VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

DETEKCE ŠKODLIVÝCH DOMÉNOVÝCH JMEN

DETECTION OF MALICIOUS DOMAIN NAMES

SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE **JIŘÍ SETINSKÝ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. PETER TISOVČÍK**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Setinský Jiří**
Program: Informační technologie
Název: **Detekce škodlivých doménových jmen**
Detection of Malicious Domain Names
Kategorie: Počítačové sítě

Zadání:

1. Nastudujte dostupnou literaturu o použití strojového učení v oblasti analýzy síťových dat.
2. Analyzujte dostupná data o síťových tocích, o bezpečnostních událostech a data z externích zdrojů se zaměřením na doménová jména.
3. Navrhněte vhodný způsob detekce škodlivých doménových jmen.
4. Navržený způsob detekce škodlivých doménových jmen implementujte.
5. Implementaci ověřte nad reálnými daty.
6. Zhodnoťte dosažené výsledky a diskutujte možnosti dalšího rozšíření.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a částečně bod 3.

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

Vedoucí práce: **Tisovčík Peter, Ing.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Bakalářská práce pojednává o detekování uměle vygenerovaných doménových jmen (DGA). Vygenerované adresy slouží jako komunikační prostředek mezi útočníkem a nakaženým počítačem. Detekcí můžeme odhalit a vystopovat nakažené počítače v síti. Samotné detekci předchází prostudování technik strojového učení, které budou následně aplikovány při tvorbě detektoru. Pro vytvoření výsledného klasifikátoru v podobě rozhodovacího stromu bylo potřeba analyzovat podobu DGA adres. Na základě jejich charakteristiky se extrahovaly atributy, podle kterých se bude výsledný klasifikátor rozhodovat. Po natrénování klasifikačního modelu na trénovací sadě byl klasifikátor implementován v cílové platformě NEMEA jako detekční modul. Po finálních optimalizacích a testování jsme dosáhli úspěšnosti klasifikátoru 99 %, což je velmi pozitivní výsledek. NEMEA modul je připraven pro nasazení do reálného provozu, aby mohl detekovat bezpečnostní incidenty. Kromě NEMEA modulu byl dodatečně vytvořen model na predikování úspěšnosti datových sad s doménovými jmény. Model je natrénován na základě charakteristiky datové sady a úspěšnosti DGA detektoru, jehož chování chceme predikovat.

Abstract

The bachelor thesis deals with the detection of artificially generated domain names (DGA). The generated addresses serve as a means of communication between the attacker and the infected computer. By detection, we can detect and track infected computers on the network. The detection itself is preceded by the study of machine learning techniques, which will then be applied in the creation of the detector. To create the final classifier in the form of a decision tree, it was necessary to analyze the principle of DGA addresses. Based on their characteristics, the attributes were extracted, according to which the final classifier will be decided. After learning the classification model on the training set, the classifier was implemented in the target platform NEMEA as a detection module. After final optimizations and testing, we achieved a accuracy of the classifier of 99 %, which is a very positive result. The NEMEA module is ready for real-world deployment to detect security incidents. In addition to the NEMEA module, another model was created to predict the accuracy of datasets with domain names. The model is trained based on the characteristics of the dataset and the accuracy of the DGA detector, whose behavior we want to predict.

Klíčová slova

strojové učení, doménová jména, rozhodovací strom, botnet, detekce umělých domén, binární klasifikace, síťová bezpečnost, NEMEA, DGA

Keywords

machine learning, domain names, decision tree, botnet, detection of generated domains, binary classification, network security, NEMEA, DGA

Citace

SETINSKÝ, Jiří. *Detekce škodlivých doménových jmen.* Brno, 2021. Semestrální projekt. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Peter Tisovčík

Detekce škodlivých doménových jmen

Prohlášení

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

.....
Jiří Setinský
10. května 2021

Poděkování

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

Obsah

Kapitola 1

Úvod

V současné době jsou naše životy doprovázeny výpočetními technologiemi na každém kroku. Bez nich bychom si asi dnešní svět nedovedli představit. Díky novým technologiím si můžeme usnadnit každodenní úkoly. Informační technologie jsou dnes velice rozsáhlé obsahující mnoho podoborů. Jednou z větví informatiky je právě strojové učení, které bude předmětem této práce.

Strojové učení neboli machine learning se zařazuje do podoblasti umělé inteligence. Když se snažíme vyřešit problém na počítači, tak vytvoříme algoritmus, pomocí kterého počítač převede daný vstup na požadovaný výstup. Nastávají situace, kdy pro danou úlohu neznáme vhodný algoritmus. V tomto případě přichází na řadu strojové učení. Abychom situaci ilustrovali na příkladě, tak si představme, že se snažíme filtrovat příchozí emaily, aby neobsahovaly spam. Jako lidé dopředu známe, jaký bude vstup (prostá emailová zpráva) a jaký by měl být výstup (je/není to spam). Na základě naší paměti a předchozích zkušeností jsme schopni rozhodnout, jaký bude výstup bez jednoznačného algoritmu. Přesně tuto myšlenku využívá strojové učení [1]. Dnešní technologie nám umožňují ukládat obrovské množství dat, které využijeme při generování předpokládaného výstupu programu. V příkladu s detekováním spamu předložíme počítači data, která obsahují spamy a on se "naučí" jak vypadají a jaké mají společné prvky. Tímto umožníme, aby program byl schopen reagovat na měnící se vstupní podmínky.

Obecná definice strojového učení zní: zkoumá algoritmy a statistické modely, které využívá počítačový systém k vykonání úlohy, aniž by použil specifikované instrukce a místo toho pracuje s opakovanými vzory chování a dedukcí. K uskutečnění predikce výpočtu využívá datové sady, na kterých se trénuje výsledný model. Model může mít funkci prediktivní (předpovídat budoucí chování) nebo klasifikační (získávat informace z dat). Aspekty daného modelu jsou přesnost, paměťová a časová náročnost. Strojového učení se uplatňuje například ve zdravotnictví, virtuální realitě, sociálních sítí, obchodování na burze, rozpoznávání obrazu nebo při detekci bezpečnostních hrozeb [1].

Právě bezpečnostní hrozby jsou hlavní motivací k vytvoření této práce. Budeme chtít detekovat DGA adresy. DGA značí domain generation algorithm. Tento algoritmus využívá malware ke komunikaci mezi botnetem a řídicím serverem, kdy botnet generuje velký objem generovaných doménových jmen za účelem těžko detekovatelného spojení s řídicím serverem [31]. Zachycením DGA adresy jsme schopni vystopovat nakažený počítač, který právě komunikuje s útočníkem pomocí DGA adres. Existují různé DGA detektory, ale přesto chceme vytvořit nový. Implementovaný detektor bude nasazen na síti CESNETu v rámci detekčního systému NEMEA. NEMEA zatím žádným detektorem DGA adres nedisponuje, a proto je našim cílem detektor implementovat.

Výsledkem práce bude právě NEMEA modul, který bude detekovat DGA. Na tuto problematiku se perfektně hodí strojové učení. Po nastudování principů strojového učení bude snaha o vyextrahování společných znaků charakterizující DGA adresy a na jejich základě natrénovat prediktivní model. Vytvořený model budeme chtít integrovat v rámci NEMEA systému. Jako finální klasifikační model byl zvolen rozhodovací strom. Po finálních implementacích a optimalizacích dosahuje model úspěšnosti 99 % s rychlostí klasifikace 30 000 adres za sekundu, což je pozitivní výsledek.

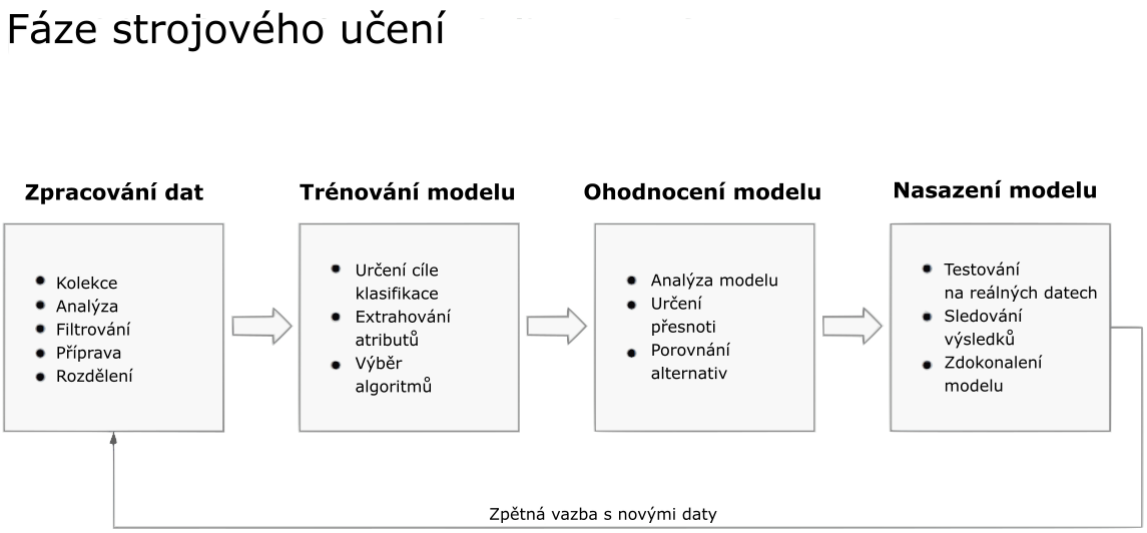
Po dokončení implementace NEMEA modulu, bude rozebráno ohodnocování datových sad obsahující uměle vytvořená doménová jména pomocí genetických algoritmů. Jelikož se přesnost modelu odvíjí od kvality datové sady, na které byl klasifikátor natrénován, tak je důležité vytvořit dobrou datovou sadu. K vytváření takových sad můžeme využít například genetické algoritmy. Uměle vytvořené datové sady budeme chtít charakterizovat, abychom na základě vytvořené charakteristiky mohli predikovat úspěšnost zkoumaného klasifikátoru.

V kapitole 2 je popsána teorie strojového učení, jeho fáze a techniky využívající se v této oblasti. Dále jsou popsány algoritmy, na kterých je založeno strojové učení. V kapitole 3 budou rozebrány doménová jména. Vysvětlíme k čemu slouží a jak jsou zneužívány. Následně bude popsáno, na jaké bázi pracuje DGA. Stručně budou uvedeny jednotlivé kategorie DGA. Následně bude představen nástroj NEMEA, který zachytává síťový provoz a detekuje síťové hrozby. Poté si ještě představíme software Rapidminer a Weka, které nám budou sloužit k analyzování datových sad s doménovými jmény. V kapitole 4 bude představen návrh klasifikátoru, který bude sloužit k detekování DGA adres v síťové komunikaci. V rámci této kapitoly budou uvedeny atributy jaké se budou u doménových jmen získávat a jaké kroky bude potřeba udělat, aby vznikl výsledný klasifikátor, který můžeme aplikovat v praxi. Kapitola 5 bude pojednávat o samotné implementaci modulu. Bude nastíněna architektura modulu, inicializace datových struktur, detailnější popis implementace některých atributů a následné integrování natrénovaného klasifikátoru do cílové platformy. Kapitola 6 bude věnována testování a vyhodnocení výsledků klasifikátoru. Klasifikátor bude otestován na množině testovacích a reálných sad. Dále bude provedeno srovnání klasifikátoru s konkurenčními DGA detektory. Na závěr bude provedena analýza využití zdrojů a časové náročnosti modelu. Poslední kapitola 7 je věnována predikování úspěšnosti datových sad. Bude vytvořen prediktivní model, který na základě charakteristiky datové sady dokáže odhadnout výslednou úspěšnost referenčního klasifikátoru.

Kapitola 2

Strojové učení

Strojové učení (Machine learning) nabývá na popularitě a velké množství firem ho začíná využívat. Proto by bylo dobré shrnout základní fáze, jak dojít k výslednému modelu. V první řadě se musí dobře specifikovat problém a zvolit vhodnou strategii. Na obrázku 2.1 jsou fáze strojového učení graficky znázorněny. Následující text bude jednotlivé bloky z obrázku popisovat. Po popisu jednotlivých fází přejdeme na rozebrání algoritmů a technik, které se při aplikování strojového učení využívají.



Obrázek 2.1: Popis a pořadí jednotlivých fází strojového učení. Převzato z [5].

2.1 Tvorba datových sad

Jednou z nepostradatelných součástí strojového učení je tvorba datových sad. Tento proces můžeme rozdělit do několika kroků [17]. Zde bude popsán běžný způsob tvorby datových sad, kdy se datová sada vytváří manuálním způsobem nebo se převezme už existující datová sada. V dnešní době v podstatě veškeré užívání moderních technologiích souvisí se sbíráním informací. Shromažďuje se velké množství dat nejen o uživatelích, ale i o procesech zajišťující

chod technologií například monitorování sítě. Tyto informace se poté stávají zdrojem pro tvorbu datových sad, jenž často slouží pro zlepšení kvality služeb.

Kolekce dat

Typ sbíraných dat je určen na základě toho, co chceme predikovat. Množství dat je relativní, protože každý problém řešený strojovým učením je unikátní. Dopředu se nedá odhadnout jaká data budou pro náš model při trénování nejefektivnější, a proto se doporučuje uchovávat veškerá nasbíraná data. Až na základě trénování modelu můžeme určit, která data jsou vhodná a vykazují nejvyšší přesnost modelu. Datové sady se dají vytvářet pomocí vlastních zdrojů nebo se může využít volně dostupných datových sad.

Selekce dat

Data se profiltrují a odstraní se prvky s nerelevantními vlastnostmi vůči výslednému modelu. Pokud budeme zachytávat data ze sítě, tak v kontextu řešeného problému nás budou pouze zajímat dotazovaná doménová jména.

Pro lepší reprezentaci velkých datových sad, které jsou často nepřehledné, se data zobrazují v čitelnější podobě formou grafů a diagramů. Vizualizaci nám umožňují data mining nástroje jako je Weka nebo Rapidminer, které nám později budou sloužit k extrahování společných rysů datové sady. Při analýze datových sad budou nepostradatelnou součástí histogramy, které nám budou sloužit k vyjádření četnosti zastoupení různých elementů zkoumaných dat. Dalším grafem, který přehledně vizualizuje data je boxplot. Tento graf vyjadřuje rozsahové vlastnosti dat. Z grafu můžeme vyčíst střední hodnotu, rozptyl nebo extrémy. Ještě se využívá bodový graf, kde je dobře vidět jestli data spadající do stejné třídy vytvářejí shluky na základě zkoumaného atributu.

Anotace dat

Tento krok se provádí jen v situaci, kdy chceme aplikovat tzv. algoritmus řízeného strojového učení (supervised machine learning). Řízené metody, které budou zmíněny později, vyžadují datovou sadu s předem známými cílovými informacemi. V našem případě půjde o získání anotované datové sady obsahující doménová jména se štítky, jestli se jedná o DGA, nebo ne. Manuální datová anotace bývá často časově náročná a vyžaduje značné úsilí.

Předzpracování dat

Datová sada musí být precizně vytvořena, od toho se odvíjí kvalita výsledného modelu. Data musí mít konzistentní formát. Právě když získáváme data z různých zdrojů, tak musíme dbát na jejich soudržnost. Poté, co máme data ve stejném formátu, tak musíme vybrat podmnožinu dat natolik obsáhlou, aby byl model přesný a zároveň časově nenáročný. Často se také slučují atributy dat pomocí agregace nebo se vytvářejí nové atributy pomocí dekompozice.

Jakmile máme úspěšně vytvořenou datovou sadu, tak dochází na řadu její rozdělení. Doporučené rozdělení je na tři části [17]. Trénovací sada slouží k natrénování modelu a extrahování rysů dat, podle kterých se model rozhoduje. Testovací sada se používá k ohodnocení natrénovaného modelu a zda je schopen reagovat na neznámá data. Validační sada má spíše optimalizační funkci, kdy se daný model ladí a testuje se jeho efektivita.

2.2 Trénování modelu

Další fází je samotné generování výsledného modelu, kde se rozhoduje, který algoritmus nejlépe dokáže zpracovat trénovací data s největší přesností. Touto problematikou se ale budeme zabývat až v kapitole 4 s návrhem klasifikačního modelu.

2.3 Implementace modelu

V momentě, kdy získáme spolehlivý výsledný model, tak přichází na řadu implementace modelu. Nejvhodnější implementační jazyk je vybírán na základě kompatibility s cílovou platformou a požadavků na rychlost. V prvních fázích implementace se často využívá vysokoúrovňových jazyků jako je Python nebo R. Usnadňuje to práci vývojářům díky vyšší abstrakci. Dále je potřeba stanovit, jak často chceme model využívat. Záleží na tom, co chceme predikovat. V oblastech, kde se data dynamicky nemění, stačí dělat predikci ve stanovených intervalech. Po uvedení modelu do provozu je třeba otestovat, jestli opravdu splňuje jeho prvotní záměr a poskytuje požadované výsledky. Po úspěšném dokončení projektu samozřejmě následuje průběžné udržování a aktualizování modelu. Neustále se musí sledovat přesnost výsledného modelu, zda je schopen se adaptovat na stále nová data [17].

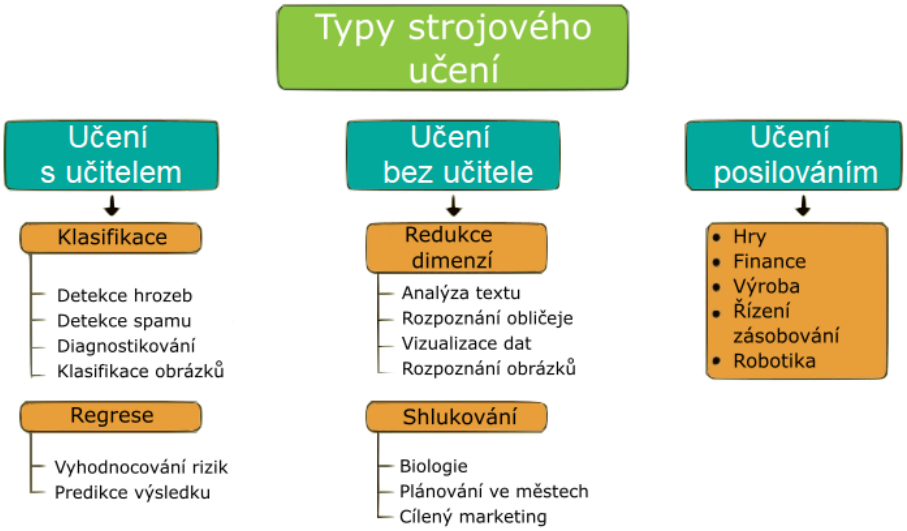
2.4 Metody klasifikace

Tato část bude pojednávat o jednotlivých modelech strojového učení a jejich způsobu užití v praxi. Modely nám slouží k predikování nebo klasifikování. Existuje velké množství přístupů a technik učení. Každá metoda má rozdílné použití. Podle použití můžeme tyto metody rozdělit do tří kategorií. První kategorie je učení s učitelem (supervised learning), dále následuje učení bez učitele (unsupervised learning) a učení posilováním (reinforcement learning).

Rozdělení do kategorií je ilustrováno na obrázku 2.2. Existuje velký počet možností, jakým způsobem můžeme rozdělit jednotlivé metody strojového učení. Jeden ze způsobů je právě zmiňované rozdělení podle stylu učení. Následující podkapitoly vysvětlí jednotlivé kategorie a ke každé z nich budou uvedeny konkrétní algoritmy. Zaměření bude kladeno hlavně na algoritmy nejčastěji používané při učení s učitelem a při učení bez učitele.

Nyní se stručně podíváme na učení posilováním. Využívá se ve více oblastech umělé inteligence. Umožňuje strojům reagovat na dynamicky se měnící okolní prostředí. V dané situaci stroj musí vyhodnotit ideální vzor chování. Stroj se rozhoduje na základě zpětné vazby, kterou získal v minulosti. A proto v delším časovém období dokáže lépe reagovat díky předešlé zpětné vazbě [29].

Na každý typ učení je vhodná určitá skupina algoritmů. To ale neznamená, že jednotlivé algoritmy se nedají použit při jiném stylu učení. Každý algoritmus má hodně modifikací a druhů. Nyní budou představeny algoritmy strojového učení, podle popularity u jednotlivých stylů učení.



Obrázek 2.2: Typy strojového učení a jejich aplikace v praxi. Převzato z [29].

2.4.1 Učení s učitelem

Tzv. učení s učitelem (supervised learning) patří mezi nejpopulárnější přístupy při aplikování strojového učení. Tento přístup se používá v případě, že máme k dispozici anotovanou datovou sadu. Na základě předem známých atributů datové sady můžeme snadno určit souvislosti mezi daty a získané informace využít k pozdějšímu predikování nebo klasifikování (využití např. při rozpoznávání obličejů). Podmínkou učení s učitelem je předem vytvořená datová sada obsahující atributy jednotlivých dat. Čím kvalitnější datovou sadu máme, tím přesněji můžeme natrénovat výsledný model [29]. Výše popsaný způsob bude využit i v této práci.

Lineární regrese

Lineární regrese vyjadřuje vztah mezi spojitými proměnnými. Regresní analýza nám umožňuje pomocí statistických metod odhadnout hodnotu náhodné veličiny. Základní princip využívá dvě proměnné, které se navzájem ovlivňují. Tento vztah jde vyjádřit pomocí lineární závislosti 2.1 zobrazené níže:

$$y = mx + c \tag{2.1}$$

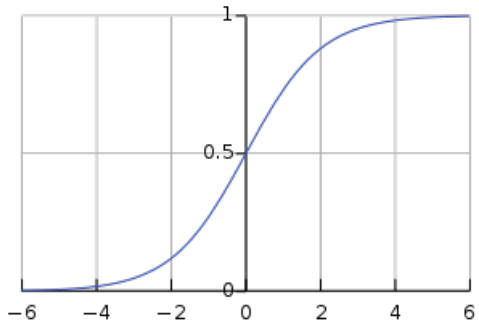
kde **y** je závislá proměnná a **x** nezávislá proměnná [30].

Logistická regrese

Logistická regrese spadá pod nelineární regresi a patří mezi nejpopulárnější algoritmy pro binární klasifikování. Výsledkem algoritmu je určení jedné ze dvou kategorií, do které spadá zkoumaná veličina. Jádrem metody je logistická funkce 2.2, která nabývá tvaru:

$$1/(1 + e^{-x}) \tag{2.2}$$

kde **e** je základ přirozeného algoritmu a **x** je číselná hodnota, kterou chceme transformovat do intervalu 0 až 1. Graf logistické funkce je zobrazen na obrázku 2.3.



Obrázek 2.3: Logistická funkce pro binární klasifikaci. Převzato z [30].

Logistickou regresi můžeme opět reprezentovat pomocí rovnice 2.3, s tím rozdílem, že výsledek bude binární hodnota. Níže je zmiňovaná rovnice:

$$y = e^{b0+b1*x}/(1 + e^{b0+b1*x}) \tag{2.3}$$

kde **y** je binární výstup, **x** je vstupní hodnota a **b** jsou koeficienty získané z trénovací datové sady [3].

Rozhodovací stromy

Rozhodovací strom vytvoří model nejpřímějším způsobem, a to na základě rozhodování pomocí reálných atributů získaných z datové sady. Listem rozhodovacího stromu je výsledná predikce. Tvoření stromů má dvě fáze. Samotné tvoření stromu a následné optimalizovaní při zachování přesnosti. Rozhodovací stromy se využívají jak na predikování, tak i na klasifikování. Díky své rychlosti a přesnosti patří mezi oblíbené algoritmy [30].

