

```
d ← funkce pro výpočet vzdálenosti
D ← inicializace množiny
for x_i ∈ {x_1, ..., x_m} do
    D ← D ∪ d(x_i, x_unknown)
# Urči k-nejbližčích sousedů záznamu x_unknown
nearest_index ← min(D, k).index          ▷ Indexy k nejmenších hodnot
return mode(Y[nearest_index])            ▷ Nejčastěji se vyskytující label
```

2.4.4 Naive Bayes

Naive Bayes, zkráceně nazývaný NB, je základní stavební prvek mnoha studií. Využívá Bayesova teorému k analýze trénovacích dat pro optimální výběr odhadované hodnoty za předpokladu, že jednotlivé vstupní příznaky jsou téměř nezávislé. Předpokládá, že vstupní příznaky jsou na sobě nezávislé, v tom případě změna hodnoty jednoho příznaků nemá vliv na žádnou hodnotu jiného příznaku [45].

Naive Bayes přiřazuje záznamu podmíněné pravděpodobnosti jednotlivých tříd za podmínky hodnot vstupních příznaků. Použitím Bayesovského pravidla a předpokladu nezávislosti lze pravděpodobnost třídy $y_k \in Y$ podmíněnou vektorem příznaků $x \in X$, kde $x = \{x_1, \ldots, x_m\}$, napsat jako:

$$P(y_k \mid \{x_1, \ldots, x_m\}) = 1/Z \cdot p(y_k) \prod_{i=1}^{m} p(x_i \mid y_k),$$

kde Z je normalizační konstanta. Klasifikátor pak klasifikuje záznam do třídy s největší pravděpodobností.

Algorithm 3 Pseudokód klasifikace nového záznamu pomocí modelu Naive Bayes *NB(L, $x_{unknown}$)*

Input: Množina klasifikačních tříd $L = \{l_1, \ldots, l_k\}$, nový záznam bez labelu $x_{unknown} \notin X$.

Output: Label l_j, pro který je podmíněná pravděpodobnost největší.

```
P ← funkce podmíněné pravděpodobnosti
max_y ← 0.0
for pro všechny l_j ∈ {l_1, ..., l_k} do
    value ← P(l_j | x_unknown)
    if value > max_y then
        max_y ← value
return max_y
```

2.5 Shlukovací nesupervizované algoritmy

2.5.1 K-Means

K-Means algoritmus představuje typického zástupce shlukovacích algoritmů, který seskupuje podobná data do stejného shluku a naopak odděluje data s odlišnými vlastnostmi do různých shluků. To vše je založeno na metodě měřící vzdálenosti jako například Euklidova vzdálenost. Každý shluk je charakterizován centroidem – středem shluku. Základním cílem K-Means algoritmu je iterativní hledání lepších centroidů. K-Means hledá takové centroidy, které minimalizují vzdálenost bodů uvnitř shluků ale zároveň maximalizují vzdálenost bodů z dalších shluků, které jsou nejblíže jeho bodům.

Algoritmus na začátku vybere K náhodných záznamů ze vstupních dat X a určí je za centroidy, zbylé body z trénovací množiny přiřadí k nejbližšímu centroidu, tím vzniknou shluky. Poté vypočítá vzdálenosti centroidů ke všem bodům ve shluku a určí novou hodnotu centroidu tak, aby nový určený bod v prostoru byl těžištěm shluku. K novým centroidům opět přiřadí body a vypočte nové vzdálenosti. Tento krok se iterativně opakuje dokud algoritmus K-Means neskončí ve stavu, kdy se centroidy již nemění nebo po určitém množství iterací.

Algorithm 4 Pseudokód algoritmu K-Means *KMeans(X, K)*

Input: Množina vektorů dat $X = \{x_1, \ldots, x_m\}$, počet hledaných centroidů $K \in \mathbb{N}$.

Output: Vektor centroidů $C = \{c_1, \ldots, c_m\}$, hodnota c_i udává do kterého shluku spadá záznam x_i.

```
d ← funkce vzdálenosti
# Množina C = {c_1, ..., c_m} nabývá hodnot z {1, ..., K}
C ← inicializace množiny o velikosti |X|
# Množina M = {m_1, ..., m_K} centroidů, m_i ∈ R^n
M ← RandomChoose(X, K)  ▷ Náhodný výběr K centroidů ze záznamů
for Opakuj dokud není splněna ukončující podmínka do
    for i ∈ {1, ..., m} do
        D ← inicializace množiny o velikosti |K|
        # Vypočti vzdálenosti záznamu x_i od všech centroidů z M
        D_xi ← {d(x_i, m_j) | ∀j ∈ {1, ..., K}}
        # Najdi centroid, který je nejblíže záznamu x_i
        c_i ← M[min(D_xi).index]
    for k ∈ {1, ..., K} do
        # Vypočti novou hodnotu centroidu, jako průměrnou hodnotu všech záznamů v centroidu
        M ← mean({x_i | ∀i : c_i == k})
return C
```

K-Means rozdělí vstupní data do K disjunktních shluků. Jsou-li na vstupu i labely, pak lze novému vstupnímu záznamu přiřadit takovou hodnotu labelu, kterou je označeno nejvíce záznamů ve shluku, kam spadá nový záznam. Nejsou-li na vstupu žádné labely, pak se musí manuálně určit, co který shluk představuje.

2.6 Ensemble klasifikační modely

Ensemble algoritmy využívají kombinace slabých klasifikátorů strojového učení k dosažení lepšího výsledku, než jakého by mohlo být dosaženo jednotlivými algoritmy zvlášť [46].

Slabý klasifikátor je takový model strojového učení, který si v řešení úlohy vede o trochu lépe, než pokud by byl výsledek predikován náhodně. Náhodná predikce u problému binární klasifikace při vybalancovaném datasetu dosahuje 50 % úspěšnosti, proces může být přirovnán hodu vyváženou mincí pro řešení binární klasifikace.

Ensemble metody lze rozdělit podle závislosti jednotlivých algoritmů na závislé a nezávislé. Jsou-li jednotlivé algoritmy na sobě závislé, jsou spouštěny

sériově za sebou a výsledky předešlé iterace ovlivňují vstup do algoritmu v následující iteraci.

Jsou-li jednotlivé algoritmy na sobě nezávislé, pak mohou být spouštěny paralelně. Nezávislé ensemble modely rozdělí vstupní trénovací dataset na tolik podmnožin, kolik je definovaných algoritmů strojového učení v konečném setu. Výsledky jednotlivých algoritmů jsou určitým způsobem zkombinovány a je určen finální výsledek.

2.6.1 Závislé ensemble metody

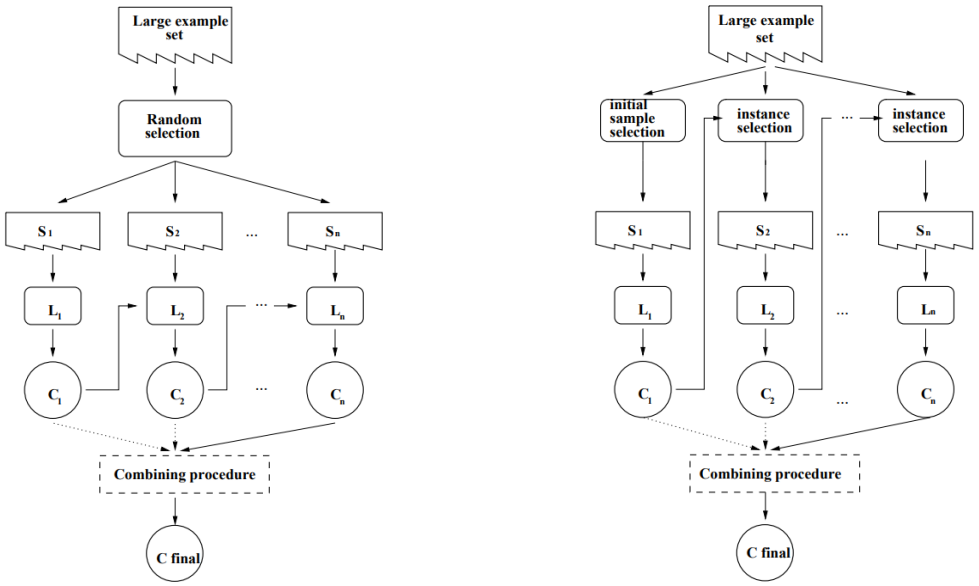
Závislé ensemble metody lze dále dělit podle způsobu využití výsledků předešlé iterace v další iteraci [5] na:

- Kaskádní přístup (Incremental batch learning [47])
  - Vstupem do dalšího algoritmu strojového učení je výsledek předešlého algoritmu spolu s novým trénovacím setem pro následující algoritmus strojového učení.
  - Lze kombinovat více různých algoritmů strojového učení. Sled kaskádně spouštěných algoritmů využívajících těchto vlastností lze souhrnně pojmenovat jako kaskádní algoritmy.
- Iterační přístup (Model-guided instance selection [46])
  - Model využívá iterativního spuštění jednoho stejného algoritmu strojového učení, kde výsledek předešlé iterace klasifikátoru je použit pro úpravu vstupu trénovacích dat do následující iterace. Další iterace se více zaměřuje na vstupní data, které byly v předešlé iteraci nesprávně klasifikovány [46].
  - Zástupcem model-guided instance selection je boosting a jeho nej známější implementace AdaBoost.

Na obrázku 2.7 je S_i podmnožina vstupních dat a L_i zvolený algoritmus strojového učení. C_i je výstup i-tého algoritmu strojového učení, například se může jednat o extrahovaný set pravidel z rozhodovacího stromu popisující vztah mezi vstupními příznaky a labelem.

2.6.1.1 AdaBoost

AdaBoost je zkratka pro Adaptive Boosting. Hlavní idea algoritmu spočívá v přiřazování vah jednotlivým záznamům, kde váha značí obtížnost klasifikace daného záznamu. Na začátku jsou všem záznamům přiřazeny stejné váhy, ale v každé iteraci je pro další slabý klasifikátor tato váha pozměněna na základě úspěchu klasifikace předešlého klasifikátoru. Jsou-li záznamy klasifikovány mylně váha se zvyšuje, naopak jsou-li záznamy klasifikovány správně



(a) Kaskádní přístup

(b) Iterační přístup

Obrázek 2.7: Rozdělení závislých ensemble metod podle způsoby využití výsledků předešlé iterace v další iteraci [5]

váha se snižuje. Výsledkem úpravy vah se další slabý algoritmus více zaměřuje na mylně klasifikované záznamy, tudíž záznamy s větší váhou. Původní algoritmus AdaBoost byl vytvořen pro řešení binární klasifikace, avšak nové verze už umožňují pracovat s vícetřídní klasifikací. Nevýhoda tohoto přístupu je, že řešení problému může vést k overfitted modelu. Takový model může dosahovat menší úspěšnosti predikce, než kdyby byl použit slabý klasifikátor samotný bez iterací a vah. Pro vyřešení nebo utlumení tohoto problému se doporučuje použít co nejméně iterací, co lze, ale přitom tolik, aby výsledný model dosahoval lepšího výsledku než samostatný slabý klasifikátor.

2.6.2 Nezávislé ensemble modely

V případě nezávislých ensemble modelů je nový vstupní záznam klasifikován na základě určité kombinace výsledků jednotlivých algoritmů, které pracují nezávisle na sobě. Metody, jež kombinují jednotlivé výsledky a určují konečný výsledek, se jmenují kombinační metody. Mezi nejčastější kombinační metody používané pro rozhodnutí klasifikace spadají metody založené na vahách jednotlivých klasifikátorů, hlasování, součtech pravděpodobností, entropie a další.

Nejznámějším zástupcem nezávislých ensemble modelů je bagging metoda a jeho konkrétní impelentace pomocí náhodných lesů.

Algorithm 5 Pseudokód algoritmu AdaBoost *AdaBoost(X, Y, T, WeakLearner)*

Input: Množina vektorů trénovacích dat $X = \{x_1, \ldots, x_m\}$, množina ground truth labelů $Y = \{y_1, \ldots, y_m\}$ pro data X, $T \in \mathbb{N}$ počet iterací, *WeakLearner* označuje slabý klasifikátor.

Inicializace vah všech záznamů, $w_i(j)$ značí váhu j-tého záznamu v i-té iteraci
$w_1(i) \leftarrow 1/m$ pro $i \in \{1, \ldots, m\}$
for $t \in \{1, \ldots, T\}$ **do**
  $p_t(i) \leftarrow \frac{w_t(i)}{\sum_i w_t(i)}$ pro $\forall i$ ▷ Výpočet distribuce vah
  # h_i značí slabý klasifikátor v i-té iteraci
  $h_t \leftarrow WeakLearner(X, Y, p_t)$
  # ϵ_i značí minimální chybu špatné klasifikace pro daný naučený model
  $\epsilon_t \leftarrow \sum_{\forall i: h_t(x_i) \neq y_i} p_t(i)$
  if $\epsilon_t > 1/2$ **then**
    # Přiřad všem váham uniformní rozdělení
    $w_t(i) \leftarrow 1/m$ pro $i \in \{1, \ldots, m\}$
    $t \leftarrow t - 1$
    Continue ▷ Začni nový cyklus
  $w_{t+1}(i) \leftarrow w_t(i) * \begin{cases} \frac{\epsilon_t}{1-\epsilon_t} & \text{pokud } h_t(x_i) = y_i, \\ 1 & \text{jinak.} \end{cases}$

2.6.2.1 Bagging

Bagging je zkrácenina pro Bootstrap Aggregating. Síla algoritmus spočívá v tvorbě vylepšeného klasifikačního modelu na základě kombinace několika slabých klasifikátorů. Bagging, narozdíl od boosting modelu, vybírá všechny záznamy se stejnou pravděpodobností. Díky tomuto přístupu bagging vyžaduje použití algoritmů, které jsou nestabilní. Nestabilní algoritmus je takový, který vykazuje výrazné změny v klasifikaci při použití drobně pozměněných trénova cích dat. Zatímco boosting vyžaduje pouze udržení množství nesprávně klasifikovaných záznamů z každé třídy okolo 50 %.

Náhodné lesy Náhodné lesy (Random Forest) je ensemble baggin metoda, která využívá rozhodovacích stromů jako slabých klasifikátorů a na základě nejčastějšího výsledku neprořezaných stromů klasifikuje nový záznam. Výhoda Random Forest je, že umí efektivně pracovat s velkým množstvím příznaků a dosáhne výsledného modelu v dobrém výpočetním čase.

Algorithm 6 Pseudokód vytvoření bagging modelu *Bagging(X, Y, t, p, WeakLearner)*

Input: Množina vektorů trénovacích dat $X = \{x_1, \ldots, x_m\}$, množina ground truth labelů $Y = \{y_1, \ldots, y_m\}$ pro data X, $p \in [0.0, 1.0]$ procentualní zastoupení datasetu pro natrénovaní slabého klasifikátoru, $t \in \mathbb{N}$ požadovaný počet modelů, slabý klasifikátor *WeakLearner*.

Output: Naučený bagging model s t slabými klasifikátory.

```
bagging ← inicializace množiny
for i ∈ {1, ..., t} do
    X_sample, Y_sample ← getRandomSample(X, Y, p)
    bagging ← bagging ∪ WeakLearner(X_sample, Y_sample)
return bagging
```

Algorithm 7 Pseudokód klasifikace nového záznamu pomocí bagging modelu

```
votes ← inicializace množiny
for i ∈ {1, ..., t} do
    votes ← votes ∪ bagging_i(x_unknown)
return mode(votes)                    ▷ Vrátí nejčastější výsledek
```

2.7 Modely pro detekci anomálií

Modely strojového učení lze také využít k hledání společných vlastností a podle toho seskupovat záznamy, ale lze využít této myšlenky a posunout jí ještě dále a hledat anomálie, neboli záznamy, které se svými hodnotami vymykají společným vlastnostem. Obvykle anomálie poukazuje na nový problém, neobvyklé chování či nechtěný jev. Definici anomálie a odlehlé hodnoty lze nalézt v této diplomové práci v sekci 4.2.

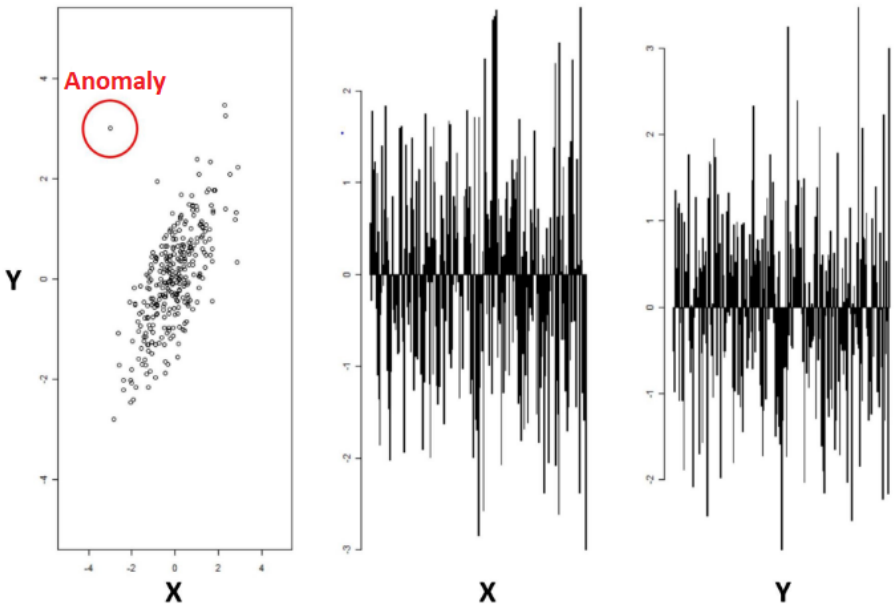
Znalost a schopnost rozlišení anomálií je velmi cenná. V některých případech stačí grafická reprezentace dat pro rozlišení neobvyklého jevu. V jiných případech grafická reprezentace není vhodná a je potřeba použít komplexnější algoritmus.

Například na obrázku 2.8 z [48] lze vidět data, která jsou popsána dvěma příznaky X a Y. V prvním grafu je označená anomálie, avšak existence této anomálie není zjevná při zkoumání samotných příznaků.

Detekce anomálií (anomaly detection) je specifická oblast strojového učení a v případě klasifikace internetového provozu na škodlivý provoz a normální se tomuto odvětví anglicky říká Network intrusion detection systems (NIDS).

Stejně jako klasifikaci internetového provozu lze detekci anomálií rozdělit podle druhu učení na:

- Supervizovaná detekce anomálií: Pro tvorbu modelu využívá trénovací



Obrázek 2.8: Detekce anomálií pro dva příznaky X a Y

dataset a ground truth labely, kde normální chování je přiřazeno do negativní třídy a anomálie do pozitivní třídy. Trénovací dataset by měl obsahovat všechny ukázky anomálií a normálního provozu, které je zapotřebí detekovat.

- Semi-supervizovaná detekce anomálií: Model je vytvořen na základě trénovacího datasetu a ground truth labelů, které označují pouze normální provoz. Díky této vlastnosti jsou semi-supervizované algoritmy více využi telné než supervizované.

- Nesupervizovaná detekce anomálií: Využívá předpokladu, že normální chování se objevuje častěji než anomálie. Pro tvorbu modelu nevyužívá žádných ground truth labelů, ale rovnou klasifikuje vstupní data.

Algoritmy, které se specializují na detekci anomálií se ještě rozdělují na outlier detection a novelty detection. Outlier detection je obvykle nesupervizovaný algoritmus a jeho hlavní charakteristikou je, že vstupní dataset obsahuje jak normální data tak anomálie. Oproti tomu novelty detection je nejčastěji semi-supervizovaný algoritmus a při učení modelu mu jsou předkládána pouze data, která neobsahují anomálie. [49]

Na první pohled se může zdát, že outlier detection a algoritmy pro hledání shluků jsou jedno a to samé, ale není to úplně pravda. Algoritmy pro hledání shluků seskupují podobné záznamy k sobě a klasifikují je podle toho, do kterého

sluku patří, zatímco outlier detection nachází záznamy, které se značně liší od zbytku dat.

Výstupem algoritmů pro detekci anomálií je třída nebo skóre, které značí míru jistoty udávající, zda je záznam anomálie.

Mezi nejznámější modely pro detekci anomálií spadají Local Outlier Factor, One-Class SVM, Isolation Forest a algoritmy založené na umělé neuronové síti (Artificial Neural Network).

2.7.1 Local Outlier Factor

Local Outlier Factor, zkráceně LOF, je nesupervizovaná metoda outlier detection. Jedná se o algoritmus založený na metodě k-nejbližších sousedů. Algoritmus hledá lokální odlehlé body na základě hustoty bodů v sousedství. [50]

Lokální odlehlý bod je nový pojem, který zavedli autoři algoritmu LOF v práci [51]. Rozdíl mezi lokálním odlehlým bodem a odlehlým bodem je jeho výstupní hodnota. Odlehlý bod nabývá hodnoty binárního rázu, tyto dvě hodnoty značí, zda bod je odlehlý či nikoliv. Oproti tomu hodnota lokálního odlehlého bodu je *outlier factor*, což je míra udávající odlehlost bodu. Přesněji *outlier factor* je skóre, které udává místní odchylku hustoty daného záznamu vzhledem k jeho nejbližším sousedům.

2.7.1.1 Výpočet hodnoty LOF

Nechť o, p a q jsou body, které reprezentují záznamy z datasetu. Funkce d vrací vzdálenost mezi dvěma body a $N_k(p)$ značí množinu k-nejbližších sousedů bodu p.

Pro výpočet skóre bodu p, $LOF(p)$, je potřeba definicí funkcí: *k-dist*, *reach-dist*, *lrd*, kde:

- *k-dist* vrací vzdálenost nejvzdálenějšího bodu z k-sousedů:

$$k\text{-}dist(p) = max\{d(p,q)|q \in N_k(p)\}.$$

- *reach-dist* značí vzdálenost dosažitelnosti bodu p vůči bodu o a vrací maximum z dvou prvků: *k-dist* vzdáleného bodu a vzdálenost mezi body:

$$reach\text{-}dist(p,o) = max\{k\text{-}dist(o), d(p,o)\}.$$

- *lrd*, local reachability density, je převrácená hodnota průměru dosažitelných vzdáleností všech k-nejbližších sousedů:

$$lrd(p) = \frac{1}{\frac{\sum_{o \in N_k(p)} reach\text{-}dist(p,o)}{|N_k(p)|}} = \frac{|N_k(p)|}{\sum_{o \in N_k(p)} reach\text{-}dist(p,o)}.$$

Výsledná hodnota *outlier factor* lokálního odlehlého bodu p, $LOF(p)$, je zprůměrovaný podíl ldr jednotlivých nejbližších sousedů vůči lrd bodu p:

$$LOF(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd(o)}{lrd(p)}}{|N_k(p)|}$$

Jsou-li všechny body přibližně stejně dalo od sebe společně s bodem p, pak hodnota $LOF \approx 1$, důkaz je uveden v práci [51] sekce 5.1. Naopak je-li bod p vzdálen od zbytku jeho nejbližších sousedů, pak hodnota $LOF \gg 1$.

2.7.2 Izolační les

Izolační les, isolation forest, je supervizovaná metoda založená na ensemble metodě náhodných lesů.

Většina algoritmů pro detekci anomálií je založena na algoritmu, který se nejdříve naučí normálnímu chování a na základě této naučené vlastnosti klasifikuje další chování, a pokud nové záznamy neodpovídají naučeným skutečnostem jsou označeny jako anomálie. Isolation forest se vydává cestou zcela jinou a to explicitním izolováním anomálií místo profilování normálního chování. [6]

Izolace v případě isolation forest znamená oddělení anomálií od zbytku dat, kterého je docíleno přes náhodné výběry příznaků a následně náhodný výběr hodnoty pro rozdělení intervalu hodnot, kterých vybraný příznak může nabývat. Náhodným dělení je docíleno včasnějšího nalezení anomáli a tím i kratší cesty do uzlu obsahující anomálii, důkaz v práci [52] sekce 2. Pro učení jednotlivých izolačních stromů, pro které navíc platí, že každý uzel stromu má právě dva potomky nebo žádného, není použit vždy celý dataset ale jenom části, vzorky. V práci [52] se uvádí, že lepších výsledků se dosáhne při udržení těchto vzorků menších. Velké množství záznamů může narušit proces izolace, neboli algoritmus ztrácí schopnost přesného izolování anomálií. Proces rozdělení datasetu na vzorky je prováděn náhodným výběrem bez možnosti náhrady. Naučený izolační les vrací skóre pro každý nový záznam.

Základní předpoklad pro použití isolation forest algoritmu je, že anomálie jsou zastoupeny v datasetu menšinou a některé z jejich příznaků se svou hodnotou značně liší od hodnot zbylých normálních záznamů.

2.7.2.1 Izolační les a výpočet skóre

Skóre nového záznamu x při počtu ψ záznamů ve vzorku je vypočteno jako:

$$s(x, \psi) = 2^{-\frac{E(h(x))}{c(\psi)}},$$

- $h(x)$ je délka cesty ve stromě k záznamu x, neboli počet hran vedoucí od kořene k listu obsahující záznam x.
- $E(h(x))$ vrací průměr všech délek cest k záznamu x ze všech izolovaných stromů.

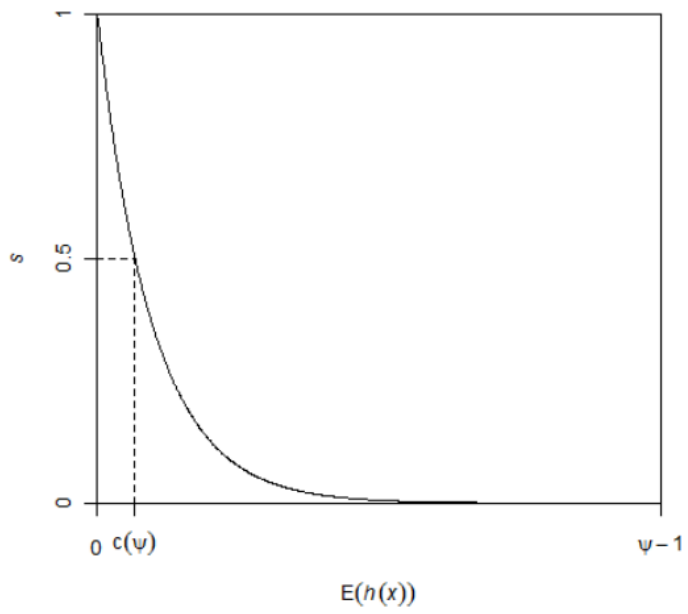
- $c(\psi)$ udává průměrnou délku cesty pro daný počet záznamů ψ ve vzorku při neúspěšném hledání v plném binárním stromu, viz sekce 10.3.3 v práci [53]. Pro výpočet $c(\psi)$ je potřeba definice funkce $H(i)$. $H(i)$ je harmonické číslo a lze ho zapsat přibližně jako $ln(i) + 0.5772156649$ (Eulerova konstanta).

$$c(\psi) = \begin{cases} 2H(\psi - 1) - (2(\psi - 1)/\psi) & \text{pro } \psi > 2, \\ 1 & \text{pro } \psi = 2, \\ 0 & \text{jinak.} \end{cases}$$

Skóre nabývá hodnot z intervalu $\langle 0, 1\rangle$. Obrázek 2.9 znázorňuje vztah mezi skórem s a průměrnou hodnotou $h(x)$.

Je-li skóre záznamu okolo hodnoty 1, pak se jistotou dá říci, že se jedná o anomálii. V případě, kdy skóre nabývá hodnoty nižší než $0,5$ lze naopak bod označit jako normální. Pohybují-li se všechny hodnoty skóre okolo $0,5$, pak algoritmus nenašel žádnou anomálii v datasetu.

$$s(x, \psi) = \begin{cases} 0 & \text{pokud } E(h(x)) = \psi - 1, \\ 0,5 & \text{pokud } E(h(x)) = c(\psi), \\ 1 & \text{pokud } E(h(x)) = 0. \end{cases}$$



Obrázek 2.9: Isolation forest a vztah hodnoty skóre s vzhledem k průměrné délce cesty záznamu $E(h(x))$ z [6]

2.7.2.2 Algoritmus

Během trénovací fáze je vytvořeno dané množství izolovaných stromů na požadovaně velkých vzorcích datasetu. Velikost vzorků ψ kontroluje velikost trénovacích dat pro stromy. Kromě lepší schopnosti izolace anomálií přináší malé vzorky i další výhody mezi které spadá nižší paměťová náročnost, rychlejší výpočetní čas, schopnost vypořádání se s velkými vstupními daty a vysokou dimenzionalitou dat a v neposlední řadě předchází problému mylné klasifikace normálního záznamu, nazývané přehazování [6].

Algorithm 8 Pseudokód algoritmu isolation forest *iForest(X, t, ψ)*

Input: Množina vektorů trénovacích dat $X = \{x_1, \ldots, x_m\}$, požadovaný počet izolovaných stromů t, velikost vzorku ψ vstupních dat do stromu.

Output: Množina izolovaných stromů *iTree* o velikosti t.

```
1: forest ← inicializace množiny
2: for i ∈ {1, ..., t} do
3:     X_sample ← sample(X, ψ)              ▷ Vytvoření vzorku o velikosti ψ
4:     forest ← forest ∪ iTree(X_sample)
5: return forest
```

Algorithm 9 Pseudokód algoritmu isolation tree *iTree(X_{sample})*

Input: Množina vektorů trénovacích dat $X_{sample} = \{x_1, \ldots, x_\psi\}$.

Output: Naučený model izolačního stromu $iTree$.

```
1: if X_sample nelze rozdělit then
2:     return exNode{Size ← |X_sample|}                    ▷ List stromu
3: else
4:     Nechť Q je množina příznaků vzorku datasetu X_sample
5:     q ← RandomChooser(Q)                          ▷ Náhodný výběr z Q
6:     Rozdělující bod p je náhodně vybrán z intervalu [min(q), max(q)]
7:     X_l ← filter(X_sample, q < p)
8:     X_r ← filter(X_sample, q ≥ p)
9:     return inNode{Left ← iTree(X_l),
                     Right ← iTree(X_r),
                     SplitAtt ← q,
                     SplitValue ← p}                       ▷ Uzel stromu
```

2.7.3 Metody založené na umělé neuronové síti

Umělá neuronová síť je výpočetní model, který je volně imspirován chováním lidského nervového systému. Stavebním prvkem neuronové sítě, jak lidské tak

umělé, je neuron a jeho chování je podobné, bere několik signálů s určitou váhou a vrací výstup na základě rozhodnutí. Rozhodnutí se v případě strojového učení nazývá aktivační funkce. Obecný vzorec pro umělý neuron, nazývaný též perceptron, je následující:

$$Y = S\left(\sum_{i=1}^{d}(w_i \cdot x_i - \Theta)\right), \text{ kde}$$

- d je velikost vstupu,
- x_i je i-tý vstup neuronu,
- w_i je i-tá váha, která modifikuje intenzitu vstupu,
- $\sum_{i=1}^{d}(w_i \cdot x_i)$ je potenciál neuronu,
- Θ je práh aktivační funkce, též nazývaný bias,
- $S(x)$ je aktivační funkce,
- Y je výstup neuronu.

Neuron indikuje signál ve svém výstupu ve formě aktivační funkce pouze v případě, přesahuje-li potenciál neuronu hodnotu biasu.

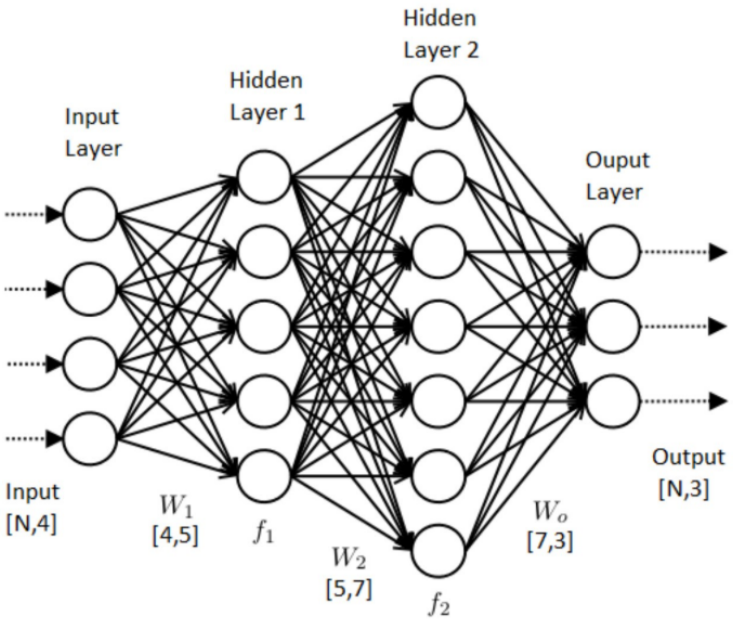
Neuronová síť představuje topologické uspořádání neuronů, ty spolu komunikují na základě orientovaných ohodnocených spojů. Charakterizace neuronové sítě se provádí na základě typu neuronů, jejich topologickým uspořádáním a strategií adaptace při fázi učení.

Nejzákladnější typ je dopředná neuronová síť. Ta je charakterizována jednosměrným průchodem signálu ze vstupu k výstupu, přičemž jednotlivé neurony jsou uspořádány do vrstev, viz obrázek 2.10. Pro jednotlivé vrstvy platí, že neurony z jedné vrstvy nejsou mezi sebou propojeny, ale jsou propojeny s neurony sousedních vrstev, zpravidla se používá propojení úplné. První vrstva neuronové sítě se nazývá vstupní vrstva a počet neuronů v ní udává i velikost vstupu, poslední vrstva se nazývá výstupní, její počet neuronů udává velikost výstupu. Ostatní mezilehlé vrstvy se nazývají skrytými vrstvami.

Neurové sítě určené pro detekci anomálií jsou naučeny nejdříve na vzorku dat spadajících do normální, negativní třídy. Po předložení testovacích dat, model vrací výsledek na základě toho, zda byl záznam přijat a odpovídá normálnímu chování a nebo zda byl odmítnut.

2.7.3.1 Plně zapojený autoencoder

Plně zapojený autoencoder je nesupervizovaná metoda pro detekci anomálií. Autoencoder je specifický typ dopředné neuronové sítě, kde velikost d výstupu X' odpovídá velikosti vstupu X, navíc výstup $x'_i \in X'$ je rekonstrukce původního vstupu $x_i \in X$ pro všechny $i \in \{1, \ldots, d\}$.



Obrázek 2.10: Uspořádání neuronů do vrstev v dopředné neuronové síti [7]

Dopředná neuronová síť je tvořena třemi komponentami: kodérem, kódem a dekodérem. Kodér zkomprimuje vstup do méně dimenzionálního prostoru a tím vytváří ze vstupu vektor, nazývaný kód. Úloha dekodéru je z kódu zreprodukovat původní vstup, viz obrázek 2.11. Na obrázku si lze povšimnout, že architektura autoencoderu je symetrická, nesymetrické arhitektury bývají spíše výjimkou [54].

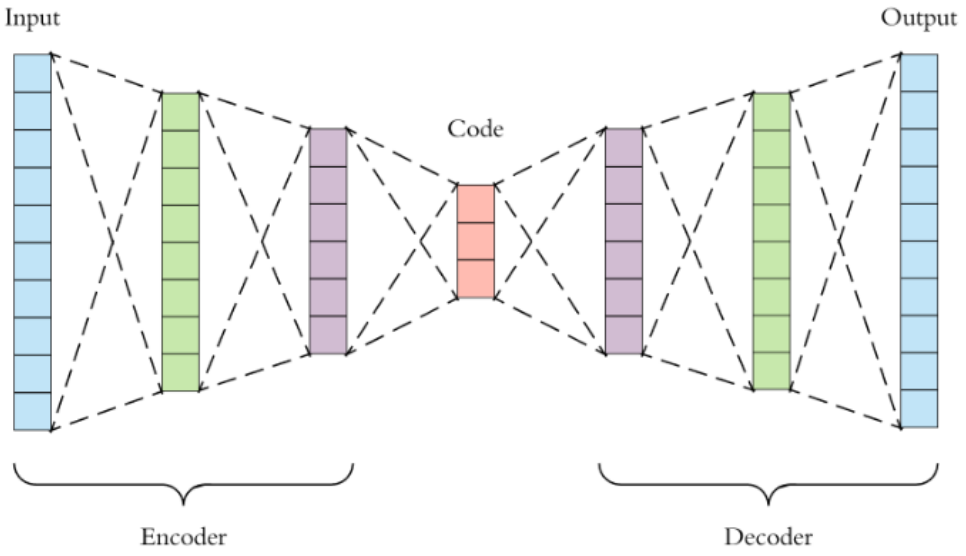
Důležitými částmi procesu je výběr vhodného tvaru kodéru, dekodéru a volba ztrátové funkce, která vrací rekonstrukční chybu.

Úkolem neuronové sítě je minimalizace rekonstrukční chyby $\sum_{i=1}^{d}(x_i - x_i')^2$, samotná hodnota rekonstrukční chyby je hodnota skóre anomálie.

2.8 Vyhodnocení výsledků klasifikačních algoritmů

Interpretace a vyhodnocení výsledků naučeného modelu jsou důležitými kroky na konci celého procesu strojového učení.

K číselnému vyhodnocení naučeného modelu se používají hodnotící metriky, jež porovnávají hodnoty ground truth labelů vůči získaným predikovaným hodnotám. Hodnotící metriky lze využít i pro validaci a komplexnější popis a analýzu chování naučeného modelu a to na základě hodnocení více testů pro jeden algoritmus strojového učení, do této skupiny hodnocení spadají například učící křivky.



Obrázek 2.11: Architektura autoencoderu [8]

2.8.1 Hodnotící metriky algoritmů

Jména hodnotících metrik v této části jsou uváděna převážně v anglickém jazyce, jelikož jejich český ekvivalent se používá pouze zřídka.

Hodnotící metriky jsou pro lepší pochopení definovány vzorci pro binární klasifikaci. Vzorce lze upravit na vícetřídní klasifikaci pomocí přístupu jeden-vs-zbytek (one-vs-rest), neboli pro každou jednotlivou třídu je vypočtena daná metrika vůči zbylým třídám, všechny výsledky se sečtou a podělí počtem tříd [55]. Vícetřídní klasifikaci lze počítat i s váženými průměry jednotlivých tříd podle počtu záznamů [56].

Většina metrik se odvíjí od počtu správně a nesprávně klasifikovaných pozitivních a negativních záznamů, jejich definice a vztah popsán níže.

True positive (TP): Počet správně klasifikovaných pozitivních záznamů.

True negatives (TN): Počet správně klasifikovaných negativních záznamů.

False positives (FP): Počet nesprávně klasifikovaných záznamů do pozitivní třídy.

False negatives (FN): Počet nesprávně klasifikovaných záznamů do negativní třídy.

Hlavním cílem vyhodnocení modelů je určit v kolika případech se výsledek klasifikátoru shoduje s ground truth labely a v kolika případech došlo k chybné klasifikaci. K zachycení těchto údajů se vyžívá matice záměn 2.1, tzv. Confusion matrix, a čtyř výše definovaných pojmů. Sloupce matice odpovídají skutečné hodnotě záznamu a řádky odpovídají predikované hodnotě klasifikátorem. Samotné buňky pak obsahují četnost toho, kolikrát došlo na klasifikovaných datech k dané shodě skutečné a predikované hodnoty. Případy

na diagonále jsou klasifikovány správně, mimo diagonálu se jedná o chyby či záměny klasifikace.

Tabulka 2.1: Matice záměn pro binární klasifikaci

		Správná třída	
		Pozitivní	Negativní
Predikovaná třída	Pozitivní	TP	FP
	Negativní	FN	TN

Metriky uvedeny dále využívají výše definovaných pojmů.

Recall, true positive rate (TPR), sensitivity: Metrika udávající poměr správně klasifikovaných pozitivních záznamů k celkovému počtu záznamů, jež správně spadají do pozitivní třídy:

$$\frac{TP}{TP + FN}.$$

Precision: Jedná se o metriku udávající jak správně dokáže model predikovat kladnou třídu záznamů:

$$\frac{TP}{TP + FP}.$$

False positive rate (FPR): Je udáván jako poměr $FP/(FP + TN)$.

Přesnost (Accuracy): Relativní počet správných rozhodnutí klasifikátoru je udáván jako poměr mezi množstvím správně klasifikovaných záznamů a celkovým množstvím všech klasifikovaných záznamů:

$$\frac{TP + TN}{TP + TN + FP + FN}.$$

Hodnota přesnosti by se měla pohybovat v intervalu $(Acc_{def}, 1]$, kde Acc_{def} je dosažená přesnost klasifikátoru, který přiřadí všechny záznamy k nejvíce zastoupené třídě.

F1 skóre: Harmonický průměr precision a recall. Uvádí se, že je vhodnější pro klasifikaci nevyváženého datasetu [57], v případě vícetřídní klasifikace pak pokud se použije průměr výsledků pro jednotlivé třídy [58], též nazývané jako makroprůměr. F1 skóre je nejčastěji používáno pokud se primárně sledují hodnoty precision a recall a požaduje se, aby obě tyto hodnoty byly co největší možné.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

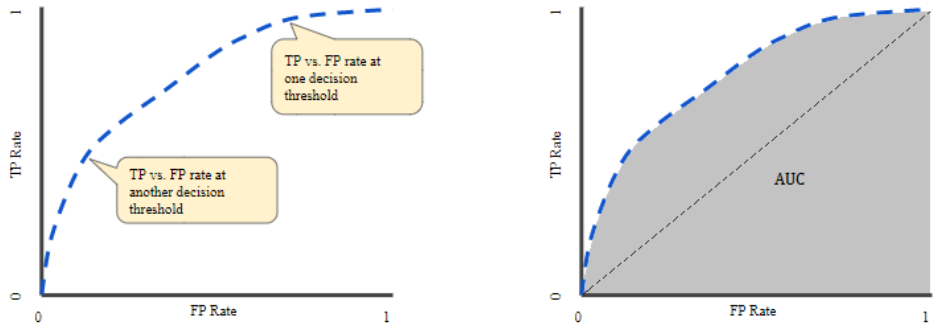
Vzoreček pro vícetřídní klasifikaci při použití makroprůměru pro G tříd je následující:

$$P_{macro} = \frac{1}{|G|}\sum_{i=1}^{|G|}\frac{TP_i}{TP_i + FP_i} = \frac{\sum_{i=1}^{|G|} P_i}{|G|},$$

$$R_{macro} = \frac{1}{|G|}\sum_{i=1}^{|G|}\frac{TP_i}{TP_i + FN_i} = \frac{\sum_{i=1}^{|G|} R_i}{|G|},$$

$$F1_{macro} = 2 \cdot \frac{P_{macro} \cdot R_{macro}}{P_{macro} + R_{macro}}.$$

Receiver Operating Characteristic curve (ROC): ROC je pravděpodobnostní křivka a počítá se pro každou klasifikovanou třídu zvlášť. Je to metrika, která je znázorněna grafem, viz obrázek 2.12, osa x vynáší hodnoty FPR a osa y TPR, neboli recall. Znázorněná křivka reprezentuje TPR a FPR pro jinou rozhodovací hranici. Pokud klasifikátor vrací pravděpodobnosti tříd místo příslušností, lze definovat různé rozhodovací hranice (např. záznamy s pravděpodobností větší než 70 % jsou klasifikovány jako pozitivní, ostatní jako negativní).



