VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ANALÝZA ŠKODLIVÉHO ŠIFROVANÉHO SÍŤOVÉHO PROVOZU

ANALYSIS OF MALICIOUS ENCRYPTED NETWORK TRAFFIC

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **BRANISLAV DUBEC**
AUTHOR

VEDOUCÍ PRÁCE **Mgr. Ing. PAVEL OČENÁŠEK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Dubec Branislav**
Program: Informační technologie
Název: **Analýza škodlivého šifrovaného síťového provozu**
Analysis of Malicious Encrypted Network Traffic
Kategorie: Web

Zadání:

1. Seznamte se s principy analýzy anomálií v prostředí systémů počítačových sítí.
2. Analyzujte požadavky na systém umožňující analýzu šifrované komunikace z nižších vrstev modelu OSI a metadat za pomocí vybraných metod umělé inteligence obecně a se zaměřením na HTTPS.
3. Navrhněte systém pro detekci bezpečnostních a provozních anomálií dle předchozího bodu.
4. Navržený systém implementujte dle instrukcí vedoucího práce.
5. Implementovaný systém ověřte na vhodně zvolených reálných datech.
6. Diskutujte získané výsledky a možnosti dalšího rozšíření.

Literatura:

- Kurose, J. F. Computer networking: A top-down approach. Pearson, Essex, 2017, ISBN 978-1-292-15359-9.
- Stallings, W. Network security essentials: Applications and standards. Hoboken, 2016, ISBN 978-0-13-452733-8.
- Bishop, M. Computer security: Art & Science. Addison-Wesley, Boston, 2003, ISBN 0-201-44099-7.
- Ahmed, M., Mahmood Naser, A., Hu, J. A survey of network anomaly detection techniques. Journal of network and computer applications. Elsevier, 2016, 60(C), s. 19-31. ISSN 1084-8045.
- Buczak, A., Guven, E.. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE Communications surveys and tutorials. IEEE, 2016, 18(2), s. 1153-1176.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 - 3 zadání.

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

Vedoucí práce: **Očenášek Pavel, Mgr. Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 27. října 2020

Abstrakt

Táto bakalárska práca sa venuje analýze škodlivej, šifroavnej sieťovej prevádzky pomocou metód umelej inteligencie. Riešením je vytvorenie systému pre detekciu bezpečnostných prienikov pomocou metódy detekcie analýzy. V teoretickej časti popisuje metódy detekcie anomálií a vysvetľuje pojem umelej, neurónovej siete. V praktickej časti experimentuje s rôznymi technikami detekcie anomálií na získanie najlepšieho výsledku.

Abstract

This bachelor thesis deals with the analysis of malicious encrypted network traffic using artificial intelligence methods. A solution is to create a system for detecting security intrusions using detection analysis methods. Theoretical part describes methods of anomaly detection, and explains the concept of artificial neural network. In the practical part, it experiments with various anomaly detection techniques in order to obtain the best results.

Kľúčové slová

detekcia anomálií, ja3, TCP tok, strojové učenie

Keywords

detection of anomalies, ja3, TCP flow, machine learning

Citácia

DUBEC, Branislav. *Analýza škodlivého šifrovaného síťového provozu.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Mgr. Ing. Pavel Očenášek, Ph.D.

Analýza škodlivého šifrovaného síťového provozu

Prehlásenie

Prehlasujem, že som bakalársku prácu vypracoval samostatne pod vedením pána Ing. Pavla Očenáška, Ph.D. Ďalšie informácie mi poskytol Ing. Petr Chmelár. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Branislav Dubec
11. mája 2021

Poďakovanie

Týmto by som chcel poďakovať vedúcemu mojej práce Ing. Pavlovi Očenáškaovi Ph.D., za jeho cenné rady a odbornú pomoc a konzultácie, a konzultantovi Ing. Petrovi Chmelárovi za jeho cenné rady, odbornú pomoc a konzultácie, ktoré mi poskytli pri vypracovávaní bakalárskej práce. Zároveň by som chcel poďakovať mojej rodine, predovšetkým mojej mamine Ing. Márii Dubcovej, za podporu a rady.

Obsah

Kapitola 1

Úvod

Počítače a počítačové siete sa používajú na kontrolovanie a riadenie výrobných procesov, burzu cenných papierov, v systémoch na kontrolu leteckého priestoru a mnoho ďalších. Dnešné siete sú veľmi heterogénne, kde vysoko kritická softwarová aplikácia (napríklad na kontrolu chodu systému) beží ruka v ruke s nie tak kritickými, sieťovo založenými aplikáciami (mail server). Preto je nebezpečný aj nepatrný útok na ne-kritickú službu, ktorý dokáže spôsobiť predom nepredvídateľné vedľajšie poškodenia [16].

V mojej práci sa preto zameriam na bezpečnosť sietí, presnejšie na systém pre zachytávanie prienikov do sietí založenej na detekovanie anomálií. Tento systém je schopný detekovať škodlivý software v zašifrovanej komunikácií, čo je podstatné pri komunikácií po zabezpečených sieťach. V kapitole číslo dva predstavím základné systémy na detekovanie prienikov a definujem bližšie systém založený na detekovanie anomálií. Vysvetlím základné aspekty pre systém, ktorý detekuje anomálie, ktoré sú vstupné dáta, ako aj druh hľadanej anomálie. Ďalej ukážem základné techniky detekcie anomálií, ich výhody a nevýhody. V tretej kapitole vysvetlím pojem umelá neurónová sieť, z čoho sa skladá biologický neurón a umelý neurón, a aktivačné funkcie umelého neurónu. Predstavím architektúry umelej neurónovej siete, jej jednotlivé vrstvy a uzly spoločne s ich významom. V ďalšej kapitole opíšem návrh môjho naimplementovaného systému, ako aj implementáciu. Opíšem technológiu JA3 fingerprintov, ktorú používam na zistenie škodlivých TCP tokov v datasetoch. Ukážem ako vyzerá môj dataset aj s úpravami pre strojové učenie. Predstavím knižnicu *scikit-learn*, ktorú používam pre tvorenie modelov. V piatej kapitole si vyberiem štyri techniky detekcie anomálií založených na klasifikácií, vyladenie ich jednotlivých parametrov a zhodnotenie výsledkov. V závere opíšem návrh na zlepšenie týchto techník, alebo vylepšenie trénovacích dát.

Kapitola 2

Metódy detekcie anomálií

2.1 Čo je to anomália?

Autori **Chandola et al.** v prieskume o detekcii anomálií [5] určujú, že anomália je určitý vzor v dátach, ktorý nevyhovuje do dobre definovanej predstavy o normálnom správaní. Anomálie môžu byť vyvolané v dátach z rôznych dôvodov, akými sú škodlivá aktivita napríklad pri podvodoch s kreditnými kartami, kyber-útokmi, teroristickými útokmi alebo slúžia na prelomenie systému, ale všetky tieto dôvody majú spoločnú charakteristiku; všetky sú zaujímavé pre človeka, čo ich analyzuje. Práve táto zaujímavosť alebo relevantnosť anomálií v reálnom živote je kľúčový faktor pri detekcii anomálií.

2.1.1 Výzvy pri určovaní anomálie

Na abstraktnej úrovni je anomália definovaná ako vzor, ktorý sa nepodobá predpokladanému normálnemu správaniu. Priamy prístup k detekovaniu anomálií je preto definovanie regiónov reprezentujúce normálne správanie a deklarovanie, čo i len najmenšej spozorovanej inštancií dát, ktoré nepatria do tohto regiónu normálneho správania, ako anomáliu. Avšak niekoľko faktorov obmedzuje aj takto zjavne jednoduchý prístup a robí ho celkom problémový:

- Definovanie normálneho regiónu, ktorý zahŕňa každé možné normálne správanie je veľmi zložité. Dokonca rozdiel medzi normálnym a anomálnym správaním je často nepresný. Preto inštancia, ktorá sa nachádza blízko ku hranici medzi týmito regiónmi, môže považovať za normálnu a zároveň za anomálnu.
- Problém nastáva, keď sú anomálie výsledkom škodlivej aktivity. Útočníci, alebo škodlivé ativity, sa často adaptujú, aby sa pre vonkajší svet tvarili ako normálne; preto je definovanie normálneho správania ťažšie ako zvyčajne.
- V mnohých oblastiach sa normálne správanie mení a evolvuje, a preto aktuálne vnímanie normálneho správania nemusí byť dostatočne reprezentujúce s odstupom času.
- Presná predstava anomálie je rozdielna pre rozdielne oblasti. Napríklad v oblasti medicíny sa už aj malá odchýlka od normálu môže považovať za anomáliu, ale podobná odchýlka v oblasti burzy cenných papierov sa považuje za normálnu. Kvôli tomuto aplikovanie jednej techniky v jednej oblasti nemusí fungovať v tej druhej.
- Dostupnosť dát pre trénovanie a validáciu modelov používanych technikami detekcie anomálií je zvyčajne veľkým problémom.

- Veľmi často dáta obsahujú šum, ktorý má sklon byť podobný anomálii, a preto je zložité ich rozlíšiť a odstrániť.

Kvôli vyššie spomenutým výzvam je detekovanie anomálií problém, ktorý nie je vo všeobecnej forme ľahko riešiteľný. V skutočnosti väčšina techník detekcií anomálií, ktorá existuje, dokáže vyriešiť alebo sa špecifikuje iba na jeden konkrétny problém. Formulácia tohto problému závisí od viacerých faktorov, akou je podstata dát, dostupnosti označení dát, typ anomálie na detekovanie a iné. Tieto faktory sú determinované oblasťou, v ktorej je potrebné hľadať anomálne správanie. Výskumníci si adoptovali koncepty rôznych disciplín akými sú štatistiky, strojové učenie, získavanie dát a aplikovali ich do špecifických problémov [5].

2.2 Detekovanie prienikov

Už v roku 1980 **Anderson** [4] predstavil koncept detekcie prienikov, kde definoval pokusy o prienik alebo hrozbu ako neoprávnený prístup do siete so snahou:

1. získať informácie,
2. manipulovať informácie,
3. zmeniť systém tak, aby nebol použiteľný alebo spoľahlivý.

Počítačové siete by mali zaručiť dôvernosť, integritu a záruku bezpečnosti [26]. Avšak to je nemožné splniť na 100%, či už je to z dôvodu zväčšenej používateľnosti počítačových sietí na internete alebo z dôvodu nedostatku finančných prostriedkov vyhradených na bezpečnosť. Útočníci využívaju všetky možné bezpečnostné chyby alebo nedostatky, aby to využili vo svoj prospech. Dokonca aj najväčšie technické firmy, ako napríklad Google, nedokážu zabrániť všetkým útokom. Dôkazom je gooligan vírus v roku 2018, ktorý prenikol do google účtov.

Ako teda zabrániť útočníkom preniknúť do počítačovej siete? Jednou možnosťou je vytvoriť takú sieť, ktorá je kompletne bezpečná. Každý užívateľ by sa pri prístupe musel identifikovať a musel by byť overený; používať rôzne kryptografické metódy. Toto je žiaľ podľa **Sundarama** [26] neuskutočniteľné hneď z niekoľkých dôvodov:

- V praxi nie je možné vytvoriť kompletne bezpečnú sieť. Útočníci využívajú najnovšie prostriedky, takže by sa musel tento systém neustále obnovovať, čo je finančne veľmi náročné a nepraktické.
- Každá kryptografická metóda je zraniteľná. Heslá môžu byť ukradnuté, prelomené alebo ich užívateľ môže zabudnúť.
- Aj dokonalý systém je zraniteľný zvnútra administrátormi s väčšími privilégiami.

Preto musíme počítať s tým, že počítačové siete sú zraniteľné. Ak sa uskutočňuje útok na sieť, je potrebné na to prísť čo najskôr, najlepšie v reálnom čase a vyvodiť dôsledky.

Systém nazývaný ako NIDS - Network Intrusion Detection System (systém na detekovanie prienikov) [14] je základný nástroj pre zisťovanie rôznych narušení v sieti správcami systémov. Existujú dva hlavné typy NIDS, ktoré sa rozdeľujú na základe metód, akými sú zisťované prieniky:

i SNIDS signature (misuse) based NIDS,

ii ADNIDS anomaly detection based NIDS.

Koncept detekcie pomocou SNIDS podľa **Kumara a Sangwana** [19] spočíva v tom, že útoky sú reprezentované ako vzor alebo ako podpis, ktorý je porovnávaný s už známymi vzormi. Tieto systémy dokážu zachytiť už známe vírusy, ale zlyhávajú pri predom neznámych útokoch alebo útokoch, ktoré zahrňujú práva v systéme. Vedomosť o útoku záleží na operačnom systéme a verzii. čiže aplikácia tohto systému je viazaná k špecifickému prostrediu.

Avšak SNIDS nedokážu dobre zachytávať prieniky, s ktorými sa ešte nestretli a ktorých vzor nepoznajú. Práve preto sa vytvárajú rôzne štúdie o anomáliach. Detekovanie anomálií sú definované viacerými faktormi, ktoré ukážem v ďalších sekciách.

2.2.1 Typy útokov na sieť

Útok alebo hrozba sa vzťahuje na všetko, čo nejakým škodlivým spôsobom mieri na kompromitujúcu sieť. Zlý dizajn siete, nedbanie o bezpečnosť ich užívateľov alebo zlá konfigurácia jej softwaru a hardwaru môžu byť zraniteľné a využité pre potencionálne hrozby [17]. **Ahmed et al.** [2] opísujú štyri základné útoky :

1. **Denial of Service** : *(DoS)* je taký typ zneužitia práv, kde sú prostriedky siete zamierené na prerušenie normálneho výpočetného prostredia a učinia tým túto službu neprístupnú. Základný príklad *DoS* útoku je odoprenie legitímnych užívateľov k prístupu na webovú službu, keď je server zaplavený veľkým počtom požiadaviek o pripojenie. Vykonanie *DoS* útoku nevyžaduje žiaden predchádzajúci prístup na cieľ, preto je považovaný za veľmi obávaný útok.

2. **Probe**: Sonda sa používa na získavanie informácií o cielenej sieti a využíva sa na výzvedné účely. Tieto útoky sú bežné spôsoby získavania informácií o typoch a počtoch strojov pripojených na sieť, alebo určenia typu softwaru nainštalovaného a používaného na sieti. Útok pomocou sondy je považovaný za prvý krok horšieho útoku na zabavenie celkovej siete. Aj keď tento útok nevykonáva žiadne viditeľné poškodenia, je považovaný za serióznu hrozbu pre korporácie, pretože útočníci môžu získať užitočné informácie pre spustenie ďalšieho, nebezpečnejšieho útoku.

3. **User to Root**: (U2R) je taký typ útoku, ktorý sa spustí, keď útočník zamieri na získanie nelegálneho prístupu do adminitratívneho účtu pre manipulovanie alebo zneužitie prostriedkov. Použitím prístupu sociálneho inžinierstva alebo získania hesla, môže útočník získať účet bežného užívateľa a využiť ho, alebo zistiť zraniteľnosť na získanie práv super-užívateľa.

4. **Remote to User**: (R2U) sa spustí, keď chce útočník získať lokálny prístup ako užívateľ zamiereného stroja, aby mal privilégium posielania pakety po sieti. Najčastejšie používa útočník metódu pokus-omyl na získanie hesla pomocou automatizovaných skriptov, hrubou silou alebo inými metódami. Existujú aj sofistikovanejšie metódy, kde útočník nainštaluje nástroj na zachytenie hesla pred vniknutím do systému.

2.3 Vstupné dáta

Dôležitým aspektom každej detekcie anomálií je v charateristike vstupných dát. Všetky dátové inštancie môžu byť popísané sadou vlastností. Tieto vlastnosti môžu byť rôznych

typov ako *binárne, kategorické* a *kontinuálne*. Každá dátová inštancia sa môže skladať iba z jednej vlastnosti (univariate) alebo z viacerých vlastností (multivariate). V prípade, že dátová inštancia je typu *multivariate*, všetky vlastnosti môžu byť toho istého typu alebo môžu byť zložením rôznych dátových typov.