Nyní se podrobněji podíváme, jak algoritmus rozhodovacího stromu funguje. Mezi nejznámější a nejlepší algoritmy patří ID3 (Iterative Dichotomiser 3). ID3 při tvoření rozhodovacího stromu využívá dvou veličin, na základě kterých se určí, podle jakého atributu se bude daný uzel rozhodovat. Tyto dvě veličiny jsou entropie a informační zisk. Entropie vykazuje míru neuspořádanosti nebo náhodnosti množiny dat. Jinými slovy nám entropie říká v rámci rozhodovacího stromu, jak moc daný atribut dokáže ovlivnit výslednou predikci. Pro ilustrování si představme, že máme minci s orlem a pannou. Pravděpodobnost, že při hodu mincí padne orel nebo panna je 0.5. V takovém případě nemůžeme určit, která strana mince padne, a proto entropie nabývá maximální hodnoty 1. V opačném případě, kdybychom měli minci jen s orlem, tak můžeme s jistotou říct, že vždy padne orel. Entropie (míra náhodnosti) potom nabývá nulové hodnoty. Vypočítání entropie množiny S je dáno vztahem 2.4, který můžeme vidět níže:

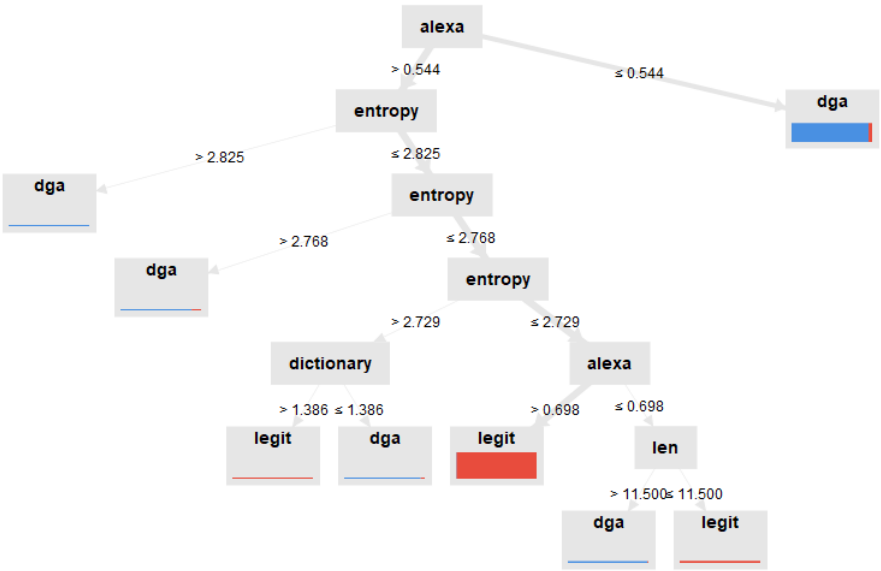
$$H(S) = \sum_{x \in X} p(x) log_2 \frac{1}{p(x)} \tag{2.4}$$

kde **X** je náhodná veličina reprezentující množinu S a $\mathbf{p(x)}$ je pravděpodobnost výskytu prvku náhodné veličiny v množině S.

Druhá veličina je informační zisk, jenž vyjadřuje změnu entropie v závislosti nad zvoleným atributem A. Vztah pro informační zisk 2.5 je vyjádřen rozdílem entropie celé množiny a entropie zvoleného atributu A. Cílem je zvolit atribut A tak, aby jeho entropie byla co nejmenší. Protože čím nižší entropie, tak daný atribut jednoznačněji dokáže predikovat výsledek. Už z názvu vyplývá, že čím vyšší informační zisk atributu, tak tím má vyšší váhu a na základě tohoto atributu se budeme rozhodovat hned v kořeni stromu [26].

$$IG(S, A) = H(S) - \sum_{i=0}^{n} P(x) * H(x) \tag{2.5}$$

Jakmile získáme atribut s největší váhou, tak se přesuneme na další podstromy kořenového uzlu a budeme řešit jednotlivé uzly rekurzivně. Tento rekurzivní proces děláme až do doby, kdy v dané větvi stromu už máme malé množství dat z trénovací sady a můžeme přejít k vytvoření koncových listů reprezentující výslednou predikci. Dalším způsobem, jak ukončit rekurzivní výpočty podstromů, je omezit algoritmus na určitou výšku stromu. Po dosažení stanovené výšky je algoritmus ukončen. Na obrázku 2.4 můžeme vidět podobu rozhodovacího stromu pro klasifikování doménových jmen. Můžeme vidět, že v diagramu se pracuje se spojitými hodnotami. Popisovaný algoritmus ID3 umí pracovat jen s diskrétními hodnotami, a proto se aplikuje algoritmus C4.5 [26], který rozšiřuje ID3 právě o práci se spojitými hodnotami, kdy se vytvoří práh rozdělující hodnoty na intervaly. Algoritmus následně pracuje s těmito vhodně rozdělenými diskrétními intervaly. Dalšími vylepšeními algoritmu C4.5 jsou například schopnost pracovat s chybějícími hodnotami nebo prořezávání stromu za účelem zrychlení klasifikace.



Obrázek 2.4: Ilustrační diagram rozhodovacího stromu.

Rozhodovací stromy se používají také v algoritmu zvaný Náhodný les (Random forest). Principem je vytvoření velkého množství navzájem nekorelujících rozhodovacích stromů. Každý strom vytvoří svůj vlastní nezávislý výsledek. Celkovým výsledkem algoritmu je pak nejčetnější individuální výsledek jednotlivých stromů. Důležité je, aby mezi stromy byla nízká korelace, kterou získáme buď pomocí náhodného výběru trénovací sady (každý strom

bude trénován na jiné podmnožině dat) nebo náhodnou selekcí atributů (výběr vhodného atributu pro uzel stromu je prováděn pouze na náhodně vybrané podmnožině). Těmito metodami docílíme rozdílnosti individuálních stromů. Algoritmus tak může předejít chybovosti jednotlivých stromů díky kolektivnímu vyhodnocování [32].

Bayesovské sítě

Bayesovské sítě jsou klasifikátory založené na podmíněné pravděpodobnosti. Využívají Bayesův vzorec 2.6:

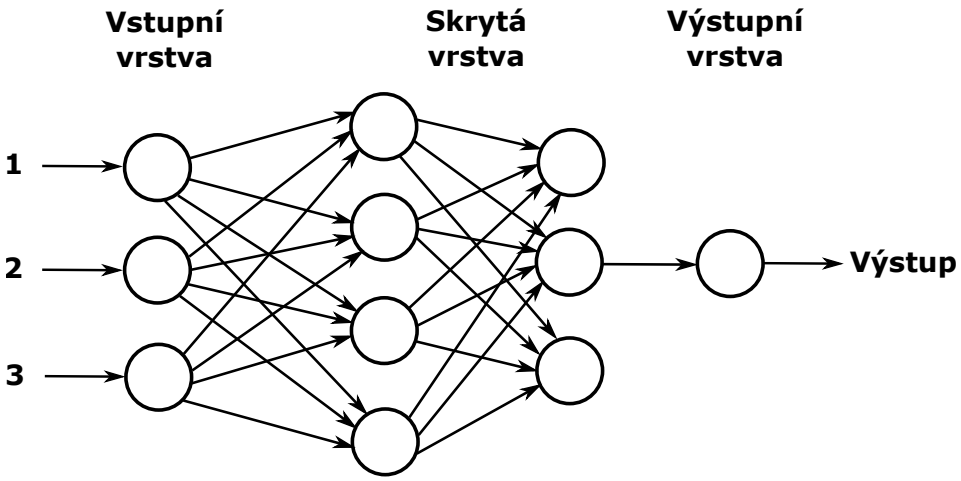
$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{2.6}$$

kde **P(c|x)** vyjadřuje podmíněnou pravděpodobnost jevu **c** za předpokladu, že nastal jev **x** a naopak **P(x|c)** je podmíněná pravděpodobnost jevu **x** za předpokladu, že nastal jev **c**.

Využívání tohoto vzorce přináší spoustu výhod. Jednoduchá implementace a algoritmus Naive Bayes nevyžaduje velkou trénovací datovou sadu, a i přesto jsou výsledky dostatečně přesné [30].

Neuronové sítě

Umělé neuronové sítě využívají stejného principu jako neurony v našem mozku. Struktura je složena z neuronů, které se chovají jako jednotlivé bloky rozložené do vrstev. Informace je propagována ze vstupní vrstvy směrem k výstupní vrstvě. Mezi vstupní a výstupní vrstvou jsou skryté vrstvy, které provádějí matematický výpočet, na základě požadovaného výstupu. Podle počtu vrstev se Neuronové sítě rozdělují na jednovrstvé zvané Perceptrony nebo vícevrstvé. Později se vytvořila nová podoblast zvaná hluboké učení. Hluboké učení se tvoří z komplexních systémů neuronových sítí. Umožňují nám zpracovat velké datové sady za poměrně krátkou výpočetní dobu. Na obrázku 2.5 můžeme vidět strukturu Neuronové sítě [30].



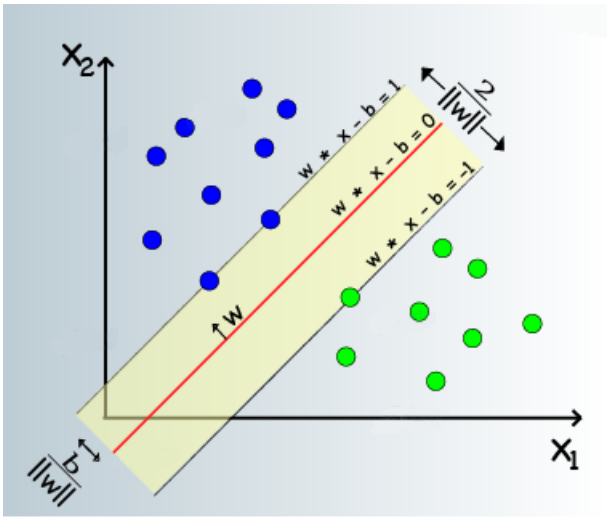
Obrázek 2.5: Struktura vícevrstvé neuronové sítě s topologií 3-4-3-1.

Algoritmus k-nejbližších sousedů

Algoritmus k-nejbližších sousedů využívá podobnosti dat. V datové sadě se snaží rozpoznat opakující se vzory a následně je rozdělit do kategorií s co největší podobností. Proces kategorizace je časově nejnáročnější fází algoritmu. Při vstupu nových dat, které chceme klasifikovat, musíme spočítat míru podobnosti s každou kategorií. Vstupní data jsou přiděleny do kategorie, která obsahuje největší míru dat s podobnými vlastnostmi [30].

Algoritmus podpůrných vektorů

Metoda podpůrných vektorů (support vector machine) je algoritmus opět založený na podobnosti dat. Trénovací data se musí kategorizovat. Proces kategorizace probíhá za pomocí zobrazení jednotlivých atributů dat do n-rozměrného prostoru. Dimenze prostoru záleží na počtu kategorií, které chceme klasifikovat. Jednotlivé atributy zobrazené v n-rozměrném prostoru chceme proložit nadrovinou, tak aby oblast kolem nadroviny měla co největší velikost a zároveň neobsahovala žádnou souřadnici atributu. Souřadnice atributů leží maximálně na hranici oblasti nadroviny. Tyto souřadnice se nazývají podpůrné vektory. Na obrázku 2.6 je graf zobrazující dvě kategorie a jejich rozdělení nadrovinou [30].



Obrázek 2.6: Graf podpůrných vektorů rozdělující prostor na dvě části. Převzato z [30].

2.4.2 Učení bez učitele

V případě učení bez učitele (unsupervised learning) už nemáme k dispozici předem známou datovou sadu. Model bez učitele je schopen se učit samostatně ze vstupních dat. V datech hledá předem neznámé souvislosti, opakující se vzory a hustoty pravděpodobnosti výskytu. Tento způsob učení se opírá o Hebbianovu teorii, která má kořeny v neurobiologii zkoumající kognitivní vlastnosti neuronů. Uplatnění tohoto způsobu učení se využívá například v oblasti marketingu při určení cílové skupiny lidí, u které je největší pravděpodobnost zájmu o nabízený produkt [29].

Shlukování

Shlukování (Clustering) je algoritmus, který vytváří množiny objektů tzv. shluky. Každá množina obsahuje objekty spolu navzájem související. Objekty s podobnými atributy budou náležet do stejné množiny. K vypočítání podobnosti se používají různé metriky. Často využívaný způsob výpočtu podobnosti je algoritmus K-means. K-means funguje na principu euklidovské vzdálenosti mezi objekty. Na začátku se určí počet shluků a těžiště každého shluku. Iterativním přístupem se snažíme zminimalizovat vzdálenost objektu shluku a jeho těžiště. Při každé iteraci algoritmu získáme nové těžiště pomocí průměru objektů shluku. Iterace probíhá, dokud se těžiště neustálí [30].

Algoritmus redukce dimenzí

Redukce dimenzí je jeden z důležitých principů strojového učení. Data můžou obsahovat velké množství atributů (dimenzí). Abychom mohli kvalitně natrénovat model, tak je zapotřebí zredukovat množství atributů bez ztráty přesnosti výsledného modelu. Zredukováním dosáhneme možnosti popsat datovou sadu s menším počtem informací. Zjednodušenou datovou sadu můžeme nadále využít při učení s učitelem. Nejpopulárnější algoritmus na redukování dimenzí je Analýza hlavních komponent (PCA) [30].

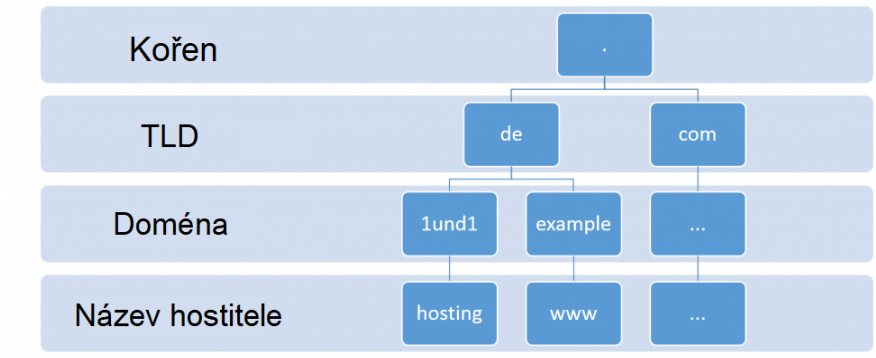
Detekce anomálií

Tato technika neuvádí konkrétní metodu, ale spíše poukazuje na způsob využití metod učení bez učitele v praxi. Detekce anomálií je velmi důležitá z hlediska bezpečnosti a stability. Úkolem je odhalit neobvyklá data, která se liší od většiny. Díky detekci této anomálie se nám podaří předejít bezpečnostním hrozbám [30].

Kapitola 3

Analýza doménových jmen v síti

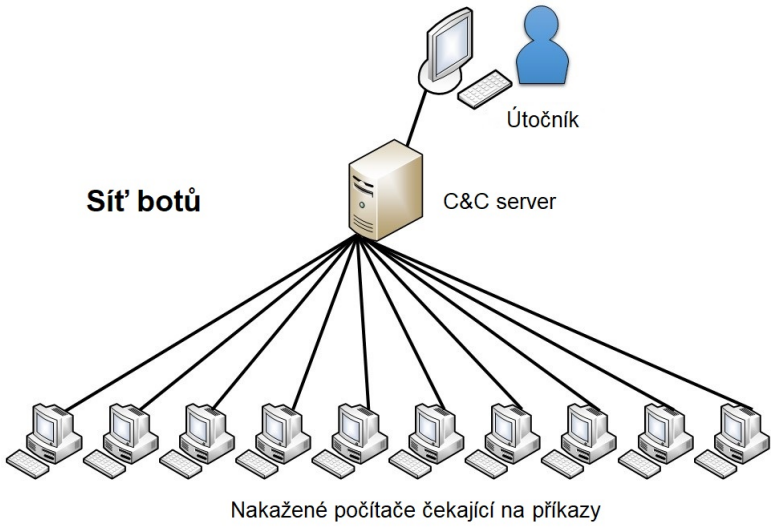
Doménové jméno je identifikační řetězec definující entitu v rámci internetu. Doménové jméno může reprezentovat síťovou doménu nebo jakoukoliv službu, jenž má svojí vlastní IP adresu, a je tak směrovatelná v rámci internetu. Příkladem je třeba osobní počítač, který je připojen do sítě, nebo server poskytující webové stránky. Hlavním účelem zavedení doménových jmen bylo zjednodušení směrování pro člověka v síti. Infrastruktura doménových jmen je řízena decentralizovaným systémem doménových jmen (DNS). Registrací libovolného řetězce do DNS se řetězec stává doménovým jménem. Díky DNS můžeme adresovat cílové zařízení pomocí doménového jména místo těžko zapamatovatelné IP adresy. Proces přeložení doménového jména na směrovatelnou IP adresu se nazývá DNS rezoluce. DNS rezolucí se postupně doptáváme doménových serverů, jestli uchovávají A (IPv4) nebo AAAA (IPv6) záznam právě překládaného doménového jména. Systém má hieratickou strukturu v podobě doménového stromu, který je zobrazený na obrázku 3.1. V kořeni adresáře je bezejmenná doména. Další úroveň zastupují tzv. TLD (top level domain) jména, mezi ně patří například *com, info, edu nebo org*. Další úrovně domén se postupně zanořují a umožňují vytvářet doménová jména, jak je známe. Od třetí úrovně doménového stromu se můžou domény kupovat u doménových registrátorů a adresovat tak lokální sítě, webové stránky nebo jakékoliv veřejné služby [19]. V připravovaném detektoru v rámci NEMEA modulu budeme chtít zachytávat právě DGA adresy. Předtím, než přejdeme k samotnému návrhu modulu, tak bude popsán princip generování umělých doménových jmen. Dále bude rozebrána struktura cílové platformy NEMEA. Na závěr této kapitoly budou uvedeny nástroje pro zpracování a analýzu dat, jako je například Weka nebo Rapidminer.



Obrázek 3.1: Hierarchie zanoření úrovní doménového jména. Převzato z [6].

3.1 Domain generation algorithm - DGA

Domain generation algorithm neboli algoritmus na generování doménových jmen je způsob, jak vytvářet škodlivá doménové jména. Tyto umělé adresy se využívají ve škodlivých softwarech (malware). Když na nakaženém počítači(bot) běží pravě takový software, většinou chce škodlivý program komunikovat s útočníkem (attacker), aby od něj mohl dostávat příkazy. Příkazy jsou následně vykonány za účelem napáchání škody nebo získání důvěrných informací. Nakažených počítačů může být obrovské množství, které dohromady na síti tvoří síť počítačů (botnet) vykonávající útočníkovy příkazy, které jsou posílány přes řídící server (command and control server). Tento způsob komunikace je anglicky nazýván *command and control communication*, jejíž koncept je vizualizován na obrázku 3.2. Mezi škodlivý software patří viry, červy, trojské koně nebo programy na odposlouchávání (získávání citlivých informací o uživateli). Ve většině času chtějí programy komunikovat s útočníkovým počítačem po síti. Před nástupem techniky využívající DGA, útočník předal nakaženému počítači pevně dané doménové jméno, pomocí kterého probíhala komunikace. Tento způsob komunikace je snadno identifikovatelný a stačí přidat doménové jméno na blacklist a zakázat komunikaci skrze toto doménové jméno.



Obrázek 3.2: Infrastruktura C&C (Command and control) komunikace. Převzato z [21].

S nástupem DGA se blokování komunikace poměrně ztěžuje. Díky velkému množství dynamicky se měnících adres je téměř nemožné blokovat všechny adresy [31]. Škodlivý program obsahuje libovolný generátor doménových jmen a na začátku komunikace obdrží od útočníka počáteční seed, na základě kterého se budou generovat doménová jména. Díky počátečnímu seedu se budou generovat stejné množiny jmen. Nakažený počítač tak může generovat obrovské množství adres. Z tohoto množství si vybere pouze podmnožinu k pokusu o navázání komunikace s útočníkem. Útočníkův počítač zná seed, takže ví na jakou množinu se infikovaný počítač bude dotazovat. Proto útočník vybere pouze pár jmen z této množiny a tyto jména následně zaregistruje a využije je ke komunikaci. Z pohledu nakaženého počítače je nevýhoda toho, že bude trvat nějakou dobu, než vybere takovou podmnožinu zahrnující registrované doménové jméno. Touto metodou, ale útočník sníží pravděpodobnost

zablokování komunikace. Metoda blokování doménových jmen a jejich přidání do blacklistu je nevhodná při generování takového množství adres pomocí DGA. S nízkou pravděpodobností se dokonce může stát, že vygenerovaná adresa existuje a platné adresy nechceme blokovat. Ideálním řešením této problematiky je použít strojové učení na detekování DGA adres a odhalit tak nakažený počítač generující tyto adresy.

Existuje desítky algoritmů, které se dají použít na vytváření podvržených doménových jmen. Analyzováním těchto algoritmů nám umožní kvalitněji určit atributy, které pro nás budou relevantnější při odhalování DGA. Nejčastěji se používají algoritmy, využívající náhodné generování. Existují také algoritmy, které tvoří domény na základě slovníků, ale tento způsob není tak populární, protože to zahrnuje obsáhlejší kód a tím se usnadní jeho odhalení. Následující text bude pojednávat o některých typech náhodných DGA algoritmů [23].

Bamital

Doména nejvyšší úrovně: [co.cc, cz.cc, info, org]

Doména druhé úrovně: md5 hashování

Příklady:
cd8f66549913a78c5a8004c82bcf6b01.info
aa24603b0defd57ebfef34befde16370.cz.cc

Cryptolocker

Doména nejvyšší úrovně: [com, net, biz, ru, org, co.uk, info]

Doména druhé úrovně: délka 12-15, znaky a-y

Příklady:
nvjwoofansjbh.ru
qgrkvevybtvckik.org

Conficker

Doména nejvyšší úrovně: [cc, cn, ws, com, net, org, info, biz]

Doména druhé úrovně: délka 5-11, znaky a-z

Příklady:
ibymtpyd.info
glrmwqh.net

Banjori

Doména nejvyšší úrovně: stejná jako původní

Doména druhé úrovně: změna prvních 4 znaků původní domény

Příklady:
earnestnessbiophysicalohax.com
kwtoestnessbiophysicalohax.com

Ccleaner

Doména nejvyšší úrovně: [com]

Doména druhé úrovně: délka 5-11, znaky a-f a 0-9

Příklady:
ab1145b758c30.com
ab890e964c34.com

Zdroj dat

Aby bylo možné detekovat škodlivé doménová jména, je potřeba shromažďovat datové toky obsahující doménová jména. Zdrojem budou data z IPFIX (protokol pro sběr síťových dat) exportéru, který právě zachytává datové toky, které tečou přes síť CESNETu. Tyto reálná data využijeme, ale až při výsledném testování NEMEA modulu. Pro trénování klasifikátoru využijeme předem označenou datovou sadu. Jedním ze zdrojů validních doménových jmen může být Alexa od Amazonu nebo Cisco umbrella, jenž agregují světově nejnavštěvovanější stránky. Níže je uveden seznam odkazů na některé sady s validními doménovými jmény [31]. Validita uvedených zdrojů ale není samozřejmě garantovaná. Uvedená doménové jména mohou zaniknout a název domény převezme nový vlastník a začne šířit malware. Dalším důvodem je fakt, že server šířící malware může mít vysokou návštěvnost, a proto je zařazen na seznam považovaný za validní.

Použité seznamy validních doménových jmen:

- Alexa top 1 milion[1]
- Quantcast[2] (vyžaduje přihlášení)
- Cisco Umbrella Top 1 million[3]
- DomCop Top 1M[4]

Poté, co získáme validní doménová jména, tak musíme získat doménová jména, jenž byly vygenerovány uměle pomocí DGA algoritmů. Obdobně jak u validních jmen existují repozitáře s DGA jmény. Takový známý zdroj DGA adres je například Netlab 360 feeds nebo Bambenek feeds. Níže jsou uvedeny úložiště, jenž tyto adresy shromažďují a dále jsou uvedeny příklady algoritmů pomocí kterých tyto adresy můžeme tvořit [31].