(a) Křivka ROC

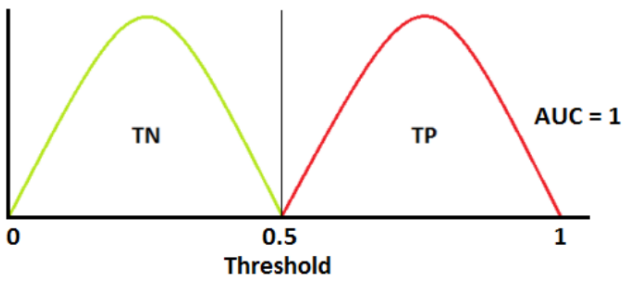
(b) Plocha pod ROC zvaná AUC

Obrázek 2.12: Zobrazení ROC a AUC metody klasifikace [9]

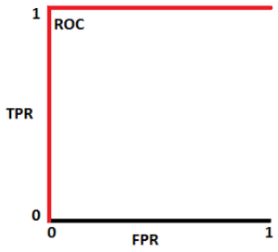
Area under the ROC (AUROC či AUC ROC): AUROC je hodnotící metrika, která zohledňuje všechny možné prahové hodnoty klasifikace a představuje stupeň či míru možné separace tříd. Přesněji popisuje schopnost modelu rozlišovat jednotlivé třídy od sebe [10]. Čím větší číslo, tím je model lepší. Hodnota AUROC se pohybuje od 0 do 1.

Níže jsou uvedeny příklady vyobrazující distribuce dvou tříd pro určité hodnoty AUC (Area Under the Curve) a příslušející ROC křivky. Červená distribuční křivka znázorňuje pozitivní třídu (například, že se jedná o internetový útok) a zelená znázorňuje negativní třídu (nejedná se o internetový útok). Obě třídy jsou popsané vždy stejnou množinou příznaků.

Graf 2.13 znázorňuje perfektní možné rozdělení dvou tříd. Avšak reálný svět není až tak jednoduchý a výjimky potvrzují pravidlo. Proto se lze obecněji



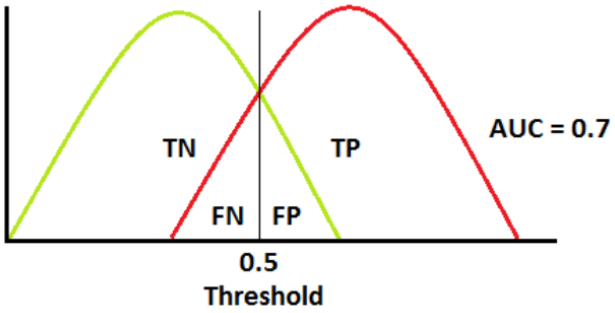
(a) Distribuce dvou tříd s $AUC = 1$



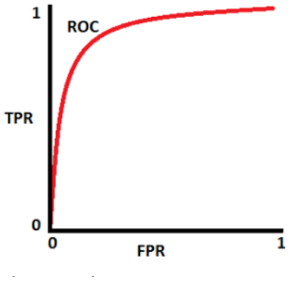
(b) ROC křivka při $AUC = 1$

Obrázek 2.13: Grafické zobrazení distribuce dvou tříd s AUC hodnotou rovno jedné a k tomu příslušející ROC křivka [10]

setkat s dějem následujícím zobrazeným v grafu 2.14a, kde dochází k překryvu distribučních křivek a tudíž nutně dochází i k mylné klasifikaci záznamů.



(a) Distribuce dvou tříd s $AUC = 0,7$



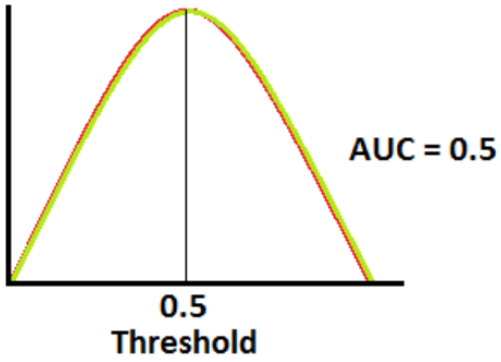
(b) ROC křivka při $AUC = 0,7$

Obrázek 2.14: Grafické zobrazení distribuce dvou tříd s AUC hodnotou rovno $0,7$ a k tomu příslušející ROC křivka [10].

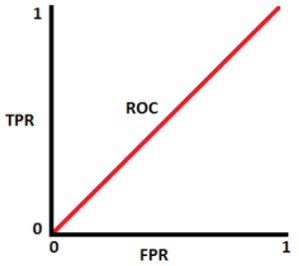
Je-li hodnota AUC rovna $0,5$, pak nastává situace, že výsledný klasifikační model nemá žádnou diskriminační sílu a nedokáže rozhodnout mezi dvěma skupinami, viz graf 2.15a. Jedná se o nejhorší možnou situaci pro rozhodovací model.

Precision-recall křivka Precision-recall křivka (PRC), se využívá pro zobrazení všech prahových hodnot mezi precision a recall. Obecně se udává, že je to vhodnější metoda než ROC pro měření úspěchu predikce pro velmi nevyvážené datasety [59], kdy kladná třída je zastoupena menšinou dat.

Z tabulky 2.2 je patrné, že precision-recall křivka nebere v potaz vůbec TN záznamy. Jinak řečeno, výpočet se striktně zaměřuje na TP záznamy a jejich mylnou klasifikaci. Proto je-li dataset nevyvážený a obsahuje více negativních záznamů, které jsou správně ohodnoceny do TN, pak jejich existence nemá žádný vliv na výslednou hodnotu precisiol-recall [60, 61].



(a) Distribuce dvou tříd s $AUC = 0,5$



(b) ROC křivka při $AUC = 0,5$

Obrázek 2.15: Grafické zobrazení distribuce dvou tříd s AUC hodnotou rovno $0,5$ a k tomu příslušející ROC křivka [10]

Tabulka 2.2: Rozdíl mezi ROC a precision-recall křivkou

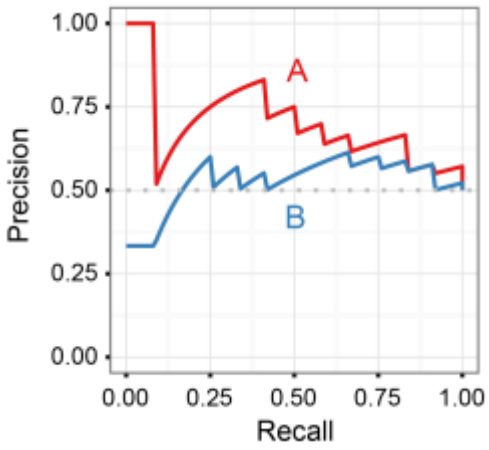
Křivka	Osa X		Osa Y	
	Metrika	Vzoreček	Metrika	Vzoreček
PRC	Recall	TP / (TP + FN)	Precision	TP / (TP + FP)
ROC	FPR	FP / (FP + TN)	Recall	TP / (TP + FN)

Obrázek 2.16 zobrazuje vzhled křivky, narozdíl od ROC se ideální maximum PRC nachází v pravém horním rohu grafu, kdežto u ROC v levém horním rohu. Stejně jako u ROC i u PRC lze z křivky vypočítat AUC, označováno jako AUPRC či AUC PRC.

2.8.2 Křížová validace

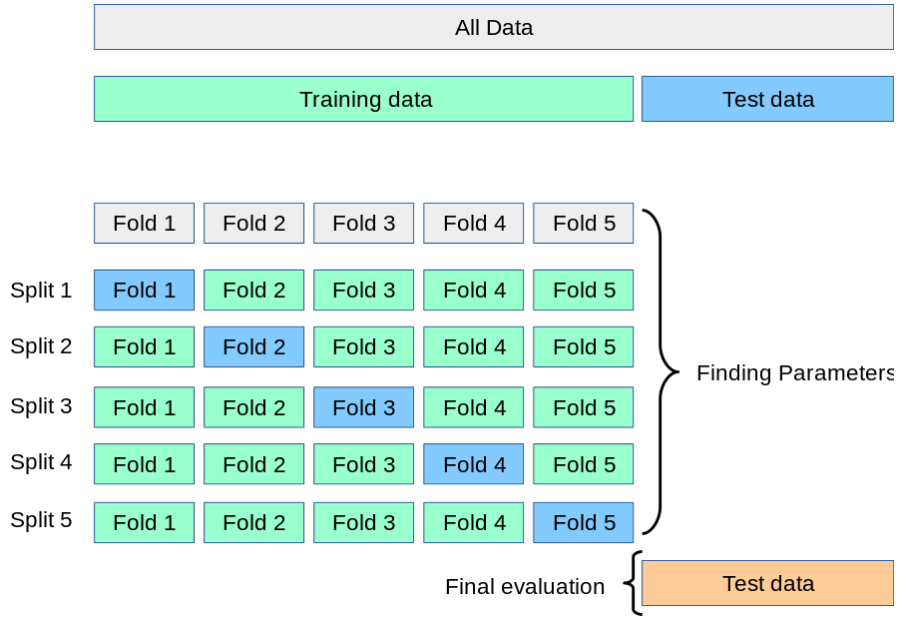
Křížová validace (cross-validation, CV) je metoda ohodnocení schopnosti modelu klasifikovat nová data, která nebyla použita během trénovací fáze. Taktéž slouží k odhalení overfitting problému a využívá se i jako metoda hodnocení modelu při hledání hyperparametrů algoritmů. Hyperparametry jsou parametry algoritmů strojového učení, které musejí být definovány ještě před procesem učení.

Původní trénovací dataset je rozdělen na předem daný počet k disjunktních podmnožin. Křížová validace postupně vybírá jednu z podmnožin jako testovací dataset a zbylé podmnožiny tvoří společně nový trénovací dataset. Následně pomocí nového trénovacího datasetu vzniká naučený model a jeho úspěšnost klasifikace je ohodnocena vybranou metrikou na vybraném testovacím datasetu. Celý tento postup se opakuje k-krát, kde je každá podmnožina vyzkoušena právě jednou jako testovací dataset a zbylé jako nový trénovací da-



Obrázek 2.16: Ukázka dvou precision-recall křivek [11]

taset, viz obrázek 2.17. Nakonec jsou výsledky zprůměrovány, a proto se této metodě také říká průměrná přesnost. Kromě průměrné přesnosti lze získat i směrodatnou odchylku úspěšnosti.



Obrázek 2.17: Ukázka rozdělení trénovacího datasetu při použití křížové validace [12]

2.8.3 Učící křivky

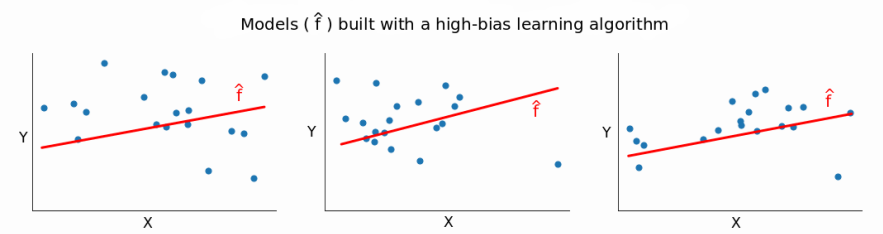
Učící křivky poskytují matematické a grafické znázornění výkonu supervizovaného algoritmu strojového učení. Na základě grafického znázornění učících křivek, kde osa x obvykle reprezentuje velikost trénovacího datasetu a osa y hodnotící metriku či chybovost predikce, lze diagnostikovat chování algoritmu strojového učení na zvoleném vstupním datasetu, zda vhodně a dostatečné popisuje data, netíhne k overfitting nebo underfitting problému. Učící křivky také dokážou poskytnout informaci, jakým směrem se lze ubírat pro získání lepšího modelu a jeho dosažených výsledků. [62]

Prvně je vhodné vysvětlit pojmy bias a variance, u kterých je snaha minimalizovat jejich hodnoty k dosažení optimálně komplexního modelu.

Bias: Vypovídá o schopnosti algoritmu strojového učení nalézt vztah mezi příznaky a ground truth labelem. Bias je indikován úspěšností či chybovostí modelu.

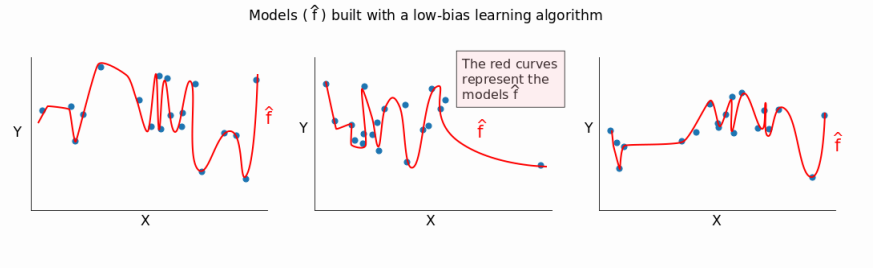
Vysoký bias je důsledkem neschopnosti modelu strojového učení nalézt více vypovídající vztah mezi příznaky a ground truth labely a proto vrací nízkou hodnotu přesnosti či vysokou hodnotu chybovosti. Algoritmus není schopen nalézt žádné větší souvislosti mezi příznaky a výstupním labelem, a tak výsledný model může skončit s underfitting problémem, viz obrázek 2.18. Obecně algoritmy mají vysoký bias, aby byly schopné se rychle učit, zato jsou ale méně flexibilní vůči vstupním datům [63, 64].

Příčinou nízkého biasu je algoritmus, která se snaží popsat všechny data co nejlépe, to ale způsobuje, že odlehlé hodnoty a šum mají velký vliv na výsledný model, viz obrázek 2.18. Výsledkem může být model, který je sužován overfitting problémem. Takový model perfektně popisuje trénovací dataset avšak pro testovací dataset vrací nízké skóre přesnosti či vysokou hodnotu chybovosti.

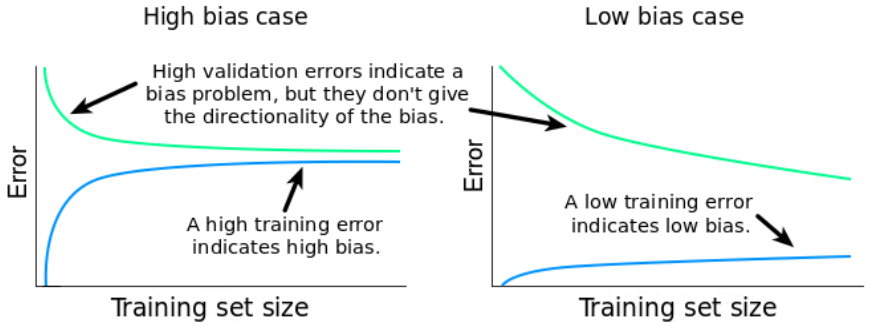


Obrázek 2.18: Vzniklé modely vytvořené algoritmem strojového učení s vysokým biasem na různých vstupních datasetech charakterující stejný problém [13]

V obrázku 2.20 jsou znázorněné učící křivky pro vysoký a nízký bias.



Obrázek 2.19: Vzniklé modely vytvořené algoritmem strojového učení s nízkým biasem na různých vstupních datasetech charakterující stejný problém [13]



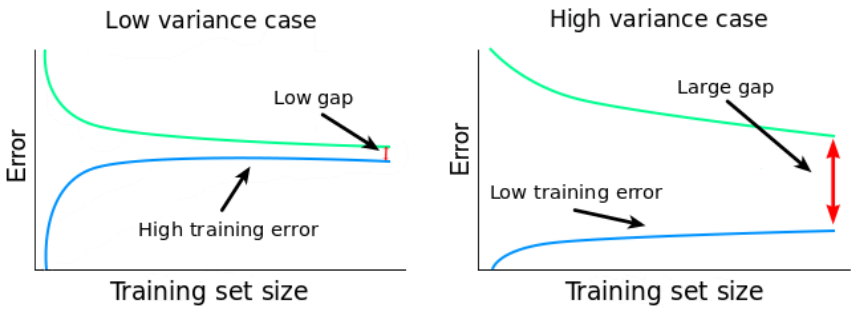
Obrázek 2.20: Učící křivky znázorňující algoritmus strojového učení s vysokým a nízkým biasem [13]

Variance: Udává závislost modelu na vstupních trénovacích datech a jak hodně je model a jeho predikce ovlivněna změnou či jinou volbou trénovacích dat pro stejný pozorovaný objekt. Variace je založena na předpokladu, že predikovaná hodnota naučeného modelu by měla být stejná, i když byly použity různé trénovací datasety, které ale popisují stejný objekt. Čím více je algoritmus strojového učení ovlivněn trénovacími daty tím větší je hodnota variance.

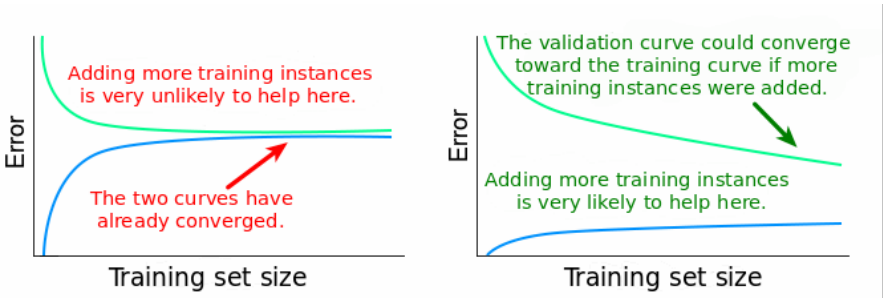
Nízká variance doprovází vysoký bias, jak je vidět na obrázku 2.18, vzniklý model se zásadně nemění i při použití různých trénovacích datasetů. Naopak vysoké variance si lze všimnou na obrázku 2.19, pro každý vstupní dataset vznikl kompletně jiný model, který je navíc doprovázen nízkým biasem.

Zda model trpí vysokou nebo nízkou variací lze odhadnout na základě velikosti mezery mezi trénovací a testovací křivkou, viz obrázek 2.21. Malá mezera značí nízkou varianci a velká mezera značí vysokou varianci.

Levý podobrázek z 2.22 odpovídá modelu s vysokým biasem a nízkou variancí. Křivka testovacích dat se velmi rychle přibližuje ke křivce trénovacích dat s vysokou hodnotu chybovosti. Z toho lze vyvodnit, že přidání nových trénovacích dat pravděpodobně nebude mít žádný zásadní vliv pro získání lepšího modelu. V případě vysokého biasu a nízké variance spíše pomůže zaměřit se na přidání nových příznaků, vyzkoušení jiných parametrů algo-



Obrázek 2.21: Učící křivky znázorňující algoritmus strojového učení s vysokým a nízkým biasem [13]

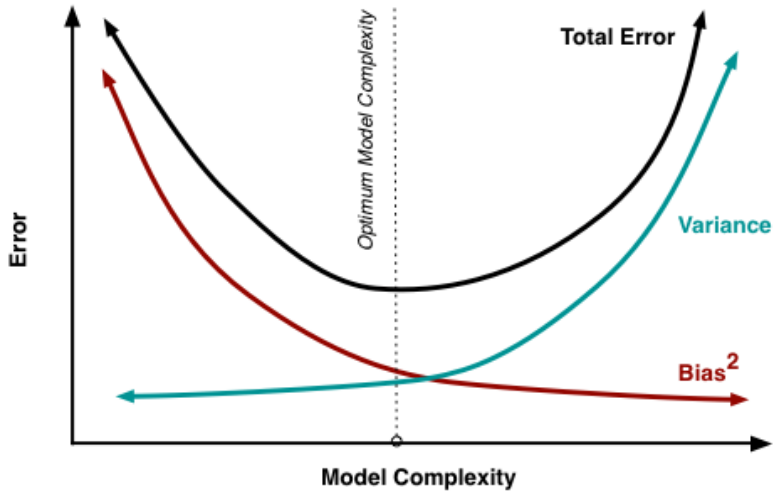


Obrázek 2.22: Dva krajní případy učících křivek: model s vysokým biasem a nízkou variancí a model s nízkým biasem a vysokou variancí [13]

ritmu strojového učení pro vytvoření komplexnějšího modelu, a nebo celkově vyzkoušení jiného algoritmu strojového učení.

Algoritmus strojového učení s nízkým biasem a vysokou variací vrací pro trénovací data sice nízkou hodnotu chyby, ale naopak pro trénovací data poměrně vysokou, viz pravý podobrázek z 2.22. Z toho lze usoudit, že přidání více trénovacích dat by mohlo pomoci ke konvergenci obou křivek blíže k sobě. Další vhodný způsob je například volba a výběr příznaků, volba méně komplexního modelu nebo použití techniky regularizace algoritmu [65].

Úkolem je dosáhnout co nejmenších hodnot biasu a variance pro zvolený algoritmus strojového učení a popisovaný objekt. Obrázek 2.23 popisuje křivky variance a druhé mocniny biasu společně s křivkou udávající chybovost modelu, pro které svislá čerchovaná čára znázorňuje ideální kompromis, neboli optimální komplexitu modelu. Komplexita modelu může být charakterizována několika ukazately, mezi které nejčastěji spadá počet příznaků a výpočetní složitost modelu.



Obrázek 2.23: Nejvhodnější komplexita modelu v závislosti na hodnotě biasu a variance [13]

2.9 Pojmy používané při klasifikaci internetového provozu

Real-time klasifikace: Je klasifikace odchyceného provozu v reálném čase s minimálním zpožděním.

Offline klasifikace: Je proces, při kterém jsou naučenému modelu předklá daná data z již předem odchyceného internetového provozu. Tímto způsobem je možné vycházet i z příznaků získaných z celého flow bez žádného časového omezení, omezení je dáno pouze délkou zachyceného internetového provozu.

Offline klasifikátor: Model natrénovaný na souborech obsahující předem odchycený internetový provoz.

Granularita klasifikace: Udává, do jakých a kolika skupin algoritmus umí klasifikovat vstupní data. Nejčastěji se rozlišuje mezi dvěma stupni tzv. hrubozrnných algoritmů (coarse-grained) a jemnozrnných algoritmů (fine-grained). Coarse-grained algoritmy dokáží klasifikovat pouze velké rodiny protokolů, jako například komunikace co patří do P2P a komunikace co nepatří do P2P, nebo HTTP a Streaming. Často se jedná o klasifikaci na úrovni typů komunikace nebo vlastností. Fine-grained klasifikace je klasifikace, která dokáže klasifikovat záznamy na úrovni aplikace, která je použila, či zařadit komunikaci do specifičtějších typů komunikace.

Background internetový provoz: Nebo též pouze background provoz je pojem, který souhrnně označuje internetovou komunikaci, která neobsahuje žádné internetové útoky a která probíhá souběžně s komunikací obsahující internetový útok. Jedná se o neškodnou komunikaci.

2.10 Přístupy ke klasifikaci síťové komunikace

Klasifikace síťového provozu je proces asociování síťového provozu s typem komunikace či jménem aplikace, která ho generuje, nebo shlukování provozu na základě stejných či obdobných charakteristik komunikace. V této podkapitole jsou představeny různé přístupy pro klasifikaci síťové komunikace, od dnes již neefektivní klasifikace založené na číslech portů, klasifikace založené na hluboké inspekci paketů a stochastické inspekci paketů, přes klasifikaci založené na chování aplikací a v neposlední řadě klasifikaci založené na statistických údajích o flow a strojovém učení.

2.10.1 Klasifikace založená na číslech portů

Klasifikace založená na číslech portů, port-based, je jedna z nejstarších klasifikací, nejrychlejších a nejjednodušších. Na začátku Internetu měly aplikace přesně definovaná čísla portů, která musely používat a na základě těchto čísel probíhala klasifikace. Čísla portů byla přidělena organizací IANA (Internet Assigned Numbers Authority) [66].

Klasifikátor založen na číslech portů získává potřebnou hodnotu portu z paketové hlavičky, následně klasifikuje síťovou komunikaci aplikací, která náleží získanému číslu portu podle seznamu od IANA.

Éra úspěchu port-based klasifikace netrvala dlouho. S nárůstem aplikací a služeb přestalo být možné přiřadit každé aplikaci specifické číslo portu a proto aplikace už nejsou dále nuceny používat definovaná čísla portů. Například P2P spojení vybírá čísla portů na začátku komunikace nejčastěji náhodně. Jiné aplikace mohou činit záměrné schovávání své komunikace pod číslem portu, které bylo definované pro jinou aplikaci. V neposlední řadě tento přístup klasifikace není schopen klasifikovat žádnou jinou aplikaci, než která má definované číslo port v seznamu od IANA. V dnešní době klasifikace založená na číslech portů dosahuje velkého množství nesprávně klasifikovaných záznamů.

Nesprávná klasifikace nastává nejen u již zmíněného P2P spojení, ale také třeba u pasivního módu protokolu pro přenášení souborů, passive File Transfer Protocol (passive FTP), síťových tunelů nebo u použití protokolu pro překlad portů síťových adres, Network Address Port Translation (NAPT). Z práce [67] je patrné, že klasifikace na základě portu není vůbec schopná klasifikovat až 30 % vstupních dat. Ze zbylých 70% klasifikovala úspěšně pouze webovou komunikaci a to z 30 % a BULK komunikaci z 45 %. Avšak, co je z práce nejvíce patrné je, že klasifikace na základě portů si neumí poradit s útoky, což je velice závažný nedostatek. Z jiné práce [68] lze vyčíst, že aplikací, co používají

nestandardní port, značně přibývá a jejich počet se zvyšuje. Od podzimu roku 2003 do jara 2005 se počet neznámé komunikace v síti pro klasifikační algoritmus založený na číslech portů zvýšil z 10%–30% na 30 %–70 %.

2.10.1.1 Vyhodnocení

- Výhody:
  - Velice rychlá klasifikace s malým nárokem na paměť.
  - Pracuje pouze s hlavičkami paketů, tudíž nevyžaduje uživatelská data.
  - Lze použít na šifrovaná uživatelská data.
- Nevýhody:
  - Správně klasifikuje pouze aplikaci, která má od IANA definované číslo portu a to striktně používá.
  - Ostatní případy klasifikuje jako neznámé nebo nesprávně.
  - Neumí vůbec detekovat a klasifikovat škodlivou komunikaci.

2.10.2 Klasifikace založená na uživatelských datech

Klasifikace založená na uživatelských datech, payload-based, předchází problému, který zužuje klasifikaci založenou na číslech portů. Základní vlastností klasifikace je hledání specifických vzorů aplikací v uživatelských datech paketu. Existují dva různé přístupy hledání vzorů:

- Hluboká inspekce paketu (DPI),
- Stochastická inspekce paketu (SPI).

Hluboká inspekce paketu prohledává obsah uživatelských dat, ve kterém hledá předem definované deterministické vzory, nebo používá regulární výrazy vůči uživatelským datům.

Stochastická inspekce paketu (SPI) se především zaměřuje na syntaxi paketu a využívá statistických vlastností uživatelských dat. Přesněji řečeno statisticky charakterizuje viděné frekvence dat, například podobně jako u chí-kvadrát testu [69].

I tyto metody mají problémy se specifickými komunikacemi, například klasifikace P2P a multimediálních aplikací selhává kvůli faktu, že tyto aplikace mají proměnlivé vzory obsahu uživatelských dat [70]. Nevýhoda přístupu je neustálé udržování aktuálních klasifikačních pravidel, ta musejí být upravena na základě jakékoliv změny v protokolu aplikace. Další problém je právního rázu, u payload-based přístupu může docházet podle lokálního práva k porušování soukromí a proto se metoda stává nepoužitelná. V některých státech

dochází až k odepření přístupu k uživatelským datům obsažených v těle paketu [71]. A v neposlední řadě tyto metody ztroskotávají na šifrovaných datech.

Na druhou stranu u dešifrovaných paketů, pro něž je dodán i set obsahující vypovídající vzory o uživatelských datech pro požadované aplikace, je tento přístup velice úspěšný.

2.10.2.1 Vyhodnocení

- Výhody:
  - Hledání podle vzorů.
  - Vysoká úspěšnost klasifikace při kvalitním datasetu vzorů.
- Nevýhody:
  - Nelze použít na šifrovaná uživatelská data (pokud není žádná možnost dešifrace) nebo uživatelská data chráněná zákonem.
  - Vzory musejí být stále upravovány na základě změn v protokolech aplikací.

2.10.3 Klasifikace založená na chování

Klasifikace založená na chování, host-behavior-based, využívá informací o „sociální interakci“ aplikace. K popisu interakce se využívá heuristických postupů. Očekává se, že každá aplikace má své specifické chování a interakci s ostatními aplikacemi [72]. Klasifikátory jsou schopné informace o interakci získat i ze šifrovaných uživatelských dat, což je jedna z výhod oproti payload-based přístupům.

Mezi host-behavior klasifikátory patří například program BLINC [73], který interakci profiluje na transportní vrstvě bez znalosti použitých portů a obsahu uživatelských dat. Interakci se snaží zachytit na třech různých úrovních: na sociální úrovni, síťové úrovni a aplikační úrovni. Na základě zkoumání [74] bylo zjištěno, že BLINC nejlépe pracuje pokud odchytává komunikaci na hranici domácí sítě. Nejlepších výsledků pak dosahuje, když je umístěn do uzlu, kterým prochází veškerá komunikace a tak BLINC může odchytit obousměrnou komunikaci.

2.10.3.1 Vyhodnocení

- Výhody:
  - Schopnost klasifikovat šifrovanou komunikaci.
- Nevýhody:
  - Nevhodné na páteřní sítě, kde nelze odchytit informace o celkové interakci.

  - Ke správné klasifikaci potřebuje získat data ideálně z obousměrné komunikace.

2.10.4 Klasifikace založená na strojovém učení z údajů o flow

Strojové učení využívající statistické údaje o flow, flow features-based, poslední dobou také nazývané Deep Flow Inspection (DFI), je využíváno jako další slibná alternativa pro síťovou klasifikaci. Klasifikace je založena na obdobném přístupu jako host-behavior-based klasifikace. Očekává, že každá aplikace má určité statistické vlastnosti, které ji charakterizují jako například velikosti paketů, délka trvání komunikace, použitý protokol a podobně. V práci [75] lze nalézt popis velkého množství statistických informací o flow, které lze použít pro klasifikaci záznamu.

Metoda nepřistupuje k uživatelským datům, je vhodná na šifrovanou komunikaci a při dostatečném množství získaných dat popisujících flow je schopna se vypořádat s dynamicky se měnícími čísly portů. Jejich přesnost sice trochu pokulhává za přesností DPI klasifikátorů, ale stále se nacházejí nové způsoby, jak přesnost zvýšit.

Další výhoda oproti DPI klasifikátoru je možná klasifikace nové aplikace. Podle použité varianty učení se odvíjí vlastnosti klasifikátoru, a to i zda dokáže klasifikovat novou komunikaci, kterou ještě neviděl a nebo se musí naučit novým vzorům.

2.10.4.1 Vyhodnocení

- Výhody:
  - Schopnost klasifikovat šifrovanou komunikaci.
  - Možnost klasifikace nové aplikace bez předešlého vzoru.
- Nevýhody:
  - Potřebuje dostatek dat, ideálně vybalancovaných, pro naučení klasifikačních vlastností či nalezení odlišností mezi daty.

Tabulka 2.3 obsahuje přehled přístupů klasifikace internetového provozu a jejich vlastnosti, jako s jakými údaji se pracuje, jak včasná je klasifikace a výpočetní složitost algoritmu.

2.11 Výzvy na poli klasifikace internetového provozu a detekce anomálií

Vylepšených algoritmů pro klasifikaci internetového provozu je celá řada a výzkumníci hledají ten nejlepší z nich. Na druhou stranu tyto algoritmy čelí

Tabulka 2.3: Souhrnná tabulka přístupů klasifikace internetového provozu a jejich vlastností [21]

Přístup	**Pro klasifikaci využívá**	**Klasifikace**	**Výpočetní složitost algoritmu**
Port-based	Čísla portů v hlavičce transportní vrstvy	Po prvním paketu	Nízká
DPI	Vzory v uživatelských datech	Po prvním paketu nesoucím uživatelská data	Středně vysoká, vyžaduje přístup k uživatelským datům
SPI	Statistické vlastnosti uživatelských dat	Po několika paketech nesoucích uživatelská data	Vysoká, potřebuje přístup k více paketům s uživatelskými daty
Host-behavior-based	Heuristických vlastností aplikací	Po skončení flow / po určitém časovém úseku	Nízká
Flow-based	Statistických vlastností flow / paketů	Po několika paketech / po určitém časovém úseku	Nízká až středně vysoká

novým a novým výzvám a zároveň některé ze současných výzev nejsou stále překonány.

Stejné tvrzení platí i o algoritmech na detekci anomálií, i ty stále čelí velkému množství výzev, které se ale odlišují na základě aplikační domény. V této práci bude brán ohled pouze na Network Intrusion Detection Systems (NIDS).

Mezi aktuální výzvy, kterým klasifikace internetového provozu čelí jsou včasná a průběžná klasifikace nových záznamů, přenositelnost a v neposlední řadě robustnost programu pro klasifikaci provozu.

Neustále se zvětšující množství internetového provozu spolu s rostoucí propustností sítě způsobuje, že včasná a průběžná klasifikace provozu a detekce anomálií je stále vnímána jako výzva a problém. Pro vyřešení tohoto problému se hledá efektivní algoritmus, který vyžaduje co nejmenší možné výpočetní prostředky pro včasnou klasifikaci takového množství provozu, anglicky se takovým algoritmům říká lightweight algorithms. Apel na rychlost se nevztahuje jenom na samotnou klasifikaci, ale i na fázi předzpracování dat a získávání příznaků z dat.

Detekce anomálií je definována jako hledání vzoru, který neodpovídá normálnímu chování. Pro detekci anomálií je tedy velmi důležité mít dataset obsahující všechny možné vzory normálního chování, avšak získat takový dataset je velmi časově náročné. Navíc je zde snaha výrobců škodlivého softwaru, aby se chování programu nelišilo od normálníhom a tak nepůsobilo podezřele, proto hranice mezi normálním chováním a podezřelým chováním není často široká a jasná.

U detekce anomálií se také dává velký zřetel na úspěšnost a hlavně správnou detekci. Je zde velká shana o potlačení falešných nálezů a snížení množství nedetekovaných anomálií. Falešné nálezy jsou časově i finančně velmi nákladné a nedetekování některé anomálie může mít fatální následek pro napadený systém.

Další problém je udržení modelu stále aktuálním. Výrobci aplikací i škodli vého softwaru stále aktualizují své programy, a proto se jejich chování také mění. Je důležité udržet krok s vývojem aplikací a tím docílit aktuálního modelu.

Nalezení spolehlivých ground truth labelů je samo o sobě častým, důležitým a zajímavým tématem pro výzkum. Řada prací vychází z ground truth labelů poskytnutých od klasifikátoru využívajícího DPI přístup, avšak tento přístup čelí netriviálnímu problému a to nemožnosti klasifikovat šifrovaná data [76].

Kapitola 3

Rešerše existujících prací

Z nepřeberného množství existujících prací na téma klasifikace internetového provozu a detekce anomálií byly brány v potaz hlavně ty, které porovnávají již vzniklé studie či existující algoritmy mezi sebou. Zmiňované algoritmy, co dosahovaly nejlepších výsledků nebo se vyskytovaly ve většině prací, byly použity i v této diplomové práci.

3.1 Rešerse pro klasifikaci provozu

Studie [77] porovnává pět často používaných algoritmů pro klasifikaci síťové komunikace. Jedná se o algoritmy C4.5, Bayesian Network, Naive Bayes Tree (NBTree) a dvě varianty Naive Bayes. Porovnání probíhá na vyváženém datasetu s dvaceti dvěma vybranými příznaky, které jsou zredukovány selekčními metodami Consistency-based a Correlation-based na sedm a devět příznaků. Redukovaný počet příznaků nedosáhl s žádným algoritmem lepších výsledků, zato se zkrátil čas potřebný pro vytvoření klasifikačního modelu a klasifikace samotné minimálně na polovinu. Autoři poukazují na důležitost redukce příznaků před vstupem do algoritmu z pohledu rychlosti sestavení modelu a počtu klasifikací nových záznamů za sekundu. Na poli rychlosti klasifikace vyhrává znatelně algoritmus C4.5. Nevyváženost datasetu studie vůbec neřeší a chybí zmínka o postupu čištění dat a zacházení s chybějícími hodnotami.

Práce od Nguyen a Armitage [78] se zabývá shrnutím studií zabývajících se klasifikací internetového provozu pomocí strojového učení vzniklých v letech 2004 až 2007. Avšak vybrané studie a jejich výsledky nejsou mezi sebou porovnávány, každá studie je prováděna na jiném datasetu, jiných vybraných příznacích i s jinou granularitou klasifikace. Autoři rozdělili algoritmy strojového učení pro klasifikaci internetového provozu do tří přístupů. Ke každému přístupu uvádí autory prací, dosažené výsledky spolu s použitou metrikou a jednotkami. U prvního přístupu shlukování jsou uvedeny čtyři práce. Tyto práce stojí na nesupervizovaných algoritmech Expectation Maximization (EM), AutoClass, Simple K-Means a K-Means. Druhý přístup zahr-

nuje supervizované algoritmy a autoři zde rozebírají algoritmy Nejbližších sousedů založený na statistických vlastnostech komunikace, Naive Bayes s Kernel Estimation, C4.5 Decision tree a REPTree. V posledním semi-supervizovaném učení autoři zmiňují modifikovaný K-means algoritmus. Autoři v práci uvádějí výpis nejčastěji použitých příznaků, mezi které spadají: délky paketů, statistiky časů příchodů paketů (inter-arrival time), počet bajtů v paketech, délka trvání flow, čísla portů a celkový počet paketů ve flow.

P. Foremski sepsal shrnutí [79] třinácti výzkumných prací na téma klasifikace internetového provozu publikovaných mezi lety 2009 až 2012. Zde bude shrnuta pouze sekce zaměřující se na flow-based algoritmy. Objevily se zde algoritmy jako K-Dimensional trees, Markov chains a ensemble přístup využívající sedm nezávislých klasifikátorů, z těchto sedmi klasifikátorů nejlepších výsledků dosahují C4.5, k-NN, Random tree a zástupce umělých neuronových sítí – Multi Layer Perceptron. Zmíněné ensemble učení dosahuje lepší klasifikační úspěšnosti na omezeném počtu paketů ve flow než při klasifikaci celého flow, proto klasifikace komunikace probíhá pouze na prvních čtyřech odchycených paketech a dosahuje 98,4 % úspěšnosti pomocí metriky accuracy. Tato práce stejně jako předešlá pouze shrňuje výzkumy, ale algoritmy samotné mezi sebou neporovnává.

Práce [80] od Bakhshi a Ghita pojednává o dvoufázovém přístupu strojového učení, který je založen na kaskádní kombinaci K-Means a algoritmu C5.0. Autoři kaskádní algoritmus testovali na příznacích z obousměrných flow odchycených na dvou koncových uzlech v síti. Předzpracovaný dataset po selekci příznaků obsahuje šestnáct příznaků, které převážně popisují statistické vlastnosti paketů ve flow. Kaskádní implementace umožňuje dataset obohatit o výstupní label K-Means algoritmu, který provádí hrubozrnnou klasifikaci dat na základě vlastností provozu. Proto do algoritmu C5.0, které má nastavené adaptivní zesilování na hodnotu 100 a prořezávání větví, vstupuje sedmnáct příznaků. Výsledný klasifikátor klasifikuje data do dvanácti tříd definující obecnější aplikační vlastnosti provozu. Dataset neobsahuje žádnou minoritně zastoupenou aplikační třídu, proto se v této práci neřeší problém nevyváženosti datasetu.

Úspěšnost klasifikace flow výsledného kaskádního algoritmu je porovnána s jinými algoritmy jako C4.5, kNN, Naive Bayes, Best-first Decision tree (BFTree), REPTree, Sequential minimal optimization (SMO), Decision tables and naive Bayes (DTNB), Bayes network. Porovnání výsledného algoritmu s ostatními algoritmy na stejném datasetu se stejnými příznaky a stejnou metrikou accuracy ukazuje, že v konečném výsledku je dvoufázový algoritmus úspěšnější o více jak 12% než ostatní nejlepší algoritmy J48, REPTree a kNN. Algoritmy jsou porovnány nejen v úspěšnosti ale i v potřebném výpočetním výkonu a paměťové složitosti a to obojí v přepočtu na flow i bajty. V tomto porovnání si dvoufázový algoritmus vede průměrně, jelikož k získání finálních výsledků musí spustit dva algoritmy strojového učení sériově za sebou.

Studie [81] porovnává dva algoritmy pro klasifikaci internetového provozu,

jeden pro supervizované učení a druhý pro nesupervizované učení. Data se klasifikují do dvou skupin abnormální a normální. Autoři popisují i použitý postup předzpracování dat z datasetu KDD Cup'99 [82] s 42 příznaky, jejichž počet neredukují. Pro určení úspěšnosti algoritmů byla zvolena metrika accuracy a spolu s výsledky jsou zmíněné i výpočetní časy trénovací a testovací fáze. Nesupervizované algoritmy zastupuje algoritmus K-Means s 39 shluky, který dosahuje 83 % úspěšnosti. Vybraný zástupce supervizovaných algoritmů je SVM algoritmus s dvěma typy jader Gaussian RBF a lineárním. Nad SVM algoritmem byly provedeny dva pokusy. První pokus byl proveden na vstupních datech, která nebyla nikterak normalizovaná. Dosažená úspěšnost na nenormalizovaných datech s lineárním jádrem je 72,6 % a 89 % s RBF jádrem. Druhý pokus byl proveden na normalizovaných datech pomocí Min-Max algoritmu. Úspěšnost se zvedla na 91,5% pro lineární jádro a 91,4 % pro RBF jádro. Studie ukazuje, že normalizace může značně pomoci k lepší predikci labelů a značně urychlit učící i trénovací fázi SVM algoritmu. Avšak autoři neuvádějí číselné hodnoty určující čas potřebný pro obě fáze před normalizací.

Studie [45] porovnává šestnáct algoritmů za účelem klasifikace internetového provozu. Mezi tyto algoritmy se řadí například SVM, k-NN, Random Forest, Rotation Forest, REPTree, Artificial Neural Network (ANN) a řada dalších. Algoritmy jsou testovány na třinácti datasetech. Datasety pokrývají internetovou komunikaci zachycenou na různých typech sítí, jak internetovou komunikaci na lokální síti (LAN) tak páteřní síti. Klasifikace probíhá na úrovni typu komunikace a rozřazuje flow do devíti tříd, jmenovitě: mail, chat, OG, VoIP, bulk, attack, MM, NW, P2P.