Vstupné dáta môžu byť kategorizované na základe vzťahu medzi jednotlivými dátovými inštanciami. Väčšina techník na detekovanie anomálií pracujú iba s dátami, medzi ktorými je predpokladané, že nemajú žiaden vzťah medzi sebou. Avšak vo všeobecnosti môžu dátové inštancie spolu súvisieť. Napríklad sekvenčné dáta sú dátové inštancie zoradené lineárne, v priestorových dátach každá dátová inštancia súvisí s ich susednými inštanciami alebo v grafových dátach sú inštancie reprezentované ako body a sú spojené s vrcholmi alebo hranami [5].

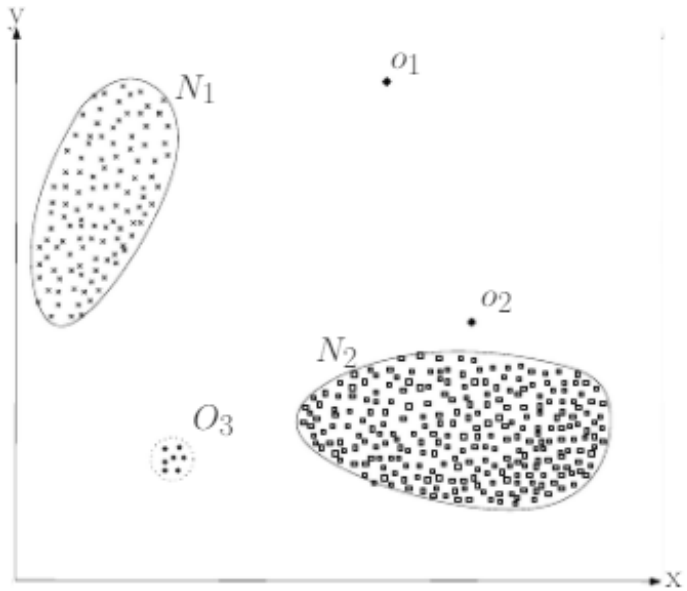
2.4 Typy anomálií

Ďalším dôležitým aspektom techník detekovania anomálií je v druhu hľadanej anomálie. **Chandola et al.** [5] zaraďujú anomálie do troch kategórií:

2.4.1 Bodové anomálie

Ak sa môže jednotlivá dátová inštancia považovať za anomáliu k zvyšným dátam, nazývame túto anomáliu ako bodovú anomáliu.

Na obrázku 2.1 body o_1 a o_2 a aj body v dátovom sete O_3 ležia mimo N_1 alebo N_2 a tým pádom hovoríme o dátových anomáliach.



Obr. 2.1: Ukážka anomálií v 2D dátovej sady. Prevzaté z [5].

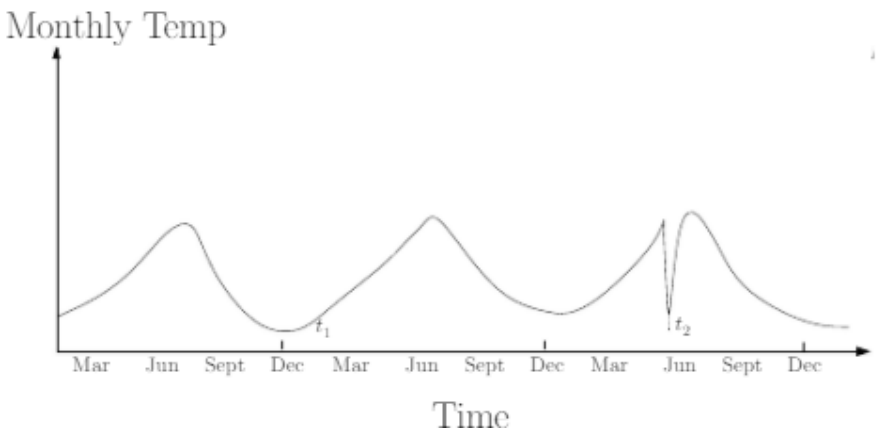
2.4.2 Kontextové anomálie

Ak je v dátovej sade anomália iba v nejakom špecifickom kontexte, ale nie inak, nazývame túto anomáliu ako kontextovú anomáliu. Pojem kontextu je predstavený v štruktúre dátových setov a musí byť špecifikovaný ako časť problému. Každá dátová inštancia je definovaná týmito vlastnosťami:

- Kontextuálne vlastnosti sa používajú, aby určili kontext pre inštanciu. Napríklad čas je v časovo sledovaných dátach kontextová vlastnosť, ktorá určuje anomálie v čase.
- Vlastnosti správania definujú nekontextové charakteristiky inštancie. Napríklad v dátovom sete opisujúce priemerné zrážky na celom svete, veľkosť zrážok na nejakom konkrétnom mieste je vlastnosť správania.

Jednotlivé anomálie sú detekované použitím vlastností správania v špecifickom kontexte. Dátová inštancia môže byť kontextuálna anomália v danom kontexte, ale identická dátová inštancia sa môže považovať ako normálna v rozdielnom kontexte. Táto vlastnosť je kľúčová pre identifikovanie kontextuálnych vlastností a vlastností správania pre detekovanie kontextuálnych anomálií.

Na obrázku 2.2 môžeme vidieť kontextovú anomáliu. Vidíme, že teplota t_1 je taká istá ako teplota t_2, ale vyskytuje sa v rozdielnom kontexte a preto sa považuje iba t_2 ako anomália [5].



Obr. 2.2: Kontextová anomália. Prevzaté z [5].

2.4.3 Kolektívne anomálie

Ak skupina dátových inštancií, ktoré majú podobné alebo rovnaké vlastnosti, sa chovajú anomálne k zvyšku dátovej sady, hovoríme, že ide o kolektívnu anomáliu. Ako ukážkový príklad si môžeme pozrieť sekvenciu akcií, ktoré sa vykonávajú v počítači:

. . . http-web, buffer-overflow, http-web, http-web, smtp-mail, ftp, http-web, ssh, smtp-mail, http-web, **ssh,buffer-overflow**,ftp, http-web, ftp, smtp-mail,http-web . . .

Hrubom označená sekvencia udalostí ukazuje typický webový útok diaľkovým strojom, za ktorým nasleduje kopírovanie dát zo serveru do diaľkového stroja pomocou *ftp*. Samotné udalosti anomáliu netvoria, iba keď sú za sebou alebo na rozličných miestach v sekvencii [5].

2.5 Označovanie dát anomáliou

Dátová inštancia sa označuje ako normálna alebo anomálna. Často je zložité, aby sme presne dokázali označiť dáta ako normálne alebo anomálne. Preto je toto označenie vykonávané zvyčajne ľudským expertom. Typicky je dané, že označiť anomálne správanie v sade anomálnych dátových inštancií, ktoré poskytujú všetky možné typy anomálneho správania, je zložitejšie ako označiť anomálne správanie pre normálne dátové inštancie. K tomu sa ešte anomálne správanie môže meniť, vyvíjať a môžu sa vynájsť aj nové. Preto **Chandola et al.** [5] delia techniky detekcie anomálií do 3 módov:

i Detekovanie anomálií s dohľadom

Techniky v móde s dohľadom predpokladajú, že tréningové dátové sady majú inštancie, ktoré sa rozdeľujú do normálnych alebo anomálnych tried. Typický prístup je vybudovanie modelu normálnej triedy a anomálnej triedy. Využíva sa pre známe typy útokov. Každá dátová inštancia je porovnávaná týmto modelom, aby sa zistilo, do ktorej triedy patrí. Týmto prístupom sa vyskytujú dva hlavné problémy. Po prvé, anomálnych inštancií je počtom menej ako normálnych inštancií v tréningovej dátovej sade. Po druhé, je obtiažne dosiahnuť presné a reprezentatívne označenie pre anomálnu triedu. Práve tento mód využívam na zistenie anomálií v kapitole číslo 5.

ii Detekovanie anomálií s čiastočným dohľadom

Techniky, ktoré pracujú v móde s čiastočným dohľadom predpokladajú, že tréningové dáta majú iba jednu normálnu triedu. Keďže nepotrebujú označenie pre anomálnu triedu, majú širšie využitie ako technika s plným dohľadom.

iii Detekovanie anomálií bez dohľadu

Techniky detekovania anomálií bez dohľadu nemajú dáta, ktoré majú označenie normálnej alebo anomálnej triedy, a preto sa využívajú najčastejšie. Tieto techniky implicitne predpokladajú, že normálne inštancie sa stávaju oveľa častejšie ako anomálie v testovacej sade. Ak je tento predpoklad nesprávny, potom táto technika má veľkú chybovosť.

2.6 Techniky detekcie anomálií

Existujú rôzne techniky detekcie anomálií. Každá má svoje výhody a nevýhody. **Chandola et al.** [5] opísali rôzne techniky detekovania, ktoré priblížim v tejto časti.

2.6.1 Detekovanie anomálií založené na klasifikácií

Metódy založené na klasifikácií fungujú v dvoj-fázovom režime. Trénovacia fáza učí klasifikátor (model) použitím dostupnou tréningovou sadou. Testovacia fáza potom klasifikuje testovaciu sadu ako normálnu alebo anomálnu použitím klasifikátora [5]. Techniky založené na klasifikácii sa spoliehajú na rozsiahle znalosti odborníkov o vlastnostiach sieťových útokov. Keď odborník na sieť poskytne detekčnému systému podrobnosti o charakteristikách útokov, je možné detekovať útok so známym vzorom hneď po jeho spustení. Výlučne to závisí od podpisu útoku ako od systému, ktorý je schopný detekovať útok, iba ak jeho podpis poskytol skôr sieťový expert. Toto demonštruje systém, ktorý dokáže zistiť iba to, o čom vie, že je škodlivé, a preto je zraniteľný voči novým, vylepšeným útokom, ktoré sa

neustále objavujú v rôznych verziách a sú nenápadne spustené. Aj keď je podpis nového útoku vytvorený a zabudovaný do systému, počiatočná strata je nenahraditeľná a postup opravy je mimoriadne drahý [2].

Metódy detekovania anomálií založené na klasifikácií fungujú na základe predpokladu, že klasifikátor, ktorý dokáže rozoznať medzi normálnou a anomálnou triedou, môže byť vytvorený na základe dostupných dát. Tieto techniky môžeme na základe tréningovej fázy rozdeliť na dve kategórie:

- Viac-triedne techniky na detekciu anomálií založené na klasifikácií predpokladajú, že tréningové dáta obsahujú prípady viacerých normálových tried. Tieto klasifikátory dokážu rozoznať medzi všetkými normálnymi triedami. Ak ani jeden klasifikátor nedokáže zaradiť testovaciu inštanciu ako normálnu, táto inštancia je deklarovaná ako anomália.

- Jedno-triedne techniky na detekciu anomálií založené na klasifkácií predpokladajú, že všetky trénovacie inštancie majú iba jednu triedu. Tieto klasifikátory vytvárajú diskriminatívne hranice okolo normálnych inštancií. Hocijaká testovacia inštancia, ktorá nespadá do týchto hraníc, je považovaná za anomáliu.

Neurónové siete

Na detekciu anomálií, založených na klasifikácií, je možné využiť neurónové siete a to pri jedno-triednych, ako aj pri viac-triednych nastaveniach. Viac-triedne techniky využívajú vo fáze tréningu neurónovú sieť na trénovanie na množine tréningových dát pre rozlíšenie rôznych tried. Potom je každá testovacia inštancia daná ako vstupné dáta do tejto neurónovej siete. Ak to neurónová sieť akceptuje, testovacia inštancia je normálová, ak nie, tak sme narazili na anomáliu. Jedno-triedne techniky používajú Replicator Neural Networks [5].

Bayesove siete

Bayesova sieť je efektívny prístup k modelovaniu domény obsahujúca neistotu. Diskrétna náhodná premenná je reprezentovaná pomocou smerovaného acyklického grafu (DAG), kde každý uzol odráža stav náhodnej premennej a obsahuje tabuľku podmienenej pravdepodobnosti (CPT). Úlohou CPT je poskytnúť pravdepodobnosť, že sa uzol bude nachádzať v konkrétnom stave. V Bayesovej sieti existuje medzi uzlami vzťah rodič-dieťa, čo naznačuje, že premenná predstavovaná podriadeným uzlom je závislá od tých, ktoré zastupujú nadradené uzly. Pretože táto sieť môže byť použitá pre schému klasifikácie udalostí, je použiteľná aj pre detekciu anomálií v sieti. Anomália nastáva, keď sa náhodná premenná nedokáže rozhodnúť podľa CPT, do ktorého uzla pôjde. **Kruegel et al.** [18] identifikujú dva hlavné problémy spôsobené vysoko falošnými pozitívami v technikách detekcie anomálií. Predpokladá sa, že systémy na zisťovanie anomálií obsahujú množstvo modelov na analýzu rôznych znakov udalostí. Prvým problémom je, že modely, ktoré poskytujú skóre alebo pravdepodobnosť normálnosti/abnormálnosti udalosti, vyžadujú, aby systém detekcie anomálií agregoval svoje rôzne výstupy, ktoré vedú k vysokým falošným pozitívam. Po druhé, systémy na zisťovanie anomálií nemôžu zvládnuť neobvyklé, ale legitímne správanie, napríklad náhle zvýšenie vyťaženia procesora, využitia pamäte atď. Ak sa vyskytne tento problém, ďalšie informácie vysvetľujú neobvyklé správanie, ktoré nie je anomálne [2].

Support Vector Machines

Esking et al. [8] uvádzajú, že základný princíp Support Vector Machine (SVM) techniky je odvodiť hyperplán, ktorý maximalizuje separačný okraj medzi normálnou a anomálnou triedou. Zaujímavá vlastnosť SVM techník je, že sa jedná o približnú implementáciu princípu minimalizácie rizika, založeného na teórii štatistického učenia. Štandardný algoritmus SVM je technika učenia pod dohľadom, ktorá na vytvorenie pravidla klasifikácie vyžaduje označené údaje. Môže sa však prispôsobiť ako algoritmus učenia bez dozoru, pričom sa pokúša oddeliť celú množinu tréningových údajov od ich pôvodu, zatiaľ čo bežný supervizovaný SVM sa pokúša oddeliť dve triedy dát v priestore funkcií pomocou hyperplánu [2].

SVM techniky sa dajú aplikovať iba pri jedno-triednych problémoch. Na sade označených trénovacích dát sa algoritmus natrénuje a vymedzí región, v ktorom sa nachádzajú normálné dáta. Následne sa počas testovacej fázy algoritmus rozhodne, či skúmaná dátová inštancia do vymedzeného regiónu spadá alebo nie. Ak sa dátová inštancia nachádza mimo vymedzeného regiónu, tak je označená za anomáliu [5].

Techniky založené na pravidlách

Tieto techniky spočívajú vo vytvorení pravidiel, ktoré zachytávajú normálne správanie. Testované inštancie, ktoré nespĺňajú tieto pravidlá, sú označené ako anomálie. Táto technika je aplikovateľná ako pre jedno-triedne, tak aj pre viac-triedne nastavenia. Počas testovania je každej inštancii nájdené pravidlo, ktoré ju najlepšie popisuje a je jej priradené skóre rovné inverzií hodnoty istoty použitého pravidla. Jedno-triedne techniky používajú pravidlo Association rule mining pre tvorenie pravidiel bez dozoru [5].

Klasifikačné techniky majú svoje dve veľké výhody, ale zároveň aj dve veľke nevýhody, na ktoré sa musí dať pozor pred tým, než ich použijeme. Medzi výhody patrí:

- viac-triedne techniky dokážu využiť silné algoritmy na rozlišovanie medzi jednotlivými triedami,
- testovacia fáza je rýchla, nakoľko sa inštancia iba porovnáva s vytvoreným modelom.

Medzi dve veľké nevýhody klasifikačných techník patrí:

- viac-triedne techniky sa spoliehajú na dostupnosť označených dát normálnych tried, čo je často nemožné,
- výstupom väčšiny klasifikačných metód je označenie každej inštancie, čo vie byť nevýhoda v prípadoch, kedy je vhodnejším výstupom skóre [5].