[1] http://s3.amazonaws.com/alexa-static/top-1m.csv.zip
[2] https://ak.quantcast.com/quantcast-top-sites.zip
[3] http://s3-us-west-1.amazonaws.com/umbrella-static/top-1m.csv.zip
[4] https://www.domcop.com/files/top/top10milliondomains.csv.zip

DGA jména:

- DGA archív[5] (omezený přístup)
- Bambenek Feeds[6] (sekce DGA domain feed)
- Netlab 360 DGA Feeds[7]

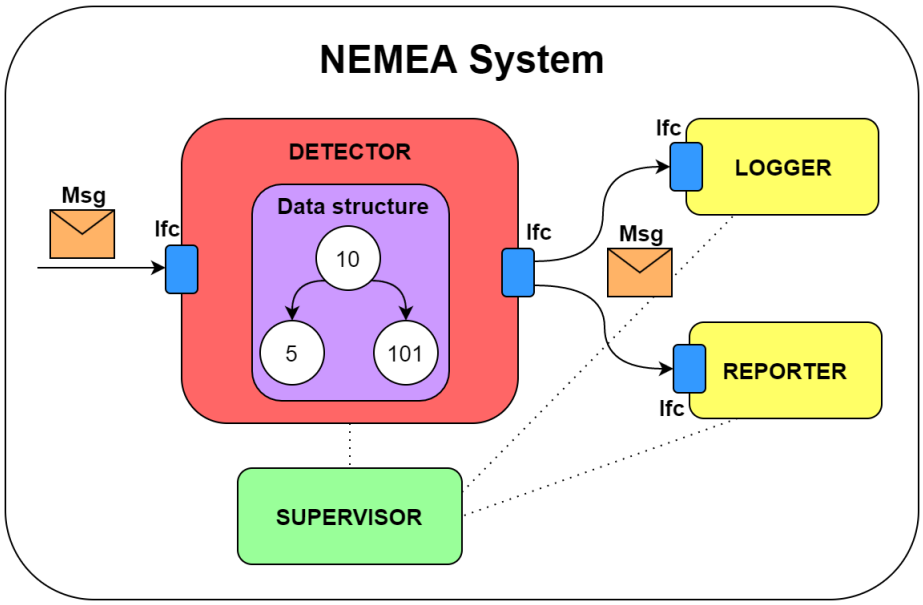
DGA algoritmy:

- baderj/domain_generation_algorithms[8] (rodina algoritmů v jazyce python)
- andrewaeva/DGA[9]
- pchaigno/dga-collection[10]

[5] https://dgarchive.caad.fkie.fraunhofer.de/
[6] https://osint.bambenekconsulting.com/feeds/
[7] https://data.netlab.360.com/dga/
[8] https://github.com/baderj/domain_generation_algorithms
[9] https://github.com/andrewaeva/DGA
[10] https://github.com/pchaigno/dga-collection

3.2 NEMEA

Klasifikátor pro detekci generovaných doménových jmen bude implementován jako detektor v rámci systému NEMEA. Následující sekce bude věnována právě zmíněné cílové platformě. NEMEA (Network Measurements Analysis) je detekční systém, který monitoruje provoz na síti. Skládá se z nezávislých modulů, které mají specifické zaměření, struktura modulů je zobrazena na obrázku 3.3. Komunikace mezi jednotlivými moduly probíhá pomocí zpráv nesoucí data jako jsou datové toky, upozornění, statistické údaje nebo výstupní data předchozího modulu [33]. Bude následovat stručné popsání jednotlivých komponent zobrazených na obrázku 3.3. Na závěr této části bude nastíněna integrace, jakým způsobem by byl modulu zachytávající DGA adresy integrován v rámci propojení NEMEA bloků.



Obrázek 3.3: Struktura NEMEA systému a komunikace jednotlivých modulů. Převzato z [33].

Komponenty NEMEA

1. Moduly - tvoří základní kostru NEMEA, zpracovávají vstupní data. Rozdělují se na dvě části [33].

- **Detektory** (červená) - jejich úkolem je zachytit podezřelý síťový provoz
- **Moduly** (žlutá) - plní exportovací funkci, formátují a filtrují nasbíraná data

2. NEMEA Framework - knihovny implementující základní funkce pro všechny moduly [33].

- **TRAP** (Traffic Analysis Platform) (modrá) - vytváří komunikační rozhraní a umožňuje zasílání zpráv mezi moduly
- **UniRec** (oranžová) - datová struktura uchovávající přeposílaná data v konzistentním formátu

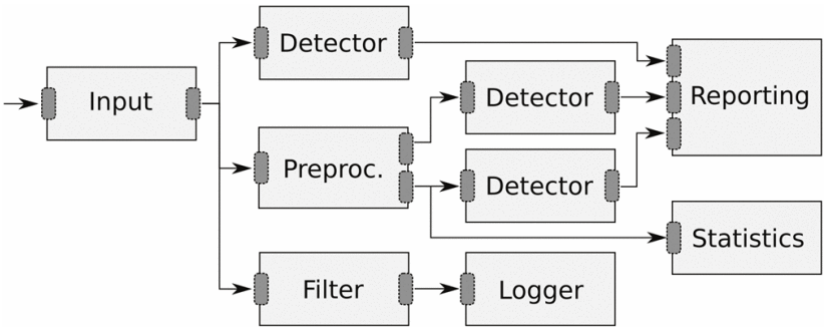
- **Common** (fialová) - implementace algoritmů a datových struktur používaných v modulech

3. Supervisor (zelený) - zajišťuje chod celého systému, dohlíží na správnou konfiguraci modulů [33].

Propojení NEMEA

Na obrázku 3.4 je znázorněno ilustrační propojení NEMEA modulů. Každý modul reprezentuje program, který má za úkol vykonávat konkrétní úlohu danou účelem modulu (zpracování dat, filtrování, detekce anomálií, logování nebo reportování událostí). Propojení jednotlivých bloků je značně variabilní a umožňuje velkou škálu konfigurací, jak propojit moduly a jaká data si budou posílat. Vstupním bodem systému je `input` modul z obrázku 3.4. Zdrojem dat pro tento vstupní uzel může být několik variant. Jednou z možností je offline způsob, kdy jsou do modulu načítány data ze souborů. NEMEA na vstupu podporuje typy souborů jako je `CSV, PCAP, nfdump`. Druhou variantou, jak nahrát data do NEMEA, je pomocí zachytávaní dat z IPFIX kolektoru. Datové toky jsou zachytávány pomocí monitorovací sondy na síti a následně jsou posílány do zmiňovaného IPFIX kolektoru. Data v kolektoru se poté pomocí pluginu přetransformují na formát, který podporuje NEMEA. Tento formát je výše uvedená UniRec struktura. Data v UniRec formátu se následně přepošlou ostatním běžícím modulům, jenž zpracují data podle své definované funkcionality [4].

Modul zachytávající DGA adresy, který bude v rámci této práce implementován, bude integrován na pozici `Detector` bloku z obrázku 3.4. Zachycené nevalidní adresy budou poslány reportujícímu modulu, který vytvoří záznam informující o zachycené události a může na událost patřičně reagovat. Tento reportující modul bude také implementován v rámci této práce.



Obrázek 3.4: Propojení NEMEA bloků a směr toku dat [4].

3.3 Nástroje na analýzu dat

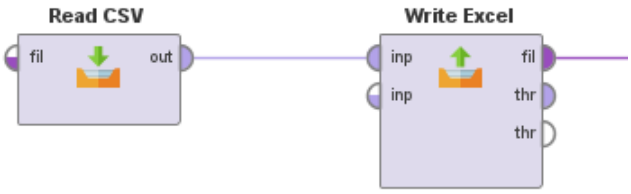
Data, která získáme ve formátu doménových jmen bude potřeba analyzovat. Manuální procházení dat by bylo velice neefektivní a nepřineslo by to žádné užitečné výsledky. Zpracovávaná datová sada bude obsahovat zkoumané doménové jméno, dále označení, jestli je jméno validní nebo DGA. Nedílnou součástí datové sady musí být charakteristika doménového

jména v podobě numerických atributů, jenž budou podrobněji popsány v kapitole 4. Na takové velké množství dat je vhodné využít software s grafickým rozhraním, který usnadňuje práci s velkým množstvím dat. Kromě toho, že nám programy umožní přehlednou vizualizaci dat, tak můžeme aplikovat na datovou sadu algoritmy strojového učení. Aplikování strojového učení pomocí těchto nástrojů je sice často pomalé a neefektivní, ale pro rychlé prototypování a vyzkoušení různých algoritmů se to hodí. Umožní nám to snadno vybrat algoritmus, který implementujeme v cílovém řešení pomocí Pythonu. Mezi takové programy patří například Weka nebo Rapidminer.

Rapidminer

Rapidminer je softwarová platforma zaměřující se na analyzování dat. Má široké spektrum využití jak v komerční, tak akademické sféře. Nástroj se využívá k formátování velkého množství dat, které jsou později analyzovány a aplikují se na ně metody strojového učení a různé prediktivní analýzy [9].
Aplikace umožňuje procházet všemi fázemi strojového učení i včetně vizualizace, a proto tato platforma bude využita k natrénování výsledného modelu. Díky školní licenci je nástroj k dispozici zdarma, jinak je v komerční oblasti zpoplatněn. Rapidminer podporuje vstupní data v běžném tabulkovém formátu `.csv` nebo `.xls`. Umožňuje také nahrát data rovnou z databáze.

Způsob práce s daty je reprezentován pomocí procesů. Proces podle zvoleného typu operátoru provede nad daty danou úlohu za účelem načtení, transformace, analýzy nebo exportování dat. Každý proces má svůj vstup a výstup. Jednotlivé procesy se mohou řetězovitě propojovat v rámci grafického rozhraní, jehož ukázku máme na obrázku 3.5. Propojováním procesů můžeme dosáhnout komplexních modifikací a analýz nad zpracovávanými daty. Rapidminer nám usnadňuje vytváření posloupnosti procesů pomocí modulů, které vytvářejí abstrakci nad návrhem toku procesů. Mezi ně patří Turbo Prep a Auto Model. Turbo Prep slouží k vizualizaci datových sad pomocí grafů a modifikaci za účelem zkvalitnění sady. Auto model umožňuje projít všemi fázemi strojového učení automatizovaným způsobem. Po výběru datové sady a specifikování, co chceme predikovat, si můžeme vybrat jaké algoritmy strojového učení chceme nad daty aplikovat. Po vyhodnocení zvolených algoritmů můžeme srovnat jednotlivé algoritmy na základě výsledných grafů a tabulek. Funkce Auto Modelu bude právě využívána pro rychlé prototypování a určení algoritmu, který bude nejvhodnější pro výslednou detekci DGA adres.



Obrázek 3.5: Ilustrace propojení procesů v rámci nástroje Rapidminer.

Weka

Oproti placenému Rapidmineru máme k dispozici open source software Weka. Tento software na analyzování dat a aplikování technik strojového učení byl vyvinut v rámci výzkumu na univerzitě Waikato na Novém Zélandu. Weka byla vydána pod GNU (General Public License), a je tak volně šiřitelná a modifikovatelná. Opět uživateli umožňuje analyzovat data prostřednictvím uživatelského rozhraní. Funkcionalita není možná tak bohatá jako u Rapidmineru (zdrojové kódy Weky byly využity při vývoji Rapidmineru), ale i tak podporuje všechny běžné techniky jako je předzpracování, shlukování, klasifikaci, regresi nebo vizualizaci dat. Dále podporuje filtrování relevantních atributů u datové sady. Vstupní data jsou opět podporována v tabulkovém formátu nebo pomocí databáze. Weka byla zpočátku vyvíjena v jazyce C a později se přešlo na multiplatformní Javu [10].

Kapitola 4

Návrh klasifikátoru

V následující kapitole bude rozebráno, jakým způsobem získáme klasifikátor doménových jmen s využitím technik strojového učení. Na úvod bude popsána sada atributů, které budeme extrahovat a počítat z doménových jmen. Po získání atributů bude vytvořena datová sada, která bude obsahovat oanotované doménová jména spolu s vypočítanými atributy. Na trénovací sadě bude aplikováno několik zmiňovaných algoritmů strojového učení a tyto modely budou následně porovnány a bude vybrán nejvhodnější model. Na závěr bude ukázáno jakým způsobem můžeme zvolený model automatizovaně exportovat, aby mohl být reálně použit.

4.1 Popis atributů

U každého zpracovávaného doménového jména bude určena sada atributů (vlastnosti doménového jména), pomocí kterých bude trénován výsledný model. Nejzákladnější počítané atributy jsou délka, entropie, podíl souhlásek, numerických a nealfanumerických znaků. Po získání základních atributů přijde na řadu analýza podřetězců domény. Budou extrahovány n-gramy, pomocí kterých budou počítány další atributy. Mezi ně bude patřit ukazatel podobný Jaccardovu indexu. Budeme zjišťovat podobnost domény vůči validními nebo DGA adresám a také jejich vzájemnou vzdálenost. Dále získáme úroveň zanoření subdomén, a existence specifických řetězců jako je "www". Dalším kritériem bude, jestli se doména skládá ze slov v anglickém slovníku. Na konec bude provedeno dotazování na doménové jméno pomocí nástrojů Whois a Host. Nástroje nám umožní ověřit existenci a získat další informace o doménovém jméně. Všechny atributy jsou získány na základě experimentování a ověřování, zda jsou atributy vhodné a přispějí tak ke kvalitnímu klasifikátoru. Některé atributy jsou přímo převzaty z odborných článků a další mohou být vlastnoručně pozměněny a upraveny pro individuální implementaci.

4.1.1 Základní atributy

K výpočtu základních atributů nebudou potřeba žádné další data kromě samotného doménového jména. Popis atributů bude ilustrován vždy na dvou příkladech doménových jmen. První adresa byla vygenerovaná pomocí DGA a druhá reprezentuje validní adresu. Výpočet atributu bude vždy aplikován na doménovou adresu bez domény nejvyšší úrovně. Atributy byly zvoleny buď na základě odborného článku [27], nebo na základě charakteristiky DGA adresy. Níže budou následovat ukázky atributů pro zvolené referenční domény.

axwscwsslmiagfah.com

- délka: 16
- podíl souhlásek: 0.75
- podíl numerických a nealfanumerických znaků: 0.0 & 0.0

facebook.com

- délka: 8
- podíl souhlásek: 0.5
- podíl numerických a nealfanumerických znaků: 0.0 & 0.0

Další vlastnost, která bude počítána je entropie řetězce. Bude využit vztah pro Shannonovu entropii. Shannonova entropie je důležitou metrikou v informační teorii, určuje neuspořádanost řetězce. Méně pravděpodobný výskyt řetězce nese více informací, a proto také obsahuje větší rozsah náhodné veličiny, která nese jednotlivé znaky řetězce [14]. Dostaneme-li diskrétní náhodnou veličinu **X** obsahující **N** znaků řetězce, který se skládá z **n** rozdílných symbolů, potom můžeme vypočítat entropii 4.1 pomocí vztahu:

$$H(X) = -\sum_{i=1}^{n} \frac{count_i}{N} \log_2 \frac{count_i}{N} \tag{4.1}$$

kde $\mathbf{count_i}$ je počet znaků $\mathbf{n_i}$ [20].
Výsledné entropie pro naše doménové jména jsou:

- $H(axwscwsslmiagfah) = 3.2806$
- $H(facebook) = 2.75$

Vyšší hodnota značí vetší rozsah náhodné proměnné, a proto můžeme říct, že jde o nahodilý řetězce. Vydělením hodnot entropie řetězců jeho délkou získáme metrickou entropii. Hodnoty blížící se k jedničce znamenají rovnoměrně rozložený řetězec. Vypočítané hodnoty pro naše referenční doménová jména:

- $H_m(axwscwsslmiagfah) = 0.2050$
- $H_m(facebook) = 0.3438$

4.1.2 Analýza podřetězců

V této fázi bude analyzována doména na úrovni podřetězců. Pro výpočet komplexnějších atributů budou potřeba externí data, která budou potřeba nahrát vždy při startu klasifikátoru. Mezi základní atributy extrahované z podřetězců patři počet zanoření doménové úrovně nebo výskyt podřetězce "www" [27]. V další fázi se doména rozdělí na n-gramy pomocí kterých získáme další množinu atributů na základě podobnosti vůči vzorové skupině domén. Jedna skupina bude reprezentovat validní domény a druhá DGA domény. Dále budou detekovány běžně známé slova z anglického slovníku. Inspirací pro zmíněné atributy se stal článek o tvorbě DGA detektoru [13].

N-gramy

Obecně je n-gram považován za jakoukoliv posloupnost **n** položek v textu nebo mluveném slově. Položky zastupují slabiky, fonémy, písmena nebo slova. V naší problematice můžeme definovat n-gram jako všechny posloupnosti **n** po sobě jdoucích znaků v analyzovaném doménovém jméně [2]. Vygenerované n-gramy budou nabývat délky 3,4,5. Následující tabulky ilustrují extrahování n-gramů z řetězce u validní a DGA domény. V tabulce 4.1 můžeme sledovat vygenerovanou sadu n-gramů pro referenční validní doménu *facebook.com* a pro umělou doménu *axwscwsslmiagfah.com*. Získané n-gramy nám poslouží jako podklad pro výpočet dalších atributů.

Tabulka 4.1: Ukázka extrahování n-gramů

3-gramy	
axwscwsslmiagfah	axw xws wsc scw cws wss ssl slm lmi mia iag agf gfa fah
facebook	fac ace ceb ebo boo ook
4-gramy	
axwscwsslmiagfah	axws xwsc wscw scws cwss wssl sslm slmi lmia miag iagf agfa gfah
facebook	face aceb cebo eboo book
5-gramy	
axwscwsslmiagfah	axwsc xwscw wscws scwss cwssl wsslm sslmi slmia lmiag miagf iagfa agfah
facebook	faceb acebo ceboo ebook

Jaccardův index

Výpočet Jaccardova indexu slouží k určení podobnosti dvou množin. Pomocí průniku nad množinami získáme, kolik prvků mají množiny společných a pomocí sjednocení množin zase získáme celkový počet unikátních prvků. Podělením těchto dvou údajů získáme jak moc jsou tyto množiny podobné. Čím se bude výsledek blížit 1, tak tím bude vyšší podobnost. Níže je uveden vztah 4.2 pro výpočet Jaccardova indexu [7].

$$J(X,Y) = |X \cap Y| \div |X \cup Y| \tag{4.2}$$

Aby mohl být využit Jaccardovův index pro detekci DGA adres, tak je potřeba vytvořit dvě množiny n-gramů. První množina bude obsahovat množiny n-gramů vytvořené z validních doménových jmen[1] a druhá naopak z nevalidních DGA jmen[2]. Právě klasifikované doménové jméno bude také rozděleno na n-gramy. Z důvodu rychlosti výpočtu bylo potřeba upravit i vzorec Jaccardova indexu 4.3. Ve jmenovateli zůstane pouze počet n-gramů právě analyzované domény.

$$J_1(X,Y) = |X \cap Y| \div |X| \tag{4.3}$$

kde **X** je množina n-gramů klasifikovaného jména a **Y** je množina n-gramů reprezentující buď validní nebo DGA skupinu domén.

Pro výpočet podobností se vezme množina n-gramů zrovna klasifikovaného jména a pomocí upraveného výpočtu Jaccardova indexu se spočítá podobnost nebo-li poměrné zastou-

[1] http://s3.amazonaws.com/alexa-static/top-1m.csv.zip
[2] https://osint.bambenekconsulting.com/feeds/

pení v rámci validní i nevalidní množiny n-gramů. Výpočet je dělán pro n-gramy o velikosti 3,4,5 a následně jsou jednotlivé hodnoty zprůměrovány. Získáme tak dvě sady hodnot vyjadřující podobnost. Jedna bude vyjadřovat podobnost s validní množinou 4.2 a druhá bude reflektovat podobnost s nevalidní množinou 4.3. Hodnota nabývající jedničky vyjadřuje maximální shodu.

Tabulka 4.2: Zastoupení validní n-gramů

	3-gramy	4-gramy	5-gramy	průměr
axwscwsslmiagfah	1.00	0.69	0.00	0.56
facebook	1.00	1.00	1.00	1.00

Tabulka 4.3: Zastoupení DGA n-gramů

	3-gramy	4-gramy	5-gramy	průměr
axwscwsslmiagfah	1.00	1.00	0.42	0.81
facebook	1.00	1.00	0.75	0.92

Rozdílem průměrných výsledků získáme náhled, jak moc se klasifikovaná doména podobá dané třídě domén. Průměr DGA podobnosti hraje ve výpočtu roli menšence a průměr validní podobnosti má roli menšitele, a proto čím vyšší číslo získáme, tak tím více zkoumaná adresa připomíná DGA adresu. Výpočet vzdálenosti podobností je ilustrován níže:

- $axwscwsslmiagfah : 0.81 - 0.56 = 0.24$
- $facebook : 0.92 - 1 = -0.08$

Zpracování subdomén

Subdoména je samostatná doména, která je součástí další domény. Zanořování může probíhat až do 127 úrovní. Nejpravější doména se nazývá TLD (top level domain), na kterou se zleva napojují další subdomény z důvodů vytvoření více podsítí [19]. Mějme doménové jméno "d.c.b.a.sk". Rozdělením domény za každou tečkou získáme pravě jednu TLD - **.sk** a 4 subdomény (**d.c.b.a, d.c.b, d.c, d**), každá doména ze závorky je pravě subdoménou zbylého doménového jména.

Díky analýze subdomén můžeme získat dva atributy. První je počet zanořených subdomén, které tvoří doménové jméno. Druhý atribut je tvořen na základě seznamu veřejně známých subdomén[3]. Každá subdoména bude testována na přítomnost ve výše uvedeném seznamu a získáme tak počet veřejně známých subdomén obsažených ve zkoumané doméně.

[3] https://github.com/publicsuffix/list/blob/master/public_suffix_list.dat

Shoda ve slovníku

Ve volně dostupném slovníku anglických slov[4] bude vyhledávána maximální shoda podřetězců domény, tak aby se získal nejdelší možný řetězec. Na obrázku 4.1 můžeme vidět postupné nalezení maximální možné shody slova "software".

Obrázek 4.1: Ilustrace nalezení maximální shody slova ve slovníku.

V tabulce 4.4 můžeme u vzorových domén vidět vyextrahovaná anglická slova, která jsou obsažena ve slovníku anglických slov. Výsledná hodnota atributu reprezentuje podíl mezi součtem délek nalezených slov a celkovou délkou domény. Na základě experimentů bylo zjištěno, že pokud se hledají jen slova, která na sebe přímo navazují, tak atribut vykazuje menší korelaci s predikovanou třídou. Na základě zmíněného poznatku se extrahují i slova, která přímo nenavazují na předchozí posloupnost znaků. Například doména *nvjwoofansjbhcomen* z tabulky 4.4 ilustruje extrahovaní na sobě nezávislých slov.

Tabulka 4.4: Zastoupení anglických slov a výpočet atributu

doména	nalezená slova	hodnota atributu
facebook	face book	1.00
softwareupdate	software update	1.00
000directory	directory	0.80
nvjwoofansjbh	woof ans	0.54
ab1145b758c30	-	0.00
axwscwsslmiagfah	-	0.00

4.1.3 Dotazování na doménu

Díky unixovým nástrojům jako je Host a Whois můžeme získat užitečné informace o doménovém jméně. Pro účely detekce nás bude hlavně zajímat, zda adresa existuje nebo ne. K tomu aby byla zachycena podvržená doménová adresa stačí znalost o její dostupnosti. Pravě zmíněnými nástroji můžeme tuto informaci získat. Při používání nástrojů narážíme ale na problém související s rychlostí. Po provedení experimentů s nástrojem Host bylo zjištěno, že průměrné vyhodnocení jedné neexistující domény trvá 1.3787 s (záleží na internetovém připojení). Ze získaného časového údaje můžeme říct, že při detekování 10 000 záznamů by se prodloužil čas klasifikace o zhruba 3.8 h, což je absolutně nepřijatelné. I kdyby díky rychlému internetovému připojení bylo vyhodnocení několikrát rychlejší, tak by se nástroje při klasifikaci nedaly použít, protože by zpoždění bylo stále v desítkách minut. Nicméně je vhodné uvést stručné informace o nástrojích, které byly v rámci práce používány.