Ve studii se srovnávají výsledky algoritmů, které přijímají: celé flow a flow s omezeným počtem paketů, uni-flow a bi-flow. Ve studii zkouší i různé sety příznaků. Výsledek testu s celým flow a flow s pouze omezeným počtem paketů ukazuje, že s omezeným počtem paketů lze dosáhnout stejných výsledků a dokonce i v jedenácti případech lepších výsledků. Výsledek mezi použitím bi-flow a uni-flow není překvapivý. Bi-flow si v přesnosti algoritmů vedlo okolo 2 % úspěšněji ve všech případech a dokonce čas potřebný pro vytvoření modelu během trénovací fáze byl v některých případech až o 40 % menší (ANN, decision table/naive bayes, všechny varianty AdaBoost).

Autoři uvádějí souhrn nejčastěji používaných příznaků pro klasifikaci a to ze 40 prací. Mezi tyto příznaky patří: průměrný inter-arrival čas, počet paketů a bajtů za sekundu, celkový počet bajtů ve flow, celkový neaktivní čas, čas trvání flow, průměrná velikost IP paketu a uživatelských dat, standardní odchylka inter-arrival času, flagy v TCP hlavičce, průměrná velikost TCP okna. Jejich výběr příznaků, se kterým dosáhli nejlepších výsledků, zahrnoval příznaky: minimální, maximální a průměrné velikosti paketu, maximální, minimální a průměrný inter-arrival čas a hodnotu směrodatné odchylky pro velikost paketu i inter-arrival čas.

3.2 Rešerse pro detekcí anomálií

Práce [6] představuje algoritmus izolačních lesů pro detekci anomálií, který porovnávají v té době s existujícími a nejčastěji používanými algoritmy: ORCA, one-class SVM a LOF. Porovnání proběhlo i na dvou datasetech s internetovou komunikací, přesněji HTTP a SMTP komunikací z KDD Cup'99. Z datasetů ale byly vybrány pouze příznaky se spojitou číselnou hodnotou, binární a kategoriální příznaky nebyly brány vůbec v potaz. HTTP dataset obsahuje $0,4$ % anomálií a SMTP dataset obsahuje pouze $0,03$ % anomálií. Jako hodnotící metrika byla použita AUC. Výsledky ukazují, že v případě obou datasetů si izolační les vede lépe o více jak 10 % oproti ostatním algoritmům. V celkové časové složitosti je algoritmus výrazně nejrychlejší.

Chandola spolu s kolegy vytvořili všeobecný průzkum na téma detekce anomálií [16]. Rozebírají výpočetní složitosti, výhody a nevýhody jednotlivých přístupů a jejich aplikační doménu. Mezi aplikačními doménami zmiňují i NIDS a k nim nejčastěji používané algoritmy: statistické modely, neuronové sítě, SVM, shlukovací modely a k-NN.

Přehled technik a nástrojů [83] pro detekci anomálií na počítačové síti z roku 2014 rozděluje techniky na statisticky založené, klasifikační, založené na evolučních algoritmech nebo fuzzy logice, shlukovací a založené na algoritmech pro detekci odlehlých hodnot, v neposlední řadě techniky založené na vědomostech a pravidlech. Přehled obsahuje seznam technik, autorů, použité datasety a slovně shrnuté výsledky a myšlenky studie. Techniky nejsou porovnávány mezi sebou a není zmíněná přesná úspěšnost na datasetech. Na závěr kapitoly autoři obecně shrnují slabiny a silné stránky jednotlivých odvětví, kde zdůrazňují, že každý přístup má své výhody i nevýhody pro určitý druh problému. Jedna z kapitol popisuje nástroje pro odchycení provozu a získávání příznaků.

Porovnání algoritmů pro detekci anomálií spolu s různým výběrem příznaků se zabývá práce [84]. Pro pokus si autoři vybrali datasety DARPA'98, KDD Cup'99 a NSL-KDD, které na začátku obsahují 41 příznaků. V práci je použito několik algoritmů pro výběr příznaku spolu s nový algoritmem pro selekci příznaků, jenž zredukoval počet příznaků na 30 a 16. Po předzpracování dat vyzkoušeli všechny vybrané příznaky na algoritmech: k-NN, NN, SVM, Naive Bayes a rozhodovacích stromech. Při výběru příznaků si nejlépe vedly ensemble algoritmy Least Absolute Selection and Shrinkage Operator (LASSO) a Significance Analysis for Microarrays (SAM). Při klasifikaci pak nejlepších výsledků dosahovaly rozhodovací stromy a k-NN, hned v závěsu za nimi je neuronová síť, ta však daleko zaostává v rychlosti, čas potřebný pro výpočet je desetkrát delší.

Jedna z mála prací, která uvádí dosažených výsledků v procentech a u některých i míru chybovosti je práce [85]. Práce ukazuje, že v letech 2009 až 2014 vzniklo více studií, které se zabývaly hybridními klasifikátory než singl klasifikátory. Hybridní klasifikátory jsou klasifikátory, které využívají různé

algoritmy strojového učení pro dosažení lepších výsledků. Singl klasifikátory jsou složené pouze z jednoho algoritmu strojového učení. Mezi nejvíce populární singl klasifikátory pro NIDS spadá SVM a neuronové sítě. U hybridních přístupů se často vyskytovalo využití k-NN společně s evolučními algoritmy či SVM. Překvapivý výsledek práce je, že ze všech 49 zahrnutých výzkumů většina nepoužila ani nezvážila žádnou formu výběru příznaků.

Kapitola 4

Předzpracování dat

Algoritmus je tak dobrý, jak dobré jsou jeho vstupní data. Proces, jak vytvořit ze získaných příznaků ještě lepší vstupní data do algoritmu strojového učení, bude rozebrán v této kapitole. Ve fázi předzpracování vstupu do algoritmu strojového učení dochází k takzvanému čištění dat, převedení na číselné hodnoty a normalizace příznaků, celý proces lze označit anglickým jménem Feature Transformation. Fáze předzpracování dat je ukončena možnou selekcí důležitých příznaků pro klasifikaci.

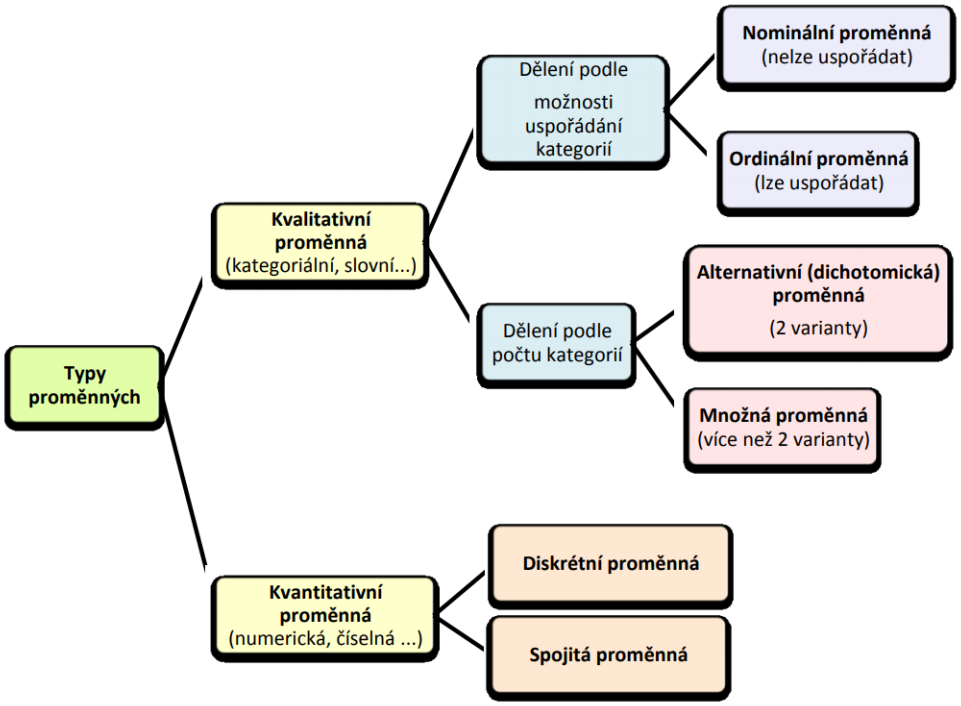
4.1 Typy dat

Řada algoritmů strojového učení umí zacházet pouze s určitým typem vstupních dat. S ostatními typy dat neumí pracovat nebo vedou k velmi slabému klasifikačnímu modelu. Rozeznat a určit typ vstupních dat je na uživatelovi a jeho znalosti o datasetu.

Základní rozdělení typů dat je na kvalitativní data, též v této diplomové práci zvaná kategoriální data, a kvantitativní data. Kvalitativní a kvantitativní data se dále rozdělují na podkategorie. V rámci této diplomové práce si lze vystačit s pojmy kategoriální data a podkategoriemi kvantitativních dat a to diskrétní data a spojitá data.

Kategoriální data popisují kvalitu a charakteristiku jevů. Nejčastější reprezentace dat je slovní a obvykle nabývají hodnot z konečné množiny. Protože hodnoty kategoriálních dat nemají definovaný poměr, nelze je mezi sebou měřit nebo jakkoliv porovnávat.

Kvalitativní data mají definovaný poměr jednotlivých hodnot, proto je lze porovnávat a provádět aritmetické operace. Mezi tato data se řadí například všechny fyzikální veličiny. Kvalitativní data se dále rozdělují na diskrétní a spojitá. Diskrétní data nabývají spočetně mnoha hodnot z konečného nebo nekonečného intervalu. Naopak spojitá data nabývají libovolných hodnot z konečného nebo nekonečného intervalu.



Obrázek 4.1: Rozdělení základních typů dat [14]

4.2 Šumu, odlehlé hodnoty a anomálie

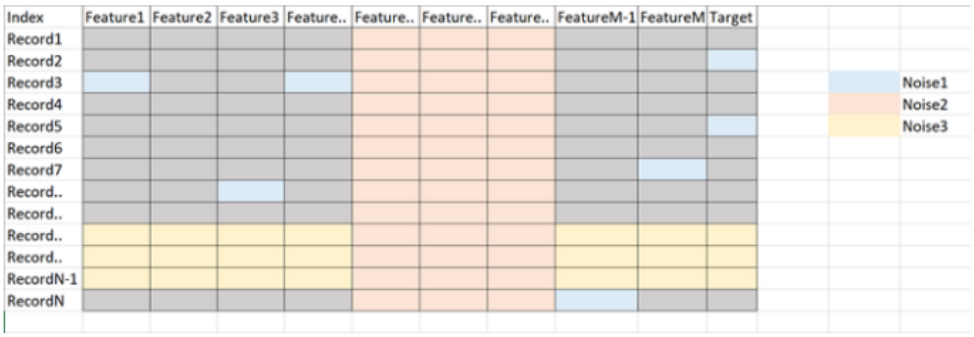
4.2.1 Šum

Šum (noise) je nekonzistentní hodnota, nejčastěji jev způsobený chybou měření, proto je šum někdy nazýván chybnou hodnotou. Nekonzistentní hodnoty porušují pravidla a omezení, která jsou na ně kladené. Proto šum může svou hodnotou způsobit, že algoritmus nenajde vypovídající vzor o datech. Příklady šumu:

- Číselný sloupec, který obsahuje i řetězce znaků.
- IPv6 a IPv4 adresa neobsahující pevně definovaný formát.
- Záporné číslo portu nebo vyšší než hodnota $2^{16} - 1 = 65535$.

Obrázek 4.2 zobrazuje různé typy šumu podle rozsahu:

- Ojedinělý šum v jednom z příznaků v záznamu (Noise1).
- Šum jako jeden příznak (Noise2).
- Šum jako celý záznam (Noise3).



Obrázek 4.2: Ukázka různých typů šumu [15]

4.2.2 Odlehlá hodnota

Odlehlá hodnota (outlier) není zapříčiněna chybou v meření, pouze se svou hodnotou výrazně odchyluje od mediánu nebo průměru rozdělení dat, do kterého spadá. Odlehlý bod je záznam, který má některou nebo více svých hodnot odlehlých. Například Hawkins, D. M, ve své práci [86] popisuje odlehlé body následovně (volný překlade autora):

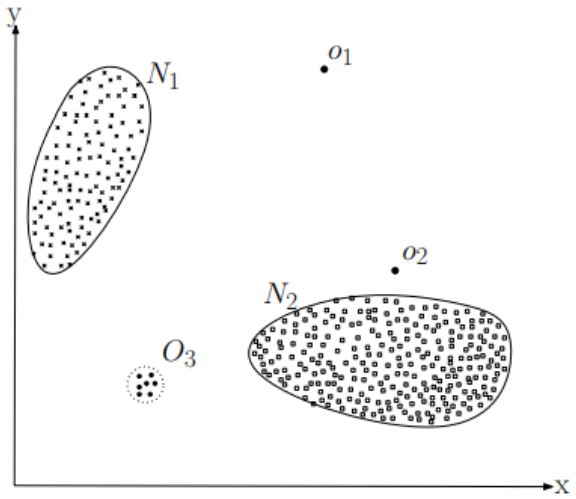
> "Odlehlý bod je měření, které se odlišuje od ostatních měření natolik až vyvolává podezření, že bylo vygenerováno jiným mechanismem."

V analýze dat značí odlehlé body v datasetu překážku, jelikož ztěžují proces nalezení souvislostí mezi daty.

4.2.3 Anomálie

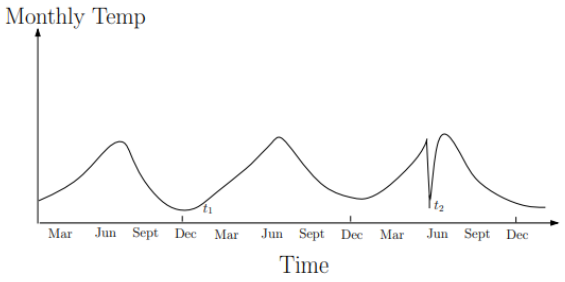
K definicím výše uvedených pojmů přichází další pojem: Anomálie. Jak už to na poli vědy bývá, každý obor si pojem anomálie definuje podle svých potřeb. Proto se přesná definice mění podle kontextu, nejčastěji se jedná o odlehlou hodnotu, která by se neměla vyskytovat v záznamech a zároveň je objektem zájmu. V této diplomové práci bude pod pojmem anomálie myšlena konzistentní hodnota, která poukazuje na škodlivou aktivitu na internetu. Varun Chandola [16] definici anomálie v kontextu detekce anomálií dále rozděluje na bodové, kontextové a kolektivní anomálie.

Bodová anomálie: Bodová či také globální anomálie je nejjednodušší typ anomálie, a také se dá i nejjednodušeji odhalit. Jedná se o záznam, který je velmi odlehlý od zbytku dat. Na obrázku 4.3 je bodová anomálie reprezentována body o_1 a o_2 a shlukem bodů O_3.



Obrázek 4.3: Bodová anomálie [16]

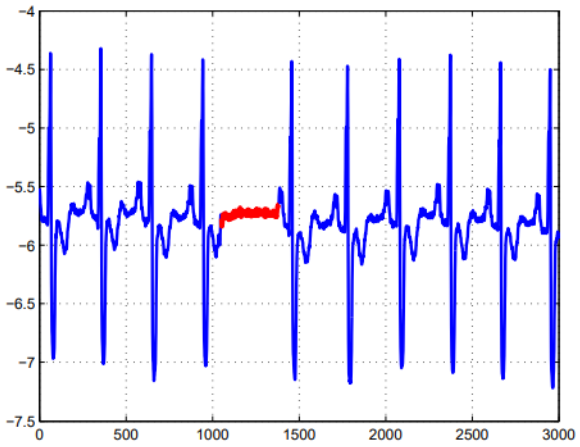
Kontextové anomálie: Jedná se o anomálii, která je odlišná od zbytku dat pouze z pohledu kontextu, ne jinak. Pro jednodušší pochopení je zde uveden obrázek 4.4.



Obrázek 4.4: Kontextová anomálie [16]

Na obrázku 4.4 jsou dva body t_1 a t_2, které mají stejnou hodnotu, ale z pohledu kontextu je patrné, že bod t_2 je vyhodnocen jako anomálie, zatímco bod t_1 je naprosto normální a očekávané chování.

Kolektivní anomálie: Kolektivní anomálie je sada záznamů, které se objevují spolu a vzhledem k zbylým datům tvoří anomálii. Individuální záznamy z kolektivní anomálie nemusí být sami o sobě anomáliemi. Například na obrázku 4.5 červeně zvýrazněná část křivky odpovídá kolektivní anomálii, avšak vezme-li se z červené části křivky pouze jeden červený bod a porovná se vůči ostatním modrým bodům, pak tento červený bod vzhledem k bodům modrým s největší pravděpodobností nepředstavuje anomálii.



Obrázek 4.5: Graf zobrazující kolektivní anomálii zvýrazněnou červenou barvou [16]

4.3 Čištění dat

Při čištění dat dochází k odebrání řádků a sloupců či pozměnění hodnot příznaků, které by mohly mít z důvodu nesprávného měření nepříznivý vliv na výpočet algoritmu.

4.3.1 Odstranění redundantních řádků

Nechat či nenechat redundantní řádky? Často kladená otázka, na kterou neexistuje jedna univerzální odpověď ano či ne. Odpověď záleží na použitém datasetu a jevu, který tento dataset popisuje. Pokud dataset neobsahuje velké množství redundantních řádků, údaje odpovídají reálnému jevu a předpokládá se, že jev způsobující redundantní hodnoty se bude nadále objevovat, pak je vhodné řádky zachovat. Stejné tvrzení platí i pokud se objevuje větší množství redundantních řádků, ale podmíněné vhodným ověřením, čím je jev způsobován a zda se jedná o správné chování. Na druhou stranu, redundantní řádky mohou být odstraněny v případě, že opakováním nevypovídají o současných ani budoucích vlastnostech reálného jevu.

Finální rozhodnutí o ponechání nebo vymazání redundantních řádků závisí na použitém modelu strojového učení.

4.3.2 Odstranění sloupců

Obecně se pokládá za správné smazat sloupce, kterým chybí většina hodnot. Například od 60 % výše, kdy jsou naměřené hodnoty zastoupeny 40 % a méně. U malých datasetů se sloupce obvykle nemažou, aby nedocházelo ke ztrátě

potřebných informací pro vytvoření modelu. U větších datasetů, obsahujících dostatek informací, si lze smazání sloupců dovolit [87].

4.3.3 Vypořádání se s chybějícími hodnotami

Nejjednodušší přístupy k řešení chybějících hodnot je odstranění řádků obsahujících chybějící hodnotu nebo doplnění hodnotou náhodně vybranou ze sloupce. Další jednoduché přístupy zahrnují využití statistických vlastností dat jako doplnění chybějící hodnoty hodnotou mediánu, modusu nebo průměru z daného sloupce dat.

Chybí-li větší množství hodnot, ani jedna z výše uvedených metod není doporučována. Nejprve je vhodné se zaměřit na příčinu chybějících dat. Pokud je možnost vyřešit tuto příčinu, dosáhne se toho nejlepšího řešení a to doplnění chybějících hodnot hodnotami reálnými. Nelze-li však změnit příčinu, nezbývá než vyřešit chybějící hodnoty samotné. V tomto případě se může využít doplnění dat pomocí strojového učení (například metoda nejbližších sousedů nebo MICE (Multivariate Imputation by Chained Equation)) nebo doplnění dat hodnotami vybranými z rozdělení dat. Kniha Flexible Imputation of Missing Data [88], rozsáhle pojednává o typech chybějících dat a metodách, jak tyto případy řešit.

4.3.3.1 Metoda nejbližších sousedů

Chybějící hodnota je doplněna na základě hodnot nejbližších sousedů, u kterých v daném sloupci hodnoty nechybí. V případě kvantitativních dat se jedná o doplnění hodnotou průměru a v případě kategoriálních dat hodnotou modusu.

Nevýhoda této metody je její časová náročnost při velkém množství chybějících hodnot, jelikož se pro každou chybějící hodnotu musí procházet celý dataset a hledat nejbližší sousedy pro záznam obsahující danou chybějící hodnotu.

4.3.3.2 Rozdělení dat

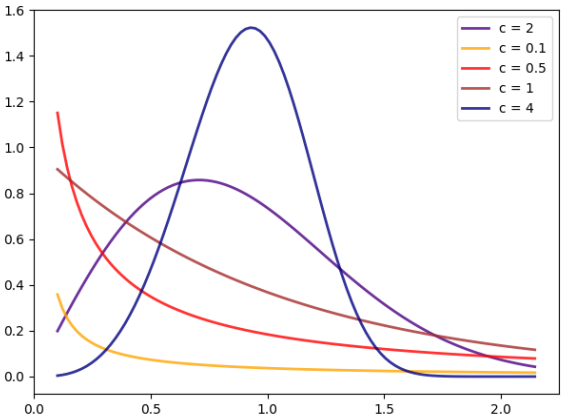
Najít známé rozdělení, které nejvěrohodněji popsije vstupní data spadá do velmi klíčového úkonu, čím přesněji bude rozdělení popisovat data, tím jsou chybějící hodnoty doplněny více vypovídající hodnotou. Jedná se však o nelehký úkol.

Gaussovské rozdělení, také nazývané jako Bellova křivka (Bell Curve) či normální rozdělění, je jedno z nejvíce využívaných rozdělení. Jedná se o rozdělení, které se objevuje u velkého množství dat, například u přírodních jevů, odhadů či strojových měření [89]. Mezi další hojně využívané rozdělení patří exponenciální nebo Poissonovo rozdělení, v případě této diplomové práce bude použito Weibullovo rozdělení.

Weibullovo rozdělení: Definice Weibullova rozdělení je brána z implementace funkce `weibull_min` [90] ve SciPy knihovně [91]. Weibullovo rozdělení pro spojité náhodné veličiny je popsáno jako funkce:

$$f(x, c) = cx^{c-1} \exp(-x^c),$$

pro $x > 0$, $c > 0$. Parametr c popisuje tvar funkce. Jeli parametr c roven jedné, potom se z Weibullovy funkce stává exponencionální funkce. Obrázek 4.6 zobrazuje graf hustoty Weibullova rozdělení pro různé hodnoty parametru c.



Obrázek 4.6: Graf hustoty Weibullova rozdělení pro různé hodnoty parametru c

4.3.4 Odstranění šumu

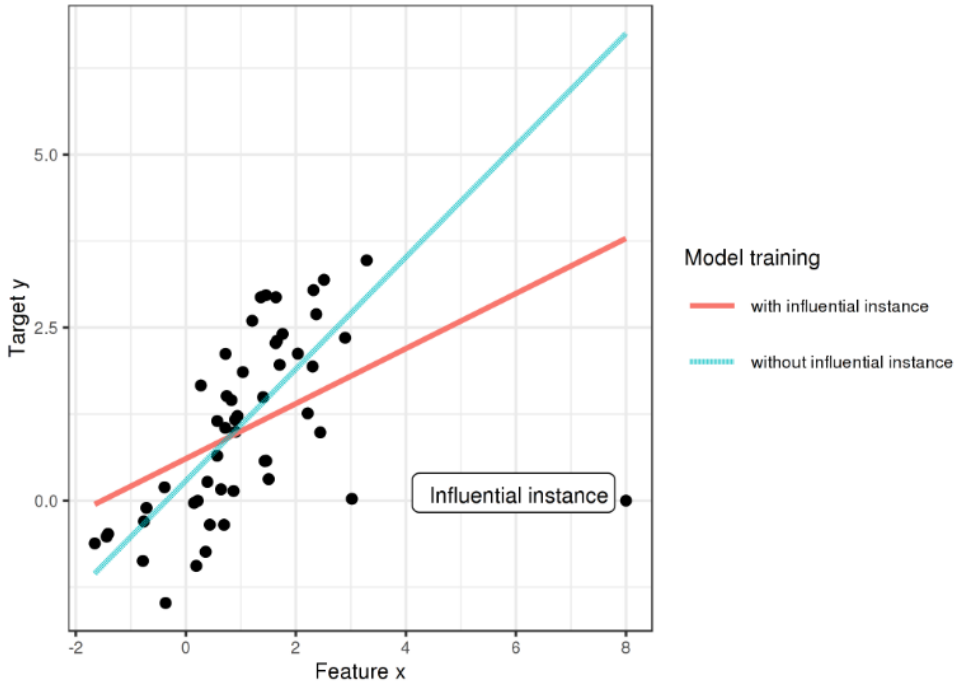
Mezi hodnoty, které mohou mít nepříznivý vliv na výpočet algoritmu patří především šum ale i odlehlé hodnoty.

Odlehlé hodnoty nemusí mít vždy špatný vliv na algoritmus, často totiž záleží na volbě algoritmu samotného. Existují algoritmy, které jsou robustní vůči odlehlým hodnotám, jako například algoritmy založené na rozhodovacích stromech. Naopak algoritmy založené na lineární regresi jsou náchylné na odlehlé hodnoty. Hodnoty, které mají špatný vliv na regresní křivku dat se nazývají influential body.

Důsledkem šumu a influential bodů může být značné prodloužení doby běhu algoritmů a hlavně zhoršení jejich přesnosti [92]. Například důvodem tohoto chování u algoritmů založených na regresi je, že hodnoty šumu a influential body ovlivňují sklon regresní křivky, viz obrázek 4.7.

Na obrázku 4.7 popisuje červená přímka lineární regresi se zahrnutým influential bodem a modrá přímka lineární regresi bez influential bodu.

Proto odstraněním či změnou hodnot influential bodů a šumu lze u některých algoritmů docílit:



Obrázek 4.7: Vliv influential bodu na sklon regresní křivky [17]

- zrychlení učícího algoritmu,
- zjednodušení modelu pro rozhodování, tím se zjednoduší i jeho následná interpretace,
- zvýšení přesnosti algoritmu,
- redukce overfitting problému.

V případě algoritmů pro detekcí anomálií je naopak vhodné ponechat odlehlé hodnoty, v tomto případě se totiž může jednat právě o anomálie.

4.4 Transformace příznaků

Proces transformace dat do optimální formy pro algoritmus strojového učení je znám jako transformace příznaků, feature transformation. Transformace příznaků označuje rodinu algoritmů, která ze vstupních příznaků vytvoří nové příznaky. Tyto nové příznaky mohou mít stejnou interpretaci jako originální, ale oproti nim mohou mít v jiném prostoru větší diskriminační sílu. Transformace příznaků má hodně podob, od jednoduchých lineárních až po nelineární funkce.

4.4.1 Kódování kategoriálních dat

Řada algoritmů strojového učení umí pracovat pouze s numerickým vstupem a neumí zacházet s kategoriálními daty. Tato vlastnost je způsobena faktem, že algoritmy strojového učení jsou matematické modely, které příjmají pouze číselné hodnoty.

Algoritmy strojového učení předpokládají, že vstupní čísla mají mezi sebou logické souvislosti a lze nad nimi provádět všechny matematické operace. Což v případě převedených kategoriálních dat pouze do číselné podoby není pravda, proto existuje možnost transformace kategoriálních data. Transformace kategoriálních dat zajistí, aby i nad jejich hodnotami šlo provádět matematické operace, bez hledání souvislostí, které data mezi sebou nemají. Procesu převedení kategoriálních dat na číselné hodnoty se říká kódování dat. V této podkapitole budou rozebrány tyto typy kódování:

- Ordinal kódování,
- OneHot a dummy kódování,
- Binární kódování,

Mezi algoritmy strojového učení, které umí pracovat s kategoriálními typy dat převedenými pouze do číselné podoby, spadají algoritmy založené na stromech. Naopak například SVM algoritmus a mnohé shlukovací algoritmy neumí pracovat s kategoriálními daty.

Ordinal kódování: První nejznámější kódování je Ordinal kódování. Jsou-li na vstupu kategoriální data obsahující N kategorií, pak algoritmus převede kategoriální hodnotu právě na jednu číselnou hodnotu v rozmezí od 0 do $N-1$, viz ukázka 4.1. Nevýhoda tohoto kódování je, že jejich číselná hodnota nereprezentuje vůbec žádný vztah mezi původními hodnotami.

OneHot a dummy kódování: Spadá mezi nejznámější druh kódování kategoriálních dat. Při OneHot kódování algoritmus zakóduje vstupní sloupec do tolika binárních sloupců, kolik obsahuje unikátních hodnot. Zakódovaný sloupec popisuje právě jednu unikátní hodnotu původního sloupce a obsahuje jedničky na těch řádcích, které obsahovaly původní hodnotu, viz ukázka 4.2. Toto kódování není vhodné pro kategoriální data obsahující velké množství unikátních hodnot.

Obdoba OneHot kódování je dummy kódování, které kóduje n unikátních hodnot do $n-1$ sloupců, přičemž jedna z hodnot je reprezentována nulami na řádků nově vzniklých sloupců.

Tabulka 4.1: Ukázka Ordinal kódování kategoriálních dat pro kód země zdroje (sCo)

(a) Kategoriální data

sCo
SK
ZZ
GB
ZZ
JP
CZ
CZ
UK
JP

(b) Použito Ordinal kódování

sCo
0
1
2
1
3
4
4
5
3

Tabulka 4.2: Ukázka OneHot kódování kategoriálních dat pro kód země zdroje (sCo)

(a) Kategoriální data

sCo
SK
ZZ
GB
ZZ
JP
CZ
CZ
UK
JP

(b) Použito OneHot kódování

sCo_cz	sCo_gb	sCo_jp	sCo_sk	sCo_uk	sCo_zz
0	0	0	1	0	0
0	0	0	0	0	1
0	1	0	0	0	0
0	0	0	0	0	1
0	0	1	0	0	0
1	0	0	0	0	0
1	0	0	0	0	0
0	0	0	0	1	0
0	0	1	0	0	0

Binární kódování: Při kódování dochází k převodu všech N unikátních hodnot na N unikátní binárních čísel, která jsou zarovnána na stejně dlouhý binární zápis, viz ukázka 4.3. Zápis binárního čísla je reprezentován sloupci, kde jeden sloupec reprezentuje právě jednu pozici v binárním čísle. Počet sloupců je roven délce binárního zápisu hodnot.

4.4.2 Transformace rozdělení dat

Během transformace rozdělení dat dochází k náhradě hodnot příznaku x_j hodnotami nějaké funkce tohoto příznaku. Jedná se o náhradu, která mění tvar distribuce příznaku nebo vztahy mezi hodnotami.

Nejčastější důvody k transformaci rozdělení dat jsou redukce šikmosti, přiblížení blíže k rovnoměrnému rozdělení, či nalezení lineárních a aditivních

Tabulka 4.3: Ukázka binárního kódování kategoriálních dat pro kód země zdroje (sCo)

(a) Kategoriální data

sCo
SK
ZZ
GB
ZZ
JP
CZ
CZ
UK
JP

(b) Binární hodnota dat

sCo
000
001
010
001
011
100
100
101
011

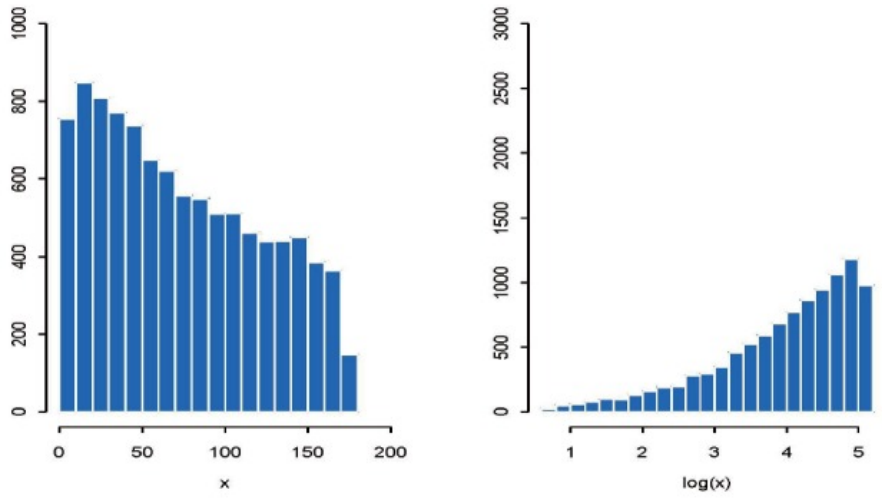
(c) Binární kódování dat

sCo_2	sCo_1	sCo_0
0	0	0
0	0	1
0	1	0
0	0	1
0	1	1
1	0	0
1	0	0
1	0	1
0	1	1

vztahů mezi příznaky.

Samotná transformace rozdělení dat pak může být interpetována jako logaritmická funkce, převrácená hodnota, umocňování, třetí odmocnina, či transformace pomocí kvartilů.

Použití transformace rozdělení vyžaduje dobrou znalost rozdělení všech příznaků z datasetu. Může se totiž jednoduše stát, že místo užitku může transformace rozdělení spíše uškodit. Například je-li logaritmická funkce použita na nevhodná vstupní data, pak se výsledná šikmost může zhoršit, tato vlastnost je rozebírána v práci [18]. Viz ukázka 4.8.



Obrázek 4.8: Logaritmická transformace rozdělení vstupního příznaku x [18]

Na obrázku 4.8 levý histogram popsisuje příznak x s koeficientem šikmosti

0,34. Pravý histogram obsahuje příznak po logaritmické transformaci. Šikmost se převrátila a zvětšila na hodnotu -1,16.

Tato diplomová práce bere v potaz pouze transformace rozdělení dat do normálního rozdělení a rovnoměrného rozdělení.

4.4.2.1 Normální rozdělení:

Pro transformaci rozdělení příznaku x_j do normálního rozdělení se používá například:

Třetí odmocnina: $y = x_j^{\frac{1}{3}}$

Logaritmická funkce: $y = log(1 + x_j)$, lze použít jen na kladná vstupní data x_j.

Parametrická monotónní funkce: Transformace rozdělení dat do normálního rozdělení pomocí Yeo-Johnson transformace a Box-Cox transformace. Metoda Yeo-Johnson umí pracovat s kladnými i zápornými hodnotami oproti metodě Box-Cox, která striktně vyžaduje kladná vstupní data.

Yeo Johnson transformace je dána vzorcem z [93], pro $\lambda \in \mathbb{R}$:

$$x_j^{(\lambda)} = \begin{cases} [(x_j + 1)^\lambda - 1]/\lambda & \text{if } \lambda \neq 0, x_j \geq 0, \\ \ln(x_j) + 1 & \text{if } \lambda = 0, x_j \geq 0, \\ -[(-x_j + 1)^{2-\lambda} - 1]/(2 - \lambda) & \text{if } \lambda \neq 2, x_j < 0, \\ -\ln(-x_j + 1) & \text{if } \lambda = 2, x_j < 0. \end{cases}$$

Box Cox transformace je dána vzorcem z [94], pro $\lambda \in \mathbb{R}$:

$$x_j^{(\lambda)} = \begin{cases} \frac{x_j^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ -\ln(x_j) & \text{if } \lambda = 0. \end{cases}$$

4.4.2.2 Rovnoměrné rozdělení

Rovnoměrného rozdělení lze docílit pomocí využití komulativní distribuční funkce a kvantilové funkce, neboli inverzní distribuční funkce. Jedná se o nelineární transformaci, která rozprostírá nejčastěji se vyskytující hodnoty po celém výstupním rozdělení:

$$y = G^{-1}(F(x_j)),$$

kde $F(x)$ je komulativní distribuční funkce příznaku x_j, funkce G má požadované výstupní rozdělení a G^{-1} je k ní inverzní distribuční funkce. Výstupní rozdělení příznaku x_j udává fuknce G a její rozdělení, proto lze vzoreček využít i pro normální rozdělení.

Transformace rozdělení na rovnoměrné rozdělení eliminuje hodnoty odlehlých bodů. Transformace rozdělení je aplikována na každý příznak nezávisle, proto nezachovává korelace a vzdálenosti napříč příznaky z datasetu.

4.4.3 Normalizace dat

Dataset může obsahovat příznaky, jejichž hodnoty se pohybují ve velmi odli šných řádech. Některé z příznaků se mohou pohybovat hodnotami v řádu jednotek, jiné například v řádu tisíců a výše. Cíl normalizace je sjednotit hodnoty do stejného měřítka, bez narušení vzdálenosti mezi daty a tím reprezentovat data v podobě, která více vyhovuje algoritmům strojového učení. Tomuto procesu se také říká škálování dat. K normalizaci dat se používají statistiky odvozené ze vstupních dat jako průměr, směrodatná odchylka, rozpětí a maximum.

Obsahuje-li dataset příznaky, které značně dominují svou hodnotou nad zbylými, může se stát, že algoritmus strojového učení bude upřednostňovat příznaky s vyššími hodnotami, než se naučí od příznaků s nižšími hodnotami. Algoritmy strojového učení jako SVM, k-nejbližších sousedů a logistická regrese vyžadují normalizované příznaky [95]. Normalizace může značně pomoci i při konvergenci do lokálního či globálního minima algoritmům využívající klesání podle gradientu (gradient descent).

4.4.3.1 Normalizace směrodatnou odchylkou

Data normalizovaná směrodatnou odchylkou mají průměr rovnen nule a směrodatnou odchylku rovné jedné. Nová hodnota pro příznak x_j se ze vstupních dat X získá odečtením výběrového průměru $\bar{x}_j$ z hodnot příznaku od původní hodnoty a podělením směrodatnou odchylkou s_j. Tato normalizace není vhodná, pokud vstupní data nemají normální rozdělení a obsahují odlehlé hodnoty. Obrázek 4.9 znázorňuje dopad normalizace na krabicový graf se šesti příznaky s odlišnými hodnotami a rozptylem dat:

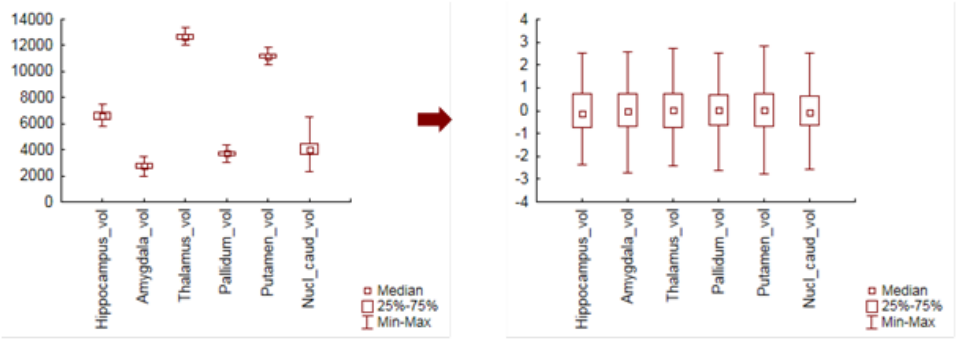
$$\bar{x}_j = \frac{1}{N}\sum_{i=1}^{N}(x_j)_i,$$

$$s_j = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}((x_j)_i - \bar{x}_j)^2},$$

$$y = \frac{x_j - \bar{x}_j}{s_j}.$$

4.4.3.2 Normalizace rozpětím

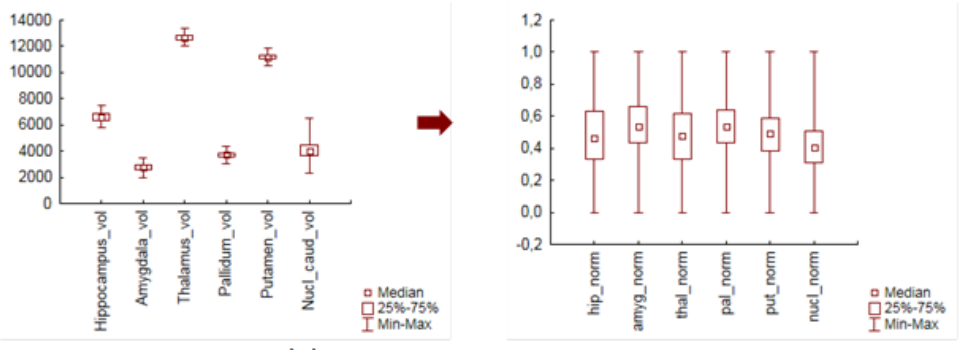
Normalizace rozpětím škáluje vstupní data X, přesněji jednotlivé příznaky x_j, do předem daného rozmezí, obvykle od 0 do 1, viz obrázek 4.10. Tato



Obrázek 4.9: Ukázka krabicového grafu původních příznaků (vlevo) a příznaků normalizovaných směrodatnou odchylkou (vpravo) [19]

normalizace je narozdíl od normalizace směrodatnou odchylkou vhodná při použití na data, která nemají normální rozdělení. Avšak stále nevhodná na data obsahující odlehlé hodnoty.

$$y = \frac{x_j - \min(x_j)}{\max(x_j) - \min(x_j)}$$



Obrázek 4.10: Ukázka krabicového grafu původních příznaků (vlevo) a příznaků normalizovaných rozpětím (vpravo) [19]

4.4.3.3 Normalizace na maximum příznaku

Jedná se o normalizaci příznaku pomocí absolutní hodnoty jeho maxima, kterou nastaví na hodnotu $1,0$ a všechny ostatní hodnoty jsou posunuty tak, aby jejich vzdálenosti byly zachovány. Jednodušeji řečeno, všechny hodnoty příznaku jsou poděleny absolutní hodnotou maximální hodnoty příznaku:

$$y = \frac{x_j}{abs(\max(x_j))}.$$

4.4.3.4 Normalizace na základě kvartilů

Střední hodnota výstupních dat je rovna nule a hodnoty jsou normalizovány na základě mezikvartilového rozpětí IQR, které se definuje jako rozdíl $IQR = q_{0.75} - q_{0.25}$, kde $q_{0.75}$ značí 75. percentil, tak zvaný horní kvartil, a $q_{0.25}$ značí 25. percentil, tak zvaný dolní kvartil:

$$y = \frac{x_j - median(x_j)}{IQR(x_j)}.$$

Normalizace na základě kvartilů je jako jediná ze zmíněných funkcí robustní vůči odlehlým hodnotám.