2.6.2 Detekovanie anomálií založené na najbližšom susedovi

Skupina metód založených na najbližšom susedovi pracuje s jedným základným predpokladom, a to takým, že normálne dáta sa vyskytujú v hustých susedstvách, zatiaľ čo anomálie sa vyskytujú osamote a ďaleko od akýchkoľvek iných dátových inštancií. Tieto techniky vyžadujú, aby bola vzdialenosť alebo podobnosť ako mierka na definovanie dvoch dátových inštancií. Túto vzdialenosť je možné vyrátať viacerými spôsobmi. Pre kontinuálne atribúty sa často používa Euklidovská vzdialenosť. Pre inštancie s viacerými atribútmi sú vzdialenosti zvyčajne počítané jednotlivo pre atribúty a potom kombinovane [5].

Medzi dva základné algoritmy patrí k-tý najbližši sused a relatívna hustota.

K-tý najbližší sused

Tento algoritmus je jeden z najznámejších a najviac používaných algoritmov pri detekcií anomálií bez učiteľa. Sústreďuje sa na detekciu globálnych anomálií a nie je schopný detekovať lokálne anomálie. Najprv sa pre každú dátovú inštanciu v datasete nájde k-najbližších susedov a následne sa pre každý bod v datasete vypočíta hodnota (skóre), ktorá určí, či sa jedná o anomáliu alebo nie [5].

Relatívna hustota

Tieto techniky odhadujú hustotu okolia všetkých inštancií. Inštancia, ktorá leží v riedkej oblasti, je označená ako anomálna a inštancia z hustej oblasti normálna. Vzdialenosť každej inštancie k jej k-tému najbližšiemu susedovi môže byť videná ako inverzia hustoty okolia (k-tých inštancií) danej inštancie. Tieto techniky nevykazujú dobré výsledky, ak dátová sada obsahuje regióny s rôznymi hustotami inštancií. Tento problém je možné adresovať technikami, ktoré počítajú hustotu okolia inštancie relatívne k hustotám okolitých inštancií [5].

Problémom základnej techniky najbližšieho suseda je vysoká výpočtová náročnosť, ktorá sa dá do istej miery zmenšiť použitím komplexnejších štruktúr k-d stromov. Aj táto metóda prináša svoje výhody a nevýhody.
Výhody techník založených na najbližšom susedovi:

- techniky bez dozoru nepotrebujú značené dáta a sú riadené čisto dátami,
- techniky s čiastočným dozorom sú lepšie v ohľade neidentifikovaných anomálií, nakoľko pravdepodobnosť anomálie blízko testovacích dát je nízka,
- jednoduchá aplikácia na inú doménu - prevažne definovanie spôsobu merania vzdialenosti.

Medzi nevýhody patrí:

- pri technikách bez dohľadu je možné, že normálne dáta nebudú dostatočne blízko pri sebe a anomálie budú príliš blízko pri sebe, spôsobujúc nesprávnu klasifikáciu,
- ak v technikách s čiastočným dozorom testované dáta nemajú dostatočnú koncentráciu normálnych inštancii, je vysoký počet falošných pozitív,
- výpočtová náročnosť,
- definovanie spôsobu merania vzdialenosti môže byť náročné pre komplexné dáta.

2.6.3 Detekovanie anomálií založené na zhlukovaní

Metódy detekcie anomálií v dátach založené na zhlukovaní sú používané takým spôsobom, že najprv sa dáta rozdelia do zhlukov. Následne sa pomocou vzdialenosti danej dátovej inštancie od centroidu rozhodne, či daný bod je anomáliou alebo nie.

Na základe uvedeného predpokladu sa dajú tieto techniky rozdeliť do troch skupín:

i Normálne dátové inštancie patria do zhlukov, zatiaľ čo anomálie nepatria do žiadneho zhluku. Tieto techniky používajú algoritmy, ktoré priradia inštancie do zhlukov, a tie, ktoré nepatria do žiadneho zhluku, sú považované za anomálne. Nevýhodou týchto algoritmov je to, že nie vždy nájdu anomáliu, keďže tieto techniky sa zameriavajú na priradenie inštancií do zhlukov.

ii Normálne dátové inštancie ležia blízko ku *stredu* ich najbližšieho zhluku a anomálie ležia ďaleko od *stredu* ich najbližšieho zhluku. Techniky operujúce na tomto predpoklade pozostávajú z dvoch krokov. V prvom kroku sú vytvorené zhluky zhlukovacím algoritmom. V druhom kroku je inštanciám pridelené skóre podľa ich vzdialenosti od stredu ich najbližšieho zhluku.

iii Normálne dátové inštancie tvoria veľké a husté zhluky, zatiaľ čo anomálie tvoria malé alebo riedke zhluky. Techniky založené na tomto predpoklade označia ako anomálie všetky inštancie, ktoré ležia v zhluku, ktorého veľkosť a/alebo hustota je pod určenou hranicou alebo inak anomálna.

Výhody zhlukovacích metód sú nasledovné:

- zhlukovacie metódy môžu fungovať samostatne,
- sú adaptívne, dokážu pracovať jednoducho aj s komplexnými dátovými typmi,
- časová zložitosť pre tréningovú fázu je rýchla, keďže počet zhlukov oproti jednotlivým inštanciam z testovacej sady je malý.

Nevýhody zhlukovacích metód:

- efektivita a výkon týchto metód záleží na algoritme pre vytváranie zhlukov z normálových inštancií,
- mnohé techniky sú primárne zamerané na zhlukovanie a nie sú optimalizované na detekciu anomálií,
- viaceré metódy zhlukovania nútia, aby každá inštancia mala svoj zhluk; tým pádom sa môže vytvoriť veľký a široký zhluk, ktorý obsahuje aj anomáliu a potom nedokáže nájsť chybné inštancie,
- niektoré techniky sú efektívne, iba ak anomálie netvoria zhluky samé o sebe [5].

2.6.4 Štatistické techniky detekcie anomálií

Základný princíp, na ktorom fungujú každé štatistické techniky detekcie anomálií je, že kde normálne dáta sa vyskytujú vo vysoko pravdepodobnostných regiónoch stochaistického modelu a anomálie zase v nízkych pravdepodobnostných regiónoch tohto modelu. Štatistické techniky určia štatistický model (zvyčajne pre normálne správanie) a potom aplikujú štatistický interferenčný test, ktorý určí, či ďalšie dátové inštancie patria do tohto modelu alebo nie. Dátové inštancie, ktoré majú nízku pravdepodobnosť vygenerovania sa do naučeného modelu, na základe aplikovaného štatistického testu, sú považované za anomálne. Existujú dve základné techniky, *parametrické a neparametrické* [5].

Parametrické techniky

Parametrické techniky predpokladajú, že normálne údaje sú generované parametrickým rozdelením s parametrami θ a funkciou hustoty pravdepodobnosti $f(x, \theta)$, kde 'x' je dátová inštancia. Skóre anomálie testovanej inštancie 'x' je inverzná funkcia funkcie hustoty pravdepodobnosti, $f(x, \theta)$. Parametre θ sú odhadované z dostupných údajov. Alternatívne môže byť vykonaný štatistický test hypotéz. Nulová hypotéza (H_0) pre takéto testy je, že

inštancia dát x bola vygenerovaná pomocou odhadovanej distribúcie (s parametrami θ). Ak štatistický test zamietne H_0, 'x' je vyhlásená za anomáliu. Test štatistickej hypotézy je spojený so štatistikou testu, ktorú možno použiť na získanie pravdepodobnostného skóre anomálie pre inštanciu dát 'x'. Na základe predpokladaného typu distribúcie môžu byť parametrické techniky ďalej kategorizované :

- Založené na Gaussovom modeli: Tieto techniky predpokladajú, že sú dáta generované z Gaussovej distribúcie. Parametre tejto distribúcie vychádzajú z Maximum Likelihood Estimates (MLE). Vzdialenosť dátových inštancií od určeného priemeru je skóre anomálie pre dané dátové inštancie. Na stanovenie anomálií sa použije prahová (threshold) hodnota anomálie. Rôzne techniky vypočítajú vzdialenosť od priemeru a prahovú hodnotu rôznymi spôsobmi.

- Založené na regresnom modeli: Základná technika detekcie anomálií založená na regresnom modeli pozostáva z dvoch krokov. V prvom kroku sa na dáta použije regresný model. V druhom kroku sa pre každý test, na stanovenie skóre anomálie, použije *residual* (zostatok) pre testovaný prípad. Zostatok je taká časť dátovej inštancie, ktorá nie je popísaná v regresnom modeli.

- Založené na zmesi parametrických techník distribucií: Takéto techniky používajú na modelovanie údajov zmes parametrických štatistických distribúcií. Techniky v tejto kategórií môžu byť zoskupené do dvoch podkategórií. Prvá podkategória techník modeluje normálne dátové inštancie a anomálie ako samostatné parametrické distribúcie, zatiaľ čo druhá podkategória techník modeluje iba bežné inštancie ako zmes parametrických distribúcií [5].

Neparametrické techniky

Techniky detekcie anomálií v tejto kategórii využívajú neparametrické štatistické modely, takže štruktúra modelu nie je apriori definovaná, ale je určená z dátových inštancií. Takéto techniky zvyčajne vytvárajú menej predpokladov týkajúcich sa údajov, napríklad hladkosť hustoty v porovnaní s parametrickými technikami.

- Založený na histograme: Základná technika detekcie anomálií, založená na histograme pre jednorozmerné (majúce jednu vlastnosť) údaje, pozostáva z dvoch krokov. Prvý krok spočíva v zostavení histogramu na základe rôznych hodnôt práve z tejto vlastnosti v tréningových dátach. V druhom kroku technika skontroluje, či testovacia inštancia spadá do ktorejkoľvek z košov (bin) histogramu. Ak áno, testovacia inštancia je normálna, inak je anomálna. Variant základnej techniky založenej na histograme je priradiť skóre anomálie každej inštancii testu na základe výšky (frekvencie) koša, v ktorom sa nachádza. Pre detekciu anomálií je kľúčová veľkosť koša použitého pri zostavovaní histogramu. Ak sú koše malé, veľa normálnych testovacích prípadov bude spadať do prázdnych alebo unikátnych košov, čo bude mať za následok vysokú mieru falošných poplachov. Ak sú koše veľké, veľa anomálnych testovacích inštancií bude spadať do častých košoch, čo vedie k vysokej miere falošne negatívnych výsledkov. Preto je kľúčová výzva pre techniky založených na histograme určenia optimálnej veľkosti košov, ktorá udržuje nízku mieru falošných poplachov a nízku mieru falošných negatívov.

- Založené na funkciach kernela (jadra): Technika detekovania anomálií založená na funkciach kernela zahŕňa použitie funkcií kernelu na aproximáciu skutočnej hustoty.

Sú podobné parametrickým metódam opísaných vyššie. Jediným rozdielom je použitá technika odhadu hustoty.

Výpočtová zložitosť techník zisťovania štatistických anomálií závisí od povahy požadovaného štatistického modelu, ktorý zodpovedá dátovým inštanciam. Vybranie vhodných jednotlivých parametrických rozdelení z exponenciálnej triedy, napríklad Gaussian, Poisson, Multinomial atď., sú zvyčajne lineárne z hľadiska veľkosti údajov, ako aj počtu atribútov. Prispôsobenie zložitých distribúcií (napríklad zmiešané modely, HMM atď.) pomocou techník iteračného odhadu, ako je napríklad maximalizácia očakávaní (EM), sú tiež zvyčajne lineárne pre jednotlivé iterácie, aj keď môžu konvergovať pomaly v závislosti od problému a/alebo kritéria konvergencie. Techniky založené na kerneloch môžu mať potenciálne kvadratickú časovú zložitosť z hľadiska veľkosti údajov.
Výhody štatistických techník sú nasledovné:

- Skóre anomálnej dátovej inštancie poskytnuté štatistickou technikou je spojené so spoľahlivosťou intervalu, ktorý je možné použiť ako ďalšiu informáciu pri rozhodovaní o ktorejkoľvek inštancii testu.
- Ak je krok odhadu distribúcie odolný voči anomáliám v dátach, štatistické techniky môžu pracovať v móde bez dohľadu.

Medzi nevýhody štatistických techník patria:

- Kľúčovou nevýhodou štatistických metód je, že sa spoliehajú na predpoklad, že sú údaje generované z konkrétnej distribúcie. Tento predpoklad často neplatí, najmä pre viac-dimenzionálne dátove inštancie.
- Zostrojenie hypotéznych testov na zložité distribúcie, ktoré sú potrebné na to, aby vyhovovali viac-dimenzonálnej dátovej sade, nie sú triviálne.
- Techniky založené na histograme sa implementujú pomerne ľahko, ale kľúčovým nedostatkom týchto techník pre údaje s viac premennými je, že nie sú schopné zachytiť interakcie medzi rôznymi atribútmi. Anomálie môžu mať atribút hodnoty, ktoré sú pre jednotlivé premenné veľmi časté, ale ktorých kombinácia je veľmi zriedkavá. Technika založená na histograme by však nebola schopná detekovať takéto anomálie [5].

2.6.5 Informačno-teoretické techniky detekcie anomálií

Informačno-teoretické techniky detekcie anomálií fungujú na predpoklade, že anomálne inštancie sa odlišujú od normálových v ich informačnom obsahu. Základná technika zahŕňa duálnu optimalizáciu, aby sa minimalizovala veľkosť podmnožiny a maximalizáciu zníženia zložitosti dátových inštancií. Vyčerpávajúci prístup, pri ktorom sa zvažuje každá možná podmnožina dátových inštancií, má exponenciálnu časovú zložitosť. Tieto techniky sa tiež používaju v takých dátových množinách, kde sú jednotlivé dátové inštancie prirodzene usporiadané, napríklad sekvenčné alebo priestorové dáta. V týchto prípadoch sú jednotlivé inštancie rozdelené do sub-štruktúr a informačno-teoretické techniky detekcie anomálií hľadajú v týchto sub-štruktúrach.
Výhody informačno-teoretických techník sú [5]:

- Môžu operovať v móde bez učiteľa.
- Nerobia predpoklady o štatistickej distribucií dátovej množiny.

Nevýhody informačno-teoretických techník sú:

- Výkon týchto techník veľmi závisí od voľby informačno-teoretickej mierky. Takéto opatrenia často dokážu zistiť prítomnosť anomálií, iba ak je v množine dát prítomný významne veľký počet anomálií.
- Techniky aplikované na sekvencie alebo priestorové dáta sa spoliehajú na veľkosť subštruktúry, ktorá je často náročná na získanie.

2.6.6 Spektrálne techniky detekcie anomálií

Spektrálne techniky detekcie anomálií sú založené na tom, že aproximujú dáta použitím kombinácií atribútov, ktoré zachytávajú rozsah rozličnosti dát. Tieto techniky ďalej pracujú s tým, že dáta môžu byť redukované na jednoduchšie dimenzie, kde sa ľahšie rozozná normálna dátová inštancia od anomálnej. Výhodami týchto metód je, že fungujú v móde bez učiteľa, ale majú typicky veľkú výpočtovú náročnosť [5].