[4] https://github.com/dwyl/english-words

Host

Host je jednoduchý nástroj na překládání doménových jmen na IP adresu a opačně. Překlad probíhá pomocí DNS serveru [11]. V případě, že se překlad nepovede, označí se doména za neexistující (NXDOMAIN). Právě Invalidní doména může značit, že se jedná o DGA adresu. Na obrázku 4.2 můžeme vidět využití nástroje host v praxi. V první části je proveden překlad nad validní adresou *facebook.com*, která je úspěšně přeložena na IP adresu. Druhá část znázorňuje překlad neexistující DGA domény *axwscwsslmiagfah.com*.

```
pc ~ $ host facebook.com
facebook.com has address 157.240.30.35
facebook.com has IPv6 address 2a03:2880:f13d:83:face:b00c:0:25de
facebook.com mail is handled by 10 smtpin.vvv.facebook.com.
pc ~ $ host axwscwsslmiagfah.com
Host axwscwsslmiagfah.com not found: 3(NXDOMAIN)
```

Obrázek 4.2: Ukázka překladu validní adresy a NXDOMAIN adresy pomocí nástroje host.

Whois

Whois je komplexnější nástroj než Host, vygeneruje větší množství informací o hledaném objektu, ale o to je složitější parsování výstupu. Whois hledá požadovaný objekt v databázích a dotazuje se severů na informace o objektu [12]. Výsledkem je souhrn informací o registrátorovi doménového jména a organizaci, která doménu registrovala. Pokud se nepodaří získat žádné informace, tak je výstupem chybová hláška. Výsledky vyhledávání jsou zobrazeny na obrázku 4.3. Opět se dotazujeme na DGA adresu a následně na validní adresu.

```
pc ~ $ whois axwscwsslmiagfah.com
No match for domain "AXWSCWSSLMIAGFAH.COM".
pc ~ $ whois facebook.com
Domain Name: FACEBOOK.COM
Registry Domain ID: 2320948_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.registrarsafe.com
Registrar URL: https://www.registrarsafe.com
Updated Date: 2019-10-17T18:52:06Z
Creation Date: 1997-03-29T05:00:00Z
Registrar Registration Expiration Date: 2028-03-30T04:00:00Z
Registrar: RegistrarSafe, LLC
Registrar IANA ID: 3237
Registrar Abuse Contact Email: abusecomplaints@registrarsafe.com
Registrar Abuse Contact Phone: +1.6503087004
Domain Status: clientDeleteProhibited https://www.icann.org/epp#clientDeleteProhibited
Domain Status: clientTransferProhibited https://www.icann.org/epp#clientTransferProhibited
Registrant Organization: Facebook, Inc.
Registrant Street: 1601 Willow Rd
Registrant City: Menlo Park
Registrant State/Province: CA
Registrant Postal Code: 94025
```

Obrázek 4.3: Hledání neexistujícího a existujícího objektu pomocí nástroje Whois.

4.2 Trénování modelu

V momentě kdy získáme dostatečné množství anotovaných dat, u kterých budeme mít vypočítané výše uvedené atributy, tak tyto data budeme analyzovat v prostředí Rapidminer. V tabulce 4.5 můžeme vidět ukázkovou strukturu formátu vstupních dat, nad kterými bude probíhat trénování modelu. Tabulka kvůli přehlednosti popisuje jen některé atributy. Výpočet atributů probíhal prozatím v prostředí programovacího jazyka Python, který nám umožnil rychle tyto výpočty prototypovat. Pro prvotní vývoj klasifikátoru byla použita předem oanotovaná datová sada. Datová sada zahrnuje zhruba 130 000 záznamů obsahující doménová jména a štítek říkající, zda je jméno legitimní nebo vygenerované pomocí DGA.

Tabulka 4.5: Formát vstupních dat k natrénování modelu v Rapidminer

Doména	Třida	Délka	Podíl souhlásek	Entropie	Validní n-gramy
seznam.cz	validní	6	0.67	2.58	1.00
facebook.com	validní	8	0.50	2.75	1.00
google.com	validní	6	0.50	1.92	1.00
youtube.com	validní	7	0.43	2.52	1.00
10bh6urqdn5tp1fty8e23yucx0.net	DGA	26	0.54	4.32	0.29
axwscwsslmiagfah.com	DGA	16	0.75	3.28	0.56
bmmikhkmdbfv.ru	DGA	12	0.92	2.86	0.60

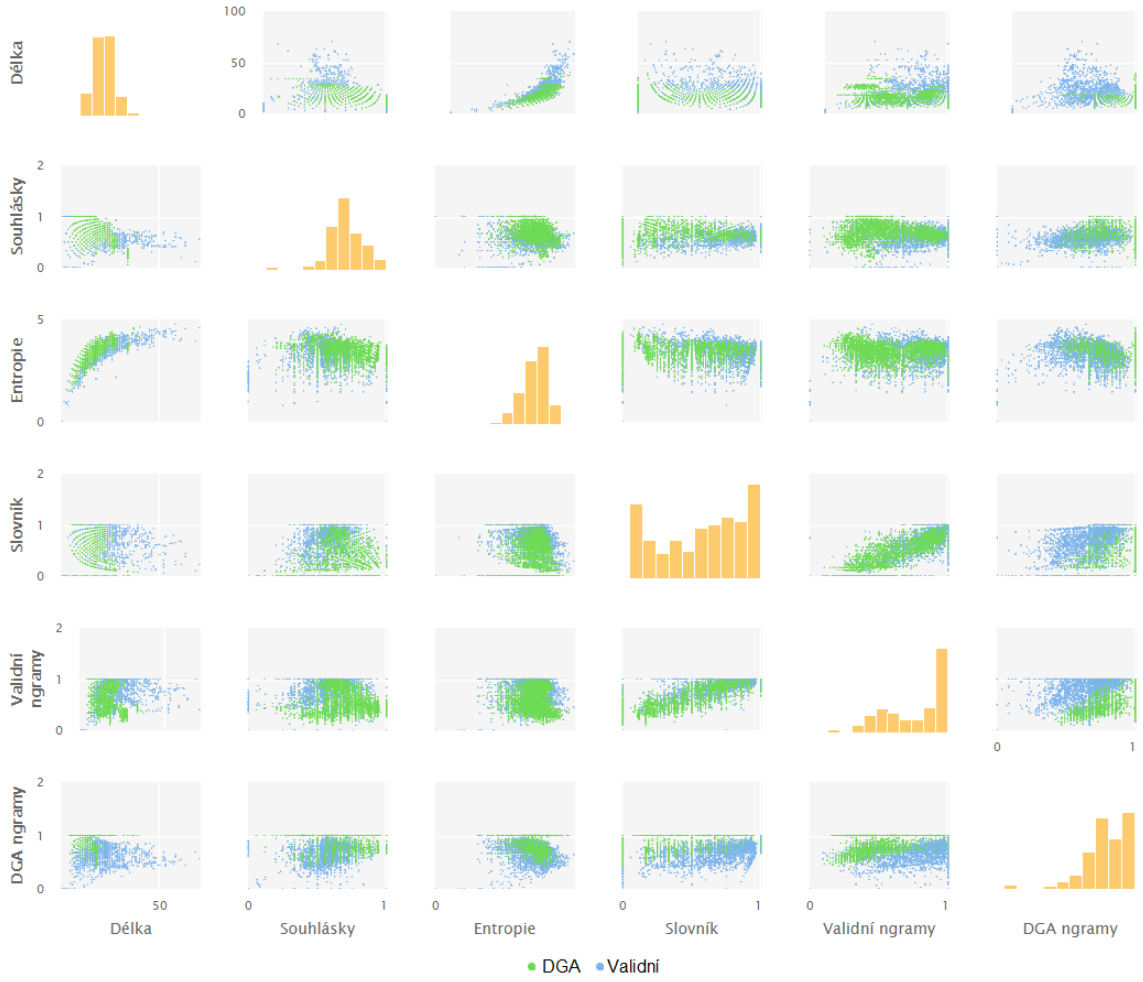
Výběr modelu

Vytvořená trénovací datová sada byla nahrána do prostředí Rapidmineru ve formátu CSV. Rapidminer nám umožňuje po nahrání datové sady nadále data upravovat, filtrovat a analyzovat. Na začátku je potřeba zvolit, který sloupec budeme chtít predikovat a poté se zobrazí seznam atributů. U každého atributu se nám zobrazí jeho skóre, jak moc je atribut vhodný pro predikci cílového sloupce (třídy). Některé atributy spolu mohou silně korelovat, a proto je vhodné vybrat jen relevantní atributy. Máme k dispozici obsáhlou škálu nástrojů, díky které můžeme vygenerovat automatizovaným procesem více modelů naráz. K automatickému generování modelů využijeme zmiňovaný Auto Model, který je ideální pro tento úkol. Máme na výběr z několika modelů, které se budou aplikovat:

- Naive Bayes
- Generalized Linear Model
- Logistic Regression
- Fast Large Margin
- Deep Learning
- Decision tree
- Random Forest
- Gradient Boosted Trees
- Support Vector Machine

Jako uživatelé si můžeme vizualizovat přesnost modelů, jejich časovou náročnost, nebo jejich chybovost. Můžeme si prohlédnout každý model zvlášť, kde získáme podrobnější informace o jednotlivých atributech a jejich váze vůči danému modelu. Dále můžeme nahlédnout na konkrétní data a jejich výslednou klasifikaci.

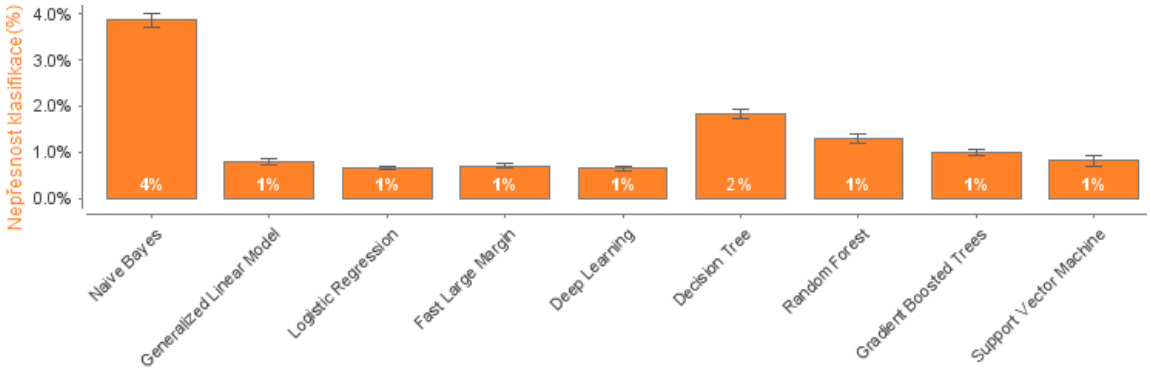
Obrázek 4.4 pomocí matice grafů vyjadřuje vzájemnou závislost mezi hlavními spojitými atributy. V diagonále můžeme sledovat histogram rozložení hodnot. Čím více spolu atributy korelují, tím více graf splývá do lineární závislosti (můžeme aproximovat přímkou). Ukázku kladné lineární závislosti můžeme vidět mezi atributem slovníku a validních n-gramů. Dále vykazuje kladnou závislost délka s entropií. Po celkovém zhodnocení můžeme říct, že atributy mají nízkou korelační hodnotu. Nízká korelace mezi atributy je pozitivní, protože získáme pro klasifikaci různorodý a málo se ovlivňující vektor atributů.



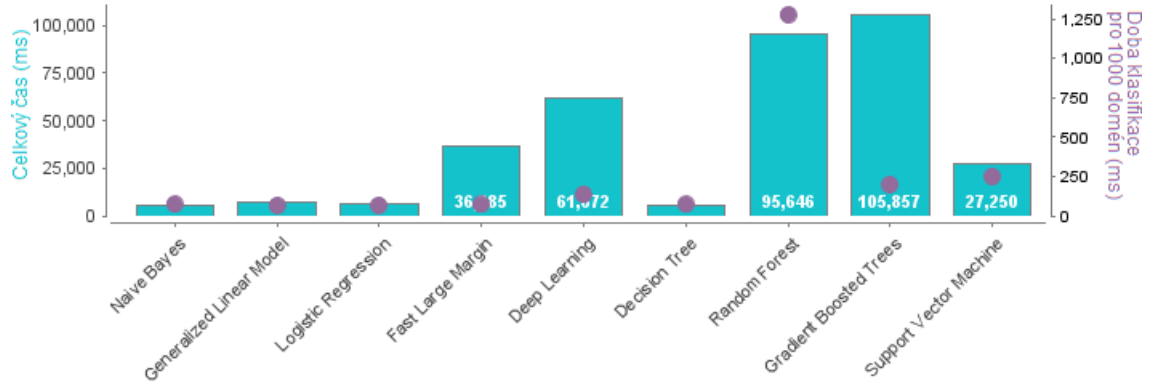
Obrázek 4.4: Grafová matice závislostí spojitých atributů. V diagonále je zobrazeno rozložení jednotlivých prvků (oranžová). Jednotlivé grafy znázorňují závislost mezi konkrétními atributy pomocí vynesení hodnot do bodového grafu. Bodový graf dodatečně rozlišuje DGA (zelená) a validní (modrá) adresu.

Pro prvotní analýzu byly vybrány všechny modely nabízené pro trénování. Po vyhodnocení se zobrazilo shrnutí a porovnání všech modelů. U každého modelu je možné zobrazit jeho podobu a základní údaje. Na obrázku 4.5 můžeme vidět sloupcový graf, kde je vyob-

razeno srovnání všech výpočetních modelů. Graf vyjadřuje procentuální míru nepřesnosti klasifikace. Sloupcový graf na obrázku 4.6 vyjadřuje časovou náročnost pro celkový výpočet modelu. Na stejném obrázku je kromě sloupcového grafu zanesen také bodový graf (fialové body) značící časovou náročnost samotné klasifikace.



Obrázek 4.5: Vizualizace procentuální nepřesnosti klasifikace na testovací sadě pro jednotlivé algoritmy strojového učení.



Obrázek 4.6: Vizualizace časové náročnosti modelů. Sloupcový graf (modře) značí celkovou dobu na trénování a testování modelu. Bodový graf (fialově) značí čistě dobu klasifikace.

Při výběru modelu byl kladen důraz na rychlost klasifikace (bodový graf na obrázku 4.6), která je stěžejní při vyhodnocování velkého množství domén. Podle získaných informací bylo rozhodnuto, že nejvhodnějším modelem bude rozhodovací strom. Rychlost jeho predikce patří mezi nejrychlejší a je dobře implementovatelný. I když má rozhodovací strom větší míru nepřesnosti, tak to nepředstavuje zásadní problém. Do budoucna bude snaha o vytvoření vlastní anotované datové sady, která zkvalitní výsledky klasifikace. Dále se může změnit způsob výpočtu některých atributů. Tyto kroky mohou vést ke zlepšení přesnosti výsledného modelu. Dalším důvodem, proč byl zvolen rozhodovací strom, je jeho přehlednost a jednoduchost.

4.3 Export modelu

Až se nám podaří úspěšně natrénovat výsledný model, tak přijde na řadu nasazení modelu a jeho konečná implementace v cílovém jazyce C v rámci NEMEA modulu. Tento proces se budeme snažit alespoň částečně zautomatizovat. Tuto automatizaci nám umožní provést prostředí Pythonu a jeho knihovny pro strojového učení `Scikit-learn`. Python disponuje různými knihovnami, které jsou určeny ke strojovému učení. Mezi ně patří například `TensorFlow, Keras, Theano, Pytorch` a zmiňovaný `Scikit-learn` [24]. Každá z knihoven má rozdílnou funkcionalitu, takže záleží na uživateli a jeho preferencích. V případě NEMEA modulu půjde hlavně o automatické transpilování[5] do nízkoúrovňového jazyka C. K tomu bude využit `sklearn-porter` modul v rámci `Scikit-learn` knihovny. Ukázka kódu 4.1 ilustruje, jakým způsobem se dá natrénovaný model rozhodovacího stromu pomocí Pythonu exportovat do jazyka C, který si později můžeme v našem NEMEA modulu volat dle potřeby.

```
from sklearn.datasets import load_iris
from sklearn.tree import tree
from sklearn_porter import Porter

# Load data and train the classifier:
samples = load_iris()
X, y = samples.data, samples.target
clf = tree.DecisionTreeClassifier()
clf.fit(X, y)

# Export:
porter = Porter(clf, language='java')
output = porter.export(embed_data=True)
print(output)
```

Výpis 4.1: Exportování modelu do jazyka C.

[5] transformování zdrojového kódu jednoho programovacího jazyka do druhého

Kapitola 5

Implementace NEMEA modulu

Kapitola bude popisovat praktickou implementaci navrženého řešení. V průběhu vývoje byla snaha dodržovat vytvořený návrh v předchozí kapitole. Postupně budou popsány jednotlivé komponenty modulu a způsob jejich implementace.

NEMEA (Network Measurements Analysis) se skládá z modulů, řídící jednotky a komunikujících rozhraních. Úkolem modulu je zachytávat na určitém rozhraní doménová jména, která jsou součástí datového toku. Všechna zachycená doménová jména budou předány modulu ke zpracování pomocí UniRec datové struktury. V první fázi modulu probíhá inicializační část, kde jsou vytvořeny nutné datové struktury, se kterými se bude v druhé fázi pracovat. Kromě inicializace rozhraní a UniRec struktury proběhne vytvoření tabulek s rozptýlenými položkami. Druhá fáze představuje samotné klasifikování doménového jména, kdy je adresa vyhodnocena buď jako validní nebo DGA. Před klasifikací každého doménového jména proběhne výpočet všech atributů, které jsou sepsány v kapitole s návrhem 4. V momentě, kdy získáme množinu všech atributů, tak jsou předány prediktivnímu modelu, který provede binární klasifikaci. Doménová jména, která byla vyhodnocena jako DGA, jsou nadále předány DGA reportéru, který uloží incident do databáze pro pozdější analýzu.

Při výběru rozhodovacího modelu byl použit software Rapidminer, jenž nám umožnil zanalyzovat a srovnat více modelů. Jako finální klasifikační model byl vybrán rozhodovací strom. Pro generování rozhodovacího stromu je využita Python knihovna strojového učení `sklearn`. Byl vytvořen jednoduchý skript, který načte trénovací datovou sadu, poté nad daty vytvoří rozhodovací strom a ten je následně exportován do jazyka C. Automatické generování nám usnadní průběžnou adaptaci pro zpřesnění výsledků a zlepšení výsledného modelu.

5.1 Architektura modulu

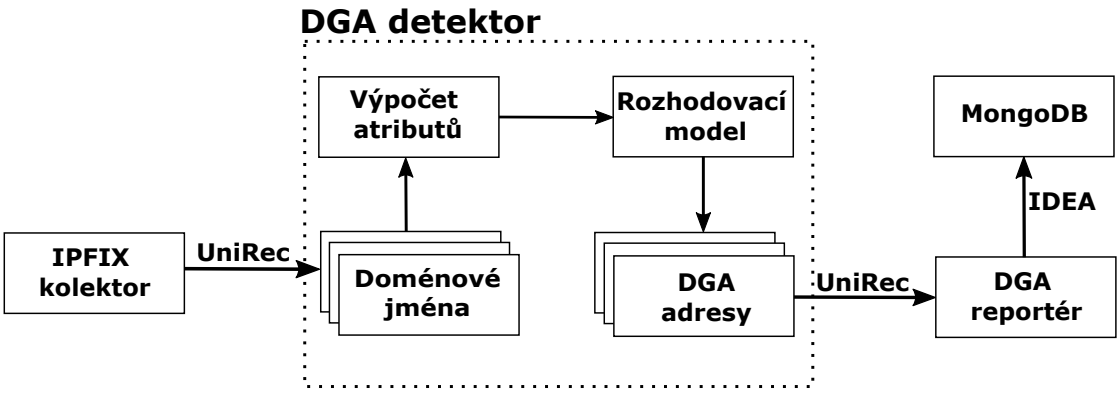
V rámci NEMEA frameworku je rozhraní každého modulu pevně dané prostřednictvím TRAP knihovny, která byla uvedena v sekci 3.2. Data, která bude mít modul na vstupu budou ve formátu UniRec záznamu. Záznam bude obsahovat informace o toku zachyceném na IPFIX sondě. Data ze sondy jsou předány IPFIX kolektoru, jehož výstupem bude právě UniRec záznam pro DGA detektor. V tabulce 5.1 můžeme vidět ukázku jednoho toku zachyceného pomocí IPFIX sondy. Pro přehlednost jsou vybrány jen relevantní informace.

Tabulka 5.1: Ukázka vstupního záznamu v rámci UniRec souboru

ipaddr DST_IP	65.55.117.41
ipaddr SRC_IP	195.113.40.9
uint16 DNS_FLAGS	33 795
time TIME_FIRST	2020-08-04T13:46:24.585
uint16 DST_PORT	53
uint16 SRC_PORT	35 345
uint16 DNS_Q_TYPE	1
string DNS_Q_NAME	static-asm-skype.trafficmanager.net

Schéma konceptu implementace

Na obrázku 5.1 můžeme vidět jakou bude mít implementace výslednou podobu. Součástí implementace je samotný DGA detektor a dále modul zajišťující hlášení nalezených DGA adres, které se uloží ve formátu IDEA do databáze. DGA detektor načte doménová jména z UniRec záznamu a nad každou doménovou adresou provede výpočet vektoru atributů. Vektor atributů slouží jako vstup pro rozhodovací model, na základě kterého provede výslednou klasifikaci. Pokud je adresa vyhodnocena jako DGA, tak se připraví pro export do výstupního UniRec záznamu. Jakmile se zpracují všechny vstupní doménová jména, tak se výsledný UniRec soubor předá právě DGA reportéru, který provede uložení do databáze. Kromě DGA adres se do databáze uloží také zdrojová IP adresa a typ incidentu (botnet).



Obrázek 5.1: Architektura detekčního modulu.

5.2 Inicializace modulu a externích dat

Prvním krokem pro vývoj libovolného modulu je zprovoznění celého frameworku NEMEA. Systém byl přeložen a nainstalován pomocí zdrojových kódů[1]. Celý proces je usnadněn díky automatizovaným nástrojům pro překlad. Jako první krok bylo potřeba přizpůsobit cílový systém pomocí příkazu `./configure` a následně byl proveden samotný překlad příkazem `make` [33]. NEMEA má k dispozici ukázkový modul, který byl použit jako vzorová kostra pro následnou implementaci. Původní modul s názvem `example_module` byl přejmenován na `dga_detector` a také se poupravil Makefile pro potřeby modulu (nalinkování matematické knihovny a přidání zdrojového souboru s rozhodovacím klasifikátorem).

Kostra ukázkového modulu obsahuje základní informace o modulu a makra pro zpracování argumentů. Dále obsahuje inicializaci komunikujícího rozhraní pomocí TRAP knihovny. Vytvoří se vstupní a výstupní UniRec struktury a alokuje se pro ně potřebná paměť. Po základní inicializaci následuje cyklus, který v každé iteraci načte jeden záznam z UniRec a ten může být poté zpracován. Na závěr proběhne dealokace všech používaných zdrojů a modul může být ukončen. V následující části bude popsáno jakým způsobem byl ukázkový modul rozšířen, aby z něj vznikl požadovaný detektor DGA adres.

Externí data

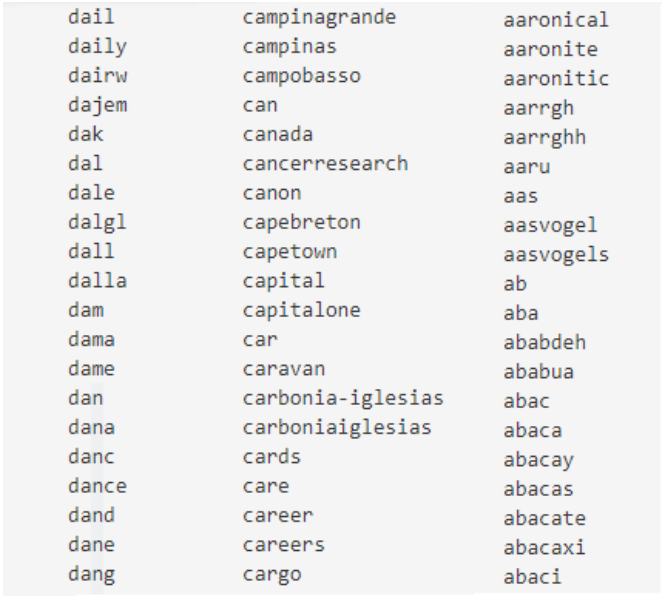
Jak už bylo zmiňováno v návrhu, tak pro strojové učení je důležité získat kvalitní datovou sadu. Pro získání klasifikačního modelu byla využita datová sada s předklasifikovanými doménovými jmény. U této datové sady je potřeba vypočítat atributy, na základě kterých bude natrénován klasifikační model. Pro atributy, jako je shoda slova ve slovníku, výskyt n-gramů nebo počítaní veřejně známých subdomén, je potřeba načíst externí data, aby se výpočet mohl uskutečnit. Pro shodu ve slovníku je použit seznam anglických slov. Seznam validních n-gramů je extrahován ze seznamu Alexa top 1 milion a seznam DGA n-gramů je získán ze stránky Bambenek Feeds. Při analyzování subdomén je použit seznam veřejně známých subdomén. Veškeré zdroje zmíněných dat jsou uvedeny v kapitole s návrhem u jednotlivých atributů 4.1.2.