4.5 Volba a výběr příznaků

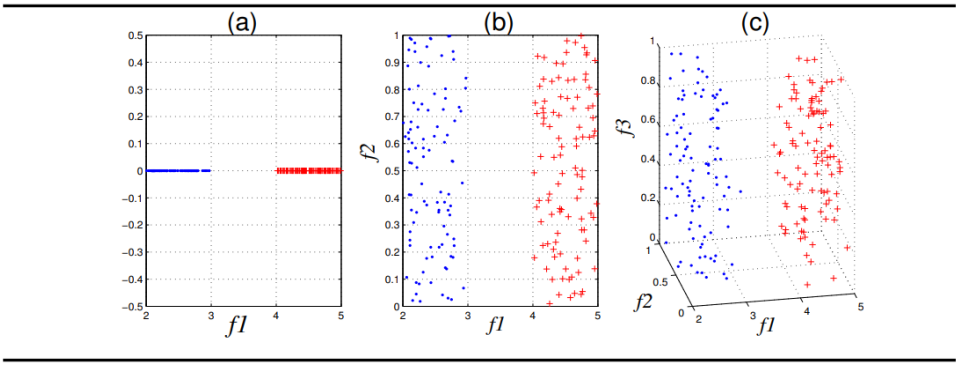
Volba a výběr příznaků (dimensionality reduction) se používá tam, kde se pracuje se stovkami či dokonce tisíci příznaky, z nichž jsou mnohé irelevantní, redundantní nebo obsahují šum. Irelevantní a redundantní hodnoty příznaků komplikují proces učení a ten se stává náročnější jak časově tak i paměťově, navíc výsledek učícího algoritmu může mít matoucí či zavádějící vlastnosti. Irelevantní příznaky nejsou přímo spojeny s cílovou hodnotou labelu a tudíž nenesou dostatečnou informativní sílu k jeho určení, ale přitom stále ovlivňují proces učení, viz obrázek 4.11. Redundantní hodnoty naopak nepřidávají žádnou novou asociaci vzhledem k labelu [96, 97].

Dataset s vysokou dimenzionalitou příznaků může vést k finálnímu modelu strojového učení který tíhne k overfitting problému [20]. Navíc s rostoucí dimenzí nastává problém velké rozptýlenosti dat po prostoru (curse of dimensionality) řídkost dat může způsobovat nižší přesnosti algoritmu strojového učení [98].

Cílem výběru příznaků je nalézt takovou podmnožinu příznaků, která je dostatečná pro zachování co nejvíce původní informace vzhledem k nějakému kritériu a vede k očekávanému rozhodnutí algoritmu strojového učení. Výběr příznaků lze také označovat jako redukci příznaků, jelikož promítá data do prostoru nižší dimenze a je prováděna pomocí selekce či kombinace příznaků z datasetu. Souhrnně lze říci, že jsou zde čtyři základní aspekty:

- Jednodušší interpretace modelu. V případě volby selekce je docíleno i následného zjednodušení procesu získávání příznaků, jelikož již není potřeba získávat všechny.
- Lepší a efektivnější predikce a klasifikace bez overfitting problému.
- Vyřešení problému velkého rozptýlení dat po prostoru s vysokou dimenzí.

Volba a výběr příznaků se rozděluje na dva přístupy: selekce příznaků, Feature Selection, a extrakce příznaků, Feature Extraction.



Obrázek 4.11: Ukázka tří příznaků a jejich relevance vůči labelu. Příznak $f1$ je relavantní, zatímco příznaky $f2$ a $f3$ jsou irelevantní. Pomocí příznaku $f1$ lze rozlišit mezi dvěma skupinami shluků. Odstranění příznaků $f2$ a $f3$ nebude mít žádný negativní efekt na přesnost algoritmu [20]

4.5.1 Selekce příznaků

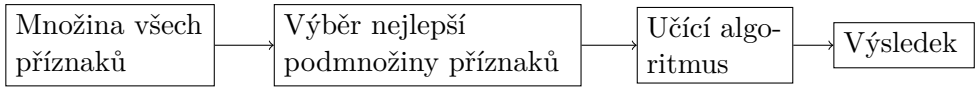
Selekce příznaků vybírá ze stávajícího počtu M příznaků podmnožinu s $N < M$ příznaky na základě hodnotícího kritéria. Během výběru nedochází ke změnám hodnot původních příznaků. Tento přístup lze také popsat jako proces, který odebírá příznaky, které podle hodnotícího kritéria nejsou relevantní.

Selekci příznaků lze rozdělit na základě předešlé znalosti ground truth labelů, na supervizovanou a nesupervizovanou. Supervizovaná selekce příznaků se dále rozděluje na filtrovací (filter), obalovací (wrapper) a embedded metody.

Nevýhoda nesupervizované selekece spočívá právě ve zvoleném přístupu, jelikož je obtížné ověřit relevantnost vybraných příznaků [99]. Tato práce se nesupervizovanou selekcí příznaků zabývat nebude.

4.5.1.1 Filtrovací metody

Filtrovací metody jsou nezávislé na použitém klasifikačním algoritmu strojového učení, tudíž zaujatost, bias algoritmu strojového učení pro klasifikaci neovlivňuje zaujatost algoritmu pro selekci příznaků [100]. Algoritmus strojového učení pro klasifikaci vychází z již redukovaných příznaků. Viz obrázek 4.12.



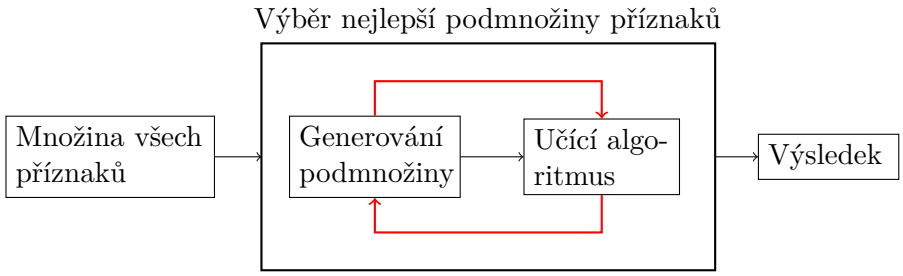
Obrázek 4.12: Proces klasifikace při výběru příznaků pomocí filtrovací metody.

Filtrovací metody pro selekci příznaků jsou založeny na měření statistických vlastností, které charakterizují vztahy mezi vstupními daty a jejich

ground truth labely, jako vzdálenost, soudržnost, závislost, vzájemnou informaci [101] nebo korelaci. Nejznámějšími zástupci této třídy jsou Fisher score, Relief, Correlation-based Feature Selection (CFS) nebo většina algoritmů založených na informačním zisku [99].

4.5.1.2 Wrapper metody

Wrapper metoda hodnotí výsledky vybraných příznaků na základě použití specifického algoritmu strojového učení, který obalují a interagují s ním. Wrapper metody postupně předkládají algoritmu strojového učení vybrané kombinace příznaků, viz obrázek 4.13. Na konci algoritmus vrátí set příznaků s nímž dosáhnul největší úspěšnosti predikce nových dat.



Obrázek 4.13: Proces klasifikace při výběru příznaků pomocí wrapper metody.

Obecná wrapper metoda, která používá k interakci klasifikační algoritmus strojového učení, lze popsat třemi kroky.

1. Výběr podmnožiny příznaků (Feature Search): Postup výběru podmnožiny z celkové množiny všech příznaků.

2. Hodnocení vybraných příznaků (Feature Evaluation): Ohodnocení výsledku klasifikačního algoritmu strojového učení na dané podmnožině příznaků.

3. Iterace 1. a 2. kroku dokud není splněna ukončovací podmínka.

Nejčastější ukončovací podmínka je ukončení výběru podmnožiny příznaků po dosažení zvoleného počtu iterací nebo pokud byly vyzkoušeny všechny možné kombinace příznaků.

Algoritmus výběru podmnožin příznaků může být i omezen na to, kolik příznaků může podmnožina obsahovat maximálně a minimálně.

Nevýhoda wrapper metod je jejich časová náročnost pro data s velkým množstvím příznaků. Velikost prohledávaného prostoru s M příznaky je $\mathcal{O}(2^M)$, problém nalezení optimální podmnožiny se řadí mezi NP-těžké problémy [102].

Proto se pro vyřešení tohoto problému používají algoritmy jako genetické algoritmy a nebo první nejlepší (Best-First) mezi kterými je nejznámější algoritmus hladového hledání (Gready Search). Tyto algoritmy dosahují pouze suboptimálního řešení.

Výhoda wrapper metody spočívá v nalezení suboptimální podmnožiny příznaků ze všech vyzkoušených podmnožin pro zvolený algoritmus strojového učení.

Příklady Wrapper metod využívající gready search jsou dopředný výběr (Forward Feature Selection) a zpětná eliminace (Backward Feature Elimination).

4.5.1.3 Embedded metoda

Embedded metoda kombinuje výhody obou výše zmíněných metod, filtrovací a wrapper. V první řadě využívá, stejně jako filtrovací metody, statistických měření k vybrání podmnožiny příznaků na základě nějakého kritéria. V druhé řadě využívá algoritmu strojového učení k výběru podmnožiny, jež dosahuje nejlepší úspěšnosti vzhledem k použitému algoritmu. Výhoda embedded modelů je ta, že výběr nejlepších příznaků probíhá již během učící fáze algoritmu, kde penalizuje nedůležité příznaky.

Mezi embedded algoritmy pro výběr příznaků patří: regularizační LASSO (Least Absolute Selection and Shrinkage Operator), Adaptive LASSO, Bridge Regularization, Elastic Net Regularization a nebo Stability Selection.

Další algoritmus spadající do embedded modelů pro výběr příznaků spadá i samotný algoritmus strojového učení zvaný Ridge regression. Jeho výstupem jsou koeficienty pro každý vstupní příznak. Korelovanost dvou a více příznaků lze odhadnout podle absolutní hodnosti koeficientů. Příznaky s hodnotou koeficientů blízkou nule do rozhodovacího procesu moc nepřispívají [103].

4.5.2 Extrakce příznaků

Extrakce příznaků zobrazuje původní hodnoty příznaků do nového prostoru, ve kterém se minimalizují lineární závislosti mezi příznaky [104]. Jedná se o zobrazení, jež transformuje původní N-rozměrný vektor na nový M-rozměrný, kde hodnota $M < N$ s pokud možno co nejmenší ztrátou informace. Narozdíl od selekce příznaků se nezachovávají původní hodnoty příznaků a vznikají zcela nové. Nový příznak je vytvořen z množiny původních příznaků na základě jejich určité kombinace.

Mezi nejznámější zástupce pro extrakci příznaků spadají například algoritmy: Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) a nebo Independent Component Analysis (ICA).

Kapitola 5

Návrh a implementace

Tato kapitola obsahuje popis návrhu skriptů a jejich implementaci. Popis návrhu skriptů pro klasifikaci internetového provozu je rozdělen do čtyř fází, jmenovitě se jedná o fáze získání příznaků, předzpracování dat, výběr příznaků a strojové učení. Popis implementace zahrnuje informace o souborech použitých pro tvorbu datasetů, programovacím jazyku, funkcích a knihovnách, a také informace o konkrétním nastavení parametrů algoritmů strojového učení.

5.1 Volba jazyka

Pro úkol této diplomové práce byl zvolen skriptovací jazyk Python verze 3.6. Python spadá mezi jazyky, které jsou programátory a vědci nejčastěji využívány pro impelementaci algoritmů strojové učení. O tom svědčí i množství optimalizovaných knihoven pro strojové učení, mezi tyto knihovny například spadá scikit-learn [44], TensorFlow [105], Keras [106], PyOD [107] a scikit-feature [108] knihovna. Python taktéž obsahuje knihovny pro přehlednou práci s maticemi, analýzu dat a matematické knihovny, například Pandas [109], Numpy [110] a SciPy [91]. Dále obsahuje knihovny Seaborn [111] a Matplotlib [112], které slouží pro přehledné grafické zobrazení dat, grafů a výsledků.

5.2 Flow

V teoretické části diplomové práce byl pojem flow definován jako obousměrný sled po sobě jdoucích paketů, které jsou popsány stejnou pěticí údajů. Volba obousměrného flow proběhla na základě studií [113, 114, 74, 115], ve kterých je dosaženo lepších výsledků s obousměrným flow, než jednosměrným, a také fakt, že ostatní studie nejčastěji používají obousměrné flow.

V praktické části diplomové práce bude definice flow navíc obohacena o maximální časový limit trvání flow, tzv. timeout. Flow na internetu může být komunikace krátká v rámci několika sekund až minut, nebo naopak několikaho-

dinová až vícedenní. K omezení takto dlouhých flow, je celý obousměrný provoz rozdělen po maximální délce trvání a to po 300 sekundách. Tím se docílí, že každá i velmi dlouhá komunikace bude moci být zpracována a klasifikována i před jejím ukončením, ale zároveň se o komunikaci nasbírá dostatek informací pro její klasifikování.

Jelikož je i v dnešní době stále nezodpovězená otázka, kolik paketů je potřeba pro správné ohodnocení flow, popřípadě jaký je dostačující časový parametr, byl časový limit 300 sekund vybrán na základě získaných statistických údajů ze zvolených datasetů a výsledků jiných prací, viz následující podkapitola.

Dále v praktické části diplomové práce bude pod pojmem flow myšleno flow s timeoutem 300 sekund. Flow bez časového omezení bude nazýváno full-flow.

5.2.1 Flow timeout v jiných studiích

Ve studii [116] je rozebrán real-time prototyp klasifikačního algoritmu, který pro klasifikaci používá pouze několik málo prvních paketů flow s volitelnou maxinální délkou trvání flow. Z práce je patrné, že klasifikace flow dosahuje velmi dobrých výsledků již u odchycení prvních 25 paketů. Výsledky ukazují, že od množství 25 odchycených paketů úspěšnost klasifikátoru nijak zásadně neroste.

Další studie [117] ukazuje flow-based klasifikaci internetového provozu, kde je flow ořezáno na základě délky trvání aktivního a neaktivního timeoutu. U aktivního timeoutu se jedná o čas 300 sekund u neaktivního timeoutu je doba zkrácená na 60 sekund. S těmito parametry pro flow dosahují velmi dobrých výsledků. V práci je uvedena průměrná doba flow na testované univerzitní sítí, od $5,4$ sekund do $13,09$ sekund.

Zkoumání průměrné délky trvání flow na jiné univerzitní síti lze také nalézt ve studii [118]. Průměrné délky trvání flow jsou následující: pro web 13,32 sekund, pro P2P spojení 123,54 sekund, pro Gnutella je to 89,35 sekund a pro BitTorrent 135,43 sekund.

Pouze pět prvních paketů bylo využito pro klasifikaci internetového flow ve studii [119]. Studie využila pro klasifikaci příznaky z obousměrného flow a nesupervizovaný shlukovací algoritmus. S tímto přístupem se dosáhlo 90 % přesnosti klasifikace pomocí metriky accuracy.

Ve výzkumu [45] se autoři zaobírají i úspěšností klasifikátorů, kterým je na vstupu předloženo flow s omezeným počtem paketů. Jejich výsledky na 16 algoritmech strojového učení ukazují, že k získání lepšího nebo alespoň podobného výsledku (74,81 % až 99,94 % accuracy) klasifikace, co s původním celým flow, lze dosáhnout již při čtyřech až osmi paketech podle typu algoritmu strojového učení. V 11 případech se dokonce dosáhlo lepších výsledků, než při použití příznaků získaných z celého flow.

5.3 Vybrané soubory pro tvorbu datasetů

Pro klasifikaci internetového provozu byly zvoleny PCAP (Packet Capture) soubory s nešifrovanými uživatelskými daty. Zvolené PCAP soubory slouží k tvorbě datasetů a nešifrovaná uživatelská data slouží k získání ground truth labelů, více o získání příznaků a ground truth labelů v sekci 5.4.

Set vybraných PCAP souborů obsahuje zachycený internetový provoz na LAN síti i páteřní síti s různou síťovou topologií.

Souhrným slovem dataset vědci občas označují i sadu několika souborů, které obsahují: celý zachycený internetový provoz v PCAP souboru, dataset získaný ze zachyceného provozu, ground truth labely, soubor s informacemi o případných útocích, typem útoku, délkou trvání útoku a infikovanými IP adresami.

5.3.1 Intrusion Detection Evaluation Dataset CICIDS2017

Dataset CICIDS2017 označuje sadu několika souborů, mezi kterými je i zachycený internetový provoz v PCAP souboru. CICIDS2017 je oproti jeho předchůdcům považován za spolehlivý dataset používaný vědci pro porovnávání výsledků algoritmů. Splňuje řadu kritérií, která jsou kladena na datasety. Mezi kritéria patří pokrytí síťové topologie, která obsahuje síťové prvky jako modemy, přepínače, směrovače, firewally a zastoupení několika operačních systémů jako Windows, Ubuntu a Mac OS X. Dalším kritériem je zachycení kompletního provozu obsahujícího lokální internetovou komunikaci i komunikaci mimo síť s Internetem a řadu použitých protokolů. Více o kritériích v [120].

Dataset byl vytvořen s ohledem na realistický background provoz a chování uživatele. Toho je docíleno programem, který napodobuje chování 25 lidí a jejich interakci na Internetu.

PCAP soubor obsahuje komunikaci z pěti dnů, od pondělí do pátku. První den, pondělí, je internetový provoz bez útoků. Každý další den jsou prováděny některé z nejznámějších a nejnovějších internetových útoků jako Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet a DDoS. Pro více informací o generované internetové komunikaci a útocích zde [121].

Dataset obsahuje i soubor s extrahovanými příznaky pomocí programu CICFlowMeter [122] spolu s grouth truth labely, zda se jedná o normální provoz či útok.

5.3.2 ČVUT datasety

Z ČVUT datasetů byly vybrány pouze datasety s označením Mixed. Tyto datasety [123] obsahují zachycenou internetovou komunikaci v PCAP souborech, která obvykle začíná bez škodlivé činnosti a až následně jsou prováděny

některé útoky. V datasetech lze nalézt následující útoky: DarkVNC, Simda Botnet, Emotet malware.

Bylo vybráno následujících 5 datasetů z roku 2018:

- CTU-Mixed-Capture-6, délka trvání přibližně 42 hodin,
- CTU-Mixed-Capture-7, délka trvání přibližně 7 hodin,
- CTU-Mixed-Capture-8, délka trvání přibližně 40 hodin,
- CTU-Mixed-Capture-9-1, délka trvání přibližně 6 dní,
- CTU-Mixed-Capture-9-2, délka trvání přibližně 6 hodin.

5.3.3 WIDE project

The MAWI Working Group [124] zprostředkovává internetový provoz odchycený na 1 Gbps WIDE (Widely Integrated Distributed Environment) páteřní síti pro vědecké účely. V této diplomové práci byly využity vzorky z WIDE sítě z dubna roku 2019. Vzorky jsou pořizovány každý den od 14:00 do 14:15 místního času a následně anonymizovány tak, aby mohly být poskytnuty veřejně. The MAWI Working Group navíc na odchycený provoz spouští jejich detekční algoritmus MAWILab [125] a poskytuje jeho výsledky, mezi které patří i soubory obsahující informace o škodlivém a podezřelém provozu.

5.3.4 Statistické vlastnosti vybraných datasetů

Na jednotlivých datasetech byl proveden pokus ohledně délky trvání full-flow, medián trvání full-flow a 75. percentil, neboli horní kvartil, medián délky trvání u komunikace, která obsahuje alespoň 25 paketů, a u komunikace, která obsahuje alespoň více jak jednu transakci.

Dále budou jednotlivé datasety označovány především zkráceným jménem. Jejich přehled je v tabulce 5.1, která zachycuje jejich původní označení a nové zkrácené.

Tabulka 5.1: Přehled zkrácených jmen datasetů

Původní celé jméno datasetu	Zkrácené jméno
CICIDS2017	IDS17
CTU-Mixed-Capture-6	CTU06
CTU-Mixed-Capture-7	CTU07
CTU-Mixed-Capture-8	CTU08
CTU-Mixed-Capture-9-1	CTU91
CTU-Mixed-Capture-9-2	CTU92
WIDE 10. 4. 2019 14:00	WIDE

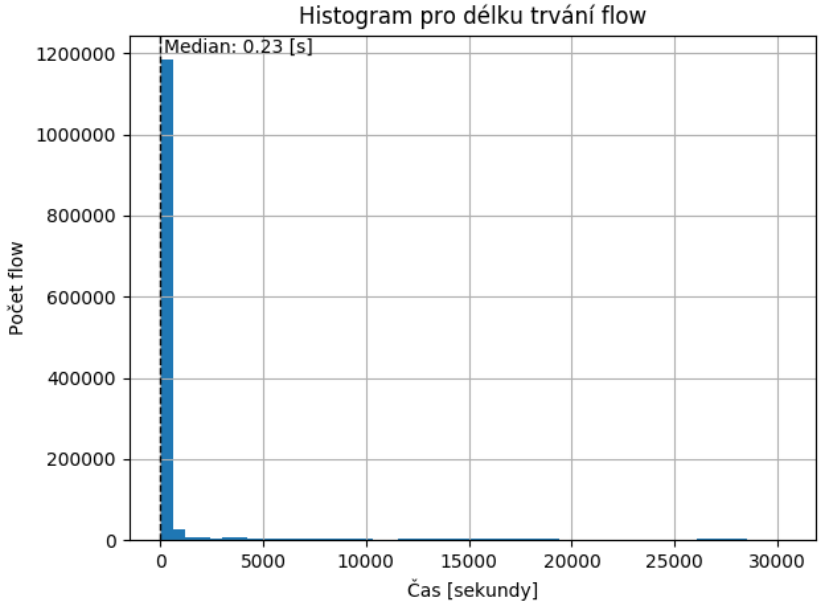
5.3.4.1 Dataset CICIDS2017

Tabulka 5.2 obsahuje přehled z pohledu počtu full-flow v datasetu CICIDS2017 a procentuálního vyjádření počtu full-flow vzhledem k celkovému počtu.

Tabulka 5.2: Početní a procentuální zastoupení full-flow v datasetu CICIDS2017

Full-flow	Celkem	Anomálie	Více jak 1 transakce	Alespoň 25 paketů	Alespoň 25 odchozích paketů	Alespoň 25 příchozích paketů
Počet	1.343.319	69.716	543.937	303.474	131.969	119.908
%	100	5,18	40,49	22,5	9,82	8,93

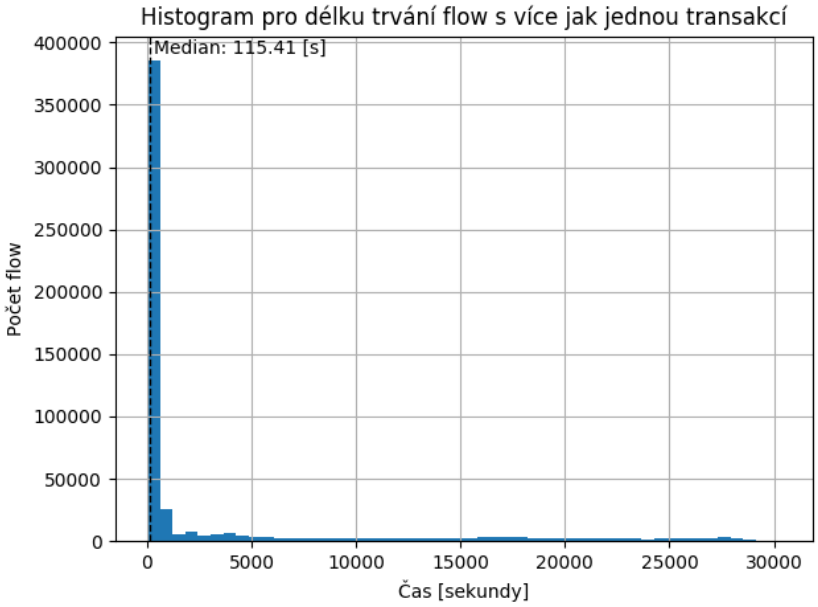
Medián délky trvání full-flow ze všech dnů je $0,23$ sekund, (horní kvartil je $50,82$ sekund), viz graf 5.1. Pokud se vezmou v potaz pouze full-flow, která se sestávají alespoň z více jak jedné transakce, vzroste medián na $115,41$ sekund (horní kvartil na $1065,42$ sekund), viz graf 5.2.



Obrázek 5.1: Histogram pro délku trvání full-flow pro CICIDS2017 dataset

Full-flow, která obsahují alespoň 25 paketů mají průměrnou délku trvání $115,44$ sekund, viz graf 5.3. Horní kvartil činní $180,42$ sekund.

Full-flow obsahující alespoň 25 a více odchozích paketů mají medián délky trvání $121,03$ sekund (horní kvartil je $690,6$ sekund). V podobném měřítku



Obrázek 5.2: Histogram pro délku trvání full-flow s více jak jednou transakcí pro CICIDS2017 dataset

to platí i pro full-flow obsahující alespoň 25 příchozích paketů, medián délky trvání je 116, 31 sekund (horní kvartil je 274, 14 sekund).

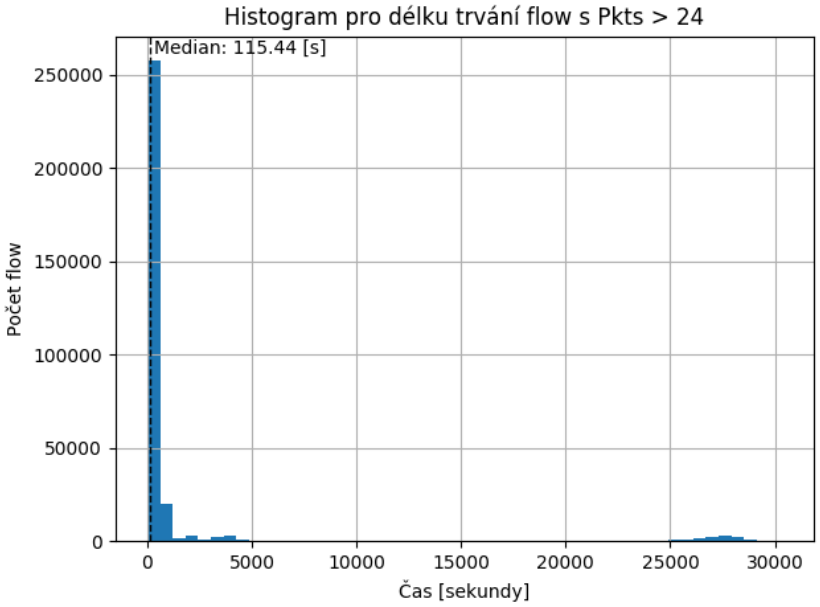
5.3.4.2 Ostatní datasety

Stejná měření jako u datasetu CICIDS2017 byla provedena i na zbylé datasety, pro které bude uvedena pouze tabulka 5.3.

První řádka konkrétního měření obsahuje jedno až dvě čísla udávající délku trvání flow. Obsahuje-li dvě čísla, pak první číslo značí medián délky trvání flow v sekundách, číslo uvedené v závorkách značí hodnotu horního kvartilu. U hodnot, které jsou příliš vysoké nebo nízké se horní kvartil neuvádí, jelikož timeout byl vybírán z rozmezí hodnot od 100 až 600 sekund. Druhá řádka obsahující procenta značí procentuální množství dat splňující podmínku, ze kterých je získaná daná statistická hodnota.

Z tabulky 5.3 je patrné, že údaje získané z datasetů CTU07 a CTU08 mají oproti ostatním datasetům vysoké hodnoty. Způsobeno je to hojným zastoupením dlouhodobé UDP komunikace. V datasetu CTU07 je poměr 34:27 UDP k TCP. U CTU08 je to skoro 1:1, přesněji 77:73 pro UDP:TCP.

Naopak WIDE dataset je tvořen krátkými komunikacemi. Vybraný WIDE dataset obsahuje 5 krát více TCP komunikace než UDP. Navíc odchycená komunikace je pouze 15 minut dlouhá, proto z údajů nelze získat žádnou delší flow než 15 minut. Výsledná data o trvání full-flow jsou tímto faktem ovlivněna.



Obrázek 5.3: Histogram pro délku trvání full-flow s více jak 25 pakety pro CICIDS2017 dataset

Ve většině prací založených na datasetech z univerzitního prostředí je uvedeno, že pro klasifikaci flow se bere v potaz i denní doba a zda se jedná o pracovní den v týdnu nebo den o víkendu. Jelikož se síť chová v každé denní době odlišně a podle toho i následně některé příznaky nemají směrodatnou vlastnost. Například počet paketů nebo bajtů za sekundu se značně liší během noční doby a denní doby [126]. V této práci se denní doba datasetu řešit nebude, jelikož polovina testovacích datasetů obsahuje pouze krátkodobou komunikaci.

5.4 Získání příznaků a tvorba datasetů

Tato sekce podrobněji popisuje postup získání příznaků z PCAP souborů spolu s ground truth labely a obsahuje přehled všech získaných příznaků.

Práce se zaměřuje na klasifikaci flow, které využívají na transportní vrstvě protokol UDP nebo TCP.

Pojem příznak reprezentuje hodnotu získanou z hlavičky na transportní vrstvě nebo statistickou hodnotu vypovídající o vlastnosti flow z informací ze síťové vrstvy. Každá flow je charakterizována stejnou množinou příznaků. Tyto příznaky jsou pak po předzpracování a možné selekci použity jako vstupní data do algoritmu strojového učení, na základě kterých je internetový provoz klasifikován a některé záznamy určené jako anomálie.

Bylo dokázáno, že hlavičky paketů mohou poskytnout dostatek informací pro charakterizování síťového provozu a dokonce i jednotlivých aplikací [74,

Tabulka 5.3: Horní kvartil a medián délek trvání full-flow pro zbylé datasety.

Délka trvání full-flow		CTU92	CTU91	CTU08	CTU07	CTU06	WIDE
Počet		13565	72818	149946	61591	14303	1042223
Vše	[s]	0,35 (15,05)	0,33 (16,47)	84072,24	12291,08	2,24 (7,84)	0,001
	[%]	100	100	100	100	100	100
Trans > 1	[s]	5839,92	118388,27	123225,28	13745,79	39021,32	56,75 (144,92)
	[%]	2,52	8,96	59,15	77,93	2,22	7,42
Pkts > 24	[s]	87,37 (240,20)	235,80 (333,52)	131868,28	13412,45	6,69 (9,16)	18,45 (97,30)
	[%]	16,24	12,01	47,61	26,38	30,22	2.43%
SrcPkt > 24	[s]	112,74 (450,6)	236,16	133877,04	18901,35	14,47 (216,04)	69,27 (201,26)
	[%]	5,62	3,46	31,53	1,77	4,24	1,05
DstPkt > 24	[s]	95,71 (336,7)	283,33	133752,81	418,76	8,78 (37,88)	10,48 (44,41)
	[%]	7,55	4,64	30,41	2,12	5,44	1,23

78, 45, 67, 68, 75, 77, 80, 116, 117, 126]. Kombinace informací z hlaviček paketů a dalších údajů, převážně statistických, vede k velmi dobrému modelu pro detekování povahy provozu i bez inspekce obsahu dat. Jak už bylo zmíněno v kapitole 2.10 tento přístup se nazývá flow-based.

5.4.1 Příznaky z Argus

Kolik a jaké příznaky lze získat z flow se různí podle použitého nástroje na získání příznaků. Po přezkoumání více nástrojů (libprotoident [127], CIC-FlowMeter, nDPI [128], Argus [129]) byl nakonec vybrán Argus, který umí ze vstupního PCAP souboru vytvořit flow s 300 sekundovým časovým limitem a ke každému flow vyextrahoval příznaky.

Argus je jeden z mála nástrojů, který umožňuje uživateli si více pohrát s výstupem. Jediná nepříznivá věc, z pohledu užívání, je zastaralá online dokumentace a neaktualizované manuální stránky.

Nástroj Argus se rozděluje na serverovou a klientskou část. Serverová část má za úkol předzpracovat PCAP soubor pro další použití pomocí příkazu *argus*. Klientská část je spouštěna nad předzpracovaným souborem k získání tolika potřebných příznaků pomocí klientů, mezi které spadá například: *ra*, *rabins*, *rasort*, *racluster*, aj. Argus klienti umožňují získání příznaků z flow i uni-flow, agregovat, rozdělit transakce na základě definovaných podmínek,

řadit je či filtrovat na základě regulárních výrazů. V této diplomové práci byl použit klient *racluster*, který agreguje informace o paketech využívající TCP nebo UDP protokol do flow o nastavené délce maximálního aktivního trvání 300 sekund.

5.4.2 Získané příznaky

Online manuální stránky nástroje Argus uvádějí popis k 97 možným příznakům k extrahování, manuální stránky nástroje samotného uvádějí pouze 86 příznaků, některé z nich nejsou uvedeny v online formě manuálu. Avšak po přezkoumáním zdrojového kódu bylo zjištěno, že lze získat i více jak 127 příznaků, ne však všechny jsou pro tuto diplomovou práci potřebné, proto byly vynechány například některé identifikátory komunikace. Nakonec pro tuto diplomovou práci vznikl skript, který získává 93 příznaků. Skript pro získání příznaků společně s konfigurací Argus serveru a klienta lze nalézt na přiloženém CD.

Ze všeh 93 získaných příznaků je 15 příznaků kategoriálních. Níže je uveden výčet získaných kategoriálních příznaků a jejich Argus notace. Více detailů o hodnotách, které nabývají kategoriální příznaky v kapitole o kódování dat 5.5.3.1.

- Číslo zdrojového a cílového portu: Sport, Dport.
- Zdrojová a cílová IP adresa: SrcAddr, DstAddr.
- Směr komunikace: Dir.
- Použitý protokol: Proto.
- Použité flagy při navazování komunikace: Flgs.
- Identifikační číslo výrobce hardwaru (OUI, Organizational Unique Identifier): SrcOui, DstOui.
- Kód státu: sCo, dCo.
- Type of service: sTos, dTos.
- Stav komunikace: State.
- TCP možnosti: TcpOpt.

Zbylé příznaky jsou kvantitativní data. Přehled všech příznaků spolu s krátkým popisem lze nalézt v tabulkách 5.4, 5.5 a 5.6.

5.4.3 Ground truth labely pro klasifikaci

Pro získání ground truth labelů pro asociování flow ke konkrétní aplikaci byly použity dvě knihovny nDPI a libprotoident. První zmíněná knihovna, nDPI, byla použita jako hlavní zdroj pro získání ground truth labelů. A jelikož libprotoident knihovna umí úspěšněji klasifikovat P2P komunikaci a Real-Time Messaging Protocol (RTMP), plnila funkci kontroly získaných labelů. Podle studií nDPI nejčastěji tyto dva případy mylně klasifikuje jako Skype nebo neznámé (Unknown) [130].

Open-source knihovna nDPI je založená na libcap a OpenDPI. Knihovna je stále aktualizována tak, aby držela krok s přibývajícími aplikacemi a protokoly. nDPI dopadla spolu s libprotoident jako nejlepší nekomerční nástroj pro klasifikaci internetového provozu [131].

Výhoda nDPI oproti libprotoident je získání dvou typů labelů, jeden label udává aplikační level klasifikace a druhý popisuje všeobecnější typ provozu, jako například video streaming, chat, P2P torrent, email client a podobně.

5.4.4 Ground truth labely pro detekci anomálií

Ground truth labely pro detekci anomálií vycházejí pro každý dataset z jiného zdroje.

CICIDS2017 dataset obsahuje soubor s příznaky z CICFLOWmetru. Tyto příznaky jsou obohaceny o sloupec se jménem *Label* udávající, které flow je útok a které je označeno jako běžný provoz.

WIDE dataset neobsahuje soubor o anomáliích přímo, ale lze ho nalézt na stránkách MAWILab, kde jsou nalezené anomálie a podezřelé komunikace z příslušného vzorku internetového provozu přehledně sepsány do souboru v csv formátu.

ČVUT datasety obsahují výstup z open-source softwaru pro detekci hrozeb na internetu, Suricaty. Výstupem je fast.log soubor, který obsahuje jednořádkové záznamy o podezřelé komunikaci.

5.5 Implementace předzpracování dat

Výsledkem implementace předzpracování dat je skript, který umožňuje individuální přizpůsobení jednotlivých kroků této fáze buďto pomocí vstupních parametrů a nebo pomocí konfiguračního souboru, který rovněž obsahuje defaultní nastavení některých parametrů.

Získané statistické informace o jednotlivých sloupcích podle typu dat jako počet chybějících údajů, průměrná hodnota sloupce, maximální a minimální hodnota sloupce, kvartily a nejčastěji se vyskytující hodnota pomohly k zjištění, které sloupce nejčastěji obsahují chybějící data a odlehlé hodnoty. Na základě toho byl zpětně prozkoumán program Argus, což vedlo k lepšímu pochopení datasetu a využití těchto znalostí k vylepšení některých kroků. Postup získání

základních statistických informací o datasetu byl rovněž implementován do skriptu.

5.5.1 Získání statistických informací o datasetu

Uživatel má možnost si vypsat statistické informace o datasetu a podle získaných informací se může rozhodnout, které kroky a fáze předzpracování dat chce aplikovat na dataset. Po výpisu informací je skript ukončen a nepokračuje dalšími fázemi.

Všeobecné informace o datasetu:

- Velikost datasetu, přesněji počet řádků a sloupců,
- jména prázdných sloupců,
- jména sloupců s chybějícími hodnotami, společně s početním a procentuálním vyjádřením.

Informace o sloupci obsahující kategoriální data, kategoriální sloupce jsou definované přes konfigurační soubor:

- Počet hodnot (nechybějících),
- počet unikátních hodnot,
- nejčastější hodnota,
- počet výskytů nejčastější hodnoty.

Informace o sloupci obsahující kvantitativní data:

- Počet hodnot (nechybějících),
- průměrnou hodnotu sloupce,
- směrodatnou odchylku sloupce,
- nejnižší hodnotu ve sloupci,
- dolní kvartil,
- medián hodnot,
- horní kvartil,
- nejvyšší hodnota ve sloupci,
- šikmost dat.

5.5.2 Implementace čištění dat

Důvod, proč Argus některé hodnoty nevrací, se nepodařilo zjistit. Avšak většina chybějících hodnot byla vyřešena a jejich doplnění bylo implementováno ve skriptu, taktéž byl odhalen šum. Níže jsou vypsané kroky, které byly implementovány a provádí se při fázi čištění dat.

5.5.2.1 Odstranění řádků bez ground truth labelů

První čištění datasetu je odstranění řádků, které neobsahují ground truth labely pro trénovací dataset. Tvorba modelu pro klasifikace provozu je závislá na ground truth labelech, proto se vyžadují.

5.5.2.2 Odstranění sloupců s málo daty

Odstranění sloupců, které obsahují málo dat či jsou úplně prázdné. Skript umožňuje volbu hranice, která označuje minimální množství dat potřebných k uchování sloupce. Hranici se říká *treshold* a je udávána pomocí desetinného čísla v rozmezí od $0,0$ do $1,0$. V konfiguračním souboru byla defaultní hodnota zvolena na 0,2, neboli 20 %, pro uchování co nejvíce cenných informací.

5.5.2.3 Vypořádání se se šumem

Při kontrole výsledků Argus bylo nalezeno pouze několik málo chybných položek v datasetu. Mezi takové patří například hodnota nekonečna v příznacích STcpMax a DTcpMax, výskyt záporné hodnoty v čistě kladných sloupcích či nesprávný datový typ.

Ve skriptech byly implementovány následující kontroly šumu:

- Kontrola na hodnotu nekonečna. Hodnota nekonečna je nahrazena hodnotou NaN (Not a Number), která značí chybějící hodnotu.
- Kontrola na výskyt záporných hodnot v čistě kladných příznacích, definovaných přes konfigurační soubor. Záporná hodnota je nahrazena NaN hodnotou.
- Kontrola správného datového typu příznaků. Pokud data v číselných sloupcích nelze převést na číselnou hodnotu, jsou nahrazena NaN hodnotou.
- Kontrola čísel portů, zda spadají do pevně definovaného rozmezí, pokud nespadají, jsou řádky s nekonzistentní hodnotou čísla portu smazány.
- Kontrola použitého protokolu. Mezi šum se řadí jakýkoliv jiný použitý transakční protokol než TCP nebo UDP. Řádky obsahující jiný protokol jsou odstraněny.

5.5.2.4 Vypořádání se s odlehlými hodnotami

Odlehlé hodnoty mohou být klíčovými informacemi pro identifikování konkrétní služby nebo i útoku, anomálie, a proto se odlehlé hodnoty nikterak neupravují. Sloupce, jež obsahují odlehlé hodnoty, se vyskytují ve všech použitých datasetech a proto je nanejvýše pravděpodobné, že se jedná o jev simulující reálné prostředí.

5.5.2.5 Vypořádání se s chybějícími daty

Chybějící hodnoty mohou být doplněny dvěma způsoby, pomocí algoritmu strojového učení k-nejbližších sousedů a nebo z vlastní implementované metody pro doplnění dat, pojmenované jako - *fill*. Obě metody doplňují chybějící hodnoty na základě protokolu. Při doplňování TCP hodnot se neberou vůbec v potaz UDP řádky a opačně.

Algoritmus k-nejbližších sousedů doplňuje chybějící kvantitativní hodnotu pomocí průměru z 5 nejbližších sousedů. Chybějící kategoriální data jsou doplněna hodnotou modusu.

Vlastní metoda je založena na získaných informací z programu Argus. Při zkoumání programu Argus bylo nalezeno pár indicií, které pomohly vyřešit většinu chybějících dat. Prvně budou rozebrána řešení pro doplnění kategoriálních dat a následně zbylá řešení.

Příznak pro TCP spojení TcpOpt: Sloupec popisující TCP vlastnosti při inicializaci spojení (TcpOpt) neobsahuje hodnoty na řádcích spadající pod UDP komunikaci. Jedná se o hodnoty, které je možno získat pouze z TCP spojení a proto je logické, že u všech UDP spojení hodnoty chybí. Chybějící hodnoty jsou při doplňování nahrazeny řetězcem -1 pro UDP protokol a řetězcem -2 pro TCP protokol, díky tomu jsou chybějící hodnoty v kroku kódování kategoriálních dat převedeny na unikátní čísla.

Identifikátor komunikace IP adresa: Chybějící hodnoty u IP adresy zdroje a cíle jsou data, která se nedoplňují. Hlavní důvody jsou dva. Za prvé jejich absence je nepravděpodobná a za druhé se jedná o sloupce, které jsou smazány na konci fáze čištění dat. Smazání sloupců obsahující IP adresy je opodstatněné faktem, že IP adresa je identifikátor a při tvorbě rozhodovacího modelu by akorát uškodil než pomohl. Sloupce, které se mají smazat na konci fáze čištění jsou definovány v konfiguračním souboru. Defaultně jsou zde nastaveny příznaky, jejichž hodnoty jsou identifikátory konkrétní komunikace či komunikačního uzlu.

Porty a protokol: Řádek neobsahující hodnotu zdrojového nebo cílového portu či použitý protokol je smazán. Číslo portu a použitý protokol jsou jedny z hlavních prvků, na kterých stojí rozhodovací algoritmus a proto při

náhodném doplnění by nesprávně určená hodnota mohla mít negativní dopad na rozhodovací model. V tabulce 5.7 lze vidět, že čísla portů a použitý protokol se umístili do první pětice nejdůležitějších příznaků podle výsledku důležitosti příznaků rozhodovacího stromu CART pro dataset WIDE. Další specifické sloupce, při kterých se maže řádek, chybí-li v něm hodnota, jsou definovány v konfiguračním souboru programu.

Tabulka 5.7: Důležitost příznaků v algoritmu CART pro dataset WIDE.