Kapitola 3

Umelá neurónová sieť

3.1 Čo je umelá neurónová sieť

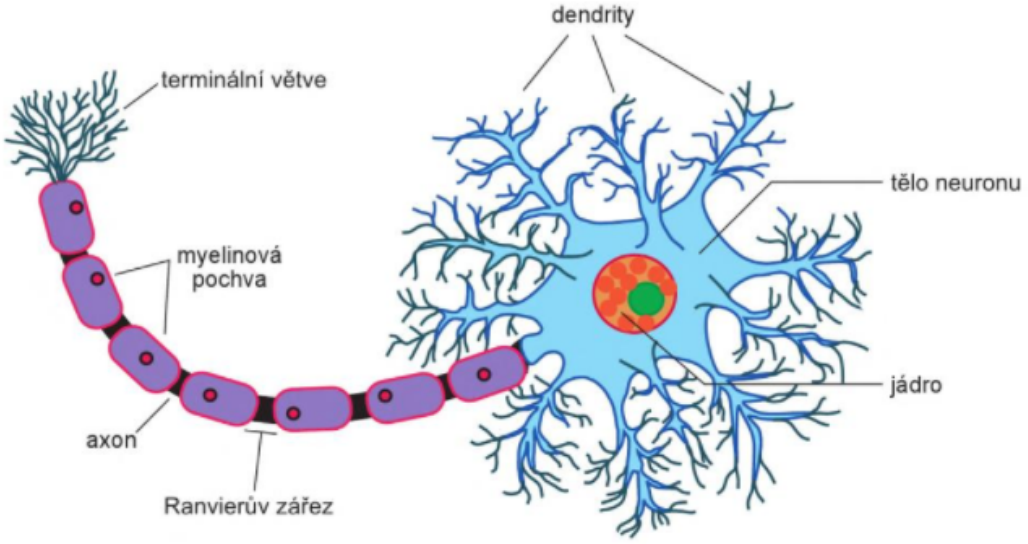
Umelá neurónová sieť je, ako indikuje jej názov, výpočtová sieť, ktorá sa pokúša simulovať proces rozhodnutí v sieťach nervovej bunky (neuróna) biologického centrálneho nervového systému. Poznatky o koncepte tejto siete si vyberá z neurofyziologických poznatkov biologických neurónov a sietí, ako sú práve biologické neuróny. Tým sa líši od bežných (digitálnych či analógových) výpočtových strojov, ktoré slúžia na nahradenie, vylepšenie alebo zrýchlenie výkonu ľudského mozgu bez ohľadu na organizáciu výpočtových prvkov a ich sietí. Musíme zdôrazniť, že táto simulácia neurónových sietí len veľmi hrubo kopíruje reálnu neurónovú sieť. Prečo vnímame umelú neurónovú sieť viac ako len nejaké cvičenie v simulácií? Keďže to, čo dokáže spraviť umelá neurónová sieť, dokáže spraviť aj obvyklý digitálny počítač; aspoň teda výpočtovo [11].

Odpoveď na túto otázku pochádza najmä z dvoch hlavných aspektov. Umelá neurónová sieť, vytvorená simuláciou biologickej neurónovej siete, je nová a neznáma počítačová architektúra s novými, neznámymi algoritmizačnými technikami v porovnaní s obvyklými počítačmi. Tieto siete dokážu použitím veľmi jednoduchých výpočtových operácií (sčítavanie, násobenie a základné logické operácie) vyriešiť komplexné, matematicky zle-definované problémy, nelineárne problémy alebo stochaistické problémy. Obvyklý algoritmus využíva množiny operácií, ktoré aplikuje na daný problém. Vďaka tomu budú umelé neurónové siete výpočtovo a algoritmicky veľmi jednoduché a bude mať samo-organizujúcu funkciu, aby dokázali používať a uchovať tieto vlastnosti pre väčší rozsah problémov [11].

3.2 Biologický a umelý neurón

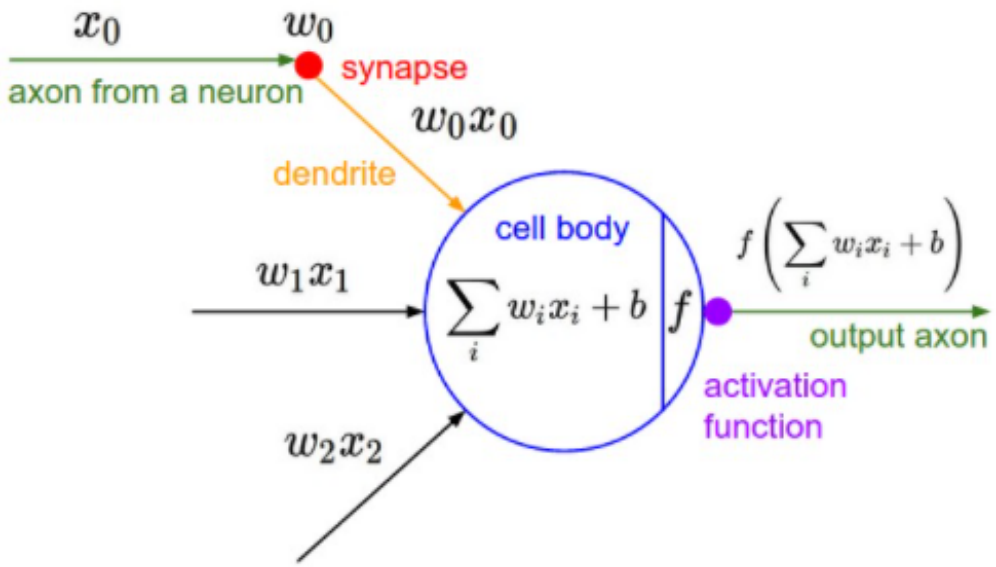
Činnosť nervovej sústavy je podmienená stavbou a funkciou jednotlivých nervových buniek a ich vzájomných vzťahov. V centrálnej nervovej sústave (CNS) vytvárajú nervové bunky komplikovanú a vzájomne mnohopočetnú prepojenú sieť, v ktorej sú ako z funkčného, tak aj z morfologického hľadiska, v úzkom spojení gliové elementy. Pojem neurón zahŕňa telo nervovej bunky aj s ich výbežkami, ako na obrázku 3.1 [21].

Neurón môžeme veľmi jednoduchým spôsobom, popísať ako jednotku, ktorá prijíma signály výbežkami zvanými *dendrity*, ktoré sú spracované v jadre neurónu a výstup ide cez výbežok *axón*. Dendritov má neurón typicky viac, bývajú kratšie a bohato sa vetvia na rozdiel od axónu, ktorý vedie ďalej od tela neurónu. Tento axón sa pripája na dendrity ďalších neurónov. Neurón sa aktivuje a pošle signál cez axón do ďalších neurónov, pokiaľ jeho den-



Obr. 3.1: Schéma biologického neurónu. Prevzaté z [21].

drity dosiahnu, kombináciou vstupných parametrov, aktivačný potenciál [21]. V skutočnosti je tento proces oveľa komplikovanejší. Na obrázku 3.2 vidíme matematický, zjednodušený



Obr. 3.2: Matematická schéma biologického neurónu. Prevzaté z [9].

koncept biologického neurónu. Signál prechádza z axónu druhého neurónu (x_0), multiplikatívne interaguje ($w_0 x_0$) s dendritami druhého neurónu podľa váhy synapsie (spojenia) (w_0). Koncept váhy týchto synapsií (označené w) sa dokážu učiť a kontrolujú silu influencie a jeho smeru (pozitívna váha alebo negatívna váha) jedného neurónu na druhý. Posledným parametrom tohto modelu je bias b. Slúži na posun dát, aby sa lepšie hodila do aktivačnej funkcie. V tomto základnom modeli, dendrity prenášajú signál do jadra neurónu, kde sú sčítané. Ak táto finálna suma je nad určeným thresholdom (prahová hranica), neurón sa

aktivuje a vyšle signál po svojom axóne. Vo výpočtovom modeli predpokladáme, že nezáleží na presnom načasovaní aktivácie a že komunikáciu sprostredkuváva iba frekvencia aktivácie. Na základe tejto frekvencie modelujeme aktiváciu neurónu s aktivačnou funkciou 'f', ktorá predstavuje frekvenciu aktivácií pozdĺž axónu. Z historického hľadiska je bežnou voľbou aktivačnej funkcie sigmoidná funkcia δ, pretože vyžaduje vstup so skutočnou hodnotou (váha signálu po súčte) a stláča ho do rozmedzí od 0 do 1 [9].

3.3 Aktivačné funkcie umelého neurónu

Umelé neuróny sa najviac rozlišujú aktivačnou funkciou. Ak sa táto aktivačná funkcia nepoužije v neurónovej sieti, výsledný signál je iba jednoduchá, lineárna funkcia, ktorá má polynomiálny stupeň jeden. Aj keď je lineárna rovnica výpočtovo jednoduchá, jej komplexita je limitovaná a nemá možnosť sa učiť alebo rozoznať komplexnejšie mapovanie z dát. Lineárne neurónové siete bez aktivačnej funkcie sa chovajú ako lineárne regresívne modely s limitovanou výkonnosťou. Tieto modely sú nedostatočné pre komplexnejšie výpočty, preto sa musia používať aktivačné funkcie. Medzi tieto aktivačné funkcie patrí [24]:

- binárna kroková aktivačná funkcia,
- lineárna aktivačná funkcia,
- sigmoidná aktivačná funkcia,
- tan_h aktivačná funkcia,
- ReLU aktivačná funkcia,
- leaky ReLU aktivačná funkcia,
- parametrizovaná ReLU aktivačná funkcia,
- exponenciálna ReLU aktivačná funkcia,
- SoftMax aktivačná funkcia .

3.3.1 Binárna kroková aktivačná funkcia

Binárna kroková funkcia je základná a najprimitívnejšia aktivačná funkcia, ktorá existuje. Podarí sa ju naimplementovať iba pomocou if-else príkazov v Pythone. Nevýhodou je, že binárna kroková funkcia využíva binárny klasifikátor, preto sa nemôže použiť v prípade multi-triednych klasifikácií. Gradient tejto funkcie je 0, čo môže spôsobiť problém v spätnom propagačnom kroku. Matematicky sa definuje binárna kroková funkcia nasledovne [24]:

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

3.3.2 Lineárna aktivačná funkcia

Lineárna aktivačná funkcia je priamo úmerná k vstupným dátam. Najväčšou nevýhodou binárnej krokovej aktivačnej funkcie je, že má nulový gradient, lebo x nemá žiaden multiplikátor. Toto sa dá odstrániť práve použitím lineárne aktivačnej funkcie, ktorá je matematicky definovaná nasledovne:

$$F(x) = ax$$

Veľkosť premennej a môže byť ľubovoľná a záleží na užívateľovi. V skutočnosti nie je veľkou výhodou používanie lineárnej aktivačnej funkcie, lebo neurónová sieť sa nebude zlepšovať na základe chyby kvôli rovnakej hodnote gradientu pre každú iteráciu. Takisto nie je schopná zachytávať komplexné vzorce v dátach. Preto sa lineárne aktivačné funkcie využívajú tam, kde je ľahká interpretovateľnosť a pre jednoduchšie zadania [24].

3.3.3 Sigmoidná aktivačná funkcia

Sigmoidná aktivačná funkcia je široko využívateľná, keďže je to nelineárna funkcia. Sigmodiná funkcia transformuje vstupné hodnoty do oboru hodnôt 0 až 1. Prakticky sa veľké negatívne hodnoty zobrazia v 0 a veľké pozitívne hodnoty zase v 1 [9]. Sigmoidná funkcia sa matematicky definuje ako [24]:

$$F(x) = 1/e^{-x}$$

Táto funkcia sa čoraz menej používa kvôli dvom hlavným chybám:

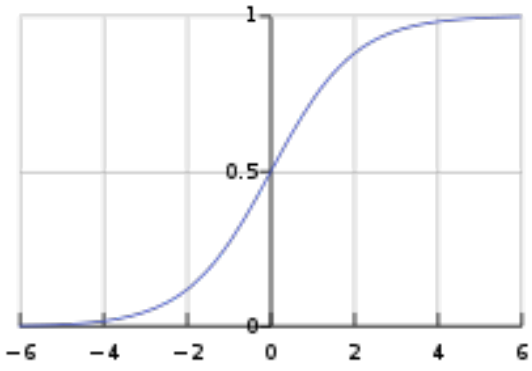
- Funkcia saturuje v hodnotách 0 a 1, kde je gradient skoro nulový.
- Sigmoidné funkcie nie sú symetrické okolo 0, čo znamená, že znaky všetkých výstupných hodnôt neurónu sú rovnaké. Tento problém môže byť napraviteľný úpravami sigmoidnej funkcie [9].

3.3.4 TANH aktivačná funkcia

Táto funkcia je *hyperbolická tangent funkcia*, ktorá je podobná sigmoidnej funkcii, ale je symetrická podľa počiatku. Výsledky rozdielnych hodnôt výstupu z predošlej vrstvy budú dané ako vstupné hodnoty do ďalšej. Matematicky je definovaná ako:

$$f(x) = 2sigmoid(2x) - 1$$

Funkcia tan_h je súvislá a diferenciovateľná, jej hodnoty sú v obore hodnôt od -1 až 1. Táto funkcia je v praxi viac preferovaná oproti sigmoid funkcii, keďže má gradienty, ktoré nie sú obmedzené v určitom smere a je symetrická podľa počiatku [24]. Na obrázku 3.3 môžeme vidieť graf tejto funkcie.



Obr. 3.3: Graf hypoerbolickej tangent funkcie. Prevzaté z[1].

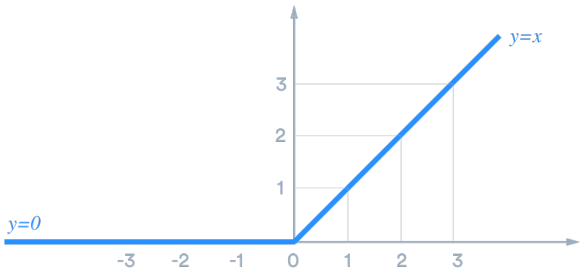
[1] Zdroj:https://en.wikipedia.org/wiki/Activation_function

3.3.5 ReLU aktivačná funkcia

ReLU, rectified linear unit, je nelineárna aktivačná funkcia, ktorá je široko využívaná v umelej neurónovej sieti. Výhodou použitia funkcie ReLU je, že všetky neuróny nie sú aktivované súčasne. To znamená, že neurón bude deaktivovaný iba keď výstup lineárnej transformácie je 0. Matematicky sa zapisuje ako:

$$f(x) = max(0, x)$$

ReLU je efektívnejšia ako ďalšie funkcie, pretože nie sú aktivované všetky neuróny naraz. Využíva sa pri spracovávaní obrazu, keďže ignoruje záporné hodnoty, čo je žiadúce pri práci s hodnotami pixlov [24]. Na obrázku 3.4 vidíme graf tejto aktivačnej funkcie.



Obr. 3.4: Graf ReLU funkcie. Prevzaté z[2].

3.3.6 Varianty ReLU aktivačných funkcií

Leaky ReLU je improvizaná ReLU funkcia, kde pre všetky negatívne hodnoty x je táto funkcia definovaná ako 'ax', kde a je extrémne malý parameter (0.01). Parametrizovaná ReLU sa líši od Leaky ReLU v parametri 'a', kde je nahradená ľubovoľným číslom. Matematicky je definovaná ako:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ ax & \text{if } x < 0; a = 0.01 \text{ pre Leaky ReLU funkciu} \end{cases}$$

Pri parametrizovanej funkcii sa tento parameter dokáže učiť.

Exponenciálna aktivačná funkcia využíva logaritmickú krivku pre definovanie záporných hodnôt; jej graf vidíme na obrázku 3.5 [24]:

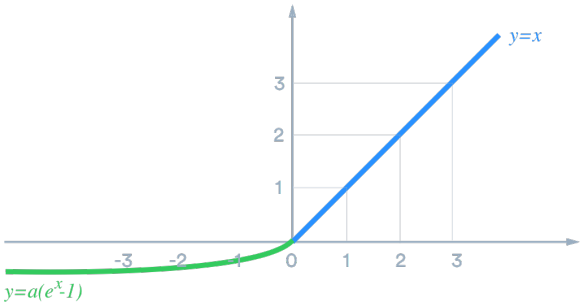
$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ a(e^x - 1)x & \text{if } x < 0 \end{cases}$$

3.3.7 SOFTMAX aktivačná funkcia

SOFTMAX aktivačná funkcia je kombinácia viacerých sigmoidných funkcií. Sigmoidná funkcia, ako už bolo spomenuté, vracia hodnoty v intervale (0,1). Softmax funkcia na rozdiel od sigmoid funkcie sa môže použiť pre multi-triedne klasifikačné problémy. Využíva sa často v poslednej vrstve. Matematicky sa funkcia zapíše [24]:

$$\delta(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} for j = 1, ...K$$

[2]Zdroj:https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7
[3]Zdroj: https://machinelearningmastery.com/



Obr. 3.5: Graf exponenciálnej ReLU funkcie. Prevzaté z [3].