Všechny výše uvedená data jsou uloženy ve formátu textových souborů, které jsou v modulu otevřeny a jejich obsah je nahrán do programu. Podobu dat můžeme vidět na obrázku 5.2, kde v prvním sloupci jsou vyextrahované validní n-gramy, uprostřed je seznam veřejně známých subdomén a v posledním sloupci je seznam anglických slov. V tabulce 5.2 lze vidět množství dat.

Tabulka 5.2: Množství externích dat

Validní n-gramy	1 315 471
DGA n-gramy	5 641 953
Veřejné subdomény	6 354
Anglická slova	370 104

[1] https://github.com/CESNET/Nemea



dail	campinagrande	aaronical
daily	campinas	aaronite
dairw	campobasso	aaronitic
dajem	can	aarrgh
dak	canada	aarrghh
dal	cancerresearch	aaru
dale	canon	aas
dalgl	capebreton	aasvogel
dall	capetown	aasvogels
dalla	capital	ab
dam	capitalone	aba
dama	car	ababdeh
dame	caravan	ababua
dan	carbonia-iglesias	abac
dana	carboniaiglesias	abaca
danc	cards	abacay
dance	care	abacas
dand	career	abacate
dane	careers	abacaxi
dang	cargo	abaci

Obrázek 5.2: Ukázka externích dat. První sloupec jsou validní n-gramy, druhý sloupec veřejné subdomény a poslední sloupec jsou anglická slova.

Způsob uložení dat v rámci programu vyžaduje efektivní a rychlý přístup, jelikož je potřeba při analýze každé doménové adresy pracovat s těmito daty. Datová struktura, která disponuje rychlým přístupem k velkému množství dat je tabulka s rozptýlenými položkami.

Tabulka s rozptýlenými položkami

Tabulka s rozptýlenými položkami neboli hashovací tabulka je abstraktní datový typ. Vyhledávání v hashovací tabulce má konstantní dobu přístupu. Tabulka má v krajním případě lineární časovou složitost, a to za předpokladu, že tabulka obsahuje velké množství kolizí. Jinak se její průměrná časová složitost blíží k O (1). Jako hashovací funkce byla použita djb2, kterou můžeme vidět v ukázce kódu 5.1. Tato funkce je doporučována pro hashování řetězců, kvůli relativně nízkému počtu kolizí.

```
unsigned long hash(char *str,uint32_t SIZE)
{
   unsigned long hash = 5381;
   int c;

   while (c = *str++)
      hash = ((hash << 5) + hash) + c; /* hash * 33 + c */

   return hash%SIZE;
}
```

Výpis 5.1: Hashovací funkce djb2.

Pro každou sadu dat je vytvořena vlastní hashovací tabulka. Při určování velikosti tabulek byl zvolen faktor zaplnění 70 % [18]. To znamená, že celková velikost tabulky bude

mít o 30 % vetší kapacitu, než je požadovaný počet vkládaných položek. Zvýšením kapacity tabulek se snažíme snížit pravděpodobnost kolizí. V kódu 5.2 můžeme vidět strukturu jednotlivých hashovacích tabulek. Položka tabulky obsahuje pouze uložená data, která slouží zároveň jako klíč. Používá se explicitní zřetězení, kde každá položka obsahuje ukazatel na další kolizní prvek a vytváří se tak jednosměrně vázaný seznam.

```
#define SIZE_NGRAM 1874958
#define SIZE_DIC 528720
#define SIZE_SUFFIX 9077
#define SIZE_DGA 7522604

typedef struct my_string string_item;
struct my_string {
   char *data;
   string_item *next;
};

string_item* ngram_table[SIZE_NGRAM];
string_item* dga_table[SIZE_DGA];
string_item* dic_table[SIZE_DIC];
string_item* suffix_table[SIZE_SUFFIX];
```

Výpis 5.2: Struktura a velikost hashovacích tabulek.

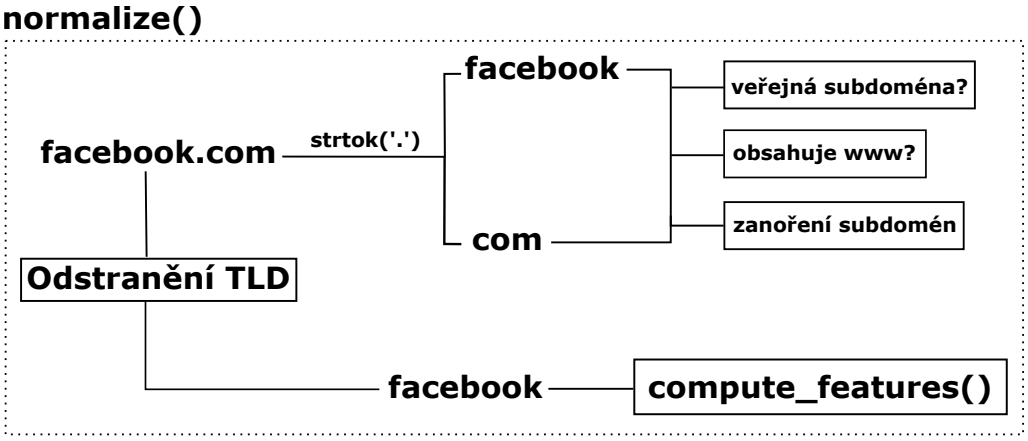
5.3 Zpracování doménové adresy

V momentě, kdy máme v paměti načtena všechna potřebná data, tak může přijít na řadu analyzování jednotlivých domén. V následující sekci bude detailněji popsán způsob, jak se z dané domény získá výsledný vektor atributů. U adresy se budou extrahovat uvedené atributy:

- délka
- podíl souhlásek
- podíl numerických a nealfanumerických znaků
- entropie
- zanoření subdomén
- přítomnost www subdomény
- počet veřejně známých subdomén
- poměrné zastoupení validních n-gramů
- poměrné zastoupení DGA n-gramů
- podobnost domény s DGA adresami
- poměrné zastoupení anglických slov

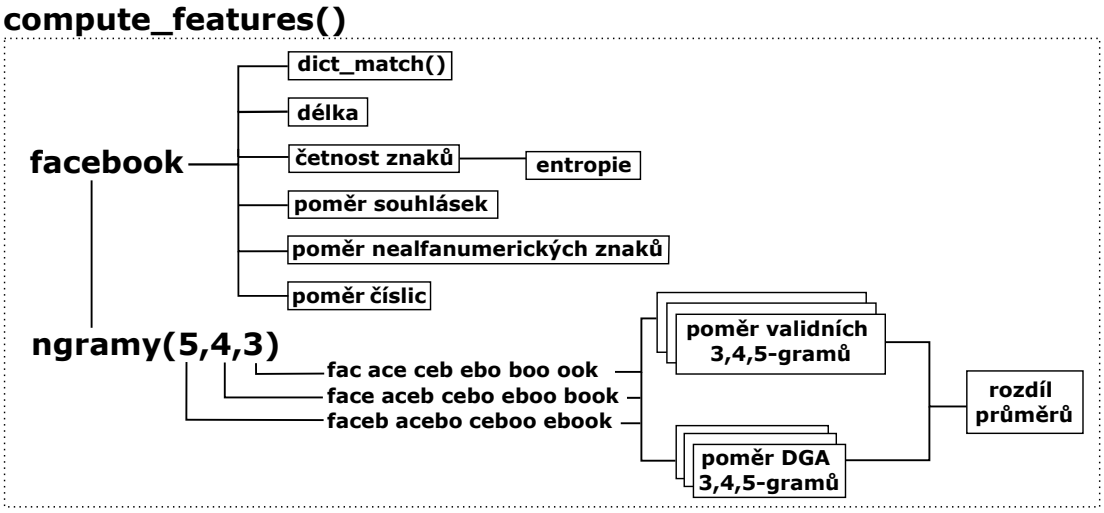
Úplně na začátku je celá doménová adresa upravena tak, aby neobsahovala velká písmena. Následně je doména předána funkci `normalize()` 5.3. V rámci funkce se doména

rozdělí na jednotlivé subdomény (tokenizace řetězce na základě tečky). V průběhu procesu se získá počet zanoření subdomén. Dále se pro každou subdoménu dotážeme, zda se vyskytuje v hashovací tabulce s veřejně známými subdoménami. Pokud ano, tak se navýší počítadlo veřejně známých subdomén. Můžeme také zjistit, jestli doména obsahuje *"www"* řetězec. Jednotlivé subdomény opět konkatujeme s vynecháním poslední (TLD) subdomény. Získáme tak řetězec, který už neobsahuje tečky a TLD doménu. Vyextrahovaný řetězec následně předáme funkci pro výpočet dalších atributů.



Obrázek 5.3: Normalizování domény provede sloučení jednotlivých subdomén a odstraní doménu nejvyšší úrovně. Během procesu se spočítají atributy související se subdoménami (počet veřejných subdomén, výskyt *"www"* řetězce a celkový počet subdomén).

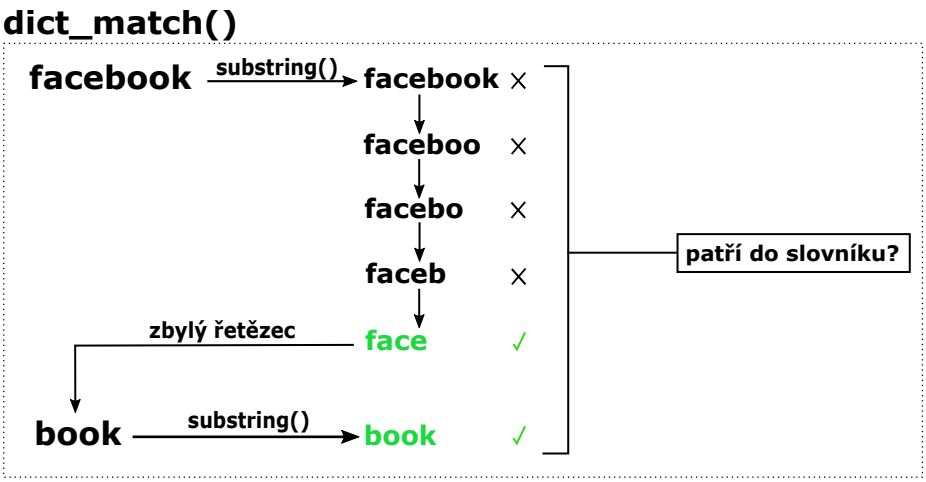
Normalizovaná doména je předána na vstup funkce `compute_features()`, jejíž schéma je vyjádřeno na obrázku 5.4. Nad doménou se poté vypočítá sada základních atributů, jako je délka, poměr souhlásek, číslic a nealfanumerických znaků. Při určování poměru konkrétního typu znaků se využívá numerická hodnota znaku v ASCII tabulce. Pro výpočet entropie se musí vypočítat četnost znaků, jenž je předána vzorci pro výpočet entropie 4.1.1.



Obrázek 5.4: Schéma výpočtu atributů. V rámci funkce **compute_features()** se postupně získá množina atributů, jenž je reprezentována obdélníkovými bloky.

Když máme vypočítány základní atributy, tak přichází na řadu analýza podřetězců. Z domény se vyextrahují 3-gramy, 4-gramy a 5-gramy. Pro získání n-gramů se prochází řetězec znak po znaku a vytváří se posloupnosti o velikosti n. N-tice jsou vytvářeny tak dlouho, dokud to umožňuje délka řetězce. Pro každou doménu získáme tři množiny n-gramů (3,4,5). U každého prvku z množiny zjistíme, zda existuje v hashovací tabulce validních a DGA n-gramů. Při nálezu se zvyšuje počitadlo příslušné třídy. Výsledný počet je vydělen celkovým počtem n-gramů řetězce, a tím je získáno poměrné zastoupení validních a DGA n-gramů. Máme tak šest atributů. Tři pro validní n-gramy a tři pro DGA n-gramy 4.2. Zprůměrováním atributů spadající pod stejnou třídu a následným provedením jejich rozdílu získáme povědomí o tom, které třídě se více doména podobá.

Nyní je na řadě ověřit kolik anglických slov se v doméně objevuje. Způsob algoritmu je nastíněn na obrázku 5.5. Postupně je z domény vytvářen menší a menší podřetězec, dokud nenalezneme anglické slovo nebo délka řetězce dosáhne délky 3. Pokud dosáhneme délky 3 a stále jsme nenalezli slovo, tak se doména zleva ořeže o jeden znak a proces hledaní se opakuje. Při nalezení slova se inkrementuje proměnná udržující součet délek všech nalezených slov v rámci domény. Při dokončení hledaní slov, se proměnná vydělí celkovou délkou a získáme tak poslední atribut vyjadřující poměrné zastoupení anglických slov.



Obrázek 5.5: Výpočet shody ve slovníku na základě postupného zkracování řetězce, dokud nenarazíme na podřetězec, jenž patři do slovníku. Na zbylý řetězec rekurzivně aplikujeme stejný algoritmus.

V této fázi máme dokončenou implementaci všech potřebných atributů, na základě kterých se bude predikovat výsledná třída. Vektor atributů je připraven k předaní rozhodovacímu modelu a ten může provést binární klasifikaci.

5.4 Integrace výsledného klasifikátoru

Pro úspěšnou predikci je potřeba mít kvalitní datovou sadu, na které se bude trénovat prediktivní model. V návrhu 4.2 už bylo provedeno prvotní trénování nad základní datovou sadou. Pro finální získání rozhodovacího modelu je použita rozmanitější sada, která více reflektuje reálný provoz. Datová sada je předem oanotovaná a obsahuje doménová jména, rozdělená podle toho, jestli jsou legitimní nebo DGA. První sloupec dat je samotná

adresa a druhý sloupec obsahuje, do které třídy adresa spadá. Nula reprezentuje validní adresu a jednička DGA adresu. Soubor s těmito daty je uložen jako CSV formát a obsahuje 600 000 domén. Před samotným trénováním modelu je potřeba dopočítat atributy. Atributy se budou počítat už v rámci implementovaného NEMEA modulu. Pro úspěšné načtení dat do modulu je potřeba provést konverzi do UniRec formátu. Na tuto operaci můžeme využít NEMEA modul s názvem `logreplay`. Syntaxe pro převedení do formátu UniRec pomocí `logreplay`:

```
logreplay -i f:trainunirecord -f ./dataset.csv
```

kde **trainunirecord** je výstupní UniRec soubor a **dataset.csv** je vstupní datová sada.

Po zpracování datové sady modulem získáme CSV soubor, který už bude obsahovat doménová jména, třídu a vektor atributů. V tabulce 5.3 můžeme vidět výslednou strukturu souboru s trénovací datovou sadou, kterou použijeme na vytvoření výsledného klasifikátoru. Z důvodu velikosti tabulky jsou prohozeny řádky se sloupci.

Tabulka 5.3: Struktura finální datové sady

Doména	facebook.com	nvjwoofansjbh.com
Třida	validní	DGA
Slovník	1.000	0.538
Průměr validních n-gramů	1.000	0.614
Validní 3-gramy	1.000	0.909
Validní 4-gramy	1.000	0.600
Validní 5-gramy	1.000	0.333
Entropie	2.750	3.239
Číslice	0.000	0.000
Souhlásky	0.500	0.769
Nealfanumerické znaky	0.000	0.000
Délka	8.000	13.000
Veřejné subdomény	1.000	1.000
Zanoření subdomén	1.000	1.000
WWW	0.000	0.000
Průměr DGA n-gramů	0.917	0.778
DGA 3-gramy	1.000	1.000
DGA 4-gramy	1.000	1.000
DGA 5-gramy	0.750	0.333
DGA podobnost	-0.083	0.164

Jakmile máme vytvořenou trénovací datovou sadu, tak můžeme přejít k samotnému získání rozhodovacího modelu. Z návrhu nejlépe vycházel při porovnání klasifikačních modelů rozhodovací strom díky své jednoduchosti a rychlosti. Při implementaci modelu chceme, aby získání klasifikátoru byl automatizovaný proces a umožnil nám tak rychlé přetrénování modelu rychlý import do NEMEA modulu. Bylo by neefektivní, kdybychom při každém přetrénování výsledného modelu museli provádět implementaci manuálně.

Implementace a transpilace do jazyka C

Z důvodu automatizace byl vytvořen skript v jazyce Python, který splní námi požadované podmínky. Skript umožňuje nahrát oanotovanou datovou sadu, aplikovat nad ní algoritmus strojového učení a exportovat výsledný klasifikátor v jazyce C.

Knihovna v rámci Pythonu, která implementuje funkcionalitu pro strojové učení se nazývá `sklearn`[2]. Datová sada je nahrána pomocí knihovny `pandas`[3], která je uzpůsobena pro práci s daty a jejich analýzu. `Pandas` nahraje CSV soubor s datovou sadou jako dataframe formát, což je jinými slovy dvourozměrné pole reprezentující tabulku. Jeho součástí je i záhlaví tabulky. Poté je potřeba vyextrahovat sloupec obsahující hodnoty, které chceme predikovat (třídu adresy). Třídy adres převedeme do číselné podoby (0 - validní, 1 - DGA). Hodnoty všech zbylých atributů musí být převedeny do dvourozměrného pole, pro jehož inicializaci využijeme knihovnu `numpy`[4], která zaručí rychlé zpracování dat. Nyní máme připravené dvě množiny dat (data, která chceme predikovat a data podle, kterých budeme predikovat). Před samotným natrénováním modelu je vhodné rozdělit datovou sadu na čistě trénovací a na testovací sadu, podle které bude ověřena kvalita klasifikátoru. Rozdělení datové sady proběhlo v poměru 70 % pro trénování a zbytek na testování.

Na závěr už stačí zvolit, že chceme natrénovat rozhodovací strom a knihovna `sklearn` se o zbytek postará. Výsledný model rozhodovacího stromu je implementován v jazyce Python, a proto se ještě musí provést transpilace do jazyka C, jenž běží na cílové platformě NEMEA. Pro převedení z jednoho jazyka do druhého bude použit již zmiňovaný `sklearn-porter` 4.3. Porter vyexportuje model jako zdrojový kód v jazyce C. Aby mohl být vyexportovaný model integrovaný s NEMEA modulem, je potřeba odstranit vyexportovanou funkci `main()`. K dispozici máme potom výsledný rozhodovací strom v podobě C funkce `predict()`, jejíž část je zobrazena zde 5.3. Funkce je uložena do souboru s názvem `estimator.c`, jenž je součástí překladu NEMEA modulu. Funkci můžeme libovolně volat z modulu a jako argument ji předáme vektor atributů. Funkce navrací třídu, do které daná adresa patří. Automatizace implementování rozhodovacího stromu nám umožňuje snadno a rychle přetrénovat daný model a umožnit tak přesnější klasifikaci.

```
int predict(float features[10]) {

    int classes[2];

    if (features[1] <= 0.5435609817504883) {
        if (features[5] <= 11.5) {
            classes[0] = 0;
            classes[1] = 2277;
        } else {
            if (features[1] <= 0.5370879769325256) {
                classes[0] = 49989;
                classes[1] = 0;
            } else {
                if (features[3] <= 0.07692299783229828) {
                    classes[0] = 399;
                    classes[1] = 0;
                } else {
                    classes[0] = 0;
                    classes[1] = 1;
                }
            }
        }
    }
```

Výpis 5.3: Podoba rozhodovacího stromu po transpilování do jazyka C.

[2]https://scikit-learn.org/stable/
[3]https://pandas.pydata.org/
[4]https://numpy.org/

Výsledky na testovací datové sadě

V rámci implementovaného skriptu proběhla ještě kromě samotného vytvoření rozhodovacího stromu také testovací fáze nad druhou částí datové sady. Nad testovací sadou byl zavolán vytvořený klasifikátor, aby provedl predikci. Po dokončení klasifikace byly opět použity funkce z knihovny `sklearn` pro ohodnocení přesnosti výsledků.

Výsledky testování budou níže ilustrovány prostřednictvím tabulek. Velikost testovací sady byla 125 601 záznamů. Celková přesnost rozhodovacího stromu dosahuje 98.9 %. V rámci testování byla také provedena křížová validace, která po pěti iteracích dosahovala úspěšnosti také 98.9 % se skoro nulovou odchylkou. Confusion matrix je zobrazena v tabulce 5.4. Sloupce vyjadřují počet adres, které byly predikovány do příslušné třídy. Řádky zase ukazují opravdové zastoupení jednotlivých tříd. Z tabulky můžeme vidět, že počet falešně pozitivních adres dosahuje 540. Číslo je nižší oproti falešně negativním, což je pozitivní vlastnost klasifikátoru. Důraz je kladen, aby co nejméně validních adres bylo klasifikováno jako DGA. Nechceme totiž, aby byly hlášeny validní doménová jména.

Tabulka 5.4: Confusion matrix rozhodovacího stromu

	Pred. validní	Pred DGA
Skutečně validní	71 700	540
Skutečně DGA	795	52 566

Z tabulky výše můžeme vypočítat další ukazatele určující kvalitu klasifikátoru. Mezi ně patří úspěšnost, přesnost, senzitivita a F1 skóre. Úspěšnost je základním ukazatelem vyjadřující podíl mezi správně klasifikovanými záznamy a celkovým počtem záznamů. Přesnost udává poměr mezi správně predikovanými záznamy a všemi záznamy predikované do konkrétní třídy. Senzitivita je podílem mezi správně predikovanými záznamy a všemi záznamy, jenž jsou doopravdy prvky predikované třídy. F1-score je potom váženým průměrem přesnosti a senzitivity. Všechny tyto metriky byli exportovány do tabulky 5.5. V tabulce jsou navíc uvedeny četnosti tříd, celkový počet testovaných záznamů a průměry jednotlivých metrik.

Tabulka 5.5: Výsledky klasifikátoru na testovací sadě

	Přesnost	Senzitivita	F1-score	Četnost
Validní	0.989	0.993	0.991	72 240
DGA	0.990	0.985	0.987	53 361
Úspěšnost			0.989	125 601
Průměr	0.989	0.989	0.989	125 601
Vážený průměr	0.989	0.989	0.989	125 601

5.5 Popis experimentů

Následující sekce bude věnována retrospektivnímu pohledu na průběh vývoje modulu. Doposud byla popisována finální podoba a aktuální stav modulu ve smyslu nejlépe dosažených výsledků. Největší změny, které byly během implementace prováděny se týkaly způsobu vý-

počtu atributů. Byl určen atribut v rámci návrhu, který se implementoval a následně se provedlo natrénování klasifikátoru pro ověření kvality atributu. Pokud daný atribut splňoval očekávání, byl zahrnut do finálního návrhu. Výslednému atributu samozřejmě předcházelo několik způsobu implementace a byla zvolena ta nejvhodnější varianta. Budou popsány jednotlivé atributy v souvislosti s jejich experimentováním na vliv kvality modelu. Ovšem ne všechny atributy vyžadovaly nějaký postupný vývoj z důvodu jejich triviální implementace. V rámci experimentování proběhlo spoustu dalších menších pokusů, které vedly ke zkvalitnění či ke zhoršení výsledků. Důležité je, že v průběhu implementace bylo postupným experimentováním dosahováno lepších výsledků. Ověřování, zda došlo ke zlepšení výsledků, probíhalo formou aplikování nejvhodnějšího algoritmu strojového učení a následného sledování ukazatelů kvality aplikovaného modelu. Důležitým ukazatelem byla celková úspěšnost a počet falešně pozitivních záznamů (důležité je propustit většinu legitimních adres). Pro určení vhodného klasifikátoru bylo provedeno porovnání jednotlivých modelů nad finální trénovací sadou. Výsledky porovnání budou uvedeny níže formou grafů. Nezlepšoval se jen výpočet jednotlivých atributů, ale celková struktura modulu. Jednou z věcí, která výrazně ovlivnila rychlost klasifikace, byla implementace hashovací tabulky pro efektivní uložení dat. Nyní budou uvedeny nejvýznamnější milníky, které měly největší dopad na přesnost klasifikace.