Pořadí	**Argus jméno příznaku**	**Hodnota důležitosti příznaku (Gini importance)**
1.	Proto	0.510448350249954
2.	Sport	0.151796368839578
3.	PCRatio	0.045991471424378
4.	Dport	0.028248524544682
5.	dTtl	0.018687209371926

OUI: SrcOui a DstOui jsou sloupce obsahující identifikační číslo výrobce hardwaru pro síťovou komunikaci. Zjistilo se, že hodnoty chybí právě tam, kde se jedná o broadcast nebo multicast komunikaci. Chybějící hodnoty jsou doplněny hodnotou Unknown.

Kód státu: Chybějící hodnoty u sloupců obsahující kód státu (country code, sCo a dCo) jsou doplněny kódem ZZ, který znamená neznámý nebo nespecifikovaný stát.

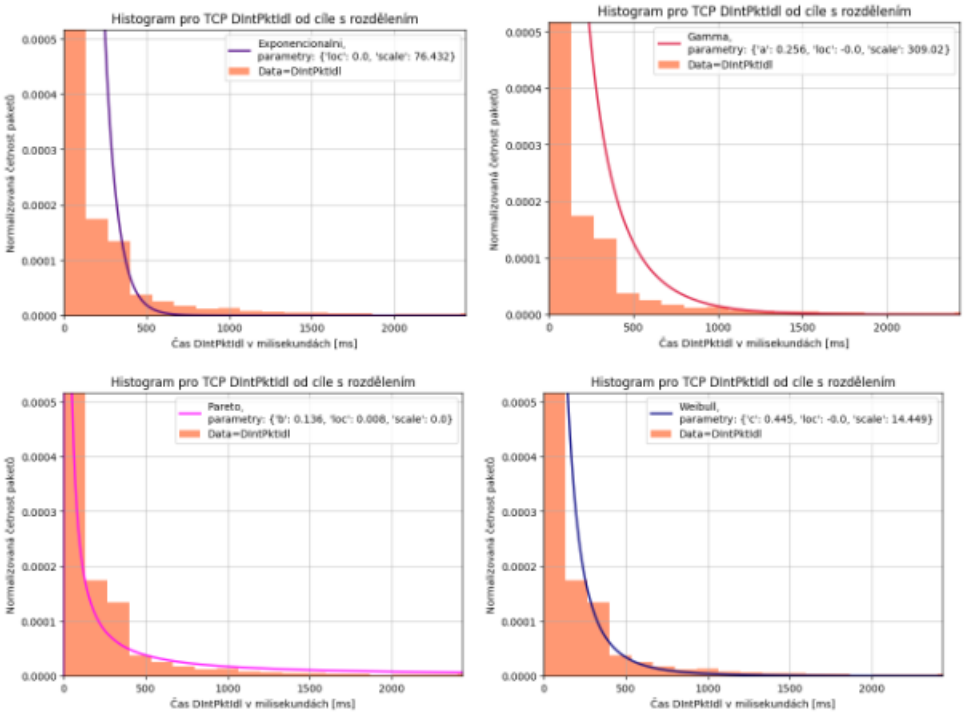
TOS: Type of Service (sTos, dTos) hodnota chybí u cíle pokud se jedná o jednosměrnou komunikaci od zdroje a naopak. Nedoplněním údaje by vznikl nový sloupec či nová hodnota při kódování daného spojení a proto se defaultně doplňuje buď modus TOS hodnot z dané IP adresy a nebo hodnota 0, pokud se žádná jiná hodnota ToS u dané IP adresy nevyskytuje.

Zbylé kategoriální příznaky: Jiné chybějící hodnoty kategoriálních dat jsou defaultně doplněny pomocí hodnoty modus, vypočtené ze všech nechybějících hodnot ve sloupci pro každý typ protokolu zvlášť.

Velikosti paketů: Největší velikost paketu ve flow chybí pouze v záznamu, kde je počet odeslaných paketů nulový, tudíž i největší a nejmenší velikost paketu je doplněna nulou. Nejmenší velikost paketů chybí i v jiných případech, chybějící hodnota je doplněna hodnotou vypočtenou z průměrné velikosti paketu a největší velikosti paketu, nebo nulou chybí-li některý z údajů.

TTL: Chybějící TTL hodnoty jsou nahrazeny jedničkou, pokud se jedná o broadcast a multicast, popřípadě nejčastější hodnotou z dané IP adresy, nebo nejčastější hodnotou mezi stejnými kódy států, v neposlední řadě číslem 64 (doporučená velikost TTL [132]) pokud není nalezena žádná z předešlých souvislostí.

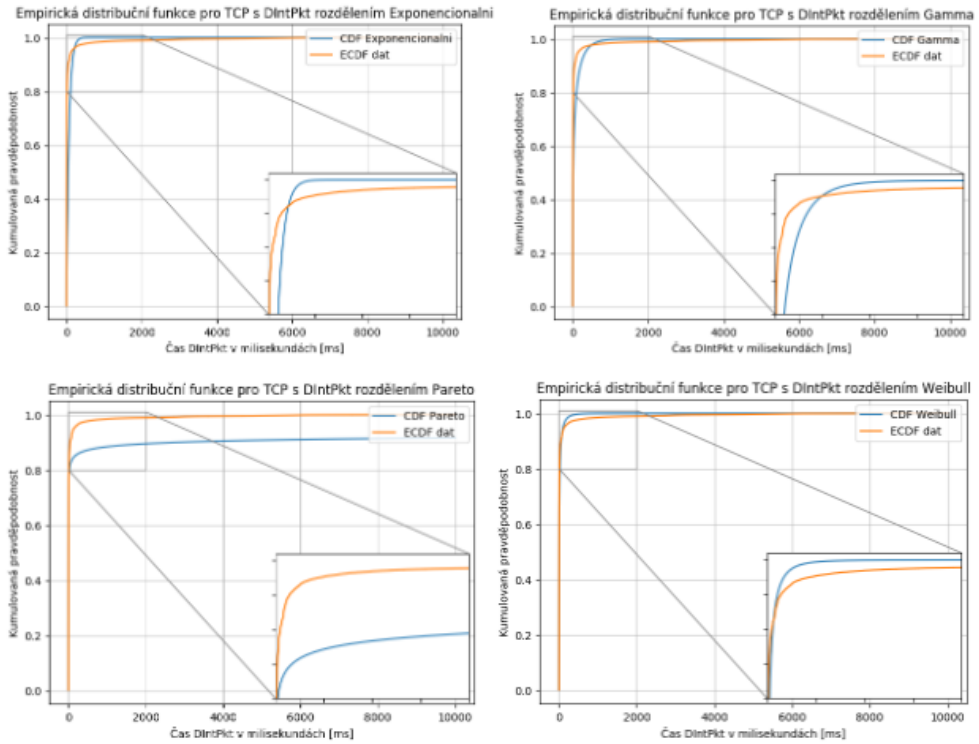
Interpacket arrival časy a jitter časy: Chybějící interpacket arrival časy a jitter časy jsou defaultně doplněny hodnotami z Weibullova rozdělení pro dané sloupce. Teoretický model počítačové sítě popisuje časy příchodů paketů pomocí Poissonova rozdělení, avšak tento model mohl být akceptovatelný pouze v původních začátcích Internetu, když se používala jedno-vrstvá architektura a pouze několik málo protokolů [133]. Při vybírání rozdělení se vycházelo z výsledků studií [134, 135, 136, 137, 138, 139, 140] a také z výsledku pokusu, během kterého se data zkoušela popsat různými rozděleními, jmenovitě se jednalo o rozdělení Weibullovo, Exponenciální, Paretovo a Gamma. Viz grafy 5.4 z pokusu.



Obrázek 5.4: Histogram DIntPktIdl a čtyři rozdělení: Exponenciální, Gamma, Paretovo a Weibullovo

Histogramy na obrázku 5.4 znázorňují distribuci interpacket arrival času DIntPktIdl pomocí různých rozdělení vzhledem k normalizované četnosti paketů. V levém horním rohu obrázku 5.4 se nachází exponenciálním rozdělením

(s parametry $loc = 0, scale = 76,432$), vedle Gamma rozdělení (s parametry $a = 0,256$, $loc = 0, scale = 309,02$), pod tím zleva Paretovo rozdělení (s parametry $b = 0,136$,$loc = 0,008$, $scale = 0$) a poslední vpravo dole je Weibullovo rozdělení (s parametry $c = 0,445$, $loc = 0$, $scale = 14,449$). Pro rozdělení s danými parametry byla i vypočtena a znázorněna empirická distribuční funkce, viz další grafy 5.5. Oranžová křivka znázorňuje empirickou distribuční funkci dat a modrá křivka je distribuční funkce daného rozdělení, opět z levého horního rohu je exponenciální, vedle Gamma, pod nimi se nachází Paretova a Weibullova empirická distribuční funkce.



Obrázek 5.5: Empirická distribuční funkce DIntPktIdl a čtyři rozdělení: Exponenciální, Gamma, Paretovo a Weibullovo

Každý časový sloupec, jenž obsahuje chybějící hodnoty, je nejdříve popsán Weibullovou distribuční funkcí. Po získání parametrů, které nejlépe popisují konkrétní sloupec, jsou nahrazeny chybějící hodnoty. Parametry se nemusí počítat pokaždé, ale uchovávají se v csv souboru na stejné cestě, kde se nachází vstupní dataset. Soubor slouží k uchování vypočtených parametrů pro další použití, jako například pro doplnění chybějících dat v trénovacím datasetu. Je-li potřeba znova vypočítat parametry pro vstupní dataset, musí se vzniklý soubor nejříve ručně smazat.

5.5.2.6 Přidání sloupců

Do fáze čištění dat bylo zasazeno přidání sloupců, které agregují hodnoty podle zdrojové IP adresy. Přidává se sloupec *unique_ports* a jsou-li příznaky z Argus, tak také *avarage_packets*. Další přidaný sloupec je *DiffSrcDstStartTime*, ten obsahuje rozdíl hodnot ve sloupcích *SrcStartTime* a *DstStartTime*, jsou-li přítomné.

unique_ports: Sloupec obsahuje počet různých cílových portů, které použila zdrojová IP adresa pro komunikaci.

avarage_packets: Sloupec obsahuje průměrný počet paketů od příslušné zdrojové IP adresy.

5.5.3 Implementace transformace příznaků

Mezi implementované části transformace příznaků spadá kódování kategoriálních dat, normalizace a transformace kvantitativních dat.

5.5.3.1 Kódování kategoriálních dat

Skript nabízí různé typy kódování kategoriálních dat. Na výběr je z možností ordinal, binary a Argus.

Ordinal a binary kódování bylo vysvětleno v 4.4.1, tudíž není potřeba dalšího vysvětlení. Kódování Argus je autorova vlastní varianta kódování pro příznaky z programu Argus. Ke vzniku jiné varianty kódování vedl případ, kdy při použití OneHot kódování s plným počtem příznaků vzniklo něco málo přes 350 sloupců, jejichž celková velikost přesáhla možnost uložení matice v paměti pro další potřeby a tak program spadl na nedostatek paměti pro výpočty. První nápad při řešení tohoto problému bylo uložení zakódovaných kategoriálních dat do řídké matice (sparse matrix) a druhá možnost byla upravit podobu kódování. Zvolena byla druhá možnost. Výstupní kategorická data z Argus mají určité vlastnosti, kterých bylo využito pro jejich úspornější zakódování než pomocí OneHot.

Argus kódování: Významy všech Argus znaků a řetězců uvedených níže lze nalézt v manuálu příkazu *ra* [141].

Sloupec obsahující použitý protokol (Proto) obsahuje dvě hodnoty TCP a UDP, proto je zakódován do jednoho sloupce pomoci dummy kódování. Sloupec obsahuje jedničky na řádcích s použitým protokolem TCP a nuly jinak.

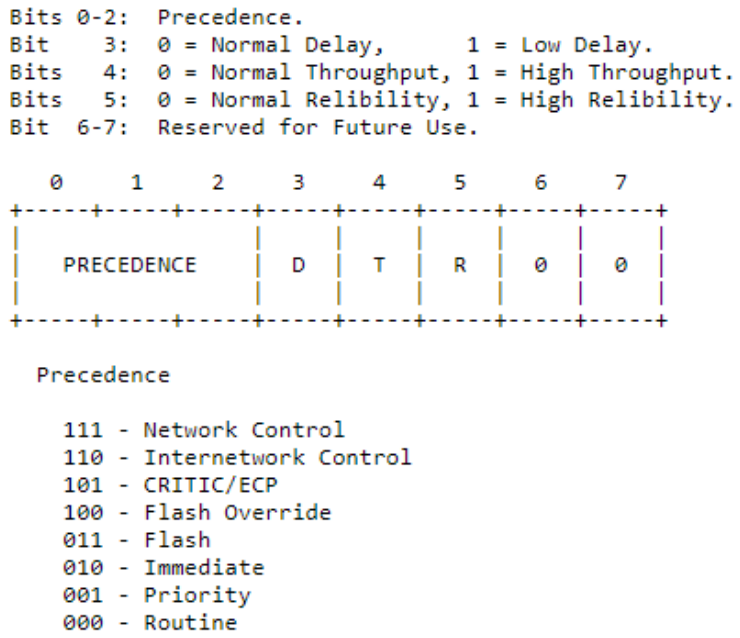
Sloupec popisující směr komunikace (Dir) může obsahovat znaky: $<$, $>$, $|$, $\circ$, $-$ a ?. Znak ? označuje nezjištěný směr komunikace. Znaky $<$ a $>$ značí směr komunikace mezi IP adresami a zbylé znaky značí stav komunikace,

viz tabulka 5.8. Pro směr komunikace je použito dummy kódování, finálně je vytvořeno pět sloupců, jeden sloupec pro každý znak <, >, |, ∘, − a znak ? je zakódován jako 0 ve všech pěti sloupcích.

Tabulka 5.8: Vysvětlení významu znaků v Dir příznaku označujícího směr komunikace

Znak	Význam
|	Transakce byla resetována
∘	Vypršel časový limit pro transakci
−	Transakce proběhla úspěšně

Sloupce obsahující TOS bajt od příjemce i odesílatele, lze zapsat jako osm bitů nesoucí informace o prioritě komunikace (první tři bity, Precedence), zpoždění (čtvrtý bit, Delay), propustnosti (pátý bit, Throughput), spolehlivosti (šestý bit, Reliability) a poslední dva bity mohou nést explicitní informaci o přetížení sítě (Explicit Congestion Notification, zkratka ENC), ale v této práci se s těmito dvěma bity nepočítá. Proto je TOS bajt rozdělen do pěti sloupců odpovídající výše zmíněnému popisu TOS bajtu: Precedence, Delay, Reliability, Throughput a poslední sloupec odpovídá všem šesti zmíněným bitům převedeným do desítkové soustavy, zvaným Differentiated services (DSCP). Více informací o významech bitů viz obrázek 5.6 a rfc791 [142].



Obrázek 5.6: Rozdělení TOS bajtu a význam jednotlivých bitů

Sloupec obsahující změnu stavu komunikace (State) obsahuje pro TCP

spojení znaky s, S, E, f, F, R a různou jejich kombinaci, pro UDP spojení řetězce CON, INT a RSP. Změna stavu komunikace je proto zakódována pomocí 6 sloupců pro každý znak v TCP spojení a dalšími 3 sloupci pro UDP spojení.

Sloupec obsahující TCP nastavení spojení (TcpOpt) má pevně definovaný formát, viz obrázek 5.7. Jedná se o 12 znaků dlouhý řetězec a na konkrétní pozici se vyskytuje pouze jeden určitý znak. Řetězec je zakódován do 13 sloupců, kde každý sloupec reprezentuje právě jedno místo v původním řetězci a poslední sloupec udává, zda hodnota chyběla v případě TCP protokolu. UDP záznamy mají ve všech nově vzniklých sloupcích nastavenou nulu.

0	1	2	3	4	5	6	7	8	9	10	11
M	w	s	S	e	E	T	c	N	O	S	D

Obrázek 5.7: TcpOpt příznak je 12-místný řetězec s pevně definovanými znaky na každém místě

Sloupec obsahující flagy komunikace (Flgs) je zakódován obdobným způsobem jako sloupec TcpOpt. Sloupec taktéž obsahuje pevně definovanou strukturu, je rozdělen do osmi místného řetězce, ale narozdíl od TcpOpt jedno místo v původním řetezci může obsahovat více odlišných znaků.

- Na první pozici se mohou vyskytovat znaky: T, N.
- Na druhé pozici znaky: *, e, E, M, m , l, v, w, p, i, G, a, p, 6, H, C, A, S, F, s, R.
- Na třetí pozici znaky: I, U, R, T.
- Na čtvrté pozici znaky: *, s, d, g, &, i, r.
- Na páté pozici: znaky: @, S, D, *, s, d.
- Na šesté pozici znaky: E, x, t.
- Na sedmé pozici znaky: V, f, F.
- Na osmé pozici znaky: O, S, L, T, +, R, A, U.

Jelikož znaky nejsou unikátní a rozlišují se podle pozice, jsou zakódovány do vlastních sloupců, sloupec nese jméno obsahující znak a jeho pozici v řetězci. Tímto způsobem vzniklo 54 sloupců, což je sice veliké číslo, ale použitím One-Hot kódování by se docílilo značně vyššího čísla.

Poslední kódovaný sloupce jsou zdrojová a cílová čísla portů (Sport, Dport), které jsou kódovány pomocí binary kódování.

5.5.4 Implementace transformace vstupních dat

Skript umožňuje provedení transformace rozdělení příznaků blíže k normálnímu nebo rovnoměrnému rozdělení pomocí metod: Uniform, YeoJohnson, BoxCox, Normal, Log, CubeRoot. Všehny transformace jsou aplikované pouze na sloupce, které mají šikmost dat větší jak $0,5$ nebo menší jak $-0,5$.

5.5.4.1 Konfigurace metod

Transofrmace se provádí pomocí funkcí z knihovny *scikit*, přičemž do funkcí vstupují pouze kvantitativní příznaky. Níže jsou uvedeny jména funkcí a nastavení jejich parametrů.

Uniform: Funkce *QuantileTransformer* s parametry $n\_quantiles = 1500$, $output_distribution = uniform$ a $copy = False$.

Normal: Funkce *QuantileTransformer* s parametry $n\_quantiles = 1500$, $output\_distribution = normal$ a $copy = False$.

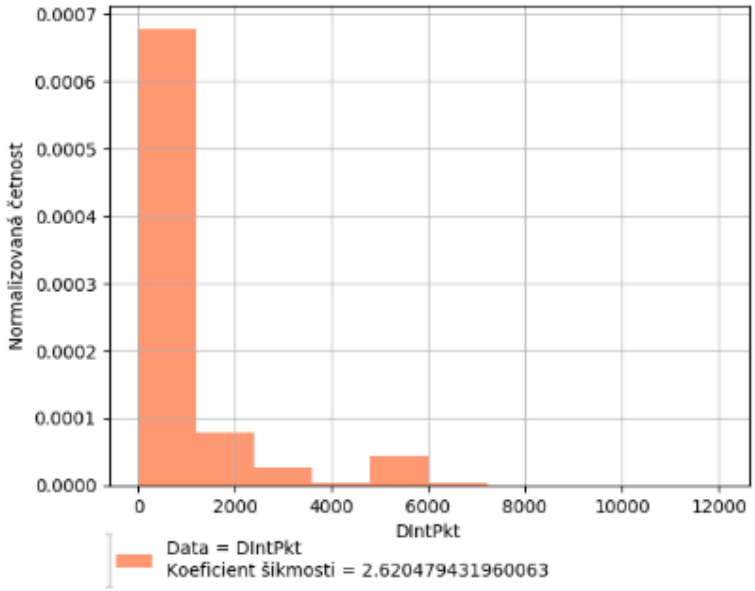
YeoJohnson: Funkce *PowerTransformer* s parametry $method = yeo\text{-}johnson$, $standardize = False$ a $copy = False$.

BoxCox: Funkce *PowerTransformer* s parametry $method = box\text{-}cox$, $standardize = False$ a $copy = False$. Tato funkce může být aplikována pouze na kladná vstupní data. Pro zaručení kladných vstupních dat pro metodu BoxCox jsou všechny data posunuta o minimální zápornou hodnotu do plusových hodnot.

Log: Funkce *FunctionTransformer* s parametry $func = numpy.log1p$ a $validate = True$. Logaritmická transformace je aplikována pouze na kladné vstupní data, zbylé sloupce obsahující záporné hodnoty jsou transformovány pomocí CubeRoot transformace.

CubeRoot: Funkce *FunctionTransformer* s parametry $func = numpy.cbrt$ a $validate = True$.

Níže je ukázka dopadu transformace na data s nízkou hodnotou šikmosti a vysokou. První graf 5.8 zobrazuje příznak DIntPkt před transformací s malou šikmostí $2,6$. Souhrný obrázek grafů 5.9 zobrazuje stejný příznak DIntPkt po transformaci BoxCox, CubeRoot, Log, Normal, Uniform a YeoJohnson. V případě Log a Normal se šikmost otočila a zvětšila na $-8,1$ pro Log transformaci a na $-4,7$ pro Normal transformaci. V ostatních případech se šikmost zmenšila, pro BoxCox na $-1,6$, CobeRoot na $0,61$, Uniform na $-1,32$ a YeoJohnson na $-1,54$.



Obrázek 5.8: Histogram pro TCP sloupec DIntPkt bez transformace

V druhém případě, kdy je koeficient šikmosti pro příznak DIntPktMax vysoký $310,8$, viz graf 5.10, si všechny transformace vedly dobře, viz souhrný obrázek grafů 5.11. Hodnoty šikmosti po transformaci jsou následující: pro BoxCox transformaci se šikmost změnila na $0,16$, pro CubeRoot na $22,45$, pro Log na $0,15$, pro Normal na $0,19$, pro Uniform na $0,18$ a pro YeoJohnson na $0,16$. U CubeRoot si lze všimnout, že koeficient šikmosti se snížil, ale stále zůstal poměrně vysoký oproti ostatním transformacím. Způsobeno je to velkým rozpětím hodnot, přičemž se ale valná většina nachází pouze v okolí jednoho místa.

5.5.5 Implementace normalizace dat

Skript umožňuje normalizovat data směrodatnou odchylkou, rozpětím dat, na maximum příznaku nebo na základě kvartilů.

5.5.5.1 Konfigurace metod

Pro implementování byly použity funkce z knihovny *scikit* s defaultním nastavením.

Normalizace směrodatnou odchylkou: Funkce *StandardScaler* s parametry $with\_mean = True$ a $with\_std = True$.

Normalizace rozpětím: Funkce *MinMaxScaler* s parametrem $feature\_range = (0, 1)$.

Normalizace na maximum příznaku: Funkce *MaxAbsScaler*.

Normalizace na základě kvartilů: Funkce *RobustScaler* s parametry $with\_centering = True$, $with\_scaling = True$ a $quantile\_range = (25.0, 75.0)$.

5.6 Implementace volby a výběru příznaků

Pro účely diplomové práce je využito několik funkcí z knihovny *scikit* pro volbu a výběr příznaků a z knihoven *skfeature* [143], *pymrmr* [144] a *mlxtend* [145]. Jedna z možností výběru příznaků je také vlastní výběr příznaků, které chce uživatel zachovat.

5.6.1 Vlastní výběr

Vlastní výběr zahrnuje výběř příznaků definovaných přes konfigurační soubor. Je-li společně s tímto parametrem zapnut i parametr *featureInfo* je uživateli zobrazeno několik grafů a vypsány informace, o tom které příznaky mají nulovou důležitost při rozhodování a které mají největší. Pro vypsání těchto informací byla využita třída *FeatureSelector* od autora Will Koehrsen [146], která byla ještě doplněna o některé funkce.

První graf zobrazuje Pearsonův korelační koeficient pro všechny dvojice příznaků, jejichž hodnota korelace přesahuje treshold $\pm 0,95$. Pro tyto příznaky platí, že je vhodné zachovat pouze jeden z nich, jelikož druhý svou hodnotou nepřidává žádnou velmi zásadní informaci. Graf 5.12 zobrazuje vysoce korelované příznaky pro dataset CTU07.

Druhý graf zobrazuje Pearsonův korelační koeficient pro všechny dvojice příznaků. Je to tabulka o rozměru $n \times n$, kde n značí počet příznaků.

Třetí graf zobrazuje třicet nejdůležitějších příznaků od nejvíce důležitého po méně důležité.

Poslední graf zobrazuje, kolik příznaků by mělo být použito, aby jejich komulativní hodnota důležitosti přesáhla hranici $0,85$. Důležitost příznaků udává klasifikační algoritmus *LGBMClassifier* z knihovny *lightgbm* [147].

Obrázky 5.13 a 5.14 znázorňují ukázku posledních dvou grafů pro dataset IDS17 při kódování Argus.

Výsledky Pearsonova korelačního koeficientu a přehled nejdůležitejších příznaků pro jednotlivé datasety lze nalézt v kapitole 6.5.

5.6.2 Selekce příznaků

Výsledný počet příznaků po selekci byl zvolen pevně na hodnotu 17 pro implementované selekční algoritmy. Toto číslo tvoří optimálního mezníka mezi úspěšností algoritmu a časem potřebným pro tvorbu modelu. Při volbě 20 a více příznaků úspěšnost algoritmů s vybranými příznaky nikterak zásadně nerostla a navíc čas potřebný pro vytvoření modelu se zvětšoval. Naopak při

použití méně jak 15 příznaků nebyl vytvořen model, který by dokázal vhodně popsat data.

Pro níže uvedené parametry se pod proměnnou n skrývá maximální počet výsledných vybraných příznaků, tedy 17.

5.6.2.1 Filtrovací metody

Mezi možné selekční filtrovací algoritmy patří *SelectKbest*, *Percentile* a *mRMR*.

SelectKbest: Funkce *SelectKbest* s parametry $k = n$ a $score\_func = mutual\_info\_classif$.

Percentile: Funkce *SelectPercentile* s parametry $percentile = 15$ a $score\_func = mutual\_info\_classif$.

mRMR: Pro minimum Redundancy Maximum Relevance (mRMR) byla použita funkce *mRMR* a parametry MIQ a počtem vybraných příznaků n. Při použití této metody je vypsáno n příznaků s největší hodnotou vzájemné informace a následně n vybraných příznaků na základě jejich hodnoty vzájemné informace a hodnoty redundance.

5.6.2.2 Wrapper metody

Mezi možné wrapper metody patří *SFS*, *SBS*, *EFS* a *RFECV*.

SBS: Pro Sequential Backward Selection (SFS) byla použita funkce *SFS* s parametry $k\_features = n$, $forward = False$, $floating = False$, $scoring = auc\_scorer$, $cv = 3$, $n\_jobs = -1$ a jako estomátor byl zvolen algoritmus *LogisticRegression* s následujícími parametry: $penalty = l1$, $solver = saga$, $multi\_class = ovr$ a $max\_iter = 2500$.

SFS: Pro Sequential Forward Selection (SFS) byla použita funkce *SFS* s parametry $k\_features = n$, $forward = True$, $floating = False$, $scoring = auc\_scorer$, $cv = 3$, $n\_jobs = -1$ a estimátor byl zvolen stejný jako v případě *SBS*.

EFS: Pro Exhaustive Feature Selector (EFS) byla použita funkce *EFS* s parametry $min\_features = 5$, $max\_features = 5$, $print\_progress = True$, $scoring = auc\_scorer$, $cv = 3$, $n\_jobs = -1$ a estimátor byl zvolen stejný jako v případě *SBS*. Při použití metody *SBS*, *SFS* nebo *EFS* je vykreslen graf znázorňující skóre AUC ROC vůči počtu příznaků. Společně s grafem jsou vypsány údaje jako nejlepší dosažené skóre, indexy a jména sloupců se kterými se dosáhlo nejlepších výsledků.

RFECV: Pro Recursive Feature Elimination with Cross-validation (RFECV) byla použita funkce *RFECV* s parametry $scoring = auc\_scorer$, $step = 3$, $min\_features\_to\_select = 5$, $cv = StratifiedKFold(3)$ a estimátor byl zvolen stejný jako v případě *SBS*. Při použití této metody je vypsáno dosažené nejlepší úspěšnosti pomocí AUC ROC a algoritmu *LogisticRegression*, optimální počet příznaků a graf znázorňující počet příznaků vůči získanému průměrnému AUC ROC z křížové validace, viz obrázek 5.15 pro dataset WIDE.

5.6.2.3 Embedded metody

Mezi možné embedded metody pro výběr příznaků spadá *LassoCV*.

LassoCV: Funkce *SelectFromModel* s parametry $max\_features = n$ a $threshold = 0.25$ a jako estimátor byl zvolen algoritmus *LassoCV* s parametry $cv = 5$, $max\_iter = 4000$. Při použití této metody jsou navíc vykresleny dva grafy s nenulovými koeficienty jednotlivých příznaků podle *LassoCV* klasifikátoru, první graf znázorňuje kladné koeficienty a druhý záporné. Navíc jsou vypsané příznaky s nulovým koeficientem a třicet příznaků s největší absolutní hodnotou koeficientu.

5.6.3 Extrakce příznaků

Ve skriptu jsou implementovány následující algoritmy: *FastICA*, *FA*, *PCA* a *LSA*. Výsledná velikost dekompozice všech příznaků je po pokusech s různými hodnotami nastavena na 6, níže označovaná jako proměnná n. U zvolených datasetů toto finální číslo příznaků vychází s nejlepšími výsledky.

FastICA: Fast Independent Component Analysis, funkce *FastICA* s parametrem $n\_components = n$.

FA: Factor Analysis, funkce *FactorAnalysis* s parametry $n\_components = n$ a $tol = 0,05$.

PCA: Principal Component Analysis, funkce *PCA* s parametry $n\_components = n$ a $svd\_solver = arpack$.

LSA: Extrakce příznaků pomocí truncated SVD, známé jako LSA. Funkce *TruncatedSVD* s parametrem $n\_components = n$.

5.7 Algoritmy strojového učení

Pro porovnání výsledků klasifikace internetového provozu a detekce anomálií bylo vybráno několik algoritmů strojového učení. Algoritmy byly vybrány

z několika knihoven, kromě *scikit* knihovny byly použity i knihovny *weka* [148] a *pyod* [107].

Mezi finální zvolené algoritmy pro klasifikaci internetového provozu spadá *AdaBoost, CART, C4.5, KMeans, kNN, NB, REPTree, RandomForest* a *SVM*. Mezi finální algoritmy se nedostaly *DBSCAN* kvůli neuspokujícím výsledkům a *XGBClassifier* z knihovny *xgboost* [149], který sice dosahoval dobrých výsledků ale v poměru časové složitosti a výsledného skóre byly upřednostněny jiné algoritmy. Algoritmy *C4.5, KMeans, kNN, NB* a *SVM* byly zvoleny a ponechány na základě jejich častého výskytu ve výzkumech zaobírající se klasifikací internetového provozu.

Algoritmy pro detekci anomálií jsou zvoleny následující: *AutoEncoder, LOF* a *ABOD*. Avšak ukázalo se, že algoritmy pro detekci anomálií nejsou nejvhodnější nástroj pro klasifikování škodlivého provozu, více informací v kapitole s výsledky 6.

Všechny tyto algoritmy prošly několika pokusy a byly vybrány takové hyperparametry algoritmů strojového učení, které vracejí nejlepší výsledky pro řešení úkolu klasifikace internetového provozu a detekci anomálií. U jednotlivých algoritmů budou uvedena i všechna nastavení hyperparametrů, která byla zkoušena, kde tučně zvýrazněná hodnota značí finálně vybranou.

Pro získání optimálních hyperparametrů algoritmů byla použita křížová validace s rozdělením vstupních dat na 5 podmnožin, též označováno jako 5-fold křížová validace.

5.7.1 Algoritmy pro klasifikaci provozu

AdaBoost: Funkce *AdaBoostClassifier* s parametry $n\_estimators = 500$, $algorithm = SAMME$, $learning\_rate = 2.5$ a $base\_estimator$ byl zvolen $DecisionTreeClassifier$ s $max\_depth = 10$. Všechny zkoušené parametry:

- N_estimators: 50, 100, 200, 300, 400, **500**, 750, 1000, 1500.
- Learning_rate: 1.0, 1.9, 2.0, 2.3, **2.5**, 2.8, 3.5, 4.0, 5.0
- Algorithm: **SAMME**, SAMME.R.
- Base_estimator: DecisionTreeClassifier s max_depth:
  - Max_depth: 5, 7, **10**, 11, 12, 13, 14, 15, 16, 18.
- Zapnutý a **vypnutý** parametr A.

C4.5: Funkce z weka knihovny *weka.classifiers.trees.J48* s parametry $C = 0,3$ a $M = 5$. Všechny zkoušené parametry:

- C: $0.2, 0.25, \mathbf{0.3}, 0.35, 0.4, 0.5$.

- M: $2, 3, 4, \mathbf{5}, 6, 7$.
- Zapnutý a **vypnutý** parametr U.
- Zapnutý a **vypnutý** parametr R.
- Zapnutý a **vypnutý** parametr A.

CART: Funkce *DecisionTreeClassifier* s parametry $criterion = gini$, $splitter = best$, $max\_depth = 42$, $min\_samples\_split = 3$, $presort = True$, $min\_samples\_leaf = 5$ a $max\_features = None$. Všechny zkoušené parametry:

- Criterion: **gini**, entropy.
- Splitter: **best**, random.
- Max_depth: $15, 20, 25, 30, 35, 40, \mathbf{42}, 43, 44, 45, 47, 48, 50, 55, 60,$ None.
- Min_samples_split: $1, 2, \mathbf{3}, 4, 5$.
- Min_samples_leaf: $\mathbf{1}, 2, 5, 10, 15$.
- Max_features: auto = sqrt, log2, **None**.
- Presort: **True**, False.

KMeans: Funkce *KMeans* s parametry $algorithm = elkan$, $n\_init = 20$, $n\_clusters = 35$, $init = k\text{-}means{+}{+}$, $max\_iter = 1000$, $tol = 0.0001$, $precompute\_distances = auto$ a $n\_jobs = -1$. Všechny zkoušené parametry:

- Algorithm: **Elkan**, full, auto.
- N_clusters: $13, 15, 16, 20, 25, 30, \mathbf{35}, 40, 45, 55, 65, 75, 85, 95, 100, 150$.
- Init: **k-means++**, random.
- N_init: $10, 20, 30$.
- Max_iter: $300, 500, 700, \mathbf{1000}$.

kNN: Funkce *KNeighborsClassifier* s parametry $weights = distance$, $p = 1$, $algorithm = kd\_tree$, $leaf\_size = 20$ a $n\_neighbors = 3$. Všechny zkoušené parametry:

- Weights: **distance**, uniform.
- Algorithm: **Kd_tree**, Ball_tree, brute.
- Leaf_size: $10, \mathbf{20}, 30$.
- N_neighbors: $\mathbf{3}, 5, 7$.
- Min_samples_leaf: $\mathbf{1}, 2, 5, 10, 15$.
- P: **1** (manhattan_distance (l1)), 2 (euclidean_distance (l2)).

NB: Funkce *GaussianNB* s parametrem $var_smoothing = 1e-02$.Všechny zkoušené parametry:

- Var_smoothing: $1e-13, 1e-12, \ldots, 1e-03, \mathbf{1e\text{-}02}, 1e-01$.

RandomForest: Funkce *RandomForestClassifier* s parametry $criterion = gini$, $n\_estimators = 500$, $min\_samples\_split = 3$, $min\_samples\_leaf = 1$, $max\_depth = 20$, $max\_features = auto$ a $bootstrap = True$, . Všechny zkoušené parametry:

- N_estimators: 100, 200, 250, 300, 400, **500**, 1000, warn.
- Max_depth: $None, 5, 10, 15, \mathbf{20}, 2530, 42$.
- Min_samples_split: $2, \mathbf{3}, 4, 5$.
- Min_samples_leaf: $\mathbf{1}, 2, 3, 4, 5, 10, 20, 25$.
- Max_features = None, **auto** = sqrt.
- Bootstrap = **True**, False.

REPTree: Funkce z weka knihovny *weka.classifiers.trees.REPTree* s parametry $N = 5$ a $M = 3$. Všechny zkoušené parametry:

- N: $3, 4, \mathbf{5}, 6$.
- M: $2, \mathbf{3}, 4, 5, 6, 7$.
- Zapnutý a **vypnutý** parametr P.

SVM: Funkce *SVC* s parametry $C = 200$, $kernel = rbf$, $gamma = 0.1$, $shrinking = True$, $decision\_function\_shape = ovr$. SVM dosáhl nejlepších výsledků s parametry $C = 1000$ a $gamma = 1$, ale jeho časová složitost byla příliš veliká, pro středně velký dataset CTU08 trvala trénovací fáze něco málo přes 7 hodin a testovací 2 hodiny, proto byly zvoleny parametry, které nedosahují až takové úspěšnosti, ale jejich časová složitost je třetinová. Všechny zkoušené parametry:

- Decision_function_shape: **ovr** (one-vs-rest), ovo (one-vs-one).
- C: $0.5, 1, 2, 20, 100, \mathbf{200}, 500, 1000$.
- Kernel: **rbf**, poly.
- Degree: $\mathbf{3}, 5, 7$.
- Gamma: $1.e-09, \ldots, \mathbf{1.e\text{-}01}, 1.e+00, 1.e+01, 1.e+02$, auto.

5.7.2 Algoritmy pro detekci anomálií

ABOD: Funkce *ABOD* z knihovny *PyOD* s parametry $method = fast$ a $contamination = 0.01$. Všechny zkoušené parametry:

- Method: **fast**, default.
- N_neighbors: $5, \mathbf{10}, 15$.
- Leaf_size: $10, \mathbf{15}, 20$.
- Contamination: $0.1, \mathbf{0.01}, 0.001$.

AutoEncoder: Funkce *AutoEncoder* z knihovny *PyOD* s parametry $output\_activation = softmax$, $loss = binary\_crossentropy$, $verbose = 2$, $preprocessing = False$, $hidden\_activation = relu$, $epochs = 10$, a $hidden\_neurons = [n, 2, 2, n]$, kde n značí počet vstupních příznaků do umělé neuronové sítě. Všechny zkoušené parametry:

- Epoch: $\mathbf{10}, 50, 100$.
- Batch_size: $\mathbf{32}, 54$.
- Output_activation: relu, sigmoid, tanh, **softmax**, elu.
- Output_activation: **relu**, sigmoid, tanh, softmax, elu.
- Loss: **binary_crossentropy**, mean_squared_error, categorical_crossentropy.

LOF: Funkce *LOF* z knihovny *PyOD* s parametry $contamination = 0.1$, $n\_neighbors = 3$, $algorithm = kd\_tree$, $leaf\_size = 15$ a $metric = l1$. Všechny zkoušené parametry:

- Algorithm: **kd_tree**, ball_tree.
- N_neighbors: **3**, 5, 20, 25.
- Leaf_size: 10, **15**, 20.
- Metric: **l1**, minkowski.

Tabulka 5.4: Přehled příznaků spolu s popisem, první část. První tři řádky představují identifikátor komunikace pro flow.

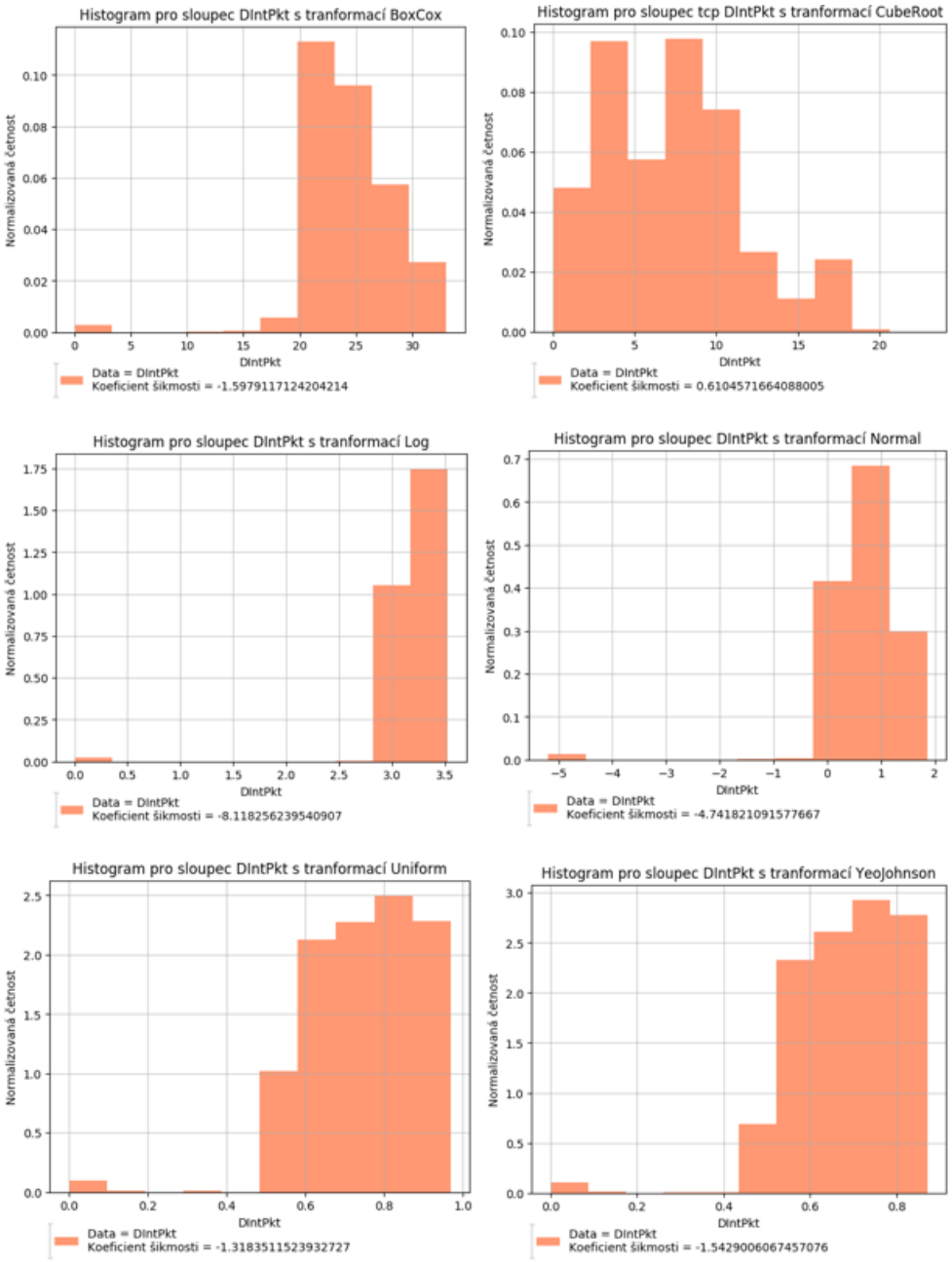
Argus jméno	**Popis**	**Typ dat**
SrcAddr / DstAddr	IP adresa zdroje / cíle.	kategoriální
Sport / Dport	Číslo portu zdroje / cíle.	kategoriální
Proto	Použitý transakční protokol pro komunikaci.	kategoriální
Dir	Směr komunikace.	kategoriální
Dur	Čas mezi prvním odeslaným paketem a posledním odeslaným paketem z flow [s].	kvantitativní
SrcDur / DstDur	Čas mezi prvním odeslaným paketem a posledním odeslaným paketem z flow ze strany zdroje / cíle [s].	kvantitativní
RelTime	Relativní odchylka časů [s].	kvantitativní
StartTime	Unixový čas poslání prvního paketu z flow [s].	kvantitativní
SrcStartTime / DstStartTime	Unixový čas poslání prvního paketu ze zdroje / z cíle ve flow [s].	kvantitativní
Trans	Celkový počet transakcí ve flow.	kvantitativní
Flgs	Použité flagy během komunikace.	kategoriální
RunTime	Celková doba aktivního přenosu všech paketů ve flow [s].	kvantitativní
IdleTime	Celková doba neaktivního přenosu všech paketů ve flow [s].	kvantitativní
Mean	Průměrná doba aktivního přenosu paketů ve flow [s].	kvantitativní
StdDev	Směrodatná odchylka doby aktivního přenosu paketů ve flow.	kvantitativní
Min	Nejkratší doba aktivního přenosu paketu ve flow [s].	kvantitativní
Max	Nejdelší doba aktivního přenosu paketu ve flow [s].	kvantitativní
SrcOui / DstOui	Zdrojové / cílové OUI.	kategoriální
sTos / dTos	Zdrojový / cílový TOS bajt.	kategoriální
sCo / dCo	Zdrojový / cílový kód státu.	kategoriální
sTtl / dTtl	Hodnota TTL (time-to-live) od zdroje k cíli / od cíle ke zdroji.	kvantitativní
SrcPkts / DstPkts	Celkový počet odeslaných paketů ze zdroje / z cíle [pkts].	kvantitativní

Tabulka 5.5: Přehled příznaků spolu s popisem, druhá část.