3.4 Architektúra neurónovej siete

Neurónové siete su modelované ako kolekcie umelých neurónov, ktoré sú spojené do grafu. Dá sa povedať, že výstupy nejakých neurónov sa môžu stať vstupmi iných neurónov. Neurónové siete sú často rozdelené do distinktívnych vrstiev (layers). Pre bežnú neurónovú sieť, najbežnejší typ vrstvy je *plne-spojená vrstva*, v ktorej všetky neuróny z každej vrstvy sú spojené s neurónmi zo susednej vrstvy, ale tieto neuróny nie sú spojené v rámci jednej vrstvy. Na obrázku 3.6 vidíme necyklickú neurónovú sieť (feedforward neural network), tvoria ju 3 vstupné uzly, ktoré nie sú plne-spojené s druhou vrstvou, ktorá je skrytá. Výstupná vrstva má 2 uzly. Na obrázku 3.7 vidíme schému cyklickej neurónovej siete [15].

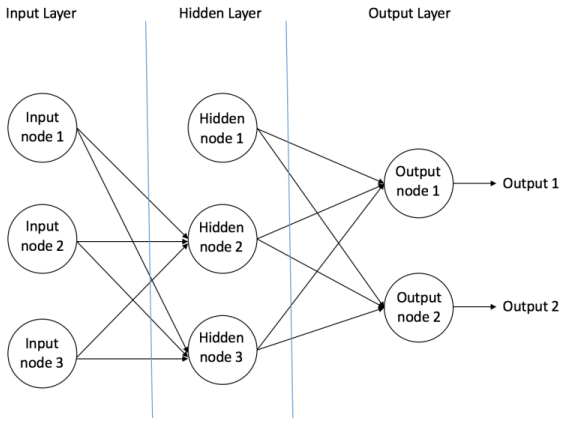
Uzly v umelých neurónových sieťach sa delia na 3 typy:

- Input (vstupné) uzly: tieto uzly v sebe nesú informáciu z vonkajšieho sveta do neurónovej siete a spoločne sú označené ako *Input Layer (vstupná vrstva)*. Žiadne vypočítavanie sa v týchto uzloch nevykonáva, iba odovzdajú informáciu ďalej do skrytých uzlov.
- Hidden (skryté) uzly: tieto uzly nemajú žiadne priame spojenie s vonkajším svetom, preto sa nazývajú skryté. Tieto uzly vykonávajú výpočet a transformujú informáciu zo vstupných uzlov do výstupných uzlov. Kolekcia týchto uzlov sa nazýva *Hidden Layer (skrýta vrstva)*. Schémy môžu mať nula týchto vrstiev, jeden alebo aj viac.
- Output (výstupné) uzly: Tieto uzly majú na starosti vypočítanie a transformáciu dát z neurónovej sieti do vonkajšieho sveta. Kolekcia týchto uzlov sa nazýva *Output Layer (výstupná vrstva)* [15].

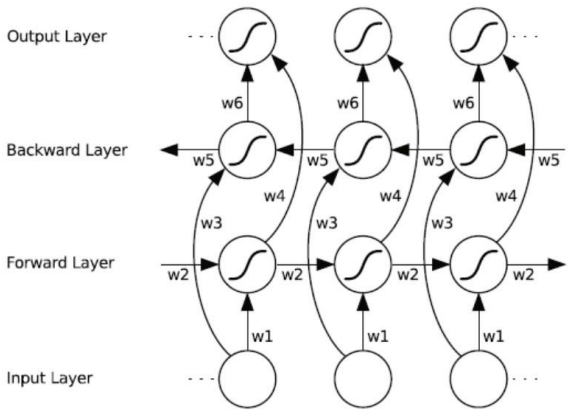
3.5 Učenie sa umelej neurónovej siete

Učenie sa umelej neurónovej siete sa dá pochopiť ako adaptácia tejto siete, aby lepšie zvládala problém na základe obhliadajúcich dát. V kapitole 2.5 som poukázal na tri druhy učenia - s dohľadom, semi-dohľadom alebo bez dohľadu. Toto učenie upravuje váhy jednotlivých spojení medzi uzlami na zlepšenie výsledku. Zlepšenie sa robí najmä pomocou minimalizácie chybovosti. To však neznamená, že po úspešnom učení je táto chybovosť nulová. Ak je chybovosť veľmi vysoká, typicky sa musí zmeniť model neurónovej siete.

[4] https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/
[5] https://www.programmersought.com/article/6338898511/



Obr. 3.6: Ukážka necyklickej neurónovej siete. Prevzaté z [4].



Obr. 3.7: Ukážka cyklickej neurónovej siete. Prevzaté z [5].

Kapitola 4

Návrh riešenia a implementácia

4.1 Vstupné dáta

Jeden z najdôležitejších aspektov pri detekovaní anomálií sú vstupné dáta. Ja som si ako vstupné dáta vybral dáta od kyberbezpečnostnej skupiny *Stratosphereips* na vysokej škole České Vysoké Učení Technické v Prahe. Táto skupina je zameraná práve na kyberbezpečnosť, strojové učenie a pomáhanie druhým [10].

Technológia Stratosphere Intrusion Prevention System (Systém na prevenciu voči útokom) sa učí na modeloch vytvorených z reálnej zachytenej sieťovej prevádzky (ďalej len pcap), ktorá obsahuje škodlivé pakety. Práve používaním a študovaním týchto pcapov dokážu zaručiť, že jednotlivé vytvorené modely sú presné a ich úspešnosť je reálna. Ďalší projekt vytvorený v tomto tíme, *Malware Capture Facility Project*, sa stará o konštatné monitorovanie hrozieb na internete a vytváranie škodlivých vzoriek. Tieto vzorky sa následne spustia na zariadeniach, ktoré zachýtavajú sieťovú prevádzku [10].

Algoritmy strojového učenia potrebujú dáta na učenie, testovanie a verifikovanie, aby zistili presnú výkonnosť na reálnych dátach. Mať dobré datasety je veľmi potrebné v sfére sieťovej bezpečnosti, keďže dáta v sieťach sú *nekonečné*, meniace sa a rozličné. Preto na týchto stránkach nájdeme pcapy, ktoré zachytávajú normálnu prevádzku a pcapy, ktoré zachytávajú normálnu a škodlivú prevádzku [10].

Jednotlivé pcapy, či už tie, ktoré zachytávajú normálnu alebo normálnu a škodlivú prevádzku, potrebujem analyzovať a vytvoriť rozumný, použiteľný dataset. Pre zistenie malwarovej činnosti som sa rozhodol použiť technólogiu *JA3 fingerprints*.

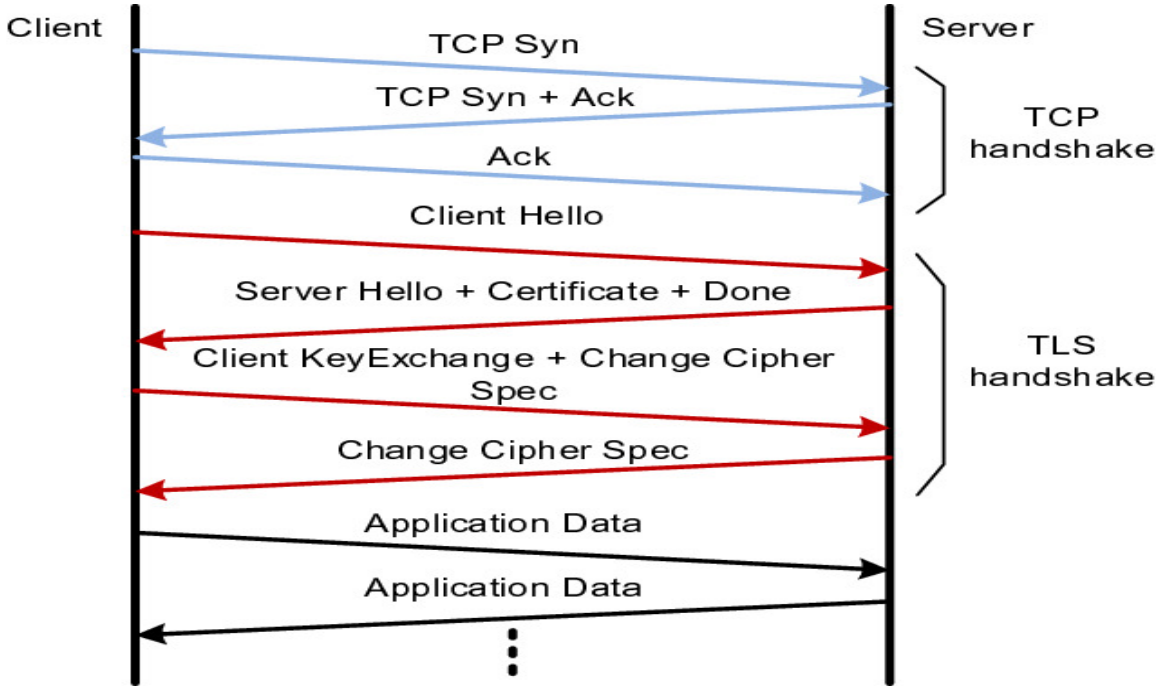
4.1.1 Ako funguje JA3

Transport Layer Security(TLS) [7] je prenosový protokol, ktorý pracuje na vrchu TCP vrstvy, kde poskytuje bezpečnosť a integritu dát pre aplikácie, ktoré medzi sebou komunikujú. Protokol je rozdelený na dve časti:

- TLS Handshake Protocol,
- TLS Record Protocol.

TLS Handshake Protocol je zodpovedný za bezpečnostné parametre komunikácie; napríklad SSL verzia, metódy na výmenu kľúčov, zašifrovanie dát, autentizáciu, integritu dát, vlastnosti bezpečnostného kanálu. TLS handshake komunikácia nie je zašifrovaná.

TLS Record Protocol zapúzdruje vysoko-úrovňové dáta a stará sa o posielanie zašifrovaných paketov [20].



Obr. 4.1: Vytvorenie TLS spojenia. Prevzaté z [20].

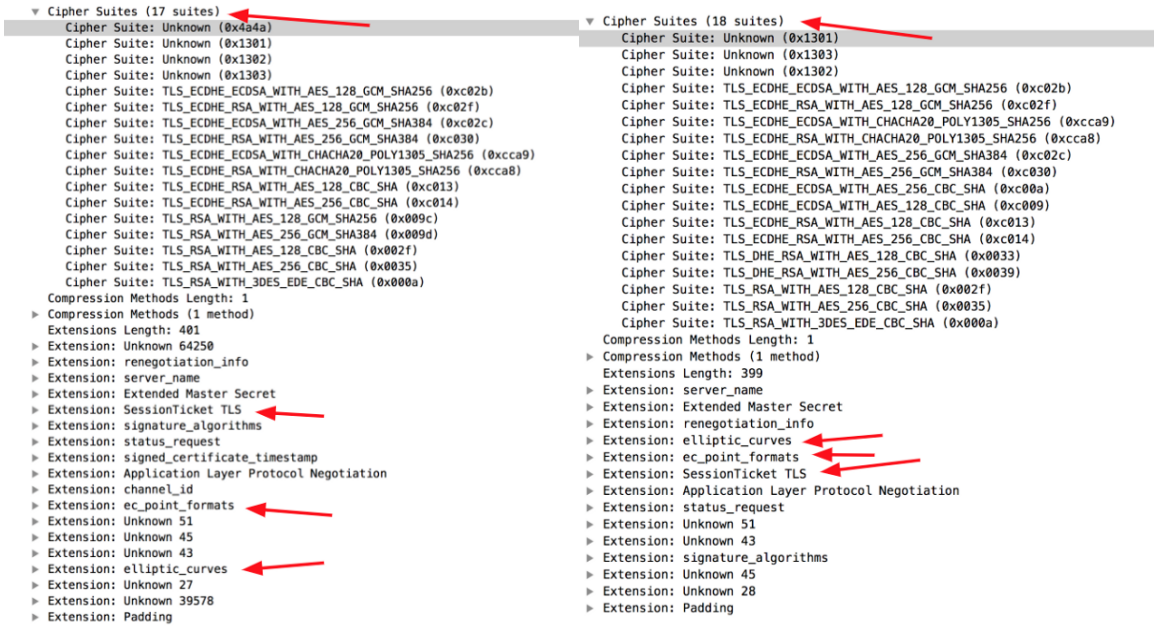
Na obrázku 4.1 vidíme, že pred TLS handshakom je najskôr TCP 3-way handshake. TCP vytvára také pripojenie, kde je potrebné vytvoriť spojenie ešte pred odosielaním dát. Najskôr si prijímajúca a odosielajúca strana zosychronizujú ich sekvenčné čísla, aby sa ich prenos mohol začať. Synchronizácia sa uskutočňuje výmenou segmentov, ktoré nesú synchronizačný bit (označovaný ako SYN). Synchronizácia vyžaduje, aby každá strana dostala potvrdenie tejto výmeny - acknowledgement (ACK). Celý tento mechanizmus sa realizuje v troch krokoch, preto sa nazýva 3-way handshake [28].

Po úspešnom vytvorení TCP pripojenia 3-way handshakom, TLS začne vytvárať bezpečnostné parametre použitím TLS Clienta (odosielateľa) a Server Hello paketov (prijímateľa). Klientská aplikácia ponúka sadu bezpečnostného zašifrovania a autentinzačné metódy použitím TLS Client Hello paketu. TLS server spracuje tieto vlastnosti komunikácie a pošle späť vlastnosti, ktoré sú podporované na serverovej strane. Server môže k tomu pripojiť serverový certifikát, aby mohol byť overený. Po ustanoveniach všetkých bezpečnostných parametrov, aplikačné dáta sa zapúzdrujú pomocou TLS Record Protocol a sú poslané po sieti [20].

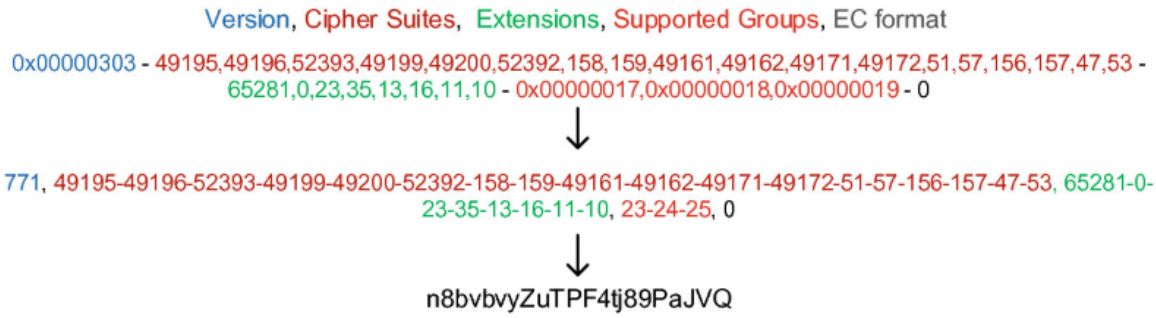
Väčšina TLS fingerprint metód používa práve prvý paket poslaný klientom: Client Hello paket. Client Hello paket obsahuje TLS konfigurácie klientskej aplikácie, ktoré závisia od použitej TLS knižnice a operačného systému. Ako jeden príklad je výskum *Network-Based HTTPS Client Identification Using SSL/TLS Fingerprinting*, kde okrem základných infromácií, akými je verzia TLS, sa predovšetkým zamerali na zdieľanie starších protokolov [13]. *JA3* fingerprint je vyrobený *MD5* hashom z piatich TLS handshake polí:

- TLS handshake version - TLS verzia aplikácie.
- Cipher suites - sada algoritmov, ktoré zvyčajne obsahujú algoritmy ako je výmena kľúčov alebo message authentication code.

- Extensions - rozšírenia, ktoré boli prvýkrát predstavené v *RFC 3456* a neskôr sa stali súčasťou TLS. Klient oznámi, ktoré rozšírenia podporuje a server potom vyberie, ktoré z týchto rozšírení sa použijú [27].
- Supported Groupes(predtým Elliptic Curve) - algoritmy, ktoré šifrujú kľúče [12].
- Elliptic Curve point format [20].



Obr. 4.2: Vybrané polia zachytené v programe *Wireshark*. Prevzaté z [3].



Obr. 4.3: Vytvorenie JA3 hashu. Prevzaté z [20].