Výsledky experimentů

Základní atributy jako je délka, poměr souhlásek nebo poměr číslic nepotřebují z principu žádné experimenty pro zlepšení výpočtu. Entropie sama o sobě taky nemá zásadní problém s výpočtem, jde jen o dosazení do vztahu. Ale už můžeme zkoumat jaký typ Entropie je vhodnější a má lepší korelaci s klasifikovanou třídou. Byla srovnána korelace mezi entropií vypočítanou pomocí vztahu 4.1 a metrickou entropií (entropie vydělená délkou). Výsledek srovnání je vidět v tabulce 5.6, kde vidíme, že obyčejná entropie kladně koreluje s DGA třídou a naopak metrická entropie záporně koreluje s DGA třídou. Na základě experimentů bylo usouzeno, že se pro výslednou klasifikaci se použijí oba dva atributy.

Tabulka 5.6: Experimenty s entropií

	Korelace s DGA třídou	Úspěšnost s referenční sadou
Metrická entropie	-0.27	98.86
Entropie	0.38	98.84
Oba dva atributy	-	98.91

Další iterativní přístup si vyžádal výpočet shody ve slovníku. Z počátku se atribut počítal za pomocí získání množiny všech existujících podřetězců domény. Následně se nad každým podřetězcem provedlo dotázání nad slovníkem, zda je podřetězec přítomen a na závěr byl počet nálezů vydělený délkou domény. Tento přístup není nejefektivnější. Postupně se výpočet dopracoval ke způsobu, který je uveden v návrhu 4.1.2. V tabulce 5.7 můžeme vidět kladné korelace s validní třídou. Korelace se zlepšila o 10 % ve finálním způsobu výpočtu a zároveň se díky tomu zmenšil počet falešně pozitivních (validní adresy klasifikované jako DGA) o 11 %, což je příznivé zlepšení.

Tabulka 5.7: Experimenty se slovníkem

	Korelace s validní třídou
Všechny podřetězce	0.38
Hledaní nejdelšího řetězce	0.48

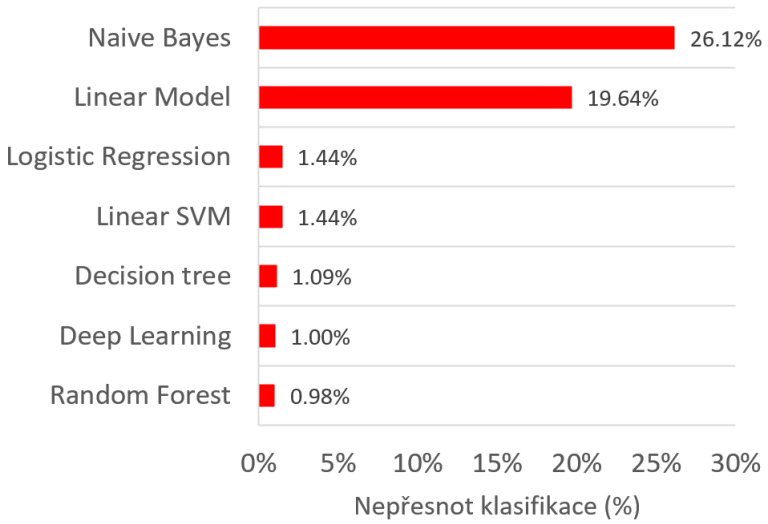
Další velké zlepšení, které přispělo k lepší klasifikaci, je zavedení výpočtu zastoupení DGA n-gramů a celkové podobnosti s DGA třídou. Zmíněné atributy vedly ke snížení počtu falešně pozitivních adres o 57 %. Na druhé straně máme atributy vyjadřující zase podobnost s validními adresami. Sada atributů kladně korelující s validní třídou zredukovala falešně pozitivní případy o 49 %.

Atributy spojené s analýzou subdomén začaly mít pozitivní dopad na klasifikaci při používaní trénovací sady, která více odráží reálný provoz a zahrnuje data s více subdoménami. Přidání atributu s počtem domén a veřejně známých subdomén opět snížilo falešně pozitivní o 37 %.

Srovnání rozhodovacích modelů

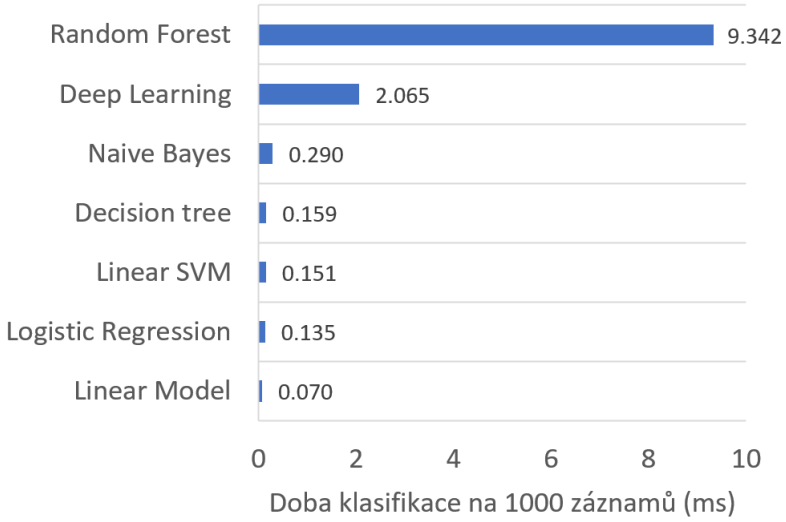
V průběhu vývoje bylo průběžně kontrolováno, zda se aplikuje vhodný rozhodovací model. Jak už bylo uvedeno v návrhu, tak úspěšnost modelu se odvíjí od dodané trénovací datové sady. V průběhu vývoje docházelo k různým výsledkům aplikovaných algoritmů z důvodu postupně vyvíjející se trénovací datové sady. Budou následovat grafy vyjadřující porovnání různých technik strojového učení v kontextu finální datové sady.

Graf na obrázku 5.6 zobrazuje procentuální nepřesnost klasifikace, což je doplněk úspěšnosti. Nejlépe si vedly tři modely. Mezi nimi je les stromů, neuronová síť a rozhodovací strom. Úspěšnost modelů se pohybovala okolo 99 %.



Obrázek 5.6: Finální porovnání rozhodovacích modelů na základě procentuální nepřesnosti.

Dalším rozhodovacím faktorem při srovnání modelů je opět rychlost klasifikace, kterou můžeme vidět na obrázku 5.7. Rychlost hraje důležitou roli při klasifikování reálného provozu, kdy se zpracovává velké množství dat. Nejrychlejší modely jsou ty, které dosahovaly nižší úspěšnosti.



Obrázek 5.7: Finální porovnání rozhodovacích modelů na základě doby klasifikace pro 1 000 domén.

Nejvhodnější model musí splňovat nároky na rychlost a zároveň musí dosahovat vyšší úspěšnosti. Nejlepším kompromisem mezi rychlostí a úspěšností je rozhodovací strom. Dobře si také vedla neuronová síť, ale kvůli své složitosti je netriviální provést transpilaci do jazyka C, a proto byl zvolen rozhodovací strom. Rozhodovací strom dosahoval úspěšnosti 98.91 % a rychlost jeho klasifikace byla 0.16 ms na 1 000 záznamů. Po výsledném porovnání můžeme říct, že byl implementován správný model pro detekování DGA adres.

Kapitola 6

Testování a vyhodnocení

6.1 Klasifikace reálných dat

V této fázi máme už naprogramovanou celkovou funkcionalitu DGA detektoru. Je na řadě ověřit, zda je výsledný model použitelný v praxi. Vstupem do detektoru bude množina datových sad, která byla vytvořena přímo pro otestování klasifikátoru. Proces vytváření datových sad je založen na postupném dělení velké datové sady (zhruba 3 488 000 adres) na menší datové sady (cca 109 000 adres). Nad datovou sadou bylo vždy provedeno náhodné promíchání adres a následně byla sada rozdělena na dvě menší se stejným zastoupením validních a DGA adres. Celkem bylo vytvořeno 32 testovacích sad. Data jsou opět anotována, aby se mohla ověřit přesnost klasifikace modulu. Testování probíhalo iterativním přístupem, kdy se v každém cyklu na vstup detektoru vložila jedna testovací sada a proběhla její klasifikace. Výsledkem klasifikace byla úspěšnost klasifikátoru nad danou sadou. V tabulce 6.1 můžeme vidět výsledky zmíněného testování. Průměrná úspěšnost klasifikátoru byla 99.22 %. Použitý klasifikátor pro testování byl natrénován ze stejné množiny domén, jako byly vytvořeny testovací sady. Z tohoto důvodu můžeme vidět o trošku vyšší úspěšnost, než při implementaci modulu.

Tabulka 6.1: Výčet úspěšností klasifikátoru nad testovacími sadami

Datová sada	Úspěšnost (%)	Datová sada	Úspěšnost (%)
Dataset0.csv	99.20	Dataset16.csv	99.26
Dataset1.csv	99.25	Dataset17.csv	99.18
Dataset2.csv	99.26	Dataset18.csv	99.18
Dataset3.csv	99.23	Dataset19.csv	99.21
Dataset4.csv	99.24	Dataset20.csv	99.21
Dataset5.csv	99.22	Dataset21.csv	99.20
Dataset6.csv	99.18	Dataset22.csv	99.22
Dataset7.csv	99.19	Dataset23.csv	99.23
Dataset8.csv	99.22	Dataset24.csv	99.20
Dataset9.csv	99.23	Dataset25.csv	99.16
Dataset10.csv	99.23	Dataset26.csv	99.21
Dataset11.csv	99.27	Dataset27.csv	99.21
Dataset12.csv	99.19	Dataset28.csv	99.24
Dataset13.csv	99.24	Dataset29.csv	99.24
Dataset14.csv	99.24	Dataset30.csv	99.20
Dataset15.csv	99.24	Dataset31.csv	99.26
Průměrná úspěšnost			99.22000
Rozptyl			0.00075

Dalším krokem bylo spuštění detektoru nad daty, které byly zachyceny z reálného provozu na síti v rámci IPFIX kolektoru. Tyto data už nejsou anotovaná a nemůžeme u nich spočítat úspěšnost klasifikátoru. Můžeme se podívat, jaké adresy byly v reálném provozu vyhodnoceny jako DGA. Testování probíhalo na dvou reálných sadách o velikosti 850 tisíc a 3 miliony. U reálných dat už můžeme využít doplňující data, která byla zachycena při sběru dat na síťové sondě. Konkrétně nás bude zajímat, zda je zachycená adresa existující. Při navázaní DGA komunikace je totiž zaregistrována pouze jedna adresa. Zbylé adresy, na které se nakažený počítač dotazuje, jsou neexistující domény. Abychom odhalili nakažený počítač, tak se stačí zaměřit právě na neexistující domény. Neexistující doménu můžeme poznat pomocí záznamu DNS_FLAGS, který je uveden v tabulce 5.1 a je součástí vstupu detektoru. Po rozparsování záznamu získáme návratovou hodnotu, kterou vrátil DNS server při odpovědi. Pokud je hodnota 3, tak se jedná o NX (non existent) doménu. V tabulce 6.2 je zobrazeno, kolik z celkového počtu domén zabírají NX domény a následně kolik adres bylo mezi NX domény klasifikováno jako DGA.

Tabulka 6.2: Zastoupení neexistujících a DGA domén v reálných datech

Počet záznamů	Počet NX domén	Počet DGA domén
3 000 000	129 255	880
851 511	76 170	464

Po provedení klasifikace nad reálnými daty se manuálně prohlédly adresy, které byly klasifikovány jako DGA. Snahou bylo najít domény, který byly vyhodnoceny jako falešně pozitivní (validní doména označena jako DGA). Nejčastěji mezi tyto adresy patřily české zkratky (qda.cvut.cz, dhi.muni.cz), dále se objevovaly reverzní dotazy na překlad z IP adresy na doménové jméno. Často byly zachyceny náhodné řetězce bez TLD domény. Tyto data se jeví jako překlepy bez hlubšího významu.

V momentě, kdy jsme zanalyzovali výslednou klasifikaci, tak jsme přešli k iterativnímu přetrénování. V každé iteraci vezmeme manuálně vyfiltrované falešně pozitivní jména a vložíme je do trénovací datové sady. Účelem iterativního přetrénování je přeučení klasifikátoru na datech, které neumí správně klasifikovat. V rámci přetrénování bylo provedeno 15 iterací z důvodu dodání dostatečného množství dat pro adaptaci modelu na neznámá data. Výsledkem iterativního přetrénování bylo snížení výskytu falešně pozitivních adres. V tabulce 6.3 je znázorněn úbytek adres označených jako DGA. Důvodem je správná predikce původně falešně pozitivních nálezů. Samozřejmě DGA data i nadále obsahují falešně pozitivní nálezy, ale díky iterativnímu přeučení je jejich počet značně minimalizován.

Tabulka 6.3: Redukce falešně pozitivních adres

Původní počet DGA adres	1 344
Počet DGA adres po přetrénování	1 022

6.2 Porovnání s referenčními detektory

Následující sekce se bude snažit porovnat vyvíjený detektor s již existujícími řešeními. Porovnání bude opět probíhat na dříve zmíněných referenčních datových sadách. Bude

porovnána časová náročnost a výsledná úspěšnost klasifikátoru. Na úvod budou popsány použité referenční klasifikátory a následně budou uvedeny výsledky jejich porovnání. Sekce bude na závěr doplněna analýzou výsledného klasifikátoru na základě závislosti úspěšnosti a doby klasifikace na výšce rozhodovacího stromu. Nyní můžeme přejít k popisu referenčních modelů.

Sappan model je klasifikátor implementován pomocí neuronové sítě v prostředí Pythonu. Model by vyvinut v rámci RWTH Aachen University. Klasifikátor je natrénovaný na stejné datové sadě jako vyvíjený model.

R model volně dostupný model[1], který implementuje rozhodovací model pomocí Random forrest algoritmu. Model je implementovaný ve vysokoúrovňovém jazyce R. Model je natrénovaný na neznámé datové sadě.

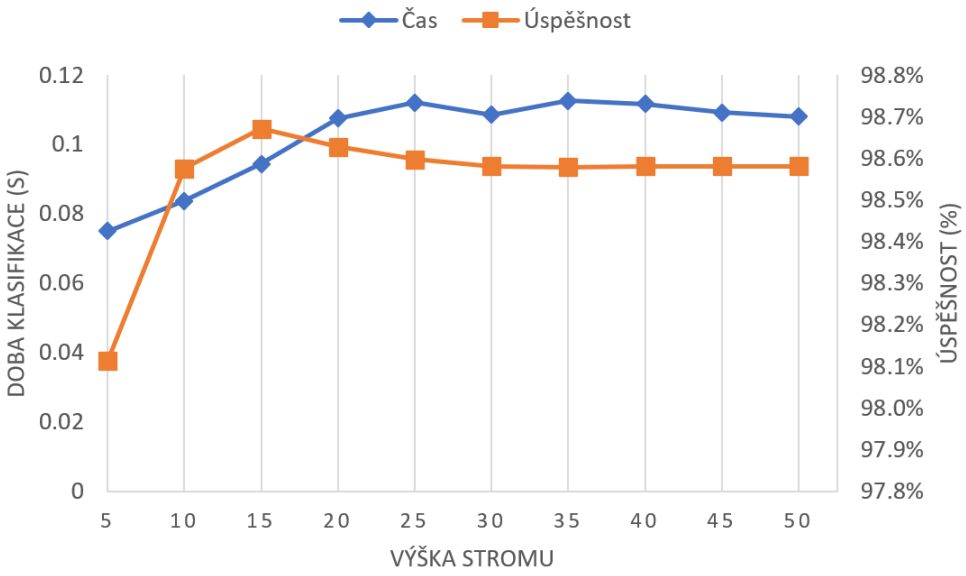
Srovnání klasifikátorů probíhalo na stejných testovacích sadách. Nad každou sadou se zavolal konkrétní klasifikátor a následně byla extrahována základní metrika úspěšnosti a jeho doba klasifikace. Po oklasifikování všech dostupných testovací sad se provedlo zprůměrování hodnot, který byly exportovány do tabulky 6.4. Podle tabulky má nejvyšší průměrnou úspěšnost 99.34 % Sappan model. Jeho doba klasifikace je 35.34 s. V porovnání s NEMEA modelem je jeho úspěšnost vyšší jen o 12 setin. I přesto, že se jedná o neuronovou síť, je rozdíl v úspěšnosti minimální. NEMEA model má samozřejmě navrch v rychlosti klasifikace. I s inicializací datových struktur má zhruba 4x menší dobu klasifikace nad stejným počtem dat. Hlavním důvodem rychlejší klasifikace je cílová platforma v jazyce C oproti Sappan modelu, jenž používá jazyk Python. Nejhůře v porovnání dopadl R model, který zaostává jak v úspěšnosti, tak v době klasifikace. Jeho průměrná klasifikace dosahuje 78.2 % a 100 000 záznamů mu trvá vyhodnotit 1 825.02 s. Navíc některé adresy (číselná subdoména, neznámé TLD, ...) není model schopen klasifikovat a skončí chybou. Poměr neklasifikovatelných adres tvořilo zhruba 8 % v rámci testovací sady.

Tabulka 6.4: Srovnání úspěšností referenčních klasifikátorů

Klasifikátor	Úspěšnost (%)	Doba klasifikace (s)
NEMEA model	99.22	8.01
Sappan model	99.34	35.54
R model	78.20	1 825.02

Během testování bylo zjištěno, že při trénování rozhodovacího stromu nad velkým množstvím dat hraje velkou roli výška rozhodovacího stromu. Pro finální trénování klasifikátoru byla právě použita největší dostupná anotovaná sada o velikosti 3 488 000 adres. Problém výšky stromu vedl k provedení analýzy, která ukáže, jak moc výška stromu ovlivňuje výslednou úspěšnost a dobu klasifikace. V rámci trénování klasifikátoru byla postupně zvedána maximální výška stromu a byla zaznamenávaná úspěšnost a doba klasifikace. V každé iteraci klasifikace se maximální výška stromu zvedla o 5. Výsledné hodnoty vynesené do grafu můžeme vidět na obrázku 6.1. Nejvyšší dosažená úspěšnost byla s maximální délkou stromu 15. S výškou 20 se dokonce úspěšnost začala snižovat a doba klasifikace se začne více prodlužovat, což vede k neefektivitě klasifikátoru.

[1] https://github.com/jayjacobs/dga



Obrázek 6.1: Závislost úspěšnosti a doby klasifikace na výšce stromu.

Na základě výše uvedeného zjištění byl klasifikátor natrénován pro různé maximální výšky stromu v intervalu 15 až 19. V tabulce 6.5 je uvedena průměrná úspěšnost a doba klasifikace s různou výškou stromu. Úspěšnost klasifikace roste s výškou stromu do hodnoty 17, ale jen v rozsahu setin. Nejnižší doba klasifikace 138 ms je také dosažena s výškou 17. Po provedení křížové validace jsme došli k závěru, že podle uvedeného grafu a tabulky bude finální rozhodovací strom podávat nejlepší výsledky s výškou stromu 17.

Tabulka 6.5: Úspěšnost a doba klasifikace na základě výšky stromu

Výška stromu	Úspěšnost (%)	Doba klasifikace (ms)
15	98.670	220
16	98.671	151
17	98.685	138
18	98.651	161
19	98.647	169

6.3 Výkonnost modulu

V této sekci bude popsáno, kolik zdrojů modul využívá. Dále bude změřeno, kolik adres dokáže modul zpracovat za sekundu. Měření výkonnosti se bude provádět na dvou referenčních strojích. První test bude probíhat na běžném notebooku a druhý test bude proveden na stolním počítači. Při testování využití zdrojů bude použit nástroj `top`, který monitoruje systémové procesy. Rychlost modulu bude testována pomocí funkce `clock()`[2]. Jelikož modul spotřebuje určitý čas na inicializaci a finalizaci potřebných datových struktur, tak by bylo nevhodné použít nástroj `time`, který počítá celkovou dobu běhu programu od startu

[2]https://www.tutorialspoint.com/c_standard_library/c_functio_clock.htm

po jeho ukončení. Proto se bude čistý čas spotřebovaný pouze na klasifikování adres měřit přímo uvnitř modulu pomocí funkce `clock()`. Funkce navrací počet taktů procesoru od spuštění programu. Zavoláním funkce před začátkem zpracování adres a po jeho dokončení můžeme rozdílem hodnot získat celkový počet taktů spotřebovaných na klasifikaci adres. Po převedení taktů na sekundy potom dostaneme celkový čas samotné klasifikace. Stejným způsobem bude také získána doba inicializace a finalizace datových struktur. Měření bude provedeno 10x za sebou a následně se časy zprůměrují. Nyní můžeme přejít k prvnímu testu.

Hardwarová specifikace osobního notebooku:

- OS: Windows 10 (Windows subsystem for Linux-Ubuntu)
- CPU: Intel Core i5-10210U/1.6 - 2.11GHz
- RAM: 8 GB DDR4
- DISK: 512 SSD

Tabulka 6.6 zobrazuje využití zdrojů procesem `dga_detector` v nástroji `top`. Nástroj `top` zobrazuje základní údaje jako je aktuální čas, doba běhu systému a jeho zatížení. Udává celkové informace o všech běžících procesech a spotřebovaného procesorového času. Dále shrnuje informace o RAM paměti a swapovacím prostoru. Jednotlivé procesy obsahují identifikační číslo procesu, údaje o prioritě nebo celkovou paměť, kterou proces použil. Zahrnuje to zdrojový kód, data uložená v operační paměti, ale i data přesunutá na disk. Vidíme, kolik paměti RAM proces využívá nebo kolik paměti sdílí s ostatními procesy. V neposlední řadě zjistíme, jaké je procentuální vytížení CPU a množství procesorového času stráveného nad procesem. Tyto poslední dva údaje jsou méně relevantní, z důvodu průběžně měnících se hodnot. Z tabulky 6.6 můžeme vidět, že proces využil celkově 625 128 KiB paměti. Paměti RAM zabral 539 384 KiB. Po převedení jednotek to odpovídá zhruba 640.1 MB a 552.3 MB.

Tabulka 6.6: Využití zdrojů procesu dga_detector na notebooku

ID procesu	7 536
Priorita	20
Celkové využití paměti (KiB)	625 128
Využití RAM (KiB)	539 384
Využití RAM (%)	8.5
Sdílená paměť s dalšími procesy (KiB)	2 844

Při měření rychlosti na notebooku s využitím funkce `clock()` jsme dosáhli průměrného výsledku 89.90 s. Čas vyjadřuje čistou dobu na klasifikování adres. Na začátku cyklu, který postupně zpracovává adresy, byla zavolána funkce `clock()` a po ukončení cyklu byla zavolána znovu. Odečtením těchto hodnot jsme získali výsledný procesorový čas strávený v cyklu. Dále byla změřena doba potřebná pro inicializaci a finalizaci datových struktur, která činila 1.84 s. Oproti celkové době klasifikace je to jen nevýznamná část. Testování rychlosti probíhalo na datové sadě o velikosti 3 000 000 adres. Z počtu adres a průměrného času na vykonaní můžeme vypočítat, kolik je schopen modul zpracovat adres za jednu sekundu.

Počet adres zpracovaných za sekundu:

$$\frac{3\,000\,000}{89.90} = 33\,370 \;\; adres \;\; za \;\; sekundu$$

Nyní přejdeme k testování stolního počítače, který má následující hardwarové specifikace:

- OS: Windows 10 (Windows subsystem for Linux-Ubuntu)
- CPU: Intel Core i5-4440/3.1GHz
- RAM: 16 GB DDR3
- DISK: 2x1 TB HDD + 120 GB SSD

Výsledky testování stolního počítače jsou zobrazeny v tabulce 6.7. Rozdíl oproti notebooku je nepatrný a v podstatě modul vykazuje stejné paměťové nároky. Celkově bylo využito 614 576 KiB paměti. Na operační paměti bylo alokováno 539 160 KiB. Po opětovném převedení jednotek získáme hodnoty 629.3 MB a 552.1 MB.

Tabulka 6.7: Využití zdrojů procesu dga_detector na stolním počítači

ID procesu	6 441
Priorita	20
Celkové využití paměti (KiB)	614 576
Využití RAM (KiB)	539 160
Využití RAM (%)	4.1
Sdílená paměť s dalšími procesy (KiB)	2 624

Nižší výpočetní rychlost procesoru stolního počítač se odráží v rychlosti modulu. Celková doba na zpracování 3 000 000 adres činí 107.91 s. Inicializační a finalizační fáze zabere 1.92 s. U stolního počítače se nám sníží počet zpracovaných adres za sekundu.
Počet adres zpracovaných za sekundu:

$$\frac{3\,000\,000}{107.91} = 27\,801 \;\; adres \;\; za \;\; sekundu$$

V porovnání s notebookem dokáže stolní počítač za sekundu zpracovat o 5 569 adres méně.

Kapitola 7

Predikce úspěšnosti datových sad

Predikování úspěšností datových sad nám přináší hned několik uplatnění. Například při generování datových sad pomocí genetických algoritmů je velice časově náročné určování kvality vygenerovaných jedinců. Pro určení kvality jedinců nám slouží fitness funkce. V souvislosti s DGA je pro nás fitness funkce DGA detektor. Zrychlením výpočtu fitness funkce můžeme uvolnit velké množství strojového času na její výpočet. Proto chceme rychle a efektivně ohodnocovat datové sady. Predikování úspěšnosti datových sad v souvislosti s DGA není jediné uplatnění. Koncept predikce úspěšností se může aplikovat i další problémy řešené pomocí strojového učení.

Princip řešení spočívá ve využití strojového učení. Výsledkem bude natrénovaný klasifikátor, jehož výstupem bude interval vyjadřující procentuální úspěšnost. Kvůli konkrétní aplikaci na datové sady obsahující doménová jména neexistují podobná řešení, která by plnila podobný úkol. Specifický problém si vyžaduje specializované řešení. Samozřejmě znalost DGA (viz 3.1) je důležitou součástí pro pochopení problematiky. Práce, která se zabývá interpretací a mírou transparentnosti modelu je například článek CHIRPS: Explaining random forest classification [8], který je o transparentnosti random forest modelu. Článek je více teoretický a stejně jako my využívá definic při řešení problému. V rámci naší práce je také kladen důraz na zkoumání, jak daný model funguje, ale náš přístup je nepřímý. Prostřednictvím datových sad, které dodáme modelu můžeme odhalit jeho slabé stránky a díky charakteristice sady i zjistíme, jaké jsou důvody. Další článek, který řeší kvalitu modelů a jejich průhlednost je The Mythos of Model Interpretability [16]. V článku se pohybují na úrovní modelů a jejich porovnání, ovšem našim cílem je určovat kvalitu na úrovni datových sad. Poslední článek řešící podobnou problematiku, je AIMQ: a methodology for information quality assessment [15]. Jejich přístup spočívá v určení obecné kvality informací pomocí informačního zisku. Rozdělují data do 4 hlavních kategorií a popisují metodologii, jakým způsobem získat kvalitu informací. Náš přístup by měl vyústit v ohodnocení datové sady pomocí stanovených metrik.

Přístup k problému spočívá v tom, že budeme mít k dispozici velké množství datových sad, vygenerovaných pomocí genetického algoritmů. Datové sady obsahují seznam doménových jmen a příznak, zda se jedná o validní nebo DGA doménu. V první fázi přijde na řadu ohodnocení sad pomocí získaných klasifikátorů detekující DGA. Získáním úspěšností jednotlivých sad získáme množinu dat, která bude cílem predikce ohodnocujícího klasifikátoru. Následně se spočítají agregované parametry datové sady, které budou sloužit jako rozhodovací prvky při trénování klasifikátoru.

Výsledným řešením jsme přispěli k desetinásobnému zrychlení fitness funkce. Dále jsme schopni vytvořit množinu parametrů charakterizující danou datovou sadu a na základě

parametrů poté určit kvalitu datové sady. Na základě charakteristiky datové sady můžeme odhalit slabiny testovaného modelu. Pro odhalení slabin dodáme datové sady, na kterých model dosahuje nízké úspěšnosti a na základě charakteristik sad můžeme získat i příčinu nízké úspěšnosti.

Zásadním problémem je výběr vhodných atributů, díky kterým budeme moci rychle a kvalitně predikovat ohodnocení datové sady. Musí se zvážit do kolika tříd se bude klasifikovat při udržení dostačující přesnosti. Pro ilustraci si představme, že máme datovou sadu a chceme predikovat její úspěšnost. Výsledkem můžou být 4 intervaly 0-25 %, 25-50 %, 50-75 %, 75-100 %. Když získáme kvalitní atributy a budeme mít k dispozici rozmanitou množinu datových sad, jenž bude obsahovat dostatečné zastoupení všech úspěšností, tak se nám může podařit tyto intervaly ještě rozmělnit, a zpřesnit tak klasifikaci. V rámci řešení bude také vyzkoušen regresní způsob predikce, při které se budeme snažit odhadnout už přímo numerickou hodnotu představující cílovou úspěšnost. Tímto přístupem se tak vyhneme vytváření klasifikačních tříd.

7.1 Trénovací datová sada

Pro ohodnocení datové sady musíme využít strojového učení. Manuálním pozorování datové sady bychom ničeho nedosáhli. Jde nám o to, abychom určili vhodné atributy, které budou charakterizovat zkoumanou datovou sadu. Atributy musí věrohodně reprezentovat datovou sadu. Jednotlivé atributy budou tedy často výsledkem průměru nad určitým atributem, jenž popisuje jedno doménové jméno. Atributy se musí rychle počítat, aby to mělo smysl. Většinou půjde o statické atributy jako je průměrná délka doménového jména, průměrná entropie, průměrný počet numerických znaků, průměrný počet souhlásek, zastoupení validních n-gramů a anglických slov. Kromě atributů bude vstupem pro trénovací sadu množina úspěšností analyzovaného klasifikátoru. Jako referenční klasifikátor byl zvolen R model (viz 6.2). Pro účely porovnání bude použit také Sappan model.(viz 6.2). Stejný princip je aplikovatelný i na další DGA klasifikátory.

7.1.1 Architektura

K dispozici budeme mít množinu vygenerovaných datových sad. Samotná množina dat byla pro práci poskytnuta v rámci většího projektu. Vygenerované datové sady budou na vstupu mého skriptu v jazyce Python. Obecná struktura, jak bude u jednotlivých sad vypočítaná úspěšnost můžeme vidět na obrázku 7.1.

Obrázek 7.1: Princip počítání úspěšnosti datových sad. Za pomocí genetického algoritmu se vytvoří množina umělých datových sad. Sady jsou následně v evaluačním skriptu klasifikovány zmiňovanými klasifikátory.

Skript v jazyce Python projde na začátku adresář s vygenerovanými datovými sadami a do seznamu si uloží všechny cesty nalezených souborů. Následně začne iterovat přes jednotlivé cesty obsahující datové sady. V každém cyklu je předána datová sada uvedeným klasifikátorům. Datová sada je reprezentována ve formátu CSV souboru. První sloupec obsahuje doménové jméno a druhý sloupec příznak, zda je adresa validní nebo DGA. Ilustrování datové sady je možné vidět v tabulce 7.1.

Tabulka 7.1: Ukázka struktury datové sady

domain	type
9qmmvy.sk	1
rjvcvy.sk	1
mqzdvy.sk	1
iazovy.sk	1
pbghvy.sk	1
pdnjvy.sk	1

Soubor je načten do prostředí Pythonu, kde doménová jména jsou uložená v listové struktuře `domains` a příznaky v seznamu `labels`. Dvě zmíněné struktury se stávají vstupy pro klasifikátory. U R modelu je využit balíček `rpy2`, jenž vytváří v Pythonu rozhraní pro práci s jazykem R, takže můžeme snadno spouštět R model přímo v Pythonu. Úspěšnost klasifikátorů se spočítala nad každou datovou sadou (vygenerovaná pomocí genetického algoritmu) o celkovém počtu 9 600.

Úspěšnosti klasifikátorů jsou vyextrahovány do CSV souboru. Výslednou strukturu množiny úspěšností je možné vidět v tabulce 7.2. Výsledná data budou sloužit jako třídy, které budeme chtít při trénování ohodnocovacího klasifikátoru predikovat.

Tabulka 7.2: Úspěšnosti klasifikátorů nad ukázkovou datovou sadou

Datová sada	gid38_acc035.46.csv
R model	75.84 %
Sappan model	35.46 %

Poté co vypočítáme úspěšnosti klasifikátorů nad datovými sadami, tak máme vytvořenou první polovinu trénovací datové sady.

Druhá fáze zahrnuje výpočet atributů, jenž budou charakterizovat celou datovou sadu. Musíme zvolit atributy, které věrohodně reprezentují datovou sadu. Půjde o množinu statických atributů, které budou počítány pro celou datovou sadu. Budeme počítat atributy u jednotlivých doménových jmen, které následně zprůměrujeme pro celou datovou sadu. Bude využit i rozptyl daných atributů.

Hlavní roli budou hrát následující atributy (popis atributů viz 4.1).

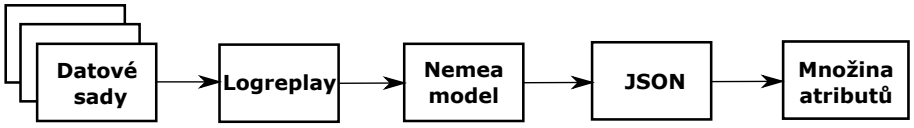
- Délka
- Zastoupení číslic
- Zastoupení Souhlásek
- Entropie
- N-gramy
- Shoda ve slovníku

Výslednou charakteristiku datové sady tvoří právě průměrné hodnoty a rozptyl výše uvedených atributů. Rozptyl může být vhodným ukazatelem z jakých dat se datová sada skládá. Proto bude vypočítán rozptyl u všech dříve zmíněných atributů, které přispějí k vytvoření kvalitativních parametrů datové sady. Rozptyl je počítán pomocí následujícího vztahu 7.1.

$$\sigma^2 = \frac{\sum x^2}{N} - \mu^2 \tag{7.1}$$

kde **x** je prvek množiny, **N** celkový počet prvků množiny a μ je průměr množiny [22].

Způsob výpočtu atributů byl zvolen na základě rychlosti. Nejrychlejším způsobem bylo využít NEMEA model. Implementace v jazyce C umožňuje rychlý výpočet atributů, i přes režii s konvertování datové sady do UniRec záznamu. Na obrázku 7.2 je možné vidět průběh výpočtu atributů. Na výstupu NEMEA modelu jsou data ve formátu JSON, který je následně zpracován skriptem `Features.py`.



Obrázek 7.2: Způsob výpočtu atributů datové sady.

Výsledné atributy jsou v rámci skriptu opět serializovány do CSV souboru. Ukázka vypočtených atributů, jenž charakterizují celou datovou sadu jsou zobrazeny v tabulce 7.3. Můžeme si všimnout, že rozptyl délky je nulový. Důvodem je stejná délka doménových jmen napříč všemi datovými sadami. Pro trénování ohodnocovacího modelu se tento atribut zatím neuplatňuje. Do budoucna se plánuje otestování na datových sadách, které obsahují domény s různou délkou.

Tabulka 7.3: Atributy charakterizující ukázkovou datovou sadu

Datová sada	gid38_acc035.46.csv
Průměrná entropie	1.647
Průměrná délka	6.000
Průměrné zastoupení validních n-gramů	0.154
Průměrné zastoupení anglických slov	0.477
Průměrné zastoupení číslic	0.103
Průměrné zastoupení souhlásek	0.667
Rozptyl entropie	0.031
Rozptyl délky	0.000
Rozptyl zastoupení validních n-gramů	0.036
Rozptyl zastoupení anglických slov	0.068
Rozptyl zastoupení číslic	0.013
Rozptyl zastoupení souhlásek	0.033

V momentě, kdy získáme vektor atributů, tak se můžeme podívat na časovou náročnost jejich výpočtu. Je důležité, aby čas na výpočet atributů zabral méně času než vyhodnocení R modelu, jinak by to postrádalo efekt. Dále je potřeba vybrat dostatečně rychlý rozhodovací model, jenž bude rozebrán v následující sekci. V tabulce 7.4 vidíme porovnání časů různých přístupů. První řádek tabulky vyjadřuje čas strávený čistě nad výpočtem atributů, v druhém řádku je celkový čas spotřebovaný celým NEMEA modulem (inicializace datových struktur), který zahrnuje i čistý čas výpočtů atributů. Poslední řádek je čas R modelu, který klasifikoval jednotlivé datové sady. Testování bylo prováděno na datových sadách s počtem 800 domén. Časové údaje jsou přepočítány na dobu pro 1 000 datových sad.

Z tabulky můžeme vidět, že celkový čas na výpočet atributů i s inicializací NEMEA modelu je 125,39 s. V porovnání s dobou klasifikace R modelu získáme atributy více něž 10x rychleji. V této fázi můžeme říct, že má smysl predikovat úspěšnost datové sady, jelikož dosahujeme až desetinásobného zrychlení.

Tabulka 7.4: Srovnání časů pro výpočet atributů

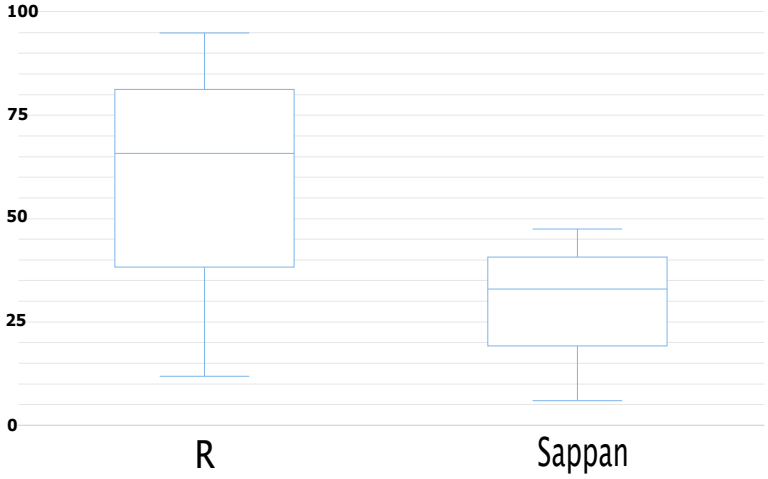
	Doba výpočtu (1 000 sad)
Čistý výpočet atributů	4.34 s
Celkový čas NEMEA modulu	125.39 s
Klasifikace R modelu	1 323.87 s

Atributy, které jsou zde uvedeny nemusí být finálním ani správným řešením. Postupně se bude pracovat na zdokonalení výběru kvalitních atributů. Čím kvalitnější získáme atributy, tím lepší a přesnější budou výsledky natrénovaného klasifikátoru, který bude datové sady hodnotit. Do budoucna se na výběru atributů bude nadále pracovat a zdokonalovat tak výsledný model.

7.2 Tvorba klasifikátoru

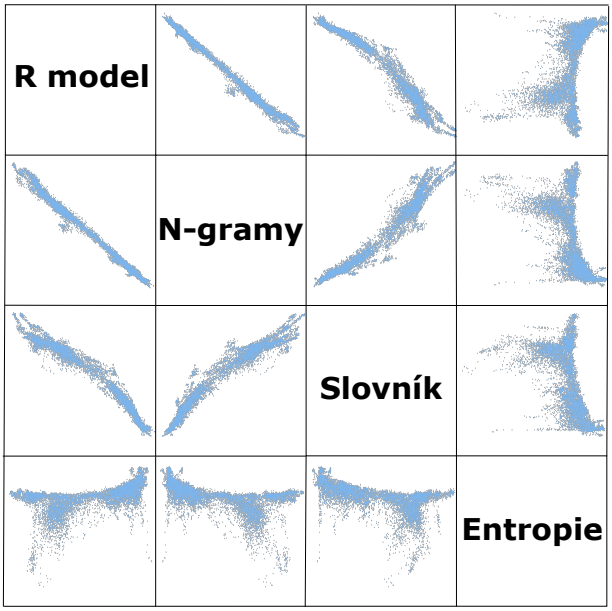
Když máme k dispozici vytvořenou trénovací datovou sadu, tak můžeme přejít ke trénování klasifikátoru a jeho následnému testování. V první fázi půjde o to vybrat vhodný rozhodovací model. Vhodným nástrojem pro prvotní srovnání rozhodovacích modelů je nástroj Rapidminer, který nám rychle a snadno umožní srovnat a vizualizovat rozhodovací modely. Jakmile zvolíme výsledný typ klasifikátoru, přijde na řadu implementace modelu v jazyce Python.

Součástí trénovací datové sady je množina úspěšností, které byly spočítány referenčními klasifikátory 7.1. Musíme zvolit úspěšnost jakého modelu chceme predikovat. Predikci nejde dělat obecně, protože každý klasifikátor vykazuje rozdílné chování. Je proto nutné pro každý model vygenerovat specializovanou datovou sadu. Vezmeme cílový model, vygenerujeme k němu dostatečný počet datových sad, které kategorizujeme a získáme tak výslednou trénovací sadu. Pro porovnání můžeme na obrázku 7.3 vidět distribuci jednotlivých úspěšností pro R a Sappan model. Z obrázku jde vidět, že datové sady vygenerované přímo pro R model mají mnohem rovnoměrnější rozložení v intervalu úspěšnosti (0-100 %). Důvodem rovnoměrného rozdělení úspěšností R modelu je využití R úspěšnosti jako fitness funkce při generování datových sad, a proto nejde vytvořit obecný model pro více klasifikátorů. Hlavním účelem generování umělých datových sad je zpřesnění klasifikátoru. Zlepšení dosáhneme dodáním kvalitní trénovací datové sady. Když budeme chtít predikovat výsledky nějakého klasifikátoru, tak je dobré mít rovnoměrné zastoupení úspěšností v trénovací datové sadě. V rámci našeho prvotního řešení rozdělíme úspěšnost do 4 tříd. Z obrázku 7.3 lze jasně určit, že mnohem vhodnější pro predikování je model R, díky jeho rovnoměrnějšímu rozdělení úspěšností nad trénovací datovou sadou.



Obrázek 7.3: Porovnání distribuce úspěšností R a Sappan modelu.

Před samotným výběrem vhodného rozhodovacího modelu se podíváme na vzájemnou korelaci některých atributů. Obrázek 7.4 vyjadřuje grafovou matici s korelacemi mezi vybranými atributy. Pro přehlednost nejsou uvedeny všechny atributy.



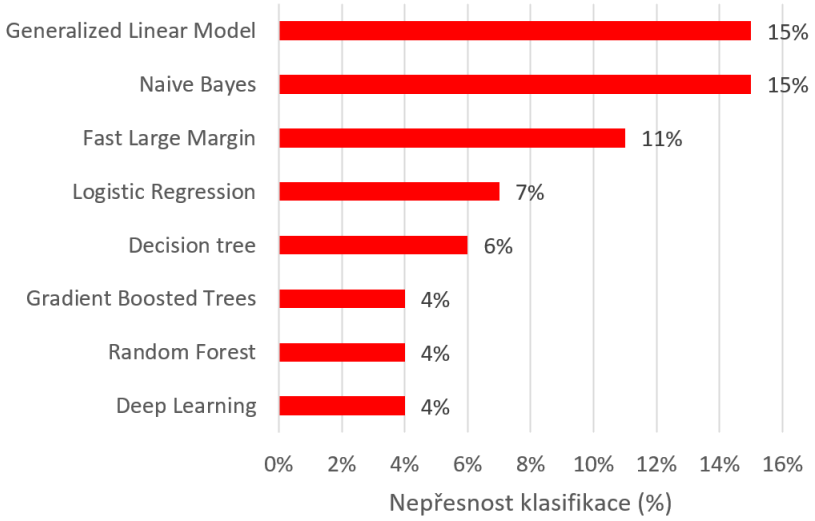
Obrázek 7.4: Grafová matice korelací mezi vybranými atributy.

Můžeme si všimnout, že úspěšnost R modelu a průměrné zastoupení n-gramů vykazují zápornou korelaci. Z toho vyplývá, že pokud je poměrné zastoupení validních n-gramů nižší, tak je pro R model snazší predikovat DGA adresu. Podobně to je u shody ve slovníku. U průměrné entropie je korelace s výslednou úspěšností horší, ale lze pozorovat lehkou tendenci ke kladné korelaci, což nám říká, že čím vyšší neuspořádanost řetězce (entropie), tak tím se zvyšuje pravděpodobnost správné predikce DGA jména.

Po prozkoumání atributů začneme nad datovou sadou aplikovat množinu algoritmů strojového učení. Jak bylo zmíněno dříve, tak pro rychlé porovnání rozhodovacích modelu je vhodné použít Rapidminer. V Rapidmineru jsme nad datovou sadou aplikovali všechny

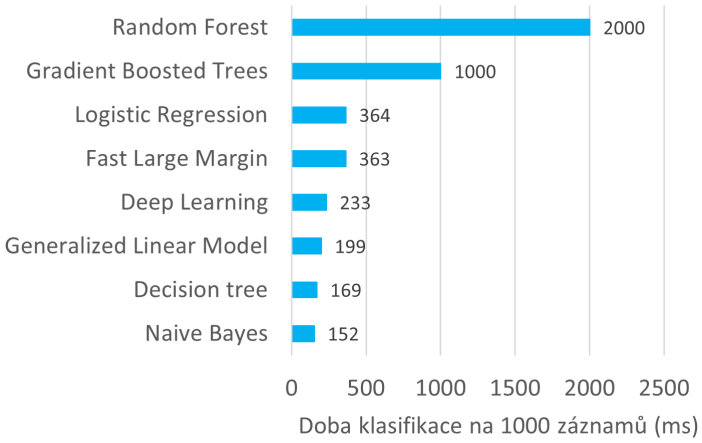
dostupné modely, které budou predikovat úspěšnost R modelu do 4 tříd. Třídy vzniknou rozdělením rozsahu úspěšností na 4 části. V rámci pozdější implementace modelu budou třídy rozděleny přesně do intervalů 0-25 %, 25-50 %, 50-75 %, 75-100 %. Po výpočtu všech modelů se zobrazilo jejich porovnání.

Srovnání bylo vyjádřeno dvěma grafy. První graf 7.5 vykresloval procentuální nepřesnost srovnávaných modelů a druhý graf 7.6 jejich časovou náročnost klasifikace. Srovnání bylo prováděno na datové sadě s velikostí 4 000 záznamů. Výsledný model by měl být hlavně rychlý při klasifikování, jinak by poztrácelo celé řešení smysl, pokud by ohodnocovací klasifikátor byl pomalejší než původní DGA detektor.



Obrázek 7.5: Procentuální nepřesnost srovnávaných modelů.

Z grafu 7.5 vyplývá, že mezi nejpřesnější modely patří Gadient Boosted Trees, Random Forest a Deep Learning. Nepřesnost modelů se pohybuje kolem 4 % a hned za nimi je Rozhodovací strom s nepřesností 6 %.



Obrázek 7.6: Časová náročnost klasifikace srovnávaných modelů.