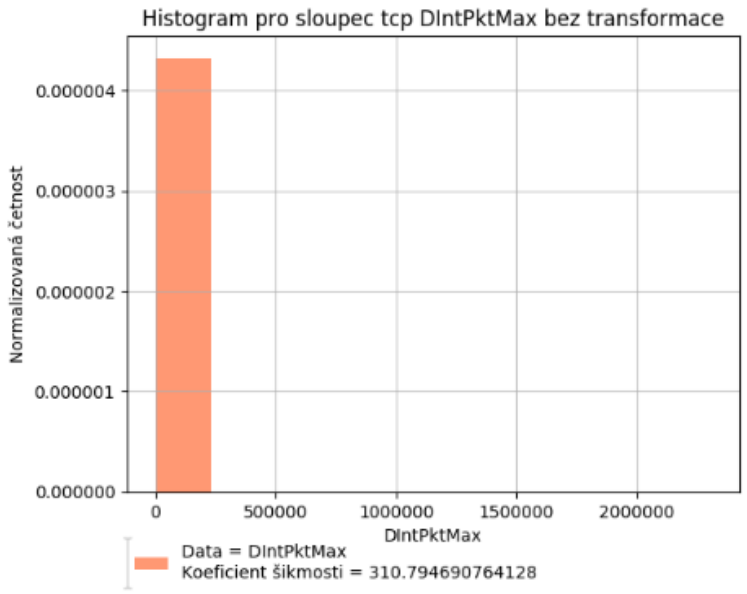
Argus jméno	**Popis**	**Typ dat**
sMaxPktSz, dMaxPktSz	Maximální velikost paketu ve flow v bajtech ze zdroje / z cíle [B].	kvantitativní
sMinPktSz, dMinPktSz	Minimální velikost paketu ve flow v bajtech ze zdroje / z cíle [B].	kvantitativní
sMeanPktSz, dMeanPktSz	Průměrná velikost paketu ve flow v bajtech ze zdroje / z cíle [B].	kvantitativní
SrcBytes / DstBytes	Celkový počet odeslaných dat ze zdroje / z cíle [B].	kvantitativní
SAppBytes / DAppBytes	Velikost všech odeslaných uživatelských dat od zdroje / z cíle [B].	kvantitativní
TF	Efektivita komunikace, poměr (SAppBytes + DAppBytes) / (SrcBytes + DstBytes).	kvantitativní
STF	Efektivita komunikace zdroje, poměr SAppBytes / SrcBytes.	kvantitativní
DTF	Efektivita komunikace cíle, poměr DAppBytes / DstBytes.	kvantitativní
PCRatio	Poměr počtu bajtů ze zdroje vůči počtu bajtů z cíle.	kvantitativní
Load	Počet bitů za sekundu ve flow [b/s].	kvantitativní
SrcLoad, DstLoad	Počet bitů za sekundu ze zdroje / z cíle [b/s].	kvantitativní
Loss	Celkový počet znovu odeslaných a zahozených paketů ve flow [pkts].	kvantitativní
SrcLoss, DstLoss	Počet znovu odeslaných a zahozených paketů se strany zdroje / cíle [pkts].	kvantitativní
pLoss	Procentuální zastoupení celkového počtu znovu odeslaných a zahozených paketů vůči všem paketům ve flow [%].	kvantitativní
pSrcLoss, pDstLoss	Procentuální zastoupení celkového počtu znovu odeslaných a zahozených paketů vůči všem paketům ve flow ze strany zdroje / cíle [%].	kvantitativní
SrcRetra, DstRetra	Počet znovu odeslaných paketů ze zdroje / z cíle [pkts].	kvantitativní
pRetran	Procentuální zastoupení celkového počtu znovu odeslaných paketů vůči všem paketům ve flow [%].	kvantitativní
pSrcRetr, pDstRetr	Procentuální zastoupení celkového počtu znovu odeslaných paketů vůči všem paketům ve flow ze zdroje / z cíle [%].	kvantitativní

Tabulka 5.6: Přehled příznaků spolu s popisem, třetí část.

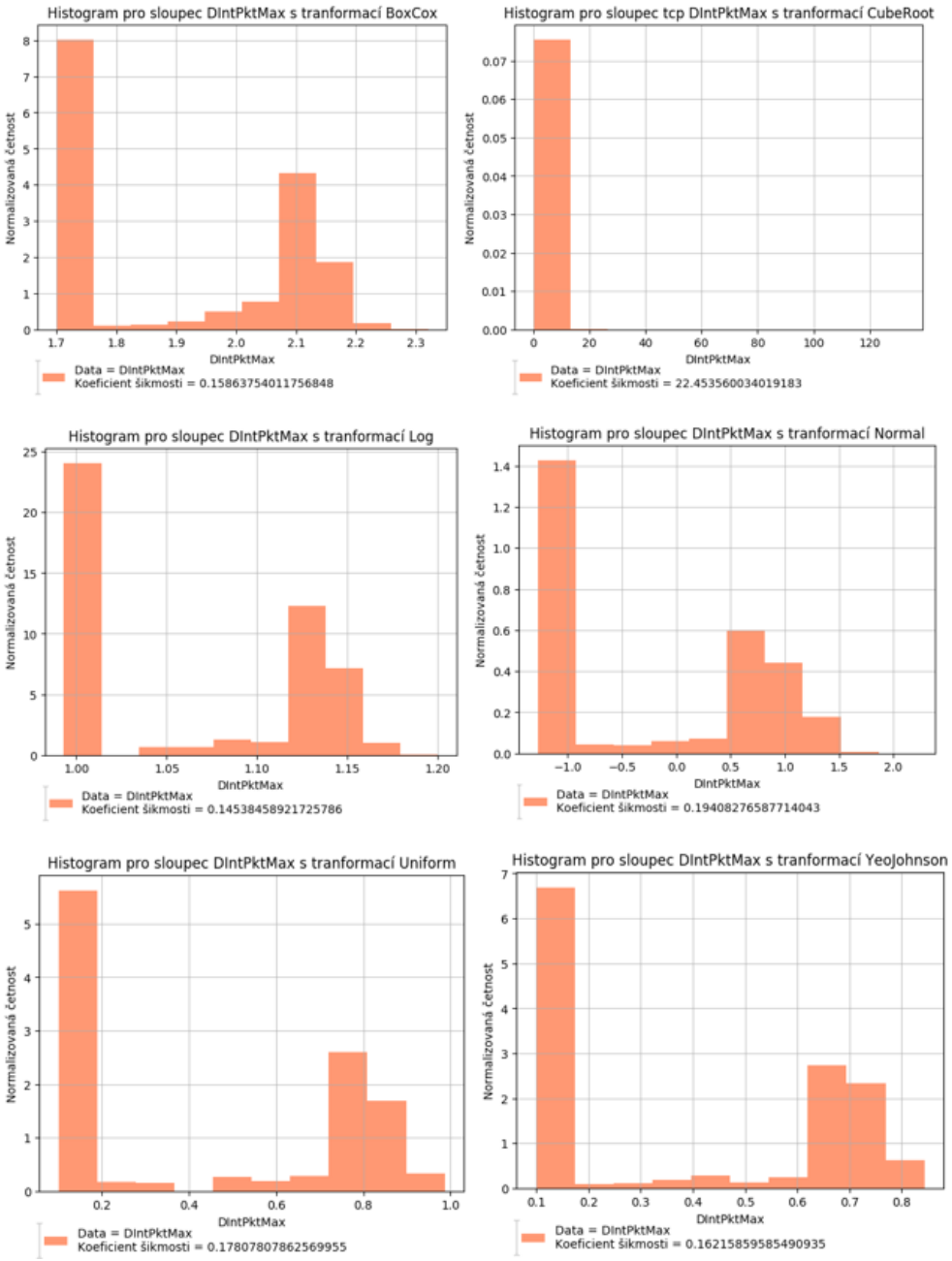
Argus jméno	**Popis**	**Typ dat**
Rate	Počet všech paketů za sekundu [pkts/s].	kvantitativní
SrcRate / DstRate	Počet paketů za sekundu putujících ze zdroje / z cíle [pkts/s].	kvantitativní
SIntPkt / DIntPkt	Čas příchodu paketů zdroje / cíle [ms]. Také uváděno pod názvy inter-arrival čas nebo v Argus jako interpacket arrival čas.	kvantitativní
SIntPktAct / DIntPktAct	Aktivního interpacket arrival čas zdroje / cíle [ms].	kvantitativní
SIntPktIdl / DIntPktIdl	Neaktivní interpacket arrival čas zdroje / cíle [ms].	kvantitativní
SIntPktMax / DIntPktMax	Nejdelší interpacket arrival čas zdroje / cíle ve flow [ms].	kvantitativní
SIntPktMin / DIntPktMin	Nejkratší interpacket arrival čas zdroje / cíle ve flow [ms].	kvantitativní
SIPActMax / DIPActMax	Nejdelší aktivní interpacket arrival čas zdroje / cíle ve flow [ms].	kvantitativní
SIPActMin / DIPActMin	Nejkratší aktivní interpacket arrival čas zdroje / cíle ve flow [ms].	kvantitativní
SIPIdlMax / DIPIdlMax	Nejdelší neaktivní interpacket arrival čas zdroje / cíle ve flow [ms].	kvantitativní
SIPIdlMin / DIPIdlMin	Nejkratší neaktivní interpacket arrival čas zdroje / cíle ve flow [ms].	kvantitativní
SrcJitter / DstJitter	Jitter čas zdroje / cíle [ms].	kvantitativní
SrcJitAct / DstJitAct	Aktivní jitter čas zdroje / cíle [ms].	kvantitativní
SrcJitIdl / DstJitIdl	Neaktivní jitter čas zdroje / cíle [ms].	kvantitativní
State	Stav transakce.	kategoriální
SynAck	Potřebný čas pro oznámení o navázání TCP spojení, délka uplynulého času mezi paketem obsahující SYN a SYN_ACK [s].	kvantitativní
AckDat	Potřebný čas potvrzující zahájení TCP spojení, délka uplynulého času mezi paketem obsahující SYN_ACK a ACK [s].	kvantitativní
TcpRtt	TCP round-trip time, suma příznaků SynAck a AckDat [s].	kvantitativní
TcpOpt	TCP možnosti viděné při zahájení komunikace.	kategoriální
STcpMax, DTcpMax	Největší použitá šířka pásma během TCP spojení zdroje / cíle.	kvantitativní



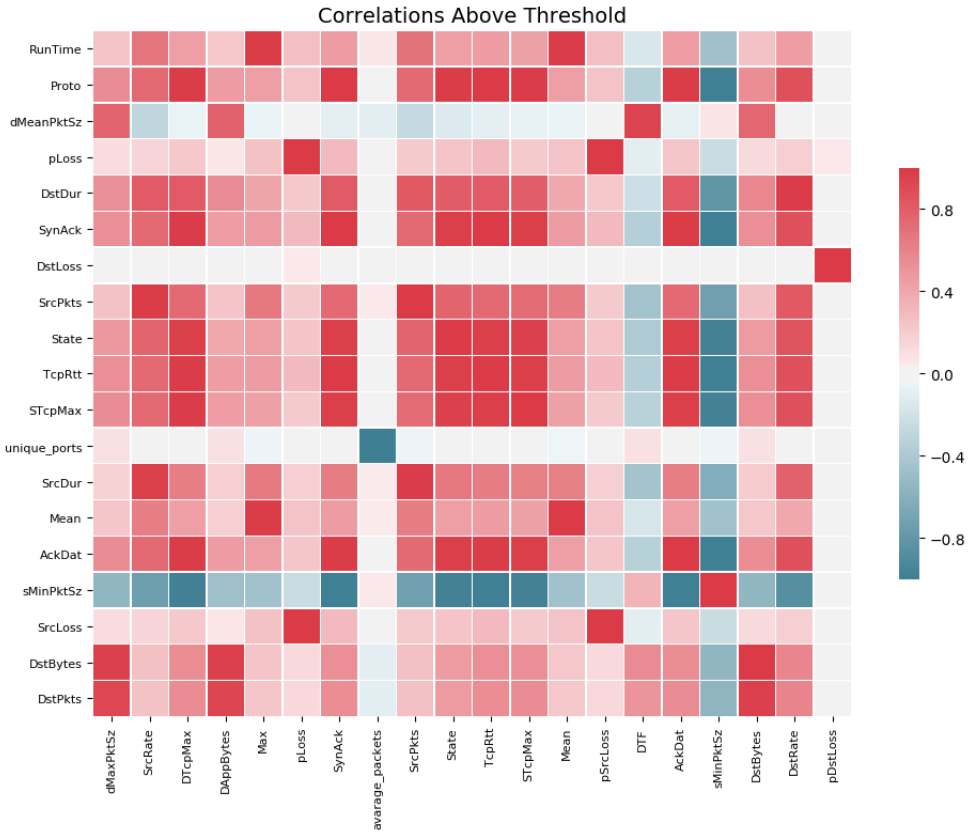
Obrázek 5.9: Histogramy pro TCP sloupec DIntPkt po transformaci



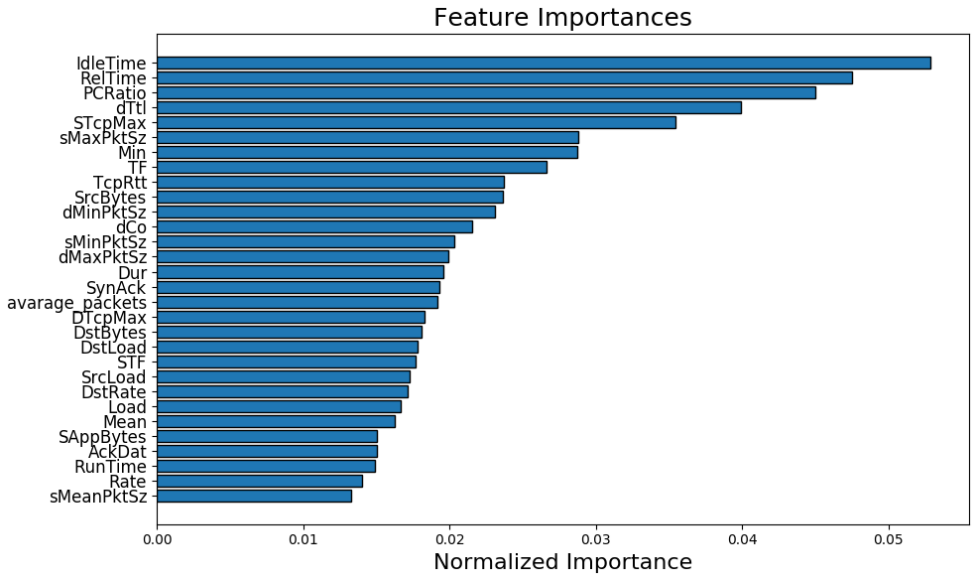
Obrázek 5.10: Histogram pro TCP sloupec DIntPktMax bez transformace



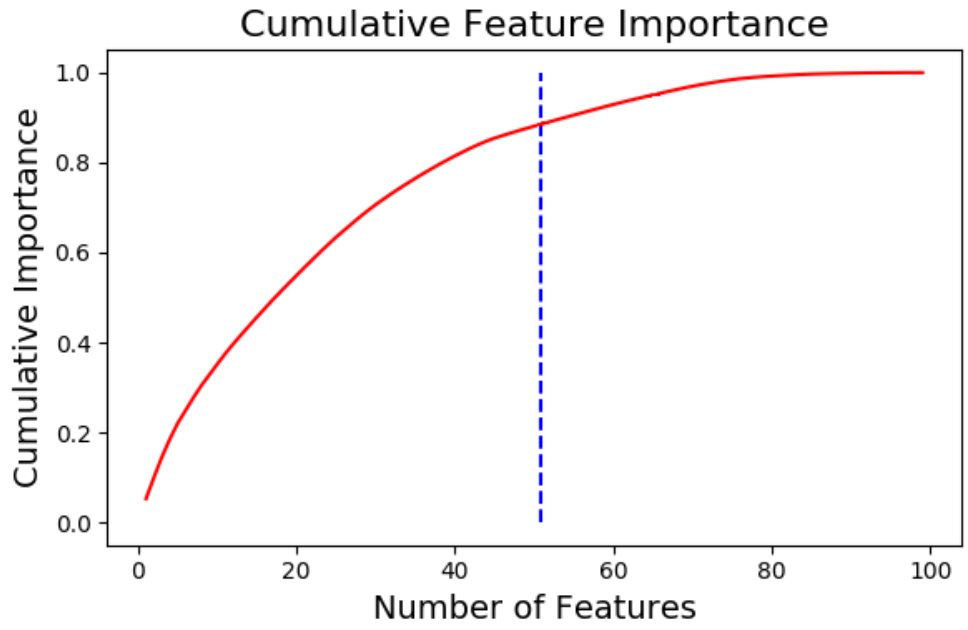
Obrázek 5.11: Histogramy pro TCP sloupec DIntPktMax po transformaci



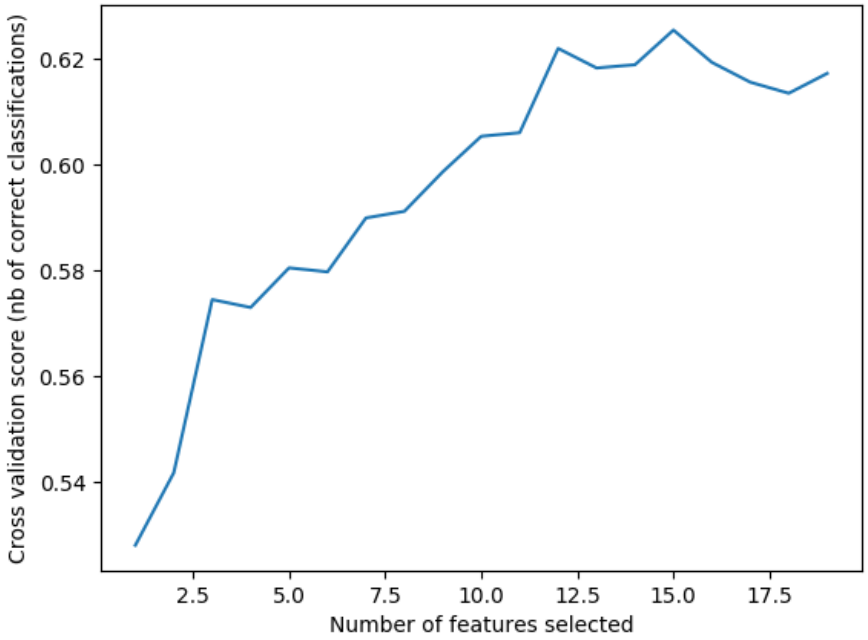
Obrázek 5.12: Dvojice příznaků s vyším korelačním koeficientem než $\pm 0,95$ pro dataset CTU07 a Ordinal kódováním



Obrázek 5.13: Třicet nejvíce důležitých příznaků podle algoritmu *LGBMClassifier* pro dataset IDS17 při Argus kódování.



Obrázek 5.14: Kumulativní důležitost příznaků s tresholdem na $0,85$ pro dataset IDS17 při Argus kódování.



Obrázek 5.15: Graf znázorňující vztah průměrného skóre AUC ROC vůči počtu příznaků pomocí metody *RFECV* pro dataset WIDE.

Kapitola 6

Výsledky

Tato kapitola shrňuje výsledky pokusů v rámci předzpracování dat a aplikace algoritmů strojového učení, a také výsledky navrženého kaskádního řešení, které nejdříve klasifikuje záznamy do třídy určující kategorii a následně klasifikuje aplikační třídu.

6.1 Výpočetní sestava

Měření probíhala ve virtualizačním nástroji VirtulaBox s operačním systému Ubuntu 18.04.2 stable, kterému bylo přiřazeno 3 GB operační paměti a 2 jádra. Na hostujícím počítači pro Virtual box je sestava následující:

- Operační systém: 64 bitový Windows 10.
- Processor: Intel(R) Core(TM) i7-7600U CPU @ 2,80 GHz.
- Operační paměť: 16 GB.

6.2 Souhrn labelů pro aplikace a kategorie

Níže je uvedena přehledná tabulka 6.1, která dává do souvislostí labely aplikací a kategorií. Pro ušetření místa jsou některé labely aplikací uvedeny s pomocí regulárních výrazů. Labely pro katgorie jsou využity pouze v kaskádním řešení.

6.3 Použité hodnotící metriky

Pro hodnocení algoritmů strojového učení byly zvolené následující metriky. Asociace záznamů k aplikaci či typy provozu je hodnocema pomocí metrik accuracy, průměrný AUC ROC (avg. AUC ROC), vážený AUC ROC dle velikosti tříd (wgt. AUC ROC), průměrný precision (avg. precision) a vážený

Tabulka 6.1: Přehled labelů pro aplikace a kategorie

Kategorie	Applikace
Chat	[DNS|HTTP].QQ, DNS.Snapchat, GoogleHangout, STUN.GoogleHangout, TeamSpeak, Viber
Cloud	[DNS|SSL].Dropbox, [DNS|HTTP|SSL].UbuntuONE, Dropbox, [HTTP|SSL].MS_OneDrive
Collaborative	[DNS|HTTP|SSL].Github, [DNS|HTTP|SSL].Office365, [DNS|QUIC|SSL].GoogleDrive
Download-FileTransfer-FileSharing	BitTorrent, [DNS|SSL].BitTorrent, [DNS|SSL].Pastebin, FTP_CONTROL
Email	DNS.GMail, SMTPS.GMail,SSL.GMail
Game	DNS.Steam
Media	[DNS|QUIC|SSL].GoogleDocs, [DNS|SSL].PPStream, [DNS|QUIC|HTTP|SSL].YouTube, RTP
Mining	Mining
Network	DHCP, DHCPV6, DNS, ICMP, Kerberos, LLMNR, MDNS, STUN, UPnP, Whois-DAS
RemoteAccess	SSH
RPC	COAP, DCE_RPC, MQTT, RX
SocialNetwork	[DNS|HTTP|QUIC|SSL].Facebook, [DNS|QUIC|SSL].GooglePlus, [DNS|SSL].Sina(Weibo), [DNS|HTTP|SSL].Instagram, SSL.WeChat, [DNS|HTTP|SSL].LinkedIn, [DNS|HTTP|SSL].Twitter
SoftwareUpdate	[DNS|QUIC|SSL].PlayStore, [DNS|HTTP|SSL].WindowsUpdate
Streaming	[DNS|HTTP].AppleiTunes, [DNS|SSL].SoundCloud, [DNS|HTTP|SSL].GenericProtocol [DNS|HTTP|SSL].Spotify, Spotify
System	BJNP, LDAP, NetBIOS, NTP, SMBv1, SMBv23, SSDP
VoIP	[DNS|HTTP|SSL].Skype, H323, SIP, Skype, Skype.SkypeCall, VHUA
VPN	Tor
Web	[DNS|HTTP|SSL].Amazon, [DNS|HTTP|SSL].Google, [DNS|HTTP|SSL].GoogleMaps, [HTTP|SSL].Cloudflare, [DNS|HTTP|SSL].GoogleServices, [DNS|HTTP|SSL].Microsoft, [DNS|HTTP|SSL].Apple, [DNS|HTTP|SSL].Wikipedia, [DNS|HTTP|SSL].Yahoo, [DNS|SSL].eBay, DNS.MSN, HTTP, HTTP.MSN, SSL, SSL_No_Cert, STUN.Amazon, Amazon, Google

precision dle velikosti tříd (wgt. precision). Detekce anomálií je hodnocena pomocí metrik recall, AUC PRC a F1 skóre.

6.4 Čištění dat

Během fáze čištění datasetu nastaly dvě situace, které si žádaly o přeimplementování či dodělání. Jednalo se o snižení hodnoty *treshold* při mazání sloupců s nedostatkem dat a smazání labelů, které nejsou reprezentovány dostatečným množtvím záznamů.

Defaultní hodnota tresholdu pro zachování sloupců s málo daty byla z počátku nastavena na pevno na 60 %. To znamenalo, že při absenci většiny hodnot ve sloupci, byl sloupec odstraněn. Avšak obsahuje-li dataset větší množství UDP komunikace než TCP, je více než pravděpodobné, že sloupce popisující pouze TCP komunikaci budou vymazány. Tato situace nastala u dvou datasetů (CTU08 a CTU07), příčinou je, že sloupce popisující TCP vlatnosti spojení neobsahují žádné hodnoty pro UDP záznamy. Proto byla zvolena nová defaultní hodnota tresholdu na 20 % a její možnost změny pomocí konfiguračního soubouru.

Avšak při nastavení nízkého tresholdu nastal druhý problém, zůstaly zachované sloupce s větším množstvím chybějících dat. Argus například nevrací žádné hodnoty pro většinu časových sloupců, bohužel se nepodařilo zjistit důvod tohoto chování. Před samotným předzpracováním datasetu si uživatel může nechat vypsat informace o datasetu, viz sekce 5.5.1 a na základě získaných informací doplnit do nově doimplementovaného parametru v konfiguračním souboru *removecolumns* jména sloupců, které se mají dodatečně vymazat. Přehled procentuálního zastoupení chybějících údajů v datasetech lze nalézt v tablce 6.2. Z důvodu velkého množství chybějících časových ůdajů pro inter-arrival časy paketů a jitter časy u většiny datasetů se s nimi dále nebude pracovat.

Dále se v datasetech objevuje řada aplikačních labelů, které jsou reprezentované pouze několika málo záznamy v řádu jednotek. Proto byl doimplementován výběr pouze těch labelů, které jsou reprezentované alespoň 18 a více záznamy. Číslo 18 bylo zvoleno na základě pokusu, který spočíval v zjištování AUC ROC hodnoty pro daný label při aplikaci algoritmu C4.5. Ukázalo se, že většina labelů s nedostačujícím počtem záznamů je úspěšně klasifikováná i při nižším čísle než 18, ale naopak pokud byl některý záznam klasifikován nesprávně hodnota AUC ROC se výrazně snížila.

Datasety po čištění obsahují následující počet flow uvedený v tabulce 6.3. Z důvodu velkých datasetů a malé operační paměti výpočetního stroje byly tři velké datasety, jmenovitě IDS17, WIDE a CTU08 rozděleny do několika menších vzorků. Počet a průměrná velikost vzorků je také uvedena v tabulce. Metriku celého datasetu tvoří suma metrik jednotlivých vzorků podělená poč-

tem vzorků. V případě datasetu IDS17 je veškerá komunikace rozdělena do 5 dnů.

6.5 Volba a výběr příznaků

Níže jsou uvedeny výsledky algoritmu C4.5 na příznacích kódovaných pomocí Ordinal kódování a výsledky algoritmu k-NN na normalizovaných a tranformovaných příznacích kódovaných Argus kódováním. Normalizace byla použita *Gaussian* a transformace *Normal*, více o výsledcích normalizace a transformace v části 6.8.

Výběr odlišných klasifikačních algoritmů pro obě kódování je ovlivněm dopadem kódovaných příznaků na algoritmy. Zatímco algoritmy založených na stromech umějí pracovat s kategoriálními příznaky, shlukovací algoritmy nikoliv. V případě algoritmů založených na stromech je dokonce dosaženo menší úspěšnosti klasifikátoru při kódování Argus, než je tomu s Ordinal kódování. Přesný opak nastává s algoritmy založenými na některé z formem výpočtu vzdáleností mezi záznamy, kdy se naopak dosahuje lepších výsledků s kódováním Argus než s Ordinal.

6.5.1 Vlastní výběr příznaků

V konfiguračním souboru se defaultně nacházejí ty příznaky z jejichž pomocí se dosud dosáhlo průměrně nejlepších výsledků:

- Kvantitativní příznaky: Rate, PCRatio, IdleTime, RunTime, Min, Mean, StdDev, AckDat, STcpMax, SynAck, TcpRtt, DAppBytes, TF, RelTime, dMinPktSz, dMaxPktSz, dMeanPktSz, SAppBytes, STF, sMinPktSz, sMaxPktSz, sMeanPktSz, SrcBytes, DiffSrcDstStartTime.
- Ordinal kódování: Sport, Dport.
- Argus kódování: State_CON, State_S, sTos_Precedence, Dport_1, Dport_9, Dport_8, Dport_7.

Jejich určení probíhalo na základě více faktorů, mezi ně spadá vyřazení jednoho z dvojice příznaků s vysokým korelačním koeficientem, vybrání či odebrání příznaku na základě vráceného koeficinetu nebo důležitosti z algoritmů *LGBMClassifier*, *DecisionTreeClassifier* a *LassoCV*, pokusech a zkoušení různých kombinací.

V seznamech důležitých příznaků se ve všech algoritmech vyskystují příznaky: TF, STF, IdleTime, Min, Mean, PCRatio, SrcBytes, RunTime. Dále mezi důlěžitými až méně důležitými se vyskytují příznaky sTtl, SrcBytes, sTos, avarage_packets, State, DiffSrcDstStartTime, TcpRtt, STcpMax, DTcpMax a SynAck.

Pro argus kódování se vyskytují následující kategoriální příznaky: Dport_1, Dport_9, Dport_8, Dport_7, Sport_5, Sport_6, Sport_8, State_CON a State_S, .

6.5.1.1 Korelovné příznaky

Mezi příznaky, které mají mezi sebou vysoký korelační koeficient při kódování Ordinal, patří nejčastěji tyto n-tice:

- Mean - Max - Runtime,
- sTos - SrcOui,
- SrcPkts - SrcDur,
- DstBytes - DstPkts,
- SrcLoss - pLoss - Loss - pSrcLoss,
- SrcRate - SrcPkts - Load - SrcLoad,
- DstRate - DstDur - DstLoad,
- SynAck - TcpRtt - DTcpMax - STcpMax - AckDat - DstRate.

Důkazem je již dříve přiložený graf 5.12 zobrazující vysoce korelované příznaky. Pro ostatní datasety se tento graf nijak zásadně nemění.

Mezi příznaky, které mají mezi sebou vysokým korelačním koeficientem při kódování Argus, patří vedle výše zmíněných kvantitativních n-tic, nejčastěji tyto n-tice s kategoriálními příznaky:

- State_s, State_f, State_F, TcpRtt, SynAck, AckDat, STcpMax, DTcpMax,
- State_INT, Proto_tcp.

6.5.2 Důležitost příznaků podle algoritmu *LGBMClassifier*

Tabulka 6.4 zobrazuje 20 nejdůležitejších příznaků pro každý dataset s použitým Ordinal kódováním podle stromového boosting algoritmu *LGBMClassifier*.

Algoritmus vybral z kategoriálních příznaků jako důležité pouze zdrojový port, cílový port a příznak dCo, ostatní vybrané příznaky jsou kvantitativního typu. Mezi nejčastěji vybrané kvantitativní příznaky se řadí velikosti paketů a z nich příznaky odvozené jako například efektivika komunikace, poměr paketů zdroje a cíle a počet paketů za sekundu, dále se zde vyskytují příznaky určující dobu trvání přenosu paketů, jako maximální, minimální, aktivní a neaktivní doba přenosu paketů.

Mezi středně důležité se řadí nejčastěji příznaky: DTF, SynAck, DstRate, DTF, SrcPkts, DstLoad, SrcRate, DstDur, SrcDur, Flgs, Proto, DAppBytes, State, dMaxPktSz a pLoss. Jako nepotřebné příznaky s nulovou důležitostí byly označeny příznaky: DstLoss, pDstLoss, sTos, dTos, DstOui a SrcOui.

Při argus kódování se důležitost příznaků nikterak zásadně nemění, kromě kategoriálních příznaků, které se posunuly o několik pozic níže, avšak stále sloupce označující zakódované bity portů se vyskytují v prvních třiceti nejvíce důležitých příznaků, nejčastěji to jsou: Dport_1, Dport_9, Dport_8 a Sport_1, Sport_2, Sport_3, Sport_4, Sport_5, Sport_6, Sport_8.

Tabulka 6.2: Procentuální vyjádření všech chybějících hodnot v datasetech

Příznak	Procento chybějících dat v datasetu [%]						
	IDS17	CTU92	CTU91	CTU08	CTU07	CTU06	WIDE
DstMac	0.231	0	0	0	0	0	0
DstOui	0.468	3.372	12.472	0.283	0.331	22.919	0
sTos	0.001	0	0	0	0	0	5.571
dTos	2.770	20.968	24.311	1.836	5.635	24.821	86.590
sCo	0.117	0.528	1.520	0.077	0.092	8.417	4.136
dCo	0.117	0.528	1.520	0.077	0.092	8.417	4.136
sTtl	0.001	0	0	0	0	0	5.571
dTtl	2.770	20.968	24.311	1.836	5.635	24.821	86.590
SIntPkt	22.977	45.437	45.500	68.610	62.724	21.687	82.783
SIntPktAct	22.977	45.437	45.500	68.610	62.724	21.687	82.783
SIntPktIdl	22.977	45.437	45.500	68.610	62.724	21.687	82.783
SIntPktMax	22.977	45.437	45.500	68.610	62.724	21.687	82.783
SIntPktMin	22.977	45.437	45.500	68.610	62.724	21.687	82.783
SIPActMax	22.977	45.437	45.500	68.610	62.724	21.687	82.783
SIPActMin	22.977	45.437	45.500	68.610	62.724	21.687	82.783
SIPIdlMax	22.977	45.437	45.500	68.610	62.724	21.687	82.783
SIPIdlMin	22.977	45.437	45.500	68.610	62.724	21.687	82.783
DIntPkt	22.977	56.668	68.143	71.495	73.429	50.395	90.977
DIntPktAct	22.977	56.668	68.143	71.495	73.429	50.395	90.977
DIntPktIdl	22.977	56.668	68.143	71.495	73.429	50.395	90.977
DIntPktMax	22.977	56.668	68.143	71.495	73.429	50.395	90.977
DIntPktMin	22.977	56.668	68.143	71.495	73.429	50.395	90.977
DIPActMax	22.977	56.668	68.143	71.495	73.429	50.395	90.977
DIPActMin	22.977	56.668	68.143	71.495	73.429	50.395	90.977
DIPIdlMax	22.977	56.668	68.143	71.495	73.429	50.395	90.977
DIPIdlMin	22.977	56.668	68.143	71.495	73.429	50.395	90.977
SrcJitter	24.069	50.721	63.566	68.971	63.489	28.785	88.171
SrcJitAct	47.022	80.963	84.652	95.926	86.932	53.792	92.142
SrcJitIdl	64.660	66.250	77.760	71.913	74.003	73.156	93.736
DstJitter	26.804	56.803	68.228	71.495	73.431	50.412	90.985
DstJitAct	43.052	73.793	83.221	97.965	97.031	58.502	98.990
DstJitIdl	63.037	64.577	74.652	71.873	73.950	72.644	91.226
State	0.001	0.097	0.081	0	0	0.011	0.031
TcpOpt	51.821	22.569	37.621	61.872	63.999	32.464	35.232
sMaxPktSz	0.002	0	0.007	0	0	0.027	5.572
dMaxPktSz	2.936	21.077	24.407	1.969	6.024	24.821	86.595
sMinPktSz	0.077	0	0.018	0.001	0.002	0.076	5.579
dMinPktSz	2.976	21.077	24.416	1.977	6.027	24.821	86.597

Tabulka 6.3: Datasety a počet flow po předzpracování

Dataset	Počet flow	Počet anomálií	Počet vzorků	Velikost vzorku
IDS17	1613590	142204	5	322718
CTU92	14473	101	-	-
CTU91	111230	595	-	-
CTU08	1729613	450300	4	432404
CTU07	241007	51570	-	-
CTU06	18339	1	-	-
WIDE	2393706	4558	7	341958

Tabulka 6.4: Dvacet nejvíce důležitých příznaků podle algoritmu *LGBMClassifier*

Nejdůležitejší příznaky pro dataset						
IDS17	CTU92	CTU91	CTU08	CTU07	CTU06	WIDE
Sport	Sport	IdleTime	Sport	Sport	Sport	Dport
RelTime	IdleTime	Sport	RelTime	RelTime	IdleTime	Sport
IdleTime	RelTime	RelTime	IdleTime	IdleTime	RelTime	IdleTime
PCRatio	PCRatio	PCRatio	Min	Min	PCRatio	RelTime
dTtl	dCo	Dport	Rate	SrcBytes	SrcBytes	sTtl
STcpMax	TF	dCo	DiffSrcDstStartTime	sMeanPktSz	TF	avarage_packets
sMaxPktSz	SrcBytes	TF	RunTime	sMinPktSz	DstBytes	unique_ports
Min	Min	SrcBytes	AckDat	PCRatio	dCo	dCo
TF	Dport	dMinPktSz	Dur	TF	sMaxPktSz	sCo
SrcBytes	Dur	Dur	StdDev	Rate	Min	sTos
Dport	DstBytes	SynAck	SrcBytes	Dur	Dur	TcpOpt
dMinPktSz	SynAck	Min	sMinPktSz	sMaxPktSz	RunTime	sMinPktSz
TcpRtt	sMinPktSz	RunTime	TF	SAppBytes	sMeanPktSz	State
sMinPktSz	dMinPktSz	DstBytes	PCRatio	STF	Mean	Min
dCo	AckDat	DTcpMax	STF	DstBytes	DAppBytes	Dur
SynAck	RunTime	TcpRtt	DstBytes	DiffSrcDstStartTime	Dport	SrcBytes
Dur	STcpMax	Rate	Mean	Mean	Rate	Load
dMaxPktSz	Rate	STcpMax	Load	RunTime	STF	Rate
DstBytes	Mean	Mean	DTcpMax	SrcPkts	dMinPktSz	DstDur
avarage_packets	sMaxPktSz	sMinPktSz	SAppBytes	DTcpMax	dMaxPktSz	TF

6.5.3 Důležitost příznaků podle algoritmu *DecisionTreeClassifier*

Tabulka 6.5 obsahuje 20 nejdůležitejších příznaků pro každý dataset s použitým Ordinal kódováním podle stromového algoritmu *DecisionTreeClassifier*.

Algoritmus vybral jako velmi důležité kategoriální příznaky Dport, Sport, dCo. Mezi méně časté kategoriální příznaky, ale pro některé datasety stále důležité, jsou Proto a Dir. Z kvantitativních příznaků jsou důležité maximální a minimální velikosti paketů od odesílatele a příjemce, jejich poměr a efektivita komunikace a opět minimální, maximální a průměrné aktivní a neaktivní časy přenosu paketů, jako tomu je u algoritmu *LGBMClassifier*. Avšak *DecisionTreeClassifier* oproti boosting algoritmu nepokoládá za tak důležité celkový počet bytů a paketů v komunikaci.

Mezi středně důležité se řadí nejčastěji příznaky: DAppBytes, SAppBytes, Dur, sMaxPktSz, dMeanPktSz a Load. Jako nepotřebné příznaky s nulovou důlěžitostí byly označeny nejčastěji příznaky: Loss, sTos, dTtl, DstOui, SrcOui a Flgs.

Při argus kódování stále zůstavájí důlěžité stejné kvantitativí příznaky, u kategoriálních příznaků zůstal důlěžitý pouze Dport_1, Dport_2, Dport_4, Dport_7, Dport_8 a Dport_9. Kategoriální příznak Sport se propadl na dno důležitosti, naopak příznak State se vyšvihl do tabulky důležitých příznaků, přesněji se jedná o State_CON, State_S a State_R.

Tabulka 6.5: Dvacet nejvíce důležitých příznaků podle algoritmu *DecisionTreeClassifier*

Nejdůležitejší příznaky pro dataset						
IDS17	CTU92	CTU91	CTU08	CTU07	CTU06	WIDE
Dport	Proto	Dport	dCo	dCo	Dport	Dport
dTtl	dMinPktSz	SrcBytes	Dport	Dport	State	sTtl
Sport	Dport	State	Sport	sMeanPktSz	sMinPktSz	sMinPktSz
TcpRtt	dCo	IdleTime	Dir	Sport	dCo	IdleTime
PCRatio	SynAck	Proto	DstPkts	SynAck	Min	unique_ports
dMaxPktSz	PCRatio	sTtl	SrcPkts	IdleTime	TF	avarage_packets
STcpMax	Mean	pSrcLoss	sTtl	DTcpMax	sMaxPktSz	Sport
sMinPktSz	sMinPktSz	dCo	RelTime	PCRatio	PCRatio	State
IdleTime	IdleTime	Sport	IdleTime	State	Dur	TcpOpt
SynAck	TcpRtt	PCRatio	Min	STcpMax	SAppBytes	Proto
dCo	Min	TcpRtt	Mean	sTtl	unique_ports	sCo
sMaxPktSz	Dir	SynAck	STcpMax	Min	Sport	DstBytes
RelTime	TF	STcpMax	PCRatio	RelTime	DstBytes	dCo
TF	Sport	dMinPktSz	sMeanPktSz	SrcRate	Rate	sTos
DstRate	sTtl	TF	SrcBytes	sMinPktSz	SynAck	TF
dMinPktSz	DTcpMax	SrcLoss	RunTime	dMeanPktSz	DAppBytes	Min
avarage_packets	STcpMax	sMeanPktSz	TF	Trans	DTF	DstPkts
DTF	SrcRate	STF	Rate	Mean	IdleTime	dMeanPktSz
SAppBytes	Rate	DstBytes	DstDur	Max	SrcBytes	Rate
dTos	dMeanPktSz	RelTime	DiffSrcDstStartTime	SrcPkts	dMinPktSz	SrcRate

6.5.4 Důležitost příznaků podle algoritmu *LassoCV*

Tabulka 6.6 obsahuje 20 nejdůležitejších příznaků pro každý dataset s použitým Ordinal kódováním podle algoritmu *LassoCV*.