Vytvorenie JA3 fingerprintu pozostáva z troch krokov:

i Extrahovanie vybraných polí z TLS Hello paketu.

ii Konkatenácia extrahovaných dát v decimálnych hodnotách; jednotlivé hodnoty polí sú oddelené pomlčkou a polia sú oddelené čiarkou.

iii Aplikovanie MD5 hashovacieho algoritmu na vytvorený string.

Výsledok je 32-bitový string v hexadecimálnej podobe. Na rozdiel od *nmap* alebo fingerprint metódy webového prehliadača, JA3 fingerprint využíva viac pasívny prístup. Proces vytvorenia TLS fingerprintov je rýchly, pretože pracuje iba s TLS hlavičkou. Bežne sieťové monitorovanie a IDS nástroje implementujú extrakciu TLS parametrov pre analýzu zašifrovanej sieťovej prevádzky. Aplikácia TLS fingerprintov na identifikovanie sieťových aplikácií vyžaduje, aby TLS fingerprint hodnoty boli jedinečné, presné a stále. Jeden z aspektov, ktoré limitujú spoľahlivosť TLS fingerprintov sú *randomizované hodnoty v poli TLS Extensions* [20] Vytvorenie tohto hashu je inšpirované autorom *Adelom Karimim*[1].

Náhodné hodnoty v poli TLS Extensions

V roku 2016 začal Google generovať randomizované rozšírenia a zachovávať rozšíriteľnosť (Generate Random Extensions And Sustain Extensibility - GREASE) hodnôt v TLS. GREASE hodnoty sú náhodne generované čísla *cipher suite, extensions* a *supported groups*, ktoré sa nachádzajú v TLS Hello pakete. Slúžia na prevenciu chybovosti rozšíriteľnosti v TLS ekosystéme. Počas TLS handshake procesu, odpovedajúca strana musí ignorovať neznáme hodnoty. Účastníci, ktorý neignorujú tieto hodnoty, zlyhajú v inter-operácii, čo znamená chybu v implementácii. Práve kvôli tomuto *RFC 8701* pridala GREASE hodnoty ako časť zoznamu cipher suites, extensions a supported groups, na detekovanie neplatných implementácii [20].

4.1.2 Vytorenie datasetu

Dataset vytváram z pcap súborov. Zvolil som si, že škodlivé pakety budem určovať pomocou JA3 fingerprintov. Tieto fingerprinty sa nachádzajú len v TCP paketách, preto som si určil, že budem analyzovať *TCP toky*. Vytvoril som si v *Pythone* DataFrame, ktorý má nasledujúce polia:

- index: určuje poradie TCP toku v pcape,
- duration: celkové trvanie TCP spojenia v pcape,
- srcIp: zdrojová IP adresa,
- srdPort: zdrojový port,
- dstIp: cieľová IP adresa,
- dstPort: cieľový port,
- service: protokol IP vrstvy,
- srcBytes: počet bajtov poslaných zo zdrojovej IP adresy,
- dstBytes: počet bajtov poslaných z cieľovej IP adresy,
- flag: počet paketov, ktoré mali nastavený príznak *TCP Reset*,
- land: 1, ak je spojenie z rovnakého portu,
- urgent: počet paketov, ktoré mali nastavený príznak *TCP Urgent*,

[1] https://github.com/0x4D31/fatt/blob/master/fatt.py

- ja3: Vytvorený JA3 fingerprint,
- ja3Ver: SSL verzia v Client Hello pakete,
- ja3Cipher: hodnoty jednotlivých cipherov oddelené pomlčkou; keďže počet sa môže meniť, nastavil som, že ich je 35; tým pádom som buď doplnil nulami alebo odstránil hodnoty, ktoré pretiekli,
- ja3Extension: hodnoty jednotlivých rozšírení oddelené pomlčkou; počet som nastavil na 25 a upravil; viď predchádzajúci riadok,
- ja3Ec: hodnoty jednotlivých supported groups oddelené pomlčkou; počet som nastavil na 5 a upravil; viď predchádzajúci riadok,
- ja3Ecpf: hodnota eliptic curve point format; počet som nastavil na 2,
- blacklisted: príznak, ktorý určuje, či je tok škodlivý[25].

Všetky polia, až na pole blacklisted, zisťujem analýzou pcap súborov pomocou pyshark knižnice. Nastavenie príznaku určujem na základe zisteného *JA3 fingerprintu*, ktorý porovnávam s databázou škodlivých JA3 fingerprintov. Pri zistení, že JA3 fingerprint sa nachádza v tejto databáze, označím priradený TCP tok ako škodlivý.

Datasety znormalizujem na hodnoty v obore hodnôt [0,1] funkciami z knižnice *scikit-learn*. Trieda *sklearn.preprocessing.MinMaxScaler(**params)* transformuje prvky vyškálovaním každého prvku do daného oboru hodnôt. Transformácia je daná ako:

$$X\_std = (X - X.min(axis = 0))/(X.max(axis = 0) - X.min(axis = 0))$$

$$X\_scaled = X\_std * (max - min) + min; \text{ min,max} = \text{obor hodnôt prvkov}$$

Tranformácia je často daná ako alternatíva nulového priemeru. Následne použijem funkciu *sklearn.preprocessing.MinMaxScaler.fit_transform(X[, y])*, ktorá dáta transformuje a vracia transformované, znormalizované dáta [22].

Vytvorenie blacklist databázy JA3 fingerprintov

Databázu *blacklistdb* som vytvoril programom postgreSQL, kde som si vytvoril tabuľku *ja3*. Dáta, t.j. JA3 fingerprinty, ktoré sú škodlivé, som použil zo švajčiarskej stránky[2], kde sa nachádza *CSV* súbor škodlivých JA3 fingerprintov[3]. Tento súbor obsahuje nasledujúce polia:

- ja3_md5: JA3 fingerprint,
- Firstseen: kedy sa fingerprint prvý raz objavil na sieťach,
- Lastseen: kedy sa fingerprint posledný raz objavil na sieťach,
- Listingreason: dôvod, prečo je škodlivý.

Tabuľka má rovnakú štruktúru, akú má zmienený CSV súbor JA3 fingerprintov. Dáta som do nej naplnil pomocou skriptu napísaneho v jazyku *Python*.

[2] https://abuse.ch/
[3] https://sslbl.abuse.ch/blacklist/ja3_fingerprints.csv

4.2 Použité technólogie

Pre celkovú implementáciu praktickej časti bakalárskej práce som sa rozhodol použiť programovací jazyk *Python*. Python je vysoko-úrovňový programovací jazyk, ktorý je vhodný aj na písanie skriptov. Pre vytvorenie databázy som použil *PostgreSQL*, kde mám vytvorenú tabuľku škodlivých JA3 fingerprintov. Na vkladanie dát používam knižnicu *psycopg2*. Pomocou tejto knižnice sa pripojím do už vytvorenej databázy a vložím potrebné údaje. Načítanie pcapov a analýzu paketov spracúvavam pomocou knižnice *pyshark*, ktorý je obalom pre *tshark*. Pyshark parsuje pakety použitím *Wireshark dissectorov*. Analýzu jednotlivých TCP tokov skontrolujem práve pomocou programu *Wireshark*.

Medzi ďalšie použité knižnice používam *os, hashlib, pandas, re, NumPy, SciPy...*

4.2.1 Umelá inteligencia

Umelú inteligenciu som naimplementoval v knižnici *scikit-learn*. *Scikit-learn* je univerzálna knižnica strojového učenia napísaná v jazyku *Python*. Poskytuje efektívne implementácie najmodernejších algoritmov, ktoré sú prístupné ľuďom, ktorí nemajú skúsenosti so strojovým učením, a sú opakovane používateľné vo všetkých vedeckých disciplínach a aplikačných oblastiach. Využíva tiež interaktivitu a modularitu programovacieho jazyka *Python* na zabezpečenie rýchleho a ľahkého prototypovania. V *scikit-learn* knižnici akceptujú všetky objekty a algoritmy vstupné dáta vo forme dvoj-dimenzionálnych polí vo veľkosti počet inštancií x počet vlastností. Tento prístup robí knižnicu *scikit-learn* všeobecnou a nezávislou na doméne. Objekty tejto knižnice zdieľajú jednotnú sadu metód, ktorá závisí od ich účelu: odhadcovia (estimators) dokážu nájsť vhodný model z dát, prediktory môžu predpovedať nové údaje pomocou modelu a transformátory prevádzajú údaje z jednej reprezentácie na druhú.

- Odhadca (estimator) je rozhranie, ktoré je jadro kniźnice. Implementuje *vhodnú (fit) metódu* pre parametre modelu učenia sa z tréningových údajov. Všetky algoritmy učenia sa, či už v móde s učiteľom alebo bez, sú k dispozícii ako objekty v tomto rozhraní.

- Prediktor je *odhadca* s metódou *predvídať (predict)*, ktorá má na vstupe pole *X_test* a spraví predikciu pre každú inštanciu v datasete. V prípade učenia sa v móde s učiteľom táto metóda typicky vracia predpovedané hodnoty vypočítané z odhadnutého modelu.

- Transformátory modifikujú alebo filtrujú dáta pred vložením do algoritmov strojového učenia. Algoritmy predspracovania, výberu funkcií a redukcie rozmerov sú všetky poskytnuté ako transformátory v knižnici *scikit-learn* [1].

Kapitola 5

Modelovanie a experimentovanie

Všetky inštancie vo vytvorených datasetoch majú označenie, či sú škodlivé alebo nie. Tieto inštancie sú teda klasifikované do dvoch tried: škodlivé alebo normálne. Preto vyberiem algoritmy klasifikácie detekovania v móde s dohľadom. Z dostupných techník som sa rozhodol použiť nasledovné:

- Support vector machine (Stroje s podpornými vektormi).
- K-nearest neighbor algorithm (algoritmus k-najbližších susedov).
- Decision tree classifier (klasifikátor na rozhodovacom strome).
- Multi-layer Perceptron (MLP) classifier (klasifikátor založený na viac-vrstvovom perceptróne).

5.1 Support vector machine

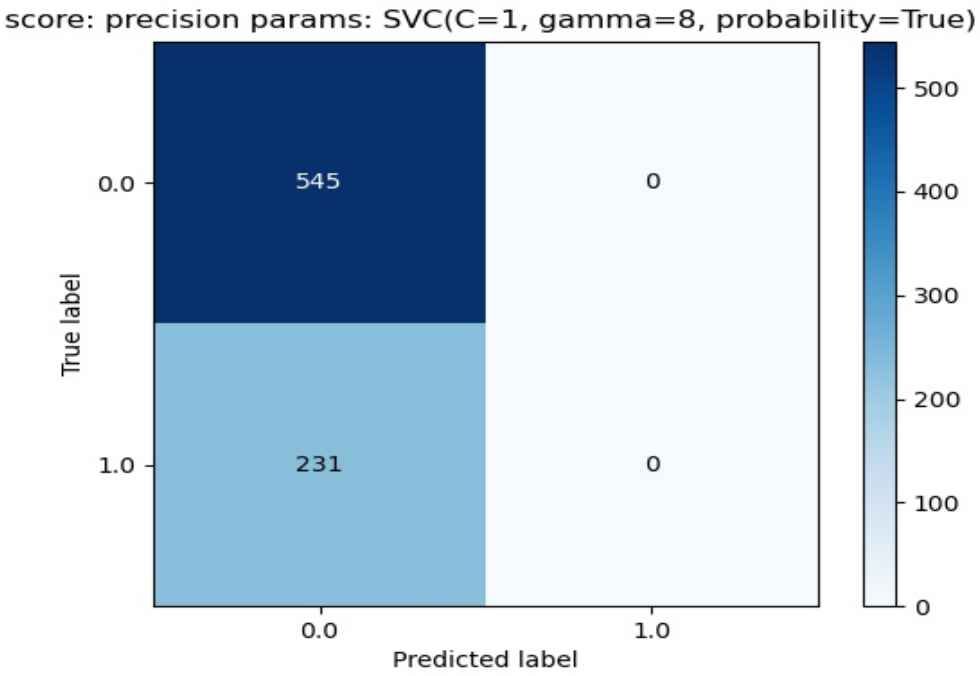
Knižnica *scikit-learn* implementuje support vector machine algoritmus v triede *sklearn.svm.SVC()*. Implementácia je založená na knižnici *libsvm*. Táto trieda má viacero parametrov, ktoré sú potrebné vyladiť:

- C: regularizačný parameter. Sila regularizácie je nepriamo úmerná C,
- kernel {'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}: špecifikuje typ kernelu použitý v algoritme,
- γ: koeficient pre vybrané kernely.

V knižnici *libsvm* autori **Chang et al.** [6] popísujú odporúčaný postup pre vyladenie týchto parametrov, keďže nie je známe pred testovaním, ktore sú najlepšie pre daný problém. Preto je nutné mať postup pre vyladenie týchto parametrov. Cieľ je identifikovať takú dvojicu (C, γ), že klasifikátor dokáže najpresnejšie predikovať neznáme hodnoty - najlepšie aplikovateľné na testovacích dátach. Je veľmi podstatné si uvedomiť, že dosiahnutie veľkej presnosti pri trénovacích dátach nie je prioritou (klasifikátor dokáže presne predikovať tréningové hodnoty, ktoré sú už známe). Bežná stratégia je separácia tréningového datasetu do dvoch častí, kde sa jedna považuje za neznámu. Presnosť predikovania na tejto časti lepšie reflektuje výkon pri klasifikovaní v neznámom, testovacom datasete. Vylepšená verzia tejto procedúry je známa ako *cross-validation (krížová validácia)*.

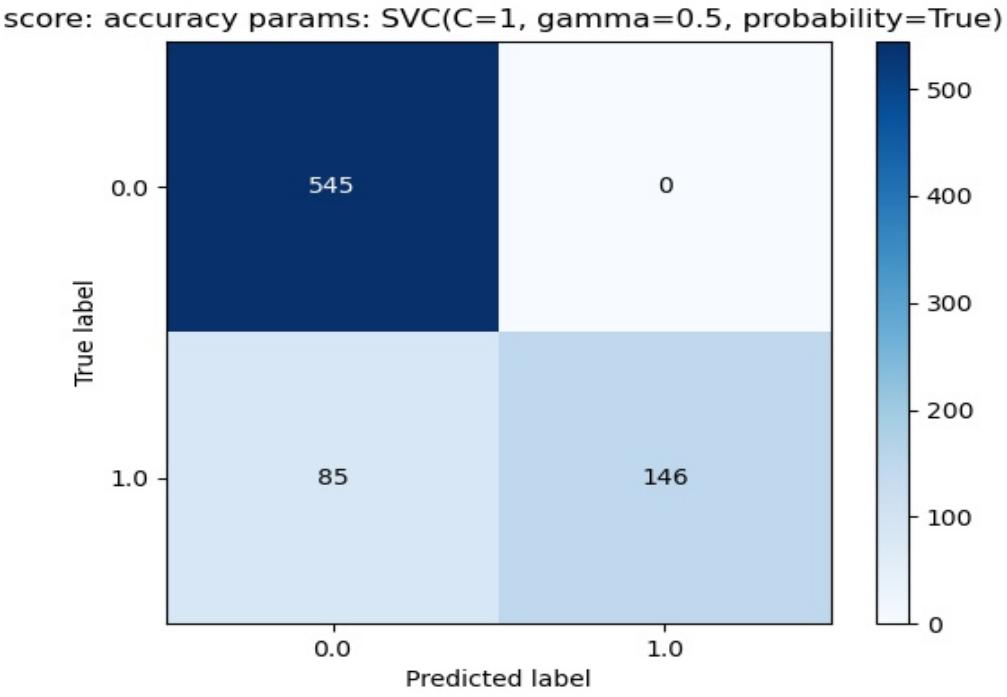
V **n-krížovej validácií** rozdelíme trénovací dataset do n podskupín datasetov rovnakej veľkosti. Jedna podskupina sa postupne testuje pomocou klasifikátora vyškoleného pre zostávajúcu časť n - 1 podmnožiny. Každá inštancia celej tréningovej sady je teda predpovedaná raz, takže presnosť krížovej validácie je percento údajov, ktoré sú správne klasifikované. Krížová validácia odstraňuje problém *overfittingu (model presne predikuje len trénovacie inštancie)*. V [6] je najodporúčanejšia metóda *grid-search (vyhľadávanie v mriežke)* spoločne s krížovou validáciou. Rôzne páry (C, γ) hodnôt sú skúšané a jedna s najlepšou presnosťou krížovej validácie je vybraná. V knižnici *scikit-learn* je táto metóda naimplementovaná triedou *sklearn.model_selection.GridSearchCV(**params)*. Rozhodol som sa spraviť vyhľadávanie na troch kerneloch: linear, rbf a sigmoid, a na hodnotách C: $2^{(-5)}, 2^{(-3)}...2^{(12)}$ a $\gamma = 2^{(-15)}, 2^{(-13)}...2^{(3)}$. Ako posledný parameter scoring som zvolil dve možnosti: *precision a accuracy*. *Sklearn.model_selection.GridSearchCV(**params)* automaticky robí 5-krížovú validáciu pri učení sa. Z dôvodu, že viac párov vyšlo s rovnakou, najlepšou hodnotou, rozhodol som sa všetky tieto páry C a γ s príslušným kernelom otestovať na testovacích datasetoch a najlepší model vybrať pre každú možnosť parametra *scoring*. Na obrázkoch 5.1 a 5.2 sú zobrazené modely s parametrami klasifikátora pre jednotlivé možnosti parametru *scoring*, ktoré majú najväčšiu presnosť trénovacích inštancií a zároveň najlepšie predpokladajú testovacie inštancie.

Na obrázku 5.1 je zobrazená chybová matica SVM algoritmu s parametrom scoring = precision. Parameter kernel je *rbf*, na obrázku nie je znázornený, lebo *scikit-learn* knižnica považuje tento kernel za predvolenú hodnotu. Presnosť na trénovacích inštanciách je 100%, či už na celkovom datasete alebo použitím 5-krížovej validácie. Presnosť na testovacých inštanciách je 70,23%, pri 5-krížovej validácií je priemerná hodnota až 83,77%, ale všetky nesprávne testované inštancie predpovedalo len do jednej triedy, preto je tento model zlý. Trvanie výpočtu je 83,58 sekúnd.



Obr. 5.1: Chybová matica SVM algoritmu s parametrom scoring: precision

Na obrázku 5.2 je zobrazená chybová matica SVM algoritmu s parametrom scoring = accuracy. Presnosť na trénovacích inštanciách je 99,8%, použitím 5-krížovej validácie je priemerná presnosť 98,67%. Presnosť na testovacích inštanciách je 89,05%, pri 5-krížovej validáacií je priemerná hodnota 98,2%. Z chybovej matice vidieť, že všetky nesprávne predpovedané inštancie (85 $\sim$ 11%) sú falošné negatíva. Trvanie algoritmu je 102.45 sekúnd.



Obr. 5.2: Chybová matica SVM algoritmu s parametrom scoring: accuracy

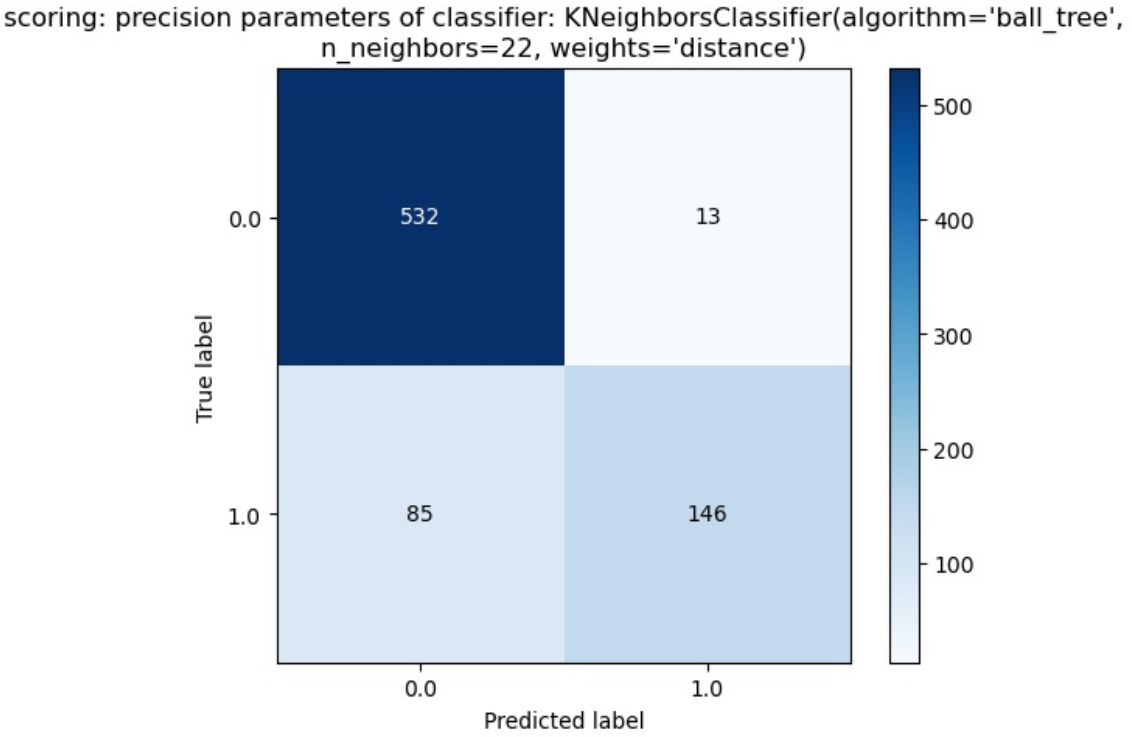
5.2 K-nearest neighbor algorithm

V knižnici *scikit-learn* sa klasifikátor pre algoritmus k-najbližšiích susedov implementuje triedou *sklearn.neighbors.KNeighborsClassifier(**params)*. Aj táto trieda má parametre, ktoré treba vyladiť, aby bol algoritmus čo najpresnejší:

- n_neighbors: počet susedov, ktorí ovplyvňujú cieľovú inštanciu,
- weights {uniform, distance}: typ ovplyvňovania susedov na cieľovú inštanciu. *Uniform*: každý sused ovplyvňuje rovnako, *distance*: sused, ktorý sa nachádza najbližšie od cieľovej inštancie ovplyvňuje najviac a sused, ktorý sa nachádza najďalej od cieľovej inštancie zase najmenej,
- algorithm {ball_tree, kd_tree, brute}: *Ball_tree* a *kd_tree* sú algoritmy na zrýchlenie algoritmu, ktoré využívajú hierarchickú štruktúru binárneho stromu. *Brute* algoritmus používa hrubú silu.

Znova využijem triedu *sklearn.model_selection.GridSearchCV(**params)* na krížové vyhľadávanie. Vyhľadávanie robím na dostupných hodnotách parametrov *weights* a *algorithm* z knižnice *scikit-learn* a počet susedov je rozsah 1-31 a parametrom scoring={precision, accuracy}.

Na obrázku 5.3 je zobrazená chybová matica algoritmu k-najbližších susedov s parametrom scoring=precision a s parametrami klasifikátora. Presnosť na trénovacích inštanciách je 100% na celkovom datasete. použitím 5-krížovej validácie je priemerná presnosť 99,53%. Presnosť na testovacích inšanciach je 87,37% a na 5-krížovej validácí je priemerná úspešnosť 94,01%. 85 ($\sim$ 11%) testovacích inštancií sú falošné negatíva a 13($\sim$ 2%) testovacích inštancií sú falošné pozitíva. Trvanie algoritmu je 38,01 sekúnd. Pri použití druhej možnosti parametra scoring, accuracy, sa presnosť na testovacích inštanciách sa nezmení, ani parametre klasifikátora. Rozdiel je len v trvaní algoritmu, trvá o 8 sekúnd menej.



Obr. 5.3: Chybová matica algoritmu k-najbližších susedov s parametrom scoring: precision

5.3 Decision tree classfier

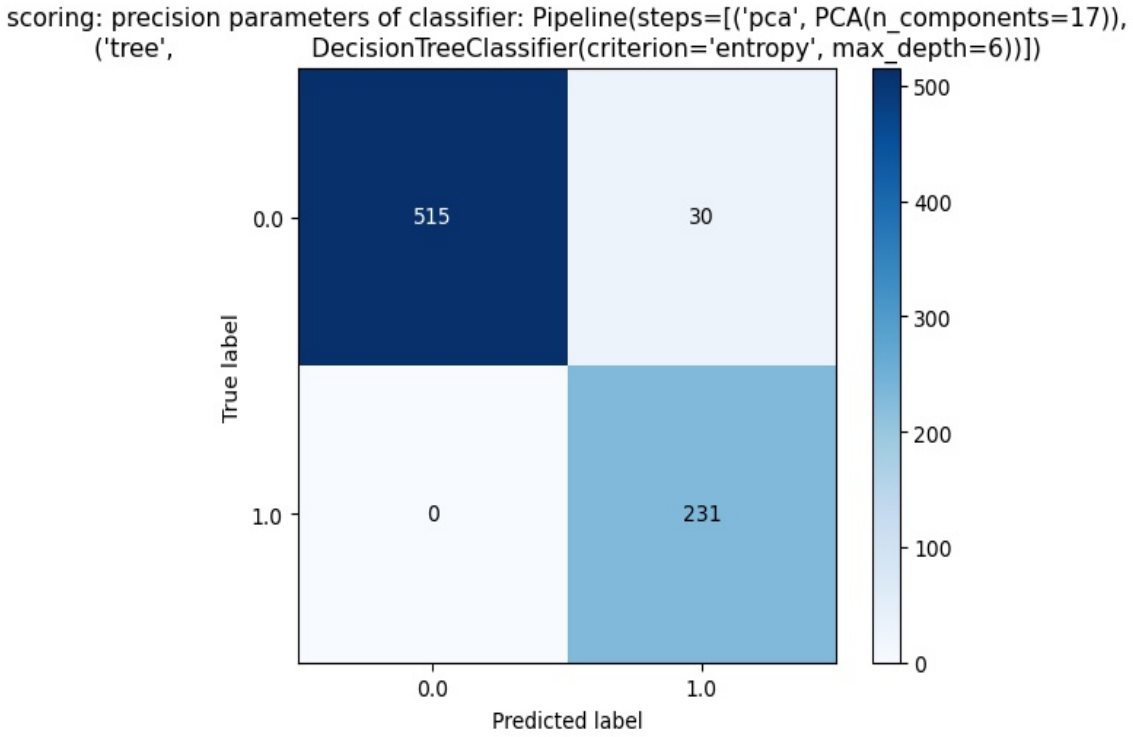
V knižnici *scikit-learn* sa klasifikátor na rozhodovacom strome implementuje v triede *sklearn.tree.DecisionTreeClassifier(**params)*. Parametre, ktoré budem vylaďovať sú:

- criterion {gini, entropy}: giniho nečistota alebo neusporiadanosť pre informačný zisk,
- max_depth: maximálna hĺbka stromu.

Na krížové vyhľadávanie použijem opäť *sklearn.model_selection.GridSearchCV(**params)*. Vyhľadávanie robím na dostupných hodnotách parametra *criterion* z knižnice *scikit-learn*,

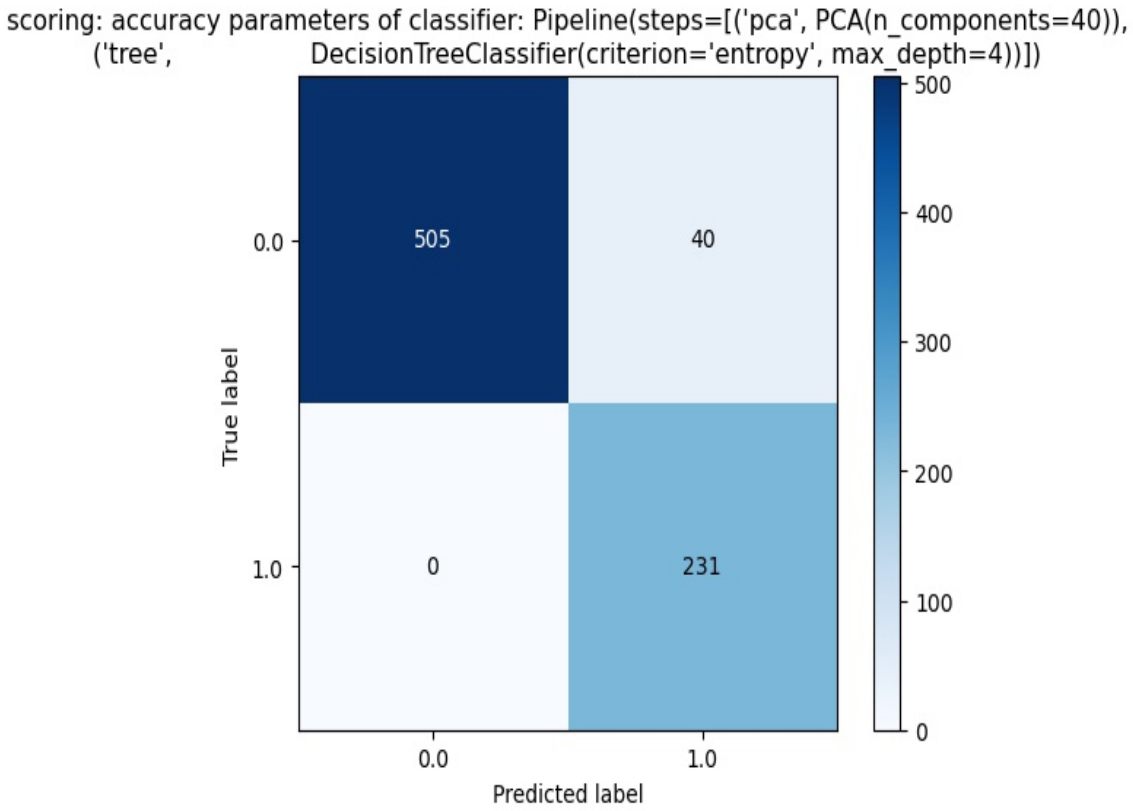
max_depth určím ako hodnoty z {2,4...10} a parametrom scoring={precision, accuracy}. Pre každú túto možnosť použijem *PCA - Principal component analysis* (Analýza hlavných komponentov), ktorá zaručí odstránenie nepodstatných dát, ako je šum v dátach, a zredukujú dimenzie datasetu. Použijem triedu *sklearn.decomposition.PCA(**params)*, ktorá spraví zníženie lineárnej dimenzionality pomocou singulárnej dekompozície údajov (SVD) na ich premietnutie do priestoru nižšej dimenzie. Pred aplikáciou SVD sú vstupné údaje pre každú funkciu vycentrované, ale nie sú zmenšené [22].

Na obrázku 5.4 je zobrazená chybová matica klasifikátora založená na rozhodovacom strome s parametrom scoring=precision. Presnosť na trénovacích inštanciách je 100%, použitím 5-krížovej validácie je priemerná hodnota 99,76%. Presnosť na testovacích inštanciách je 96,13%, použitím 5-krížovej validácie je priemerná hodnota 94,22%. Z chybovej matice je vidno, že všetky nesprávne predpovedané inštancie (30 $\sim$ 3.87%) sú falošné pozitíva. Trvanie algoritmu je 94,5 sekúnd.



Obr. 5.4: Chybová matica klasifikátora založená na rozhodovacom strome s parametrom scoring: precision

Na obrázku 5.5 je zobrazená chybová matica klasifikátora založená na rozhodovacom strome s parametrom scoring=accuracy. Presnosť na trénovacích inštanciách je 100%, použitím 5-krížovej validácie je priemerná hodnota 99,7%. Presnosť na testovacích inštanciách je 94,85%, použitím 5-krížovej validácie je priemerná hodnota 98,07%. Z chybovej matice vidieť, že všetky nesprávne predpovedané inštancie (40 $\sim$ 5.15%) sú falošné pozitíva. Trvanie algoritmu je 89,44 sekúnd.



Obr. 5.5: Chybová matica klasifikátora založená na rozhodovacom strome s parametrom scoring: accuracy

5.4 Multi-layer perceptron

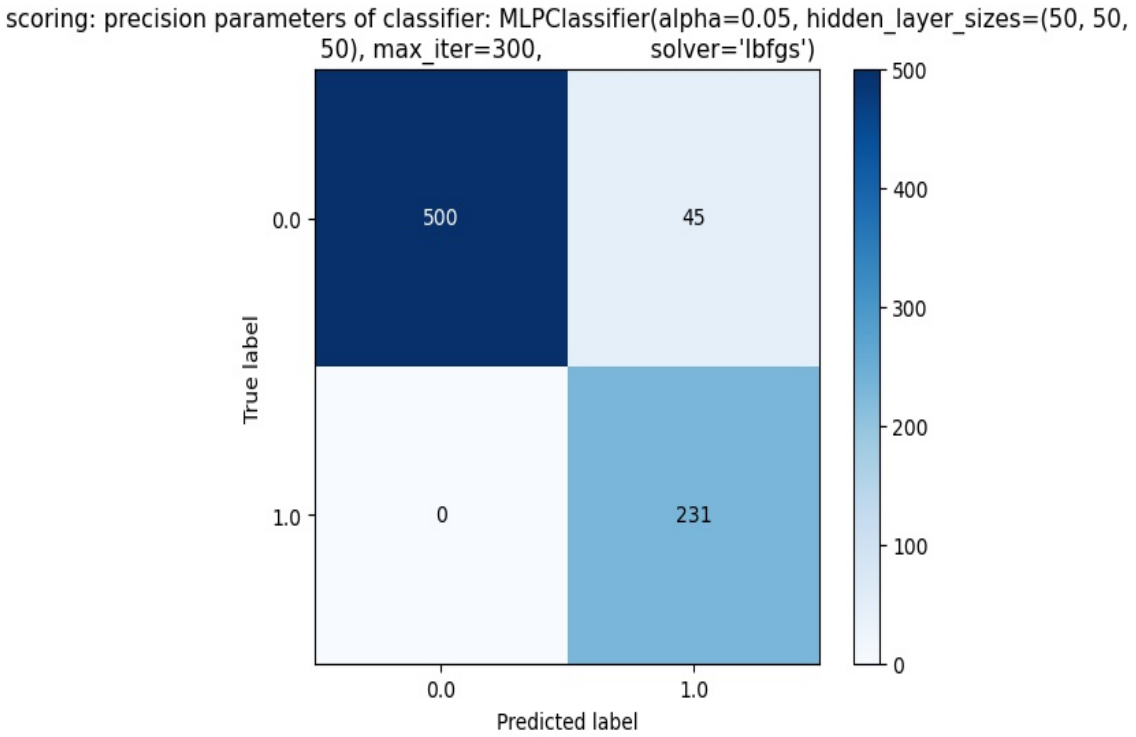
Ako poslednú tehchniku použijem klasifikátor, ktorý je založený na viac-vrstvovom perceptróne. Knižnica *scikit-learn* ponúka triedu *sklearn.neural_network.MLPClassifier(**params)* pre implementáciu MLP klasifikátora. Nasledujúce parametre triedy vyladím na dosiahnutie najlepšieho výsledku:

- hidden_layer_sizes: počet uzlov vo vrstvách, viď sekciu 3.4,
- activation {tanh, relu, logistic}: aktivačné funkcie, viď. sekciu 3.3,
- solver {lbfgs, sqd, adam}: optimalizátor. *Lbfgs* je optimalizátor *quasi-Newtonoých* techník, *sqd* označuje stochastický gradientový zostup a *adam* označuje stochastický optimalizátor založený na gradiente, ktorý navrhli Kingma, Diederik a Jimmy Ba,
- alpha: L2 regulizátor,
- learning_rate {constant, adaptive}: Rýchlosť učenia pre aktualizácie váh jednotlivých uzlov.

Vyhľadávanie najlepších parametrov viac-vrstvového perceptrónu implementujem pomocou triedy *sklearn.model_selection.GridSearchCV(**params)*. Hodnoty parametra *hidden_layer_sizes* zvolím {(50,50,50), (50,100,50), (100,)}, parametra *activation* {tanh, relu},

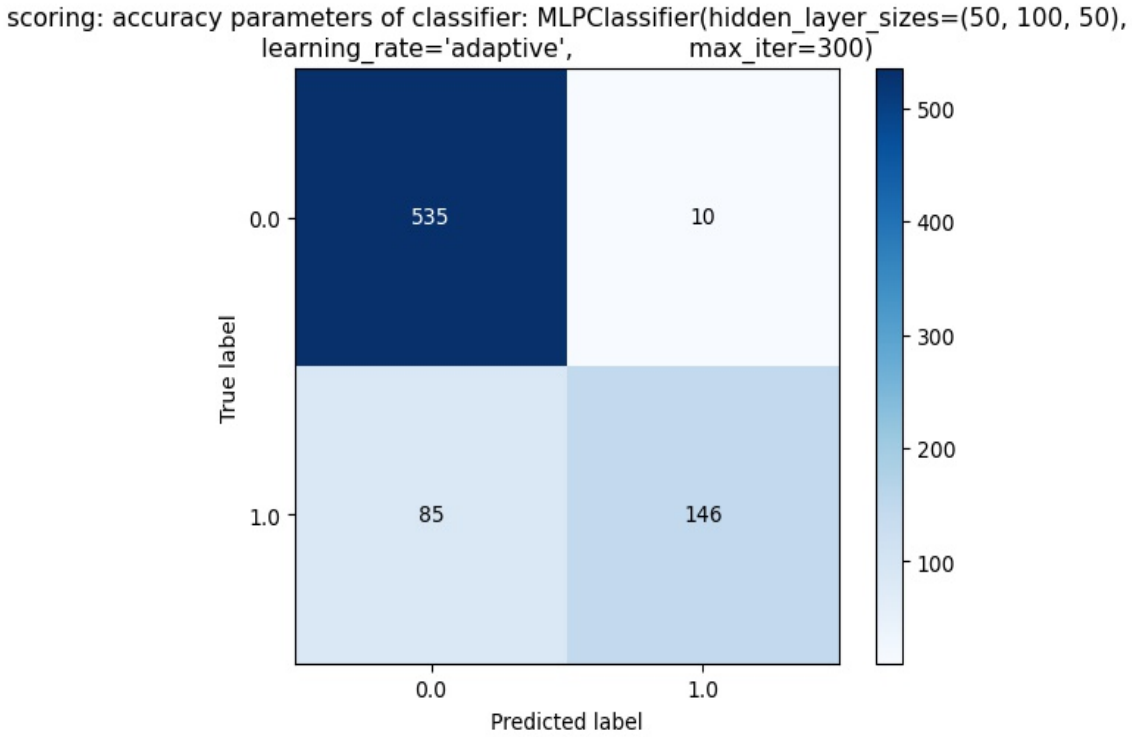
parametra *solver* {lbfgs, adam, sgd}, parametra *alpha* {0.0001, 0.05} a parametra *learning_rate* {constant, adaptive}.

Na obrázku 5.6 je zobrazená chybová matica klasifikátora založená na viac-vrstvovom perceptróne s parametrom scoring=precision. Presnosť na trénovacích inštanciách je 100% a použitím 5-krížovej validácie je priemerná hodnota 99,53%. Presnosť na testovacích inštanciách je 94,02% a použitím 5-krížovej validácie je priemerná hodnota 96,65%. Z chybovej matice je vidno, že všetky nesprávne predpovedané inštancie (45 $\sim$ 5.85%) sú falošné pozitíva. Trvanie algoritmu je 130,44 sekúnd.



Obr. 5.6: Chybová matica klasifikátora založená na viac-vrstvovom perceptróne s parametrom scoring: precision

Na obrázku 5.7 je zobrazená chybová matica klasifikátora založená na viac-vrstvovom perceptróne s parametrom scoring=accuracy. Presnosť na trénovacích inštanciách je 100%, použitím 5-krížovej validácie je priemerná hodnota 99,56%. Presnosť na testovacích inštanciách je 87.76% a použitím 5-krížovej validácie je priemerná hodnota 95.75%. Z chybovej matice je vidno, že 85 ($\sim$ 10,95%) testovacích inštancií sú falošné negatíva a 15 ($\sim$ 3%) sú falošné pozitíva. Trvanie algoritmu je 131,3 sekúnd.



Obr. 5.7: Chybová matica klasifikátora založená na viac-vrstvovom perceptróne s parametrom scoring: accuracy

5.5 Zhodnotenie experimentov

Výsledky experimentov na rovnakých testovacích dátach ukázujú dobré výsledky. Najlepšie výsledky má klasifikátor založený na rozhodovacom strome, kde najskôr znížime dimenzie pomocou funkcie *sklearn.decomposition.PCA(n_components=17)* a parametre tohoto stromu sú: *criterion=entropy, max_depth=6*. Úspešnosť je 96,33%, ale všetky neúspešné testovacie inštancie spadajú do triedy falošných pozitív, čo v realite znamená, že všetky útoky sú zachytené, ale normálne stavy sú tiež vyhlásené za škodlivé. Na druhú stranu algoritmus SVM, s parametrami: *C=1, gamma=0.5, kernel=rbf* má síce úspešnosť 89,05%, ale všetky testovacie inštancie spadajú do triedy falošných negatív. Taktiež je nutné poznamenať, že klasifikátor založený na rozhodovacom strome je o niečo rýchlejší ako SVM algoritmus.

Kapitola 6

Záver

V práci som sa zaoberal analýzou škodlivej, šifrovanej sieťovej prevádzky pomocou metód umelej inteligencie. Hlavnou náplňou tejto práce je vytvorenie systému pre detekciu bezpečnostných anomálií pomocou vybraných metód umelej inteligencie.

Systém na detekciu prienikov detekovaním anomálií je založený na umelej inteligencii. To znamená, že množstvo správnych, vyrovnaných trénovacích a testovacích dát a vytvorenie patričných datasetov je veľmi dôležité. Vytvoril som datasety, počet trénovacích inštancí je 975, z toho je 556 normálnych a zvyšok anomálnych. Vyrovnaný počet normálnych a anomálnych trénovacích inštancií je veľmi užitočné na správne detekovanie testovacích dát. Počet testovacích inštancií je 776, z toho 545 je normálnych a 231 je anomálnych.

Následne som použil štyri techniky detekcie anomálií založené na klasifikácií. Tieto techniky najlepšie vyhovujú počtu a povahe mojich trénovacích dát. Na týchto technikách som vylaďoval čo najlepšie ich parametre pre dosiahnutie najlepších výsledkov. V sekcii 5.5 som zhodnotil dve najlepšie techniky, ktoré dosiahli najväčšiu presnosť na testovacích dátach.

Zlepšenie týchto štyroch techník je určite nájdenie viac dátových inštancií na trénovanie a testovanie, upravenie hodnôt, ktoré chýbajú v datasetoch alebo nájdenie odľahlých (outlier) inštancií a transformovanie alebo odstránenie týchto inštancií z trénovacej sady. Ako ďalší návrh môže byť nájdenie vlastností, ktoré najviac ovplyvňujú klasifikáciu a pridelenie väčších váh týmto vlastnostiam [23].

Literatúra

[1] Abraham, A., Pedregosa, F., Eickenberg, M., Gervais, P., Mueller, A. et al. Machine learning for neuroimaging with scikit-learn. *Frontiers in neuroinformatics*. Frontiers. 2014, zv. 8, s. 14.

[2] Ahmed, M., Mahmood, A. N. a Hu, J. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*. Elsevier. 2016, zv. 60, s. 19–31.

[3] Althouse, J. *TLS Fingerprinting with JA3 and JA3S*. Salesforce Engineering, Jan 2019. Dostupné z: https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967.

[4] Anderson, J. P. Computer security threat monitoring and surveillance, James P. *Anderson Co., Fort Washington, PA*. 1980.

[5] Chandola, V., Banerjee, A. a Kumar, V. Anomaly Detection: A Survey. *ACM Comput. Surv.* Júl 2009, zv. 41. DOI: 10.1145/1541880.1541882.

[6] Chang, C.-C. a Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*. 2011, zv. 2, s. 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[7] Dierks, T. a Rescorla, E. The transport layer security (TLS) protocol version 1.2. RFC 5246, August. 2008.

[8] Eskin, E., Arnold, A., Prerau, M., Portnoy, L. a Stolfo, S. A geometric framework for unsupervised anomaly detection. In: *Applications of data mining in computer security*. Springer, 2002, s. 77–101.

[9] Fei Fei Li, R. K. *Convolutional Neural Networks for Visual Recognition*. Dostupné z: https://cs231n.github.io/neural-networks-1/.

[10] Garcia S., V. V. *Stratoshphere Research Laboratory* [https://www.stratosphereips.org/]. Accessed: 2021-04-30.

[11] Graupe, D. *Principles of artificial neural networks*. World Scientific, 2013.

[12] Hankerson, D., Menezes, A. J. a Vanstone, S. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[13] Husák, M., Cermák, M., Jirsík, T. a Celeda, P. Network-based HTTPS client identification using SSL/TLS fingerprinting. In: IEEE. *2015 10th international conference on availability, reliability and security*. 2015, s. 389–396.

[14] Javaid, A., Niyaz, Q., Sun, W. a Alam, M. A deep learning approach for network intrusion detection system. In: *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. 2016, s. 21–26.

[15] Karn, U. *A Quick Introduction to Neural Networks*. Aug 2016. Dostupné z: https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/.

[16] Kemmerer, R. A. Cybersecurity. In: IEEE. *25th International Conference on Software Engineering, 2003. Proceedings.* 2003, s. 705–715.

[17] Kendall, K. K. R. *A database of computer attacks for the evaluation of intrusion detection systems*. 1999. Dizertačná práca. Massachusetts Institute of Technology.

[18] Kruegel, C., Mutz, D., Robertson, W. a Valeur, F. Bayesian event classification for intrusion detection. In: IEEE. *19th Annual Computer Security Applications Conference, 2003. Proceedings.* 2003, s. 14–23.

[19] Kumar, V. a Sangwan, O. P. Signature based intrusion detection system using SNORT. *International Journal of Computer Applications & Information Technology*. 2012, zv. 1, č. 3, s. 35–41.

[20] Matoušek, P., Burgetová, I., Ryšavỳ, O. a Victor, M. On Reliability of JA3 Hashes for Fingerprinting Mobile Applications. In: Springer. *International Conference on Digital Forensics and Cyber Crime*. 2020, s. 1–22.

[21] Otomar, K. et al. *Lékařská fyziologie*. Grada Publishing as, 2011.

[22] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011, zv. 12, s. 2825–2830.

[23] RayI, R. *How To Increase Accuracy Of Machine Learning Model*. Analytics Vidhya, Jun 2020. Dostupné z: https://www.analyticsvidhya.com/blog/2015/12/improve-machine-learning-results/.

[24] Sharma, S. Activation functions in neural networks. *Towards data science*. 2017, zv. 6.

[25] Stolfo, J., Fan, W., Lee, W., Prodromidis, A. a Chan, P. K. Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection. *Results from the JAM Project by Salvatore*. 2000, s. 1–15.

[26] Sundaram, A. An introduction to intrusion detection. *Crossroads*. 1996, zv. 2, č. 4, s. 3–7.

[27] Susanka, T. *What are TLS extensions?* On Cryptography and Security, Sep 2017. Dostupné z: https://blog.susanka.eu/what-are-tls-extensions/.

[28] Trnavská, N. *Three-way handshake*. 2021. Dostupné z: http://labyrint.vadkerti.sk/three-way-handshake/.

Príloha A

Obsah pamäťového média

Na priloženom pamäťovom médiu sú uložené všetky mnou vytvorené zdrojové súbory, popis inštalácie, táto práca a zdrojové súbory v LaTeX, ako i všetky ostatné súbory potrebné pre preklad. Priložené sú aj použité pcap súbory, ako aj vytvorené csv súbory. Nachádza sa tu tiež backup databázy, pre zrýchlenie otestovania techník umelej inteligencie.

tex Zdrojové LaTeXsúbory pre vytvorenie tohto pdf súboru.

tex/obrazky Obrázky použité v mojej práci.

pdf Tento pdf súbor.

db_backup Záloha databáze.

src Zdrojové súbory, popis inštalácie.

src/pcap_used Pcap súbory použité na analýzu.

src/csv_used Vytvorené csv súbory.

src/csv_used/normalized Normalizované csv súbory.