Pokud zvážíme znalosti z dalšího grafu 7.6 s časovou náročností klasifikace, tak zjistíme, že nejvhodnější model je právě rozhodovací strom. Rozhodovací strom dokáže klasifikovat 1 000 záznamů za 169 ms. Další přesné modely zaostávají právě v rychlosti klasifikace, což je pro nás rozhodující faktor. Deep Learning je jediný srovnatelný s rozhodovacím stromem, ale jeho nevýhodou je složitost. Když dáme dohromady čas výpočtu atributů 7.4 a dobu klasifikace rozhodovacího stromu, tak jsme schopni predikovat úspěšnost 1 000 datových sad za 125.55 s. Tímto přístupem získáme výslednou úspěšnost až desetkrát rychleji, ale s menší odchylkou od reálné úspěšnosti. Další část textu bude obsahovat právě implementaci rozhodovacího stromu v prostředí Python.

Prvním krokem v rámci implementace bylo zvolení knihovny, která umožňuje implementovat rozhodovací strom. Jednou z mnoha knihoven pro strojové učení je `scikit-learn`[1], která bude při implementaci využita. Dále přišlo na řadu načtení a zpracování trénovací sady. Pro práci s daty byly použity knihovny `pandas`[2] a `numpy`[3], které urychlily zpracování dat. Před samotným aplikováním klasifikátoru bylo potřeba rozdělit sadu na data, která chceme predikovat (úspěšnost R modelu) a atributy, podle kterých se klasifikátor bude rozhodovat. Predikovaná data se musely podrobit reklasifikaci a rozdělit úspěšnosti do 4 tříd. Reklasifikace je naznačena níže.

- Třída 1 — 0-25 %
- Třída 2 — 25-50 %
- Třída 3 — 50-75 %
- Třída 4 — 75-100 %

Jakmile máme vytvořené pole s predikovanými třídami a vícerozměrné pole s atributy, tak můžeme přejít k samotnému trénování a testování rozhodovacího stromu. Pro otestování kvality rozhodovacího stromu byly použity dva způsoby validace. První způsob spočívá v běžném rozdělení datové sady na trénovací a testovací sadu v poměru 2:1. Druhý způsob je založený na křížové validaci, kdy se v 5 iteracích vytvoří vždy nová trénovací a testovací datová sada. Výsledkem křížové validace je směrodatnější úspěšnost výsledného modelu díky zprůměrování více individuálních trénovacích procesů.

Nyní si podrobněji rozebereme výsledky prvního způsobu rozdělení datové sady. Na rozdělení byla použita funkce `train_test_split()`, která stratifikovala datovou sadu podle predikovaných tříd. Jinými slovy při rozdělení zajistila rovnoměrné zastoupení tříd. Takže po rozdělení máme dvě třetiny původních dat určené k trénování (6 720) a zbylou třetinu pro testování (2 880). V tabulce 7.5 máme zobrazenou matici vyjadřující výsledky klasifikace. Řádky vyjadřují, jak daný klasifikátor predikoval příslušnost záznamů k jednotlivým třídám. A na druhou stranu sloupce zastupují opravdovou příslušnost záznamů ke třídám. Tabulka přehledně ukazuje, kolik záznamů bylo správně klasifikováno. Můžeme si všimnout, že při chybné klasifikaci vždy došlo jen k zařazení do vedlejší kategorie úspěšnosti. Je to pro nás pozitivní znamení a můžeme říct, že model funguje podle očekávání a nepredikuje diametrálně jinou kategorii.

Z tabulky 7.5 můžeme vypočítat další ukazatele [28] určující kvalitu klasifikátoru. Mezi ně patří úspěšnost, přesnost, senzitivita a F1 skóre. Všechny tyto metriky byly exporto-

[1] https://scikit-learn.org/stable/
[2] https://pandas.pydata.org/
[3] https://numpy.org/

vány do tabulky 7.6. V tabulce jsou navíc uvedeny četnosti tříd, celkový počet testovaných záznamů a průměry jednotlivých metrik. Můžeme vidět, že úspěšnost modelu je 96 %.

Tabulka 7.5: Confusion matrix

	Skutečnost			
	1	2	3	4
Pred. 1	181	32	0	0
Pred. 2	17	951	14	0
Pred. 3	0	17	516	22
Pred. 4	0	0	20	1 110

Tabulka 7.6: Výsledky klasifikátoru na testovací sadě

	Přesnost	Senzitivita	F1-score	Četnost
Třída 1	0.85	0.91	0.88	198
Třída 2	0.97	0.95	0.96	1 000
Třída 3	0.93	0.94	0.93	550
Třída 4	0.98	0.98	0.98	1 132
Úspěšnost			0.96	2 880
Průměr	0.93	0.95	0.94	2 880
Vážený průměr	0.96	0.96	0.96	2 880

Druhý způsob testování klasifikátoru spočíval v křížové validaci. Jak už bylo zmíněno dříve, tak křížová validace v každém kroku vytvoří novou trénovací a testovací datovou sadu a provede natrénování a otestování modelu. Bylo zvoleno 5 kroků. V každém kroku se vzala náhodně jedna pětina celkové datové sady a zbytek dat byl označen jako trénovací sada. Zmíněným způsobem ověříme, zda úspěšnost prvního způsobu testování nebyla jenom náhoda. V tabulce 7.7 jsou uvedeny úspěšnosti jednotlivých iterací. Můžeme vidět, že celková úspěšnost po křížové validaci je 94 %.

Tabulka 7.7: Výsledky křížové validace

	Úspěšnost
Iterace 1	0.938
Iterace 2	0.941
Iterace 3	0.936
Iterace 4	0.957
Iterace 5	0.948
Průměr	0.94
Směrodatná odchylka	0.01

V průběhu práce byly dodány nové datové sady s rovnoměrnějším zastoupení do kategorií, a proto můžeme zkusit rozmělnit predikované intervaly. Experimenty budou prováděny nad kategoriemi o velikosti 5 a 10. Výsledné třídy vzniknou vždy rozdělením intervalu 100 požadovaným číslem. Proces implementace modelů je totožný s předchozím případem se čtyřmi třídami. V tabulkách 7.8 a 7.9 jsou zobrazeny confusion matice pro testované intervaly.

Tabulka 7.8: Confusion matrix s 5 třídami

	Skutečnost				
	1	2	3	4	5
Pred. 1	676	81	45	44	0
Pred. 2	61	683	89	37	0
Pred. 3	38	53	614	90	0
Pred. 4	17	33	71	702	13
Pred. 5	0	0	0	16	848

Tabulka 7.9: Confusion matrix s 10 třídami

	Skutečnost									
	1	2	3	4	5	6	7	8	9	10
Pred. 1	137	7	4	2	11	5	7	0	0	0
Pred. 2	11	123	11	7	4	4	12	0	0	0
Pred. 3	10	4	137	19	1	2	7	0	0	0
Pred. 4	7	9	12	133	22	3	3	4	0	0
Pred. 5	9	6	4	14	110	15	8	5	0	0
Pred. 6	2	5	11	4	10	124	29	13	0	0
Pred. 7	10	9	1	4	10	20	75	27	0	0
Pred. 8	0	0	0	3	2	5	39	141	27	1
Pred. 9	0	0	0	0	0	0	0	25	141	10
Pred. 10	0	0	0	0	0	0	0	1	17	173

V momentě, kdy se snažíme intervaly rozmělnit, tak klesá i výsledná přesnost klasifikátoru. Z uvedených matic jde vidět, že většina predikcí se stále drží ve správné kategorii, ale ovšem chybné predikce už zasahují nejen do vedlejších tříd. Také si můžeme všimnout kvalitnější klasifikace datových sad s vysokou úspěšností. Metriky získané z natrénovaných modelů nad testovacími daty jsou opět vygenerované do tabulek 7.10 a 7.11. Datová sada pro 5 kategorií obsahuje 14 035 záznamů a sada pro 10 kategorií 6 091 záznamů. Testovací sady zastupovali 30 % z celkového počtu. Model pro 5 tříd dosahuje po křížové validaci úspěšnosti 83 %. Model pro 10 tříd dosáhl křížovou validací průměrné úspěšnosti 72 %. Můžeme sledovat klesající přesnost s narůstajícím počtem kategorií.

Tabulka 7.10: Výsledky klasifikátoru pro 5 tříd

	Přesnost	Senzitivita	F1-score	Četnost
0-20	0.80	0.85	0.83	792
20-40	0.79	0.80	0.79	850
40-60	0.77	0.75	0.76	819
60-80	0.84	0.79	0.81	889
80-100	0.98	0.98	0.98	861
Úspěšnost			0.84	4 211
Průměr	0.84	0.84	0.84	4 211
Vážený průměr	0.84	0.84	0.84	4 211

Tabulka 7.11: Výsledky klasifikátoru pro 10 tříd

	Přesnost	Senzitivita	F1-score	Četnost
0-10	0.79	0.74	0.76	186
10-20	0.72	0.75	0.73	163
20-30	0.76	0.76	0.76	180
30-40	0.69	0.72	0.70	186
40-50	0.64	0.65	0.65	170
50-60	0.63	0.70	0.66	178
60-70	0.48	0.42	0.45	180
70-80	0.65	0.65	0.65	216
80-90	0.80	0.76	0.78	185
90-100	0.91	0.94	0.92	184
Úspěšnost			0.71	1 828
Průměr	0.71	0.71	0.71	1 828
Vážený průměr	0.71	0.71	0.71	1 828

Doposud bylo přistupováno k problematice pomocí rozdělení úspěšností do tříd. Nyní budeme chtít provádět regresní predikci. Princip je založen na odhadování přímo numerické hodnoty. V praxi to znamená, že dodáme modelu vektor atributů a výsledkem bude přímo číslo, které bude reprezentovat predikovanou úspěšnost datové sady. Pro ověření, které modely jsou vhodné pro tento přístup byla už použita rovnou zmiňovaná knihovna `sklearn`. Knihovna poskytuje množinu regresních modelů, které splňují naše požadavky. Na základě příspěvku o predikování numerických hodnot [25], byly nad daty vyzkoušeny tyto modely:

- Linearní regresor
- Regresor rozhodovacího stromu
- Random forrest regresor
- Regresor neuronové sítě
- Regresor K-nejbližších sousedů

Po vyhodnocení všech regresních modelů, přišlo na řadu jejich porovnání. Metriky[4], jenž udávají kvalitu regresního modelů, se liší od třídních modelů. Mezi hlavní metriky patří průměrná absolutní chyba, jejíž hodnota udává o kolik se predikovaná úspěšnost průměrně liší v porovnání s opravdovou hodnotu. Dále můžeme určit maximální chybu, která zachycuje největší rozdíl mezi predikovanou a reálnou hodnotou. Poslední zjišťovaná metrika je vysvětlený rozptyl (explained variance) vyjadřující rozptyl mezi predikovanými a reálnými hodnotami. Nejlepší hodnota metriky dosahuje jedničky a nižší hodnoty značí vyšší variaci mezi hodnotami dvou skupin. V tabulce 7.12 je zobrazeno porovnání jednotlivých modelů a zmíněných metrik.

Z tabulky můžeme vyčíst, že nejlepších výsledků dosahuje model Random forrest s průměrnou chybou 0.82. Ostatní modely vykazují též kvalitní výsledky. Ovšem nevýhodou

[4] https://scikit-learn.org/stable/modules/model_evaluation.html #regression-metrics

Tabulka 7.12: Porovnání regresních modelů

Model	Prům. chyba	Max. chyba	Vys. rozptyl
Linearní regrese	1.26	7.48	0.9946
Rozhodovací strom	1.15	8.46	0.9952
Random forrest	0.82	5.83	0.9976
Neuronová sít	1.02	5.54	0.9965
K-nejbližších sousedů	1.22	9.99	0.9945

Random forrestu může být jeho pomalejší klasifikace vyplývající z podstaty jeho implementace (delší čas z důvodu vyhodnocení jednotlivých stromů). Z toho důvodu by byl vhodnější spíše rozhodovací strom nebo neuronová síť, která ale není natolik transparentní.

Celkový úspěch implementace můžeme považovat za zdařilý. Když vezmeme v potaz, že do budoucna se budeme snažit získat kvalitnější množinu atributů, která umožní vytvořit kvalitnější rozhodovací model. Za zmínku také stojí různá kvalita klasifikátorů na základě trénovací sady. Trénovací datová sada má velký vliv na konečný výsledek. Když jsme aplikovali nad dvěma různými sadami stejný klasifikátor pro 4 třídy, tak výsledná úspěšnost se lišila až o 12 %.

Proti třídní klasifikaci jsme postavili kontinuální predikci numerických hodnot pomocí regresních technik. V porovnání s třídní klasifikací dosahovala přesnějších výsledků. Nejhorší regresivní model dosahoval maximální odchylky 9.99, což je maximální chyba u klasifikace do 10 tříd v případě správné predikce. Nehledě na to, že při špatné klasifikaci může dojít k mnohem vyšší odchylce. V případě regrese hrála roli trénovací datová sada. V jednom případě jsme dosáhli průměrné odchylky kolem 1 a v tom horším případě kolem 5.

Kapitola 8

Závěr

Cílem práce bylo vytvořit NEMEA modul, který bude sloužit k detekování DGA adres. K vyřešení problematiky bylo potřeba nastudovat strojové učení, princip DGA adres a cílovou platformu NEMEA. Praktická část obsahovala návrh atributů a implementaci finálního klasifikátoru v podobě rozhodovacího stromu. Výsledný detektor prošel testovacím a optimalizačním procesem.

Výsledná úspěšnost DGA detektoru se pohybuje okolo 99 % na testovací datové sadě. U reálných dat nejsme schopni určit přesnost kvůli neoznačeným datům ze sítě. Modul dokáže zpracovat zhruba 30 000 adres za sekundu (záleží na výkonu zařízení). Po provedení testu na paměťovou náročnost, modul zabírá okolo 640 MB operační paměti. DGA detektor se stal součástí oficiálního repozitáře NEMEA. Nasazením DGA detektoru v rámci NEMEA můžeme odchytávat nebezpečný provoz na síti. Systém NEMEA se tak rozšíří o další detektor, který může být použit k odhalení bezpečnostních incidentů na monitorované síti. Zachycením podvržených doménových adres můžeme vystopovat nakažené počítače v síti. Do budoucna je možné DGA detektor snadno přetrénovat, takže se bude snadno udržovat jeho přesnost. Dalším plánem je možná klasifikace na úrovni subdomén. Pokud to bude výkonnostně dostačující, tak by se atributy počítaly pro každou subdoménu do určité úrovně zanoření zvlášť.

Cílem kapitoly s predikcí úspěšnosti bylo vytvořit klasifikátor, který bude predikovat úspěšnost datových sad s doménovými jmény. Součástí byla tvorba atributů, jenž charakterizují datovou sadu. Nejvhodnějším modelem je rozhodovací strom pro třídní i regresní klasifikaci.

Predikce do 4 tříd dosahovala úspěšnosti 94 %. Při chybné klasifikaci dojde vždy jen k zařazení do sousední kategorie. Dokazuje to funkční a realizovatelný vývoj klasifikátoru. U více kategorií už byla klasifikace méně přesná. Regresní techniky oproti třídnímu přístupu vykazovaly vyšší přesnost s průměrnou odchylkou 1. Hlavním přínosem predikování úspěšností je zrychlení generování datových sad, jenž obsahují doménové jména. Navržený způsob ohodnocovaní může až desetkrát urychlit získání finální úspěšnosti. R model jako referenční klasifikátor dokáže zpracovat 1 000 datových sad za 1323.87 s. Ohodnocovací klasifikátor dokáže stejný počet sad ohodnotit za 125.55 s. Dále získáme schopnost ohodnotit datovou sadu pomocí množiny parametrů, pomocí kterých můžeme odhalit slabiny zkoumaného modelu. Predikování úspěšností je součástí většího projektu, na kterém pracuje můj vedoucí a po dalším zkvalitnění stávajících výsledků by mohl tento přístup rapidně urychlit proces generování umělých datových sad pro DGA klasifikaci.

Literatura

[1] Alpaydin, E. *Introduction to Machine Learning*. 2. vyd. London, England: The MIT Press, 2010. ISBN 9780262012430.

[2] Bowles, J. *N-gram Models: Part 1* [online]. 2019 [cit. 20. ledna 2020]. Dostupné z: https://nextjournal.com/jbowles/n-gram-models-part-1.

[3] Brownlee, J. Logistic Regression for Machine Learning. *Machine Learning Algorithms* [online]. 2016. Aktualizováno 12. 8. 2019 [cit. 20. ledna 2020]. Dostupné z: https://machinelearningmastery.com/logistic-regression-for-machine-learning/.

[4] Cejka, T., Bartos, V., Svepes, M., Rosa, Z. a Kubatova, H. *NEMEA: A Framework for Network Traffic Analysis*. 2016. DOI: 10.1109/CNSM.2016.7818417. ISBN 978-1-5090-3236-5. Dostupné z: http://dx.doi.org/10.1109/CNSM.2016.7818417.

[5] Chang, M. *4 Stages of the Machine Learning (ML) Modeling Cycle* [online]. LinkedIn, listopad 2017 [cit. 2020-12-15]. Dostupné z: https://www.linkedin.com/pulse/4-stages-machine-learning-ml-modeling-cycle-maurice-chang.

[6] Editor. Fully Qualified Domain Name (FQDN). *Network Encyclopedia* [online]. 2020 [cit. 25. prosince 2020]. Dostupné z: https://networkencyclopedia.com/fully-qualified-domain-name-fqdn/.

[7] Glen, S. Jaccard Index / Similarity Coefficient. *StatisticsHowTo.com: Elementary Statistics for the rest of us!* [online]. Prosinec 2016 [cit. 21. prosince 2020]. Dostupné z: https://www.statisticshowto.com/jaccard-index/.

[8] Hatwell J., G. M. . A. R. *Explaining random forest classification* [Artif Intell Rev 53]. Červen 2020 [cit. 25. března 2021]. Dostupné z: https://doi.org/10.1007/s10462-020-09833-6.

[9] Hofmann, M. a Klinkenberg, R. *RapidMiner: Data Mining Use Cases and Business Analytics Applications*. 1. vyd. Taylor & Francis, 2013. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. ISBN 9781482205497. Dostupné z: https://books.google.cz/books?id=5zYTAgAAQBAJ.

[10] Holmes, G., Donkin, A. a Witten, I. *WEKA: a machine learning workbench*. 1994. DOI: 10.1109/ANZIIS.1994.396988. Dostupné z: https://ieeexplore.ieee.org/document/396988.

[11] Hope, C. Linux host command. *Computer Hope Free computer help since 1998* [online]. 2019. Aktualizováno 05. 04. 2019 [cit. 20. ledna 2020]. Dostupné z: https://www.computerhope.com/unix/host.htm.

[12] Hope, C. Linux whois command. *Computer Hope Free computer help since 1998* [online]. 2019. Aktualizováno 05. 04. 2019 [cit. 20. ledna 2020]. Dostupné z: https://www.computerhope.com/unix/uwhois.htm.

[13] Jacobs, J. Feature Engineering. *Building a DGA Classifier* [online]. 2014 [cit. 1. března 2021]. Dostupné z: https://datadrivensecurity.info/blog/posts/2014/Oct/dga-part2/.

[14] Kozlowski, L. *Shannon entropy calculator: Real example how to calculate and interpret information entropy* [online]. 2003. Aktualizováno 2020 [cit. 20. ledna 2020]. Dostupné z: https://www.shannonentropy.netmark.pl/.

[15] Lee, Y. W., Strong, D. M., Kahn, B. K. a Wang, R. Y. *AIMQ: a methodology for information quality assessment*. 2002. DOI: https://doi.org/10.1016/S0378-7206(02)00043-5. ISSN 0378-7206. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0378720602000435.

[16] Lipton, Z. C. *The Mythos of Model Interpretability*. 2016. Dostupné z: http://arxiv.org/abs/1606.03490.

[17] Lytvynova, K. Machine Learning Project Structure: Stages, Roles, and Tools. *Datafloq* [online]. 2018. Aktualizováno 01. 05. 2018 [cit. 20. ledna 2020]. Dostupné z: https://datafloq.com/read/machine-learning-project-structure-stages-roles/4941.

[18] Mischel, J. How to choose size of hash table? *Stack Overflow* [online]. 2014 [cit. 28. června 2020]. Dostupné z: https://stackoverflow.com/questions/22741966/how-to-choose-size-of-hash-table.

[19] Mockapetris, P. *DOMAIN NAMES - CONCEPTS AND FACILITIES* [Internet Requests for Comments]. RFC 1034. RFC Editor, November 1987. Dostupné z: https://tools.ietf.org/html/rfc1034.

[20] Mroman. Entropy. *Rosetta Code* [online]. 2013. Aktualizováno 13. 01. 2020 [cit. 20. ledna 2020]. Dostupné z: https://rosettacode.org/wiki/Entropy.

[21] mtt5. *RPZ and Botnet Command and Control Server Traffic* [online]. Březen 2015 [cit. 25. prosince 2020]. Dostupné z: https://blogs.kent.ac.uk/unseenit/rpz-and-botnet-command-and-control-server-traffic/.

[22] Pierce, R. *'Standard Deviation and Variance' Math Is Fun* [online]. Prosinec 2020 [cit. 26. ledna 2021]. Dostupné z: http://www.mathsisfun.com/data/standard-deviation.html.

[23] Ramnit. DGA. *Netlab OpenData Project* [online]. 2019. Aktualizováno 19. 01. 2020 [cit. 20. ledna 2020]. Dostupné z: https://data.netlab.360.com/dga/.

[24] Rathi, S. Top 8 Python Libraries for Machine Learning & Artificial Intelligence. *Hackernoon* [online]. 2019. Aktualizováno 09. 8. 2019 [cit. 20. ledna 2020]. Dostupné z: https://hackernoon.com/top-8-python-libraries-for-machine-learning-and-artificial-intelligence-y08id3031.

[25] Ronaghan, S. *Machine Learning: Trying to predict a numerical value* [online]. Červenec 2018 [cit. 26. března 2021]. Dostupné z: https://srnghn.medium.com/machine-learning-trying-to-predict-a-numerical-value-8aafb9ad4d36.

[26] Saha, S. *What is the C4.5 algorithm and how does it work?* [online]. 2018 [cit. 26. prosince 2020]. Dostupné z: https://en.wikipedia.org/wiki/C4.5_algorithm.

[27] Schüppen, S., Teubert, D., Herrmann, P. a Meyer, U. FANCI : Feature-based Automated NXDomain Classification and Intelligence. In: USENIX. *27th USENIX Security Symposium*. 2018. ISBN 978-1-939133-04-5. Dostupné z: https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-schuppen.pdf.

[28] Solutions, E. How to evaluate the performance of a model in Azure ML and understanding "Confusion Metrics". *Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures* [online]. Září 2016 [cit. 29. ledna 2021]. Dostupné z: https://bit.ly/39tjN5M.

[29] team, D. F. Learn Types of Machine LearningAlgorithms with Ultimate Use Cases. *Machine Learning Tutorials* [online]. 2019. Aktualizováno 05. 8. 2019 [cit. 20. ledna 2020]. Dostupné z: https://data-flair.training/blogs/types-of-machine-learning-algorithms/.

[30] team, D. F. 11 Top Machine Learning Algorithms used by Data Scientists. *Machine Learning Tutorials* [online]. 2019. Aktualizováno 11. 10. 2019 [cit. 20. ledna 2020]. Dostupné z: https://data-flair.training/blogs/machine-learning-algorithms/.

[31] Trost, J. *Getting Started with DGA Domain Detection Research* [online]. Červenec 2019 [cit. 15. prosince 2020]. Dostupné z: https://medium.com/@jason_trost/getting-started-with-dga-domain-detection-research-89af69213257.

[32] Yiu, T. How the Algorithm Works and Why it Is So Effective. *Understanding Random Forest* [online]. Červenec 2019 [cit. 26. prosince 2020]. Dostupné z: https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

[33] Čejka, T. CESNET/Nemea. *GitHub* [online]. 2015. Aktualizováno 14. 05. 2018 [cit. 20. ledna 2020]. Dostupné z: https://github.com/CESNET/Nemea.

Příloha A

Obsah přiloženého paměťového média - CD

Na paměťovém médiu se nacházejí následující adresáře:

- **/Text BP** - Zdrojové soubory práce pro LATEX a práce ve formátu PDF
- **/DGA detektor** - Zdrojové soubory detekčního NEMEA modulu a skript pro vytvoření klasifikátoru
- **/Ohodnocovací klasifikátor** - Zdrojové soubory pro vytvoření modelu pro predikci úspěšností