Algoritmus vybral jako velmi důležité kategoriální příznaky Proto, Dport, Dir, DstOui, Flgs, State a sTos. Velikosti a počet paketů se vyskytly jako důležité příznaky pouze u dvou datasetů, taktéž časové příznaky o délce trvání přenosu paketů se posunuly mezi méně důležité. Místo toho byly nahrazeny výše zmíněnými kategoriálními příznaky a kvantitativními příznaky Loss, sTtl, StdDev, AckDat.

Mezi středně důležité spadají nejčastěji příznaky: RunTime, dMinPktSz, dTtl a sMinPktSz. Mezi nedůležité spadají příznaky: DstBytes a SrcLoss. Zbylým příznaků se značně liší důležitost dle použitého datasetu.

Tabulka 6.6: Dvacet nejvíce důležitých příznaků podle algoritmu *LassoCV*

Nejdůležitější příznaky pro dataset						
IDS17	CTU92	CTU91	CTU08	CTU07	CTU06	WIDE
Proto	AckDat	AckDat	DstOui	DstOui	AckDat	sTos
sMeanPktSz	TF	TF	STF	pDstLoss	sTos	sMaxPktSz
SrcPkts	STF	sTos	Proto	STF	TF	SrcPkts
DstOui	Proto	DstOui	TF	TF	DstOui	DstPkts
dMaxPktSz	DTF	dTos	Dport	Dport	IdleTime	dMeanPktSz
TF	IdleTime	Proto	DTF	Proto	SrcOui	RunTime
dTos	SrcOui	TcpRtt	PCRatio	DTF	Proto	sMeanPktSz
Trans	StdDev	Mean	unique_ports	unique_ports	StdDev	Proto
STF	Dir	STF	Loss	PCRatio	STF	SrcBytes
sTtl	sTos	Min	State	AckDat	Max	SrcRate
avarage_packets	DstLoss	Dir	Trans	DstLoss	Dport	sTtl
Dir	Flgs	IdleTime	sTtl	State	sCo	sMinPktSz
DstLoad	PCRatio	DTF	avarage_packets	avarage_packets	PCRatio	Dur
StdDev	Min	SrcOui	Mean	sTtl	Min	Trans
Flgs	State	PCRatio	dCo	RunTime	Mean	StdDev
Load	Loss	Max	RunTime	IdleTime	dTos	Dport
DiffSrcDstStartTime	avarage_packets	StdDev	dTtl	StdDev	Flgs	SAppBytes
AckDat	DiffSrcDstStartTime	Flgs	IdleTime	Min	Dir	TcpRtt
SrcBytes	TcpRtt	pDstLoss	TcpRtt	dCo	unique_ports	dCo
PCRatio	pLoss	Trans	SynAck	pLoss	pLoss	DAppBytes

6.6 Výsledky selekčních algoritmů

Výsledky selekčních algoritmů jsou reprezentovány pouze pomocí metrik accuracy a avg. AUC ROC, které byly brány jako hlavní ukazatele. Výsledky jsou uvedeny pomocí desetinného čísla a udávají úspěšnost na testovacím datasetu.

Výsledky wrapper algoritmů nebyly získány z důvodu jejich velké časové náročnosti a malé výpočetní kapacity pro provedení selekce přes všechny wrapper metody a datasety. Níže jsou vypsané příznaky, které vrátila metoda RFECV pro datasety WIDE a IDS17, z níž je i vidět, že největší úspěšnost pro stromové algoritmy je dosažena při přítomnosti takměř všech příznaků:

- WIDE: Dur, SrcDur, DstDur, Sport, Dir, Dport, Trans, Flgs, RunTime, IdleTime, Mean, StdDev, Min, Max, SrcOui, DstOui, sTos, sCo, dCo, sTtl, SrcPkts, DstPkts, SrcBytes, DstBytes, SAppBytes, DAppBytes, PCRatio, SrcLoad, DstLoad, SrcLoss, pLoss, pSrcLoss, SrcRate, DstRate, State, TcpRtt, AckDat, TcpOpt, sMaxPktSz, sMinPktSz, sMeanPktSz, dMeanPktSz, TF, STF, DTF, STcpMax, DTcpMax.
- IDS17: Dur, DstDur, Dir, Dport, Proto, Trans, RunTime, Mean, Min, SrcOui, DstOui, sTos, dTos, dCo, sTtl, dTtl, SrcPkts, DstPkts, SrcBytes, SAppBytes, DAppBytes, PCRatio, SrcLoad, DstLoad, SrcLoss, DstLoss, pLoss, pSrcLoss, pDstLoss, SrcRate, DstRate, State, TcpRtt, SynAck, TcpOpt, sMaxPktSz, dMaxPktSz, sMinPktSz, dMinPktSz, sMeanPktSz, dMeanPktSz, TF, STF, DTF, STcpMax, DTcpMax.

Tabulka 6.7 popisuje úspěšnost algoritmu C4.5 pro Ordinal kódování a algoritmu k-NN pro Argus kódování všech selekčních algoritmů pomocí metrik accuracy a průměrné hodnoty AUC ROC přes všechny datasety. Tučně zbýrazněné hodnoty accuracy odpovídají nejlepším výsledkům pro daný dataset vůči příznakům vybraným selekčním algoritmem. Z tabulky je patrné, že v případě stromového algoritmu C4.5 je nejlepších výsledků dosaženo pomocí všech příznaků, druhého nejlepšího výsledku je dosaženo pomocí příznaků z kofiguračního souboru zvolených přes seleci *Own*. Překvapivý výsledek je u algoritmu k-NN, kde v některých případech si vedl mnohem lépe při vybraných příznacích z konfiguračního souboru a v případě datasetu CTU92, CTU07 a WIDE si vedl nejlépe při přítomnosti všech příznaků. Naopak v případě CTU08 se ukázalo, že čím méně příznaků je vybráno, tím je větší skóre accuracy.

Tabulka 6.7: Selekce příznaků pro všechny datasety a obě kódování

Kódování	Metoda	Metrika	Dataset						
			IDS17	CTU92	CTU91	CTU08	CTU07	CTU06	WIDE
Ordinal, algoritmus C4.5	Všechny příznaky	Accuracy	**0.9091**	**0.8696**	**0.9003**	**0.9876**	**0.9717**	**0.9282**	**0.9842**
		avg. AUC ROC	0.9342	0.9493	0.9259	0.9521	0.9363	0.9603	0.9583
	Own	Accuracy	**0.8839**	**0.8649**	**0.8953**	0.9705	**0.9716**	**0.9228**	**0.9792**
		avg. AUC ROC	0.9271	0.9456	0.9154	0.9437	0.9697	0.9614	0.9758
	SelectKbest	Accuracy	0.8790	0.8071	0.8714	**0.9866**	0.9699	0.9118	0.9740
		avg. AUC ROC	0.9234	0.9393	0.9199	0.8699	0.8533	0.954	0.9787
	Percentile	Accuracy	0.8298	0.7975	0.8712	0.9865	0.9696	0.9131	0.9724
		avg. AUC ROC	0.8871	0.9394	0.9193	0.8727	0.8572	0.9561	0.9826
	LassoCV	Accuracy	0.7732	0.8536	0.8769	**0.9866**	0.9708	0.9064	0.8668
		avg. AUC ROC	0.8419	0.9520	0.9124	0.8711	0.9559	0.9508	0.9311
	mRMR	Accuracy	0.7032	0.7685	0.8283	0.9574	0.9667	0.9128	0.7579
		avg. AUC ROC	0.9228	0.9298	0.9156	0.8718	0.8640	0.9623	0.8561
Argus, algoritmus k-NN	Všechny příznaky	Accuracy	0.5177	**0.8595**	**0.7916**	0.9605	**0.9676**	0.4650	**0.9547**
		avg. AUC ROC	0.5370	0.7698	0.7468	0.8341	0.7936	0.6226	0.8762
	Own	Accuracy	**0.8721**	**0.8479**	**0.8937**	0.9679	**0.9652**	**0.9359**	0.7850
		avg. AUC ROC	0.8570	0.8139	0.7798	0.8205	0.7844	0.9422	0.6192
	SelectKbest	Accuracy	**0.8434**	0.8029	0.7882	**0.9856**	0.9669	0.9172	**0.9510**
		avg. AUC ROC	0.8396	0.7615	0.8297	0.7947	0.7925	0.9191	0.8286
	Percentile	Accuracy	0.8396	0.8003	0.7668	**0.9860**	**0.9700**	0.9175	0.9490
		avg. AUC ROC	0.7924	0.8228	0.7587	0.7987	0.7956	0.9212	0.8275
	LassoCV	Accuracy	0.7899	0.8516	0.7883	0.9848	0.9562	**0.9397**	0.8333
		avg. AUC ROC	0.7505	0.7931	0.8304	0.8372	0.7669	0.9440	0.6433
	mRMR	Accuracy	0.6655	0.7461	0.6902	0.9511	0.9588	0.8215	0.9351
		avg. AUC ROC	0.6187	0.7222	0.6568	0.7689	0.7395	0.8401	0.7760

6.7 Výsledky algoritmů pro extrakci příznaků

V případě extrakce příznaků je získáváno 6 příznaků odvozených ze všech vstupních příznaků po zakódování. Číslo 6 tvoří mezníka mezi úspěšností extrakce příkazů pomocí algoritmu *FastICA* a ostatních. Při použití většího čísla je dosaženo na vzniklých příznacích z algoritmu *FastICA* lepší predikce nových dat, ale v případě ostatních extrakčních algoritmů je dosaženo horší. A přesně naopak.

Tabulka 6.8 popisuje úspěšnost algoritmu C4.5 pro Ordinal kódování a algoritmu k-NN pro Argus kódování všech extrakčních algoritmů pomocí metrik accuracy a průměrné hodnoty AUC ROC přes všechny datasety. Pro porovnání úspěšnosti tabulka obsahuje i dosažených výsledků pro všechny příznaky. V případě Ordinal kódování a stromového algoritmu C4.5 si extrakce příznaků vede v průměru o 9 % hůře než při zanechání příznaků v původní podobě, i tak z vybraných extrakčních metod si nejlépe vede FA a LSA. V případě Argus kódování a k-NN algoritmu si nejlépe vede opět FA, které dokonce dosahuje lepších výsledků, než při použití všech původních příznaků, druhá nejlepší metoda je PCA.

Při porovnávní výsledků selekčních a extrakčních algoritmů je patrné, že pro Ordinal kódování a stromový algoritmus C4.5 se vyplatí použít spíše selekční algoritmus. Totéž tvrzení platí i pro argus kódování a algoritmus k-NN, avšak zde si extrakce nevedla až o tolik hůře.

Tabulka 6.8: Extrakce příznaků pro všechny datasety a obě kódování

Kódování	Metoda	Metrika	Dataset						
			IDS17	CTU92	CTU91	CTU08	CTU07	CTU06	WIDE
Ordinal, algoritmus C4.5	Všechny příznaky	Accuracy	**0.9091**	**0.8696**	**0.9003**	**0.9876**	**0.9717**	**0.9282**	**0.9842**
		avg. AUC ROC	0.9342	0.9493	0.9259	0.9521	0.9363	0.9603	0.9583
	FastICA	Accuracy	0.6901	0.6646	0.7302	0.9556	0.9641	0.8018	0.6410
		avg. AUC ROC	0.8142	0.8332	0.8687	0.8698	0.8209	0.8776	0.6826
	FA	Accuracy	0.7163	**0.7538**	**0.8630**	**0.9826**	0.9658	**0.8672**	0.7203
		avg. AUC ROC	0.7696	0.9081	0.8843	0.8415	0.8296	0.9209	0.7382
	PCA	Accuracy	0.7244	0.6781	0.8245	0.9738	0.9651	0.8283	0.7547
		avg. AUC ROC	0.7605	0.8535	0.8482	0.8320	0.8268	0.8848	0.7399
	LSA	Accuracy	**0.7251**	0.6778	0.8296	0.9723	**0.9660**	0.8290	**0.7636**
		avg. AUC ROC	0.7641	0.8535	0.8516	0.8296	0.8178	0.8857	0.7488
Argus, algoritmus k-NN	Všechny příznaky	Accuracy	0.5177	**0.8595**	**0.7916**	**0.9605**	**0.9676**	0.4650	**0.9547**
		avg. AUC ROC	0.5370	0.7698	0.7468	0.8341	0.7936	0.6226	0.8762
	FastICA	Accuracy	0.6997	0.8306	0.7704	0.8579	0.9626	**0.8966**	0.9346
		avg. AUC ROC	0.6573	0.7264	0.7176	0.7243	0.7364	0.9039	0.8339
	FA	Accuracy	**0.7541**	**0.8995**	**0.8560**	**0.9821**	0.9638	0.8695	0.8639
		avg. AUC ROC	0.7245	0.7951	0.8520	0.8114	0.7761	0.8790	0.6535
	PCA	Accuracy	**0.7079**	0.8360	0.7615	0.8765	0.9647	**0.9073**	**0.9368**
		avg. AUC ROC	0.6632	0.7380	0.7179	0.7508	0.7528	0.9123	0.8366
	LSA	Accuracy	0.6963	0.8350	0.7690	0.8773	**0.9648**	0.8929	0.9245
		avg. AUC ROC	0.6558	0.7367	0.7105	0.7514	0.7559	0.9023	0.8155

6.8 Normalizace a transformace vstupních dat

Dopad normalizace a transformace vstupních dat na výsledky algoritmů strojového učení je shrnut v této kapitole.

Pro pokus byly použity algoritmy v nastavení uvedeném v kapitole 5.7. Pro algoritmy na stromech bylo použito Ordinal kódování a vlastní volba příznaků. Pro zbylé algoritmy *SVM, kNN, NB* a *KMeans* bylo použito Argus kódování a taktéž vlastní výběr příznaků. Všechny výsledky algoritmů jsou uvedeny pomocí desetinného čísla a značí úspěšnost na testovacím datasetu. Jména algoritmů v tabulce *AdaBst.* a *RForest* jsou zkrácená jména algoritmů *AdaBoost* a *RandomForest*. Tučně zvýrazněná čísla odpovídají nejlepším dosaženémým výsledkům pro accuracy.

Jak už bylo zmíněno, stromy jsou algoritmy, které nevyžadují transformaci ani normalizaci dat, proto výsledek není překvapivý. Normalizace a škálování dat nikterak zásadně nepomohlo ani nezhoršilo výsledky algoritmů založených na stromech, viz tabulky pro dataset CTU06 s přehledem normalizace 6.9 a transformace 6.16. Proto další výsledné tabulky normalizace a transformace pro ostatní datasety již nezahrnují výsledky stromových algoritmů.

V některých tabulkách nejsou uvedeny výsledky algoritmu SVM, to je způsobeno jeho velkou časovou náročností, kdy čas potřebný pro naučení modelu a následnou klasifikaci nových záznamů přesáhl 2 hodiny. Z důvodu hledání algorimu, který je vhodný pro klasifikování internetového provozu je časová složitost algoritmu jedna z podmínek. Proto v dalších kapitolách nebude s algoritmem SVM dále pracováno.

6.8.1 Výsledky normalizace

Tabulky 6.9, 6.10, 6.11, 6.12, 6.13, 6.14 a 6.15 obsahují výsledky normalizace a zachycují důkaz, že normalizace značně pomohla k dosažení lepších výsledků pro algoritmy *SVM, kNN* a *NB*. Nejlepších výsledků je dosaženo při normalizaci *Gaussian* a *MaxAbsScaler*.

Tabulka 6.9: Normalizace a její dopad na CTU06 dataset

Normalizace	Metrika	Algoritmus								
		SVM	kNN	NB	KMeans	C4.5	REPTree	CART	AdaBst.	RForest
Žádná	Accuracy	0.1820	0.1438	0.2181	**0.0139**	0.9208	0.9112	0.8496	0.9273	0.9448
	Avg. AUC ROC	0.5000	0.5129	0.5080	0.5025	0.9607	0.9700	0.8516	0.9200	0.9444
	Wgt. AUC ROC	0.5000	0.5242	0.5234	0.5294	0.5012	0.9889	0.9201	0.9600	0.9700
	Avg. precision	0.0070	0.2053	0.0625	0.0280	0.0123	0.8486	0.7444	0.9094	0.9197
	Wgt. precision	0.0331	0.1427	0.1053	0.1088	0.9253	0.9142	0.8744	0.9274	0.9449
Gaussian	Accuracy	**0.8771**	**0.9032**	**0.5974**	0.0085	0.9228	0.9112	0.8490	0.9279	0.9455
	Avg. AUC ROC	0.8625	0.9048	0.8041	0.5136	0.9621	0.9700	0.8514	0.9167	0.9461
	Wgt. AUC ROC	0.9326	0.9477	0.7907	0.5004	0.9854	0.9700	0.9198	0.9602	0.9703
	Avg. precision	0.8064	0.8350	0.5354	0.0147	0.8864	0.8656	0.7438	0.9149	0.9224
	Wgt. precision	0.8736	0.9035	0.7276	0.0147	0.9283	0.9131	0.8742	0.9287	0.9455
MinMaxScaler	Accuracy	0.8410	0.8985	0.5676	0.0041	0.9224	0.9111	0.8520	0.9311	0.9451
	Avg. AUC ROC	0.7886	0.9045	0.7790	0.4950	0.9626	0.9700	0.8528	0.9214	0.9456
	Wgt. AUC ROC	0.9109	0.9451	0.7738	0.4928	0.9858	0.9889	0.9213	0.9620	0.9701
	Avg. precision	0.6268	0.8287	0.5133	0.0010	0.8779	0.8659	0.7530	0.9131	0.9215
	Wgt. precision	0.8006	0.8976	0.6823	0.0028	0.9262	0.9133	0.8755	0.9312	0.9451
RobustScaler	Accuracy	0.8279	0.8547	0.1853	0.0003	0.9208	0.9112	0.8491	0.9286	0.9451
	Avg. AUC ROC	0.8251	0.8528	0.5011	0.4986	0.9616	0.9700	0.8510	0.9182	0.9465
	Wgt. AUC ROC	0.9075	0.9214	0.5023	0.4997	0.9850	0.9889	0.9199	0.9606	0.9701
	Avg. precision	0.7248	0.7367	0.0674	0.0187	0.8792	0.8488	0.7431	0.9162	0.9195
	Wgt. precision	0.8363	0.8522	0.1857	0.0779	0.9251	0.9143	0.8744	0.9296	0.9451
MaxAbsScaler	Accuracy	0.8434	0.8912	0.5729	0.0602	0.9224	0.9112	0.8519	0.9311	0.9437
	Avg. AUC ROC	0.7916	0.8975	0.7772	0.5034	0.9626	0.9700	0.8528	0.9214	0.9446
	Wgt. AUC ROC	0.9122	0.9411	0.7760	0.5240	0.9858	0.9213	0.9213	0.9620	0.9693
	Avg. precision	0.6644	0.8232	0.4902	0.0168	0.8894	0.8658	0.7529	0.9131	0.9172
	Wgt. precision	0.8078	0.8901	0.6708	0.1978	0.9283	0.9130	0.8754	0.9312	0.9435

Tabulka 6.10: Normalizace a její dopad na CTU07 dataset při Argus kódování a vlastním výběru příznaků

Normalizace	Metrika	Algoritmus		
		kNN	NB	KMeans
Žádná	Accuracy	0.6235	0.0702	0.0103
	Avg. AUC ROC	0.5023	0.5000	0.5007
	Wgt. AUC ROC	0.5300	0.5000	0.4991
	Avg. precision	0.0348	0.0878	0.0101
	Wgt. precision	0.5791	0.4928	0.5147
Gaussian	Accuracy	**0.9607**	0.6788	0.0002
	Avg. AUC ROC	0.7854	0.7049	0.4964
	Wgt. AUC ROC	0.9535	0.8164	0.4922
	Avg. precision	0.6043	0.3413	0.0041
	Wgt. precision	0.9492	0.9383	0.0023
MinMaxScaler	Accuracy	0.9595	0.9132	0.0234
	Avg. AUC ROC	0.7748	0.7147	0.5017
	Wgt. AUC ROC	0.9525	0.9283	0.4883
	Avg. precision	0.5868	0.3763	0.0117
	Wgt. precision	0.9477	0.9353	0.2387
RobustScaler	Accuracy	**0.9620**	0.7001	0
	Avg. AUC ROC	0.7792	0.5141	0.4899
	Wgt. AUC ROC	0.9539	0.5017	0.4999
	Avg. precision	0.6023	0.1200	0.0029
	Wgt. precision	0.9492	0.6192	0.0008
MaxAbsScaler	Accuracy	0.9597	**0.9266**	**0.0146**
	Avg. AUC ROC	0.7688	0.7122	0.4983
	Wgt. AUC ROC	0.9524	0.9344	0.5064
	Avg. precision	0.5848	0.3460	0.0112
	Wgt. precision	0.9475	0.9340	0.6799

6.8.2 Výsledky transformace

Ukázalo se, že samotná transformace má na výsledky algoritmů spíše negativní vliv, viz tabulka 6.16 pro menší dataset CTU06 a tabulka 6.17 pro větší dataset WIDE.

Transformace má pozitivní vliv na výsledky pouze společně s normalizací. Díky tomu i řada implementací algoritmů pro transformaci nabýzí rovnou možnost normalizace příznaků po jejich transformaci. Po více pokusech nakonec vychází nejlépe dvojice transformace *Normal* spolu s normalizací *Gaussian* nebo *MaxAbsScaler*. Normalizace *Gaussian* dosahuje lepších výsledků s algoritmem k-NN, zbylé algoritmy NB, SVM a KMeans vycházejí lépe s norma-

Tabulka 6.11: Normalizace a její dopad na CTU08 dataset při Argus kódování a vlastním výběru příznaků

Normalizace	Metrika	Algoritmus		
		kNN	NB	KMeans
Žádná	Accuracy	0.5434	0.3787	0.0106
	Avg. AUC ROC	0.5051	0.5000	0.4990
	Wgt. AUC ROC	0.5546	0.5000	0.5001
	Avg. precision	0.0562	0.0191	0.0107
	Wgt. precision	0.5249	0.3541	0.6328
Gaussian	Accuracy	**0.9687**	0.5876	0.0017
	Avg. AUC ROC	0.7941	0.7207	0.4954
	Wgt. AUC ROC	0.9704	0.7377	0.5000
	Avg. precision	0.6076	0.3215	0.0160
	Wgt. precision	0.9617	0.8626	0.2589
MinMaxScaler	Accuracy	0.9677	0.5121	0.0282
	Avg. AUC ROC	0.7833	0.5551	0.4973
	Wgt. AUC ROC	0.9697	0.6656	0.4975
	Avg. precision	0.5919	0.3578	0.0112
	Wgt. precision	0.9612	0.8989	0.5634
RobustScaler	Accuracy	0.9656	0.3587	0.0054
	Avg. AUC ROC	0.7924	0.5248	0.4821
	Wgt. AUC ROC	0.9677	0.5193	0.4577
	Avg. precision	0.6127	0.1343	0.0064
	Wgt. precision	0.9581	0.5414	0.0052
MaxAbsScaler	Accuracy	0.9665	**0.5212**	**0.0229**
	Avg. AUC ROC	0.7810	0.7479	0.4990
	Wgt. AUC ROC	0.9687	0.7152	0.5030
	Avg. precision	0.5913	0.3515	0.0155
	Wgt. precision	0.9597	0.8533	0.7533

lizací *MaxAbsScaler*. Úspěšosti algoritmů po transformaci a normalizaci na zvolené datasety je průměrně o necelé 1 % větší, oproti použití pouze normalizace.

Tabulka 6.12: Normalizace a její dopad na CTU91 dataset při Argus kódování a vlastním výběru příznaků

Normalizace	Metrika	Algoritmus		
		kNN	NB	KMeans
Žádná	Accuracy	0.4914	0.2833	0.0098
	Avg. AUC ROC	0.5093	0.5000	0.4989
	Wgt. AUC ROC	0.5262	0.5000	0.4998
	Avg. precision	0.1137	0.0201	0.0103
	Wgt. precision	0.4704	0.1516	0.1114
Gaussian	Accuracy	**0.8468**	0.3680	**0.0206**
	Avg. AUC ROC	0.7697	0.7166	0.4991
	Wgt. AUC ROC	0.9157	0.6795	0.5023
	Avg. precision	0.5936	0.2594	0.0214
	Wgt. precision	0.8415	0.7037	0.3324
MinMaxScaler	Accuracy	0.8387	0.3854	0.0036
	Avg. AUC ROC	0.7560	0.6910	0.4987
	Wgt. AUC ROC	0.9113	0.6867	0.4913
	Avg. precision	0.5777	0.2387	0.0015
	Wgt. precision	0.8323	0.6769	0.0009
RobustScaler	Accuracy	0.8073	0.2950	0.0105
	Avg. AUC ROC	0.7338	0.5085	0.4998
	Wgt. AUC ROC	0.8945	0.5188	0.5034
	Avg. precision	0.5087	0.0297	0.0126
	Wgt. precision	0.8017	0.2034	0.0451
MaxAbsScaler	Accuracy	0.8322	**0.3935**	0.0084
	Avg. AUC ROC	0.7444	0.6874	0.4976
	Wgt. AUC ROC	0.9079	0.6902	0.5003
	Avg. precision	0.5533	0.2371	0.0102
	Wgt. precision	0.8255	0.6741	0.2457

Tabulka 6.13: Normalizace a její dopad na CTU92 dataset při Argus kódování a vlastním výběru příznaků

Normalizace	Metrika	Algoritmus			
		SVM	kNN	NB	KMeans
Žádná	Accuracy	0.2122	0.1762	0.2122	**0.0141**
	Avg. AUC ROC	0.5000	0.5196	0.5002	0.5032
	Wgt. AUC ROC	0.5000	0.5329	0.5018	0.5020
	Avg. precision	0.0048	0.0628	0.0095	0.0136
	Wgt. precision	0.0450	0.1739	0.0836	0.1406
Gaussian	Accuracy	**0.7636**	**0.7701**	0.2322	0.0028
	Avg. AUC ROC	0.7334	0.7736	0.7228	0.4988
	Wgt. AUC ROC	0.8658	0.8717	0.6082	0.4942
	Avg. precision	0.5525	0.6154	0.3129	0.0017
	Wgt. precision	0.7362	0.7626	0.5611	0.0075
MinMaxScaler	Accuracy	0.7311	0.7638	0.3062	0.0024
	Avg. AUC ROC	0.6825	0.7707	0.7109	0.4960
	Wgt. AUC ROC	0.8429	0.8686	0.6417	0.4954
	Avg. precision	0.5182	0.5906	0.2975	0.0018
	Wgt. precision	0.6892	0.7557	0.5126	0.0041
RobustScaler	Accuracy	0.7117	0.7535	0.1976	0.0016
	Avg. AUC ROC	0.7227	0.7685	0.5102	0.4970
	Wgt. AUC ROC	0.8378	0.8630	0.5078	0.4992
	Avg. precision	0.5733	0.5813	0.0651	0.0224
	Wgt. precision	0.7262	0.7504	0.1336	0.1272
MaxAbsScaler	Accuracy	0.7344	0.7549	**0.3272**	0.0035
	Avg. AUC ROC	0.6837	0.7641	0.7075	0.4944
	Wgt. AUC ROC	0.8453	0.8637	0.6505	0.4962
	Avg. precision	0.4938	0.5938	0.2989	0.0068
	Wgt. precision	0.6827	0.7498	0.5146	0.1369

Tabulka 6.14: Normalizace a její dopad na IDS17 dataset při Argus kódování a vlastním výběru příznaků

Normalizace	Metrika	Algoritmus		
		kNN	NB	KMeans
Žádná	Accuracy	0.3847	0.4651	**0.0097**
	Avg. AUC ROC	0.5246	0.5017	0.4997
	Wgt. AUC ROC	0.5572	0.5001	0.4997
	Avg. precision	0.0896	0.0300	0.0190
	Wgt. precision	0.3344	0.2180	0.2213
Gaussian	Accuracy	**0.7966**	0.2700	0.0055
	Avg. AUC ROC	0.7415	0.6950	0.5015
	Wgt. AUC ROC	0.8562	0.6152	0.5000
	Avg. precision	0.5683	0.2573	0.0114
	Wgt. precision	0.7754	0.6373	0.0731
MinMaxScaler	Accuracy	0.7536	0.5895	0.0019
	Avg. AUC ROC	0.7110	0.6517	0.5041
	Wgt. AUC ROC	0.8310	0.7303	0.4932
	Avg. precision	0.5011	0.2262	0.0038
	Wgt. precision	0.7285	0.6010	0.0021
RobustScaler	Accuracy	0.7795	0.4662	0.0010
	Avg. AUC ROC	0.7316	0.5043	0.5015
	Wgt. AUC ROC	0.8453	0.5030	0.5001
	Avg. precision	0.5356	0.0485	0.0128
	Wgt. precision	0.7578	0.2516	0.0331
MaxAbsScaler	Accuracy	0.7470	**0.6007**	0.0024
	Avg. AUC ROC	0.7048	0.6448	0.5024
	Wgt. AUC ROC	0.8269	0.7352	0.4939
	Avg. precision	0.4827	0.2258	0.0040
	Wgt. precision	0.6778	0.5905	0.0079

Tabulka 6.15: Normalizace a její dopad na WIDE dataset při Argus kódování a vlastním výběru příznaků

Normalizace	Metrika	Algoritmus		
		kNN	NB	KMeans
Žádná	Accuracy	0.4796	0.0473	0.0094
	Avg. AUC ROC	0.526	0.5000	0.5012
	Wgt. AUC ROC	0.5181	0.5000	0.5001
	Avg. precision	0.0754	0.0083	0.0107
	Wgt. precision	0.4488	0.4018	0.4315
Gaussian	Accuracy	**0.7307**	0.0186	0.0015
	Avg. AUC ROC	0.6028	0.6143	0.5032
	Wgt. AUC ROC	0.7685	0.5087	0.4939
	Avg. precision	0.2446	0.1145	0.0031
	Wgt. precision	0.7193	0.7618	0.1187
MinMaxScaler	Accuracy	0.6999	**0.1964**	0.0029
	Avg. AUC ROC	0.5776	0.5938	0.5015
	Wgt. AUC ROC	0.7283	0.5856	0.4996
	Avg. precision	0.2224	0.0759	0.0116
	Wgt. precision	0.6807	0.7009	0.6164
RobustScaler	Accuracy	**0.7346**	0.1397	0.0002
	Avg. AUC ROC	0.6103	0.5002	0.5000
	Wgt. AUC ROC	0.7728	0.5015	0.5000
	Avg. precision	0.2693	0.0186	0
	Wgt. precision	0.7238	0.5553	0
MaxAbsScaler	Accuracy	0.6932	**0.1967**	0.0027
	Avg. AUC ROC	0.5731	0.5913	0.5009
	Wgt. AUC ROC	0.7186	0.5847	0.4977
	Avg. precision	0.2142	0.0582	0.003
	Wgt. precision	0.6703	0.6294	0.0438

Tabulka 6.16: Transformace a její dopad na CTU06 dataset

Transformace	Metrika	Algoritmus								
		SVM	kNN	NB	KMeans	C4.5	REPTree	CART	AdaBst.	RForest
Žádná	Accuracy	0.4071	0.1438	0.2181	0.0139	0.9208	0.9112	0.8496	0.9273	0.9448
	Avg. AUC ROC	0.5000	0.5129	0.508	0.5025	0.9607	0.9700	0.8516	0.9200	0.9444
	Wgt. AUC ROC	0.5000	0.5234	0.5294	0.5012	0.9848	0.9889	0.9201	0.9600	0.9700
	Avg. precision	0.0678	0.0625	0.028	0.0123	0.8846	0.8486	0.7444	0.9094	0.9197
	Wgt. precision	0.1657	0.1427	0.1053	0.1088	0.9253	0.9142	0.8744	0.9274	0.9449
Uniform	Accuracy	0.4071	0.2936	0.4071	0.0084	0.9289	0.9121	0.9132	0.9349	0.9453
	Avg. AUC ROC	0.5000	0.5165	0.5000	0.4987	0.9754	0.9763	0.9172	0.9243	0.9454
	Wgt. AUC ROC	0.5000	0.5184	0.5000	0.4986	0.9871	0.9879	0.9532	0.9642	0.9702
	Avg. precision	0.0678	0.1956	0.0678	0.0086	0.8879	0.8654	0.8577	0.9227	0.9211
	Wgt. precision	0.1657	0.2869	0.1657	0.2112	0.9287	0.9128	0.9139	0.9358	0.9453
YeoJohnson	Accuracy	0.4071	0.2936	0.4071	0.0084	0.9290	0.9124	0.9148	0.9320	0.9457
	Avg. AUC ROC	0.5000	0.5165	0.5000	0.4987	0.9756	0.9764	0.9175	0.9213	0.9467
	Wgt. AUC ROC	0.5000	0.5184	0.5000	0.4986	0.9872	0.9879	0.9541	0.9627	0.9704
	Avg. precision	0.0678	0.1956	0.0678	0.0086	0.8868	0.8656	0.8596	0.9152	0.9213
	Wgt. precision	0.1657	0.2869	0.1657	0.2112	0.9288	0.9132	0.9154	0.9321	0.9456
Normal	Accuracy	0.4071	0.2936	0.4071	0.0084	0.9286	0.9122	0.9135	0.9304	0.9445
	Avg. AUC ROC	0.5000	0.5165	0.5000	0.4987	0.9751	0.9764	0.9174	0.9202	0.9449
	Wgt. AUC ROC	0.5000	0.5184	0.5000	0.4986	0.9870	0.9879	0.9534	0.9617	0.9697
	Avg. precision	0.0678	0.1956	0.0678	0.0086	0.8872	0.8659	0.8579	0.9134	0.9235
	Wgt. precision	0.1657	0.2869	0.1657	0.2112	0.9284	0.9130	0.9142	0.9310	0.9448
Log	Accuracy	0.4071	0.2936	0.4071	0.0084	0.9287	0.9124	0.9153	0.9284	0.9449
	Avg. AUC ROC	0.5000	0.5165	0.5000	0.4987	0.9748	0.9764	0.9196	0.9217	0.9460
	Wgt. AUC ROC	0.5000	0.5184	0.5000	0.4986	0.9869	0.9879	0.9543	0.9606	0.9700
	Avg. precision	0.0678	0.1956	0.0678	0.0086	0.8893	0.8656	0.8649	0.9109	0.9202
	Wgt. precision	0.1657	0.2869	0.1657	0.2112	0.9285	0.9132	0.9156	0.9287	0.9449
BoxCox	Accuracy	0.4071	0.4071	0.0948	0.0084	0.9195	0.9124	0.9102	0.9293	0.9447
	Avg. AUC ROC	0.5000	0.5000	0.5000	0.4987	0.9676	0.9764	0.9171	0.9212	0.9452
	Wgt. AUC ROC	0.5000	0.5000	0.5000	0.4986	0.9820	0.9879	0.9517	0.9612	0.9699
	Avg. precision	0.0678	0.0678	0.0158	0.0086	0.8748	0.8656	0.8558	0.9139	0.9181
	Wgt. precision	0.1657	0.1657	0.009	0.2112	0.9221	0.9132	0.9122	0.9299	0.9445

Tabulka 6.17: Transformace a její dopad na WIDE dataset při Argus kódování a vlastním výběru příznaků

Transformace	Metrika	Algoritmus		
		kNN	NB	KMeans
Žádná	Accuracy	0.4796	0.0473	0.0094
	Avg. AUC ROC	0.5260	0.5000	0.5012
	Wgt. AUC ROC	0.5181	0.5000	0.5001
	Avg. precision	0.0754	0.0083	0.01070
	Wgt. precision	0.4488	0.4018	0.4315
Uniform	Accuracy	0.4763	0.0633	0.0103
	Avg. AUC ROC	0.5262	0.5000	0.5001
	Wgt. AUC ROC	0.5165	0.5000	0.5003
	Avg. precision	0.0682	0.0083	0.0110
	Wgt. precision	0.4461	0.4018	0.4366
YeoJohnson	Accuracy	0.4714	0.0633	0.0103
	Avg. AUC ROC	0.5078	0.5000	0.5001
	Wgt. AUC ROC	0.5097	0.5000	0.5003
	Avg. precision	0.0414	0.0083	0.0110
	Wgt. precision	0.4379	0.4018	0.4366
Normal	Accuracy	0.4783	0.0633	0.0103
	Avg. AUC ROC	0.5263	0.5000	0.5001
	Wgt. AUC ROC	0.5183	0.5000	0.5003
	Avg. precision	0.0688	0.0083	0.0110
	Wgt. precision	0.4484	0.4018	0.4366
Log	Accuracy	0.4773	0.0633	0.0103
	Avg. AUC ROC	0.5257	0.5000	0.5001
	Wgt. AUC ROC	0.5176	0.5000	0.5003
	Avg. precision	0.0719	0.0083	0.0110
	Wgt. precision	0.4479	0.4018	0.4366
BoxCox	Accuracy	0.4739	0.0014	0.0014
	Avg. AUC ROC	0.5000	0.5000	0.5000
	Wgt. AUC ROC	0.5000	0.5000	0.5000
	Avg. precision	0.0162	0	0
	Wgt. precision	0.4837	0	0

6.9 Algoritmy strojového učení

Tabulky 6.18 a 6.19 shrňují výsledky vybraných algoritmů strojového učení v nastavení uvedeném v sekci 5.7 při výběru příznaků z konfiguračního souboru. Metriky jsou reprezentovány pomocí desetiného čísla a reprezentují

výsledek testovacího datasetu po 5-fold křížové validaci, navíc jsou uvedené i časy potřebné pro natrénování modelu a predikce nových záznamů (v tabulce označeno jako Fit time a Predict time). V případě algoritmů C4.5 a REPTree je uveden pouze průměrný čas pro jeden process křížové validace. Pro stromové algoritmy bylo použito kódování Ordinal a pro algoritmus k-NN bylo použito kódování Argus, normalizace Gaussian a transformace Normal.

Z tabulek výsledků 6.18 a 6.19 vychází průměrně nejlépe algoritmus C4.5 s průměrným accuracy přes všechny datasety 0.9255, hned za ním je REPTree s průměrným accuracy 0.9205 a na třetím místě se umístil Random Forest s průměrným accuracy 0.8733.

Z pohledu časové složitosti algoritmu se čas potřebný pro vytvoření modelu odvíjí od velikosti vstupních dat. Nejvíce času pro vytvoření modelu a klasifikování nových záznamů potřebuje Adaboost, naopak je to v případě CART modelu, který je nejrychlejší a dokáže rychle klasifikovat i poměrně velký dataset.

Z matice záměn lze vyčíst, že nejčastěji se mylně klasifikují záznamy, které spadají do jedné obecnější skupiny provozu. Například dochází více mylným klasifikacím mezi DNS, DNS.Facebook, DNS.Google a DNS.GoogleServices. Stejné tvrzení platí pro ostatní HTTP, SSL a QUIC skupiny. Tudíž záměny docházejí alespoň v rámci stejného druhu.

Tabulka 6.18: Výsledky vybraných algoritmů pro klasifikaci internetového provozu, část první

Algoritmus	Metrika	Dataset						
		IDS17	CTU92	CTU91	CTU08	CTU07	CTU06	WIDE
k-NN	Accuracy	0.8224	0.7701	0.8468	0.9687	0.9607	0.9032	0.7041
	Avg. AUC ROC	0.7437	0.7736	0.7697	0.7941	0.7854	0.9048	0.6016
	Wgt. AUC ROC	0.8607	0.8717	0.9157	0.9704	0.9535	0.9477	0.7531
	Avg. precision	0.5815	0.6154	0.5936	0.6076	0.6043	0.8350	0.2456
	Wgt. precision	0.8096	0.7626	0.8415	0.9617	0.9492	0.9035	0.6914
	Fit time	54.48	0.04	2.93	51.85	15.75	0.06	371.56
	Predict time	307.77	1.5	33.39	147.48	70.06	1.81	575.36
CART	Accuracy	0.8884	0.6389	0.7796	0.9600	0.9588	0.8040	0.6314
	Avg. AUC ROC	0.7881	0.7672	0.7420	0.7928	0.7512	0.8149	0.7898
	Wgt. AUC ROC	0.9147	0.8041	0.8785	0.9647	0.9549	0.8967	0.8076
	Avg. precision	0.7412	0.5714	0.5350	0.6888	0.5794	0.6716	0.5780
	Wgt. precision	0.8835	0.6956	0.8092	0.9569	0.9519	0.8555	0.9334
	Fit time	71.84	0.29	7.64	34.87	13.85	0.23	27.81
	Predict time	2.26	0.17	1.09	2.37	1.33	0.12	6.09
C4.5	Accuracy	0.8951	0.8538	0.8918	0.97135	0.9702	0.9123	0.9840
	Avg. AUC ROC	0.9065	0.9345	0.9062	0.9381	0.9341	0.9544	0.9727
	Wgt. AUC ROC	0.8910	0.9375	0.9038	0.9959	0.9406	0.9833	0.9844
	Avg. precision	0.7589	0.7136	0.6820	0.75665	0.7278	0.8025	0.8579
	Wgt. precision	0.7989	0.7114	0.6353	0.9790	0.8367	0.9112	0.8632
	Time CV	91.43	1.56	22.26	100.47	34.88	2.66	108.77

Tabulka 6.19: Výsledky vybraných algoritmů pro klasifikaci internetového provozu, část druhá

Algoritmus	Metrika	Dataset						
		IDS17	CTU92	CTU91	CTU08	CTU07	CTU06	WIDE
REPTree	Accuracy	0.8882	0.8353	0.8872	0.9711	0.9695	0.909	0.9835
	Avg. AUC ROC	0.9033	0.9504	0.9276	0.9481	0.9557	0.9682	0.9756
	Wgt. AUC ROC	0.8549	0.9540	0.9254	0.9960	0.9604	0.9875	0.9480
	Avg. precision	0.7220	0.6910	0.6442	0.7540	0.7085	0.7733	0.8054
	Wgt. precision	0.5832	0.6978	0.6038	0.9790	0.8414	0.8596	0.8911
	Time CV	33.79	1.05	16.22	39.97	11.29	0.88	52.65
RamdomForest	Accuracy	0.7665	0.7422	0.8319	0.9651	0.9666	0.8856	0.9555
	Avg. AUC ROC	0.9068	0.8100	0.7507	0.7982	0.7534	0.8677	0.6804
	Wgt. AUC ROC	0.8205	0.8568	0.9057	0.9663	0.9535	0.9383	0.9504
	Avg. precision	0.8981	0.7350	0.6489	0.6288	0.5794	0.8186	0.6087
	Wgt. precision	0.8139	0.7780	0.8499	0.9563	0.9467	0.8974	0.9516
	Fit time	15.96	20.31	174.15	469.93	279.97	17.34	167.5
	Predict time	0.89	5.08	539.3	94.34	65.3	4.84	35.48
AdaBoost	Accuracy	0.7310	0.7111	0.7839	0.9627	0.9631	0.8562	0.9759
	Avg. AUC ROC	0.8352	0.7485	0.7075	0.7882	0.7219	0.8201	0.8067
	Wgt. AUC ROC	0.7653	0.8348	0.8785	0.9639	0.9495	0.9213	0.9770
	Avg. precision	0.8122	0.7124	0.6178	0.6580	0.5477	0.7761	0.8749
	Wgt. precision	0.6775	0.7343	0.8104	0.9524	0.9421	0.8669	0.9743
	Fit time	15.69	114.85	878.57	2582.27	1493.36	106.65	2160.87
	Predict time	0.82	7.41	66.18	37.63	87.29	6.49	109.08

6.10 Algoritmy pro detekci anomálií

Z tabulky 6.3 je vidět kolik anomálií jaký dataset obsahuje. Bohužel při volbě datasetů se nebrala v potaz odlišnost způsobu získaných labelů pro identifikování škodlivé komunikace či nechtěného chování, souhrně označováno jako anomálie, a tudíž i to, že každý daset obsahuje jiný způsob určování těchto anomálií. Například v datasetu CTU06 se mezi anomálie řadí pouze nesprávné použití protokolu pro přístup na server, naopak dataset CTU08 obsahuje obrovské množství komunikace označené jako útok TROJAN. Proto i výsledky detekce anomálií se značně liší, viz tabulka 6.20. Nulová úspěšnost detekce anomálií u datasetu CTU06 je způsobena faktem, že dataset obsahuje pouze jednu jedinou anomálií po čištění dat. Algoritmus LocalOutlierFactor si vedl velmi dobře v detekci anomílií u datasetu IDS17, avšak u ostatních datasetů jeho úspěšnost výrazně klesla. I tak LocalOutlierFactor dosahuje nejlepších výsledků pro všechny datasety kromě WIDE. U datastu WIDE bylo nejlepší detekce anomálií dosaženo pomocí algoritmu AutoEncoder.

Výsledky u dvou datasetů, jmenotivě CIC17 a WIDE, dokazují, že lze docílit úspěšné detekce anomálií pomocí zvolených nesupervizovaných algoritmů. Důvod proč si algoritmy nepočínají moc dobře při detekci anomálií v případě datasetů CTU je pravěpodobně nevhodně zvolený způsob získání ground truth labelů pro tuto úlohu.

Tabulka 6.20: Výsledky vybraných algoritmů pro detekci anomálií

Algoritmus	Metrika	Dataset						
		IDS17	CTU92	CTU91	CTU08	CTU07	CTU06	WIDE
LocalOutlierFactor	Recall	0.9314	0.1622	0.0402	0.1281	0.1279	0	0.0371
	AUC PRC	0.1880	0.0068	0.0055	0.2641	0.2169	0	0.0051
	F1 Score	0.3294	0.0162	0.0040	0.1748	0.1623	0	0.0045
	Fit time	30.28	0.22	2.37	7.32	2.93	0.23	175.59
	Predict time	7.37	0.21	1.41	3.16	1.33	0.11	46.33
ABOD	Recall	0.0100	0	0.004	0.0082	0.0078	0	0.6639
	AUC PRC	0.0236	0.0064	0.0056	0.2621	0.2159	0	0.0248
	F1 Score	0.0142	0	0.0031	0.0159	0.0149	0	0.0659
	Fit time	101.92	3.36	22.14	84.45	43.47	4.01	801.01
	Predict time	40.77	1.47	13.97	51.19	27.41	1.87	392.41
AutoEncoder	Recall	0.0904	0.0541	0.0763	0.0968	0.0949	0	0.7572
	AUC PRC	0.0233	0.0063	0.0055	0.2615	0.2152	0	0.0312
	F1 Score	0.0343	0.0066	0.0079	0.1398	0.1299	0	0.0753
	Fit time	108.61	10.41	48.41	168.37	88.35	10.74	196.91
	Predict time	1.623	0.096	0.59	2.83	1.38	0.13	4.23

6.11 Kaskádní řešení

Jedním z úkolů diplomové práce bylo pokusit se vylepšit některý postup. Proto tuto úlohu byla zvolena myšlenka využití labelů pro určení kategorie k lepší asociaci flow ke jménu aplikace a vylepšení detekce anomálií.

Tato myšlenka vznikla během procesu získávání labelů, kdy nástroj nDPI vrací právě dva druhy labelů, pro aplikaci a pro kategorii. Myšlenku navíc utvrdil pokus, během kterého bylo nejlepším algoritmům pro klasifikaci provozu přidán příznak určující kategorii. Pokus byl proveden na třech datasetech CTU06, CTU91, CTU92. Úspěšnost asociace jména aplikace a flow se zvýšila průměrně o 8 %, viz tabulka 6.21. Například v případě datasetu CTU92 algoritmu Random Forest se úspěšnost metriky accuracy zvýšila o 15 %. V případě algoritmů C4.5 a REPTree se úspěšnost accuracy zvýšila v průměru o 7 % v případě Random Forestu až o 9.5 %.

Obdobný pokus s příznakem určující kategorii byl provoden pro detekci anomálií, ale ukázalo se, že úspěšnost detekce anomálií se nikterak nenavýšila. Navíc kvůli nerozhodnutosti, který algoritmus pro detekci anomálií použít jako finální se od detekce anomálií upustilo a zaměřilo se pouze na klasifikaci provozu, který je taktéž důležitý.

Tabulka 6.21: Výsledky vybraných algoritmů a datasetů při přidaném příznaků určující kategorii

Algoritmus	Metrika	Výsledky s příznakem kategorie			Původní výsledky		
		CTU06	CTU91	CTU92	CTU06	CTU91	CTU92
C4.5	Accuracy	0.9667	0.9641	0.9371	0.9123	0.8918	0.8538
	Avg. AUC ROC	0.9845	0.9741	0.9741	0.9544	0.9062	0.9345
	Wgt. AUC ROC	0.9902	0.9690	0.9876	0.9833	0.9038	0.9375
	Avg. precision	0.9142	0.8748	0.8736	0.8025	0.682	0.7136
	Wgt. precision	0.9426	0.8413	0.8511	0.9112	0.6353	0.7114
REPTree	Accuracy	0.9611	0.9619	0.9318	0.9090	0.8872	0.8353
	Avg. AUC ROC	0.9884	0.9822	0.9827	0.9682	0.9276	0.9504
	Wgt. AUC ROC	0.9915	0.9832	0.9925	0.9875	0.9254	0.9540
	Avg. precision	0.9012	0.8677	0.8607	0.7733	0.6442	0.6910
	Wgt. precision	0.9330	0.8226	0.8075	0.8596	0.6038	0.6978
RamdomForest	Accuracy	0.9437	0.9029	0.8977	0.8856	0.8319	0.7422
	Avg. AUC ROC	0.9234	0.8588	0.9037	0.8677	0.7507	0.8100
	Wgt. AUC ROC	0.9702	0.9485	0.9451	0.9383	0.9057	0.8568
	Avg. precision	0.9043	0.8204	0.8791	0.8186	0.6489	0.7350
	Wgt. precision	0.9547	0.9324	0.9156	0.8974	0.8499	0.7780

Jako hlavní stavební prvek kaskádního řešení byl zvolen algoritmus Random Forest, z důvodu nejvyšší úspěšnosti predikce kategorie. Výsledný kaskádní algoritmus je sestaven z iterativního spouštění algoritmu Random Forest, kdy po prvním průchodu je trénovací dataset rozšířen o příznak nesoucí ground truth labely kategorie a testovací dataset je rozšířen o příznak odpovídající predikovaným hodnotám kategorie.

Tabulka 6.22 obsahuje výsledky kaskádního algoritmu jak pro první iteraci při předikci kategorie, tak pro druhou, finální, iteraci pro predikci jména aplikace. Ukázalo se, že u některých datasetů je dosaženo lepší klasifikace záznamů, než při použití samostatného algoritmu. Avšak čím lepší je predikce kategorie záznamu o to víc je úspěšnější predikce aplikace, například u datasetu WIDE nestačila ani 98.5 % úspěšnost predikce kategorie.

Tabulka 6.22: Výsledky kaskádního algoritmu s klasifikátorem Random Forest

Fáze	Metrika	Dataset						
		IDS17	CTU92	CTU91	CTU08	CTU07	CTU06	WIDE
Kategorie	Accuracy	0.8919	0.9269	0.9266	0.9848	0.9743	0.9591	0.9853
	Avg. AUC ROC	0.7661	0.8771	0.7922	0.8237	0.798	0.9584	0.74475
	Wgt. AUC ROC	0.901	0.9194	0.9488	0.9756	0.9594	0.9698	0.97205
	Avg. precision	0.8521	0.9085	0.6777	0.8507	0.8294	0.9624	0.73245
	Wgt. precision	0.8929	0.924	0.9244	0.9815	0.9716	0.9593	0.98475
	Fit time	419.12	13.16	93.37	416.13	229.42	12.68	335.29
	Predict time	16.58	0.58	4.92	12.25	7.78	0.52	25.87
Aplikace	Accuracy	0.8746	0.8876	0.9051	0.9842	0.9721	0.9414	0.9709
	Avg. AUC ROC	0.7576	0.8456	0.7986	0.7797	0.7946	0.9394	0.6963
	Wgt. AUC ROC	0.9	0.9362	0.9471	0.9754	0.9587	0.9681	0.96955
	Avg. precision	0.7253	0.8339	0.768	0.748	0.8463	0.9169	0.631
	Wgt. precision	0.8627	0.8852	0.9018	0.9771	0.9648	0.9422	0.96695
	Fit time	473.25	15.33	113.05	272.33	171.15	16.77	459.94
	Predict time	264.93	1.01	9.13	21.41	8.99	1.11	77.22

Závěr

Cílem práce bylo nastudovat postupy strojového učení pro klasifikaci internetového provozu. Mezi studované postupy spadal vhodný výběr algoritmů, předzpracování dat a výběr příznaků pro úspěšnou klasifikaci provozu.

Z klasifikačních algoritmů se jako nejvhodnější ukázaly stromové algoritmy, přesněji C4.5, REPTree a Random Forest. Spolu s nimi bylo vyzkoušeno dalších 6 algoritmů a vlastní algoritmus založený na kaskádní implementaci algoritmu Random Forest. Výsledky na zvolených datasetech ukázaly, že dva často zmiňované algoritmy v jiných pracích zabývající se stejnou problematikou nedosahují dostatečné úspěšnosti měřené v accuracy, přesněji se jedná o algoritmy KMeans a Naive Bayes (NB).

Během fáze předzpracování dat bylo dokázané tvrzení, že stromové modely nepotřebují normalizovat ani transformovat vstupní data. Naopak nutnost normalizace a transformace dat byla dokázána pro zvolené algoritmy SVM, NB, k-NN a KMeans. Normalizace vstupních dat zvýšila úspěšnost těchto 4 algoritmů závratně. Překvapením bylo, že transformace dat samotná nepomohla k lepším výsledkům, pouze pokud se použila společně s normalizací. Avšak ani při kombinaci obouch technik se úspěšnost algoritmů už o moc nezvětšila. Průměrně se metrika accuracy při použití transformace spolu s normalizací zvýšila o necelé jedno procento.

Při výběru příznaků byly využity algoritmy, které vracejí určitým způsobem důležitost příznaků. Nakonec vznikl vlastní seznam příznaků, které vracejí s použitými algoritmy jedny z nejlepších výsledků. Mezi vybrané příznaky spadá například doba aktivního a neaktivního přenosu paketů, efektivita komunikace, maximální a minimální velikost paketů v bajtech, čas potřebný pro navázání spojení a čísla Portů.

Mezi další možné kroky této práce spadá nalezení lepšího modulu pro klasifikování záznamu do obecnějších vlastností komunikace než je Random Forest. Jelikož čím přesnější je určení vlastností provozu, tím přesněji je následně určena aplikace, která vygenerovala provoz.

Literatura

[1] Bhande, A.: What is underfitting and overfitting in machine learning and how to deal with it. 2018, [vid. 10 Července 2019]. Dostupné z: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76

[2] Pant, A.: Workflow of a Machine Learning project. Leden 2019, [vid. 10 Května 2019]. Dostupné z: https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94

[3] Zararsiz, G.; Elmali, F.; Ozturk, A.: Bagging Support Vector Machines for Leukemia Classification. *IJCSI International Journal of Computer Science Issues*, ročník 9, 11 2012: s. 355–358.

[4] Pant, A.: Machine Learning Crash Course. Srpen 2017, [vid. 2019-04-07]. Dostupné z: https://medium.com/@danishkhan.jamia/machine-learning-5e0bd87311a4

[5] Provost, F.; Fayyad, U.: A Survey of Methods for Scaling Up Inductive Algorithms. *Knowledge Discovery and Data Mining*, ročník 3, Září 1999: s. 1–42.

[6] Liu, F. T.; Ting, K.; Zhou, Z.-H.: Isolation-Based Anomaly Detection. *ACM Transactions on Knowledge Discovery From Data - TKDD*, ročník 6, 03 2012: s. 1–39.

[7] Ahire, J. B.: The Artificial Neural Networks handbook: Part 1. *Medium*, Aug 2018. Dostupné z: https://medium.com/coinmonks/the-artificial-neural-networks-handbook-part-1-f9ceb0e376b4

[8] Dertat, A.: Applied Deep Learning - Part 3: Autoencoders. *Medium*, Jun 2018. Dostupné z: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798#f686

[9] Markham, K.: Simple guide to confusion matrix terminology. *Data School*, Květen 2019. Dostupné z: https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology

[10] Narkhede, S.: Understanding AUC - ROC Curve. *Medium*, Květen 2019. Dostupné z: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[11] Introduction to the precision-recall plot. Jun 2015, [Online; accessed 7. Dec. 2019]. Dostupné z: https://classeval.wordpress.com/introduction/introduction-to-the-precision-recall-plot

[12] learn 0.21.3 documentation, S.: Cross-validation: evaluating estimator performance. Oct 2019, [Online; accessed 24. Oct. 2019]. Dostupné z: https://scikit-learn.org/stable/_images/grid_search_cross_validation.png

[13] Tutorial: Learning Curves for Machine Learning in Python for Data Science. Jan 2018, [Online; accessed 23. Oct. 2019]. Dostupné z: https://www.dataquest.io/blog/learning-curves-machine-learning

[14] Litschmannová, M.: Průzkumová analýza jednorozměrných dat (Teorie). 2011, [Online; vid. 22. Dubna 2019]. Dostupné z: http://gisak.vsb.cz/pan/source/data/skolenia/eda/Pruzkumova_analyza_dat.pdf

[15] Rathi, A.: Dealing with Noisy Data in Data Science. *Medium*, Jul 2019. Dostupné z: https://medium.com/analytics-vidhya/dealing-with-noisy-data-in-data-science-e177a4e32621

[16] Chandola, V.; Banerjee, A.; Kumar, V.: Anomaly Detection: A Survey. *ACM Comput. Surv.*, ročník 41, 07 2009: s. 1–72, doi:10.1145/1541880.1541882.

[17] Molnar, C.: *Interpretable Machine Learning*. Jul 2019. Dostupné z: https://christophm.github.io/interpretable-ml-book

[18] Feng, C.; Hongyue, W.; Lu, N.; aj.: Log-transformation and its implications for data analysis. *Shanghai archives of psychiatry*, ročník 26, 04 2014: s. 105–109.

[19] Holčík, J.; Komenda, M.; aj.: *Matematická biologie: e-learningová učebnice: Standardizace dat*. Brno: Masarykova univerzita, první vydání, Aug 2015, ISBN 978-80-210-8095-9, [Online; vid 10. Února 2019]. Dostupné z: http://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--vicerozmerne-metody-pro-analyzu-dat--vicerozmerna-rozdeleni-pravdepodobnosti--transformace-dat--standardizace-dat

[20] Alelyani, S.; Tang, J.; Liu, H.: Feature Selection for Clustering: A Review. In *Data Clustering: Algorithms and Applications*, 08 2013, s. 32–55.

[21] Valenti, S.; Rossi, D.; Dainotti, A.; aj.: *Reviewing Traffic Classification.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, s. 123–147.

[22] Pietrzyk, M.; Plissonneau, L.; Urvoy-Keller, G.; aj.: On profiling residential customers. In *Traffic Monitoring and Analysis*, 2011, ISBN 978-3-642-20305-3, s. 1–14.

[23] Geyer, F.; Carle, G.: Learning and Generating Distributed Routing Protocols Using Graph-Based Deep Learning. In *Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, Big-DAMA '18, Srpen 2018, ISBN 978-1-4503-5904-7, s. 40–45.

[24] Asadi, A.; Abdollahi Azgomi, M.: Modeling the inter-arrival time of packets in network traffic and anomaly detection using the Zipf's law. *Journal of Information Systems and Telecommunication*, ročník 6, Březen 2018: s. 76–87.

[25] Boutaba, R.; Salahuddin, M. A.; Limam, N.; aj.: A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, ročník 9, č. 1, Červen 2018: str. 16, ISSN 1869-0238.

[26] Ayoubi, S.; Limam, N.; Salahuddin, M. A.; aj.: Machine Learning for Cognitive Network Management. *IEEE Communications Magazine*, ročník 56, č. 1, Leden 2018: s. 158–165, ISSN 0163-6804.

[27] ICT facts and figures 2017. Červenec 2017. Dostupné z: `https://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2017.pdf`

[28] Measuring the Information Society Report. 2018. Dostupné z: `https://www.itu.int/en/ITU-D/Statistics/Documents/publications/misr2018/MISR-2018-Vol-1-E.pdf`

[29] By The Numbers, Projecting the future of digital transformation (2017 – 2022). Únor 2019. Dostupné z: `https://www.cisco.com/c/en/us/solutions/service-provider/vni-network-traffic-forecast/vni-forecast-info.html`

[30] The Wireshark Foundation: About Wireshark. Dostupné z: `https://www.wireshark.org/about.html`

[31] The Tcpdump Group: Tcpdump and Libpcap. Dostupné z: `https://www.tcpdump.org/#latest-release`

[32] Moore, A. W.; Zuev, D.; Crogan, M. L.: Discriminators for Use in Flow-Based Classification. Srpen 2005: str. 3.

[33] Everitt, B.; Skrondal, A.: *The Cambridge Dictionary of Statistics*. Cambridge University Press, 2010, ISBN 9780521766999, s. 314, 440.

[34] Brownlee, J.: Overfitting and Underfitting With Machine Learning Algorithms. Březen 2018, [vid. 15 Května 2019]. Dostupné z: `https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/`

[35] Wu, X.; Kumar, V.; Ross Quinlan, J.; aj.: Top 10 algorithms in data mining. *Knowledge and Information Systems*, ročník 14, č. 1, Leden 2008: s. 1–37.

[36] Salzberg, S. L.: C4.5: Programs for Machine Learning by J. Ross Quinlan, Morgan Kaufmann Publishers, Inc., 1993. *Machine Learning*, ročník 16, č. 3, Září 1994: s. 235–240.

[37] C4.5 Decision Trees with Missing Data. [vid. 15 Června 2019]. Dostupné z: `https://inferate.blogspot.com/2015/06/c45-decision-trees-with-missing-data.html`

[38] Quinlan, J. R.: *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, ISBN 1-55860-238-0.

[39] Breiman, L.: *Classification and regression trees*. Wadsworth statistics/probability series, Wadsworth International Group, 1984, ISBN 9780534980535.

[40] Devasena, L.; Vasconcelos, G. C.; Adeodato, P. J. L.; aj.: Comparative Analysis of Random Forest, REP Tree and J48 Classifiers for Credit Risk Prediction. *International Journal of Computer Applications*, 2015: s. 30–36.

[41] Vapnik, V.; Cortes, C.: Support-Vector Networks. *Machine Learning*, ročník 20, č. 3, Září 1995: s. 273–297, ISSN 1573-0565.

[42] Cristianini, N.; Shawe-Taylor, J.; Shawe-Taylor, D.: *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Inc and Cambridge University Press, 2000, ISBN 9780521780193.

[43] Muller, K. R.; Mika, S.; Ratsch, G.; aj.: An Introduction to Kernel-based Learning Algorithms. *Trans. Neur. Netw.*, ročník 12, č. 2, Březen 2001: s. 181–201, ISSN 1045-9227.

[44] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; aj.: Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, ročník 12, 2011: s. 2825–2830.

[45] Jaiswal, R. C.: *Investigation on Traffic Modelling and Recognition Techniques in Computer Networks*. Dizertační práce, Savitribai Phule Pune University, Prosinec 2015.

[46] Rokach, L.: Ensemble-based classifiers. *Artificial Intelligence Review*, ročník 33, č. 1, Únor 2010: s. 1–39.

[47] Clearwater, S. H.; Cheng, T.-P.; Hirsh, H.; aj.: Incremental Batch Learning. In *Proceedings of the Sixth International Workshop on Machine Learning*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, ISBN 1-55860-036-1, s. 366–370.

[48] Flovik, V.: How to use machine learning for anomaly detection and condition monitoring. *Medium*, Nov 2019. Dostupné z: `https://towardsdatascience.com/how-to-use-machine-learning-for-anomaly-detection-and-condition-monitoring-6742f82900d7`

[49] 2.7. Novelty and Outlier Detection — scikit-learn 0.21.3 documentation. Nov 2019, [Online; accessed 20. Nov. 2019]. Dostupné z: `https://scikit-learn.org/stable/modules/outlier_detection.html`

[50] Wenig, P.: Local Outlier Factor for Anomaly Detection. *Medium*, Aug 2019. Dostupné z: `https://towardsdatascience.com/local-outlier-factor-for-anomaly-detection-cc0c770d2ebe`

[51] Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; aj.: LOF: Identifying Density-Based Local Outliers. *ACM SIGMOD Record*, ročník 29, č. 2, 2000: s. 93–104.

[52] Liu, F. T.; Ting, K.; Zhou, Z.-H.: Isolation Forest. 01 2009, s. 413 – 422.

[53] Preiss, B.: *Data Structures and Algorithms with Object-Oriented Design Patterns in Java*. 01 2000.

[54] Aggarwal, C. C.: *Outlier Analysis*. Springer Publishing Company, Incorporated, druhé vydání, 2016, ISBN 3319475770, 9783319475776, 98 – 106 s.

[55] Landgrebe, T.; Duin, R.: A simplified extension of the area under the ROC to the multiclass domain. *Seventeenth Annual Symposium of the Pattern Recognition Association of South Africa*, Leden 2006: s. 241–245.

[56] Landgrebe, T. C.; Duin, R. P.: Approximating the multiclass ROC by pairwise analysis. *Pattern Recognition Letters*, ročník 28, č. 13, 2007: s. 1747 – 1758.

[57] Swalin, A.: Choosing the Right Metric for Evaluating Machine Learning Models. *Medium*, Apr 2019. Dostupné z: https://medium.com/usf-msds/choosing-the-right-metric-for-evaluating-machine-learning-models-part-2-86d5649a5428

[58] Performance Measures for Multi-Class Problems. Dec 2018, [Online; accessed 22. Sep. 2019]. Dostupné z: https://www.datascienceblog.net/post/machine-learning/performance-measures-multi-class-problems

[59] Precision-Recall — scikit-learn 0.21.3 documentation. Nov 2019, [Online; accessed 12. Oct. 2019]. Dostupné z: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html

[60] Precision-recall curves – what are they and how are they used? Nov 2019, [Online; accessed 20. Nov. 2019]. Dostupné z: https://acutecaretesting.org/en/articles/precision-recall-curves-what-are-they-and-how-are-they-used

[61] Saito, T.; Rehmsmeier, M.: The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*, ročník 10, 03 2015: s. 1–21.

[62] Anzanello, M. J.; Fogliatto, F. S.: Learning curve models and applications: Literature review and research directions. *International Journal of Industrial Ergonomics*, ročník 41, č. 5, 2011: s. 573 – 583, ISSN 0169-8141.

[63] Aich, A.: Bias-Variance Tradeoff in Machine Learning - An Overview. Jul 2019, [Online; accessed 27. Oct. 2019]. Dostupné z: https://www.knowledgehut.com/blog/data-science/bias-variance-tradeoff-in-machine-learning

[64] Brownlee, J.: Gentle Introduction to the Bias-Variance Trade-Off in Machine Learning. Mar 2016, [Online; accessed 2. Nov. 2019]. Dostupné z: https://machinelearningmastery.com/gentle-introduction-to-the-bias-variance-trade-off-in-machine-learning

[65] Machine Learning Explained: Regularization - Enhance Data Science. Jul 2017, [Online; accessed 8. Dec. 2019]. Dostupné z: http://enhancedatascience.com/2017/07/04/machine-learning-explained-regularization

[66] Service Name and Transport Protocol Port Number Registry. Jul 2019, [Online; vid. 21. Červenec 2019]. Dostupné z: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

[67] Moore, A. W.; Papagiannaki, K.: Toward the Accurate Identification of Network Applications. In *Proceedings of the 6th International Conference on Passive and Active Network Measurement*, PAM'05, Berlin, Heidelberg: Springer-Verlag, 2005, ISBN 3-540-25520-6, 978-3-540-25520-8, s. 41–54.

[68] Madhukar, A.; Williamson, C.: A Longitudinal Study of P2P Traffic Classification. In *14th IEEE International Symposium on Modeling, Analysis, and Simulation*, Září 2006, s. 179–188.

[69] Finamore, A.; Mellia, M.; Meo, M.; aj.: KISS: Stochastic Packet Inspection. In *Traffic Monitoring and Analysis*, editace M. Papadopouli; P. Owezarski; A. Pras, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, s. 117–125.

[70] Dashevskiy, M.; Luo, Z.: *Network Traffic Classification and Demand Prediction*. Prosinec 2014, s. 231–259.

[71] Sicker, D. C.; Ohm, P.; Grunwald, D.: Legal Issues Surrounding Monitoring During Network Research. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, New York, NY, USA: ACM, 2007, ISBN 978-1-59593-908-1, s. 141–148.

[72] Roche, V. P.; Arronategui, U.: *Behavioural Characterization for Network Anomaly Detection*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, s. 23–40.

[73] Karagiannis, T.; Papagiannaki, K.; Faloutsos, M.: BLINC: Multilevel Traffic Classification in the Dark. *SIGCOMM Comput. Commun. Rev.*, ročník 35, č. 4, Srpen 2005: s. 229–240, ISSN 0146-4833.

[74] Kim, H.; Claffy, K.; Fomenkov, M.; aj.: Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, New York, NY, USA: ACM, 2008, ISBN 978-1-60558-210-8, s. 11:1–11:12.

[75] Moore, A.; Zuev, D.: Discriminators for Use in Flow-Based Classification. Leden 2005.

[76] Gringoli, F.; Salgarelli, L.; Dusi, M.; aj.: GT: Picking Up the Truth from the Ground for Internet Traffic. *SIGCOMM Comput. Commun. Rev.*, ročník 39, č. 5, Říjen 2009: s. 12–18.

[77] Williams, N.; Zander, S.; Armitage, G.: A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification. *SIGCOMM Comput. Commun. Rev.*, ročník 36, č. 5, Říjen 2006: s. 5–16, ISSN 0146-4833.

[78] Nguyen, T. T. T.; Armitage, G.: A survey of techniques for internet traffic classification using machine learning. *Commun. Surveys Tuts.*, ročník 10, č. 4, Duben 2008: s. 56–76.

[79] Foremski, P.: On different ways to classify Internet traffic: a short review of selected publications. *Theoretical and Applied Informatics*, ročník 25, č. 2, 2013: s. 119–136.

[80] Bakhshi, T.; Ghita, B.: On Internet Traffic Classification. *J. Comput. Netw. Commun.*, ročník 2016, Červen 2016: s. 1–21.

[81] Kong, L.; Huang, G.; Wu, K.; aj.: Comparison of Internet Traffic Identification on Machine Learning Methods. In *2018 International Conference on Big Data and Artificial Intelligence (BDAI)*, June 2018, s. 38–41.

[82] Bay, S. D.; Kibler, D.; Pazzani, M. J.; aj.: The UCI KDD Archive of Large Data Sets for Data Mining Research and Experimentation. In *SIGKDD Explorations*, 2000, s. 200–0.

[83] Bhuyan, M. H.; Bhattacharyya, D. K.; Kalita, J. K.: Network Anomaly Detection: Methods, Systems and Tools. *IEEE Communications Surveys Tutorials*, ročník 16, č. 1, January 2014: s. 303–336, ISSN 2373-745X.

[84] Iglesias, F.; Zseby, T.: Analysis of network traffic features for anomaly detection. *Machine Learning*, ročník 101, č. 1, Oct 2015: s. 59–84, ISSN 1573-0565.

[85] Haq, N.; Avishek, M.; Shah, F.; aj.: Application of Machine Learning Approaches in Intrusion Detection System: A Survey. *International Journal of Advanced Research in Artificial Intelligence*, ročník 4, 03 2015.

[86] Hawkins, D.: *Identification of Outliers*. Monographs on applied probability and statistics, Chapman and Hall, 1980, ISBN 9780412219009.

[87] 5 Ways To Handle Missing Values In Machine Learning Datasets. Únor 2018, [Online; vid. 28. Dubna 2019]. Dostupné z: `https://www.analyticsindiamag.com/5-ways-handle-missing-values-machine-learning-datasets`

[88] van Buuren, S.: *Flexible imputation of missing data*. Boca Raton, FL CRC Press, 2018. Dostupné z: `https://stefvanbuuren.name/fimd/`

[89] Normal Distribution in Statistics. Apr 2018, [Online; vid 15. Leden 2019]. Dostupné z: `https://statisticsbyjim.com/basics/normal-distribution`

[90] scipy.stats.weibull_min — SciPy v1.3.0 Reference Guide. May 2019, [Online; vid 6. Listopadu 2018]. Dostupné z: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.weibull_min.html`

[91] Jones, E.; Oliphant, T.; Peterson, P.; aj.: SciPy: Open source scientific tools for Python. 2001, [Online; vid. 12.1.2019]. Dostupné z: http://www.scipy.org/

[92] Acuña, E.; Rodríguez, C.: An empirical study of the effect of outliers on the misclassification error rate. *Transactions on Knowledge and Data Engineering*, 2005.

[93] Yeo, I.; Johnson, R. A.: A new family of power transformations to improve normality or symmetry. *Biometrika*, ročník 87, č. 4, 12 2000: s. 954–959.

[94] Box, G. E. P.; Cox, D. R.: An Analysis of Transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, ročník 26, č. 2, 1964: s. 211–243.

[95] Importance of Feature Scaling — scikit-learn 0.21.3 documentation. Aug 2019, [Online; vid 9. Únor 2019]. Dostupné z: https://scikit-learn.org/stable/auto_examples/preprocessing/plot_scaling_importance.html#sphx-glr-auto-examples-preprocessing-plot-scaling-importance-py

[96] Dash, M.; Liu, H.: Feature Selection for Classification. *Intell. Data Anal.*, ročník 1, č. 3, Květen 1997: s. 131–156, ISSN 1088-467X.

[97] Sun, M.; Chen, J.; Zhang, Y.; aj.: A New Method of Feature Selection for Flow Classification. *Physics Procedia*, ročník 24, 12 2012: s. 1729–1736.

[98] Shetty, B.: Curse of Dimensionality. 2019, [vid. 12 Července 2019]. Dostupné z: https://towardsdatascience.com/curse-of-dimensionality-2092410f3d27

[99] Tang, J.; Alelyani, S.; Liu, H.: *Feature selection for classification: A review*. CRC Press, 01 2014, ISBN 9781466586741, s. 37–64.

[100] Liu, H.; Motoda, H.: *Computational Methods of Feature Selection (Chapman & Hall/Crc Data Mining and Knowledge Discovery Series)*. Chapman & Hall/CRC, 2007, ISBN 1584888784.

[101] Vergara, J.; Estevez, P.: A Review of Feature Selection Methods Based on Mutual Information. *Neural Computing and Applications*, ročník 24, 01 2014.

[102] Guyon, I.; Elisseeff, A.: An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.*, ročník 3, Březen 2003: s. 1157–1182, ISSN 1532-4435.

[103] Bhattacharyya, S.: Ridge and Lasso Regression: A Complete Guide with Python Scikit-Learn. *Medium*, Feb 2019. Dostupné z: https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b

[104] Iglesias, F.; Zseby, T.: Analysis of network traffic features for anomaly detection. *Machine Learning*, ročník 101, č. 1, Oct 2015: s. 59–84, ISSN 1573-0565.

[105] Abadi, M.; Agarwal, A.; Barham, P.; aj.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015, software available from tensorflow.org. Dostupné z: https://www.tensorflow.org/

[106] Chollet, F.; aj.: Keras. https://keras.io, 2015.

[107] Zhao, Y.; Nasrullah, Z.; Li, Z.: PyOD: A Python Toolbox for Scalable Outlier Detection. *Journal of Machine Learning Research*, ročník 20, č. 96, 2019: s. 1–7. Dostupné z: http://jmlr.org/papers/v20/19-011.html

[108] Li, J.; Cheng, K.; Wang, S.; aj.: Feature Selection: A Data Perspective. *arXiv:1601.07996*, 2016.

[109] McKinney, W.: Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, editace S. van der Walt; J. Millman, 2010, s. 51 – 56.

[110] Oliphant, T. E.: *A guide to NumPy*, ročník 1. Trelgol Publishing USA, 2006.

[111] Waskom, M.; Botvinnik, O.; Hobson, P.; aj.: seaborn: v0.5.0 (November 2014). Listopad 2014, doi:10.5281/zenodo.12710. Dostupné z: https://doi.org/10.5281/zenodo.12710

[112] Hunter, J. D.: Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, ročník 9, č. 3, 2007: s. 90–95, doi:10.1109/MCSE.2007.55.

[113] Minarik, P.; Vykopal, J.; Krmicek, V.: Improving Host Profiling with Bidirectional Flows. In *2009 International Conference on Computational Science and Engineering*, ročník 3, Aug 2009, s. 231–237.

[114] Vlăduţu, A.; Comaneci, D.; Dobre, C.: Internet traffic classification based on flows' statistical properties with machine learning: INTERNET TRAFFIC CLASSIFICATION BASED ON FLOWS' STATISTICAL PROPERTIES. *International Journal of Network Management*, ročník 27, 04 2016.

[115] Vargas Muñoz, M.; Martínez-Peláez, R.; Velarde Alvarado, P.; aj.: Classification of network anomalies in flow level network traffic using Bayesian networks. 02 2018, s. 238–243.

[116] Li, W.; Abdin, K.; Dann, R.; aj.: Approaching Real-time Network Traffic Classification. 10 2006.

[117] Velan, P.; Medková, J.; Jirsík, T.; aj.: Network traffic characterisation using flow-based statistics. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, April 2016, ISSN 2374-9709, s. 907–912.

[118] Basher, N.; Mahanti, A.; Mahanti, A.; aj.: A Comparative Analysis of Web and Peer-to-peer Traffic. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, 2008, ISBN 978-1-60558-085-2, s. 287–296.

[119] Bernaille, L.; Teixeira, R.; Akodkenou, I.; aj.: Traffic Classification on the Fly. *SIGCOMM Comput. Commun. Rev.*, ročník 36, č. 2, Apr 2006: s. 23–26, ISSN 0146-4833.

[120] IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. [Online; vid. 22. ledna 2019]. Dostupné z: `https://www.unb.ca/cic/datasets/ids-2017.html`

[121] Sharafaldin, I.; Lashkari, A. H.; Ghorbani, A. A.: Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *4th International Conference on Information Systems Security and Privacy (ICISSP)*, ročník 4, 2018.

[122] Habibi Lashkari, A.; Draper Gil, G.; Mamun, M.; aj.: Characterization of Encrypted and VPN Traffic Using Time-Related Features. In *2nd International Conference on Information Systems Security and Privacy (ICISSP 2016)*, 02 2016, s. 407–414.

[123] Garcia, S.: Malware Capture Facility Project. Dostupné z: `https://mcfp.felk.cvut.cz/publicDatasets/`

[124] WIDE MAWI WorkingGroup. Jun 2015, [Online; vid 26. 5. 2019]. Dostupné z: `https://mawi.wide.ad.jp`

[125] R. Fontugne, P. A. K. F., P. Borgnat: MAWILab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *ACM CoNEXT 2010*, 12 2010.

[126] Velan, P.; Medková, J.; Jirsík, T.; aj.: Network traffic characterisation using flow-based statistics. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, April 2016, s. 907–912.

[127] Alcock, S.; Nelson, R.: Libprotoident: Traffic Classification Using Lightweight Packet Inspection. 01 2012. Dostupné z: `https://github.com/wanduow/libprotoident`

[128] Deri, L.; Martinelli, M.; Bujlow, T.; aj.: nDPI: Open-source high-speed deep packet inspection. In *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, Aug 2014, s. 617–622.

[129] QoSient, LLC: Argus: Auditing network activity. [online], 2000–2013. Dostupné z: `http://www.qosient.com/argus/`

[130] Bujlow, T.; Carela-Español, V.; Barlet-Ros, P.: *Extended Independent Comparison of Popular Deep Packet Inspection (DPI) Tools for Traffic Classification*. Universitat Politècnica de Catalunya, 1 2014, 1–440 s.

[131] Bujlow, T.; Carela-Español, V.; Barlet-Ros, P.: Independent Comparison of Popular DPI Tools for Traffic Classification. *Comput. Netw.*, ročník 76, č. C, jan 2015: s. 75–89, ISSN 1389-1286.

[132] Internet Protocol Version 4 (IPv4) Parameters - IP TOS Parameters. Květen 2018, [Online; vid. 14. Únor 2019]. Dostupné z: `https://www.iana.org/assignments/ip-parameters/ip-parameters.xhtml`

[133] Guster, D.; Robinson, D.; Safonov, P.: Packet Inter-Arrival Distribution in Computer Network Workloads. 2005, [Online; vid. 3. Březen 2019]. Dostupné z: `http://what-when-how.com/information-science-and-technology/packet-inter-arrival-distributions-in-computer-network-workloads/`

[134] Luo, S.; Marin, G.: Generating Realistic Network Traffic for Security Experiments. 04 2004, ISBN 0-7803-8368-0, s. 200 – 207.

[135] Garsva, E.; Paulauskas, N.; Grazulevicius, G.; aj.: Packet Inter-arrival Time Distribution in Academic Computer Network. *Electronics and Electrical Engineering*, ročník 20, 03 2014.

[136] Arfeen, M.; Pawlikowski, K.; Mcnickle, D.; aj.: The role of the Weibull distribution in Internet traffic modeling. In *Proceedings of the 2013 25th International Teletraffic Congress (ITC)*, 09 2013, s. 1–8.

[137] Arshadi, L.; Jahangir, A.: An empirical study on TCP flow interarrival time distribution for normal and anomalous traffic. *International Journal of Communication Systems*, ročník 30, č. 1, 10 2014: s. 1–18.

[138] Ostrowsky, L.; Fonseca, N.; Melo, C.: A traffic model for UDP flows. 06 2007, s. 217–222.

[139] Basher, N.; Mahanti, A.; Mahanti, A.; aj.: A comparative analysis of web and peer-to-peer traffic. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, 04 2008, ISBN 978-1-60558-085-2, s. 287–296.

[140] Muslhtaq, S. A.; Rizvi, A. A.: Statistical analysis and mathematical modeling of network (segment) traffic. In *Proceedings of the IEEE Symposium on Emerging Technologies, 2005.*, Sep. 2005, s. 246–251.

[141] *ra(1) Linux User's Manual*. Třetí vydání, November 2007. Dostupné z: `https://qosient.com/argus/man/man1/ra.1.pdf`

[142] Postel, J.: Internet Protocol. RFC 791, RFC Editor, September 1981. Dostupné z: `https://www.rfc-editor.org/info/rfc791`

[143] Li, J.; Cheng, K.; Wang, S.; aj.: Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, ročník 50, č. 6, 2018: str. 94. Dostupné z: `http://featureselection.asu.edu/`

[144] Hanchuan Peng; Fuhui Long; Ding, C.: Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 27, č. 8, Aug 2005: s. 1226–1238, ISSN 1939-3539. Dostupné z: `https://github.com/fbrundu/pymrmr`

[145] Raschka, S.: MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. *The Journal of Open Source Software*, ročník 3, č. 24, Duben 2018, doi:10.21105/joss.00638. Dostupné z: `http://joss.theoj.org/papers/10.21105/joss.00638`

[146] Koehrsen, W.: A Feature Selection Tool for Machine Learning in Python. Přebráno v Listopadu 2020. Dostupné z: `https://github.com/WillKoehrsen/feature-selector/blob/master/feature_selector/feature_selector.py`

[147] Ke, G.; Meng, Q.; Finley, T.; aj.: LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30*, editace I. Guyon; U. V. Luxburg; S. Bengio; H. Wallach; R. Fergus; S. Vishwanathan; R. Garnett, Curran Associates, Inc., 2017, s. 3146–3154. Dostupné z: `http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf`

[148] Witten, I. H.; Frank, E.; Hall, M. A.: *Data Mining: Practical Machine Learning Tools and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., třetí vydání, 2011, ISBN 0123748569.

[149] Chen, T.; Guestrin, C.: XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International onference on Knowledge Discovery and Data Mining*, KDD '16, New York, NY, USA: ACM, 2016, ISBN 978-1-4503-4232-2, s. 785–794, doi: 10.1145/2939672.2939785. Dostupné z: http://doi.acm.org/10.1145/2939672.2939785

Příloha **A**

Seznam použitých zkratek

ABOD Angle-base Outlier Detection

ANN Artificial Neural Network

CART Classification and Regression Tree

CTU Czech Technical University

CV Cross Validation

DPI Deep Packet Inspection

FA Factor Analysis

FTP File Transfer Protocol

HTTP HyperText Transfer Protocol

IANA Internet Assigned Numbers Authority

IP Internet Protocol

K-NN K-Nearest Neighbors

LASSO Least Absolute Selection and Shrinkage Operator

LAN Local Area Network

LOF Local Outlier Factor

NAPT Network Address Port Translation

NB Naive Bayes

NIDS Network Intrusion Detection System

P2P Peer to Peer

PCA Principal Component Analysis

PCAP Packet Capture

REPTree Reduced Error Pruning Tree

ROC Receiver operating characteristic Curve

SVM Support Vector Machine

TCP Transmission Control Protocol

TOS Type of Service

UDP User Datagram Protocol

WIDE Widely Integrated Distributed Environment

QOS Quality of Service

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF