

Slovenská technická univerzita v Bratislave  
Fakulta informatiky a informačných technológií  
FIIT-182905-74582

Bc. Zoltán Csengődy

IDENTIFIKÁCIA BEZPEČNOSTNÝCH RIZÍK A ANALÝZA DÁT Z PROSTREDIA  
POČÍTAČOVÝCH SIETÍ

Diplomová práca

Študijný program:	Inteligentné softvérové systémy
Študijný odbor:	18. Informatika
Miesto vypracovania:	Ústav počítačového inžinierstva a aplikovanej informatiky
Vedúci práce:	Ing. Rudolf Grežo

máj 2020



## Zadanie diplomovej práce

*Meno študenta:* **Bc. Zoltán Csengődy**

*Študijný program:* Inteligentné softvérové systémy

*Študijný odbor:* Softvérové inžinierstvo – hlavný študijný odbor  
Umelá inteligencia – vedľajší študijný odbor

*Názov práce:* **Identifikácia bezpečnostných rizík a analýza dát z prostredia počítačových sietí**

Samostatnou výskumnou a vývojovou činnosťou v rámci predmetov Diplomový projekt I, II, III vypracujte diplomovú prácu na tému, vyjadrenú vyššie uvedeným názvom tak, aby ste dosiahli tieto ciele:

*Všeobecný cieľ:*

Vypracovaním diplomovej práce preukážte, ako ste si osvojili metódy a postupy riešenia relatívne rozsiahlych projektov, schopnosť samostatne a tvorivo riešiť zložité úlohy aj výskumného charakteru v súlade so súčasnými metódami a postupmi študovaného odboru využívanými v príslušnej oblasti a schopnosť samostatne, tvorivo a kriticky pristupovať k analýze možných riešení a k tvorbe modelov.

*Špecifický cieľ:*

Vytvorte riešenie zodpovedajúce návrhu textu zadania, ktorý je prílohou tohto zadania. Návrh bližšie opisuje tému vyjadrenú názvom. Tento opis je záväzný, má však rámcový charakter, aby vznikol dostatočný priestor pre Vašu tvorivosť.

Riadťe sa pokynmi Vášho vedúceho.

Pokial' v priebehu riešenia, opierajúc sa o hlbšie poznanie súčasného stavu v príslušnej oblasti, alebo o priebežné výsledky Vášho riešenia, alebo o iné závažné skutočnosti, dospejete spoločne s Vaším vedúcim k presvedčeniu, že niečo v texte zadania a/alebo v názve by sa malo zmeniť, navrhnite zmenu. Zmena je spravidla možná len pri dosiahnutí kontrolného bodu.

*Miesto vypracovania:* Ústav počítačového inžinierstva a aplikovanej informatiky, FIIT STU Bratislava

*Vedúci práce:* **Ing. Rudolf Grežo**

*Termíny odovzdania:*

Podľa harmonogramu štúdia platného pre semester, v ktorom máte príslušný predmet (Diplomový projekt I, II, III) absolvovať podľa Vášho študijného plánu

*Predmety odovzdania:*

V každom predmete dokument podľa pokynov na [www.fiit.stuba.sk](http://www.fiit.stuba.sk) v časti:  
home > Informácie o > štúdiu > harmonogram štúdia > diplomový projekt.

V Bratislave dňa 11. 2. 2019

**SLOVENSKÁ TECHNICKÁ UNIVERZITA  
V BRATISLAVE**

Fakulta informatiky a informačných technológií  
Ilkovičova 2, 842 16 Bratislava 4

1

prof. Ing. Pavol Návrat, PhD.  
riaditeľ Ústavu informatiky, informačných systémov  
a softvérového inžinierstva



# Návrh zadania diplomovej práce

Finálna verzia do diplomovej práce<sup>1</sup>

## Študent:

**Meno, priezvisko, tituly:** Zoltán Csengődy, Bc.

**Študijný program:** Inteligentné softvérové systémy

**Kontakt:** csengody4@gmail.com

## Výskumník:

**Meno, priezvisko, tituly:** Rudolf Grežo, Ing.

## Projekt:

**Názov:** Identifikácia bezpečnostných rizík a analýza dát z prostredia počítačových sietí

**Názov v angličtine:** Data analysis and security risk identification in computer networks

**Miesto vypracovania:** Ústav počítačového inžinierstva a aplikovanej informatiky,  
FIIT STU

**Oblast’ problematiky:** počítačové siete, analýza dát, bezpečnosť

## Text návrhu zadania<sup>2</sup>

Počítačové siete nás sprevádzajú každodenným životom, pričom jedným z aspektov pri práci s nimi je zvýšenie spoľahlivosti a bezpečnosti siete. Spoľahlivosť a bezpečnosť siete sú kľúčové pre poskytovanie rôznych služieb siete, ako napríklad prenos informácií a zdieľanie prostriedkov. Rozvoj v oblasti sietí je veľmi rýchly a tým pádom sa môžu stať zraniteľnými voči novým druhom útokov. Preto je potrebné venovať sa zabezpečeniu siete a hľadaniu nových spôsobov detekcie sieťových útokov.

Analyzujte súčasný stav problematiky, typy útokov a metódy použiteľné pri analýze dát z počítačových sietí. Vzhľadom na identifikáciu bezpečnostných rizík a útočnej premávky navrhnite modifikáciu existujúcich algoritmov alebo metód so zameraním na spracovanie tohto typu dát. Zamerajte sa najmä na voľne dostupné dátové množiny. Pred samotným identifikovaním bezpečnostných rizík a útočnej premávky analyzujte a spracujte rozsiahle množiny dát.

Vami navrhnutý algoritmus alebo metódu implementujte vo forme softvérového nástroja. S ohľadom na jednoduchosť, modulárnosť, rozšíriteľnosť a použiteľnosť riešenia sa snažte využiť existujúce softvérové moduly. Výsledné riešenie overte na množine existujúcich testovacích dát a ukážte jeho prínos voči podobným riešeniam.

<sup>1</sup> Vytlačiť obojstranne na jeden list papiera

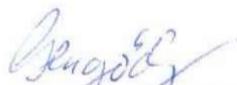
<sup>2</sup> 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

### Literatúra<sup>3</sup>

- LIU, Gao, Zheng YAN a Witold PEDRYCZ. Data collection for attack detection and security measurement in Mobile Ad Hoc Networks: A survey. *Journal of Network and Computer Applications* [online]. 2018, 105, s. 105-122. DOI: 10.1016/j.jnca.2018.01.004.
- ZAMAN, Marzia a Chung-Horng LUNG. Evaluation of machine learning techniques for network intrusion detection. In: NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium [online]. 2018, s. 1-5. DOI: 10.1109/NOMS.2018.8406212.

Vyššie je uvedený návrh diplomového projektu, ktorý vypracoval(a) Bc. Zoltán Csengődy, konzultoval(a) a osvojil(a) si ho Ing. Rudolf Grežo a súhlasí, že bude takýto projekt viest' v prípade, že bude pridelený tomuto študentovi.

V Bratislave dňa 28.1.2019



Podpis študenta



Podpis výskumníka

### Vyjadrenie garanta predmetov Diplomový projekt I, II, III

Návrh zadania schválený: áno / nie<sup>4</sup>

Dňa: ..... 11. 2. 2019 .....



Podpis garanta predmetov

<sup>3</sup> 2 vedecké zdroje, každý v samostatnej rubrike a s údajmi zodpovedajúcimi bibliografickým odkazom podľa normy STN ISO 690, ktoré sa viažu k téme zadania a preukazujú výskumnú povahu problému a jeho časťnosť (uvedte všetky potrebné údaje na identifikáciu zdroja, pričom uprednostnite vedecké príspevky v časopisoch a medzinárodných konferenciách)

<sup>4</sup> Nehodiace sa prečiarknite

Čestne vyhlasujem, že som túto prácu vypracoval samostatne, na základe konzultácií a s použitím uvedenej literatúry.

V Bratislave, 07.05.2020

Bc. Zoltán Csengődy



# Anotácia

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Študijný program: Inteligentné softvérové systémy

Autor: Bc. Zoltán Csengődy

Diplomová práca: Identifikácia bezpečnostných rizík a analýza dát z prostredia počítačových sietí

Vedúci diplomovej práce: Ing. Rudolf Grežo

máj 2020

Počítačové siete nás sprevádzajú každodenným životom, pričom jedným z aspektov pri práci s nimi je zvýšenie spoločalivosti a bezpečnosti siete. S rozvojom tejto technologickej oblasti prichádzajú nové spôsoby a typy útokov, voči ktorým sa treba chrániť. Táto práca je venovaná výskumu v oblasti odhalenia počítačových útokov metódami strojového učenia. Cieľom tejto diplomovej práce je vytvorenie programového modulu, ktorý vhodným spôsobom dokumentuje vybrané algoritmy strojového učenia. Hlavnou motiváciou je vytvorenie jednotnej analýzy vplyvov rôznych nastavení klasifikačných algoritmov a rôznych spôsobov predspracovania vybraných dátových množín na výsledky odhalenia sietových útokov. Súčasťou tejto práce je vlastný návrh riešenia, ktorý vyplýva z faktu, že hlavným nedostatkom použitia algoritmov strojového učenia je nedostatočná dokumentácia použitia, tvorba architektúry a nastavenia parametrov. Dnešný spôsob použitia týchto metód spočíva predovšetkým v skúšaní a optimalizácii najlepšieho riešenia pre daný model. Na základe rôznych nastavení a vstupov do metód dokážeme optimalizovať klasifikáciu a tým pádom pri vhodných nastaveniach dosahovať lepšie výsledky hodnotenia modelu. V tejto práci sa venujeme hľadaniu anomálií v sietovej premávke a metódam, ktoré sú určené na ich odhalovanie. Súčasťou práce je taktiež vhodné predspracovanie dát vybranej dátovej množiny a odhalenie závislostí medzi jeho atribútmi, ktoré majú značný vplyv na odhalovanie útokov. Výstupom tejto práce je programový modul na predspracovanie dátovej množiny a programový modul strojového učenia. Na základe výsledkov z programového modulu porovnáme výhody a nevýhody použitých metód strojového učenia a výsledky interpretujeme vhodným spôsobom.



# **Annotation**

Slovak University of Technology in Bratislava

FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Degree course: Intelligent software systems

Author: Bc. Zoltán Csengődy

Master's thesis: Data analysis and security risk identification in computer networks

Supervisor: Ing. Rudolf Grežo

2020, May

Computer networks accompany us with everyday life and one aspect of working with them is to increase network reliability and security. With the development of this technological area, new ways and types of attacks emerge and we must protect ourselves against them. This work is devoted to research in the field of computer attacks detection using machine learning methods. The aim of this master thesis is to create a program module, which in an appropriate way documents the selected machine learning algorithms. The main motivation is to create a unified analysis of the effects of the different settings of the classification algorithms and the different methods of pre-processing the selected datasets to the results of network attack detection. Part of this work is a custom design solution, which stems from the fact that the main drawback of using machine learning algorithms is insufficient documentation of the use, creation of architecture and parameter settings. Today's use of these methods lies primarily in testing and optimizing the best solution for a given model. Based on the various settings and inputs to the methods we can optimize the classification and thus achieve better model evaluation results with appropriate settings. In this work, we look for anomalies in network traffic and methods intended to detect them. Part of the work is also a suitable pre-processing of the selected dataset and revealing the dependencies between its attributes, which have a significant impact on the attack's detection. The outcome of this work is a program module for dataset preprocessing and a machine learning program module. Based on the results from the program module, we compare the advantages and disadvantages of used machine learning methods and interpret the results in an appropriate way.



# Obsah

<b>1.</b>	<b>Úvod.....</b>	<b>1</b>
<b>2.</b>	<b>Analýza problematiky.....</b>	<b>3</b>
<b>2.1.</b>	<b>Typy systémov na detekciu sietových útokov .....</b>	<b>3</b>
2.1.1.	Anomálne založené detekčné systémy .....	5
2.1.2.	Charakteristiky IDS.....	8
<b>2.2.</b>	<b>Architektúra IDS .....</b>	<b>10</b>
<b>2.3.</b>	<b>Existujúce nástroje .....</b>	<b>11</b>
<b>2.4.</b>	<b>Spôsob vyhodnocovania IDS.....</b>	<b>12</b>
<b>2.5.</b>	<b>Sietové útoky.....</b>	<b>16</b>
2.5.1.	Detekcia anomálií.....	17
2.5.2.	Detekcia zneužitia .....	17
2.5.3.	Monitorovanie cieľa.....	17
2.5.4.	Špionáž.....	18
2.5.5.	Typy sietových útokov .....	18
2.5.6.	Odhalenie sietových útokov .....	20
<b>2.6.</b>	<b>Strojové učenie .....</b>	<b>22</b>
2.6.1.	Klasifikačné algoritmy .....	23
2.6.2.	Neurónová sieť .....	26
<b>2.7.</b>	<b>Dátové množiny.....</b>	<b>29</b>
2.7.1.	NSL-KDD .....	29
2.7.2.	UNSW-NB15 .....	31
2.7.3.	ISCX.....	34
2.7.4.	Predspracovanie dát .....	36
<b>2.8.</b>	<b>Zhodnotenie analýzy.....</b>	<b>39</b>
<b>3.</b>	<b>Špecifikácia požiadaviek.....</b>	<b>41</b>
<b>3.1.</b>	<b>Funkcionálne požiadavky.....</b>	<b>41</b>
<b>3.2.</b>	<b>Nefunkcionálne požiadavky .....</b>	<b>42</b>
<b>3.3.</b>	<b>Prípady použitia.....</b>	<b>43</b>
3.3.1.	Scenáre prípadov použitia .....	44
<b>4.</b>	<b>Návrh riešenia .....</b>	<b>49</b>
<b>4.1.</b>	<b>Vývojové prostredie.....</b>	<b>50</b>
<b>4.2.</b>	<b>Programový modul predspracovania dátovej množiny .....</b>	<b>50</b>
<b>4.3.</b>	<b>Opis činností programového modulu predspracovania dátovej množiny .....</b>	<b>55</b>
<b>4.4.</b>	<b>Programový modul strojového učenia .....</b>	<b>56</b>

<b>4.5.</b>	<b>Opis činností programového modulu strojového učenia.....</b>	<b>59</b>
<b>5.</b>	<b>Implementácia .....</b>	<b>61</b>
<b>5.1.</b>	<b>Nastavenie vývojového prostredia.....</b>	<b>61</b>
<b>5.2.</b>	<b>Programový modul predspracovania dátovej množiny .....</b>	<b>61</b>
5.2.1.	Výber dátovej množiny a metódy predspracovania .....	62
5.2.2.	Načítanie dátovej množiny.....	62
5.2.3.	Analýza dátovej množiny.....	62
5.2.4.	Čistenie dát.....	65
5.2.5.	Doplnenie chýbajúcich hodnôt.....	66
5.2.6.	Spracovanie kategorických atribútov.....	68
5.2.7.	Obohacovanie dát.....	68
5.2.8.	Štandardizácia .....	69
5.2.9.	Vzorkovanie .....	69
5.2.10.	Uloženie súborov .....	70
<b>5.3.</b>	<b>Programový modul strojového učenia .....</b>	<b>71</b>
5.3.1.	Načítanie, zjednotenie a transformácia predspracovanej dátovej množiny .....	71
5.3.2.	Trénovanie a vyhodnotenie modelu strojového učenia.....	72
5.3.3.	Porovnanie metód strojového učenia .....	76
5.3.4.	Uloženie súborov .....	77
<b>6.</b>	<b>Overenie riešenia.....</b>	<b>79</b>
<b>6.1.</b>	<b>Experimentálne overenie riešenia .....</b>	<b>80</b>
<b>6.2.</b>	<b>Vyhodnotenie výsledkov riešenia .....</b>	<b>84</b>
<b>7.</b>	<b>Záverečné myšlienky.....</b>	<b>107</b>
<b>8.</b>	<b>Zhodnotenie a záver .....</b>	<b>109</b>
<b>Bibliografia .....</b>		<b>113</b>

**Príloha A: Plán práce**

**Príloha B: Technická dokumentácia**

**Príloha C: Článok na konferenciu IIT.SRC 2020**

**Príloha D: Opis digitálnej časti práce**

# Zoznam obrázkov

Obrázok 1 – Všeobecná architektúra IDS [41] .....	3
Obrázok 2 – Architektúra HIDS [41].....	4
Obrázok 3 – Architektúra NIDS [41].....	4
Obrázok 4 – Bodová, kontextová a kolektívna anomália [8] .....	6
Obrázok 5 – Funkcionality IDS [53].....	10
Obrázok 6 – ROC krivka [46].....	15
Obrázok 7 – Návrh platformy na odhalenie siet'ových útokov [29] .....	21
Obrázok 8 – Prostredie odhalenia útoku [29] .....	21
Obrázok 9 – Štruktúra modulu na detekciu siet'ových útokov [29].....	22
Obrázok 10 – Neurón [42] .....	26
Obrázok 11 – Feed-forward neurónová siet' [19] .....	27
Obrázok 12 – Model neurónovej siete pre IDS [42] .....	28
Obrázok 13 – Proces klasifikácie siet'ových útokov .....	28
Obrázok 14 – Zastúpenie siet'ovej premávky v dátovej množine UNSW-NB15 [26] .....	32
Obrázok 15 – Vzorka dát UNSW-NB15.....	33
Obrázok 16 – Diagram prípadov použitia .....	43
Obrázok 17 – Diagram aktivít programového modulu predspracovania dátovej množiny .....	55
Obrázok 18 – Diagram aktivít programového modulu strojového učenia .....	60
Obrázok 19 – Graf distribúcie normálnej a útočnej premávky .....	63
Obrázok 20 – Graf distribúcie útočnej premávky .....	64
Obrázok 21 – Korelačná matica .....	64
Obrázok 22 – Korelácia atribútov .....	65
Obrázok 23 – Distribúcia hodnôt pre zdrojové bity .....	66
Obrázok 24 – Porovnanie metód strojového učenia .....	76
Obrázok 25 – ROC krivka najlepšieho modelu náhodného lesa .....	81
Obrázok 26 – ROC krivka SGD klasifikátora .....	82
Obrázok 27 – ROC krivka rozhodovacieho stromu .....	82
Obrázok 28 – ROC krivka náhodného lesa.....	83
Obrázok 29 – ROC krivka najlepšieho modelu XGradient Boosting .....	92
Obrázok 30 – ROC krivka najhoršieho modelu XGradient Boosting .....	93
Obrázok 31 – ROC krivka najlepšieho modelu náhodného lesa .....	94
Obrázok 32 – ROC krivka najlepšieho modelu LSTM.....	95

Obrázok 33 – Architektúra LSTM .....	97
Obrázok 34 – Krivka učenia sa najlepšieho modelu LSTM pre správnosť .....	98
Obrázok 35 – Krivka učenia sa najlepšieho modelu LSTM pre stratu .....	98
Obrázok 36 – ROC krivka najhoršieho modelu LSTM .....	99
Obrázok 37 – Krivka učenia sa najhoršieho modelu LSTM pre správnosť.....	99
Obrázok 38 – Krivka učenia sa najhoršieho modelu LSTM pre stratu.....	99
Obrázok 39 – ROC krivka perceptronu .....	100
Obrázok 40 – ROC krivka najlepšieho SGD klasifikátora .....	101
Obrázok 41 – Google Drive úložný priestor .....	B-1
Obrázok 42 – Prostredie Google Colab .....	B-3
Obrázok 43 – Počítačový modul Google .....	B-3
Obrázok 44 – Autentifikácia – krok 1.....	B-4
Obrázok 45 – Autentifikácia – krok 2.....	B-4
Obrázok 46 – Autentifikácia – krok 3.....	B-5
Obrázok 47 – Google Drive adresárová štruktúra v prostredí Google Colab .....	B-5
Obrázok 48 – Obsah programového modulu .....	B-6
Obrázok 49 – Ukončenie spojenia s počítačovým modulom.....	B-6
Obrázok 50 – Nastavenie programového modulu na predspracovanie dátovej množiny .....	B-7
Obrázok 51 – Prahová hodnota - threshold.....	B-8
Obrázok 52 – Nastavenie programového modulu pre strojové učenie .....	B-9
Obrázok 53 – Načítanie predspracovanej dátovej množiny .....	B-11
Obrázok 54 – Nastavenie LSTM neurónovej siete .....	B-12

# Zoznam tabuľiek

Tabuľka 1 – Typy IDS [31] .....	5
Tabuľka 2 – Kontingenčná tabuľka .....	14
Tabuľka 3 – Útoky v testovacom súbore dát NSL-KDD [39] .....	30
Tabuľka 4 – Tabuľka výsledkov experimentu v nástroji WEKA [13] .....	31
Tabuľka 5 – Tabuľka výsledkov pre sieťovú foreznnú schému [33].....	33
Tabuľka 6 – Tabuľka presnosťí klasifikačných algoritmov pre ISCX IDS 2012 [51].....	35
Tabuľka 7 – Scenár prípadu použitia spracuj vybranú dátovú množinu .....	44
Tabuľka 8 – Scenár prípadu použitia aplikuj metódu strojového učenia na množinu dát .....	45
Tabuľka 9 – Scenár prípadu použitia vyber dátovú množinu .....	46
Tabuľka 10 – Scenár prípadu použitia vyber metódu spracovania dátovej množiny .....	46
Tabuľka 11 – Scenár prípadu použitia vyber metódu strojového učenia.....	47
Tabuľka 12 – Výsledky experimentálnych testov – správnosť/F1-skóre .....	80
Tabuľka 13 – Výsledky testov – správnosť .....	88
Tabuľka 14 – Výsledky testov – F1-skóre .....	88
Tabuľka 15 – Výsledky testov – ROC-AUC .....	88
Tabuľka 16 – Výsledky testov – strata .....	89
Tabuľka 17 – Výsledky testov – počet epoch.....	89
Tabuľka 18 – Hyperparametre stromovo založených klasifikátorov .....	90
Tabuľka 19 – Hyperparametre LR, SGD a P klasifikátora .....	90
Tabuľka 20 – Kontingenčná tabuľka pre najlepší XGradient Boosting .....	92
Tabuľka 21 – Kontingenčná tabuľka pre najlepší náhodný les.....	94
Tabuľka 22 – Kontingenčná tabuľka pre najlepšiu LSTM neurónovú sieť.....	96
Tabuľka 23 – Kontingenčná tabuľka pre perceptron .....	100
Tabuľka 24 – Kontingenčná tabuľka pre najlepší SGD klasifikátor.....	101
Tabuľka 25 – Výsledky TPOT.....	102
Tabuľka 26 – Overenie riešenia na simulovanej sieťovej premávke – správnosť .....	103
Tabuľka 27 – Overenie riešenia na simulovanej sieťovej premávke – F1-skóre.....	103
Tabuľka 28 – Kontingenčná tabuľka pre náhodný les pre 500 000 paketov .....	103
Tabuľka 29 – Kontingenčná tabuľka pre LSTM neurónovú sieť pre 500 000 paketov .....	103
Tabuľka A.30 – Plán práce k DP I.....	A-1
Tabuľka A.31 – Plán práce k DP II.....	A-1
Tabuľka A.32 – Plán práce k DP III .....	A-2



# 1. Úvod

S rozvojom internetových technológií počítačové siete postupne menia životy ľudí a čoraz viac uľahčujú prácu a spôsob práce ľudí. Rozvoj tejto oblasti je veľmi rýchly a tým pádom je aj čoraz zraniteľnejší voči počítačovým útokom. S rozvojom tejto technologickej oblasti prichádzajú nové spôsoby a typy útokov.

Nebezpečenstvo počítačového útoku a jeho zabránenie je dôležitým aspektom, ktorý sa výskumníci v tejto oblasti snažia vyriešiť. Keďže počítačová sieť môže byť otvorená (voľne dostupná) a medzinárodne zdieľaná, tak údaje, ktoré sú v nej prenášané nie sú v bezpečí. Preto je potrebné zaviesť technické opatrenia na zabezpečenie ochrany údajov v sietovom prostredí.

Zhromažďovanie údajov v počítačovej sieti môže výrazne pomôcť pri detekcii sietových útokov a pomáhať pri správe siete. Vďaka monitorovaniu, testovaniu, kontrole a vyhodnocovaniu v reálnom čase sú správcovia siete schopní získať informácie o výkonnosti sietového systému, vyhodnotiť kvalitu služieb (QoS) a zistiť poruchu siete. Vďaka napredujúcej technológií 5G a podpore technológie Internet vecí (IoT), sa veľkoplošné a vysokorychlostné siete sa stávajú súčasťou výskumu a vývoja s cieľom účinne zhromažďovať a analyzovať údaje o sieti.

Danú problematiku je potrebné riešiť z dôvodu predchádzania škodlivým útokom prostredníctvom predikcie na základe analýzy dát. V rámci riešenia tejto problematiky je dôležité navrhnúť, vytvoriť a implementovať bezpečnostné metódy na zabránenie takýchto útokov. Definícia pojmu sietová bezpečnosť je podľa autorov Xia a Wang [59] nasledovná: „Pomocou zachytávania a integrácie všetkých druhov informácií, ktoré odrážajú bezpečnostný stav, možno predpovedať trend bezpečnosti siete“ (Xia Wei-Wei a Wang Hai-Feng, 2010, str. 616). Predčasné odhalenie škodlivej činnosti zabezpečí lepšiu ochranu siete od budúcich trendov v tejto oblasti, ktoré prinášajú nové, komplexné a sofistikovanejšie útoky. Taktiež zabezpečuje vybudovanie nákladovo efektívnej stratégie v prípade nového útoku.

Prvá časť tejto práce je venovaná analýze dát z prostredia počítačových sietí. Vzhľadom na identifikáciu bezpečnostných rizík a útočnej premávky je potrebné analyzovať existujúce algoritmy a procesy zamerané na spracovanie dát tohto typu. Naším cieľom je analýza súčasného stavu problematiky a metód použiteľných pri analýze dát z prostredia počítačových sietí. V tejto práci sa venujeme analýze systémom na detekciu sietového narušenia - IDS, ich architektúre a spôsobu vyhodnocovania. Taktiež sa venujeme sietovým útokom, opisujeme typy sietových útokov a priblížime si spôsob ich odhalenia. Bližšie opisujeme strojové učenie a jednotlivé algoritmy strojového učenia. Zvlášť v rámci kapitoly 2.6 Strojové učenie sa venujeme neurónovým sietiam. V našej práci sme pri výbere zohľadňovali kritériá, ktoré sú kladené na dnešné moderné metódy dolovania v dátach. To nás priviedlo k umelej inteligencii - neurónovým sietiam. Tie poskytujú v

dnešnej dobe veľmi intuitívne a moderné riešenia. Táto metóda klasifikácie je jedna z najpresnejších. Klasifikačné metódy na báze stromov ako XGradient Boosting, náhodný les či rozhodovací strom majú tiež veľký potenciál. Na záver analýzy sa venujeme vybraným dátovým množinám, kde uvádzame výsledky experimentov iných autorov pre budúce možné porovnanie s výsledkami našej práce a metódy predspracovania týchto dátových množín.

V druhej časti tejto práce sa venujeme návrhu vlastného riešenia. Navrhнемe dva programové moduly. Prvý programový modul je určený na spracovanie rozsiahlych dátových množín a druhý programový modul je určený na aplikovanie metód strojového učenia na vybranej predspracovanej dátovej množine. Taktiež navrhнемe prostredie v ktorom sa budú programové moduly vyvíjať. Určíme funkcionálne a nefunkcionálne požiadavky, ktoré majú programové moduly spĺňať. Nakoniec na základe návrhu implementujeme navrhnuté programové moduly. Postup implementácie jednotlivých častí programových modulov taktiež detailnejšie opíšeme. V poslednej časti tejto práce overíme navrhnuté riešenie. Riešenie overíme na základe miery splnenia funkcionálnych požiadaviek. Na záver vykonáme experimentálne aj záverečné akceptačné testy a na základe výsledkov vyhodnotíme úspešnosť klasifikácie vybraných metód strojového učenia. Na základe výsledkov testov vyberieme najlepšie metódy strojového učenia a overíme ich na simulovannej siet'ovej premávke. Na základe týchto testov vieme potom určiť, že ktoré metódy strojového učenia sú najvhodnejšie pre aké typy predspracovania dátovej množiny. Výsledky interpretujeme vo forme percentuálnych úspešností, tabuliek a grafov.

Táto práca má dostatočne priblížiť, vysvetliť jednotlivé pojmy a súvislosti medzi nimi a uviesť čitateľa do danej problematiky.

## 2. Analýza problematiky

V tejto kapitole sa budeme venovať analýze systémov na detekciu sietových útokov (Intrusion Detection System - IDS) a priblížime jednotlivé prístupy detekcií sietových útokov.

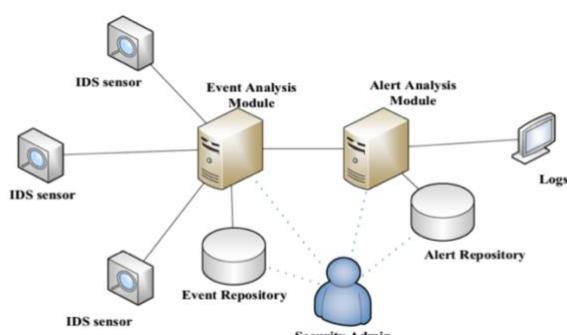
### 2.1. Typy systémov na detekciu sietových útokov

Na základe predpokladu, že správanie sa útočníka na sieti je odlišné od bežného správania sa používateľa môžeme takúto sietovú premávku identifikovať ako útočnú. Prostredníctvom skúmania anomalií v sieti je možné detegovať známe, ale aj neznámy typy útokov. Anomaliám sa venujeme v kapitole 2.1.1 Anomálne založené detekčné systémy.

Systémy na detekciu sietových útokov sú implementované ako druhá obranná línia popri autentifikácii používateľa a ďalších bezpečnostných mechanizmov. IDS je softvér, hardvér alebo ich kombinácia, ktorý monitoruje počítačovú sieť pre odhalenie škodlivých aktivít alebo narušení siete. Narušenie siete je akt odhalenia nepriateľského používateľa (útočníka), ktorý sa pokúša získať neoprávnený prístup do siete alebo sa snaží narušiť služby a odmietnuť služby legitímnym používateľom. Pri odhalení narušenia siete, IDS systémy vytvárajú správy pre správcov bezpečnosti siete, ktorí sa rozhodnú o ďalších postupoch zaobchádzania sa s narušením. Tieto systémy môžu byť nasadené priamo u používateľa siete alebo priamo integrované v sieti na analýzu sietovej premávky.

Bhattacharyya a Kalita [8] vo svojej knihe uvádzajú nasledujúce metódy detekcie anomalií v počítačovej sieti:

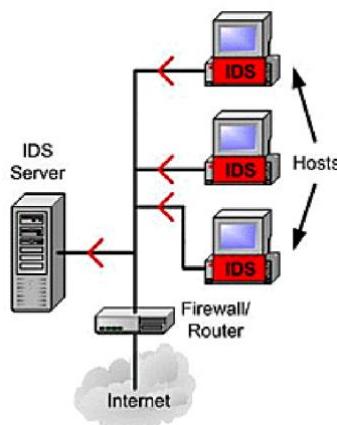
- **Intrusion Detection System (IDS)** – Systémy detekcie narušenia je nasadzovaný ako druhá obranná línia spolu s ďalšími preventívnymi bezpečnostnými mechanizmami, ako je autentifikácia používateľov a kontrola prístupu. Tento systém detekcie narušenia je založený na tom, že správanie útočníka je výrazne odlišné od správania bežného používateľa. Na nasledujúcom obrázku môžete vidieť všeobecnú architektúru IDS.



Obrázok 1 – Všeobecná architektúra IDS [41]

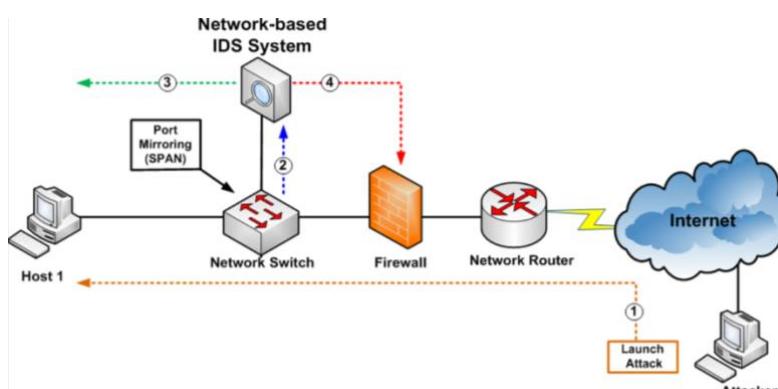
- **Host-based IDS (HIDS)** – Systém monitoruje vnútro výpočtového systému a sleduje, či niekto neobišiel bezpečnostnú politiku z vnútra (interne) alebo z vonka (externe). Podľa Jajish [24] HISD je softvérová aplikácia (agent) nainštalovaný na pracovných staniciach, ktoré sa majú monitorovať. Agenti monitorujú operačný systém a zapisujú údaje do protokolových súborov a/alebo spúšťajú poplach. Tento typ systému môže monitorovať iba jednotlivé pracovné stanice na ktorých sú agenti nainštalovaní, nemôže monitorovať celú sieť. HIDS môže zistiť internú aktivitu, ako napríklad program, ktorý pristupuje k zdrojom a pokúša sa k neoprávnenému prístupu.

Host Based IDS



Obrázok 2 – Architektúra HIDS [41]

- **Network-based IDS (NIDS)** – Keďže sieť je prepojená väčšinou s internetom pre komunikáciu so zvyškom sveta, tak NIDS číta všetky prichádzajúce pakety alebo toky a snaží sa nájsť podozrivé vzory. Podľa [24] Zvyčajne sa skladajú zo sietového zariadenia s kartou sietového rozhrania (NIC) pracujúci v promiskuitnom režime. V prípade ak je paket prepojený s podpisom útočníka, tak je generované upozornenie alebo je paket zaznamenaný do databázy.



Obrázok 3 – Architektúra NIDS [41]

Autori Liu, Yan a Pedrycz [31] vo svojej práci IDS ďalej rozdeľujú do päť typov (vid'. tabuľku č. 1):

- Anomaly-Based Intrusion Detection System (ABIDS)
- Knowledge-Based Intrusion Detection System (KBIDS)
- Specification-Based Intrusion Detection System
- Hybrid Intrusion Detection System (HIDS)
- Other Intrusion Detection System (OIDS)

Tabuľka 1 – Typy IDS [31]

<b>IDS</b>	<b>Popis</b>
ABIDS	Odvodzuje model (profil) podľa priateľných činností a správania a generuje poplach, ak sa monitorované činnosti alebo správanie sa výrazne odlišuje od tohto profilu.
KBIDS	Zachováva vzory konkrétnych útokov a spustí poplach ak sa pozorované udalosti zhodujú so vzormi.
SBIDS	Vyberá špecifikácie, ktoré definujú správne operácie siete s určitými obmedzeniami a identifikuje narušenie ak sa monitorované operácie odlišujú od špecifikácie.
HIDS	Je kombináciou ABIDS, KBIDS a SBIDS.
OIDS	Nepatrí do vyššie uvedených typov IDS.

### 2.1.1. Anomálne založené detekčné systémy

Anomálie v sietiach sa môžu vyskytnúť z viacerých dôvodov, napríklad z dôvodu prevádzky siete či z dôvodu škodlivej činnosti. Na základe takýchto výkyvov v sieti dokážeme relatívne ľahko identifikovať činnosť, ktorá je odlišná od bežnej činnosti siete a určiť, či je táto činnosť škodlivá alebo nie. Bhattacharyya a Kalita [8] vo svojej knihe tvrdia, že anomálie v sieti sú detektovateľné prostredníctvom strojového učenia, ktoré odhaluje dve hlavné oblasti/kategórie vplyvu anomálií na počítačovú sieť. Anomálie podľa [8] môžu ovplyvňovať výkonnosť a bezpečnosť siete.

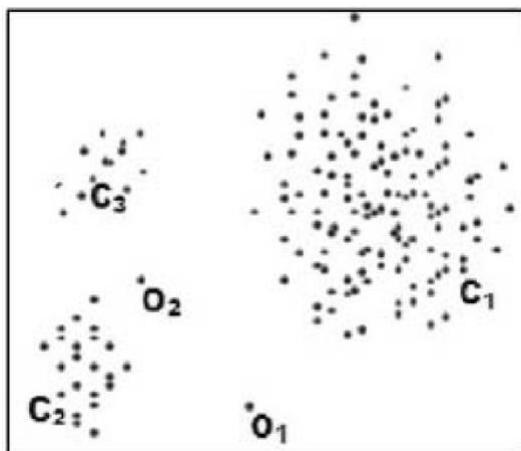
Našim hlavným cieľom je venovanie sa anomáliám spôsobujúcim bezpečnostné riziká v sieti, a to konkrétnie anomáliám spôsobené škodlivými aktivitami. Škodlivé činnosti v sieti môžu mať rôzne typy, ako sú bodové anomálie, kontextové anomálie či kolektívne anomálie.

Nasledujúci zoznam stručne opisuje predchádzajúce tri typy škodlivých činností:

- „**Bodová anomália:** Bodové anomálie sú prípady, ktoré sú mimoriadne alebo nezvyčajné vzhľadom na ostatné údaje. Napríklad mimoriadne výdavky na výpise kreditnej karty v

porovnaní s predchádzajúcimi transakciami. Na obrázku č. 4 môžete vidieť, že objekt  $O_1$  je izolovaný od inej skupiny objektov  $C_1$ ,  $C_2$  a  $C_3$ . Objekt  $O_1$  je bodová anomália.

- **Kontextová anomália:** V danom kontexte (napr. v rámci daného rozsahu), ak je inštancia anomálna alebo výnimočná. Na obrázku č. 4 môžete vidieť, že objekt  $O_2$  je izolovaný v kontexte skupiny objektov  $C_2$ . Objekt  $O_2$  je kontextová anomália.
- **Kolektívna anomália:** Ak sa zistí s ohľadom na dané normálne správanie, že skupina prípadov sa odchýli anomálne, celá skupina anomálnych prípadov sa označí ako kolektívna anomália. Na obrázku č. 4 je skupina  $C_3$  odlišná od skupín  $C_1$  a  $C_2$  z hľadiska počtu prípadov a kompaktnosti, a preto môže byť  $C_3$  označená ako kolektívna anomália.“ (Dhruba Kumar Bhattacharyya a Jugal Kumar Kalita [8], 2014, str. 46)



Obrázok 4 – Bodová, kontextová a kolektívna anomália [8]

Podľa [8], účel anomálne založených detekčných systémov je analýza, porozumenie, charakteristika sieťovej premávky, a zároveň identifikácia a klasifikácia abnormálnej premávky. Anomálne klasifikačné metódy existujú v štyroch kategóriách:

- **Dohliadaná anomálna detekcia** – Technika trénovania potrebuje trénovaciu dátovú množinu, ktorá má označenú normálnu a anomálnu sieťovú premávku. V prípade dát, ktoré sa nepodarilo zaklasifikovať (označiť) sú porovnané voči modelu už klasifikovaných tried a na základe výsledku sú zaradené do príslušnej triedy. Nevýhodou tejto techniky je, že anomálne dátá sú málo porovnávané voči normálnym triedam. Ďalej, je náročné presne a reprezentatívne klasifikovať anomálnu sieťovú premávku.
- **Čiastočne dohliadaná anomálna detekcia** – Technika, pri ktorej trénovacia dátová množina má označenú iba normálnu sieťovú premávku. Anomálne triedy nemajú označenie a tak využitie tejto techniky je viac aplikovateľná.

- **Nedohliadaná anomálna detekcia** – Technika nepoužíva trénovaciu dátovú množinu a tým pádom je dosiahnutie presnej detektie anomálnej sietovej premávky náročné. Väčšinou pri tejto technike sa vychádza z predpokladu, že normálna sietová premávka je vo väčšej miere zastúpená v testovacej dátovej množine ako anomálna.
- **Hybridná anomálna detekcia** – Technika kombinujúca dohliadanú a nedohliadanú metódu detektie. Táto technika má výhodu v tom, že vďaka dohliadanej technike má vysokú mieru odhalenia útokov a nízku mieru falošných poplachov. Na druhej strane, v prípade nedohliadanej technike má výhodu v odhalení neznámych útokov. Preto hybridná technika detektie sietových útokov je schopná identifikovať známe, ale aj neznáme sietové útoky.

Podľa Samrina a Vasumathi [41] anomálne založené detekčné systémy majú nasledovné výhody:

- Na identifikáciu nových útokov sa nevyžaduje aktualizácia databázy.
- Po nainštalovaní softvéru je potrebná údržba.
- Súbežne sleduje správanie sa siete a vytvára profily sietových aktivít.
- Najefektívnejšie identifikujte hrozby vo väčšom systéme.

a nevýhody:

- Abnormálne správanie sa siete v normálnej premávke neodošle upozornenie správcovi.
- Veľa falošných poplachov.

Ďalšími stratégiami odhalenia sietových útokov sú detekčné systémy založené na podpisoch (*Signature-based IDS*) podľa Warzyński a Kołaczek [56] a systémy na rozpoznanie zneužitia (*Misuse IDS*) podľa Saxena a spol. [44]. Detekčné systémy založené na podpisoch sú založené na podpisoch známych útokov a pravidel definovaných administrátorom. Takéto systémy môžu klasifikovať známe útoky porovnaním pozorovanej aktivity s uloženými vzormi, ale nemôžu identifikovať nové útoky. V prípade systémov na rozpoznanie zneužitia je najprv definované abnormálne správanie systému a potom všetky ostatné správania sú definované ako normálne. Je v rozpore s prístupom na detekciu anomalií, ktorý využíva opačný postup. Pri detekcii zneužitia je všetko neznáme, normálne správanie.

## 2.1.2. Charakteristiky IDS

Spôsob reakcie IDS systémov na siet'ové útoky môže byť dvojáká podľa Ahmed a spol. [1]: *pasívna* a *aktívna* reakcia. V prípade pasívnych detekčných systémov, systém pri odhalení siet'ového útoku priamo nereaguje na útok, ale nechá rozhodnutie na iný systém. V prípade aktívneho detekčného systému, systém pri odhalení útoku priamo reaguje na útok napríklad zablokovaním siet'ovej premávky pre detegovaného siet'ového útočníka.

Jedným z najdôležitejších cieľov NIDS je odhalenie útočnej premávky v reálnom čase. V prípade aktívnych IDS je táto skutočnosť možná, pretože priamo reagujú v momente detektie útoku. Žiaľ takéto systémy trpia vysokou mierou falošných poplachov, z čoho vyplývajú nasledovné nedostatky IDS [41]:

- **False positive:** Predpoved' falošných útokov. Ak je táto miera vysoká, potom normálny útok sa predpovedá ako útok.
- **False negative:** Vysoká miera falošne negatívnych hodnôt spôsobuje problém tak, že keď sa vyskytne narušenie siete, IDS nevytvorí žiadne upozornenie.
- **True positive:** Vyskytne sa vtedy, keď dôjde ku skutočnému útoku a IDS naň odpovie vyvolaním poplachu.
- **True negative:** Keď nenastane žiadny útok a IDS nevyvolá poplach.

**Poplach** [8] – Výstraha generovaná systémom NIDS, ktorá dostatočne a zmysluplne identifikuje dôvod, ktorý spôsobil udalosť poplachu a zdroj/ciel útoku. Mal by pomáhať správcovi systému alebo analytikovi pri určovaní vhodnej reakcie na konkrétné upozornenie.

Bhattacharyya a Kalita [8] vo svojej knihe zadefinoval nasledujúce otvorené výzvy čakajúce na vyriešenie:

- **Obmedzenie výkonnosti pri práci v reálnom čase** – Vyvinutý NIDS by mal v ideálnom prípade v reálnom čase zachytiť a skontrolovať každý jeden paket podľa aktuálneho siet'ového scenára pre vhodnú analýzu a presnú detekciu anomálie.
- **Zniženie falošného poplachu** – NIDS alebo iná metóda detektie by sa mala vyhnúť vysokej mieri hlásenia falošných útokov.
- **Redukcia dimenzie** – Vyvinúť vhodnú metódu na výber optimálneho súboru parametrov na detekciu anomálií bez znižovania výkonnosti detektie.

- **Zvýšenie výpočtovej sily** – Na zvládnutie sofistikovanejších a komplexnejších vrstvených útokov sú mechanizmy na ochranu siete postavené na existujúcom systéme s pridruženými výpočtovými modulmi, ktoré v prípade rapídneho vývoja vysokorýchlosného internetu sú kontraproduktívne a môžu spôsobovať uviaznutia či zníženie výpočtovej sily.
- **Generický systém** – Platformovo nezávislý systém či metóda.
- **Spracovanie sofistikovaných anomalií** – Aktualizovanie NIDS alebo iných detekčných metód na aktuálne anomálie, ktoré sa vyskytli v lokálnej sieti alebo na internete.

Garg a Maheshwari [16] ďalej uvádzajú:

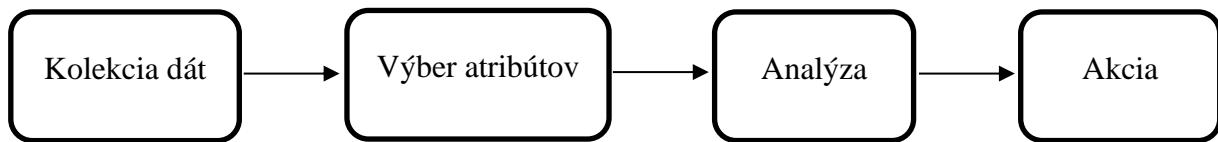
- **Špecifickosť** – Po identifikácii útoku musia byť k dispozícii dostatočné podrobne informácie, aby sa dosiahla lepšia reakcia.
- **Škálovateľnosť** – Možnosť aplikovať na veľké a malé siete.
- **A priori informácie** – Táto vlastnosť potrebuje vopred informácie týkajúce sa potenciálnych útočníkov a ich stratégií.

Bhattacharyya a Kalita ďalej spomínajú otvorené výzvy v súvislosti so spracovaním útokov ako: prispôsobovanie sa novým útokom, detekcia a spracovanie veľkých útokov, odhalenie útokov útočiace na sietovú infraštruktúru, ovládanie, resp. odolávanie útokom s vysokou mierou frekvencie a identifikovanie nových, vynaliezavých útokov.

Autori Bhattacharyya, Kalita a Liu a spol. sa však zhodnú na rovnakom názore/závere, že nedostatok spoločlivých a kvalitatívnych údajov je veľkou prekážkou v presnosti klasifikácie útoku. Väčšina dátových množín je neúplná, keďže množstvo systémov založených na anomaliách bolo testované na množine vytvorených údajov a tým pádom hodnotenie systému je obmedzené kvalitou údajov. Taktiež sa znížuje efektívnosť zhromažďovania údajov súvisiacich s bezpečnosťou v dôsledku duplicity údajov a veľkosti ich množstva. Dátovým množinám sa venujeme v kapitole 2.7 Dátové množiny.

## 2.2. Architektúra IDS

Tiwari a spol. [53] vo svojom článku opisujú štyri základné moduly, z ktorých sa skladá vnútorná štruktúra každého IDS. Tieto moduly sú nasledovné: kolekcia dát, výber vhodných atribútov, analýza, akcia (viď. obrázok č. 5).



Obrázok 5 – Funkcionality IDS [53]

1. **Kolekcia dát** – Modul pre vstupné dátá pre IDS. Dáta sú zbierané a ukladané do súboru pre ďalšiu analýzu.
2. **Výber atribútov** – Modul pre výber vhodných ohodnotených atribútov z dátovej množiny.
3. **Analýza** – Modul, ktorý pre pravidlami riadené IDS systémy analyzuje dátá a v prípade prichádzajúceho sietového toku dát sú dátá porovnávané voči vzorom a podpisom. V prípade anomálnych IDS systémov je analyzované správanie sa siete a následne je aplikovaný matematický model.
4. **Akcia** – Modul definujúci reakciu na útok. Reakcia môže byť informatívna pre systémového/sietového administrátora alebo aj samotný IDS systém (aktívny) môže rozhodnúť o akcii (napríklad zahodenie paketov).

[53] uvádzajú tri základné komponenty IDS: senzor, backend, frontend. Túto skutočnosť potvrdzuje aj práca autora Schaelicke a spol. [45], ktorí tvrdia, že senzor je častokrát implementovaný ako univerzálny počítačový systém so softvérom na detekciu narušenia siete. Samostatný systém môže byť hostiteľom databázy alebo podobného softvéru na zabezpečenie dlhodobého ukladania dát pre ďalšiu možnosť analýzy.

1. **Senzor** – Hlavnou funkciou tohto modulu je detekcia a oznamovanie. Obsahuje rozhranie pre kolekciu dát a rozhranie pre sietový manažment. Priebeh funkcionality senzora je nasledovný: senzor počúva na sietovom rozhraní a ukladá zachytené dátá do buffer pamäte. Následne nástroj na detekciu analyzuje pozbierané dátá a spustí sa analýza sietových protokolov. V tomto komponente prebieha aj detekcia podpisov a anomálií.

2. **Backend** – Predstavuje hlavnú funkciu IDS. Každá senzorom zachytená udalosť je ukladaná do databázy a následne vyhodnocovaná, že aká akcia sa podnikne systémom.
3. **Frontend** – Modul predstavuje používateľské rozhranie a tzv. *Command & Control*. Údaje, udalosti a log zápisu z backend-u sa zobrazia v tomto komponente, kde používateľ ich môže manažovať.

## 2.3. Existujúce nástroje

Bhattacharyya a Kalita [8] vo svojej knihe opísali veľké množstvo (20) nástrojov na detekciu útokov alebo narušenia, ktoré identifikujú známe aj neznáme útoky pomocou štatistických metód, dolovania v dátach alebo softvérových prístupov. V nasledujúcim zozname opíšeme niektoré vybrané nástroje.

1. **Bro** – Voľne dostupný nástroj pre platformu Linux, ktorý pasívne monitoruje sieťovú prevádzku a snaží sa identifikovať narušenia siete v reálnom čase. Je schopný identifikovať útoky založené na podpisoch, útoky orientované na udalosti a niektoré nezvyčajné útoky. Umožňuje tiež sledovanie správania, viacvrstvovú analýzu, presadzovanie politík a činnosti pri registrácii paketov.
2. **Snort** – IDS je založený na ľahkých podpisoch, ktorý kontroluje návštevnosť protokolu TCP/IP s cieľom identifikovať narušenia siete na základe pravidel funkcií a zhody obsahu.
3. **HIDE** – Hierarchický systém založený na anomaliách, vyvinutý pomocou štatistického modelovania a neurónových sietí. Skladá sa z niekoľkých úrovní, kde každá vrstva obsahuje niekoľko detektorov narušenia (*Intrusion Detection Agents* - IDA), ktoré sú IDS komponentmi, ktoré monitorujú činnosti hostiteľa alebo siete.
4. **CAD** – Používa dátovú štruktúru - stromy (*Change Aggregation Trees* - CAT) na detekciu distribuovaných záplavových útokov (DDoS) na úrovni toku. Hlavným cieľom je odhaliť náhle zmeny prevádzky vo viacerých sieťových doménach.
5. **MINDS** – Populárny nástroj založený na metódoch dolovania v dátach. Využíva *Netflow v. 5* na zbiera dát zo sietovej premávky pomocou nástrojov na odchytávanie toku v sieti. Pred vyhodnotením anomálnej premávky sa tieto dát prečistia od nezaujímových dát, resp. vzorov sietovej premávky. Systém na detekciu anomalií využíva *outlier* algoritmus, ktorý priradí jednotlivým sieťovým spojeniam anomálnu hodnotu, na základe ktorej sa vyhodnocuje podozrivá sieťová premávka.
6. **N@G** – Hybridný IDS s podporou odhalovania útokov na strane klienta (host-based) a siete (network-based). Analyzuje premávku v reálnom čase na základe štatistických

techník na strane klienta. Zastrešuje kontrolu používateľského rozhrania a manažmentu dát. Podporuje ochranu Layerd Service Provider (LSP) Domain Name Server (DNS) a dokáže dynamicky aplikovať Access Control List (ACL) pre blokovanie sieťovej premávky, resp. útoku.

Po analýze existujúcich nástrojov na bezpečnosť sietí autori knihy [8] prišli k záveru, že:

- Väčšina existujúcich NIDS je závislý od viacerých vstupných parametrov používateľa a ich výkon je veľmi citlivý na tieto parametre.
- Takmer všetky NIDS na báze anomalií pracujú takmer v reálnom čase alebo offline. Navyše väčšina trpí veľkým počtom falošných poplachov.

Autori knihy [8] identifikovali nasledujúce nedostatky v existujúcich riešeniaciach vhodné na ďalší výskum:

- Vyvinúť detekčný systém útokov v reálnom čase pre útoky s nízkou a vysokou frekvenciou DDoS (Distributed Denial of Service) útokov bez ovplyvnenia používateľov alebo bežných služieb.
- Vyvinúť NIDS založený na detekcii útokov pomocou anomalií, ktorého schopnosť detegovať útoky závisí od minimálneho počtu užívateľských parametrov a je schopný zaobchádzať so známymi aj neznámymi útokmi v reálnom čase s minimálnym počtom falošných poplachov.

## 2.4. Spôsob vyhodnocovania IDS

Hodnotenie je dôležité pre pochopenie kvality použitého modelu alebo techniky. Na základe získaných hodnôt môžeme ladiť použitie parametrov v iteratívnom procese učenia sa, pre výber najpriateľnejšieho modelu alebo techniky z daného súboru modelov alebo techník. Preto existuje niekoľko kritérií na hodnotenie modelov a techník.

Podľa Schaelicke a spol. [45] výkon systému detektie narušenia siete je charakterizovaný pravdepodobnosťou, že útok je detegovaný v kombinácii s počtom falošných upozornení. Rovnako dôležitá je však schopnosť systému spracovať prevádzku pri maximálnej rýchlosťi, ktorú ponúka sieť s minimálnou stratou paketov. Významná strata paketov môže zanechať množstvo nezistených útokov a zhoršiť celkovú efektívnosť systému.

Autorka Bhardwaj [7] vo svojom diele uvádza, že pri klasifikačných problémoch je prirodzené merať výkon klasifikátora z hľadiska chybovosti. Klasifikátor predpovedá triedu každej inštancie ak je správne, počíta sa ako úspech, ak nie, tak ide o chybu. Miera chybovosti je len podiel chýb

vykonaných v celom súbore inštancií a meria celkový výkon klasifikátora. Najznámejšie metódy vyhodnotenia výkonnosti klasifikátora sú nasledovné:

- **Cross-validation** – Rozdelenie dátovej množiny na menšie celky ( $k$  podmnožín) pre odhad rizika každého algoritmu. Časť údajov (trénovacia vzorka) sa používa na tréning každého algoritmu a zostávajúca časť (validačná vzorka) sa používa na analýzu. To znamená, že jedna podmnožina  $k$  sa použije na testovanie a ostatné  $k-1$  podmnožín na trénovanie. Tento postup sa opakuje pokiaľ sa nepoužije každá podmnožina  $k$  na test. Nakoniec sa výsledky testov skombinujú do výsledného odhadu a vyberie sa algoritmus s najmenším odhadovaným rizikom.
- **Holdout metóda** – Najjednoduchší druh krížovej validácie. Dátový súbor je rozdelený do dvoch skupín (trénovacia a validačná vzorka). Klasifikátor predpovedá výstupné hodnoty pre dátá v testovacej sade. Chybovost klasifikátora sa spriemeruje v absolútnej hodnote a používa sa na vyhodnotenie modelu. Hodnotenie môže do značnej miery závisieť od toho, ktoré dátá skončia v trénovacom súbore a ktoré skončia v testovacom súbore dát. Preto výsledok vyhodnotenia môže mať vysokú mieru variability.
- **Random sub-sampling** – Táto metóda vychádza z predchádzajúcej metódy, ktorá sa môže opakovať niekoľko krát kvôli zlepšeniu odhadu výkonnosti klasifikátora. Nevýhodou tejto metódy je, že nemá žiadnu kontrolu nad tým koľkokrát sa každý záznam použije na testovanie a tréning.
- **K-fold cross-validation** – Metóda na vylepšenie holdout metódy. Dátový súbor je rozdelený do  $k$  podmnožín a metóda holdout sa opakuje  $k$ -krát. Zakaždym, keď sa jedna z  $k$  podmnožín použije ako množina testov, tak ostatné podmnožiny  $k-1$  sa zostavia tak, aby vytvorili tréningovú množinu. Potom sa vypočíta priemerná chyba vo všetkých pokusoch  $k$ . Nevýhodou tejto metódy je, že tréningový algoritmus musí byť opakovaný  $k$ -krát.
- **Leave-one-out** – K-násobná krížová validácia, pričom  $k$  sa rovná  $n$ , kde  $n$  je počet inštancií v dátovej množine. Testuje sa vždy iba na jednom zázname a ostatné slúžia ako trénovacia vzorka. Týmto spôsobom sa zabezpečuje trénovanie na najväčšej možnej vzorke dát. Výpočet priemernej chybovosti sa použije na vyhodnotenie modelu.
- **Bootstrap** – Záznam, ktorý sa vybral na trénovanie sa vloží naspäť do pôvodného súboru záznamov, tým pádom môže byť s rovnakou pravdepodobnosťou znova vybraný.
- **Confusion matrix** – Binárny klasifikačný model klasifikuje každú inštanciu do jednej z dvoch tried: správna a chybová trieda. Z toho vyplývajú štyri možné klasifikácie pre každý prípad: skutočný pozitívny (*true positive* - TP), skutočný negatívny (*true negative* - TN), falošný pozitívny (*false positive* - FP) alebo falošný negatívny (*false negative* - FN). Tieto

štyri klasifikácie znázorňuje kontingenčná tabuľka, resp. Confusion matrix. (viď. tabuľku č. 2).

Tabuľka 2 – Kontingenčná tabuľka

Kontingenčná tabuľka		Predpovedaná trieda	
		Negatívny (0)	Pozitívny (1)
Skutočná trieda	Negatívny (0)	TN	FP
	Pozitívny (1)	FN	TP

Vysvetlenie klasifikácií:

- **TP** – Predpovedáme triedu 1, zatialčo skutočná trieda je 1. To znamená, že správne predpovedáme, že trieda je pozitívna.
- **FP** – Predpovedáme triedu 1, zatialčo skutočná trieda je 0. To znamená, že nesprávne predpovedáme, že trieda je pozitívna.
- **FN** – Predpovedáme triedu 0, zatialčo skutočná trieda je 1. To znamená, že nesprávne predpovedáme, že trieda je negatívna.
- **TN** – Predpovedáme triedu 0, zatialčo skutočná trieda je 0. To znamená, že správne predpovedáme, že trieda je negatívna.

Z maticy môže odvodiť rad ďalších výkonnostných metrík modelu:

- **Presnosť** (precision) – je pomer správne predpovedaných pozitívnych pozorovaní k celkovým predpokladaným pozitívnym pozorovaniám a počítá sa nasledovne:

$$\text{Presnosť} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **Odvolanie** (recall) – je pomer správne predpovedaných pozitívnych pozorovaní ku všetkým pozorovaniám v skutočnej triede a počítá sa nasledovne:

$$\text{Odvolanie} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **F1-skóre** (F1-score) – je vážený priemer presnosti a spätného volania. Toto skóre berie do úvahy falošne pozitívny aj falošné negatívny a počítá sa nasledovne:

$$\text{F1 - skóre} = \frac{2 * (\text{Odvolanie} * \text{Presnosť})}{\text{Odvolanie} + \text{Presnosť}}$$

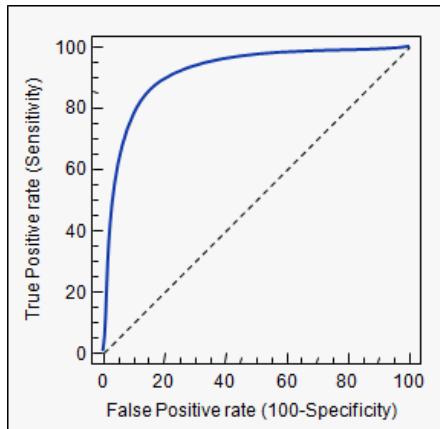
- **Správnosť** (accuracy) – predstavuje pomer správneho predpovedaného pozorovania k celkovým pozorovaniám a počítá sa nasledovne:

$$\text{Správnosť} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

*Receiver Operating Curves* (ROC) intuitívnejším a robustnejším spôsobom vizuálne sprostredkúva rovnaké informácie ako confusion matrix. ROC je dvojrozmerný graf, ktorý vizuálne zobrazuje výkonnostný a výkonový kompromis klasifikačného modelu. Pre skonštruovanie ROC krivky je potrebné zaviesť dve nové výkonnostné metriky: skutočná pozitívna miera (*True positive rate* - TPR), ktorej vzorec je rovnaký ako pre *odvolanie* a falošná pozitívna miera (*False positive rate* - FPR). Táto metrika zodpovedá podielu negatívnych pozorovaní, ktoré sa mylne považujú za pozitívne vzhľadom na všetky negatívne pozorovania. Vzorec pre falošnú pozitívnu mieru je nasledovný:

$$\text{Falošná pozitívna miera} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

Grafy ROC sú konštruované vykreslením skutočnej pozitívnej miery proti falošne pozitívnej miere (vid'. obrázok č. 6).



Obrázok 6 – ROC krvka [46]

Autori Zaman a Lung [60] vo svojej práci vyhodnotili výkonnosť siedmich techník strojového učenia na dátovej množine Kyoto 2006+ a zistili, že ROC je vhodnejšia na vyhodnotenie techník ako metriky presnosť, odvolanie a správnosť. Mieru presnosti modelu môžeme odvodiť zo zakrivenia ROC krivky. Čím je ROC krivka bližšie k ľavému hornému rohu, tým je vyššia celková presnosť modelu.

Podľa Bhardwaj [7] na výber zaujímavých pravidiel zo súboru všetkých možných pravidiel je možné využiť obmedzenia rôznych meradiel významu a prínosu. Najznámejšie sú:

- **Podpora (support)** – Vyjadruje pravdepodobnosť, ako často sa v súbore údajov objavuje množina položiek. Je definovaný ako podiel transakcií  $T$  v súbore údajov, ktoré obsahujú množinu položiek  $X$ .

- **Spoločnosť (confidence)** – Vyjadruje podiel tých inštancií, ktoré vyhovujú pravidlu  $(X \cup Y)$  oproti tým, na ktoré sa dá aplikovať pravidlo X.

$$\text{conf}(X \Rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)}$$

Na meranie účinnosti IDS bolo navrhnutých niekoľko metrik podľa Kumar [27]. Tieto metriky môžu byť rozdelené do troch tried a to, prahová (*threshold*), hodnotiaca (*ranking*) a pravdepodobnostná (*probability*) trieda.

Prahové metriky zahŕňajú mieru klasifikácie (*Classification rate - CR*), F-meranie (*F-measure - FM*) a náklady na príklad (*Cost per example - CPE*). Nie je dôležité, ako je blízka predikcia k prahovej hodnote, dôležité je ak je nad alebo pod prahovou hodnotou. Hodnoty prahovej triedy ležia v rozsahu 0 až 1.

Trieda hodnotenia zahŕňa FPR, detekčnú frekvenciu (*Detection rate - DR*), presnosť (*Precision - PR*), oblasť pod krivkou ROC (AUC) a schopnosť detekcie narušenia (*Intrusion detection capability - CID*). Tieto metriky merajú ako dobre sú inštancie útoku usporiadane pred normálnymi inštanciami a môžu byť interpretované ako súhrn výkonnosti modelu na všetkých možných prahoch.

Do triedy pravdepodobnosti patrí chyba strednej hodnoty (*Root mean square error - RMSE*). Hodnota RMSE leží v rozsahu 0 až 1. Trieda je minimalizovaná, keď sa predpovedaná hodnota pre každú triedu útoku zhoduje so skutočnou podmienenou pravdepodobnosťou, že táto trieda je normálna.

V prípade metód hodnotenia klastrovaním je vyhodnotenie oveľa náročnejšie. Jediný spôsob, ako realisticky vyhodnotiť klastrovanie je, že či sa výsledok zhlukovania v kontexte aplikácie ukáže ako užitočný. Klastrovanie možno vyhodnotiť z perspektívy dĺžky popisu pomocou princípu *Minimum Description Length (MDL)*. Princíp MDL spočíva v tom, že najlepšia hypotéza pre daný súbor údajov je tá, ktorá vedie k najlepšej kompresii údajov. Technika klastrového učenia rozdeľuje tréningový súbor do  $k$  klastrov. Najlepšie klastrovanie podporí najefektívnejšie kódovanie.

## 2.5. Sieťové útoky

Podľa Liu a spol. [31] sieťový útok využíva diery v sieťovom systéme, v chybných protokoloch, v chybách hardvéru alebo softvéru a údaje sieťového systému pri neautorizovanom správaní. Sieťová komunikácia je v súčasnosti konfrontovaná so štyrmi druhmi bezpečnostných hrozieb: odpočúvanie informácií, prerušenie komunikácie, falšovanie správ a falšovania informácií. Ďalej sa tieto štyri typy útokov kategorizujú do dvoch kategórií a to, aktívne a pasívne útoky. Zachytávanie informácií je pasívny útok. Prerušenie komunikácie, manipulácia so správou alebo útok

falšovania je aktívny útok. V súčasnosti predstavuje režim siet'ového útoku viaceru prostriedkov, voči čomu sa ľudia ľažko bráňia. Tieto režimy by mohli byť rozdelené do štyroch kategórií: odmietnutie služby útoku, využitie typu útoku, zhromažďovanie informácií útoku, falošné informácie útoku. Potom sa môžeme zameriavať na niekoľko základných útočných prostriedkov, ako je napríklad: IP Spoofing Attack, ARP Attack, UDP Flooding Attack, TCP SYN Flooding Attack, ICMP Flooding Attack atď.

Sheela a spol. [50] vo svojej práci uvádzajú, že IDS využíva rôzne techniky na detekciu vniknutí do počítačovej siete. Používa na detekciu útočníkov jednoduchú techniku alebo kombináciu techník. Tieto techniky zahŕňajú detekciu anomálií, detekciu zneužitia, monitorovanie cieľa a špináž.

### **2.5.1. Detekcia anomálií**

Podľa [50] táto technika uchováva normálne správanie počítačovej siete, ako sú informácie o siet'ových paketoch, informácie o softvéri, systémové log udalosti, informácie o operačnom systéme, informácie o jadre operačného systému (kernel) atď. Ak sa deteguje rozdiel vo vyššie uvedených parametroch, tak sa zistí anomália a vygeneruje sa alarm. Detekcia anomálií je užitočná pri zisťovaní podvodov, vniknutí do siete a pri iných nezvyčajných činnostach v systéme. Detekcia anomálie je označovaná aj ako detekcia založená na správaní sa. Identifikuje odchýlky systému od normálneho správania. Táto metóda má schopnosť odhaliť nové a neznáme útoky analýzou údajov o audite.

Žiaľ, táto metóda má vysokú mieru falošného poplachu. Niekoľko možností môže byť legitímne správanie systému klasifikované ako anomálne a označené ako útok.

### **2.5.2. Detekcia zneužitia**

Podľa [50] táto technika ukladá sekvenciu vzorov, útočné signály, vzory narušenia atď. do databázy. Uložené udalosti zo systému sa porovnávajú s uloženými informáciami v databáze zistených útokov. Ak sa zistí zhoda, systém vygeneruje alarm. Keďže táto metóda porovnáva podpisy, niekoľko možností môže byť označené ako detekcia založená na podpisoch. Tieto techniky automaticky aktualizujú svoju databázu na rôznych vstupných údajoch tak, aby zahŕňali nové typy útokov. Techniky detektívania zneužívania majú vysoký stupeň presnosti pri detekcii známych útokov a ich variantov. Tieto techniky však nedokážu zistiť neznáme/nové útoky, pretože sú závislé od existujúcich podpisov.

### **2.5.3. Monitorovanie cieľa**

Podľa [50] táto technika hľadá modifikáciu na špecifických súboroch. Monitorovanie cieľa nehľadá anomálie. Funguje to ako korekčná kontrola, ktorá obnovuje súbor potom, čo bol súbor

modifikovaný útočníkom. Využíva *cryptographic hash computing* na obnovenie upraveného obsahu. Táto technika je ľahko implementovateľná, pretože neustále sledovanie prevádzky správcom nie je potrebný. Posielanie alarmu do siete alebo do systému sa vykonáva vtedy, keď existuje nesúlad údajov (check sum). Výpočet check sum môže byť vypočítaný v rôznych intervaloch.

#### 2.5.4. Špionáž

Podľa [50] táto technika deteguje útočníkov, ktorí zostávajú v sieti po dlhú dobu. Vo všeobecnosti útočníci dlhodobo kontrolujú zraniteľnosť systému a otvárajú porty a čakajú na ďalšiu dlhú dobu útoku. Táto technika kontroluje všetky takéto metodické útoky zhromažďovaním širokej škály údajov o celom systéme. Technika vyžaduje veľké množstvo vzoriek odobratých z rôznych počítačov a sietí na objavenie takýchto útokov. Na tento účel kombinuje detekciu anomalií a detekciu zneužitia.

#### 2.5.5. Typy siet'ových útokov

Autori Ananthi a Vengatesa [4] vo svojej práci vysvetľujú ako funguje jeden z najčastejších počítačových útokov, Denial of Service (DoS). V prípade DoS na zaplavenie servera paketmi (TCP/UDP) sa používa jeden počítač a jedno internetové pripojenie. Distributed DoS (DDoS) je botnet útok a je jedným z typov záplavového útoku. Počas tohto útoku sa útočné uzly pokúšajú naraz napadnúť jeden uzol (zvyčajne server). Namiesto jedného počítača a jedného internetového pripojenia využíva útok DDoS mnoho počítačov a mnoho pripojení. Počítače za takýmto útokom sú často distribuované po celom svete. Hlavným rozdielom medzi DoS a DDoS je, že cieľový server bude preťažený stovkami alebo dokonca tisícami požiadaviek v prípade útoku DDoS. V dôsledku toho dochádza k pretečeniu pamäte a odmietnutiu poskytovanej služby (nemožno ďalej poskytovať danú službu). DoS ďalej vysvetľujú vo svojej práci autori Hussain a Mishra [21]. DoS je typ útoku, kedy útočník zahlcuje pamäťové prostriedky a tým pádom zabránia, aby slúžili legitímnym siet'ovým požiadavkám. Takto napadnutý systém odopiera užívateľom prístup k počítaču alebo iným službám poskytované počítačom. DoS útok je iniciovaný tromi spôsobmi:

1. Zneužívanie legitímnych vlastností počítača.
2. Zacielenie implementačných chýb.
3. Využitie nesprávnej systémovej konfigurácie.

Útočník po úspešnom dokončení útoku ďalej poskytuje iné (svoje) služby, ktoré sú nedostupné pre autentické použitie a sú založené na rovnakých princípoch ako DoS útok, napríklad:

*apache, smurf, neptune, ping of death, back, mail bomb, UDP storm* atď. DoS je jedným z útokov ktorý klasifikuje aj dátová množina NSL-KDD, ktorej sa venujeme v kapitole 2.7 Dátové množiny.

Ďalšie známe útoky podľa [21] sú:

- **Remote to Local (R2L)** – Týka sa neoprávneného prístupu od vzdialeného počítača. Útočník útočí na vzdialene umiestnený počítač odoslaním paketov cez internet. Útok využíva privilégiá, ktoré by mal mať lokálny používateľ na počítači. Príklady takýchto útokov sú: *xlock, xnsloop, phf, sendmail, dictionary* atď.
- **User to Root (U2R)** – Je spojený s neoprávneným prístupom lokálnych privilégií super používateľa (root). Pri týchto typoch útokov útočník začína v systéme s bežným používateľským kontom so snahou zneužiť zraniteľné miesta v systéme na získanie privilégií super užívateľov. Príklady: *perl, Xstream* atď.
- **Probe** – Útočník skenuje počítač alebo sietové zariadenia na odhalenie zraniteľných alebo slabých miest, ktoré sa neskôr môžu zneužiť, aby sa narušil systém. Táto technika je primárne spojená s dolovaním v dátach ako: *satan, saint, portsweep, mscan, nmap* atď.
- **Scan** – Podľa Al-Jarrah a Arafat [2] skenovanie portov sa pokúša objavíť spustené služby na hostiteľskom počítači alebo sa pokúsi overiť dostupnosť určitej služby. Je dobre známe, že každá sietová aplikácia bežiaca na počítači má jedinečné číslo portu, na ktoré počúva, ako napríklad port 80 pre prehliadanie webu. Zistením, ktoré služby sú spustené môže byť spustený určitý útok proti objavenej službe. Techniky skenovania portov sú rozdelené do troch typov na základe portov a hostiteľov:
  1. **Jeden hostiteľ – rôzne porty:** Útočník prehľadáva rôzne porty na určitom hostiteľovi, čo je typické správanie skenovania portov. Poradie portov nie je dôležité, skenovanie môže byť sekvenčné alebo náhodné.
  2. **Rôzny hostitelia – jeden port:** Útočník prehľadáva viacero hostiteľov naraz s rovnakým číslom portu. Tento útok je spustený proti sieti hostiteľov, ktorí hľadajú hostiteľov, ktorí prevádzkujú určitú službu ako DNS, SMTP alebo HTTP.
  3. **Rôzny hostitelia – rozdielne porty:** Útočník prehľadáva viacero hostiteľov naraz a každý hostiteľ je skenovaný iným portom. Toto je pokročilá technika skenovania portov, ktorá sa snaží skryť svoju aktivitu iniciovaním náhodných skenov portov medzi náhodnými hostiteľmi. Tento útok je najkomplikovanejším skenovaním portov.

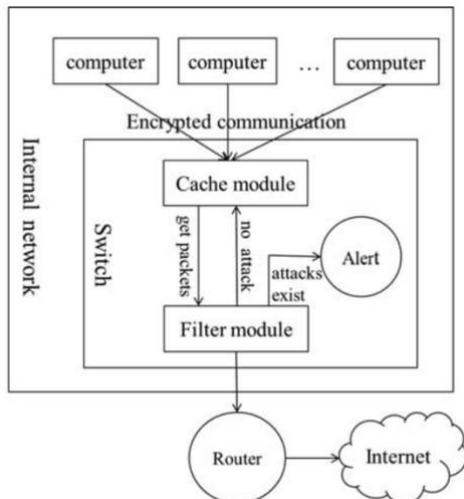
Najznámejšie útoky na skenovanie portov sú: TCP scanning, ACK scanning, UDP scanning, FIN scanning, NULL scan, X-mas a UDP/ICMP Error a ďalšie.

- **Eavesdropping** [50] – Útočník monitoruje komunikáciu iných ľudí neautorizovaným spôsobom. To môže byť napríklad odpočúvanie telefónnych hovorov, prezeranie e-mailov a správ a ďalších internetových služieb. Odpisluch je ľahké zistiť, pretože nemá vplyv na normálnu prevádzku siete. Keďže je ľahké odhaliť tento typ útoku, tak je vo všeobecnosti najväčším problémom, ktorému väčšina správcov čelí v podniku. Použitím silných šifrovacích schém môžu byť dátá chránené pred týmto typom útoku.
- **Man-in-the-Middle** [50] – Pri tomto útoku útočník zachytáva konverzáciu medzi dvomi stranami a vydáva sa za nich, čím získa prístup k dôležitým informáciám. Obe strany sa domnievajú, že spolu priamo komunikujú, aj keď útočník sa nachádza v strede ich konverzácie. Tento typ útoku je najčastejšou hrozbou pre online bezpečnosť, pretože útočníkovi je umožnené zachytiť a upraviť citlivé informácie v transakciách v reálnom čase. Útočník môže tiež zneužiť chyby zabezpečenia v konfiguráciách zabezpečenia siete.

### 2.5.6. Odhalenie sietových útokov

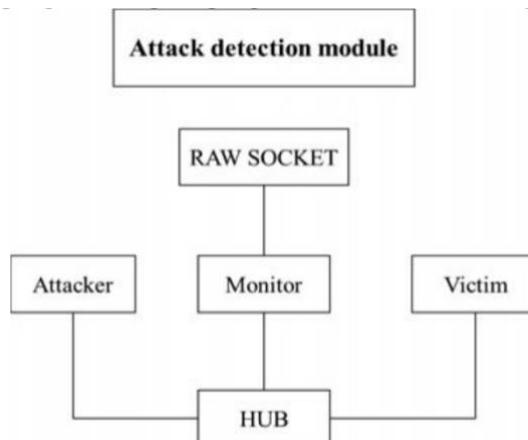
Sietový útok odkazuje na zneužitie chýb a nedostatkov v bezpečnostných nastaveniach siete, chybných protokolov, chybu v hardvéri alebo softvéri, násilný útok na hardvér alebo softvér či na údaje pri neautorizovanom správaní sa v sietovom prostredí. V súčasnosti predstavuje sietový útok viaceru prostriedkov, voči čomu sa ľudia len ľahko bránia. Preto je potrebné vyvinúť obranný mechanizmus na ochranu sietovej premávky voči škodlivej činnosti, tak ako aj autori článku Li a spol. [29].

Li a spol. navrhli sietový modul detektie útokov. Modul je založený predovšetkým na filtrovaní paketov. Rozpoznáva typ sietového útoku a má aj funkciu monitorovania siete, či funkciu upozornenia, ktorá nemá vplyv na výkon základnej sietovej komunikácie. Na nasledujúcom obrázku č. 7 môžete vidieť návrh platformy na odhalenie sietových útokov.



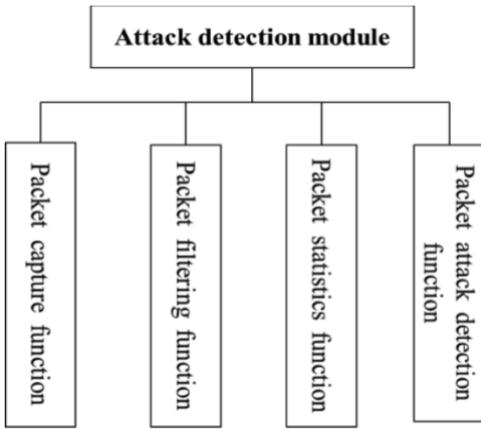
Obrázok 7 – Návrh platformy na odhalenie sietových útokov [29]

Modul najskôr prijíma pakety na detekciu útoku, ktoré sú následne filtrované vo filtračnom module (Filter module). Potom podľa charakteristík útoku siete, sa analyzujú pakety. Vývojové prostredie sa skladá z troch počítačov (útočník, obet a detektor), ktoré sú prepojené HUB-om, ako je znázornené na obrázku č. 8. Na detektore beží modul detekcie útoku. Tento spôsob spojenia je v súlade so skutočnou aplikáciou detekčného stroja, ktorý nebráni počítačom odosielat a prijímať dátové pakety. Pod inštaláciou operačného systému Linux je zabezpečená podpora raw socketov na zachytenie všetkých ethernet rámcov.



Obrázok 8 – Prostredie odhalenia útoku [29]

Na obrázku č. 9 je znázornená konštrukcia modulu na detekciu sietových útokov, ktorá slúži predovšetkým na zachytenie dátových paketov, ich filtrovanie, robenie dátovej štatistiky a na odhalenie útoku.



Obrázok 9 – Štruktúra modulu na detekciu sieťových útokov [29]

## 2.6. Strojové učenie

Strojové učenie môže byť klasifikované mnohými rôznymi spôsobmi, ale najčastejšie sa rozlišuje strojové učenie pod dohľadom a bez dohľadu. Strojové učenie pod dohľadom je potrebné naučiť na sade príkladov pre danú problematiku, pričom strojové učenie bez dohľadu dokáže sa sám naučiť vzory z poskytnutých údajov bez akéhokoľvek ľudského zásahu alebo poradenstva.

Bhattacharyya a Kalita [8] vysvetľujú, že strojové učenie s dohľadom sú metódy prediktívneho modelu pre normállové a anomálne triedy, kedy model porovnáva dátové inštancie s modelom pre určenie do ktorej triedy patrí. Na rozdiel strojové učenie bez dohľadu predpokladá, že normállové prípady sú oveľa častejšie ako anomálne prípady a anomálne prípady sú štatisticky odlišné od bežných prípadov. Avšak ak tieto predpoklady nie sú pravdivé, tak takéto metódy vyhodnocovania trpia vysokými falošnými poplachmi.

### Strojové učenie s dohľadom:

- Rozhodovacie a regresné stromy
- Klasifikačné a regresné stromy (CART)
- Podporné vektorové stroje (SVM)

### Strojové učenie bez dohľadu:

- Algoritmy hierarchického zhlukovania
- Algoritmy zhlukovania založené na hustote
- Modulárne algoritmy
- Softvérové algoritmy založené na soft computingu

V prípade strojového učenia bez dohľadu existuje mnoho ďalších typov, ktoré sme vyšie nezahrnuli napríklad: Skryté Markov modely, Algoritmus maximalizácie očakávaní, Soft computing, Rough sets, Evolučný algoritmus a ďalšie.

[8] NIDS systém vyžaduje ľudský zásah vo vytvorení, overení a nasadení pravidiel. Na riešenie tohto problému boli zavedené NIDS s anomáliami, ktoré používajú algoritmy strojového učenia. Problém falošných poplachov však stále postihuje väčšinu takýchto systémov. Preto vhodná kombinácia metód strojového učenia pre túto problémovú oblasť môže viesť k lepšiemu rozvoju NIDS.

Na využití umelej inteligencie (UI) – neurónových sietí v prostredí počítačových sietí, konkrétnie pre IDS sa zhodnú aj autori článku [30], Li a Dong. Tvrdia, že UI má mnoho užitočných vlastností ako: paraleлизmus, vysoký stupeň tolerancie voči chybám, seba-reguláciu, seba-poznávanie a silné nelineárne mapovanie a zovšeobecnenie pre komplexný systém.

### 2.6.1. Klasifikačné algoritmy

Na detekciu anomálií sa vo všeobecnosti používajú algoritmy strojového učenia. Tieto algoritmy vytvárajú model detekcie alebo predikčný model vo fáze učenia sa pomocou tréningového algoritmu na tréningových dátach. Tento predikčný model sa potom testuje na nových údajoch v testovacej fáze. Vstupné dáta potrebujú predspracovanie, aby boli zrozumiteľné pre algoritmy strojového učenia. Vstupné údaje predstavujú pozorovania alebo záznamy, a každý záznam je reprezentovaný atribútom.

Algoritmy, ktoré vyžadujú plne označené údaje, sa nazývajú dohliadané algoritmy učenia. Algoritmy, ktoré nepotrebuju označené údaje sa nazývajú algoritmy učenia bez dozoru. Tieto algoritmy nájdú skrytý vzor v údajoch, keďže dokážu nájsť vzťah medzi údajmi a ich triedou. Poslednou klasifikáciou algoritmov strojového učenia sú čiastočne dohliadané algoritmy, ktoré nepotrebuju všetky záznamy, aby boli označené.

Nasledujúci zoznam podľa Jain a Bhupendra [22] uvádzajú niektoré klasifikačné algoritmy na odhalenie sietových útokov:

- **K-means** – Na vyriešenie hlavného problému klastrovania je algoritmus K-means najbežnejší a najjednoduchší algoritmus učenia sa bez dozoru. Rozdeľuje  $n$  pozorovaní do  $k$  klastrov, každé pozorovanie patrí do klastra s najbližším priemerom a slúži ako prototyp klastra. Výsledkom je rozdelenie dátového priestoru do Voronoiho buniek.
- **Bayesovské siete** – Používajú sa na vyjadrenie poznatkov o nejednoznačnej doméne súboru údajov. Pravdepodobnostné závislosti medzi zodpovedajúcimi náhodnými premennými sú reprezentované hranami tohto modelu. Uzly reprezentujú premenné a hrany kódujú

podmienené závislosti medzi premennými. Bayesovské siete sú riadené acyklické grafy (*Directed Acyclic Graphs* - DAG). Stavy náhodnej premennej a tabuľka podmienenej pravdepodobnosti (*Conditional Probability Table* - CPT) sa nachádza v každom uzle.

- **J48** – Voľne dostupný klasifikátor algoritmu C4.5. C4.5 je program, ktorý vytvára rozhodovací strom založený na súbore označených vstupných údajov. Rozhodovacie stromy sa môžu použiť na klasifikáciu, a preto sa C4.5 často označuje ako štatistický klasifikátor. Podľa Mehmood a Rais [32], rozhodovací strom závisí od atribútov v tréningových dátach, pretože z nich pracuje na základe získaných informácií. Koreňový uzol (root) obsahuje funkciu, ktorá má najviac informácií. Algoritmus J48 je navrhnutý s tými funkciami, ktoré ľahko riešia medzery, ktoré sú prítomné v ID3.
- **ID3** – Dohliadaný algoritmus, ktorý využíva rozhodovací strom založený na matematických výpočtoch. Vykonáva zhora nadol greedy vyhľadávanie prostredníctvom danej tréningovej množiny na testovanie každého atribútu v každom uzle, na vytvorenie rozhodovacieho stromu.
- **NB Strom** – Vysoko škálovateľný hybridný prístup pre veľké databázy. Je vhodný pre prípady, keď je pre klasifikáciu mnoho relevantných atribútov. V takýchto prípadoch je databáza veľká a je žiaduca interpretovateľnosť klasifikátora, atribúty nie sú nevyhnutne nezávislé (t.j. atribúty nie sú podmienene nezávislé). NB strom výrazne zlepšuje výkonnosť svojich zložiek indukcíou vysoko presných klasifikátorov.
- **Náhodný les** – Po anglicky Random Forest je jedným z algoritmov klasifikácie stromov. Hlavným cieľom tohto algoritmu je zvýšiť klasifikátory stromov na základe koncepcie lesa. Náhodné klasifikátory lesov majú akceptovanú mieru presnosti a môžu byť implementované na spracovanie hodnôt šumu súboru údajov. Počas klasifikácie nie je proces opäťovnej modifikácie. Pri implementácii tohto algoritmu by sa mal počet stromov v lese odhadnúť, pretože každý jednotlivý strom v rámci lesa predpovedá očakávaný výstup. Potom sa použije technika hlasovania, ktorá sa používa na výber očakávaného výstupu. Náhodný les je pomalý v trénovaní, je náchylný pretrénovaniu a taktiež je príliš jednoduchý pre komplexné problémy.
- **Rozhodovacie stromy** – Jednotkové stromové štruktúry, ktoré predstavujú rozhodovacie súbory. Tieto súbory generujú pravidlá, ktoré sa používajú na klasifikáciu údajov.
- **Support Vector Machine** – Nová generácia učiacich sa algoritmov. Používa sa na klasifikáciu a regresiu. SVM sú v popredí v oblasti strojového učenia vďaka dôsledným matematickým základom z optimalizácie a teórie štatistického učenia. Podľa [32] SVM oddeluje triedy pomocou hyper plánu a používa údaje označené triedou v tréningovej fáze rovnako ako ostatné klasifikačné algoritmy učenia pod dohľadom. Aj keď SVM je binárny

klasifikátor, môže byť použitý aj na klasifikáciu viacerých tried. Na klasifikáciu viacnásobných tried sa používajú dve rôzne metódy, *one-vs-all* a *one-vs-one*.

- **Logistická regresia** – Podľa autora internetového článku Navlani [37], logistická regresia je štatistická metóda na predpovedanie binárnych tried. Výsledná alebo cieľová premenná má dichotomickú povahu, pričom dichotómia znamená, že existujú iba dve možné triedy. V našom prípade sa jedná o normálnu (0) alebo útočnú premávku (1).
- **Multilayer perceptron** – Mapuje množinu vstupných dát na vhodnú množinu výstupov. Je graf pozostávajúci z viacerých vrstiev, kde každá vrstva je plne pripojená k ďalšej. Multilayer perceptron - MLP využíva spätné šírenie (generalizácia), kontrolovanú výučbovú techniku pre tréning siete. Môže klasifikovať údaje, ktoré nie sú lineárne oddeliteľné. MLP pozostáva najmenej z troch vrstiev uzlov: vstupná vrstva, skrytá vrstva a výstupná vrstva.

Pri klasifikačných algoritnoch je dôležité brať do úvahy vhodný výber parametrov. Na nájdenie optimálnych hyperparametrov modelu sa používa *Grid-search*. Optimálny výber hyperparametrov zaručuje presnejšie predpovede modelu. Hyperparameter modelu je vlastnosť, ktorá je mimo modelu a ktorého hodnota sa nedá odhadnúť z údajov. Hodnota hyperparametru sa musí nastaviť pred začiatkom procesu učenia.

Podľa Williamsa a spol. [57] na výber vhodných parametrov sa používajú algoritmy na báze konzistencia (*Consistency-based Feature selection* - CON) a na báze korelácie (*Correlation-based Feature Selection* - CFS). Tieto algoritmy hodnotia rôzne kombinácie funkcií na identifikáciu optimálnej podmnožiny.

- **Na báze konzistencia** – Vyhodnocuje podmnožiny parametrov a vyberie optimálnu podmnožinu. Optimálna podmnožina je najmenšia podmnožina parametrov, ktoré môžu identifikovať inštancie triedy rovnako konzistentne ako kompletnejšia množina.
- **Na báze korelácie** – Používa hodnotiacu heuristiku, ktorá skúma užitočnosť jednotlivých parametrov spolu s úrovňou vzájomnej korelácie medzi vlastnosťami. Vysoké skóre je priradené k podmnožinám obsahujúcim atribúty, ktoré sú vysoko korelované s triedou a majú nízku vzájomnú koreláciu.

Podmnožiny parametrov podľa [57], ktoré sa majú vyskúsiť sa generujú použitím nasledovných techník vyhľadávania podmnožín:

- **Greedy search** – Pre danú rodičovskú množinu skúška všetky možné potomkové podmnožiny prostredníctvom pridania alebo odstránenia parametrov. Potomková

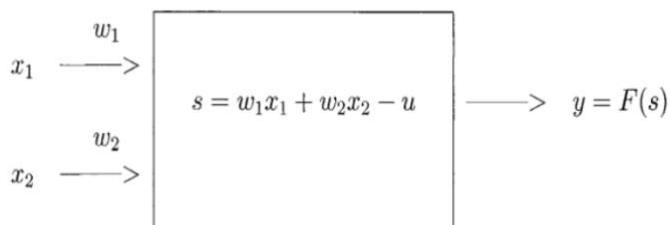
podmnožina, ktorá ukazuje najvyššiu mieru úspechu potom nahradenú rodičovskú podmnožinu a proces sa opakuje. Proces sa ukončí, keď už nie je možné vykonať ďalšie zlepšenia.

- **Best First search** – Je podobná predchádzajúcemu algoritmu, ale na rozdiel od Greedy, Best First search vytvára nové podmnožiny založené na pridaní alebo odstránení parametrov aktuálnej podmnožiny. Má však schopnosť späťne sa pohybovať pozdĺž cesty výberu podmnožiny, aby sa preskúmali rôzne možnosti, keď aktuálna cesta už nevykazuje zlepšenie. Aby sa predišlo spätnému šíreniu, obmedzuje sa počet nevylepšených podmnožín.

## 2.6.2. Neurónová siet'

Haddadi a spol. [19] tvrdia, že neurónová siet' je inšpirovaná ľudským nervovým systémom a používa sa v rôznych oblastiach, ako je rozpoznávanie vzorov, optimalizácia, riadenie a pod. Dokáže riešiť komplexné nelineárne problémy a výsledky hodnotenia sú presné na základe veľkého počtu parametrov, ktoré zvyšujú predvídateľnosť.

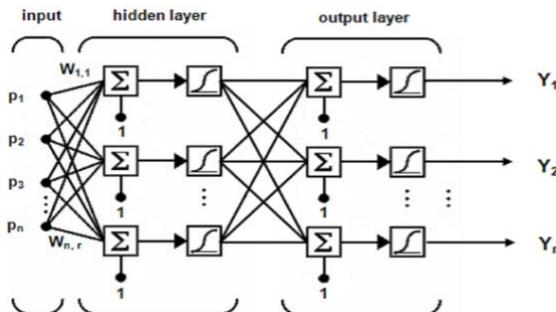
Neurónová siet' sa skladá z niekoľkých procesných jednotiek (uzlov) a riadených väzieb medzi nimi. Tieto spojenia reprezentujú vzťah medzi vstupnými a výstupnými neurónmi. Podľa Sani a spol. [42] výsledok transformácie je určený charakteristikami uzlov a váhami určenými pre prepojeniami medzi nimi. Teda nastavením váh môže byť výstup regulovaný. Proces aktualizácie váh a prahov sa nazýva učenie. Na obrázku č. 10 môžete vidieť jednoduchý neurón, kde parametre  $x_1$  a  $x_2$  predstavujú vstup,  $w_1$  a  $w_2$  váhy,  $u$  - prah,  $s$  - súhrnný blok,  $F$  - aktivačnú funkciu a  $y$  - výstup. V prípade umelej neurónovej siete (Artificial neural networks – ANN) je potrebné si uvedomiť, že predtým ako začneme proces trénovania je potrebné zadefinovať nastavenie neurónovej siete. Je potrebné určiť počet neurónov, počet vrstiev neurónovej siete, algoritmus a prenosovú funkciu pre trénovanie.



Obrázok 10 – Neurón [42]

Neurónová siet' je klasifikovaná do dvoch tried/architektúr na základe spojení: Feed-forward a Recurrent siete.

- **Feed-forward** – Podľa [19] a [42] neuróny sú usporiadane tak, že vstupy do prvej vrstvy neurónov sú vstupmi do neurónovej siete. Výstup každého neurónu v prvej vrstve je vstupom do každého neurónu v druhej vrstve a tak sa to opakuje vo všetkých nasledujúcich vrstvách, až kým nedôjdeme k výstupnej vrstve, ktorej výstupy sú jediným výstupom neurónovej siete. Úlohou tréningového procesu je nájsť správne váhy pre neurónové spojenia, ktoré v kombinácii so vstupmi dosiahnu požadovaný výstup. Tento proces sa uskutočňuje algoritmom spätného šírenia. Na obrázku č. 11 môžete vidieť takúto neurónovú sieť.



Obrázok 11 – Feed-forward neurónová sieť [19]

- **Recurrent** – Sieť svoje výstupy vracia späť do vlastných vstupov.

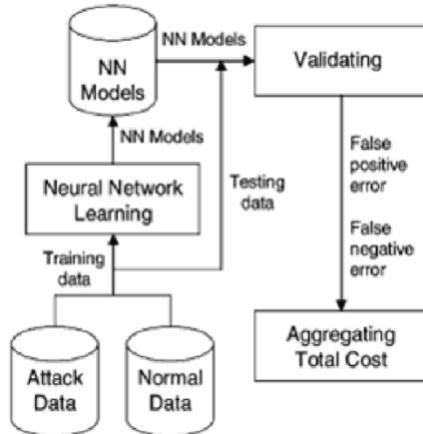
Podľa Haddadi a spol. [19] IDS založené na neurónových sieťach trpia nasledujúcimi problémami:

- **Pretrénovanie** – Ak je neurónová sieť pretrénovaná časťou údajov. V takom prípade je len veľmi málo chýb v tréningu, ale vysoký podiel chýb v teste, pretože neurónová sieť stráca schopnosť generalizácie.
- **Nedostatok pamäte** – Obrovské množstvo dát má za následok vysokú pamäťovú náročnosť. Neurónové siete trpia nízkou pamäťou vo fáze tréningu. Výber správnej tréningovej funkcie môže tento problém riešiť.
- **Rézia** – V zložitých neurónových sieťach je veľa výpočtov a to spôsobuje režijné náklady. Rézia rastie zložitosťou systému.

Typický model neurónovej siete pre IDS je znázornený na obrázku č. 12. V tomto modeli je predpoklad, že prichádzajúce pakety sú extrahované pomocou ľubovoľne dostupnej metódy. Pre tréningovú fázu, máme dátovú množinu pre útočnú a normálnu premávkou. Je potrebné poznamenať, že dátové súbory a vzdelávací modul neurónovej siete sú pripojené k modelu neurónovej siete z dôvodu výberu vhodnej metódy na trénovanie. Pre každý tréningový cyklus je výstup porovnaný s

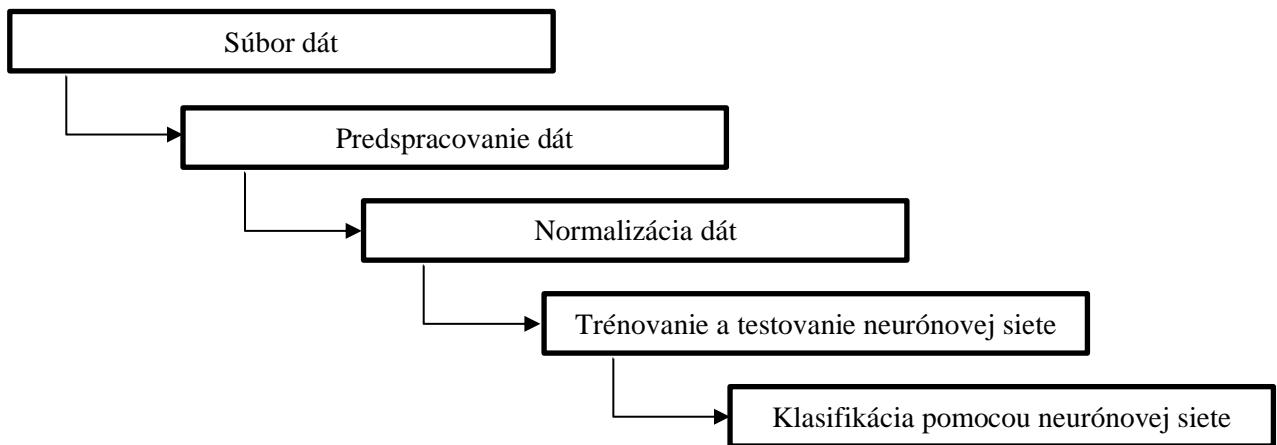
očakávaným výstupom na validačnom module. Trénovanie prebieha nepretržite, kým model neurónovej siete nie je plne natrénovaný. V niektorých IDS trénovanie sa vykonáva pravidelne alebo nepretržite.

Pre vhodný návrh neurónovej siete je potrebné brat' do úvahy predchádzajúce tri problémy (pretrénovanie, nedostatok pamäte, rézia), ktorými trpia neurónové siete.



Obrázok 12 – Model neurónovej siete pre IDS [42]

Na nasledujúcom obrázku č. 13 môžete vidieť proces spracovania dátovej množiny a klasifikácie sieťových útokov.



Obrázok 13 – Proces klasifikácie sieťových útokov

## 2.7. Dátové množiny

Dátová množina je súbor údajov. Najčastejšie dátový súbor zodpovedá obsahu jednej databázovej tabuľky, kde každý stĺpec tabuľky predstavuje konkrétnu premennú a každý riadok zodpovedá prvku z príslušného súboru údajov.

### 2.7.1. NSL-KDD

Dátové množiny Kanadského inštitútu pre kybernetickú bezpečnosť [11] sú používané na celom svete univerzitami, súkromným priemyslom a nezávislými výskumníkmi.

Dátová množina NSL-KDD je vylepšenou verziou starej dátovej množiny KDD'99. Revathi a Malathi [39] tvrdia, že až 75-78% záznamov tvorilo duplicitu v starej verzii KDD. Vykonala sa štatistická analýza tohto súboru údajov a zistili sa problémy, ktoré výrazne ovplyvňujú výkonnosť IDS a vedú k zlému hodnoteniu prístupov detekcie anomalií. Na vyriešenie týchto problémov sa navrhol nový súbor údajov - NSL-KDD, ktorý sa skladá len z vybraných záznamov z kompletného súboru údajov KDD.

Podľa inštitútu pre kybernetickú bezpečnosť v Kanade [11], NSL-KDD nezahŕňa nadbytočné duplicitné záznamy a tak výkon klasifikátorov nie je ovplyvnená metódami, ktoré majú lepšiu mieru detekcie na častých záznamoch. Inštitút ďalej uvádza, že počet vybraných záznamov z každej skupiny obtiažnosti je nepriamo úmerný percentu záznamov v pôvodnom súbore údajov KDD. V dôsledku toho, miery klasifikácie metód strojového učenia sa líšia v širšom rozsahu, čo znamená dosahovanie presnejších hodnotení pre rôzne metódy strojového učenia. Ďalej, počet záznamov v trénovacej a testovacej sade údajov sú primerané, čo umožňuje vykonávanie experimentov na kompletnom súbore dát, bez potreby náhodného výberu menšej časti. Výsledky hodnotenia experimentov tak budú konzistentné a porovnateľné.

Podľa [39] NSL-KDD sa skladá z 21 rôznych útokov z 37 prítomných v súbore testovacích dát. Známe typy útokov sú prítomné v súbore údajov pre trénovanie, zatiaľ čo nové útoky sú v súbore dát pre testovanie, t.j. nie sú dostupné v trénovacom súbore údajov. Typy útokov sú zoskupené do štyroch kategórií: *DoS*, *Probe*, *U2R* a *R2L*. Tabuľka č. 3 uvádzá známe typy útokov.

Tabuľka 3 – Útoky v testovacom súbore dát NSL-KDD [39]

Trieda útoku	Typ útoku
DoS	Back, Land, Neptune, Pod, Smurf, Teardrop, Mailbomb, Processtable, Udpstorm, Apache2, Worm
Probe	Satan, IPsweep, Nmap, Portsweep, Mscan, Saint
U2R	Guess_password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Http tunnel, Sendmail, Named
R2L	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps

Dhanabal a Shantharajah [13] vo svojej práci detailne opisujú jednotlivé atribúty dátovej množiny NSL-KDD a taktiež opisuje experiment v automatizovanom nástroji WEKA na dolovanie v dátach. WEKA implementuje mnoho štandardných procesov dolovania v dát, ako je čistenie, predspracovanie, klastrovanie, klasifikácia, regresia, vizualizácia a výber funkcií dát. Je použitý na vykonávanie klasifikačných experimentov na 20 percentnom NSL-KDD dátovom súbore. Dátový súbor sa predspracuje a normalizuje na rozsahu 0 až 1. Normalizácia dát je dôležitá, pretože niektoré klasifikátory poskytujú lepšiu mieru presnosti na normalizovanom súbore dát. Dhanabal a Shantharajah vo svojej práci použili metódu výberu atribútov na základe korelácie. Dimenzia atribútov sa znížila z počtu 41 na 6. Následne sa vykonalá klasifikácia pomocou algoritmov J48, SVM a Naïve Bayes. Výsledky sú zaznamenané v tabuľke č 4.

Tabuľka 4 – Tabuľka výsledkov experimentu v nástroji WEKA [13]

Klasifikačný algoritmus	Názov triedy útoku	Presnosť (%)
J48	Normal	99.8
	DoS	99.1
	Probe	98.9
	U2R	98.7
	R2L	97.7
SVM	Normal	98.8
	DoS	98.7
	Probe	91.4
	U2R	94.6
	R2L	92.5
Naïve Bayes	Normal	74.9
	DoS	75.2
	Probe	74.1
	U2R	72.3
	R2L	70.1

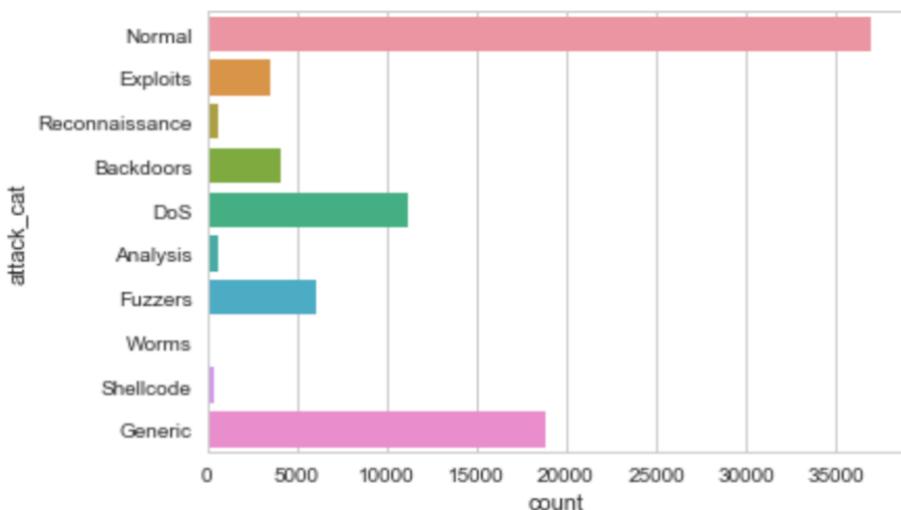
Podľa Thomasa a Pavithrana [52] aj napriek tomu, že súbor údajov NSL-KDD trpí niektorými problémami, je veľmi efektívny súbor údajov, ktorý možno použiť na výskumné účely. Dnes je ľahké získať reálne súbory údajov vzhľadom na povahu bezpečnostnej domény, a preto súbor údajov NSL-KDD sa považuje za jeden z najlepších pre výskum detekcie anomálií.

### 2.7.2. UNSW-NB15

Poľa oficiálnej stránky [55] UNSW-NB15 je neupravený dátový súbor sietových paketov vytvorený nástrojom IXIA PerfectStorm v laboratóriu Cyber Range v Austrálskom centre pre kybernetickú bezpečnosť (*Cyber Range Lab of the Australian Centre for Cyber Security - ACCS*). Bol vytvorený na generovanie bežného hybridu skutočných moderných činností a syntetického súčasného správania pri útokoch.

Na zachytenie sietovej prevádzky vo forme paketov sa používa nástroj *tcpdump*. Tento nástroj sa použil na zachytenie 100 GB surovej dátovej prevádzky. UNSW-NB15 súbor údajov má 9 typov útokov: *Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode a Worms*. Dátový súbor pozostáva z 2 540 044 záznamov a 49 atribútov/vlastností s označením triedy.

Nasledujúci obrázok zobrazuje zastúpenie normálnej a útočnej premávky v jednej z časti dátovej množiny UNSW-NB15.



Obrázok 14 – Zastúpenie sieťovej premávky v dátovej množine UNSW-NB15 [26]

Podľa Moustafa a Slay [35] nástroj IXIA obsahuje všetky informácie o nových útokoch, ktoré sa priebežne aktualizujú zo stánky CVE (*Common Vulnerabilities and Exposures*). CVE je stránka/slovník verejne známych zraniteľností a ohrození bezpečnosti informácií.

Moustafa a Slay [34] opisujú päť kategórií, do ktorých sú jednotlivé vlastnosti dátového súboru začlenené:

- **Vlastnosti toku** – Zahŕňa atribúty medzi hostiteľmi.
- **Základné** – Zahŕňa atribúty, ktoré predstavujú protokoly.
- **Obsahové** – Zahŕňa atribúty TCP/IP, obsahuje aj niektoré atribúty HTTP služieb.
- **Časové** – Obsahuje časové atribúty, napr. čas príchodu medzi paketmi, čas návratu protokolu TCP.
- **Ďalšie** – Túto kategóriu možno ďalej rozdeliť do dvoch skupín:
  - **Funkcie všeobecného účelu** – Každý atribút má svoj vlastný účel.
  - **Funkcie pripojenia** – Sú vytvorené z toku 100 záznamov pripojenia podľa posledného času.

Nasledujúci obrázok zobrazuje zoznam prvých riadkov vybraného súboru údajov. Prvých šest riadkov predstavuje bežnú komunikáciu a posledné riadky predstavuje útočnú komunikáciu

kategorizované ako typ útoku Exploits či Fuzzers. V súbore údajov je normálna sietová premávka označená hodnotou 0 a útok je označený hodnotou 1.

	srcip	sport	dstip	dsport	proto	state	dur	sbytes	dbytes	sttl	dttl	sloss	dloss	service	Sload	Dload	Spkts	Dpkts	swin
0	59.166.0.9	7045	149.171.126.7	25	tcp	FIN	0.201886	37552	3380	31	29	18	8	smtp	1.459438e+06	1.307669e+05	52	42	255
1	59.166.0.9	9685	149.171.126.2	80	tcp	FIN	5.864748	19410	1087890	31	29	2	370	http	2.640454e+04	1.481983e+06	364	746	255
2	59.166.0.2	1421	149.171.126.4	53	udp	CON	0.001391	146	178	31	29	0	0	dns	4.198418e+05	5.118620e+05	2	2	0
3	59.166.0.2	21553	149.171.126.2	25	tcp	FIN	0.053948	37812	3380	31	29	19	8	smtp	5.503374e+06	4.893601e+05	54	42	255
4	59.166.0.8	45212	149.171.126.4	53	udp	CON	0.000953	146	178	31	29	0	0	dns	6.128017e+05	7.471144e+05	2	2	0
5	59.166.0.0	59922	149.171.126.8	6881	tcp	FIN	8.633186	25056	1094788	31	29	38	390	-	2.316642e+04	1.013311e+06	446	858	255
6	175.45.176.0	49582	149.171.126.12	80	tcp	FIN	0.189983	13304	268	254	252	6	1	http	5.291000e+05	9.432423e+03	18	6	255
7	175.45.176.1	0	149.171.126.11	0	sctp	INT	0.000009	440	0	254	0	0	0	-	1.955556e+08	0.000000e+00	2	0	0
8	175.45.176.1	0	149.171.126.11	0	sctp	INT	0.000009	440	0	254	0	0	0	-	1.955556e+08	0.000000e+00	2	0	0
9	175.45.176.1	0	149.171.126.11	0	sctp	INT	0.000009	440	0	254	0	0	0	-	1.955556e+08	0.000000e+00	2	0	0

Obrázok 15 – Vzorka dát UNSW-NB15

Moustafa a Slay [33] navrhli sietovú forenznú schému, kde prvý krok zachytáva sietové pakety cez sniffing a ukladá zachtevané pakety do databázy, aby sa uľahčilo vyšetrovanie útokov. Druhý krok vyberá dôležité funkcie a odstraňuje cudzie a redundantné informácie, ktoré by mohli negatívne ovplyvniť odhalenie útokov. Dôležité funkcie/atribúty sa vyberajú využívajúc štatistiku - chí-kvadrát. Tretí krok skúma útoky a ich pôvod pomocou novej techniky correntropy-variation. Výsledky sú zaznamenané v nasledujúcej tabuľke č. 5 (výsledky sú uvedené v percentách).

Tabuľka 5 – Tabuľka výsledkov pre sietovú forenznú schému [33]

Typ útoku	Veľkosť vzorky		
	100 000	200 000	300 000
Normal	92.12	93.16	93.29
Analysis	88.26	89.45	90.22
DoS	95.71	95.13	97.55
Exploits	76.47	77.82	77.19
Fuzzers	64.33	65.23	66.28
Generic	83.56	87.52	88.87
Reconnaissance	58.38	59.24	60.32
Backdoors	54.42	71.23	72.42
Shellcode	65.76	66.48	65.98
Worms	45.82	45.92	48.87

Navrhnutá technika sa porovnala s troma najmodernejšími prístupmi, a to technikami: podporného vektorového stroja (*Filter-based Support Vector Machine* - FSVM), multivariačnou

korelačnou analýzou (*Multivariate Correlation Analysis* - MCA) a umelou metódou techniky imunitného systému (*Artificial Immune System* - AIS). Na základe porovnaní, siet'ová forenzná schéma je lepšia, pokial' ide o presnosť a frekvenciu falošného poplachu. Forenzná technika poskytuje najlepšie výsledky, pretože odhaduje hodnoty correntropie<sup>1</sup> pre normálne a testované vzorky a následne identifikuje vzorky, ktoré sú viac ako dve štandardné odchýlky od priemeru normálnych vzoriek ako útoky.

### 2.7.3. ISCX

Podľa Kanadského inštitútu pre kybernetickú bezpečnosť [11] sa v ISCX zavádzajú systematický prístup na generovanie požadovaných súborov údajov. Základný princíp je založený na koncepcii profilov, ktoré obsahujú podrobné popisy útokov a abstraktné distribučné modely pre aplikácie, protokoly alebo siet'ové entity nižšej úrovne. Vytvoria sa profily pre agentov, ktorí generujú reálnu prevádzku pre protokoly HTTP, SMTP, SSH, IMAP, POP3 a FTP. Agenti boli potom naprogramovaní tak, aby ich činnosť čo najúčinnejšie napodobnila aktivitu užívateľa. Útokové scenáre boli navrhnuté a vykonané tak, aby vyjadrili skutočné prípady škodlivého správania. Aplikovali sa v reálnom čase na fyzických zariadeniach prostredníctvom ľudskej pomoci.

Dátová množina UNB ISCX IDS 2012 sa skladá z označených siet'ových záznamov, vrátane úplného paketového zaťaženia vo formáte *pcap*. Záznamy sú označené dvadsiatimi atribútmi a podľa Soheily-Khah a spol. [51] má dátová množina viac ako dva miliónov záznamov z čoho 2% dát predstavujú útok. Skladá sa zo siet'ovej aktivity zaznamenanéj počas siedmich dní. Tieto aktivity predstavovali normálnu siet'ovú premávku a útočnú (infiltrácie siete z vnútra, HTTP DoS, DDoS pomocou IRC Botnet a Brute Force SSH) a má nasledujúce charakteristiky:

- **Realistická siet'ová prevádzka** – V ideálnom prípade súbor údajov by nemal vykazovať žiadne nežiadúce siet'ové vlastnosti. Toto zabezpečuje jasnejší obraz o skutočných účinkoch útokov na siet'.
- **Označenie súboru údajov** – Táto charakteristika má obrovský význam pri hodnení rôznych detekčných mechanizmov. Vytváranie súboru údajov v kontrolovanom a deterministickom prostredí umožňuje rozlišovať anomálnu aktivitu od normálnej siet'ovej prevádzky.
- **Zachytenie celkovej siet'ovej interakcie** – Tieto informácie sú nevyhnutné pre hodnenie a správnu interpretáciu výsledkov.

<sup>1</sup> Crrentropia [58] – Štatistické meranie, ktoré odhaduje podobnosť medzi dvoma alebo viacerými náhodnými premennými integráciou funkcie hustoty pravdepodobnosti pozdĺž hlavnej diagonály vektorového priestoru.

- **Úplné zachytenie** – Sietové záznamy sú vytvorené v kontrolovanom prostredí *testbed*, čím sa úplne odstránia obavy týkajúce sa súkromia súvisiace so zdieľaním sietových stôp, t.j. sa zachová prirodzenosť výsledného súboru údajov.
- **Rôzne scenáre narušenia** – Prostredníctvom vykonávania útočných scenárov a uplatňovania abnormálneho správania sa vytvoril rôznorodý súbor útokov na základe nedávnych trendov v bezpečnostných hrozbách.

Autori [51] písu, že mnoho článkov ukázalo, že pokial' ide o heterogénne mnohorozmerné údaje, klasifikátor náhodný les patrí medzi najúčinnejšie metódy. Preto navrhli klasifikačnú metódu postavenú na spomínanom klasifikátore. Výsledky predspracovania K-menas metódy sú použité ako vstupné údaje pre klasifikátor náhodný les.

Autori navrhli, aby toky boli klasifikované podľa ich aplikačnej vrstvy (HTTPWeb, SSH, FTP, ICMP atď.). Vzhľadom na to, že bežné prevádzkové modely sú odlišné v závislosti na aplikáciu alebo službu, je oveľa efektívnejšie postaviť detektor narušenia pre každú z týchto vrstiev. Vo svoje práci porovnávajú navrhovaný algoritmus detekcie narušenia algoritmami ako sú: SVM, algoritmus vyhľadávania najbližšieho suseda (Nearest Neighbor Search – NNS), Naïve Bayes, Rozhodovacie stromy, Neurónová sieť a Náhodný les. Porovnávala sa presnosť, miera detekcie, a frekvencia falošných poplachov. V tabuľke č. 6 môžete vidieť výsledky porovnania na základe presnosti (accuracy). Hodnoty sú uvedené v percentách.

Tabuľka 6 – Tabuľka presnosťí klasifikačných algoritmov pre ISCX IDS 2012 [51]

	<b>SVM</b>	<b>NNS</b>	<b>Naïve Bayes</b>	<b>Rozhodovací strom</b>	<b>Neurónová sieť</b>	<b>Náhodný les</b>	<b>K-means a Náhodný les</b>
HTTPWeb	98.99	99.70	98.04	99.89	99.02	99.88	99.91
SSH	99.47	99.90	99.22	99.87	99.89	99.89	99.98
ICMP	99.83	99.95	99.90	99.99	99.93	99.99	100.00
FTP	99.62	99.95	99.54	99.97	99.94	99.97	99.97
DNS	99.98	99.99	96.18	99.98	99.98	99.99	99.99

Na základe predchádzajúcej tabuľky je zrejmé, že metóda navrhnutá autormi [51] má najlepšie výsledky presnosti, potom nasleduje Rozhodovací strom a algoritmus vyhľadávania najbližšieho suseda.

## 2.7.4. Predspracovanie dát

Podľa Bhardwaj [6] údaje v surovom stave sú vysoko citlivé na šum, chýbajúce hodnoty a nekonzistentné rozdelenie. Kvalita údajov vo vysokej miere ovplyvňuje výsledky klasifikačných metód. Spracovanie údajov je jedným z najkritickejších krokov v procese dolovania v dátach, ktoré sa zaoberá prípravou a transformáciou pôvodného súboru údajov. Metódy predspracovania údajov sú nasledovné:

- **Čistenie dát** –Metóda založená na doplnení chýbajúcich hodnôt, vyhľadení šumu v údajoch, identifikáciu a odstránenie vybočujúcich hodnôt (outliers) a riešení nezrovnalostí.

V dátových množinách reálneho sveta sa stále nájdú chýbajúce hodnoty. Ich korekcia je klúčová, pretože žiadny model nedokáže pracovať s chýbajúcimi dátami, ktoré sú často označované ako NULL či NaN. Odstránenie celých stĺpcov či riadkov z dátovej množiny môže viest' k strate cenných dát, a preto takáto metóda spracovania chýbajúcich hodnôt nie je vhodná. Samozrejme ak máme dátovú množinu vo veľkosti viac sto tisíc záznamov, tak odstránenie dvoch, troch záznamov nemá žiadny vplyv, ale v prípade ak pracujeme s malou dátovou množinou napríklad dvesto záznamov pričom dvadsať percent zo všetkých záznamov sú chýbajúce hodnoty, tak v takom prípade odstránenie záznamov s chýbajúcimi hodnotami nepripadá do úvahy.

V prípade doplnenia chýbajúcich hodnôt sa používajú nasledovné metódy:

- Vynechanie n-tice, ak chýba označenie triedy.
- Manuálne doplnenie hodnôt.
- Doplnenie globálnej konštanty (napr. „neznáme“).
- Doplnenie priemeru všetkých hodnôt.
- Doplnenie priemeru určitej triedy.
- Doplnenie najpravdepodobnejšej hodnoty.
- Predikcia chýbajúcich hodnôt na základe existujúcich hodnôt v rámci daného atribútu.

V prípade odstránenia šumu v údajoch sa používajú nasledovné metódy:

- Lineárna, viacnásobná lineárna regresia.
- Vybočujúce hodnoty možno identifikovať pomocou kombinácie počítačovej a ľudskej kontroly.

- Vybočujúce hodnoty môžu byť detegované klastrovaním, kedy podobné hodnoty sú usporiadané do skupín.
- **Transformácia dát** – Táto činnosť transformuje údaje do vhodnej formy pre metódy dolovania v dátach. Transformácia dát zahŕňa nasledujúce metódy:
  - Normalizácia, t.j. transformácia údajov do rozsahu -1.0 – 1.0 alebo 0.0 – 1.0.
  - Odstránenie šumu.
  - Agregácia údajov.
  - Zovšeobecňovanie údajov.
- **Redukcia dát** – Proces redukcie objemu alebo rozmerov (počet atribútov) súboru dát. Práca s redukovaným súborom údajov je efektívnejšia. Nasledujúci zoznam vymenúva niektoré metódy redukcie dát:
  - Aplikovanie agregačných operácií pre údaje v konštrukcii dátovej kocky.
  - Odstránenie nepodstatných, nerelevantných alebo nadbytočných atribútov.
  - Zmenšenie veľkosti súboru údajov kompresiou, napríklad wavelet transformácia.
  - Nahradenie alebo odhad údajov menšími alternatívnymi údajmi, ako sú parametrické metódy alebo neparametrické metódy (klastrovanie, vzorkovanie alebo použitie histogramov).
  - Generovanie diskretizácie a konceptovej hierarchie, kde surové hodnoty údajov pre atribúty sú nahradené rozsahmi alebo vyššími koncepcnými úrovňami.
- **Spracovanie kategorických atribútov** – Kategorické hodnoty nadobúdajú diskrétné hodnoty ako napríklad farba. Kategorické hodnoty sa ďalej rozdeľujú do dvoch podtypov:
  - **Ordinálne kategorické atribúty** – Ordinálne hodnoty sa dajú zoradiť, napríklad: veľký, stredný, malý.
  - **Nominálne kategorické atribúty** – Nominálne hodnoty sa nedajú zoradiť.

V prípade transformácie kategorických atribútov na numerické sa používajú nasledovné metódy:

- **Mapovanie** – Slovník určujúci mapovanie kategorických hodnôt na numerické.
- **Label encoding** – Transformácia kategorických textových údajov na numerické - zrozumiteľné číselné údaje pre model.
- **One-Hot Encoding** – Enkóder zoberie stĺpce, ktoré obsahujú kategorické hodnoty a na základe unikátnych hodnôt vytvorí ďalšie stĺpce pre numerickú reprezentáciu

kategorických hodnôt. Novovytvorené stĺpce majú hodnoty 0 alebo 1 na základe toho, ktorý riadok má danú kategorickú hodnotu.

- **Obohacovanie dát** – Podľa Hintona [20] obohacovanie dát je definované ako zlúčenie údajov z tretích strán z externého autorizovaného zdroja s existujúcou databázou údajov s cieľom vylepšiť údaje. Hinton ďalej uvádza, že obohacovanie dát má jednu klíčovú podmienku, a to aktuálnosť dát, ktorá vedie k procesu obohacovania dát 24/7.

## 2.8. Zhodnotenie analýzy

V tejto kapitole sme sa venovali analýze danej problematiky. Analýzu problémovej oblasti pokladáme za kľúčovú, pretože sme sa dopracovali k podstatným výsledkom, ktoré pokladáme za potrebné k ďalšej práci na danej úlohe.

Podrobnejšie sme sa venovali typom systémom na detekciu narušenia siete. Popísali sme jednotlivé typy IDS z hľadiska nasadenia. Podrobnejšie sa venovať anomálne založeným detekčným systémom sme pokladali za dôležité, pretože takéto systémy majú schopnosť detegovať doteraz neznáme typy útokov. Potreba venovať sa týmto typom IDS je dôležité, pretože v počítačových sietach sa objavujú nové zraniteľné miesta a útoky sú čoraz sofistikovanejšie. Opísali sme jednotlivé charakteristiky IDS, pretože si myslíme, že je potrebné vedieť limity týchto detekčných systémov, ak chceme prispieť do tejto problémovej oblasti výsledkom našej práce. V krátkosti sme sa venovali architektúre detekčných systémov. Opísali vybrané existujúce nástroje na detekciu útokov alebo narušenia a na záver tejto sekcie sme opísali jednotlivé spôsoby vyhodnocovania IDS. Táto kapitola je veľmi dôležitou súčasťou analýzy, pretože vďaka nej dokážeme pochopiť kvalitu použitého modelu alebo techniky, ktorá sa použila pri klasifikácii.

Ďalej sme sa venovali analýze sietových útokov. Kedže IDS využíva rôzne techniky na detekciu vniknutí do počítačovej siete, sme pokladali analýzu tejto oblasti taktiež za kľúčovú. Popísali sme jednotlivé typy sietových útokov, proti ktorým budeme konfrontovaný aj v našej ďalšej práci. Vďaka naštudovanej literatúry sme zahrnuli do kapitoly sietové útoky aj všeobecný proces odhalovania sietových útokov vo forme obranného mechanizmu.

Ďalšia kľúčová oblasť v rámci tematiky, ktorú táto práca rieši je strojové učenie. Klasifikačné algoritmy budú tvoriť jadro tejto práce, a preto analýza existujúcich algoritmov a podrobnejší popis najpoužívanejšej metódy – neurónová sieť je veľmi dôležitá.

Na záver sme zanalyzovali tri najpoužívanejšie dátové množiny. Opísali sme základné charakteristiky vybraných dátových množín. V tejto kapitole sa nám podarilo na základe preštudovanej literatúry uviesť výsledky experimentov citovaných autorov pre budúce porovnanie s našou prácou. Myslíme si, že takáto forma analýzy je lepšia z hľadiska možnosti porovnania sa voči existujúcim riešeniam nad vybranými dátovými množinami ako samotná analýza dátových množín bez údajov z experimentov. Ďalej sa nám podarilo spísať kroky potrebné pre prípravu dát, pred tým než ich použijeme ako vstup pre metódy dolovania v dátach. Príprava dát je dôležitou súčasťou pri získavaní údajov, pretože údaje v reálnom svete sú zvyčajne neúplné, obsahujú šum a sú nekonzistentné.



### **3. Špecifikácia požiadaviek**

V tejto kapitole sú špecifikované funkcionálne a nefunkcionálne vlastnosti navrhovaného riešenia.

#### **3.1. Funkcionálne požiadavky**

Našim cieľom je vytvoriť dva programové moduly. Prvý programový modul je určený na spracovanie rozsiahlej množiny dát. Druhý programový modul je určený na odhalenie anomálnej sietovej premávky prostredníctvom inteligentných metód strojového učenia. Obe programové moduly budú vhodným spôsobom dokumentovať priebeh ich činností. Výstupom modulov budú grafické vizualizačné prvky pre vhodnú reprezentáciu výsledkov a súbory, ktoré vzniknú. Funkcie programových modulov budú pokrývať nasledujúce oblasti:

- Výber rôznych metód strojového učenia.
  - Programový modul strojového učenia umožní používateľovi vybrať metódu strojového učenia a následne spustiť klasifikáciu nad vybranou množinou predspracovaných dát.
- Výber dátovej množiny na základe typu predspracovania, na ktorú sa má aplikovať zvolená metóda strojového učenia.
  - Programový modul strojového učenia umožní používateľovi vybrať, nad akou dátovou množinou chce aplikovať vybranú metódu strojového učenia. Používateľ bude mať na výber sadu predspracovaných dátových množín.
- Zobrazenie a úprava parametrov vybranej metódy strojového učenia.
  - Programový modul strojového učenia umožní používateľovi zobraziť parametre vybranej metódy strojového učenia. Taktiež umožní upraviť hodnoty týchto parametrov.
- Vytvorenie výstupu klasifikácie v textovej aj grafickej podobe.
  - Programový modul strojového učenia zobrazí výstup klasifikácie po aplikovaní vybranej metódy strojového učenia nad vybranou dátovou množinou dát vo forme tabuľky, grafu a percentuálnej úspešnosti.
- Porovnanie rôznych metód strojového učenia.
  - Programový modul umožní porovnať metódy strojového učenia. Výsledok porovnania bude v podobe tabuľky a grafu.

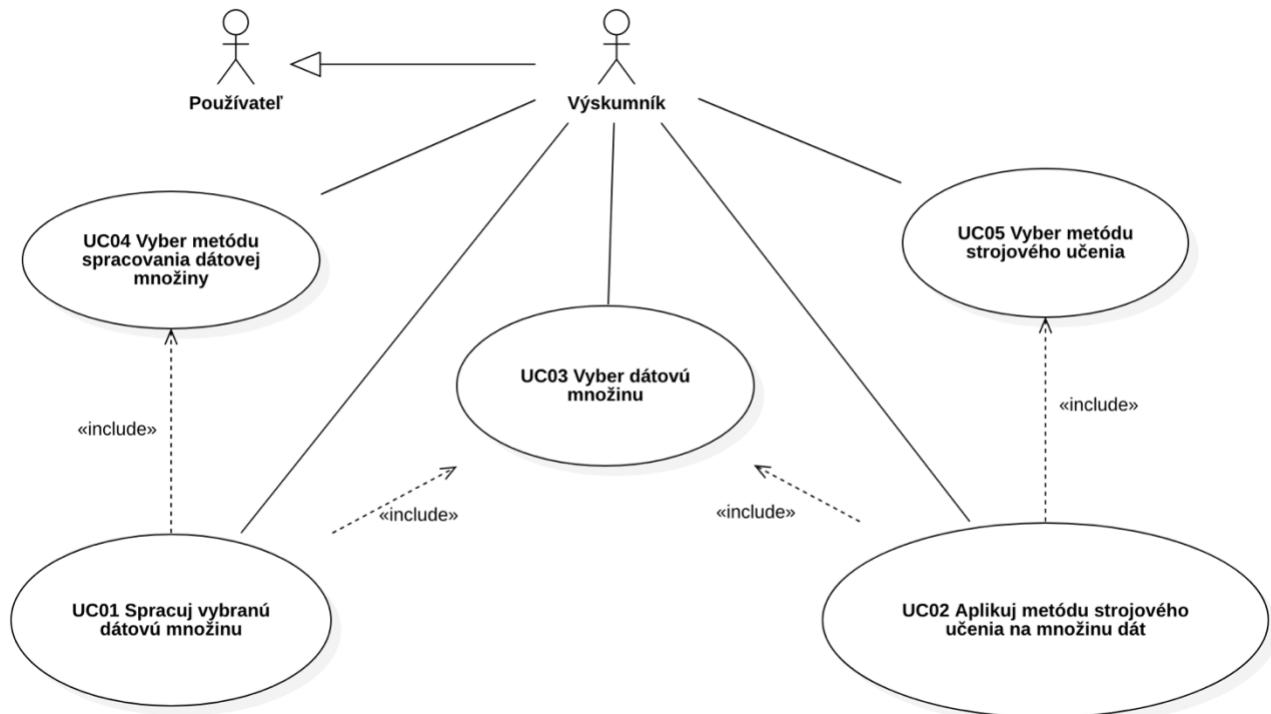
- Výber dátovej množiny na spracovanie.
  - Programový modul určený na spracovanie dátových množín umožní používateľovi vybrať nepredspracovanú dátovú množinu.
- Výber rôznych metód spracovania dátovej množiny.
  - Programový modul určený na spracovanie dátovej množiny umožní používateľovi vybrať metódu spracovania dátovej množiny a následne spustiť spracovanie nad vybranou množinou dát.
- Možnosť zobraziť a uložiť proces spracovania dátovej množiny.
  - Programový modul určený na spracovanie dátovej množiny umožní uložiť proces spracovania dátovej množiny spolu s výstupmi jednotlivých krokov spracovania do dokumentu vo formáte HTML. Spracovanú dátovú množinu bude taktiež možné uložiť vo formáte CSV.
- Možnosť zobraziť a uložiť proces programového modulu strojového učenia.
  - Programový modul strojového učenia umožní uložiť proces práce strojového učenia spolu s výstupmi jednotlivých krokov do dokumentu vo formáte HTML. Taktiež umožní uložiť model metódy strojového učenia vo formáte SAV/H5 pre opakované použitie.

### 3.2. Nefunkcionálne požiadavky

Navrhované riešenie bude implementované ako programový modul, a preto nebude mať klasické používateľské rozhranie. Modul musí byť voľne prístupný pre verejnosť. Mal by využívať voľne dostupný úložný priestor pre ukladanie vzniknutých súborov. Taktiež musí byť zabezpečená vhodná dokumentácia. Spôsob ovládania modulu musí byť prehľadný a intuitívny pre používateľa a bez redundantných informácií. Ovládanie, nastavovanie parametrov a prístup k výstupom modulu nemôže byť náročný, aby sa používateľ vedel rýchlo zorientovať a používať vytvorený programový modul.

### 3.3. Prípady použitia

Na obrázku č. 16 je zobrazený diagram prípadov použitia, ktorý vyplýva zo špecifikácie funkcionálnych vlastností programového modulu.



Obrázok 16 – Diagram prípadov použitia

#### Aktér:

- Výskumník – Osoba ktorá pracuje s programovým modulom.

### 3.3.1. Scenáre prípadov použitia

V tabuľkách č. 7 až 11 sa nachádza detailný opis jednotlivých prípadov použitia.

Tabuľka 7 – Scenár prípadu použitia spracuj vybranú dátovú množinu

<b>Názov prípadu použitia</b>	Spracuj vybranú dátovú množinu		
<b>Identifikačné číslo</b>	UC01		
<b>Hlavný scenár</b>	Umožňuje používateľovi spracovať vybranú dátovú množinu prostredníctvom výberu preddefinovanej metódy predspracovania. Po dokončení spracovania dátovej množiny sa vygeneruje HTML dokument o priebehu spracovania dátovej množiny. Taktiež po dokončení spracovania dátovej množiny sa uloží predspracovaná dátová množina vo formáte CSV.		
Krok	Rola	Akcia	
1.	Používateľ	Vyberie dátovú množinu. Akcia je realizovaná prípadom použitia UC04 Vyber dátovú množinu.	
2.	Používateľ	Vyberie metódu predspracovania dátovej množiny. Akcia je realizovaná prípadom použitia UC06 Vyber metódu spracovania dátovej množiny.	
3.	Používateľ	Spustí proces predspracovania dátovej množiny.	
4.	Systém	Predspracuje dátovú množinu. Po dokončení sa vytvorí HTML dokument o priebehu spracovania dátovej množiny a CSV súbor predspracovanej dátovej množiny a uloží sa na Google Drive úložisko.	
5.	Systém	Prípad použitia končí.	

Tabuľka 8 – Scenár prípadu použitia aplikuj metódu strojového učenia na množinu dát

<b>Názov prípadu použitia</b>	Aplikuj metódu strojového učenia na množinu dát			
<b>Identifikačné číslo</b>	UC02			
	Umožňuje používateľovi aplikovať metódu/y strojového učenia na vybranú predspracovanú dátovú množinu. Po dokončení scenára sa vygeneruje HTML dokument o priebehu aplikovania strojového učenia, ktorý obsahuje aj výsledky hodnotení. Taktiež po dokončení scenára sa uloží model metódy strojového učenia vo formáte SAV/H5. Používateľ v tomto scenári má možnosť porovnať vybrané metódy strojového učenia.			
	<b>Krok</b>	<b>Rola</b>	<b>Akcia</b>	
<b>Hlavný scenár</b>	1.	Používateľ	Vyberie dátovú množinu. Akcia je realizovaná prípadom použitia UC04 Vyber dátovú množinu.	
	2.	Používateľ	Vyberie metódu strojového učenia. Akcia je realizovaná prípadom použitia UC03 Vyber metódu strojového učenia.	
	3.	Používateľ	Spustí sa proces odhalenia sietových útokov pomocou inteligentných metód strojového učenia.	
	4.	Systém	Vykonajú sa operácie strojového učenia nad vybranou množinou dát. Ak sa vybralo porovnanie metód strojového učenia z kroku č. 2, tak sa vykoná aj porovnanie. Po dokončení sa vytvorí HTML dokument o priebehu odhalovania sietových útokov a SAV/H5 súbor modelu metódy strojového učenia a uloží sa na Google Drive úložisko.	
	5.	Systém	Prípad použitia končí.	
<b>Alternatívny scenár</b>	Alternatívny scenár zabezpečuje prípad, kedy používateľ vyberie viac metód strojového učenia.			
	<b>Od kroku</b>	<b>Do kroku</b>	<b>Rola</b>	<b>Akcia</b>
	2a.	3.	Používateľ Systém	Vybral viac ako jednu metódu strojového učenia. Zaevíduje potrebu porovnania metód. Po vykonaní porovnania sa vytvorí výstup v podobe tabuľky a grafu, ktorý zobrazí výsledok porovnania jednotlivých metód strojového učenia.

Tabuľka 9 – Scenár prípadu použitia vyber dátovú množinu

<b>Názov prípadu použitia</b>	Vyber dátovú množinu		
<b>Identifikačné číslo</b>	UC03		
<b>Hlavný scenár</b>	Umožňuje používateľovi vybrať dátovú množinu v rámci určeného typu dátovej množiny.		
<b>Krok</b>	<b>Rola</b>	<b>Akcia</b>	
1.	Používateľ	Používateľ vyberie dátovú množinu zo zoznamu dostupných dátových množín.	
2.	Používateľ	Potvrdí výber.	
3.	Systém	Zaeviduje výber a prípad použitia končí.	

Tabuľka 10 – Scenár prípadu použitia vyber metódu spracovania dátovej množiny

<b>Názov prípadu použitia</b>	Vyber metódu spracovania dátovej množiny		
<b>Identifikačné číslo</b>	UC04		
<b>Hlavný scenár</b>	Umožňuje používateľovi vybrať jednu z preddefinovaných metód spracovania vybranej dátovej množiny.		
<b>Krok</b>	<b>Rola</b>	<b>Akcia</b>	
1.	Používateľ	Používateľ vyberie metódu spracovania vybranej dátovej množiny.	
2.	Používateľ	Potvrdí výber.	
3.	Systém	Zaeviduje výber a prípad použitia končí.	

Tabuľka 11 – Scenár prípadu použitia vyber metódu strojového učenia

<b>Názov prípadu použitia</b>	Vyber metódu strojového učenia		
<b>Identifikačné číslo</b>	UC05		
<b>Hlavný scenár</b>	Umožňuje používateľovi vybrať metódu z množiny preddefinovaných metód strojového učenia, ktorá sa aplikuje na vybranú množinu dát s cieľom zistieť mieru úspešnosti odhalenia sieťových útokov.		
<b>Krok</b>	<b>Rola</b>	<b>Akcia</b>	
1.	Používateľ	Používateľ vyberie metódu strojového učenia z množiny preddefinovaných metód.	
2.	Používateľ	Potvrdí výber.	
3.	Systém	Zaeviduje výber a prípad použitia končí.	



## 4. Návrh riešenia

Na základe podrobnej analýzy anomálne založených detekčných systémov a klasifikačných algoritmov, ktoré sa bežne používajú na detekciu útočnej sietovej premávky sme sa rozhodli vybrať práve tento smer, ktorým sa naša práca bude uberať.

V tejto kapitole navrhнемe dva programové moduly. Prvý programový modul bude slúžiť na predspracovanie dátovej množiny. Druhý programový modul bude implementovať rôzne metódy strojového učenia pre odhalenie útočnej/anomálnej sietovej premávky nad predspracovanou dátovou množinou. Táto práca bude mať výskumný charakter z hľadiska hlbšej analýzy jednotlivých metód predspracovania dátovej množiny a metód strojového učenia, ktorým sa budeme podrobne venovať v jednotlivých kapitolách nižšie v tejto práci. Metódy strojového učenia implementujeme nad vybranou predspracovanou dátovou množinou s cieľom porovnať výsledky hodnotení a vyhodnotiť, ktorá metóda je pre aký typ problematiky sietovej narušenia najvhodnejšia. Zameriame sa predovšetkým na hľadanie anomalií v sietovej premávke a na metódy, ktoré sú určené na ich odhalovanie. Veľkým problémom a nedostatkom v tejto oblasti je vhodné nastavenie metód strojového učenia. Preto naša práca bude venovaná aj výskumu vhodných nastavení parametrov pre vybrané metódy. Ako bolo spomenuté aj v kapitole 2.6.2 Neurónová siet, veľa závisí od nastavenia neurónovej siete, kde je potrebné určiť počet neurónov, počet vrstiev neurónovej siete, algoritmus a prenosovú funkciu pre trénovanie. Podobným spôsobom je potrebné určiť parametre aj pre ostatné metódy strojového učenia. Na základe rôznych nastavení a vstupov do metód dokážeme optimalizovať klasifikáciu a tým pádom pri vhodných nastaveniach dosahovať lepšie výsledky hodnotenia modelu.

Spomenuli sme vstupné dátá pre klasifikačnú metódu, ktoré sú taktiež klúčové pre dosahovanie dobrých výsledkov. V tejto práci bude klúčové vhodné predspracovanie vybranej dátovej množiny, a odhalenie závislostí medzi jeho atribútmi, ktoré majú značný vplyv na odhalovanie útokov. Na základe konzultácie s vedúcim tejto práce budú vybrané metódy implementované na vybranej množine dát - UNSW-NB15, ktorej sme sa venovali v kapitole 2.7 Dátové množiny.

Vybranej dátovej množine je potrebné dokonale porozumieť. Zanalyzujeme jednotlivé atribúty vybranej dátovej množiny a po absolvovaní dostatočnej analýzy dát a pochopení súvislostí medzi atribútmi predspracujeme a transformujeme dátá do formy vstupu pre metódy dolovania v dátach. Následne na základe analýzy spôsobu vyhodnocovania metód strojového učenia (viď. kapitolu 2.4 Spôsob vyhodnocovania IDS) vhodným spôsobom vyhodnotíme vybrané metódy a na základe výsledkov spíšeme výhody a nevýhody použitých metód. Výsledky použitých metód porovnáme a interpretujeme textovým aj grafickým spôsobom.

V tejto kapitole navrhнем aj vývojové prostredie vhodné na prácu s rozsiahloou množinou dát. Navrhнем také prostredie, ktoré podporuje dostatok výpočtového výkonu aj pre prácu s metódami strojového učenia.

Našim cieľom je, aby táto práca prispela svojim výsledkom v tejto problémovej oblasti tým, že spíšeme jednotnú analýzu vplyvov rôznych nastavení metód strojového učenia a rôznych spôsobov predspracovania vybranej dátovej množiny.

## 4.1. Vývojové prostredie

Rozhodli sme sa pre prostredie Google Colab, v ktorom sa budú programové moduly vyvíjať. Colaboratory je voľne dostupná clouдовá služba, ktorá integruje prostredie Jupyter Notebook. Prostredníctvom prehliadača dokážeme spúštať, ukladať a zdieľať svoje analýzy. Obrovskou výhodou tohto prostredia je dostupnosť k vysokým výpočtovým zdrojom. Google [54] uvádza, že tieto zdroje sú postavené na Tensor Processing Unit (TPU). Cloud TPU je stroj ASIC<sup>2</sup> navrhnutý na mieru, ktorý sprostredkuje produkty spoločnosti Google, ako sú prekladač, fotografie, vyhľadávanie, asistent a e-mail. Cloud TPU<sup>3</sup> verzia 3 ponúka 420 TFLOPS<sup>4</sup> a 128 GB pamäte s veľkou šírkou pásma (High Bandwidth Memory - HBM). Google Colab sa dá prepojiť aj s ďalšou službou od Google, a to s Google Drive úložným priestorom kam budeme ukladať samotné súbory programových modulov. Výstupy programových modulov sa budú ukladať vo formáte HTML, modely metód strojového učenia vo formáte SAV<sup>5</sup> (modely neurónových sietí vo formáte Hierarchical Data Format – H5) a v prípade dátových množín predspracované dátové množiny vo formáte CSV. Programový modul bude naprogramovaný programovacím jazykom Python vo verzii 3.

## 4.2. Programový modul predspracovania dátovej množiny

V prostredí Google Colab si založíme osobitný Jupyter Notebook, ktorý bude spracovávať rozsiahlu množinu dát. Bude obsahovať sekvenciu krokov potrebných na predprípravu dát, ktoré budú neskôr vstupom pre metódu strojového učenia vo forme trénovacej, validačnej a testovacej podmnožiny. Na úvod používateľ určí, ktorú dátovú množinu chce predspracovať a spôsob predspracovania. Navrhнем súbor metód predspracovania dátovej množiny na základe

<sup>2</sup> Application-Specific Integrated Circuit (ASIC) [5] – Mikročip určený na špeciálne použitie a nie na všeobecné použitie.

<sup>3</sup> <https://cloud.google.com/tpu/>

<sup>4</sup> Floating-point Operations Per Second (FLOP) [14] – Jednotka výpočtového výkonu počítača, ktorá udáva počet operácií v pohyblivej rádovej čiarke za sekundu.

<sup>5</sup> SAV [43] – Prípona súboru použitá pre uložený dátum Statistical Package for Social Sciences

spomínaných metód predspracovania dátových množín, ktorým sme sa venovali v kapitole 2.7.4 Predspracovanie dát. Spôsoby predspracovania dátovej množiny sa budú lísiť iba v niektorých krokoch. Pri výbere metódy predspracovania dátovej množiny sa určí, aký typ metódy sa má aplikovať pre daný krok predspracovania. Niektoré kroky predspracovania sa budú môcť vynechať. Toto neplatí pre kroky nevyhnutné pre predspracovanie dátovej množiny do akceptovateľnej podoby metódou strojového učenia. Tieto kroky sú nasledovné: načítanie, analýza a čistenie dát dátovej množiny. Metódy predspracovania dátovej množiny budú zahŕňať rôzne voľby ako: metóda doplnenia chýbajúcich hodnôt, spôsob spracovania kategorických atribútov, obohacovanie dát, štandardizácia a vzorkovanie. Výber metódy predspracovania dátovej množiny je tiež súčasťou implementácie riešenia diplomového projektu.

V programovom module sa najskôr načíta vybraná dátová množina a následne sa spustí predspracovanie. Prvým krokom bude analýza dátovej množiny, kedy zistíme či dáta sú úplné, jednotné, či obsahujú chýbajúce hodnoty, redundantné informácie, nepotrebné atribúty, alebo vybočujúce hodnoty. Ďalej zistíme koreláciu jednotlivých atribútov dátovej množiny, ktorá je užitočná z hľadiska pochopenia vzťahov medzi atribútmi. V programovacom jazyku Python existuje mnoho metód na analýzu dát. My si vyberieme iba niektoré z nich. Na základe predošej analýzy dokážeme vhodne aplikovať ďalšie metódy na predspracovanie dátovej množiny.

Ďalším krokom je čistenie dát. V tomto kroku musíme dbať na to aké dáta ideme meniť resp. dopĺňať, pretože akákoľvek zmena v množine môže mať neskôr veľký vplyv na výsledok vyhodnotenia metódy strojového učenia.

Vybočujúce hodnoty v prípade dátovej množiny tohto charakteru je riskantné odstraňovať, pretože práve v prípade extrémnych výkyvov v hodnotách jednotlivých atribútov dátovej množiny môže byť zaznamenaný útok resp. anomália. Samozrejme toto nie je pravidlo, taktiež sa môže stať, že pri vysokom výkyve, napríklad v počte prenesených byte-ov sa jednoducho iba posiela súbor s veľkou veľkosťou. Túto skutočnosť najjednoduchšie odhalujú grafické metódy zobrazenia distribúcie hodnôt pre jednotlivé atribúty dátovej množiny počas analýzy dát.

V prípade korekcie chýbajúcich hodnôt v našom prípade máme viacero možností. Z hľadiska, že sa jedná o citlivé údaje v doméne sietí, aj malá zmena napríklad v počte prenesených byte-ov či dĺžke spojenia môže mať kontraproduktívny dopad. Pre čo najpresnejšie doplnenie chýbajúcich hodnôt sme sa preto rozhodli využiť regresiu t.j. strojové učenie určené na predpoved' hodnoty na základe ostatných známych hodnôt. Iným slovom povedané, regresia je určená na odhad hodnoty nejakej spojitej premennej. Táto metóda oproti metódam spomenutým v kapitole 2.7.4 Predspracovanie dát je najrelevantnejšia a najkomplexnejšia, ktorú potvrdzuje aj autor Jimoh v článku [25]. Jimoh uvádzá, že jedinou nevýhodou tohto prístupu je to, že ak neexistuje žiadna korelácia medzi atribútmi s chýbajúcimi údajmi a inými atribútmi v množine údajov, potom model pre

predpovedanie chýbajúcich hodnôt bude skreslený. Na základe predošej analýzy ale budeme vedieť predpovedať mieru skreslenia modelu. Skreslenie modelu následne dokážeme korigovať nájdením optimálnych hyperparametrov modelu, ktorého výsledkom sú „presnejšie“ predpovede. V tomto prípade je potrebné si uvedomiť, že miera presnosti predpovede chýbajúcich hodnôt silno závisí od počtu existujúcich hodnôt, na ktorých sa dokáže model naučiť predikciu.

Ošetrenie kategorických atribútov v dátovej množine je ďalšia diskutovaná oblasť procesu predspracovania množiny dát. Najskôr je potrebné si uvedomiť rozdiel medzi nominálnymi a ordinálnymi kategorickými atribútmi. Podľa autora Dhairy [12], najväčšiu chybu, ktorú väčšina ľudí robí, je to, že nie sú schopní rozlíšiť rozdiel medzi ordinálnymi a nominálnymi atribútmi. Podľa neho, ak použijeme funkciu mapovania alebo Label encoder s nominálnymi atribútmi, model si bude myslieť, že existuje určitý vzťah medzi nominálnymi atribútmi ako medzi kategorickými atribútmi. Tako získané výsledky nemusia byť najoptimálnejšie, ale na druhej strane takýmto štýlom vieme značne zredukovať dimenziu dátovej množiny. Myslíme si, že aj napriek tomu sa dá nájsť vhodný spôsob ako použiť mapovanie pre nominálne atribúty. Napríklad takým spôsobom, že zistíme aké nominálne atribúty sa vyskytujú pri útočnej siet'ovej premávke a k nim priradíme mapovaním vyššie numerické hodnoty ako pre nominálne hodnoty pri normálnej neanomálnej siet'ovej premávke. Tým pádom vznikne vzťah medzi nominálnymi atribútmi, ktorý model strojového učenia môže interpretovať tak, že tie nominálne atribúty, ktoré majú vyšší príznak majú väčšiu pravdepodobnosť výskytu pri útočnej siet'ovej premávke. Ďalší prístup vysporiadania sa s nominálnymi atribútmi je ich štandardné predspracovanie kedy nehľadíme na to, aké hodnoty po predspracovaní nadobúdajú. V tomto prístupe berieme skôr do úvahy inteligenciu inteligentných metód strojového učenia. Tento prístup je založený na dôvere inteligencie strojového učenia s predpokladom, že vybraná metóda strojového učenia dokáže nájsť určitú spojitosť medzi atribútmi.

V prípade úpravy hodnôt jednotlivých atribútov môže nastať situácia, kedy hodnoty jednotlivých atribútov sa vyvíjajú v takom smere, že vzniknú nové korelácie alebo dokonca vzniknú príliš vysoké korelácie. Tento fenomén sa nazýva multikolinearita. Podľa internetového článku [36] je multikolinearita stav veľmi vysokých vzájomných vzťahov alebo vzájomných súvislostí medzi nezávislými premennými. Ide teda o druh narušenia údajov. Ak sú v údajoch prítomné štatistické závery o údajoch tak nemusia byť spoľahlivé. Ďalej sa uvádzá, že tento fenomén môže vzniknúť pri aplikovaní metódy One-Hot Encoding s čím súhlasí aj autor Dhairy. Multikolinearita môže vzniknúť aj vplyvom opakovania tej istej premennej rovnakého druhu či zahrnutím premennej, ktorá sa počíta z iných premených. Multikolinearitu možno zistiť pomocou tolerancie a rozptylného inflačného faktoru (Variance Inflation Factor - VIF). Ak je hodnota tolerancie menšia ako 0,2 alebo 0,1 a súčasne hodnota VIF je 10 a vyššia, potom je multikolinearita problematická. Problémom odstraňovania multikolinearity sa zaoberá aj Frost [15] vo svojom článku.

Pokladáme za dôležité spomenúť aj proces obohacovania dát, ktorému sme sa venovali v kapitole 2.7.4 Predspracovanie dát. V našom prípade by sa jednalo o obohacovanie dát z hľadiska IP adries. Keďže dátová množina má zdrojovú a cieľovú adresu ako nominálnu hodnotu, tak dohľadanie krajiny IP adresy by nemal byť žiadnený problém. IP adresy ako nominálne hodnoty nemá význam predspracovať štandardným spôsobom pomocou One-Hot Encoding metódy, pretože rozsah IPv4 adries je viac ako 4 miliardy. Možný problém vidíme v špeciálnych IP adresách ako sú súkromné, multicast či localhost IP adresy. Tieto adresy sa dajú kategorizovať na základe rozsahu IP adresy. Po prekonvertovaní IP adresy na názov alebo kód krajiny, alebo na jednu z predchádzajúcich hodnôt, môžeme aplikovať mapovanie či One-Hot Encoding. Týmto spôsobom predspracovania IP adries dokážeme získať ďalší užitočný atribút, o ktorom môžeme s určitosťou povedať, že je relevantný pre metódu strojového učenia. Keďže určitú metódu dokážeme natrénovať na istej množine dát, tak tento model sa naučí klasifikovať známe IP adresy z danej množiny dát. V prípade nasadenia vybranej metódy strojového učenia do reálneho prostredia musíme IP adresy osobitne predspracovať v samostatnom module. Ako sme aj v analýze spomenuli, obohacovanie je proces, ktorý musí byť neustále aktuálny, a tak musíme myslieť na prípady resp. ošetríť také prípady, kedy sa IP adresa nedá predspracovať na základe existujúcej množiny dát tretej strany z dôvodu neúplnosti dát.

Ďalším možno štandardným krokom je štandardizácia, iným slovom normalizácia dátovej množiny. O tom, prečo je tento krok nevyhnutý píše autorka Jaitley v článku [23]. Podľa autorky nie každá dátová množina vyžaduje normalizáciu hodnôt, tento proces je nevyhnutný iba vtedy, ak dátová množina má rôzne rozsahy hodnôt dát. Iným slovom, štandardizácia je systematický spôsob, ako zabezpečiť, aby bola dátová štruktúra vhodná na všeobecné vyhľadávanie bez určitých nežiaducích charakteristík. Po štandardizácii model strojového učenia lepšie dokáže „porozumieť“ dátam a dosiahnuť tak lepšie výsledky. Túto skutočnosť dokázala aj Jaitley vo svojom experimente. Tvrdí, že tento krok je potrebný pre dosahovanie lepších výsledkov.

Podľa Alencara [3] prevzorkovanie je všeobecne prijatá technika riešenia vysoko nevyvážených súborov dát. Vzorkovanie pozostáva z odstránenia vzoriek z väčšinovej triedy (undersampling) a/alebo pridaním ďalších záznamov/vzoriek z menšinovej triedy (oversampling). Prevzorkovanie údajov má za cieľ lepšie reprezentovať menšinové triedy, aby vybraný klasifikátor mal viac vzoriek, z ktorých sa môže učiť alebo menej vzoriek, aby lepšie odlišil vzorky menšinových tried od väčšinových. V prípade oversamplingu vznikajú ďalšie záznamy, aby sa vyrovnal počet vzoriek menšinovej triedy väčšinovej. Tým pádom nám vznikne viac záznamov a dátová množina sa zväčší čo môže mať dopad na čas učenia modelu strojového učenia, hlavne pri rádovo sto tisíc až milión záznamoch. Tento krok predspracovania dátovej množiny pokladáme za diskutabilný, pretože v reálnej prevádzke počítačovej siete nemáme rovnomerné zastúpenie normálnej a útočnej sietovej

premávky. Zvyčajne je množstvo normálnej, neútočnej sietovej premávky viac ako desaťnásobne väčšie ako množstvo útočnej premávky. V našom prípade overíme oba prípady, t.j. klasifikáciu sietovej premávky na prevzorkovanej množine dát a na množine dát bez vzorkovania.

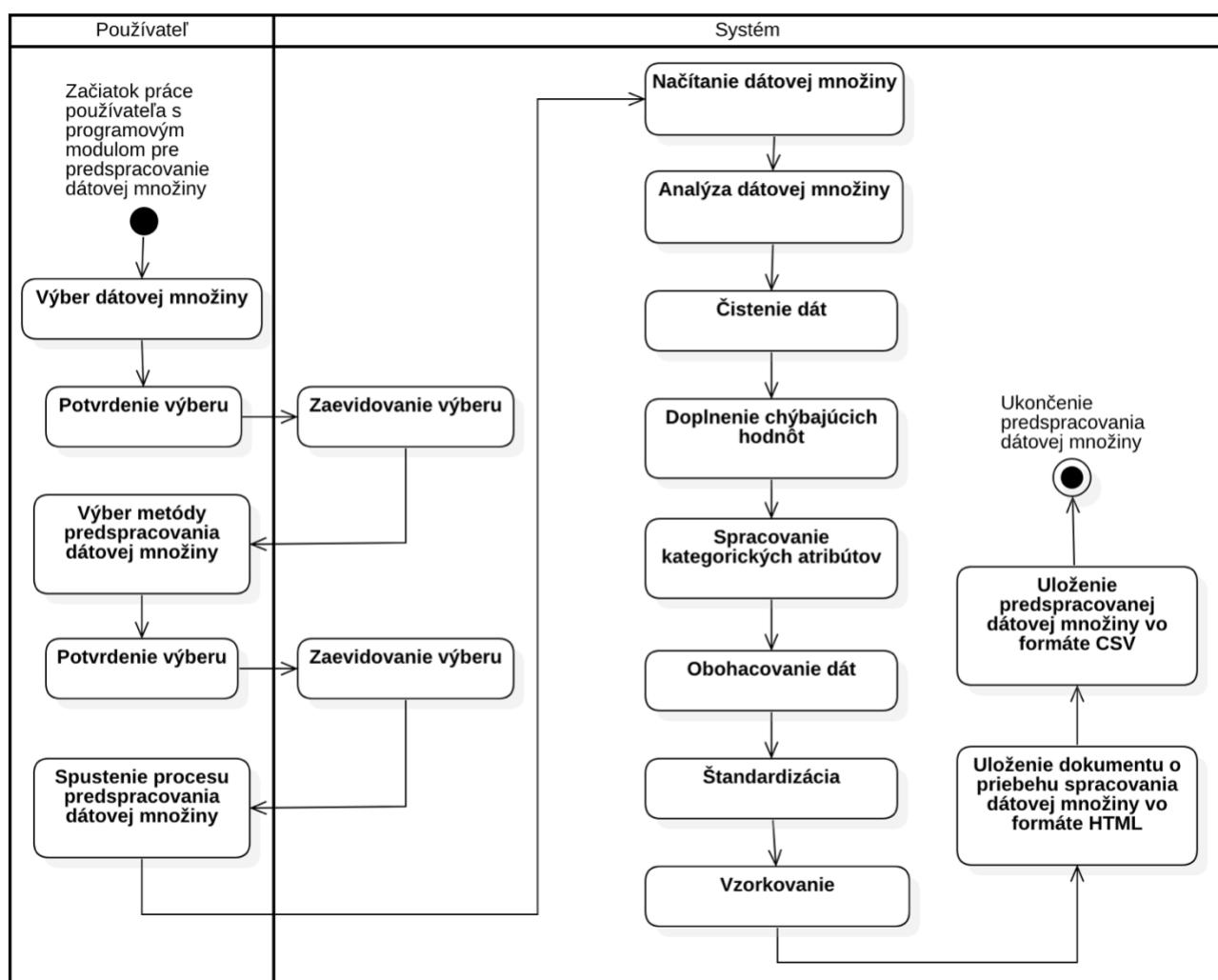
Na záver sa uloží správa o priebehu spracovania dátovej množiny vo formáte HTML. Predspracovaná dátová množina sa uloží vo formáte CSV. Výsledné dokumenty sa uložia na Google Drive úložisko.

## 4.3. Opis činností programového modulu predspracovania dátovej množiny

Diagram aktivít predspracovania dátovej množiny na obrázku č. 17 predstavuje proces spracovania vybranej dátovej množiny.

Používateľ inicializuje proces predspracovania výberom dátovej množiny a metódy predspracovania. Pri výbere metódy predspracovania sa určí, aký typ metódy sa má aplikovať pre daný krok predspracovania. Niektoré kroky predspracovania sa budú môcť vynechať. Toto neplatí pre kroky nevyhnutné pre predspracovanie dátovej množiny do akceptovateľnej podoby metódou strojového učenia ako bolo spomenuté aj v predchádzajúcej kapitole. Systém následne vykoná predspracovanie vybranej dátovej množiny sekvenčiou krovok metód predspracovania.

Výsledkom je dokument vo formáte HTML, ktorý predstavuje správu o priebehu spracovania dátovej množiny. Predspracovaná dátová množina sa uloží vo formáte CSV. Výsledné dokumenty sa ukladajú na Google Drive úložisko.



Obrázok 17 – Diagram aktivít programového modulu predspracovania dátovej množiny

## 4.4. Programový modul strojového učenia

Po spracovaní dátovej množiny programovým modulom určeným na predspracovanie rozsiahlej dátovej množiny, ktorej sme sa venovali v kapitole 4.2 Programový modul predspracovania dátovej množiny môžeme postúpiť k metódam výpočtovej inteligencie strojového učenia.

V prostredí Google Colab si založíme osobitný Jupyter Notebook, pomocou ktorého nad množinou dát spustíme rôzne metódy strojového učenia. Tento osobitný programový modul je určený na vyhodnotenie úspešnosti predikcie odhalenia sietových útokov. Bude obsahovať sekvenciu krokov potrebných na načítanie predspracovanej dátovej množiny. Taktiež bude podporovať zlúčenie viacerých dátových množín, ktoré sa spracovali rovnakou metódou predspracovania. Táto vlastnosť programového modulu je užitočná vtedy, keď chceme, aby model mohol použiť väčšie množstvo dát, na ktorých sa môže učiť.

V prípade ak je mierny rozdiel medzi dátovými množinami určenými na zlúčenie, treba tieto nezrovnalosti adekvátnie vyriešiť. Takáto situácia môže nastať napríklad v prípade, ak sa v jednej dátovej množine nachádza nejaký nominálny atribút (stĺpec po predspracovaní metódou One-Hot Encoding), ktorý sa v druhej dátovej množine nenachádza. V takom prípade, používateľ musí rozhodnúť o ďalšom procese spracovania. Prvé riešenie spočíva v odstránení atribútu z dátovej množiny určenej na zlúčenie s ďalšou dátovou množinou, v ktorej sa daný atribút nenachádza. Túto metódu je vhodné aplikovať iba vtedy, keď daný atribút neovplyvní kvalitu dát. Tejto problematike sme sa venovali v kapitole 2.7.4 Predspracovanie dát, časť čistenie dát. Druhá metóda spočíva v tom, že sa daný atribút či atribúty sa doplnia do dátovej množiny, ktorá ich neobsahuje s neutrálnymi hodnotami. Pre účely tejto diplomovej práce sme sa rozhodli zjednodušiť proces zlúčenie dátových množín. Zlúčenie dátových množín je možné ak obe dátové množiny určené na zlúčenie obsahujú rovnaký počet atribútov.

Načítaná a spracovaná dátová množina sa potom pretransformuje na menšie časti: trénovacia podmnožina, validačná podmnožina a testovacia podmnožina dát. Autor Shah [48] vysvetľuje vo svojom článku rozdiely medzi jednotlivými podmnožinami:

- **Trénovacia podmnožina** – Vzorka údajov určená na učenie modelu.
- **Validačná podmnožina** – Vzorka údajov použitá na vykonanie nezaujatého vyhodnotenia modelu, ktorý bol naučený na vzorke trénovacích dát. Táto vzorka je určená predovšetkým na vyladenie hyperparametrov modelu, pričom *hyperparameter* je parameter, ktorého hodnota je nastavená pred začiatkom procesu učenia modelu.
- **Testovacia podmnožina** – Vzorka údajov použitá na objektívne vyhodnotenie konečného modelu, ktorý bol naučený na vzorke trénovacích dát.

Pred tým než sa spustí učenie modelu, používateľ bude môcť vybrať metódu strojového učenia. Výberom metódy sa určí, aké metódy/modely strojového učenia sa majú použiť, či sa majú zlúčiť načítané dátové množiny, v akom pomere sa majú transformovať dátové množiny a akou metódou sa majú optimalizovať hyperparametre. Používateľ ďalej bude mať možnosť rozhodnúť sa, či chce učenie spustiť nad celou množinou dát so všetkými atribútmi alebo len nad určitou množinou atribútov, ktoré sa vyberú aplikovaním metódy na výber najlepších atribútov. Keďže chceme myslieť na dobu trénovania metódy strojového učenia, tak redukcia dimenzie predspracovanej dátovej množiny môže túto dobu výrazne ovplyvniť. S týmto záverom súhlasí aj autor Shaikh [49]. Podľa Shaikha výber najlepších atribútov má aj ďalšie výhody ako: zníženie pravdepodobnosti preučenia (overfitting), zvýšenie presnosti z dôvodu odstránenia nepodstatných atribútov, pretože nerelevantné alebo čiastočne relevantné atribúty môžu mať negatívny vplyv na výkonnosť modelu. Naivný prístup vedie k vyčítaniu dôležitých atribútov z korelačnej tabuľky, ale my aplikujeme vybranú metódu strojového učenia na ich výber.

Implementovali sme osem metód strojového učenia. Tieto metódy sú nasledovné:

- Náhodný les
- XGradient Boosting klasifikátor
- Rozhodovací strom
- Logisticálna regresia
- SGD klasifikátor
- KNN klasifikátor
- Perceptron
- Neurónová sieť
  - Long Short-Term Memory (LSTM) rekurentná neurónová sieť
  - Rekurentná neurónová sieť (Recurrent Neural Network – RNN)
  - Hlboká neurónová sieť (Deep Neural Network – DNN)

Metódam strojového učenia ako náhodný les, rozhodovací strom a logisticálna regresia sme sa venovali v kapitole 2.6.1 Klasifikačné algoritmy. Náhodný les bol vybraný, pretože vo všeobecnosti poskytuje vysokú presnosť pri klasifikačných úlohách, je odolný proti pretrénovaniu a je schopný spracovať veľké množstvo údajov s vysokou dimensiou. Rovnako ako náhodný les, Gradient Boosting je ďalšou technikou na vykonávanie dohliadaných úloh strojového učenia, ako je klasifikácia a regresia. Gradient Boosting je strojové učenie súboru metód čo znamená, že konečný model je založený na zbierke modelov. Predikčná sila týchto jednotlivých modelov je slabá a náhylná k pretrénovaniu, ale kombinácia mnohých takýchto slabých modelov v súbore vedie k

celkovo lepšiemu výsledku. Extreme Gradient Boosting (v skratke XGradient Boosting) je špecifická implementácia metódy Gradient Boosting, ktorá používa presnejšie approximácie na nájdenie najlepšieho modelu stromu. Ďalej sme sa rozhodli použiť rozhodovací strom kvôli jeho robustnosti. Existuje množstvo rôznych paradigiem rozhodovacieho stromu, ako napríklad ID3, C4.5, CART alebo CHAID. ID3 a C4.5 sme sa venovali v kapitole 2.6.1 Klasifikačné algoritmy. Rozhodovací strom dokáže spracovať údaje bez štandardizácie a vo všeobecnosti si vyžaduje predbežné spracovanie údajov menšie úsilie. Logistickú regresiu sme vybrali z dôvodu, lebo ide o štatistickú metódu na predpovedanie binárnych tried, ako je aj náspríklad normálnej a útočnej sietovej premávky vo vybranom súbore údajov. Klasifikátor SGD je tiež lineárny klasifikátor ako logistická regresia. Je kombináciou SVM (binárny klasifikátor) a logistickej regresie s výcvikom SGD. Tento model implementuje regularizované lineárne modely so stochastickým gradientom učenia (Stochastic Gradient Descent - SGD). Gradient straty sa odhaduje pre každú vzorku v čase a model sa aktualizuje podľa postupu so znižujúcou sa mierou učenia. Algoritmus K-Nearest Neighbors (KNN) sme vybrali, pretože je automaticky nelineárny, dokáže detegovať lineárne alebo nelineárne distribuované údaje. KNN má tendenciu fungovať veľmi dobre so širokou škálou údajových bodov. Na druhej strane je potrebné KNN algoritmus správne nastaviť, kde kritickým bodom je výber hodnoty  $k$  a metriky vzdialenosťi. Ďalším klasifikátorom je perceptron, ktorý je lineárny binárny klasifikátor - jednovrstvová neurónová siet. Neurónovým sieťam sme sa venovali v kapitole 2.6.2 Neurónová siet. Neurónová siet má schopnosť dynamicky vytvárať zložité predikčné funkcie a napodobňovať ľudské myšlenie spôsobom, ktorým nemôže žiadny iný spomínaný algoritmus. Neurónové siete predstavujú triedu štatistických metód schopných approximácie univerzálnych funkcií a sú schopné naučiť sa nelineárne vzťahy medzi závislými a nezávislými premennými priamo z údajov bez predchádzajúcich predpokladov o štatistických rozdeleniach. Viacvrstvové neurónové siete (MLP) predstavujú prominentnú triedu neurónových sietí, ktorá implementuje paradigmu metódy dohliadanej výučby, ktorá sa bežne používajú pri empirickej klasifikácii a získavania údajov.

Neurónová siet môže mať viac variácií, ako napríklad nami vybraté tri typy neurónových sietí, ktoré sú: LSTM, RNN a DNN. Vo všeobecnosti platí, že ak neurónová siet, ktorá má viac ako jednu skrytú vrstvu, sa označuje ako hlboká neurónová siet. Rekurentné neurónové siete majú široké uplatnenie a tak sme sa rozhodli vybrať najbežnejšiu variantu neurónových sietí a to RNN. RNN je veľmi úspešná napríklad v spracovaní prirodzeného jazyka, detekcii anomálií a monitorovaní stavu. Je úspešná najmä s variantom LSTM, ktorý je schopný pozerať späť dlhšie ako RNN tzn., že klúčovým atribútom rekurentných neurónových sietí je ich schopnosť zotrvať v informáciách alebo v stave neurónu pre použitie v neurónovej sieti neskôr.

Pred samotným učením modelu/ov strojového učenia je potrebné vybrať model/y ktoré sa majú natrénovať a následne sa má vykonať vyhodnotenie miery úspešnosti modelu. V prípade ak

používateľ zvolil viac ako jednu metódu, tak sa zvolené metódy strojového učenia porovnajú a výsledok porovnania sa zapíše do tabuľky porovnaní a vytvorí sa ROC-AUC graf. Týmto spôsobom získame sumár porovnaní metód strojového učenia, na základe ktorého môžeme ľahko odvodiť, ktorá metóda je najlepšia pre daný problém. Spôsobom vyhodnocovania sme sa venovali v kapitole 2.4 Spôsob vyhodnocovania IDS.

Po nastavení predošlých nastavení môže prebehnúť proces trénovania metód strojového učenia. Pred každým trénovaním modelu strojového učenia navrhujeme optimalizáciu hyperparametrov. Je dôležité poznamenať rozdiel medzi parametrami modelu a hyperparametrami. Podľa Prabhu [38] parametre modelu sú vlastnosti údajov, ktoré sa model naučil sám počas trénovania. Tieto hodnoty môžu byť váhy jednotlivých spojení, odchýlky či rozhodovacie body v prípade rozhodovacích stromov. Hyperparametre zase riadia proces trénovania a nastavujú sa pred spustením samotného trénovania modelu. Každý model strojového učenia má vlastný súbor takýchto hyperparametrov a neexistuje jednotný, všeobecný návod na ich správne nastavenie. Existujú však metódy, ktorými dokážeme hyperparametre optimalizovať. Našim cieľom je aplikovať optimalizáciu hyperparametrov, jednou z existujúcich metód a kombináciou viacerých metód.

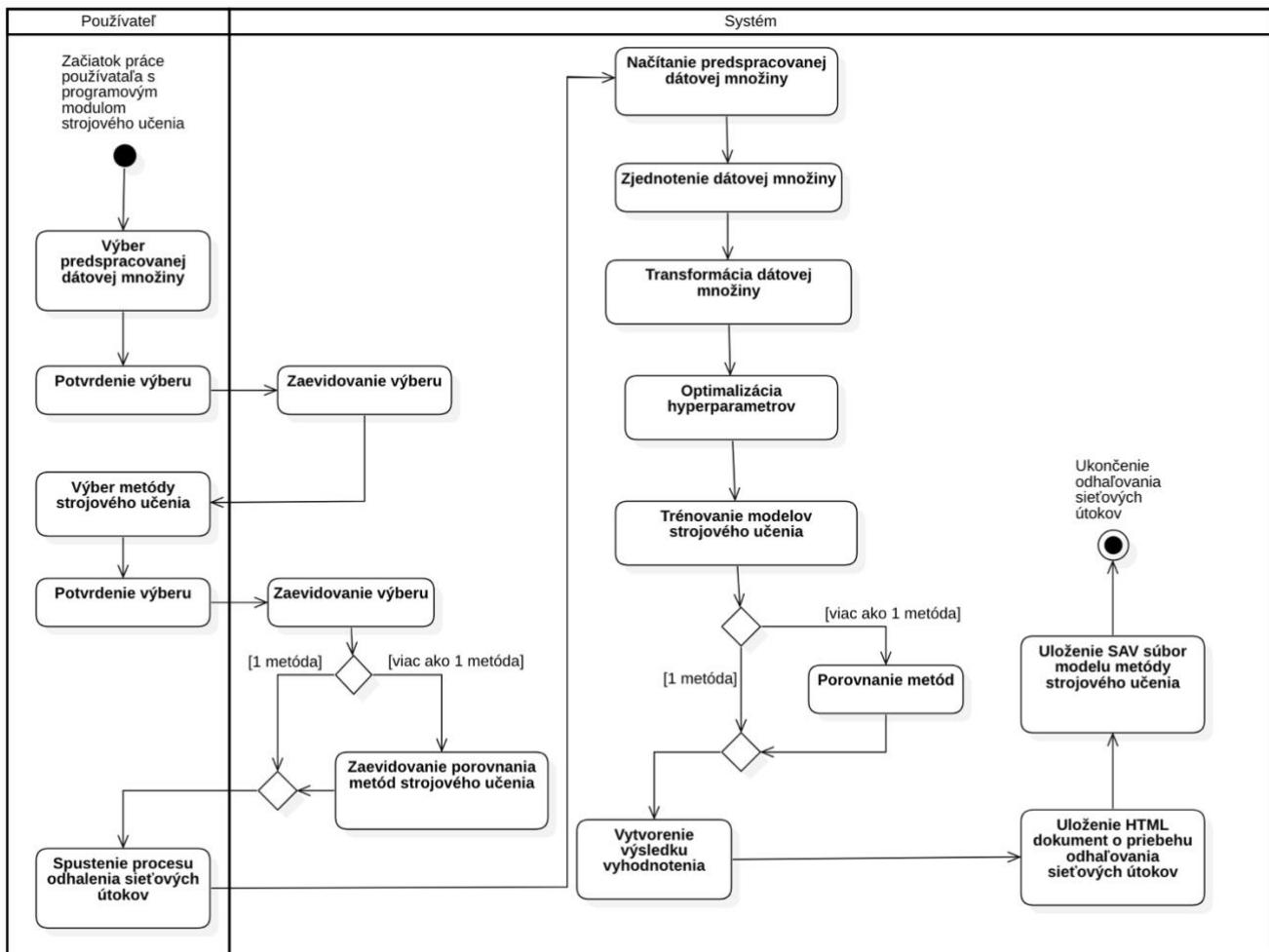
Výsledky metód strojového učenia interpretujeme vo forme percentuálnych úspešností, kontingenčných tabuliek a grafov.

## 4.5. Opis činností programového modulu strojového učenia

Diagram aktivít odhaľovania sieťových útokov pomocou metód strojového učenia na obrázku č. 18 predstavuje proces aplikácie metód strojového učenia na množinu predspracovaných dát, ktorá je výstupom programového modulu predspracovania dátovej množiny.

Používateľ inicializuje proces výberom predspracovanej dátovej množiny a metód/y strojového učenia. Pri výbere metódy strojového učenia sa zistí, či používateľ vybral jednu alebo viac metód. V prípade viacerých metód strojového učenia sa vykoná ich porovnanie po natrénovaní modelu strojového učenia. Výstupom je sumár porovnaní metód strojového učenia, na základe ktorého môžeme ľahko odvodiť, ktorá metóda je najlepšia. Ďalej sa vykoná súbor krokov, ktoré sú potrebné na predprípravu predspracovanej dátovej množiny pre model. Pred samotným trénovaním modelu je dôležité si všimnúť aktivitu optimalizácie hyperparametrov. Tento krok je klúčový pre dosahovanie lepších výsledkov v odhaľovaní anomálnej sieťovej premávky.

Výsledkom je dokument vo formáte HTML, ktorý predstavuje správu o priebehu odhaľovania sieťových útokov. Model metódy strojového učenia sa uloží vo formáte SAV/H5. Výsledné dokumenty sa ukladajú do Google Drive úložiska.



Obrázok 18 – Diagram aktivít programového modulu strojového učenia

## 5. Implementácia

Programové moduly, ktorým sme sa venovali v kapitole 4 Návrh riešenia sa nám podarilo plne implementovať na základe návrhu riešenia. Výsledná architektúra programových modulov je zhodná s architektúrou, ktorú sme navrhli v kapitolách číslo 4.2 až 4.5.

### 5.1. Nastavenie vývojového prostredia

Podľa návrhu vývojového prostredia, ktorému sme sa venovali v kapitole 4.1 Vývojové prostredie sme zvolili vývojové prostredie v prostredí Google Colab. K tomu, aby sme mali prístup k súborom ako: dátové množiny, predspracované dátové množiny, Jupyter Notebook, knižnice a ďalšie súbory potrebné pre beh programových modulov sa musíme prihlásiť do úložného priestoru Google Drive cez náš Google účet pomocou aplikácie Google Drive File Stream. Toto prihlásenie zabezpečuje knižnica *google.colab.drive*. K autentifikácii je potrebné skopírovať do prihlásovacieho poľa unikátny kľúč, ktorý sa vygeneruje po pridelení súhlasu prístupu aplikácie Google Drive File Stream k súborom používateľa. Proces autentifikácie je spoločný pre oba programové moduly.

Ďalšou spoločnou črtou oboch programových modulov je nastavenie runtime typu na Python 3, hardvérového urýchľovača na TPU a tvaru runtime na High-RAM. Nastavenie behu prostredia na vysokovýkonných zariadeniach od spoločnosti Google prebehne spustením kódu [54] určeným na správne rozpoznanie TPU zariadení. Tento kód pridáme do notebooku do novej bunky. Kód by mal vrátiť zoznam osem TPU zariadení dostupných v našom prostredí Colab.

Tím pádom máme pripravené vysoko výkonné výpočtové prostredie od spoločnosti Google, ktoré budeme využívať hlavne v prípade trénovania neurónových sietí, pretože TPU podporuje trénovanie iba vrstvových modelov (sequential) ako je neurónová sieť.

V prípade ak by nám nestačila verzia Google Colab zadarmo, tak Google poskytuje Pro<sup>6</sup> verziu, ktorá poskytuje prioritný prístup k rýchlejším GPU<sup>7</sup> a TPU. Ďalej verzia Pro poskytuje dlhšie spúšťanie Notebookov - procesov a kratší čas nečinnosti čo znamená, že vás odpojuje menej často. V neposlednom rade ponúka viac pamäte RAM – 38GB.

### 5.2. Programový modul predspracovania dátovej množiny

Pre prácu s vybranou dátovou množinou UNSW-NB15 [55] ju najskôr potrebujeme uložiť do nášho Google Drive úložného priestoru, pretože programový modul je vybudovaný tak, aby

<sup>6</sup> <https://colab.research.google.com/signup>

<sup>7</sup> Graphics Processing Unit (GPU) – Grafický procesor

pristupoval k zdrojom z Google Drive úložného priestoru. Ďalej je potrebné stiahnuť databázu GeoLite2Country [18], ktorá je bezplatná geolokačná databáza IP adries. Geolokačnú databázu GeoLite2Country nájdete v prílohe D v adresári \Prilohy\GeoLiteDatabaza\GeoLite2-Country.mmdb. GeoLite2 databázy v porovnaní s databázami GeoIP2 spoločnosti MaxMind sú menej presné. Databázu IP adries budeme potrebovať pri procese obohacovania dátovej množiny.

Programový modul je rozdelený do buniek, kde v každej bunke je časť kódu zodpovedajúca za vykonanie daného kroku predspracovania vybranej dátovej množiny. Jednotlivé kroky programového modulu predspracovania dátovej množiny nájdete na diagrame aktivít na obrázku č. 17. Proces spracovania dátovej množiny opisujú kapitoly nižšie.

### **5.2.1. Výber dátovej množiny a metódy predspracovania**

Vývojové prostredie Google Colab umožňuje pretransformovať bunky Jupyter Notebook do podoby formulárov a tým pádom vieme vytvoriť jednoduché používateľské rozhranie pre jednotlivé nastavenia. V rámci nastavení používateľ zadá názov dátovej množiny, ktorú chce predspracovať a zároveň určí názov predspracovanej dátovej množiny pod akým menom sa má uložiť po spracovaní. V časti nastavenia metódy predspracovania používateľ má možnosť si vybrať akou metódou chce nominálne atribúty predspracovať, či chce použiť One-Hot Encoding alebo vzorkovanie. Ostatným metódam predspracovania dátovej množiny sme sa venovali v kapitole 2.7.4 Predspracovanie dát. Tieto metódy sú v časti zakomponované aj do metódy predspracovania, ktorú používateľ zvolí.

### **5.2.2. Načítanie dátovej množiny**

Dátová množina vo formáte CSV sa načíta do dátovej štruktúry *DataFrame*. Zvyčajne dátové množiny neobsahujú hlavičku, ktorá predstavuje názvy jednotlivých atribútov/stĺpcov, a preto je potrebné zvlášť načítať atribúty a nastaviť ich ako hlavičku dátovej množiny. Načítanie dátovej množiny sa realizuje pomocou metódy *read\_csv*<sup>8</sup> z knižnice *pandas*.

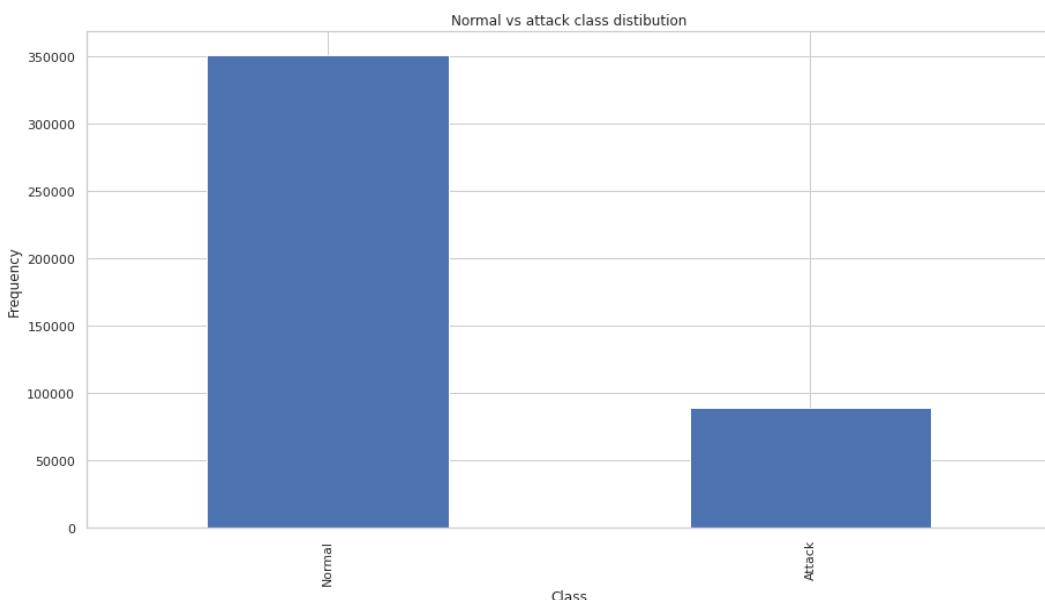
### **5.2.3. Analýza dátovej množiny**

V procese analýzy dátovej množiny najskôr zistíme základné informácie o dátovej množine, ako počet záznamov a počet atribútov. Metódou *describe*<sup>9</sup> z knižnice *pandas.DataFrame* zobrazíme popisné štatistiky dátovej množiny, ktoré sumarizujú centrálnu tendenciu, rozptyl a tvar distribúcie. Táto metóda neberie do úvahy nominálne atribúty a hodnoty s hodnotou NaN. Z popisu dokážeme

<sup>8</sup> [https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read\\_csv.html](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html)

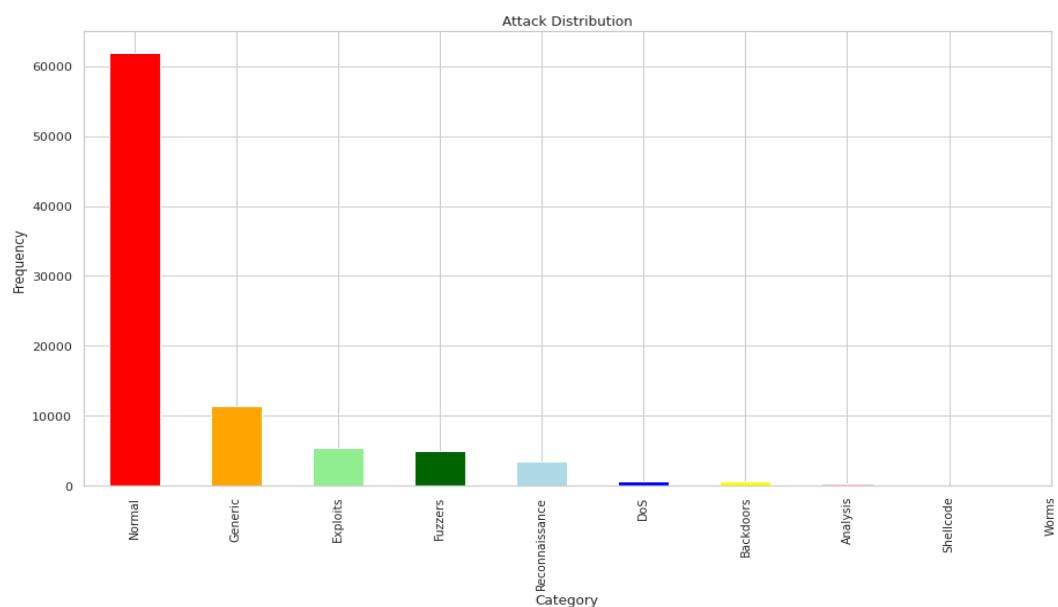
<sup>9</sup> <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html>

vyčítať počet záznamov, aritmetický priemer, smerodajnú odchýlku, minimálnu, maximálnu hodnotu a kvartáli, pričom päťdesiat percentný kvartál predstavuje medián. Tento popis je užitočný z hľadiska zistenia vybočujúcich hodnôt. V tejto fáze predspracovania analyzujeme dátovú množinu aj grafickým spôsobom, tzn. grafmi zobrazíme distribúciu hodnôt jednotlivých atribútov. Tento spôsob je taktiež vhodný na analýzu vybočujúcich hodnôt, kedy z grafu dokážeme vyčítať prahovú hodnotu, od ktorej hodnoty vybočujú v porovnaní s rozložením väčšiny dát. Graf normálnej a útočnej sietovej premávky nám dá prehľad o pomere počtov záznamov, ktorý zohľadníme neskôr pri otázke vzorkovania. Ukážkové grafy normálnej a útočnej premávky zobrazujú obrázky č. 19 a 20. Ďalšia metóda na analýzu dát je korelačná matica, ktorá farebne aj číselne určí mieru korelácie atribútov. Každá bunka v tabuľke zobrazuje koreláciu medzi dvomi atribútmi. Korelačná matica sa používa hlavne na zhrnutie údajov ako vstup do pokročilejšej analýzy. Ukážku korelačnej matice s vysokými koreláciami zobrazuje obrázok č. 21. Použili sme *pearsonov korelačný koeficient* na meranie asociácie medzi dvomi premennými. Na to, aby sme v ďalšom kroku vedeli vhodne aplikovať metódy na čistenie dát musíme najskôr zobraziť počty unikátnych hodnôt pre jednotlivé atribúty pomocou metódy *value\_counts<sup>10</sup>* z knižnice *Series*. Tento spôsob odhalí to, aké hodnoty jednotlivé atribúty nadobúdajú a v akom počte. Tento prístup sa zväčša aplikuje na nominálne atribúty. Ďalej vo fáze analýzy dátovej množiny sa venujeme distribúcii útočnej premávky voči normálnej. Analýza distribúcie údajov je dôležitá z hľadiska kvality údajov.

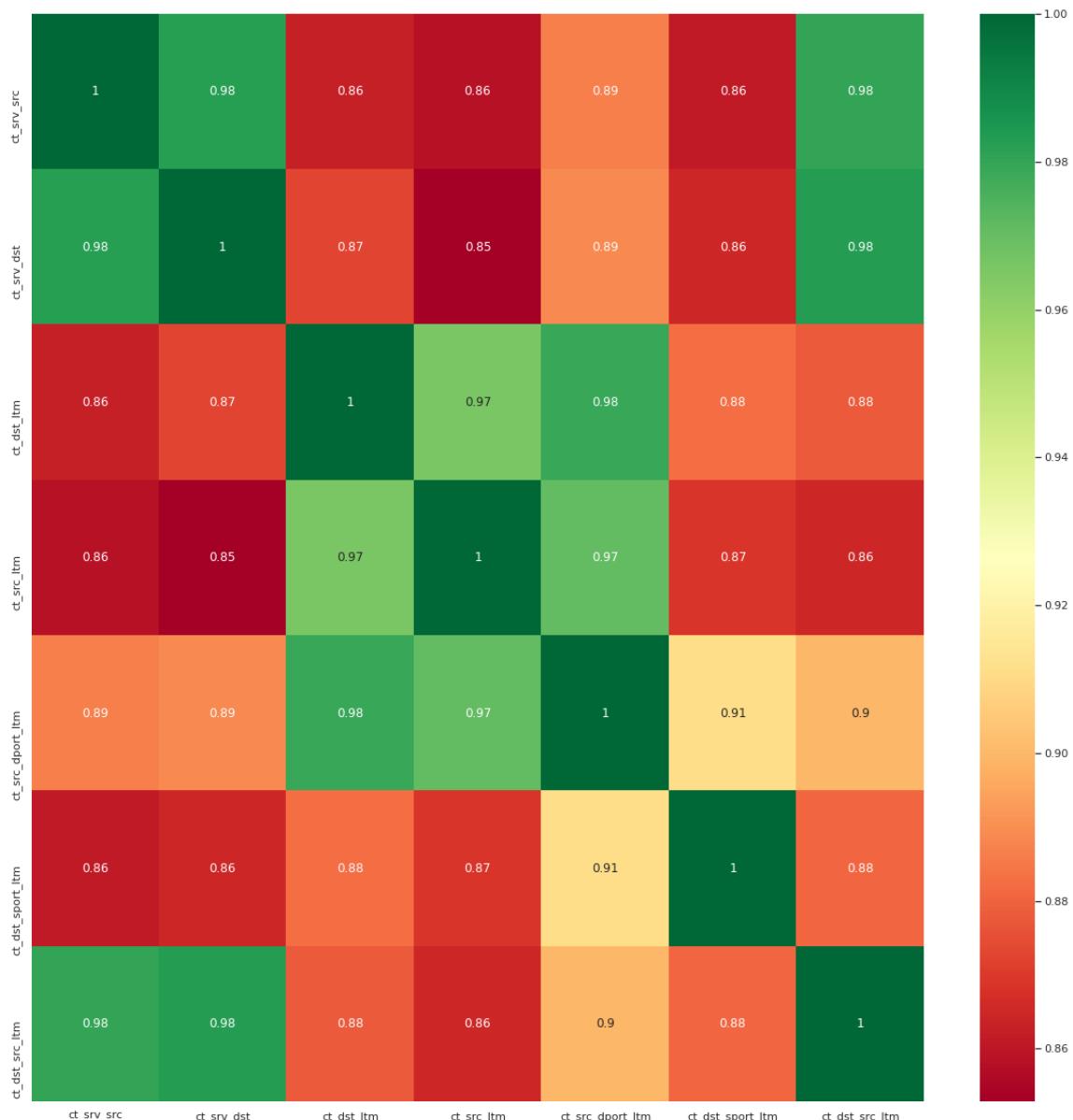


Obrázok 19 – Graf distribúcie normálnej a útočnej premávky

<sup>10</sup> [https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.value\\_counts.html](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.value_counts.html)



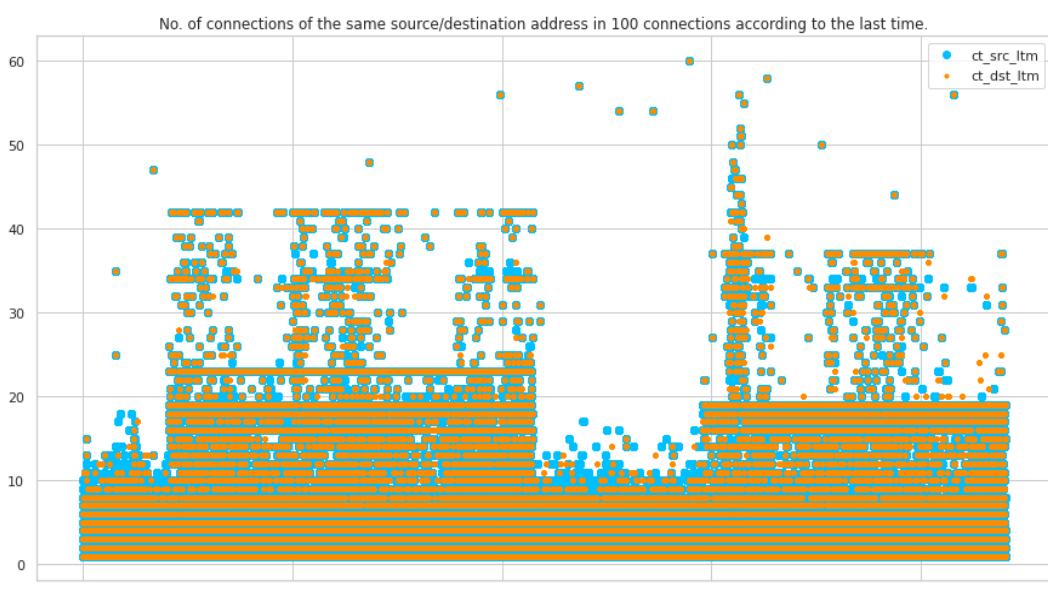
Obrázok 20 – Graf distribúcie útočnej premávky



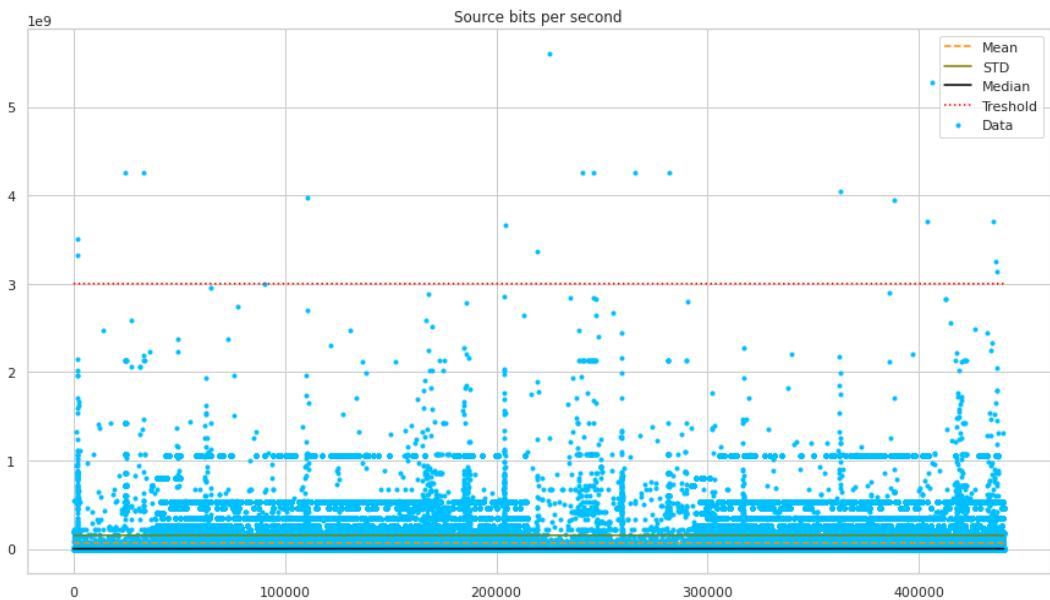
Obrázok 21 – Korelačná matica

#### 5.2.4. Čistenie dát

Z posledného kroku analýzy dátovej množiny dokážeme vyčítať hodnoty, ktoré v danom rozsahu nadobudnutých hodnôt nedávajú zmysel alebo sú jednoducho nekategorizované. V prípade ak týchto hodnôt je málo, tak ich môžeme odstrániť. Samozrejme musíme brať do úvahy prístup spomenutý v kapitole 2.7.4 Predspracovanie dát, časť čistenie dát. Počas fázy čistenia dát sme došli k záveru, že v dátovej množine sa nachádzajú nekategorizované nominálne hodnoty, ktoré sme bud' zmazali alebo premenovali kvôli lepšej interpretácii. Ďalej sme zistili, že hodnoty zdrojových a cieľových portov nadobúdajú hexadecimálny tvar. Tieto hodnoty sme prekonvertovali na decimálny tvar. V súvislosti s číslami portov sme zistili, že niektoré hodnoty presahujú rozsah dynamických portov, ktorej maximálna hodnota je 65 535. Dĺžka času spojenia ukazovala vybočujúce hodnoty s dĺžkou času nad 8 000ms oproti strednej hodnote, ktorá je 60ms. Týchto záznamov bolo iba zopár a na základe dôkladnej analýzy sme ich mohli zmazať. Ostatné atribúty tiež obsahovali vybočujúce hodnoty, ale po analýze typov spojení a toho či ide o normálnu alebo útočnú premávku sme došli k záveru, že vo vybočujúcich hodnotách určitých atribútov je vysoké zastúpenie anomálnej siet'ovej premávky, a tak sme tieto záznamy nemohli zmazať, pretože by sa znížilo už aj tak nízke zastúpenie anomálnej premávky na ešte nižší počet. Tento prípad znázorňuje aj obrázok č. 22, ktorý zodpovedá spojeniam s rovnakou zdrojovou/cieľovou adresou pri 100 pripojeniach podľa posledného času. Ďalej sme zistili, že pri istých atribútoch s vybočujúcimi hodnotami existuje vysoká korelácia s inými atribútmi a tým pádom sme mohli dospiť k záveru, že dané vysoko korelované atribúty budú môcť byť súčasťou vybraných atribútov pri strojovom učení. Obrázok č. 23 zobrazuje distribúciu hodnôt pre zdrojové bity atribútu *Sload*. Nad prahom (červená trhaná čiara) sa nachádza 6 záznamov normálnej a 13 záznamov útočnej premávky.



Obrázok 22 – Korelácia atribútov



Obrázok 23 – Distribúcia hodnôt pre zdrojové byty

### 5.2.5. Doplnenie chýbajúcich hodnôt

Ako sme aj v časti Analýza dátovej množiny spomenuli, tak popisné štatistiky nezohľadňujú chýbajúce hodnoty, a preto ich zobrazenie riešime cez metódu *isna* z už spomenutej knižnice *pandas.DataFrame*. Táto metóda nám vráti zoznam atribútov s chýbajúcimi dátami. Prvým atribútom je *attack\_cat*, ktorý je nominálneho typu a určuje typ útoku. V našom prípade tento atribút môžeme zmazať, pretože našim cieľom nie je predikcia viacerých tried, ale iba binárna predikcia tzn. či je dané sieťové spojenie normálne alebo útočné. Ďalším prístupom k spracovaniu tohto atribútu je jeho ponechanie, pretože od typu útoku môže závisieť ďalšia akcia v prípade zistenia anomálnej premávky. Ponechanie atribútu pre typ útoku vo fáze dopĺňania chýbajúcich hodnôt môže mať tiež pozitívny vplyv, pretože kategória útoku môže ovplyvňovať variabilitu nadobúdaných hodnôt iných atribútov. Ďalšie atribúty s chýbajúcimi hodnotami sú numerické. Tieto atribúty sú nasledovné: *ct\_flw\_http\_mthd*, *is\_ftp\_login*, *ct\_ftp\_cmd* a zodpovedajú: počtu tokov, ktoré majú metódy, ako GET a POST v rámci HTTP; či k FTP relácii pristupuje užívateľ pomocou hesla a počet tokov, ktoré majú príkaz v relácii FTP.

Pre doplnenie numerických chýbajúcich hodnôt sme sa rozhodli pre regresiu. Využívame metódu *RandomForestRegressor*<sup>11</sup> z knižnice *sklearn.ensemble*, pretože používa priemerovanie na zlepšenie predikčnej presnosti a má kontrolu nad pretrénovaním. Predtým, než by sme spustili trénovanie regresného náhodného stromu sme optimalizovali hyperparametre pomocou dvoch metód:

<sup>11</sup> <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>

*RandomizedSearchCV*<sup>12</sup> a *GridSearchCV*<sup>13</sup>, obe z knižnice *sklearn.model\_selection*. Podľa Senapatia [47] náhodné vyhľadávanie (Random Search) je technika, pri ktorej sa na nájdenie najlepšieho riešenia postaveného modelu používajú náhodné kombinácie zvolených hyperparametrov. Ďalej uvádza, že šanca na nájdenie optimálneho hyperparametru je v náhodnom vyhľadávaní pomerne vyššia z dôvodu náhodného vyhľadávacieho vzoru. Maticové vyhľadávanie oproti náhodnému vyhľadávaniu hyperparametrov vyskúša každú kombináciu predvoleného zoznamu hodnôt hyperparametrov a vyhodnotí model pre každú kombináciu. Po vyhodnotení všetkých kombinácií hyperparametrov sa za najlepší model považuje ten, ktorý dosiahol najvyššiu presnosť. Naše riešenie spočíva v kombinácii týchto dvoch metód.

Najprv sme zadefinovali množinu hyperparametrov s hodnotami, ktoré môže každý hyperparametr nadobudnúť podľa dokumentácie danej regresnej metódy. Najskôr sme spustili náhodné vyhľadávanie a na základe výsledku náhodného vyhľadávania sme dokázali určiť dolnú hranicu hodnôt hyperparametrov. Vytvorili sme druhú množinu hyperparametrov s hodnotami vyššími alebo rovnými aké sme dostali náhodným vyhľadávaním a spustili sme optimalizáciu hyperparametrov maticovým vyhľadávaním. Výslednú množinu hyperparametrov sme použili ako parameter v modeli regresného náhodného lesa. Pre každý model predikcie chýbajúcich hodnôt určitého atribútu sme optimalizovali hyperparametre zvlášť. Pre dosahovanie čo najpresnejších odhadov sme použili päťnásobnú krížovú validáciu po päť iterácií. Krížová validácia sa vo veľkej miere používa pri predikcii, ak je potrebné odhadnúť presnosť výkonu prediktívneho modelu.

Pseudo kód na rozdelenie dátovej množiny je nasledovný:

Rozdeľ údaje do dvoch množín, s chýbajúcimi hodnotami a bez chýbajúcich hodnôt bez atribútu, ktorý chceme predpovedať. Množinu s chýbajúcimi hodnotami označ `X_test`. Množinu bez chýbajúcich hodnôt označ `X_train`. Množina `y_train` predstavuje známe hodnoty predpovedaného atribútu. Natrénuj model na množine `X_train` a `y_train` a predpovedaj hodnoty pre `X_test`. Zlúč pôvodné hodnoty s hodnotami množiny `X_test`.

Na základe predchádzajúceho pseudo kódu sme postupne predpovedali chýbajúce hodnoty pre všetky atribúty s chýbajúcimi hodnotami zvlášť. Takýmto spôsobom sme zabezpečili závislosť

<sup>12</sup> [https://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.RandomizedSearchCV.html](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html)

<sup>13</sup> [https://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.GridSearchCV.html](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)

hodnôt, pretože model pri predikcii chýbajúcich hodnôt druhého atribútu bral do úvahy predpovedané hodnoty prvého atribútu atď.

### 5.2.6. Spracovanie kategorických atribútov

Kategorické atribúty sme sa rozhodli spracovať dvomi spôsobmi. Prvý spôsob je priradenie statických numerických hodnôt jednotlivým kategorickým hodnotám. Tento spôsob sa nazýva mapovanie, kedy sa vytvorí slovník s presným mapovaním kategorických hodnôt na numerické. Taký slovník má nasledovnú podobu: `{'ftp': 1, 'smtp': 2, 'dns': 3, ...}`. Pri výbere metódy predspracovania kategorických atribútov pomocou slovníka máme dve možnosti vytvorenia slovníka. Prvý prístup spočíva v priradení numerických hodnôt sekvenčne tak, v akom poradí sú kategorické atribúty. Druhý prístup berie do ohľadu početnosť, čiže kategorické atribúty s najvyššou početnosťou dostanú priradenú hodnotu 1.

Druhým spôsobom predspracovania kategorických atribútov je metódou One-Hot Encoding. Pomocou metódy `get_dummies14` z knižnice `pandas` sme pretransformovali nominálne atribúty na numerické. Týmto metódam sme sa podrobnejšie venovali v kapitole 2.7.4 Predspracovanie dát. Spracovaniu nominálneho atribútu zdrojovej a cieľovej IP adresy sa venujeme v nasledujúcej kapitole.

Kategorické atribúty ktoré sme spracovávali v tejto fáze sú nasledovné: `port, proto, state` a `service`. V prípade protokolov, ktorých je celkový počet 133 sme pridali možnosť ich skratiť pri predspracovaní formou výberu najpoužívanejších protokolov, ktoré sú nasledovné: `tcp, udp, arp, ospf, icmp, gre` a `sctp`. Tieto protokoly zároveň predstavujú protokoly, ktoré Bc. Daniel Gabriš vo svojej diplomovej práci, ktorá je zameraná na detekciu a prevenciu sieťových útokov v softvérovo definovaných sietiach dokáže vyčítať zo simulovanej sietovej premávky. Ostatné menej používané protokoly sme označili príznakom "other".

### 5.2.7. Obohacovanie dát

Nominálny atribút zdrojovej a cieľovej IP adresy sme pretransformovali na ďalší nominálny atribút, ktorý predstavuje krajinu IP adresy pomocou vyššie spomínamej databázy GeoLite2. Je potrebné nainštalovať balík `geoip2`, ktorý poskytuje rozhranie pre programovanie aplikácií pre webové služby a databázy GeoIP2. Instalačný príkaz je nasledovný:

```
!pip -q install geoip2
```

<sup>14</sup> [https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get\\_dummies.html](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html)

Pomocou knižnice *geoip2.database* sme zavolali metódu *country.iso\_code* na získanie označenia krajiny (ISO kód), ku ktorej daná IP adresa patrí. Krajinu IP adresy sme získali pomocou metódy *country(IP)*. V prípade IP adres ktoré sa nenachádzajú v databáze sme ošetrili tak, že sme podľa rozsahu IP adres priradili jeden z príznakov: "Private", "Localhost", "Multicast". Privátne IP adresy sa pohybujú v troch rozmedziach:

- 192.168.0.0 – 192.168.255.255
- 172.16.0.0 – 172.31.255.255
- 10.0.0.0 – 10.255.255.255

Localhost IP adresa je 127.0.0.1. Multicast IP adresy sa pohybujú v rozmedzí od 224.0.0.0 do 239.255.255.255.

Následne sa aplikuje proces spracovania kategorických atribútov z kapitoly vyššie. Tento proces je časovo náročný v prípade transformácie IP adres vo väčšom množstve.

Dátovú množinu sme obohatili aj o dva ďalšie atribúty, ktoré opisujú typ portu. Na základe rozsahu čísla jednotlivých portov vieme určiť, či sa jedná o dobre známy (well-known) port, registrovaný alebo súkromný. Známe porty sa pohybujú v rozmedzí od 0 do 1 023, registrované porty od 1 024 až 49 151 a dynamické porty (tiež nazývané ako súkromné porty) od 49 152 do 65 535. Následne sa aplikuje proces spracovania kategorických atribútov z kapitoly vyššie.

### 5.2.8. Štandardizácia

Pre proces štandardizácie sme použili metódu *StandardScaler*<sup>15</sup> z knižnice *sklearn.preprocessing*, ktorý normalizuje vlastnosti odstránením strednej hodnoty a škáluje hodnoty podľa rozptylu jednotiek. Dôvod, prečo je tento proces nevyhnutný pre metódy strojového učenia je podrobne vysvetlený v kapitole 4.2 Programový modul predspracovania dátovej množiny.

### 5.2.9. Vzorkovanie

Proces vzorkovania je realizovaný dvomi metódami: *SMOTE*<sup>16</sup> a *ADASYN*<sup>17</sup> z knižnice *imblearn.over\_sampling*. Tieto metódy sú prepojené rúrou (pipeline). Rúra slúži na to, aby sa výstup jednej metódy použil ako vstup pre druhú metódu. Podľa Lemaitre a spol. [28] štandardná metóda *RandomOverSampler* v porovnaní s metódami *SMOTE* a *ADASYN* vzorkuje duplikovaním

<sup>15</sup> <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>

<sup>16</sup> [https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over\\_sampling.SMOTE.html](https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html)

<sup>17</sup> [https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over\\_sampling.ADASYN.html](https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.ADASYN.html)

niektorých pôvodných vzoriek menšinovej triedy. *SMOTE* a *ADASYN* generujú nové vzorky interpoláciou. Vzorky použité na interpoláciu/generovanie nových syntetických vzoriek sa však líšia. *ADASYN* sa zameriava na vytváranie vzoriek vedľa originálnych vzoriek, ktoré sú nesprávne klasifikované pomocou klasifikátora K-Nearest Neighbors (KNN). Zatiaľ čo *SMOTE* nerozlišuje rozdiel medzi vzorkami, ktoré sa majú klasifikovať pomocou pravidla najbližších susedov.

### 5.2.10. Uloženie súborov

Uloženie dokumentu vo formáte HTML o priebehu správy predspracovania vybranej dátovej množiny sa realizuje cez príkaz *jupyter* volaním metódy *nbconverter*, ktorý prekonvertuje notebook do HTML formátu a uloží do príslušného adresára v adresnom priestore Google Drive. Príklad takejto správy nájdete v prílohe D v adresári \Prilohy\Spravy\dataset\_3\_MAP\_2020.html.

Pre ďalší účel hlbšej analýzy dátovej množiny sme implementovali *ProfileReport<sup>18</sup>* z knižnice *pandas\_profiling*. Inštalačný príkaz pre *ProfileReport* je:

```
!pip -q install pandas-profiling[notebook,html]
```

Pomocou tohto nástroja dokážeme vygenerovať správu z dátovej štruktúry *DataFrame*. Popri spomínamej metóde *describe*, ktorá tvorí základ pre analýzu a prieskum údajov sme rozšírili nás programový modul na predspracovanie dátovej množiny o správu v HTML formáte pre rýchlu analýzu dát. Táto správa obsahuje informácie ako:

- Typy atribútov v dátovej množine
- Základná analýza: jedinečné, chýbajúce a najčastejšie hodnoty
- Štatistická analýza vrátane popisnej štatistiky
- Histogram
- Korelačné matice (Spearman, Pearson a Kendall)
- Tepelná mapa (heatmap) a dendrogram chýbajúcich hodnôt
- Textová analýza

Príklad takejto správy nájdete v prílohe D v adresári \Prilohy\Spravy\dataset\_3\_MAP\_2020-ProfileReport.html.

<sup>18</sup> <https://github.com/pandas-profiling/pandas-profiling>

Predspracovaná dátová množina sa prekonvertuje do formátu CSV pomocou knižnice *google.colab.files* zavolaním metódy *to\_csv*. Následne sa vygeneruje CSV súbor a uloží sa do príslušného adresára v adresnom priestore Google Drive.

## 5.3. Programový modul strojového učenia

Programový modul je rozdelený do buniek, kde v každej bunke je časť kódu zodpovedajúca za vykonanie daného kroku predspracovania vybranej predspracovanej dátovej množiny pre metódu strojového učenia. Kroky predspracovania predspracovanej dátovej množiny pre metódy strojového učenia sú zjednotenie a transformácia dátovej množiny. Týmto krokom sa venujeme nižšie v podkapitolách tejto kapitoly.

Jednotlivé kroky programového modulu strojového učenia nájdete na diagrame aktivít na obrázku č. 18. Proces programového modulu strojového učenia opisujú kapitoly nižšie.

### 5.3.1. Načítanie, zjednotenie a transformácia predspracovanej dátovej množiny

Predspracovaná dátová množina vo formáte CSV sa načíta do dátovej štruktúry *Set* (sada), pričom jednotlivé položky sady predstavujú samotnú predspracovanú dátovú množinu, ktorá je uložená v dátovej štruktúre *DataFrame*. Prístup k dátovej množine je nasledovný: `datasets['nazovDatovejMnoziny']`. Predspracované dátové množiny sa ukladajú spolu s hlavičkami, a preto nie je potrebné ich osobitne načítať a nastaviť ako hlavičku predspracovanej dátovej množiny. Načítanie dátovej množiny sa realizuje pomocou rovnakej metódy, ktorú sme opísali v kapitole 5.2.2 Načítanie dátovej množiny.

V prípade, ak používateľ načítať viac ako jednu dátovú množinu s rovnakou metódou predspracovania, tak sa tieto dátové množiny môžu zlúčiť, voľba je na používateľovi. Problému zlúčenia dátových množín sme sa venovali v kapitole 4.4 Programový modul strojového učenia. Je dôležité poznamenať, že v prípade zlúčenia rozsiahlych predspracovaných dátových množín môže dôjsť k vyčerpaniu pamäte RAM v prostredí Google Colab. Vyčerpanie pamäte zvyčajne môže nastat pri procese výberu hyperparametrov alebo počas trénovania modelu strojového učenia. Počas testovania prototypu sa nám viackrát stalo, že pamäť RAM veľkosti 38GB nebola postačujúca.

Predspracovaná dátová množina sa pretransformuje na menšie množiny dát podľa návrhu. Štandardne sa využíva metóda *train\_test\_split*<sup>19</sup> z knižnice *model\_selection*. V tejto metóde sa určí veľkosť validačnej vzorky dát vo forme percent. Návratová hodnota metódy je trénovacia a validačná

<sup>19</sup> [http://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.train\\_test\\_split.html](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)

vzorka dát. K tomu, aby sme dostali aj testovaciu vzorku dát potrebujeme metódu dvakrát aplikovať<sup>20</sup>. Vzorky sú rozdelené v pomere trénovacia vzorka: 60%, validačné vzorka: 20%, testovacia vzorka: 20%.

Tým pádom je predspracovaná dátová množina pripravená na použitie metódou strojového učenia.

### 5.3.2. Trénovanie a vyhodnotenie modelu strojového učenia

Rozhodli sme sa vybrať nasledovné inteligentné metódy strojového učenia: Náhodný les<sup>20</sup>, XGradient Boosting klasifikátor<sup>21</sup>, Rozhodovací strom<sup>22</sup>, Logistická regresia<sup>23</sup>, SGD klasifikátor<sup>24</sup>, KNN klasifikátor<sup>25</sup>, Perceptron<sup>26</sup> a neurónové siete<sup>27</sup>: Long Short-Term Memory (LSTM) rekurentná neurónová sieť, Rekurentná neurónová sieť a Hlboká neurónová sieť. Inteligentným metódam strojového učenia sme sa venovali v kapitole 4.4 Programový modul strojového učenia. Klasifikátor XGradient Boosting je potrebné nainštalovať v prostredí Google Colab pomocou príkazu:

```
!pip -q install xgboost
```

Používateľ má možnosť určiť či chce model strojového učenia trénovať na dátovej množine so sadou vybraných atribútov. Nás programový modul umožňuje používateľovi vybrať z troch prístupov výberu najlepších atribútov. Prvý prístup je výber  $k$  najlepších atribútov z celkovej dátovej množiny pomocou metódy *SelectKBest*<sup>28</sup> z knižnice *sklearn.feature\_selection*, pričom  $k$  je počet najlepších atribútov. Druhý prístup je automatizovaný výber používateľom zadefinovaných atribútov. Tretí prístup spočíva vo výbere najlepších atribútov z modelu strojového učenia pomocou vybraných metód.

Pre logistickú regresiu sme použili metódu *RFE*<sup>29</sup> (Recursive Feature Elimination), ktorá vytvára rebríček hodnotení jednotlivých atribútov s rekurzívnym odstránením. Pre parameter

<sup>20</sup> <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

<sup>21</sup> <https://xgboost.readthedocs.io/en/latest/>

<sup>22</sup> <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

<sup>23</sup> [https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LogisticRegression.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)

<sup>24</sup> [https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.SGDClassifier.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html)

<sup>25</sup> <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>

<sup>26</sup> [https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.Perceptron.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html)

<sup>27</sup> <https://keras.io/layers/recurrent/>

<sup>28</sup> [https://scikit-learn.org/stable/modules/generated/sklearn.feature\\_selection.SelectKBest.html](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html)

<sup>29</sup> [https://scikit-learn.org/stable/modules/generated/sklearn.feature\\_selection.RFE.html](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html)

*estimator* (dozorca) sme použili model logistickej regresie. Metóda vracia zoznam atribútov s príslušným hodnotením. Vyberú sa atribúty, ktorých  $p$  hodnota je menšia ako 0.05. Podľa konvencie je pravdepodobnosťná hodnota  $p$  bežne stanovená na hodnotu 0.05 pri testovaní štatistickej hypotézy. Pre SGD klasifikátor sme použili metódu *Nystroem*<sup>30</sup>, ktorá konštruuje približnú mapu atribútov pre ľubovoľné jadro s použitím podmnožiny údajov ako základ. Pre metódy rozhodovací strom, extreme gradient boosting a náhodný les sme ako metódu na výber najlepších atribútov použili metódu *SelectFromModel*<sup>31</sup>, ktorý je meta-transformátor pre výber najlepších atribútov na základe dôležitých váh. Pre ostatné metódy strojového učenia platí druhý prístup výberu najlepších atribútov. Metódy *RFE* a *SelectFromModel* sú súčasťou knižnice *feature\_selection*. Metóda *Nystroem* sa nachádza v knižnici *kernel\_approximation*.

Pseudo kód výberu najlepších atribútov je nasledovný:

Ak je príznak výberu najlepších atribútov pravdivý a predspracovaná dátová množina nebola predspracovaná One-Hot Encodingom, tak optimalizuj hyperparametre modelu so všetkými atribútmi predspracovanej dátovej množiny. Po optimalizovaní hyperparametrov vyber najlepšie atribúty z modelu a znova optimalizuj hyperparametre pre model s najlepšími atribútmi. Výsledný model s optimalizovanými hyperparametrami použi na trénovanie. V prípade ak predspracovaná dátová množina bola predspracovaná metódou One-Hot Encoding, vyber najlepšie atribúty zo základného modelu (bez alebo s minimálnym počtom parametrov) a potom optimalizuj hyperparametre pre model s najlepšími atribútmi. Výsledný model s optimalizovanými hyperparametrami použi na trénovanie.

Pre výber optimálnych hyperparametrov sme použili kombináciu *RandomizedSearchCV* a *GridSearchCV*, ktoré sme podrobnejšie popísali v kapitole 5.2.4 Čistenie dát. Kombinácia *RandomizedSearchCV* a *GridSearchCV* je použitá pri klasifikátoroch na báze stromu. Pre model SGD klasifikátor sme spočiatku použili metódu *bestFit* z knižnice *parfit* od Carpentera [9], ktorá je určená na paralelizáciu učenia modelu a flexibilné vyhodnocovanie modelov strojového učenia. Inštalačia knižnice *parfit* sa realizuje pomocou príkazu:

<sup>30</sup> [https://scikit-learn.org/stable/modules/generated/sklearn.kernel\\_approximation.Nystroem.html](https://scikit-learn.org/stable/modules/generated/sklearn.kernel_approximation.Nystroem.html)

<sup>31</sup> [https://scikitlearn.org/stable/modules/generated/sklearn.feature\\_selection.SelectFromModel.html](https://scikitlearn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html)

```
!pip -q install parfit
```

Táto metóda umožňuje užívateľovi vykonať vyčerpávajúce mriežkové vyhľadávanie optimálnych parametrov modelu. Tento prístup sme časom zavrhli z dôvodu vyčerpania pamäte RAM. Výber optimálnych hyperparametrov sme nahradili už spomínanou metódou *RandomizedSearchCV*. *RandomizedSearchCV* sa ďalej používa pre perceptron, KNN a logistická regresia. Pre dosahovanie čo najpresnejších odhadov sme opäť použili päťnásobnú krížovú validáciu po päť iteráciu.

Dôležité je poznamenať výnimku v pseudo kóde, kedy sa pri výbere najlepších atribútov predspracovanej dátovej množiny nerobí dvojnásobný výber hyperparametrov ak predspracovaná dátová množina bola predspracovaná metódou One-Hot Encoding. Toto je z dôvodu prečerpania pamäte RAM, ktorý sme spomenuli v kapitole č. 5.3.1. Tu vzniká problém nedostatku pamäte z dôvodu veľkého objemu dát po spracovaní kategorických atribútov metódou One-Hot Encoding.

Optimalizácia hyperparametrov pre neurónové siete je zložitejší proces. Pri neurónových sietach je najbežnejšie používaným optimalizačným algoritmom GD (Gradient Descent) podobne ako pri SGD klasifikátore. Objektívna funkcia používaná pri GD je stratová funkcia *loss*. Naším cieľom je preto minimalizovanie tejto funkcie. Rozhodli sme sa využiť metódy knižnice Keras<sup>32</sup> pre optimalizáciu neurónovej siete. Keras je vysokoúrovňové rozhranie pre programovanie aplikácií pre neurónové siete napísané v programovacom jazyku Python a schopné bežať na TensorFlow<sup>33</sup> platforme. Keras bol vyvinutý so zameraním na umožnenie rýchleho experimentovania.

Pri neurónových sietach máme viac možností ich optimalizácie, jednou z nich je použitie sady funkcií pomocou spätného volania *callback*<sup>34</sup>. Spätné volanie je skupina funkcií, ktoré sa majú uplatniť v daných fázach výcvikového postupu. Pomocou spätných volaní môžeme získať prehľad o vnútorných stavoch a štatistikách modelu počas výcviku. Jednotlivé metódy spätných volaní sa volajú v každej fáze výcviku. Vybrané metódy spätných volaní sú nasledovné:

- **EarlyStopping** – Metóda, ktorá zastaví tréning, keď sa sledovaná metrika prestane zlepšovať.  
V našom prípade sa sleduje hodnota stratovej funkcie *loss*.
- **LearningRateScheduler** – Metóda, ktorá mení rýchlosť výučby počas tréningu, tzv. *learning rate*. Parametrom tejto funkcie je časovač (scheduler), ktorý v ideálnom prípade má mať tvar klesajúcej exponenciálnej krivky.

<sup>32</sup> <https://keras.io>

<sup>33</sup> <https://www.tensorflow.org>

<sup>34</sup> <https://keras.io/callbacks/#callback>

- **ReduceLROnPlateau** – Ak sa sledovaná metrika prestane zlepšovať, metóda zníži rýchlosť učenia.
- **TerminateOnNaN** - Spätné volanie, ktoré ukončí výcvik, keď sa vyskytne strata NaN.

Ďalšími optimalizačnými parametrami neurónovej siete sú: počet epoch (epochs) a veľkosť šarže (batch size). Pri trénovaní neurónovej siete nepostačuje priechod celého súboru údajov naraz neurónovou sieťou. Neurónovej siete musíme celý súbor údajov odovzdať na viackrát, pretože aj samotná optimalizácia výučby – Gradient Descent je iteratívny proces. Aktualizácia váh medzi neurónmi pomocou jedného priechodu alebo jednej epochy teda nestačí. Veľkosť šarže zas predstavuje celkový počet záznamov trénovacej množiny prítomných v jednej šarži. Trénovací súbor údajov je teda potrebné rozdeliť na istý počet šarží. Výhodou použitia šarží je hned' niekoľko. Použitie šarží vyžaduje menej pamäte RAM, pretože neurónová sieť sa trénuje pomocou menšieho počtu vzoriek a tým pádom celkový tréning vyžaduje menej pamäte. Sieť pomocou malých šarží taktiež zvyčajne trénuje rýchlejšie. Šaržu je potrebné nastaviť na optimálnu hodnotu, ináč v prípade nízkej šarže bude nižšia presnosť gradientu odhadu. Ďalším optimalizačným parametrom je *dropout*, ktorý zodpovedá za tzv. ignorovanie jednotiek neurónov počas tréningovej fázy určitého súboru neurónov, ktorý je vybraný náhodne. Dôvod prečo je potrebné zaviesť tento parameter je preto, lebo plne prepojená vrstva neurónovej siete využíva väčšinu parametrov, a tým pádom si neuróny počas tréningu vytvárajú vzájomnú závislosť. Táto závislosť neurónov obmedzuje individuálnu silu každého neurónu, čo vedie k pretrénovaniu. Za štvrtý vplyvný optimalizačný parameter pokladáme optimalizačnú funkciu. Optimalizačná funkcia je používaná na zmenu atribútov neurónovej siete, ako sú váhy a rýchlosť učenia. Taktiež zodpovedá za znižovanie straty a zabezpečenie čo najpresnejších výsledkov. Na nájdenie optimálnych hodnôt vyššie spomenutých a mnoho ďalších parametrov neurónovej siete sme použili taktiež metódu *RandomizedSearchCV* s krížovú validáciu pomocou tzv. wrapperu *KerasClassifier*<sup>35</sup> z Keras rozhrania pre programovanie aplikácií.

Po výbere najlepších atribútov predspracovanej dátovej množiny je potrebné pretransformovať trénovaciu, validačnú a testovaciu vzorku dát na základe najlepších atribútov.

Na vyhodnotenie úspešnosti modelu používame nasledovné metriky:

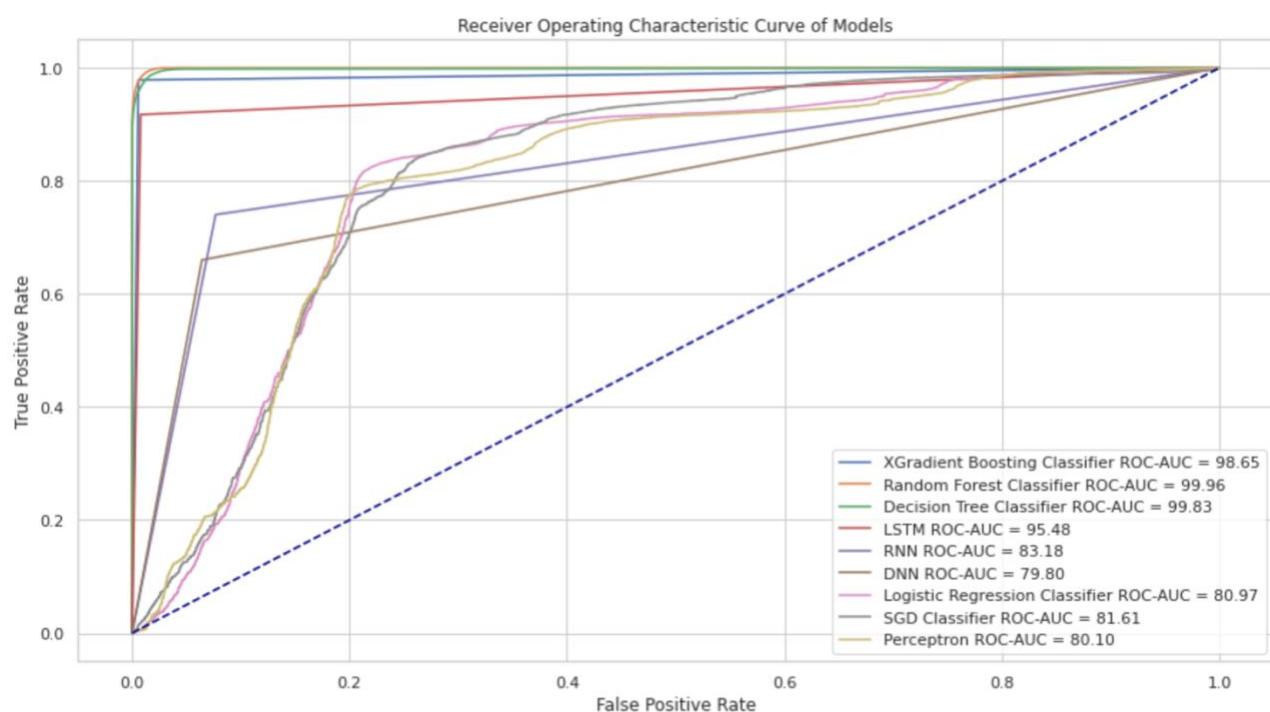
- Správnosť, presnosť, F1-skóre, podporu
- Štandardnú desať-násobnú krížovú validáciu
- Stratifikovanú desať-násobnú K-fold krížovú validáciu
- Kontingenčná tabuľka

<sup>35</sup> <https://keras.io/scikit-learn-api/>

- Krivka Receiver operating characteristic (ROC) – Area under curve (AUC)

### 5.3.3. Porovnanie metód strojového učenia

Pre účely rýchleho porovnania metód strojového učenia sme sa rozhodli vybrať nasledovné metriky: správnosť, F1-skóre, ROC-AUC. Výsledky jednotlivých metód strojového učenia sa ukladajú v JSON formáte. Pri interpretácii výsledkov sa najskôr výsledky zoradia podľa správnosti zostupne. Formou tabuľky vypíšeme výsledky jednotlivých metód určené na porovnanie. Zároveň sa vytvorí jeden graf ROC krviek jednotlivých modelov s príslušnými AUC hodnotami. Krvky sú farebne odlíšené. Takéto porovnanie znázorňuje nasledujúci obrázok.



	Model	Accuracy	F1-score
Evaluation	XGradient Boosting Classifier	99.09	97.88
	Random Forest Classifier	99.04	97.76
	Decision Tree Classifier	98.66	96.87
	LSTM	97.61	94.29
	RNN	88.39	73.28
	DNN	87.65	69.69
	Logistic Regression Classifier	79.42	62.84
	SGD Classifier	78.52	0.19
	Perceptron	77.69	23.81

Obrázok 24 – Porovnanie metód strojového učenia

Pre ďalšie porovnanie metód strojového učenia sme sa rozhodli implementovať automatizovaný nástroj TPOT<sup>36</sup>. TPOT je nástroj automatizovaného strojového učenia Python, ktorý optimalizuje rúry strojového učenia pomocou genetického programovania. TPOT je potrebné nainštalovať v prostredí Google Colab pomocou príkazu:

```
!pip -q install tpot
```

Výstup TPOT-u je Python súbor s kódom pre implementáciu najlepšej metódy strojového učenia.

#### 5.3.4. Uloženie súborov

Uloženie dokumentu vo formáte HTML o priebehu správy klasifikácie sietových útokov sa realizuje cez príkaz *jupyter* volaním metódy *nbconverter*, ktorý prekonvertuje notebook do HTML formátu a uloží do príslušného adresára v adresnom priestore Google Drive. Príklad takejto správy nájdete v prílohe D v adresári \Prilohy\Spravy\UNSW-NB15\_3-4\_MAP\_SIM.html.

Model strojového učenia sa uloží vo formáte SAV (modely neurónových sietí vo formáte H5) do príslušného adresára v adresnom priestore Google Drive. Metóda na uloženie modelu strojového učenia je *dump* knižnice *joblib*. Príklady modelov strojového učenia nájdete v prílohe D v adresári \Prilohy\Modely.

<sup>36</sup> <http://epistasislab.github.io/tpot/>



## 6. Overenie riešenia

Overenie správnosti riešenia programových modulov budeme realizovať na základe jednotlivých bodov funkcionálnych vlastností z kapitoly 3.1 Funkcionálne požiadavky.

- Výber rôznych metód strojového učenia.
  - Programový modul strojového učenia umožňuje používateľovi vybrať metódy strojového učenia a následne spustiť klasifikáciu nad vybranou množinou predspracovaných dát.
- Výber dátovej množiny na základe typu predspracovania, na ktorú sa má aplikovať zvolená metóda strojového učenia.
  - Programový modul strojového učenia umožňuje používateľovi vybrať, nad akou dátovou množinou chce aplikovať vybranú metódu strojového učenia. Používateľ má na výber sadu predspracovaných dátových množín.
- Zobrazenie a úprava parametrov vybranej metódy strojového učenia.
  - Programový modul strojového učenia zobrazuje parametre vybranej metódy strojového učenia. Taktiež umožňuje nastaviť tieto parametre.
- Vytvorenie výstupu klasifikácie v textovej aj grafickej podobe.
  - Programový modul strojového učenia zobrazuje výstup klasifikácie po aplikovaní vybranej metódy strojového učenia nad vybranou dátovou množinou dát vo forme tabuľky, grafu a percentuálnej úspešnosti.
- Porovnanie rôznych metód strojového učenia.
  - Programový modul strojového učenia vytvára porovnanie metód strojového učenia formou tabuľky a grafu.
- Výber dátovej množiny na spracovanie.
  - Programový modul určený na spracovanie dátových množín umožňuje používateľovi vybrať nepredspracovanú dátovú množinu.
- Výber rôznych metód spracovania dátovej množiny.
  - Programový modul určený na spracovanie dátovej množiny umožňuje používateľovi vybrať rôzne metódy predspracovania dátovej množiny a následne spustiť predspracovanie nad vybranou množinou dát.
- Možnosť zobraziť a uložiť proces spracovania dátovej množiny.
  - Programový modul určený na spracovanie dátovej množiny ukladá proces spracovania dátovej množiny spolu s výstupmi jednotlivých krokov spracovania do dokumentu vo

formáte HTML. Spracovanú dátovú množinu taktiež ukladá vo formáte CSV. HTML dokument a CSV súbor sa ukladá do Google Drive úložného priestoru používateľa.

- Možnosť zobraziť a uložiť proces programového modulu strojového učenia.
  - Programový modul strojového učenia ukladá proces práce strojového učenia spolu s výstupmi jednotlivých krokov do dokumentu vo formáte HTML. Taktiež ukladá model metódy strojového učenia vo formáte SAV/H5. HTML dokument a SAV/H5 súbor sa ukladá do Google Drive úložného priestoru používateľa.

## 6.1. Experimentálne overenie riešenia

Na základe prvotných predspracovaných dátových množín sme aplikovali prvotné vybrané metódy strojového učenia (viď. kapitolu 4.4 Programový modul strojového učenia). Programový modul strojového učenia vyhodnotil vybrané metódy strojového učenia podľa tabuľky č. 12. Hodnoty sú uvedené pre správnosť a F1-skóre v percentách pre testovaciu vzorku dát.

Tabuľka 12 – Výsledky experimentálnych testov – správnosť/F1-skóre

Dátová množina	Logistická regresia	SGD klasifikátor	Rozhodovací strom	Náhodný les
dataset_4_MAP_TOPF_F	98.43/96	98.02/95	99.25/98	99.44/99
dataset_4_MAP_TOPF_T	98.34/96	<b>79.66/0</b>	98.85/97	98.95/98
dataset_4_MAP_RESAMP_TOPF_F	98.90/99	98.65/99	99.40/99	99.67/100
dataset_4_MAP_RESAMP_TOPF_T	98.85/99	<b>50.09/67</b>	98.93/99	99.26/99
dataset_4_OHE_TOPF_T	-/-	<b>72.55/10</b>	99.48/99	99.41/99
dataset_4_OHE_RESAMP_TOPF_T	-/-	<b>49.56/63</b>	98.91/99	99.50/99

Vysvetlenie tabuľky:

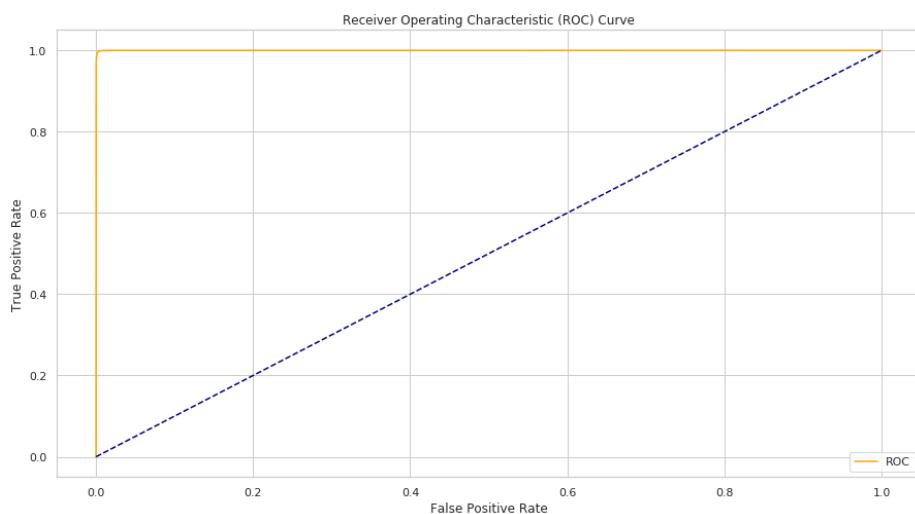
- Označenie MAP v názve dátovej množiny označuje mapovanie a OHE označuje One-Hot Encoding kategórických atribútov. Dátové množiny s označením MAP majú počet atribútov 50 a dátové množiny s označením OHE majú 218 atribútov.
- Označenie RESAMP v názve dátovej množiny označuje, že dátová množina je prevzorkovaná. Prevzorkované dátové množiny majú počet záznamov 702 296 oproti dátovej množine bez vzorkovania s počtom záznamov 440 042.
- Označenie TOPF\_T/F v názve dátovej množiny označuje výber najlepších atribútov z modelu strojového učenia. V prípade T (pravda) sa vybrali najlepšie atribúty a v prípade F (nepravda) sa nevybrali.

- Hodnoty označené červenou farbou majú veľký výkyv správnosti voči F1-skóre.
  - Prázdné hodnoty boli spôsobené vznikom fenoménu multikolinearita.

Na základe experimentálnych testov môžeme dôjsť k záveru, že najspoľahlivejšou metódou strojového učenia je náhodný les, pretože v každom experimentálnom teste dosahoval veľmi podobné, stále a vysoké hodnoty správnosti a F1-skóre. Rozhodovací strom oproti náhodnému lesu dosahoval o niečo menej presné hodnoty správnosti klasifikácie a F1-skóre. Treťou najsprávnejšou metódou klasifikácie je logistická regresia. V prípade predspracovania kategorických atribútov metódou One-Hot Encoding sme sa stretli s fenoménom multikolinearita, ktorú opisujem v kapitole č. 4.2., a preto metódu logistická regresia sme nedokázali vyhodnotiť v niektorých prípadoch aj napriek aplikovania metódy postupného odoberania vybratých atribútov. Najhoršie výsledky dosahoval SGD klasifikátor. Výsledky tejto metódy strojového učenia majú veľké výkyvy v hodnotách správnosti modelu voči F1-skóre a tým pádom môžeme dôjsť k záveru, že model nie je spoľahlivý. Dôležité je poznamenať, že pre výber najlepších hyperparametrov pri experimentálnom overení riešenia sme použili pre modely logistická regresia a SGD klasifikátor metódu *bestFit*, ktorej sme sa venovali v kapitole 5.3.2 Trénovanie a vyhodnotenie modelu strojového učenia.

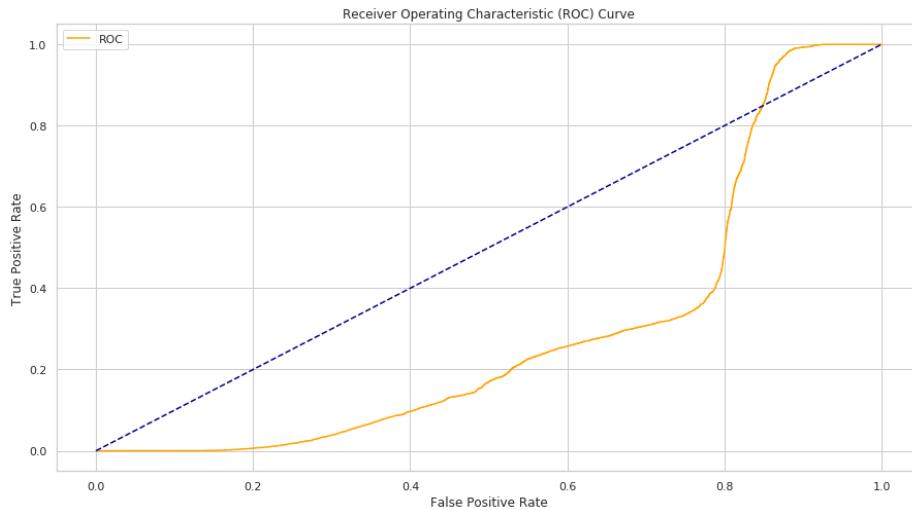
Ďalej sme zistili, že výber najlepších atribútov má za následok mierny pokles správnosti vyhodnotenia modelu. Ďalším zistením je fakt, že medzi rôznymi metódami predspracovania dátovej množiny nie je veľký rozdiel vo výsledkoch. Strojové učenie, ktoré sa učilo na dátovej množine predspracovanou metódou s prevzorkovaním dosahovalo lepšie výsledky.

Na nasledujúcom obrázku č. 25 je ROC krivka pre najlepší model náhodného lesa so správnosťou 99.67%, F1-skóre 100% a AUC hodnotou 99.99%.



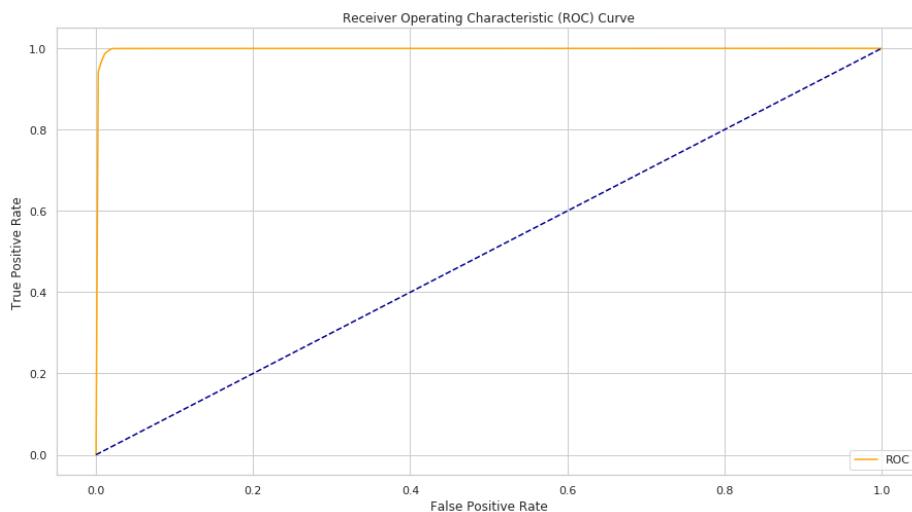
Obrázok 25 – ROC krivka najlepšieho modelu náhodného lesa

Na nasledujúcom obrázku č. 26 môžete vidieť ROC krivku SGD klasifikátora s úspešnosťou 79.66%, F1-skóre 0% a AUC hodnotou 29.01%. V porovnaní s náhodným lesom môžeme dôjsť k záveru, že model SGD klasifikátora v tomto prípade nedokázal rozpoznať normálnu siet'ovú premávku od anomálnej.

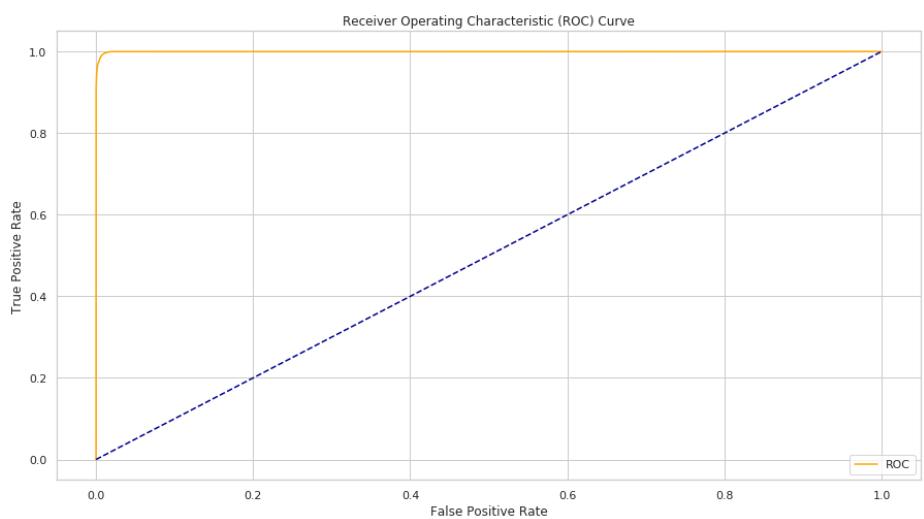


Obrázok 26 – ROC krivka SGD klasifikátora

Nasledujúce obrázky porovnávajú mieru zakrivenia ROC krivky (oblasť pod krivkou) rozhodovacieho stromu a náhodného lesa. Obrázok č. 27 zobrazuje ROC krivku rozhodovacieho stromu s AUC hodnotou 99.81%. Obrázok č. 28 znázorňuje ROC krivku náhodného lesa s AUC hodnotou 99.97%. Hodnota AUC určuje, na koľko percent je schopný model rozlísiť rozdiel medzi dvomi triedami.



Obrázok 27 – ROC krivka rozhodovacieho stromu



Obrázok 28 – ROC krivka náhodného lesa

## 6.2. Vyhodnotenie výsledkov riešenia

Pre účely overenia riešenia sme predspracovali dve dátové podmnožiny UNSW-NB15\_3 a UNSW-NB15\_4 so štyrmi metódami predspracovania. Tieto štyri metódy sú nasledovné:

1. Mapovanie kategorických atribútov bez prevzorkovania a s výberom najlepších atribútov
2. Mapovanie kategorických atribútov s prevzorkovaním a s výberom najlepších atribútov
3. Mapovanie kategorických atribútov bez prevzorkovania a s výberom najlepších atribútov rozšírene o nové atribúty vzniknuté obohacovaním dátovej množiny
4. Mapovanie kategorických atribútov s prevzorkovaním a s výberom najlepších atribútov rozšírene o nové atribúty vzniknuté obohacovaním dátovej množiny
5. One-Hot Encoding kategorických atribútov bez prevzorkovania a s výberom najlepších atribútov
6. One-Hot Encoding kategorických atribútov s prevzorkovaním a s výberom najlepších atribútov
7. One-Hot Encoding kategorických atribútov bez prevzorkovania s výberom najlepších atribútov rozšírene o nové atribúty vzniknuté obohacovaním dátovej množiny
8. One-Hot Encoding kategorických atribútov s prevzorkovaním a s výberom najlepších atribútov rozšírene o nové atribúty vzniknuté obohacovaním dátovej množiny

Už počas analýzy dátovej množiny vo fáze predspracovania dátových množín sme spozorovali vysoko korelované atribúty. Koreláciu týchto atribútov znázorňuje aj korelačná matica na obrázku č. 21. Autor Moustafa a Slay [61] taktiež odkazujú na rovnaké atribúty, ktoré sme vybrali za najlepšie. Najlepšie a zároveň najkorelovanejšie atribúty sme ďalej rozšírili o ďalšie štyri. Výsledný zoznam najlepších dvanásť atribútov je:

1. proto – Použitý protokol
2. Spkts – Počet prenesených paketov zo zdroja do cieľa
3. Dpkts – Počet prenesených paketov z cieľa do zdroja
4. sbytes – Počet prenesených bajtov zo zdroja do cieľa
5. dbytes – Počet prenesených bajtov z cieľa do zdroja
6. ct\_srv\_src – Počet spojení, ktoré obsahujú rovnakú službu a zdrojovú adresu v 100 pripojeniach podľa posledného času
7. ct\_srv\_dst – Počet spojení, ktoré obsahujú rovnakú službu a cieľovú adresu v 100 pripojeniach podľa posledného času

8. `ct_src_ltm` – Počet spojení s rovnakou zdrojovou adresou v 100 pripojeniach podľa posledného času
9. `ct_dst_ltm` – Počet spojení s rovnakou cieľovou adresou v 100 pripojeniach podľa posledného času
10. `ct_src_dport_ltm` – Počet spojení s rovnakou zdrojovou adresou a cieľovým portom v 100 pripojeniach podľa posledného času
11. `ct_dst_sport_ltm` – Počet spojení rovnakej cieľovej adresy a zdrojového portu v 100 pripojeniach podľa posledného času
12. `ct_dst_src_ltm` – Počet spojení toho istého zdroja a cieľovej adresy v 100 pripojeniach podľa posledného času

V prípade protokolov, ktorých je celkový počet 133 sme pridali možnosť ich skratiť pri predspracovaní formou výberu najpoužívanejších protokolov, ktoré sú nasledovné: *tcp*, *udp*, *arp*, *ospf*, *icmp*, *gre*, *sctp*. Ostatné menej používané protokoly sme označili príznakom "other".

Druhá množina atribútov s novými atribútmi, ktoré vznikli vďaka obohacovaniu dátovej množiny rozširujú najlepšie atribúty zo zoznamu vyššie. Druhá podmnožina atribútov je:

1. `srcip` – ISO kód krajiny zdrojovej IP adresy
2. `dstip` – ISO kód krajiny cieľovej IP adresy
3. `sport` – Číslo zdrojového portu
4. `sporttype` – Typ zdrojového portu
5. `dsport` – Číslo cieľového portu
6. `dsporttype` – Typ cieľového portu
7. `service` – Použitá služba

Výsledné množiny atribútov:

1. `proto, Spkts, Dpkts, sbytes, dbytes, ct_srv_src, ct_srv_dst, ct_src_ltm, ct_dst_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm`
2. `srcip, dstip, sport, sporttype, dsport, dsporttype, service, proto, Spkts, Dpkts, sbytes, dbytes, ct_srv_src, ct_srv_dst, ct_src_ltm, ct_dst_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm`

Korelačná matica na obrázku KM\_UNSW-NB15\_3-4\_MAP\_SIM.PNG z prílohy D v adresári \Prilohy\KorelacneMatice zobrazuje vysokú koreláciu vybratých atribútov (12) z prvej množiny so spôsobom predspracovania č 1. Časť atribútov v poradí 6 až 12 z prvej množiny má vysokú pozitívnu koreláciu, ale nízku zápornú koreláciu s ostatnými atribútmi 1 až 5. Medzi atribútmi v poradí 1 až 5 je vo väčšine prípadov vysoká pozitívna korelácia.

Korelačná matica na obrázku KM\_UNSW-NB15\_3-4\_OHE\_SIM.PNG z prílohy D v adresári \Prilohy\KorelacneMatice zobrazuje už menšie hodnoty korelácií vybratých atribútov z prvej množiny so spôsobom predspracovania č. 5. Po predspracovaní dátovej množiny s týmto spôsobom sme získali dátovú množinu s 19 atribútmi. Z korelačnej matice môžeme vidieť, že spôsob predspracovania č. 5 spôsobuje stratu korelácie medzi atribútmi. Korelácie medzi jednotlivými atribútmi nadobúdajú nízke až stredné hodnoty. Dôležité je všimnúť veľmi vysokú koreláciu medzi protokolmi TPC a UDP. Tieto protokoly sú zároveň najpoužívanejšími v celej dátovej množine so zastúpením počtom TCP: 1 495 074 a UDP: 990 435. Protokol TCP d'alej nadobúda záporné stredne vysoké korelačné hodnoty s atribútmi v poradí 6 až 12 z prvej množiny atribútov. Protokol UDP nadobúda kladné stredne vysoké korelačné hodnoty s atribútmi 6-12 z prvej množiny atribútov.

Podobný jav ako pri protokoloch z predchádzajúceho príkladu môžeme pozorovať aj na obrázku KM\_UNSW-NB15\_3-4\_MAP\_SIM\_2.PNG z prílohy D v adresári \Prilohy\KorelacneMatice. Tento obrázok zobrazuje korelačnú matice druhej množiny atribútov so spôsobom predspracovania č. 3. Po predspracovaní dátovej množiny s týmto spôsobom sme získali dátovú množinu s 19 atribútmi. V tomto prípade tento jav sa vyskytuje pri zdrojovej a cielovej IP adrese, pričom korelácie majú o stupeň vyššie hodnoty.

Korelačná matica na obrázku KM\_UNSW-NB15\_3-4\_OHE\_SIM\_2.PNG z prílohy D v adresári \Prilohy\KorelacneMatice zobrazuje koreláciu medzi atribútmi z druhej množina atribútov so spôsobom predspracovania č. 7. Po predspracovaní dátovej množiny s týmto spôsobom sme získali dátovú množinu s 49 atribútmi. Táto matice je príliš komplexná pre detailnejšiu analýzu človekom, a preto úspešnosť tohto spôsobu predspracovania vyhodnocujeme na základe výsledkov nižšie.

V prípade spôsobu predspracovania dátovej množiny s prevzorkovaním, korelačné matice v vo väčšine prípadov vykazujú rovnaký jav. Korelačné matice pre prevzorkované údaje (označenie RESAMP v názve obrázka z prílohy D v adresári \Prilohy\KorelacneMatice) pre obe množiny atribútov vykazujú o niečo vyššie korelácie ako bez vzorkovania.

Ako sme aj vyššie spomenuli, tak sme predspracovali dve podmnožiny 3 a 4 z celkovej dátovej množiny UNSW-NB15, ktorá je rozdelená do štyroch častí. Voľba padla na posledné dve podmnožiny z dôvodu väčšieho zastúpenia útočnej premávky oproti normálnej premávke, oproti zastúpeniu rozdelenia sietovej premávky v podmnožinách 1 a 2. V zlúčenej dátovej množine je

normálna sieťová premávka v zastúpení 893 726 záznamov čo predstavuje 78.39% z celkovej sietovej premávky a útočná sieťová premávka v zastúpení 246 319 záznamov čo predstavuje 21.61% z celkovej sietovej premávky. Tým pádom celkový počet záznamov v zlúčenej dátovej množine UNSW-NB15\_3-4 bez prevzorkovania je 1 140 045. V prípade prevzorkovania, útočná sieťová premávka je vyrovnaná k normálnej sietovej premávke a zastúpenie oboch tried je 893 726 t.j. 50%. Tým pádom celkový počet záznamov v zlúčenej dátovej množine UNSW-NB15\_3-4 s prevzorkovaním je 1 787 452.

Na predspracované dátové množiny (vid'. zoznam nižšie) sme aplikovali vybrané metódy strojového učenia (vid'. kapitolu 4.4 Programový modul strojového učenia). Programový modul strojového učenia vyhodnotil vybrané metódy strojového učenia podľa tabuľiek č. 13 až 17. Hodnoty sú uvedené pre správnosť, F1-skóre a ROC-AUC v percentách pre testovaciu vzorku dát. V prípade neurónových sietí uvádzame aj hodnotu straty (loss) a počet epoch. V prípade neurónových sietí sa zvyčajne snažíme chybu minimalizovať. Preto sa objektívna funkcia často označuje ako stratová funkcia a hodnota vypočítaná stratovou funkciou sa nazýva strata.

Dátové množiny:

1. UNSW-NB15\_3-4\_MAP\_SIM
2. UNSW-NB15\_3-4\_MAP\_RESAMP\_SIM
3. UNSW-NB15\_3-4\_MAP\_SIM\_2
4. UNSW-NB15\_3-4\_MAP\_RESAMP\_SIM\_2
5. UNSW-NB15\_3-4\_OHE\_SIM
6. UNSW-NB15\_3-4\_OHE\_RESAMP\_SIM
7. UNSW-NB15\_3-4\_OHE\_SIM\_2
8. UNSW-NB15\_3-4\_OHE\_RESAMP\_SIM\_2

Vysvetlenie označenia dátových množín:

- Označenie MAP v názve dátovej množiny označuje mapovanie a OHE označuje One-Hot Encoding kategorických atribútov.
- Označenie RESAMP v názve dátovej množiny označuje, že dátová množina je prevzorkovaná.
- Označenie SIM v názve dátovej množiny označuje, že model natrénovaný na predspracovanej dátovej množine sa používa na overenie aj v reálnej simulovanej sietovej premávke.
- Označenie číslom 2 zodpovedá výberu najlepších atribútov z druhej množiny atribútov. Dátové množiny bez číselného označenia obsahujú atribúty z prvej množiny.

Tabuľka 13 – Výsledky testov – správnosť

Dátová množina	LR	SGD	P	XGB	RF	DT	LSTM	DNN	RNN
1	<b>79.42</b>	<b>78.52</b>	<b>77.69</b>	99.09	99.04	98.66	97.61	<b>87.65</b>	<b>88.39</b>
2	80.36	72.46	<b>67.22</b>	99.31	99.05	98.76	98.32	88.8	87.14
3	96.61	97.09	<b>93.20</b>	99.26	99.21	99.04	98.35	98.35	96.71
4	97.91	98.09	72.93	99.48	99.49	99.24	98.68	98.10	98.23
5	<b>79.34</b>	<b>80.25</b>	<b>78.41</b>	99.08	98.97	98.65	97.47	<b>87.10</b>	97.50
6	82.64	82.80	74.33	99.26	99.05	98.13	97.45	91.25	79.79
7	97.78	97.70	<b>80.58</b>	99.23	99.22	99.07	98.68	98.41	98.27
8	97.47	98.44	98.28	99.50	99.25	99.11	98.86	98.63	98.41

Tabuľka 14 – Výsledky testov – F1-skóre

Dátová množina	LR	SGD	P	XGB	RF	DT	LSTM	DNN	RNN
1	<b>62.84</b>	<b>0.19</b>	<b>23.81</b>	97.88	97.76	96.87	94.29	<b>69.69</b>	<b>73.28</b>
2	81.48	78.30	<b>75.04</b>	99.31	99.06	98.77	98.34	89.73	88.28
3	92.17	93.72	<b>83.25</b>	98.30	98.17	97.79	96.22	96.22	92.76
4	97.95	98.13	78.56	99.48	99.48	99.24	98.69	98.14	98.26
5	<b>41.76</b>	<b>16.73</b>	<b>34.82</b>	97.88	97.63	96.87	94.11	<b>63.12</b>	94.32
6	83.98	84.77	72.88	99.27	99.06	98.13	97.46	91.78	79.82
7	94.98	94.95	<b>18.80</b>	98.22	98.18	97.84	96.94	96.35	96.05
8	97.46	98.46	98.31	99.50	99.25	99.11	98.87	98.64	96.33

Tabuľka 15 – Výsledky testov – ROC-AUC

Dátová množina	LR	SGD	P	XGB	RF	DT	LSTM	DNN	RNN
1	80.97	81.61	80.10	98.65	99.96	99.83	95.48	79.80	83.18
2	83.71	83.23	75.83	99.31	99.96	99.90	98.32	88.79	87.13
3	98.87	97.60	94.99	98.85	99.97	99.94	97.75	98.04	96.76
4	98.96	97.85	97.35	99.48	99.99	99.96	98.68	98.10	98.23
5	85.78	83.05	80.96	98.66	99.96	99.88	95.92	74.03	96.92
6	85.64	85.26	84.56	99.26	99.96	99.82	97.45	91.26	79.79
7	99.82	99.89	99.74	98.83	99.97	99.95	98.05	97.89	97.79
8	99.73	99.89	98.74	99.50	99.99	99.94	98.87	98.63	97.69

Hodnoty označené červenou farbou majú veľký výkyv správnosti voči F1-skóre.

Tabuľka 16 – Výsledky testov – strata

Dátová množina	LSTM	DNN	RNN
1	11.19	24.95	23.82
2	4.94	27.72	31.67
3	3.40	3.67	13.69
4	3.27	28.99	10.32
5	13.59	25.63	9.07
6	11.45	24.01	46.49
7	2.56	3.10	3.43
8	2.70	3.46	3.69

Tabuľka 17 – Výsledky testov – počet epoch

Dátová množina	LSTM	DNN	RNN
1	41	83	61
2	67	42	24
3	7	10	87
4	12	6	12
5	51	12	20
6	38	100	100
7	45	44	15
8	58	13	64

Vysvetlenie skratiek v tabuľkách:

- **LR** – Logistická regresia
- **SGD** – SGD klasifikátor
- **P** – Perceptron
- **XGB** – XGradient Boosting klasifikátor
- **RF** – Náhodný les
- **DT** – Rozhodovací strom
- **LSTM** – Long Short-Term Memory rekurentná neurónová siet'
- **DNN** – Hlboká neurónová siet'
- **RNN** – Rekurentná neurónová siet'

Hyperparametre stromovo založených klasifikátorov sme optimalizovali s nasledovnými počtami parametrov: XGB – 6, RF – 7 a DT – 6 kombináciou metód *RandomizedSearchCV* a *GridSearchCV*, ktorým sme sa venovali v kapitole 5.3.2 Trénovanie a vyhodnotenie modelu strojového učenia. Na základe podrobnej analýzy jednotlivých parametrov pre vyššie spomenuté klasifikačné metódy sme sa rozhodli vybrať nasledovné parametre (vid'. tabuľku č. 18).

Tabuľka 18 – Hyperparametre stromovo založených klasifikátorov

<b>XGB</b>	<b>RF</b>	<b>DT</b>
n_estimators	n_estimators	criterion
max_depth	criterion	splitter
tree_method	bootstrap	max_features
sampling_method	max_features	max_depth
scoring	max_depth	min_samples_split
objective	min_samples_split	min_samples_leaf
	min_samples_leaf	

Hyperparametre klasifikačných metód strojového učenia LR, SGD a P sme optimalizovali s počtami parametrov 9, 7 a 8. Hyperparametre sme optimalizovali iba pomocou metódy *RandomizedSearchCV*. Vybrané hyperparametre pre vyššie spomenuté klasifikačné algoritmy zobrazuje nasledujúca tabuľka č. 19.

Tabuľka 19 – Hyperparametre LR, SGD a P klasifikátora

<b>LR</b>	<b>SGD</b>	<b>P</b>
penalty	eta0	eta0
dual	learning_rate	penalty
C	penalty	max_iter
tol	max_iter	n_iter_no_change
solver	fit_intercept	fit_intercept
max_iter	early_stopping	shuffle
multi_class	average	early_stopping
fit_intercept		warm_start
warm_start		

Klasifikátor KNN aj napriek úspešnej implementácii sme nedokázali natrénovať na vybranej dátovej množine, a preto nie je súčasťou tabuľky výsledkov. KNN klasifikátor sme sa snažili optimalizovať so štyrmi parametrami v dvoch iteráciách s päťnásobnou krížovou validáciou, ale proces optimalizácie hyperparametrov trval príliš dlho ( $\sim 9.7$  hodín) tak sme proces ukončili. Dlhý proces učenia sa KNN klasifikátora je zapríčinený tým, že KNN nie je založený na modeloch a má nízku zaujatosť (bias). KNN klasifikátor je pomalou metódou strojového učenia, pretože je závislá na výbere dobrej hodnoty  $k$ , a preto nie je vhodná v mnohých prípadoch, ako sa dokázalo aj v našom prípade.

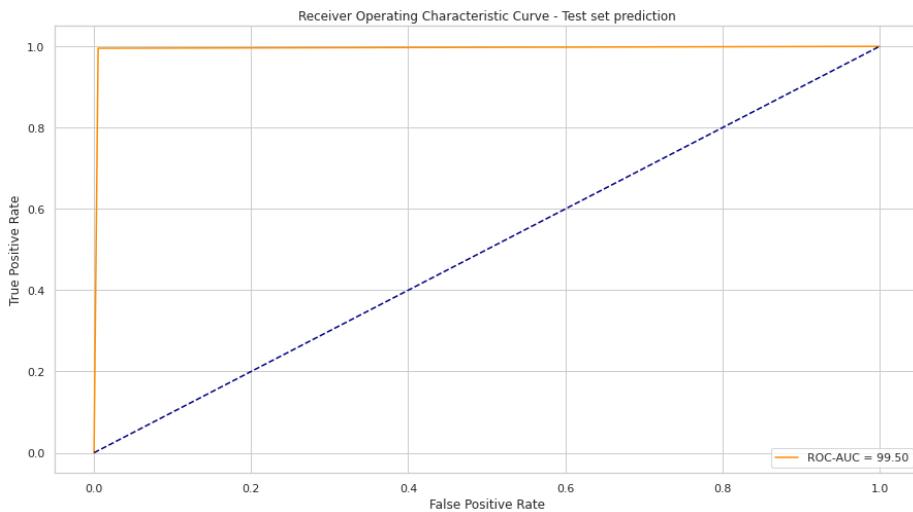
V prípade neurónových sietí sme sa rozhodli zostaviť architektúru pre LSTM a RNN s tromi vrstvami neurónov, tzn. vstupná vrstva, skrytá vrstva a výstupná vrstva (Dense). Pre DNN architektúra je štvorvrstvová, tzn. vstupná vrstva, dve skryté vrstvy a výstupná vrstva. Všeobecne je platné, že dve vrstvy postačujú na riešenie komplexnejších problémov. Viac vrstiev môžu byť lepšie, ale taktiež môžu byť ľažšie na trénovanie. Všeobecne platí pravidlo, že skrytá vrstva pracuje s jednoduchými problémami a ostatné dve vrstvy (vstupná, výstupná) sú na to, aby našli primerane komplexné vlastnosti. V súvislosti s neurónmi je potrebné spomenúť aktivačnú funkciu. Aktivačné funkcie sú matematické rovnice, ktoré určujú výstup neurónovej siete. Funkcia je pripojená ku každému neurónu v sieti a určuje, kedy má byť neurón aktivovaný na základe toho, či je vstup neurónu relevantný pre predikciu modelu. Počet neurónov pre jednotlivé vrstvy a aktivačnú funkciu sme optimalizovali spolu s hyperparametrami, viď<sup>2</sup>. kapitolu 5.3.2 Trénovanie a vyhodnotenie modelu strojového učenia. Neurónovým sietiam sme sa venovali v kapitole č. 2.6.2..

Analýzou hodnôt úspešnosti, F1-skóre, ROC-AUC a stratových hodnôt z predchádzajúcich tabuliek sme dospeli k nasledovným zisteniam.

Na základe finálnych testov môžeme dôjsť k záveru, že najspoločnejšimi metódami strojového učenia na báze stromov sú XGradient Boosting a náhodný les, pretože v každom teste dosahovali veľmi podobné, stále a vysoké hodnoty správnosti a F1-skóre. Rozhodovací strom oproti predošlým metódam dosahoval o niečo menej presné hodnoty správnosti klasifikácie a F1-skóre, ale aj napriek tomu je spoločne. Metódy strojového učenia na báze stromov nevykazujú žiadne kolísania hodnôt správnosti voči F1-skóre a aj ROC-AUC hodnoty sú veľmi vysoké.

Z tejto skupiny klasifikátorov je najlepší XGradient Boosting. Tento model strojového učenia dosiahol najvyššiu správnosť na dátovej množine č. 8. To znamená že na type predspracovania kategorických atribútov pomocou One-Hot Encoding s prevzorkovaním a s najlepšími atribútmi z druhej množiny atribútov. Všetky tri metriky merania úspešnosti klasifikácie modelu strojového učenia majú konštantnú hodnotu 99.50%. Podobne vysokú úspešnosť 99.48%, pre všetky merané metriky získal XGradient Boosting aj v prípade klasifikácie pre dátovú množinu č. 4. Rozdiel je iba

v spôsobe predspracovania kategorických atribútov mapovaním. Tým pádom vieme odvodiť, že rozdiel v predspracovaní kategorických atribútov mapovaním alebo metódou One-Hot Encoding nespôsobuje veľký rozdiel v úspešnosti klasifikácie. Nasledujúci obrázok č. 29 zobrazuje ROC krivku pre najlepší model XGradient Boosting.



Obrázok 29 – ROC krivka najlepšieho modelu XGradient Boosting

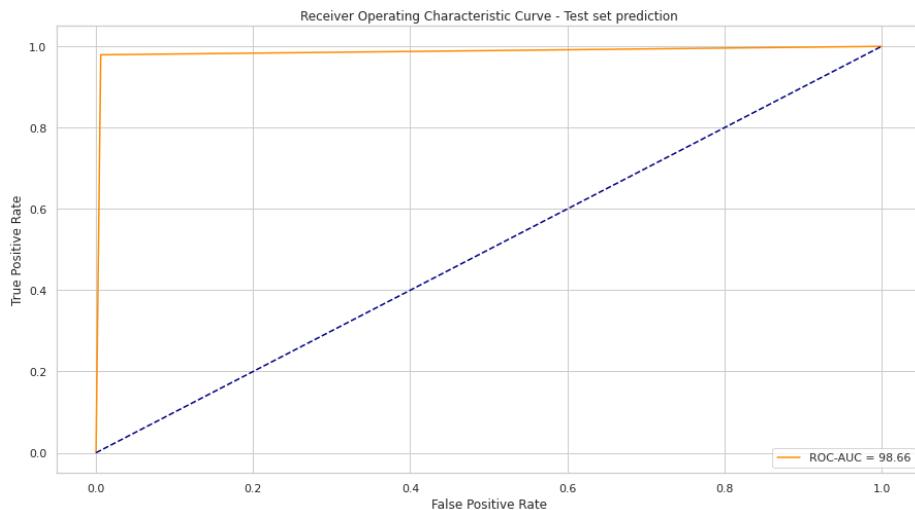
Kontingenčná tabuľka pre najlepší XGradient Boosting klasifikátor je nasledovná.

Tabuľka 20 – Kontingenčná tabuľka pre najlepší XGradient Boosting

XGradient Boosting		Predpovedaná trieda	
		Negatívny	Pozitívny
Aktuálna trieda	Negatívny	108251	571
	Pozitívny	507	107702

Z kontingenčnej tabuľky môžeme vyčítať, že z celkového počtu 217 031 záznamov pre testovaciu množinu dát klasifikátor XGradient Boosting nedokázal identifikovať 507 útokov a 517 záznamov normálnej siet'ovej premávky klasifikoval ako útok.

Pre porovnanie, obrázok č. 30 znázorňuje ROC krivku pre najhorší model XGradient Boosting pre dátovú množinu č. 5 so správnosťou 99.08%, F1-skóre 97.88% a ROC-AUC hodnotou 98.66%.



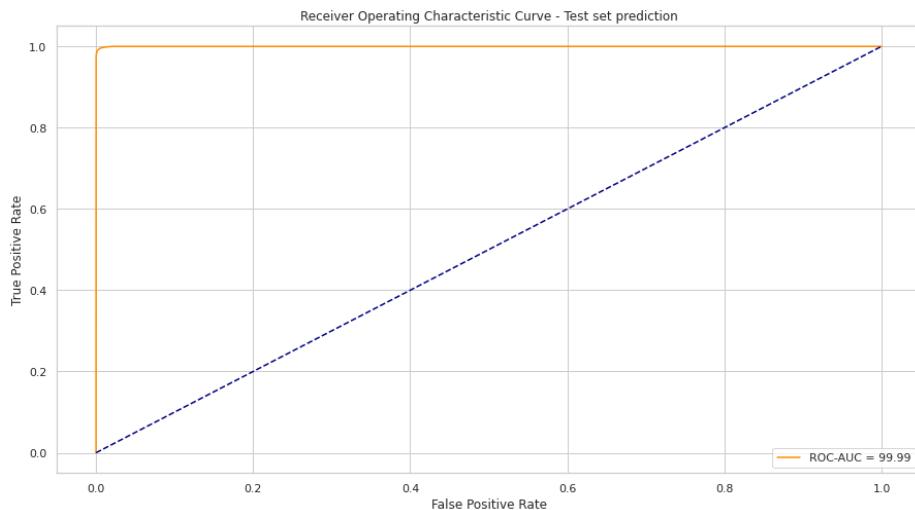
Obrázok 30 – ROC krivka najhoršieho modelu XGradient Boosting

Najlepší model XGradient Boosting má nasledovné nastavenie:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=16,
              min_child_weight=1, missing=None, n_estimators=75,
              n_jobs=-1, nthread=None, objective='binary:hinge',
              random_state=0, reg_alpha=0, reg_lambda=1,
              sampling_method='uniform', scale_pos_weight=1, scoring='auc',
              seed=None, silent=None, subsample=1, tree_method='auto',
              verbosity=1)
```

Pripomenuj by sme najlepší model náhodného lesa z experimentálneho overenia riešenia, ktorý dosahoval hodnotu správnosti 99.67%, F1-skóre 100% a ROC-AUC hodnotu 99.99%. V novom overení riešenia náhodný les pre rovnaký spôsob predspracovania dátovej množiny, ale s menším počtom atribútov dosiahol správnosť 99.05%, F1-skóre 99.06% a ROC-AUC hodnotu 99.96%. Náhodný les s najlepšou mierou správnosti klasifikácie dosiahol 99.49%, F1-skóre 99.48 a ROC-AUC hodnotu 99.99% na dátovej množine č. 4. Na rovnakej dátovej množine dosiahol najlepší výsledok aj rozhodovací strom so správnosťou 99.24%, F1-skóre 99.24% a ROC-AUC hodnotou 99.96%.

ROC krivka najlepšieho náhodného lesa je znázornený na obrázku č. 31.



Obrázok 31 – ROC krivka najlepšieho modelu náhodného lesa

Kontingenčná tabuľka pre najlepší náhodný les je nasledovná.

Tabuľka 21 – Kontingenčná tabuľka pre najlepší náhodný les

Náhodný les		Predpovedaná trieda	
		Negatívny	Pozitívny
Aktuálna trieda	Negatívny	177862	1013
	Pozitívny	828	177788

Z kontingenčnej tabuľky môžeme vyčítať, že z celkového počtu 357 491 záznamov pre testovaciu množinu dát klasifikátor náhodný les nedokázal identifikovať 828 útokov a 1013 záznamov normálnej sieťovej premávky klasifikoval ako útok.

Najlepší model náhodného lesa má nasledovné nastavanie:

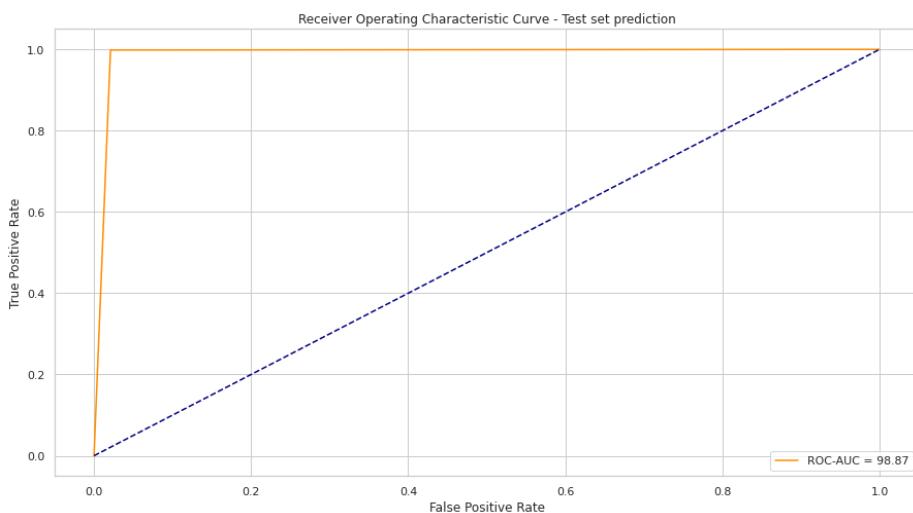
```
RandomForestClassifier(bootstrap=False, ccp_alpha=0.0, class_weight=None,
                      criterion='entropy', max_depth=75, max_features='sqrt',
                      max_leaf_nodes=None, max_samples=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=10, min_samples_split=12,
                      min_weight_fraction_leaf=0.0, n_estimators=125,
                      n_jobs=-1, oob_score=False, random_state=None,
                      verbose=0, warm_start=False)
```

Treťou skupinou najsprávnejších klasifikačných metód strojového učenia sú neurónové siete. Zo skupiny neurónových sietí najlepšie správnosti dosahovala LSTM neurónová sieť. Pre tento model strojového učenia taktiež platí, že dosahoval veľmi podobné, stále a vysoké hodnoty správnosti a F1-skóre. Neurónové siete DNN a RNN dosahovali už o niečo horšie výsledky. Z tabuľiek č. 13 a 14 vieme taktiež vyčítať kolísanie hodnôt správnosti voči F1-skóre (vyznačené hodnoty červenou farbou). Obe neurónové siete mali horšiu mieru úspešnosti klasifikácie pre dátovú množinu č. 1, t.j. spracovanie kategorických atribútov mapovaním bez prevzorkovania a s výberom najlepších atribútov z množiny č. 1.

Do skupiny neurónových sietí taktiež patrí aj perceptron, t.j. lineárny binárny klasifikátor - jednovrstvová neurónová sieť. Zistili sme, že jednovrstvová neurónová sieť nie je vhodná na riešenie danej problematiky, pretože vo väčšine prípadov metriky úspešnosti klasifikácie sú kolísavé a tým pádom je klasifikácia nepresná.

Všetky typy neurónových sietí dosahovali najlepšie výsledky na dátovej množine č. 8 podobne ako klasifikátor XGradient Boosting. Najlepšia LSTM neurónová sieť pre túto dátovú množinu dokázal odhaliť sietový útok so správnosťou 98.86%, F1-skóre 98.87% a ROC-AUC hodnotou taktiež 98.87%. Na druhej strane strata 2.7% nie je najnižšia a ani dĺžka učenia sa počas 58 epochách z celkových 100 nie je najkratšia. Iné je to pre dátovú množinu č. 3, kedy LSTM neurónová sieť dosiahla správnosť 98.35%, F1-skóre 96.22%, ROC-AUC hodnotu 97.75%, stratu 3.4% a to všetko sa naučila iba v 7 epochách.

Nasledujúci obrázok č. 32 zobrazuje ROC krivku pre najlepší model LSTM neurónovej siete.



Obrázok 32 – ROC krivka najlepšieho modelu LSTM

Kontingenčná tabuľka pre najlepšiu LSTM neurónovú siet' je nasledovná.

Tabuľka 22 – Kontingenčná tabuľka pre najlepšiu LSTM neurónovú siet'

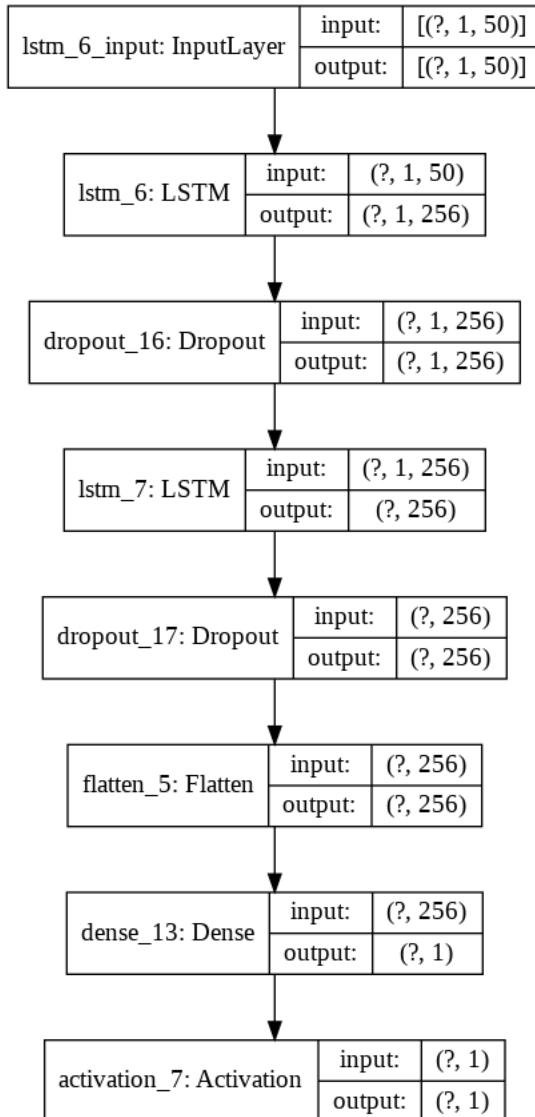
LSTM neurónová siet'		Predpovedaná trieda	
		Negatívny	Pozitívny
Aktuálna trieda	Negatívny	106536	2286
	Pozitívny	178	108031

Z kontingenčnej tabuľky môžeme vyčítať, že z celkového počtu 217 031 záznamov pre testovaciu množinu dát LSTM neurónová siet' nedokázala identifikovať 178 útokov a 2 286 záznamov normálnej sieťovej premávky klasifikovala ako útok.

Nastavenie pre najlepšiu LSTM neurónovú siet' je nasledovné:

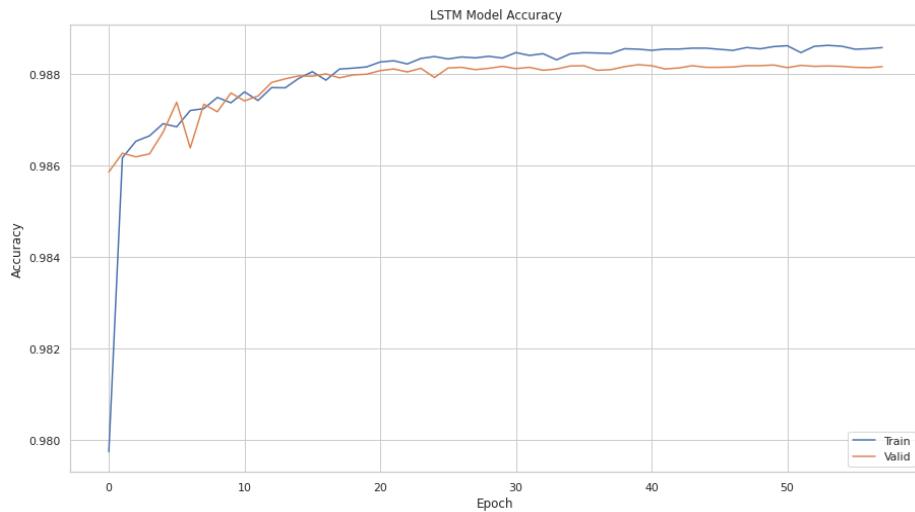
```
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(256, input_shape=(X_train.shape[1],
X_train.shape[2]), kernel_initializer=random_normal, activation=relu,
recurrent_dropout=0.2, return_sequences=True))
model.add(tf.keras.layers.Dropout(dropout))
model.add(tf.keras.layers.LSTM(256, kernel_initializer=random_normal,
activation=relu, recurrent_dropout=0.2, return_sequences=False))
model.add(tf.keras.layers.Dropout(0.4))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(1))
model.add(tf.keras.layers.Activation(tanh))
model.compile(loss='binary_crossentropy', optimizer=nadam, metrics=['acc'])
```

Z nastavenia dvojvrstvovej LSTM neurónovej siete vyplýva, že každá vrstva neurónovej siete má 256 neurónov. Architektúra najlepšej LSTM neurónovej siete je nasledovná.

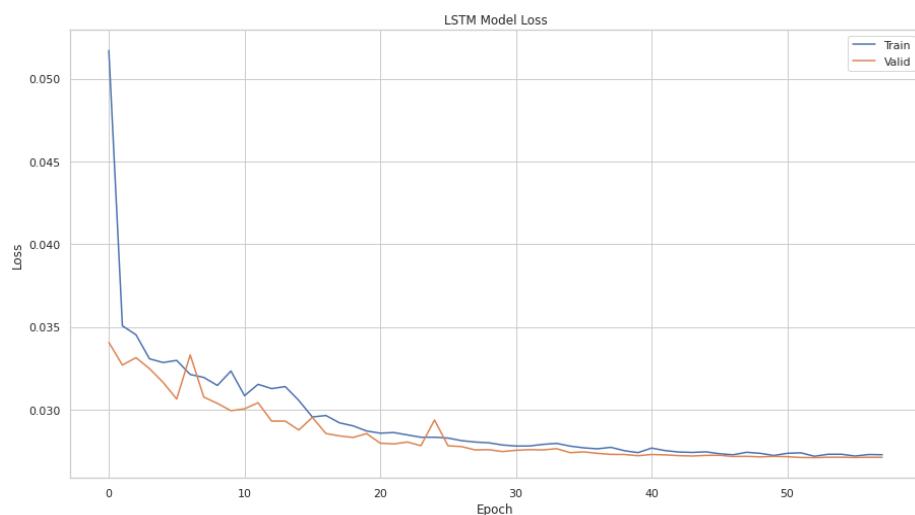


Obrázok 33 – Architektúra LSTM

Nasledujúce obrázky zobrazujú grafy priebehov učenia sa najlepšej neurónovej siete LSTM pre správnosť a stratu.



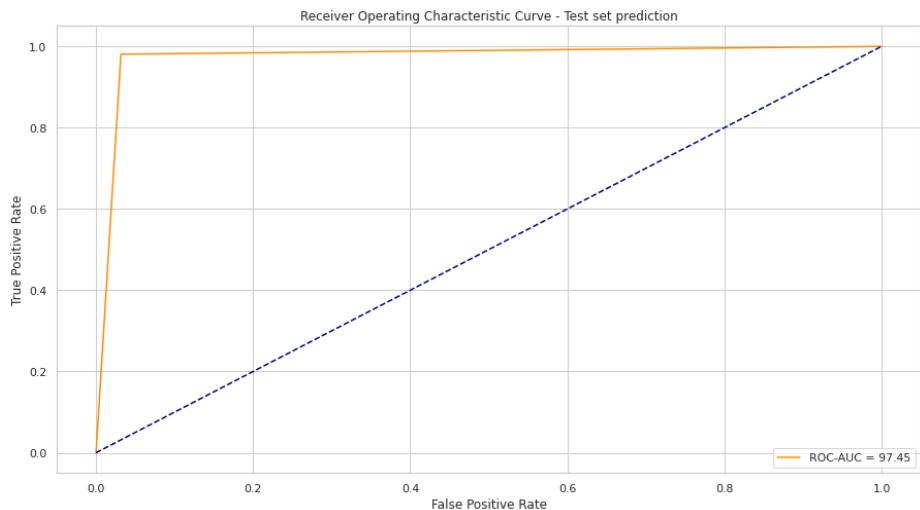
Obrázok 34 – Krivka učenia sa najlepšieho modelu LSTM pre správnosť



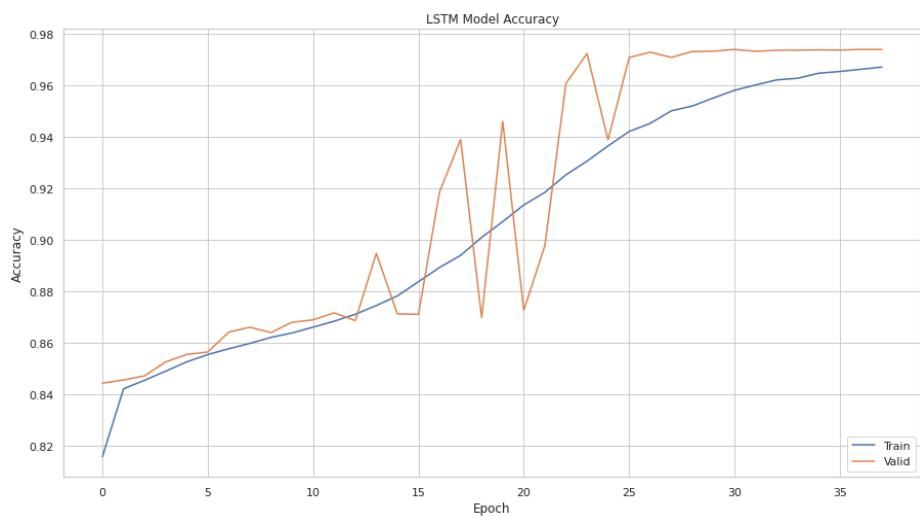
Obrázok 35 – Krivka učenia sa najlepšieho modelu LSTM pre stratu

Graf kriviek učenia sa je dobrý vtedy, ak krivka straty počas trénovania (modrá krivka) klesá do bodu stability a krivka validácie (oranžová krivka) taktiež klesá do bodu stability a je malý rozdiel medzi stratou trénovania a validácie. Dlhodobé trénovanie modelu vedie k pretrénovaniu, a preto je potrebné trénovanie v čas zastaviť, tejto problematike sme sa venovali v časti optimalizácia neurónovej siete v kapitole č. 5.3.2. Príkladom menej reprezentatívneho grafu učenia sa sú obrázky č 37 a 38.

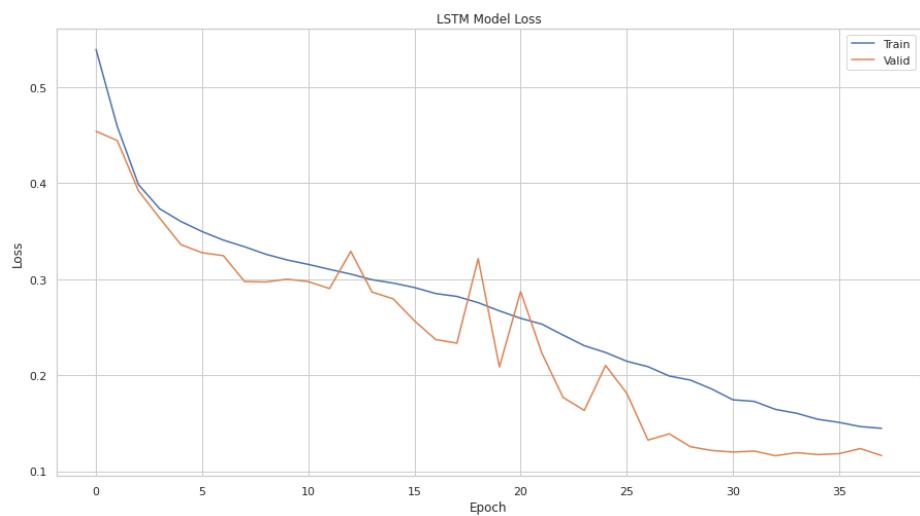
Pre porovnanie ROC kriviek a kriviek učenia sa, obrázky č. 36 až 38 znázorňujú najhorší model LSTM neurónovej siete so správnosťou 97.45%, F1-skóre 97.46%, ROC-AUC hodnotou 97.45%, stratou 11.45% a dĺžkou učenia 38 epochov.



Obrázok 36 – ROC krivka najhoršieho modelu LSTM

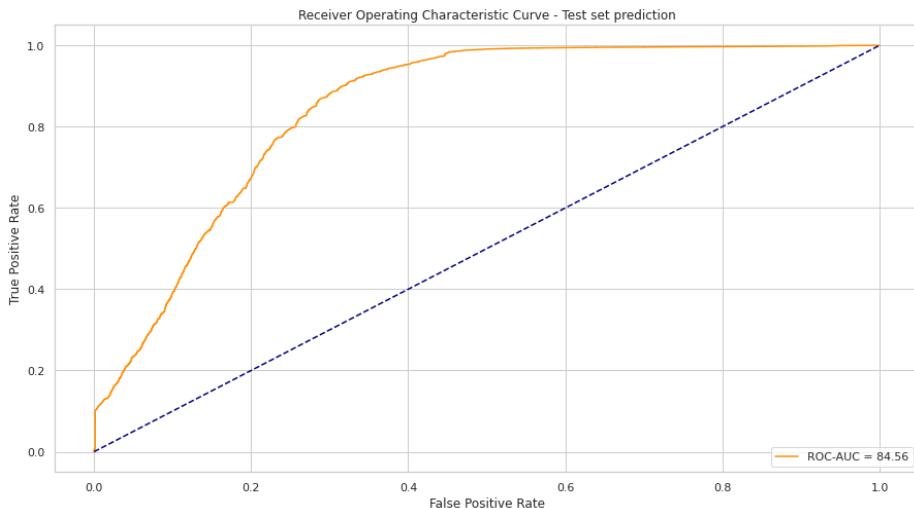


Obrázok 37 – Krivka učenia sa najhoršieho modelu LSTM pre správnosť



Obrázok 38 – Krivka učenia sa najhoršieho modelu LSTM pre stratu

Pre účel porovnania LSTM s perceptronom pre rovnakú dátovú množinu č. 6, obrázok nižšie zobrazuje najhorší model perceptronu so správnosťou 74.33%, F1-skóre 72.88% a ROC-AUC hodnotou 84.56%.



Obrázok 39 – ROC krivka perceptronu

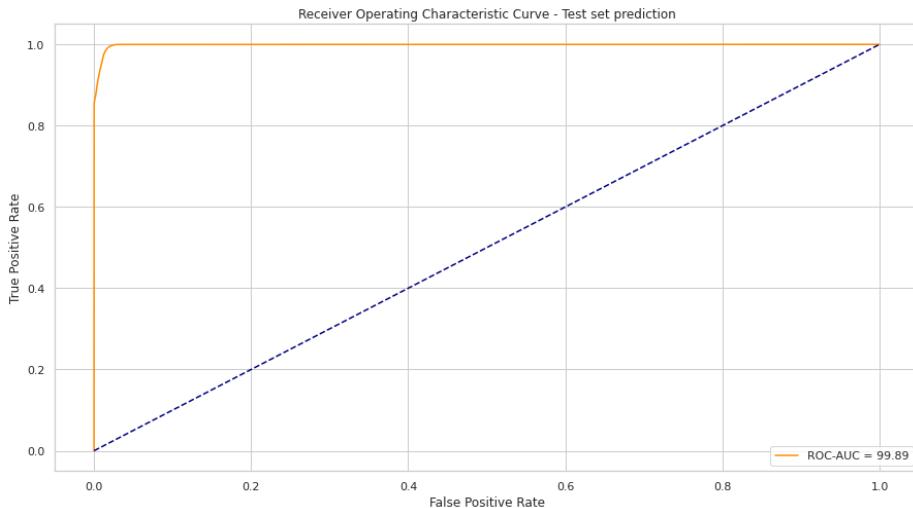
Kontingenčná tabuľka pre vyššie spomínaný perceptron je nasledovná.

Tabuľka 23 – Kontingenčná tabuľka pre perceptron

Perceptron		Predpovedaná trieda	
		Negatívny	Pozitívny
Aktuálna trieda	Negatívny	142408	36414
	Pozitívny	55351	123318

Z kontingenčnej tabuľky môžeme vyčítať, že z celkového počtu 357 491 záznamov pre testovaciu množinu dát perceptron nedokázal identifikovať až 55 351 útokov a 36 414 záznamov normálnej sietovej premávky klasifikoval ako útok.

Ďalej sa nám podarilo lepšie zo optimalizovať logistickú regresiu a SGD klasifikátor. Výsledky týchto metód strojového učenia vo väčšine prípadov už nemajú veľké výkyvy v hodnotách správnosti modelu voči F1-skóre a tým pádom môžeme dôjsť k záveru, že sa nám podarilo zo optimalizovať modely na akceptovateľnú úroveň. Dôležité je poznamenať, že pre výber najlepších hyperparametrov pri overení riešenia sme použili pre modely logistická regresia a SGD klasifikátor metódu *RandomizedSearchCV*. Podarilo sa nám vytvoriť konkurencie schopný SGD klasifikátor, ktorého najlepšia správnosť dosahuje hodnotu 98.44%, F1-skóre 98.46% a ROC-AUC hodnotu 99.89%. Obrázok nižšie zobrazuje ROC krivku najlepšieho SGD klasifikátora.



Obrázok 40 – ROC krivka najlepšieho SGD klasifikátora

Kontingenčná tabuľka pre spomínaný SGD klasifikátor je nasledovná.

Tabuľka 24 – Kontingenčná tabuľka pre najlepší SGD klasifikátor

SGD klasifikátor		Predpovedaná trieda	
		Negatívny	Pozitívny
Aktuálna trieda	Negatívny	105443	3379
	Pozitívny	2	108207

Z kontingenčnej tabuľky môžeme vyčítať, že z celkového počtu 217 031 záznamov pre testovaciu množinu dát SGD klasifikátor nedokázal identifikovať iba 2 útoky a 3 379 záznamov normálnej sietovej premávky klasifikoval ako útok.

Ďalej sme vykonali klasifikáciu nad vybranými dátovými množinami pomocou TPOT automatizovaného strojového učenia, ktorý optimalizuje rúry strojového učenia pomocou genetického programovania (viď. kapitolu 5.3.3). Výsledky porovnania s nástrojom TPOT slúžia pre reprezentatívne účely, kedy sa porovnávame voči existujúcemu riešeniu. Na základe tabuľky č. 25 vieme odvodiť záver, že naše riešenie má väčšiu mieru správnosti aj keď použitá metóda nie je rovnaká ako metóda strojového učenia TPOT. Ďalej musíme podotknúť, že TPOT našiel najoptimálnejšie modely strojového učenia rúrovaním ďalších metód a v našom prípade sme optimalizovali iba jeden model bez ďalších podporných metód strojového učenia. Ďalšou veľkou výhodou nášho riešenia programového modulu strojového učenia je možnosť konfigurácie, škálovania a monitorovania, pričom TPOT je riešenie tzv. black box.

Tabuľka 25 – Výsledky TPOT

Dátová množina	TPOT metóda strojového učenia	TPOT správnosť	Naša metóda strojového učenia	Naša správnosť
1	ExtraTreesClassifier s DT	98.39	XGB	99.09
3	ExtraTreesClassifier	99.25	XGB	99.26
5	XGB	95.32	XGB	99.08
7	ExtraTreesClassifier	99.20	XGB	99.23

Autor Zelaya [17] vykonal experiment, kde zistoval aký má vplyv vzorkovanie na výsledok vyhodnotenia klasifikátora náhodný les. Autor vykonal podvzorkovanie (undersampling) aj prevzorkovanie (oversampling) pomocou metódy *SMOTE* nad vybranou množinou dát. Vzorkovaniu sme sa venovali v kapitole 5.2.9 a v našej práci podobne riešime danú problematiku. Autor na porovnanie výkonnosti vybraného klasifikátora použili jednoduchú správnosť definovanú ako podiel správne predpovedaných údajových bodov k celému súboru údajov. Pre oversampling pomocou *SMOTE* jeho klasifikátor dosahoval rovnaké výsledky so zanedbateľným rozdielom oproti výsledkom klasifikácie bez vzorkovania. V našom prípade rozdiel vo výsledkoch správností sú významnejšie. V priemere metódy strojového učenia dosahovali o 2.56% lepšiu správnosť na dátových množinách bez prevzorkovania.

Autor Crone [10] taktiež skúma vplyv vzorkovania na výkon modelu strojového učenia. Autor na základe analýzy dospel k záveru, že podvzorkovanie je sub-optimálne oproti prevzorkovaniu vo všetkých použitých metódach strojového učenia. Podľa autorovej prípadovej štúdie podvzorkovanie vedie k výrazne zvýšenému, ale irrelevantnému výkonu pre podvzorkované dátá za cenu zníženého výkonu oproti vzorke dát bez vzorkovania bez ohľadu na klasifikačnú metódu. Okrem toho uvádzia, že podvzorkovanie vyvoláva nekonzistentný výber najlepších parametrov pre metódu strojového učenia. Ich korelačná analýza ďalej potvrdzuje vysoké korelácie medzi výcvikom, validáciou a výkonnosťou testu pre prevzorkovanie.

Na základe zistenia, že metódy strojového učenia vykazujú v priemere vyššiu správnosť na dátových množinách s predspracovaním bez prevzorkovania sme poskytli natrénované modely strojových učení diplomantovi Bc. Danielovi Gábrišovi, ktorý overil správnosť našich modelov na simulovanej sieťovej premávke. Podľa Gábriša modely strojového učenia boli otestované na reálnej sieťovej premávke zachytenej počas vytvárania dátovej množiny UNSW-NB15. Reálna sieťová premávka pozostáva z paketov zo súboru PCAP17. Na dátach z tohto súboru boli aj naše modely strojového učenia trénované. Gábriš odosnal pakety zo súboru cez softvérovo definovanú sieť - SDN, kde sa zachytávali štatistické údaje o prenesenej sieťovej premávke. Zachytené štatistické údaje z prepínačov v SDN sieti boli extrahované riadiacim uzlom a odoslané do aplikácie na detekciu

siet'ových útokov. Táto aplikácia vykonala predspracovanie štatistických údajov a vytvorila tak vstupné dátá do vybraného modelu strojového učenia. Ten vykonal vyhodnotenie a predpovedané hodnoty zapísal do súboru. Záznamy zo súboru sa následne porovnali s dátami z CSV súborov, čím sa zistila reálna hodnota, či sa jedná o útok alebo nie. Na základe toho sa získali vypočítané hodnoty pre správnosť a F1-skóre. Tabuľky č. 26 až 29 zobrazujú výsledky tohto overenia.

Tabuľka 26 – Overenie riešenia na simulovanej siet'ovej premávke – správnosť

Počet paketov	LR	SGD	P	XGB	RF	DT	LSTM	DNN	RNN
200 000	96.57	96.08	96.36	92.78	99.22	83.35	98.88	98.33	98.68
500 000	96.81	96.62	96.59	92.62	99.64	83.65	99.11	98.22	98.71

Tabuľka 27 – Overenie riešenia na simulovanej siet'ovej premávke – F1-skóre

Počet paketov	LR	SGD	P	XGB	RF	DT	LSTM	DNN	RNN
200 000	16.67	0.00	0.00	49.48	90.00	27.60	82.71	79.76	83.83
500 000	9.63	3.26	0.00	46.58	94.71	26.51	85.27	77.93	82.65

Tabuľka 28 – Kontingenčná tabuľka pre náhodný les pre 500 000 paketov

Náhodný les		Predpovedaná trieda	
		Negatívny	Pozitívny
Aktuálna trieda	Negatívny	5077	15
	Pozitívny	4	107

Tabuľka 29 – Kontingenčná tabuľka pre LSTM neurónovú siet' pre 500 000 paketov

LSTM neurónová siet'		Predpovedaná trieda	
		Negatívny	Pozitívny
Aktuálna trieda	Negatívny	5073	5
	Pozitívny	42	136

Výsledky z tabuľiek vyššie potvrdzujú naše overenie riešenia. Výsledky správností a F1-skóre jasne vykazujú lepšiu mieru správnej klasifikácie siet'ovej premávky zvyšujúcim sa počtom paketov. To znamená, že metóda strojového učenia s vyššou pravdepodobnosťou odhalí siet'ový útok ak má dostatočné množstvo vstupných dát. Objem 500 000 paketov zodpovedá v priemere približne 5 277 záznamov z čoho v priemere približne 178 záznamov tvorí útok.

Z tabuľiek vyššie sa potvrdili metódy strojového učenia ako náhodný les a LSTM neurónová siet' ako najlepšie. Gábriš preto vykonal ďalšie testy na celej dátovej množine UNSW-NB15 pre počet paketov 500 000 a naše modely dosiahli výborné výsledky:

- **Náhodný les**

- správnosť: 99.71%
- F1-skóre: 95.60%

- **LSTM neurónová siet'**

- správnosť: 99.39%
- F1-skóre: 90.42%

Rekurentná neurónová siet' dosahuje taktiež veľmi dobré výsledky. Štvrtou najlepšou klasifikačnou metódou je hlboká neurónová siet'. Na druhej strane klasifikátor XGradient Boosting vykazuje prekvapivo nízke F1-skóre voči správnosti a tým pádom nie je vhodné na reálne použitie. Rovnako zlé výsledky dosiahol aj rozhodovací strom. Výsledky pre klasifikátory logistická regresia, SGD a perceptron taktiež dokazujú, že nie sú v hodnej na riešenie problému takéhoto charakteru, pretože neboli schopné identifikovať takmer žiadny útok.

Autor Kanimozhi [26] vykonal experiment na dátovej množine UNSW-NB15 pomocou klasifikátora náhodný les s výberom najlepších štyroch atribútov (*sbytes*, *sttl*, *sload*, *ct\_dst\_src\_ltm*) pomocou metódy *RFE* a dosiahol správnosť 98.30% a F1-skóre 97.50%. Spôsob, akým Kanimozhi predspracoval dátovú množinu nie je známy. Autor ďalej vykonal na rovnakej množine atribútov druhý experiment s doprednou neurónovou sietou a dosiahol správnosť 89% a F1-skóre 91%.

Roy [40] vykonal experiment s použitím BLSTM neurónovej siete, ktorá je obojsmerná (bidirectional) rekurentná neurónová siet'. Podľa Roya BLSTM v porovnaní oproti klasickej LSTM sa učí na poskytnutých dátach raz od začiatku do konca a raz od konca do začiatku. Autor použil päť atribútov (*service*, *sbytes*, *sttl*, *smean*, *ct\_dst\_sport\_ltm*) trénovacej dátovej množiny UNSW-NB15 na odhalenie anomálnej sietovej premávky. Kategorický atribút *service* pretransformoval na numerický a následne použil normalizáciu hodnôt dátovej množiny. Roy taktiež vykonal optimalizáciu hyperparametrov pre BLSTM neurónovú siet'. Autorove výsledky sú nasledovné: správnosť 95.71% a F1-skóre 98%. Na základe autorovej kontingenčnej tabuľky sme zistili, že z celkového počtu 4 206 záznamov pre testovaciu množinu dát, BLSTM neurónová siet' nedokázala identifikovať 166 útokov a 10 záznamov normálnej sietovej premávky klasifikoval ako útok.

Na základe vyššie spomenutých výsledkov autorov sa dokážeme porovnať predovšetkým vo výbere a počte atribútov. Obe prípadové štúdie dosiahli vysokú úspešnosť na veľmi nízkom počte

atribútov. V porovnaní s našim riešením sme použili dvanásť atribútov, čo predstavuje trojnásobok počtu atribútov, ktoré použil Kanimozhi. Tieto merania zodpovedajú dátovým množinám č. 1, 2, 5 a 6. Každý model nášho náhodného lesa vykazuje lepšie výsledky ako náhodný les Kanimozhiho. To isté platí aj pre neurónové siete. Naše modely neurónových sietí vo väčšine prípadov dosiahli vyššie hodnoty správností a F1-skóre. Rozdiely medzi týmito metrikami boli tiež menšie ako pre neurónové siete vyššie spomínaných autorov. Pre porovnanie naša LSTM neurónová sieť pre dátovú množinu č. 2 dosiahla správnosť 98.32% a F1-skóre 98.34% a BLSTM neurónová sieť autora Roya dosiahla správnosť 95.71% a F1-skóre 98%.



## 7. Záverečné myšlienky

Podľa Zelaya [17] väčšina výskumov sa zameriava na problém vysvetlenia modelu strojového učenia. Tiež upozorňuje na to, že mnohé z rozhodnutí, ktoré ovplyvňujú prediktívne správanie modelu sa vykonávajú už počas fázy predbežného spracovania údajov. Tieto rozhodnutia sú zakódované ako konkrétné kroky pre transformáciu údajov ako súčasť fázy trénovania. Autor článku vykonal výskum dopadu toho, ako môžu metódy predbežného spracovania súborov údajov ovplyvniť výsledok predikcie metódy strojového učenia. Analyzujú sa najmä nasledovné otvorené problematiky. Porozumenie citlivosti výsledkov určitých dátových bodov na rôzne techniky predbežného spracovania. Zisťovanie či proces predbežného spracovania zvyšuje alebo znížuje zaujatosť pre konkrétné zraniteľné skupiny dát, a či zvyšuje transparentnosť celého strojového učenia. Či umožňuje vytvárať lepšie modely analýzou ktoréhokoľvek zo špecifických krov predbežného spracovania údajov. Pokial' ide o túto problematickú oblasť, Crone [10] meria vplyv rôznych metód predbežného spracovania na metriky výkonnosti určitých klasifikátorov. Taktiež poukazuje na obmedzenú dokumentáciu a takmer žiadne konkurenčné hodnotenie problému predspracovania dát.

Crone poukazuje na rovnaké techniky predspracovania dátového súboru, ktoré sme aj my v našej práci implementovali. Spomína prístup náhodného prevzorkovania menšinovej alebo väčšinovej triedy pre vysporiadanie sa s odchýlkami bez zmeny klasifikátora. Okrem toho spomína sofistikované techniky, ako napríklad odstránenie šumu v dátach, odstránenie hraničných alebo nadbytočných záznamov väčšinovej triedy, alebo vytvorenie nových záznamov menšinovej triedy ako kombinácia dvoch susedných tried. V našej práci sme taktiež zisťovali či proces predbežného spracovania zvyšuje alebo znížuje zaujatosť pre konkrétné zraniteľné skupiny dát, a či vôbec zvyšuje transparentnosť celého strojového učenia. Túto problematiku vyzdvihol aj Zelaya vyššie. Toto zahrňa výber najlepších atribútov na základe analýzy.

Podľa Crone redukcia dát sa vykonáva výberom jednotlivých atribútov. Cieľom výberu atribútov je identifikovať najrelevantnejšie vysvetľujúce vstupné premenné v rámci súboru údajov. Okrem zlepšenia výkonnosti klasifikátorov, výber atribútu uľahčuje lepšie pochopenie základného procesu, ktorý generoval dátu. Zmenšením súboru vektorov sa zníži aj veľkosť súboru údajov, urýchli sa trénovanie klasifikátora a tým sa zvýsi aj efektívnosť výpočtovej techniky.

Podľa štúdie Crone iba necelá polovica všetkých štúdií využíva a dokumentuje prístupy k znižovaniu údajov, zatiaľ čo iba 64% analyzovaných štúdií zvažuje projekciu údajov. Veľmi málo publikácií poskytuje informácie o zaobchádzaní s kategorickými údajmi, aj keď kategorické údaje sa používajú a dokumentujú vo väčšine z štúdií. Naopak, informácie o príslušných postupoch ladenia parametrov metód strojového učenia sú uvedené vo väčšine z publikácií.



## 8. Zhodnotenie a záver

V tejto práci sme sa venovali návrhu vlastného riešenia. Navrhli sme dva programové moduly. Prvý programový modul je určený na spracovanie rozsiahlych dátových množín a druhý programový modul je určený na aplikovanie metód strojového učenia na vybranej predspracovanej dátovej množine. Taktiež sme navrhli vhodné prostredie, v ktorom sme navrhnuté programové moduly neskôr implementovali. Na výber vhodného prostredia sme kládli veľký dôraz z dôvodu práce s rozsiahlymi množinami dát a z dôvodu práce s metódami strojového učenia, ktoré vyžadujú vysoký výpočtový výkon. Určili sme funkcionálne a nefunkcionálne požiadavky, ktoré majú programové moduly splňať. Na základe návrhu riešenia a funkcionálnych požiadaviek sme úspešne implementovali prototypy programových modulov. Postup implementácie jednotlivých častí programových modulov sme taktiež podrobne opísali. Podarilo sa nám stiahnuť rozsiahlu množinu dát UNSW-NB15, nad ktorou sme aplikovali rôzne metódy predspracovania dátovej množiny. Podarilo sa nám predspracovať dátové množiny rôznymi spôsobmi.

V našej práci sme kládli veľký dôraz na spôsob predspracovania vybranej dátovej množiny. Počas fázy predprípravy dátovej množiny sme museli myslieť na ďalšiu fazu strojového učenia, a tak v tomto duchu sme predspracovávali dátové množiny. Veľký dôraz sme kládli na sofistikované prístupy a výber vhodných metód pre tieto účely. Vedeli sme, že mnohé z rozhodnutí, ktoré zavedieme počas predprípravy dátovej množiny budú mať vplyv na prediktívne správanie modelu strojového učenia. Potrebovali sme porozumieť samotnej dátovej množine, aby sme mohli vykonať zmenu nad dátami. Tieto zmeny potom mali neskôr vplyv na ďalšie dáta a rôzne techniky predbežného spracovania. Zmeny nad dátovou množinou, ktoré sme aplikovali mali za následok rozšírenie dátovej množiny o ďalšie užitočné atribúty, tento proces sa nazýva obohacovanie dát. Obohacovaním dát sme získali možnosť lepšie odhalovať útoky.

Najlepšie atribúty z dátovej množiny boli vybraté na základe vysokých korelácií. Zúžením množiny atribútov dátovej množiny na menší počet sme dostali menšiu dátovú množinu, ktorej výhoda sa neskôr odzrkadlila na výkone a čase počas klasifikácie. Taktiež sme zistili, že isté formy predspracovanej dátovej množiny vyčerpávajú výpočtovú a pamäťovú kapacitu vývojového prostredia, v ktorom sa trénovanie strojového učenia vykonávalo. V tom prípade sme museli zastaviť proces učenia a znova optimalizovať parametre metód strojového učenia, aby sa mohlo strojové učenie dokončiť. Niektoré prípady zas nám ukázali, že môžeme natrafiť aj fenomén ako multikolinearita.

Postupnou analýzou rôznych metód predspracovania dátovej množiny a metód strojového učenia sme dokázali vytvárať lepšie modely. Zistili sme, že kroky analýzy a predbežného spracovania údajov majú vplyv na výsledky metód vybraných klasifikátorov. Pre druhý programový modul

strojového učenia sme implementovali optimalizáciu parametrov metód strojového učenia. Počas vypracovania diplomovej práce sme vyskúšali rôzne ďalšie techniky a metódy, ale iba osvedčené sme nakoniec nechali zapracované. Neskôr sa nám podarilo zapracovať ďalšie metódy strojového učenia. Implementácia neurónovej siete a jej optimalizácia je veľkým prínosom tejto práce. Okrem neurónových sietí sme optimalizovali aj ostatné metódy strojového učenia. Na optimalizáciu hyperparametrov sme kládli veľký dôraz. Pomocou optimalizácie sme dokázali vytvoriť konkurencieschopné modely čo zabezpečuje aj možnosť reprezentatívneho porovnania výsledkov ich úspešnosti. Podarilo sa nám pomocou nášho programového modulu pre strojové učenia natrénovali rôzne modely pre rôzne typy predspracovania dátovej množiny UNSW-NB15. Naše modely vykazujú vysoké miery presnosti nielen na dátovej množine UNSW-NB15, ale aj na samotnej simulovanej sietovej premávke. Podľa nášho výskumu vieme určiť ktoré metódy strojového učenia či metódy predspracovania dátovej množiny sú najvhodnejšie pre danú problematiku. Nad predspracovanými dátovými množinami sme aplikovali metódy strojového učenia, konkrétnie: logistická regresia, lineárny klasifikátor (SVM, logistická regresia) s výcvikom SGD, náhodný les, rozhodovací strom, XGradient Boosting klasifikátor, KNN klasifikátor, perceptron, LSTM, DNN a RNN.

V poslednej časti tejto práce sme overili navrhnuté riešenie. Vykonali sme experimentálne aj záverečné testy a na základe výsledkov vyhodnotení úspešností klasifikácií vybraných metód strojového učenia vybrali najlepšiu metódu strojového učenia. Experimentálna časť overenia výsledkov zobrazuje priebežný stav diplomového projektu, kedy sa nám podarilo otestovať prvotné metódy predspracovania dátovej množiny na vybraných metódach strojového učenia. Dosiahli sme vysoké miery presnosti odhalenia sietových útokov. Najlepšou metódou na základe výsledkov experimentálnych testov je XGradient Boosting a náhodný les, pretože v každom teste dosiahli veľmi podobné, stále a vysoké hodnoty správností. Túto skutočnosť potvrdzuje aj overenie modelu strojového učenia pre náhodný les aj na simulovanej sietovej premávke. Na základe testov sme dospeli k záveru, že niektoré metódy strojového učenia nie sú vhodné pre určité typy predspracovania dátovej množiny. Najprv, najhoršie výsledky dosahovali klasifikačné algoritmy ako logistická regresia a SGD klasifikátor, ale ďalšou optimalizáciou sa nám ich podarilo vylepšiť. Najhoršie výsledky dosahuje jednoúrovňová neurónová siet - perceptron, pretože hodnoty správností voči hodnotám F1-skóre boli veľmi kolísavé a tým pádom môžeme túto metódu pokladať sa nespôľahlivú v doméne odhalovania sietových útokov. Na druhej strane sa nám podarilo v prípade týchto klasifikátorov natrénovali aj modely s vysokými presnosťami odhalenia sietových útokov, ako napríklad LSTM či hlboká neurónová siet. Overenie metód strojového učenia na reálnej simulovanej sietovej premávke ukázalo, že okrem perceptronu aj ďalšie metódy neuspeli, ktoré predtým boli veľmi úspešné. Výsledky testov sme interpretovali vo forme tabuľiek a grafov.

Programové moduly výborne poslúžia pre experimentálne a analyzačné účely výskumníkov v oblasti umelej inteligencie a dobovania v dátach. Programové moduly sú ľahko modifikovateľné a udržateľné. Ich obsluha a prevádzkovanie je tiež veľmi jednoduchá, keďže sme kládli dôraz na vývoj voľne dostupného a automatizovaného nástroja s možnosťou škálovania. Oba programové moduly vhodným spôsobom interpretujú, dokumentujú a ukladajú výsledky v na to vyhradenom úložnom priestore pre ďalšiu analýzu či znovupoužitie. Ďalej sme veľký dôraz kládli na detailnú dokumentáciu použitých metód, techník a ich odôvodneniu v tejto práci.

Podarilo sa nám vytvoriť dva plne automatizované programové moduly, ktoré sú výbornou pomôckou pre riešenie problémov podobného charakteru. Naše programové moduly zovšeobecňujú a automatizujú pracovné postupy a sprístupňujú ich širokému publiku, aby mohli používatelia získať prístup k akémukoľvek klasifikátoru či dátovej množine. Rozšírenie programového modulu na predspracovanie dátovej množiny vidíme v zapracovaní ďalších metód predspracovania dátovej množiny a integrácie s modulom pre strojové učenie. Programový modul strojového učenia je možné rozšíriť o ďalšie metódy strojového učenia a optimalizáciu ďalších hyperparametrov. Oba programové moduly je ďalej možné vylepšiť v spôsobe používania. Naše riešenie je možné prirovnáť k sade voľne dostupných nástrojov umelej inteligencie Fairness 360<sup>37</sup> od IBM vydaný v roku 2018, ktorý umožňuje komukoľvek vykonávať transparentnú analýzu na akomkoľvek súbore údajov a klasifikátore.

<sup>37</sup> <https://www.ibm.com/blogs/research/2018/09/ai-fairness-360/>



## Bibliografia

- [1] AHMED, Martuza, Rima PAL, Md. Mojammel HOSSAIN, Md. Abu Naser BIKAS a Md. Khalad HASAN. NIDS: A Network Based Approach to Intrusion Detection and Prevention. In: 2009 International Association of Computer Science and Information Technology - Spring Conference [online]. IEEE, 2009, 2009, s. 141-144 [cit. 2019-05-12]. DOI: 10.1109/IACSIT-SC.2009.96. ISBN 978-0-7695-3653-8. Dostupné z: <http://ieeexplore.ieee.org/document/5169326/>
- [2] AL-JARRAH, Omar a Ahmad ARAFAT. Network Intrusion Detection System using attack behavior classification. In: 2014 5th International Conference on Information and Communication Systems (ICICS) [online]. IEEE, 2014, 2014, s. 1-6 [cit. 2019-05-12]. DOI: 10.1109/IACS.2014.6841978. ISBN 978-1-4799-3023-4. Dostupné z: <http://ieeexplore.ieee.org/document/6841978/>
- [3] ALENCAR, Rafael. Resampling strategies for imbalanced datasets. Kaggle [online]. 2017 [cit. 2019-12-04]. Dostupné z: <https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets>
- [4] ANANTHI, J. Vijitha a S. VENGATESAN. Detection of various attacks in wireless adhoc networks and its performance analysis. In: 2017 International Conference on Inventive Computing and Informatics (ICICI) [online]. IEEE, 2017, 2017, s. 754-757 [cit. 2019-05-12]. DOI: 10.1109/ICICI.2017.8365237. ISBN 978-1-5386-4031-9. Dostupné z: <https://ieeexplore.ieee.org/document/8365237/>
- [5] ASIC (application-specific integrated circuit) [online]. [cit. 2019-12-04]. Dostupné z: <https://whatis.techtarget.com/definition/ASIC-application-specific-integrated-circuit>
- [6] BHARDWAJ, Anshu. Data Preprocessing Techniques for Data Mining. Data Mining Techniques and Tools for Knowledge Discovery in Agricultural Datasets [online]. New Delhi: Division of Computer Applications Indian Agricultural Statistics Research Institute (ICAR), s. 139-144 [cit. 2019-05-22]. Dostupné z: [http://iasri.res.in/ebook/win\\_school\\_aa/notes/Data\\_Preprocessing.pdf](http://iasri.res.in/ebook/win_school_aa/notes/Data_Preprocessing.pdf)

- [7] BHARDWAJ, Anshu. Evaluation Measures for Data Mining Tasks. Data Mining Techniques and Tools for Knowledge Discovery in Agricultural Datasets [online]. New Delhi: Division of Computer Applications Indian Agricultural Statistics Research Institute (ICAR), s. 145-152 [cit. 2019-05-19]. Dostupné z: [http://iasri.res.in/ebook/win\\_school\\_aa/notes/Evaluation\\_Measures.pdf](http://iasri.res.in/ebook/win_school_aa/notes/Evaluation_Measures.pdf)
- [8] BHATTACHARYYA, Dhruba Kumar a Jugal Kumar KALITA. Network Anomaly Detection [online]. Chapman and Hall/CRC, 2013 [cit. 2019-05-12]. DOI: 10.1201/b15088. ISBN 9780429166877.
- [9] CARPENTER, Jason. Parfit — quick and powerful hyper-parameter optimization with visualizations. Medium: ML Review [online]. 2017, 26.11.2017 [cit. 2019-12-07]. Dostupné z: <https://medium.com/mlreview/parfit-hyper-parameter-optimization-77253e7e175e>
- [10] CRONE, Sven F., Stefan LESSMANN a Robert STAHLBOCK. The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing. European Journal of Operational Research [online]. 2006, 173(3), 781-800 [cit. 2020-05-02]. DOI: 10.1016/j.ejor.2005.07.023. ISSN 03772217. Dostupné z: <https://linkinghub.elsevier.com/retrieve/pii/S0377221705006739>
- [11] Canadian Institute for Cybersecurity [online]. Fredericton [cit. 2019-05-13]. Dostupné z: <https://www.unb.ca/cic/datasets/index.html>
- [12] DHAIRYA, Kumar. Introduction to Data Preprocessing in Machine Learning. Towards Data Science [online]. 2018, 25.12.2018 [cit. 2019-12-03]. Dostupné z: <https://towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d>
- [13] DHANABAL, L. a S. P. SHANTHARAJAH. A Study On NSL-KDD Dataset For Intrusion Detection System Based On Classification Algorithms. International Journal of Advanced Research in Computer and Communication Engineering [online]. 2015, 6(4), 446-452 [cit. 2019-05-13]. DOI: 10.17148/IJARCCE.2015.4696. ISSN 2278-1021. Dostupné z: <https://pdfs.semanticscholar.org/1b34/80021c4ab0f632efa99e01a9b073903c5554.pdf>

- [14] FLOPS (floating-point operations per second) [online]. [cit. 2019-12-04]. Dostupné z: <https://whatis.techtarget.com/definition/FLOPS-floating-point-operations-per-second>
- [15] FROST, Jim. Multicollinearity in Regression Analysis: Problems, Detection, and Solutions. Statistics By Jim - Making statistics intuitive [online]. [cit. 2019-12-04]. Dostupné z: <https://statisticsbyjim.com/regression/multicollinearity-in-regression-analysis/>
- [16] GARG, Akash a Prachi MAHESHWARI. A hybrid intrusion detection system: A review. In: 2016 10th International Conference on Intelligent Systems and Control (ISCO) [online]. IEEE, 2016, 2016, s. 1-5 [cit. 2019-05-12]. DOI: 10.1109/ISCO.2016.7726909. ISBN 978-1-4673-7807-9. Dostupné z: <http://ieeexplore.ieee.org/document/7726909/>
- [17] GONZALEZ ZELAYA, Carlos Vladimiro. Towards Explaining the Effects of Data Preprocessing on Machine Learning. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE) [online]. IEEE, 2019, 2019, s. 2086-2090 [cit. 2020-05-02]. DOI: 10.1109/ICDE.2019.00245. ISBN 978-1-5386-7474-1. Dostupné z: <https://ieeexplore.ieee.org/document/8731532/>
- [18] GeoLite2 Free Downloadable Databases [online]. [cit. 2019-12-06]. Dostupné z: <https://dev.maxmind.com/geoip/geoip2/geolite2/>
- [19] HADDADI, Fariba, Sara KHANCHI, Mehran SHETABI a Vali DERHAMI. Intrusion Detection and Attack Classification Using Feed-Forward Neural Network. In: 2010 Second International Conference on Computer and Network Technology [online]. IEEE, 2010, 2010, s. 262-266 [cit. 2019-05-12]. DOI: 10.1109/ICCNT.2010.28. ISBN 978-1-4244-6961-1. Dostupné z: <http://ieeexplore.ieee.org/document/5474495/>
- [20] HINTON, Todd. Introduction to Data Preprocessing in Machine Learning. Redpoint Global [online]. 2018, 10.7.2018 [cit. 2019-12-03]. Dostupné z: <https://www.redpointglobal.com/blog/what-is-data-enrichment/?fbclid=IwAR3lOtF1cBM08sGB8yzgj3evwzK00Nj8Af03oMttUmbpv9pd4Fh2J0tiZH>

- [21] HUSSAIN, Jamal a Aishwarya MISHRA. Performance Analysis of Some Neural Network Algorithms using NSL-KDD Dataset. International Journal of Computer Trends and Technology [online]. 2017, 50(1), 43-49 [cit. 2019-05-12]. DOI: 10.14445/22312803/IJCTT-V50P107. ISSN 22312803. Dostupné z: <http://www.ijcttjournal.org/archives/ijctt-v50p107>
- [22] JAIN, Anurag; BHUPENDRA, L. Classifier selection models for intrusion detection system (IDS). Informatics Engineering, an International Journal (IEIJ), 2016, 4.1: 1-11.
- [23] JAITLEY, Urvashi. Why Data Normalization is necessary for Machine Learning models. Medium [online]. 2018, 8.10.2018 [cit. 2019-12-04]. Dostupné z: <https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029>
- [24] JAJISH, Thomas. Intrusion Detection Systems (IDS), Network Intrusion Detection System (NIDS), Host Intrusion Detection System (HIDS), Signatures, Alerts, Logs, False Alarms. Sensor. Free Networking tutorials, System Administration Tutorials and Security Tutorials - omniseчу.com [online]. [cit. 2019-05-12]. Dostupné z: <http://www.omnisechu.com/security/infrastructure-and-email-security/intrusion-detection-systems-ids.php>
- [25] JIMOH, Hafeez. The tale of missing values in Python. Towards Data Science [online]. 2018, 6.1.2018 [cit. 2019-12-04]. Dostupné z: <https://towardsdatascience.com/the-tale-of-missing-values-in-python-c96beb0e8a9d>
- [26] KANIMOZHI, V. a PREM Jacob, Prem. (2019). UNSW-NB15 dataset feature selection and network intrusion detection using deep learning. International Journal of Recent Technology and Engineering. 7. 443-446.
- [27] KUMAR, Gulshan. Evaluation Metrics for Intrusion Detection Systems - A Study. International Journal of Computer Science and Mobile Applications [online]. 2014, (11), 11-17 [cit. 2019-05-12]. ISSN 2321-8363. Dostupné z: [https://www.researchgate.net/publication/311108073\\_Evaluation\\_Metrics\\_for\\_Intrusion\\_Detection\\_Systems-A\\_Study](https://www.researchgate.net/publication/311108073_Evaluation_Metrics_for_Intrusion_Detection_Systems-A_Study)

- [28] LEMAITRE, G., F. NOGUEIRA, D. OLIVEIRA a C. ARIDAS. Over-sampling. Imbalanced-learn [online]. 2017 [cit. 2019-12-06]. Dostupné z: [https://imbalanced-learn.readthedocs.io/en/stable/over\\_sampling.html](https://imbalanced-learn.readthedocs.io/en/stable/over_sampling.html)
- [29] LI PENG, YUEMING LU, JIEFU GAN a HANG CHENG. Design and implementation of network attacks detection module. In: Third International Conference on Cyberspace Technology (CCT 2015) [online]. Institution of Engineering and Technology, 2015, 2015, 5 .-5 . [cit. 2019-05-12]. DOI: 10.1049/cp.2015.0861. ISBN 978-1-78561-089-9. Dostupné z: <https://digital-library.theiet.org/content/conferences/10.1049/cp.2015.0861>
- [30] LI, Jing a Chunbo DONG. Research on Network Security Situation Prediction-Oriented Adaptive Learning Neuron. In: 2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing [online]. IEEE, 2010, 2010, s. 483-485 [cit. 2019-05-12]. DOI: 10.1109/NSWCTC.2010.247. ISBN 978-1-4244-6597-2. Dostupné z: <http://ieeexplore.ieee.org/document/5480921/>
- [31] LIU, Gao, Zheng YAN a Witold PEDRYCZ. Data collection for attack detection and security measurement in Mobile Ad Hoc Networks: A survey. Journal of Network and Computer Applications [online]. 2018, 105, 105-122 [cit. 2019-05-12]. DOI: 10.1016/j.jnca.2018.01.004. ISSN 10848045. Dostupné z: <https://linkinghub.elsevier.com/retrieve/pii/S1084804518300043>
- [32] MEHMOOD, Tahir a Helmi B Md RAIS. Machine learning algorithms in context of intrusion detection. In: 2016 3rd International Conference on Computer and Information Sciences (ICCOINS) [online]. IEEE, 2016, 2016, s. 369-373 [cit. 2019-05-13]. DOI: 10.1109/ICCOINS.2016.7783243. ISBN 978-1-5090-2549-7. Dostupné z: <http://ieeexplore.ieee.org/document/7783243/>
- [33] MOUSTAFA N., SLAY J. (2018) A Network Forensic Scheme Using Correntropy-Variation for Attack Detection. In: Peterson G., Shenoi S. (eds) Advances in Digital Forensics XIV. DigitalForensics 2018. IFIP Advances in Information and Communication Technology, vol 532. Springer, Cham

[34] MOUSTAFA, Nour a Jill SLAY. The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Information Security Journal: A Global Perspective [online]. 2016, 25(1-3), 18-31 [cit. 2019-05-19]. DOI: 10.1080/19393555.2015.1125974. ISSN 1939-3555. Dostupné z: <http://www.tandfonline.com/doi/full/10.1080/19393555.2015.1125974>

[35] MOUSTAFA, Nour a Jill SLAY. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS) [online]. IEEE, 2015, 2015, s. 1-6 [cit. 2019-05-13]. DOI: 10.1109/MilCIS.2015.7348942. ISBN 978-1-4673-7007-3. Dostupné z: <http://ieeexplore.ieee.org/document/7348942/>

[36] Multicollinearity [online]. [cit. 20819-12-04]. Dostupné z: <https://www.statisticssolutions.com/multicollinearity/>

[37] NAVLANI, Avinash. Understanding Logistic Regression in Python. DataCamp [online]. 2018, 7.9.2018 [cit. 2019-12-03]. Dostupné z: <https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python>

[38] PRABHU. Understanding Hyperparameters and its Optimisation techniques. Towards Data Science [online]. 2018, 3.7.2018 [cit. 2019-12-05]. Dostupné z: <https://towardsdatascience.com/understanding-hyperparameters-and-its-optimisation-techniques-f0debba07568>

[39] REVATHI, S. a A. MALATHI. A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection. International Journal Of Engineering Research & Technology [online]. IJERT, 2013, 12(2), 1848-1853 [cit. 2019-05-13]. ISSN 2278-0181. Dostupné z: <https://www.ijert.org/research/a-detailed-analysis-on-nsl-kdd-dataset-using-various-machine-learning-techniques-for-intrusion-detection-IJERTV2IS120804.pdf>

[40] ROY, Bipraneel a Hon CHEUNG. A Deep Learning Approach for Intrusion Detection in Internet of Things using Bi-Directional Long Short-Term Memory Recurrent Neural Network. In: 2018 28th International Telecommunication Networks and Applications Conference (ITNAC) [online]. IEEE, 2018, 2018, s. 1-6 [cit. 2020-05-02]. DOI: 10.1109/ATNAC.2018.8615294. ISBN 978-1-5386-7177-1. Dostupné z: <https://ieeexplore.ieee.org/document/8615294/>

- [41] SAMRIN, Rafath a D VASUMATHI. Review on anomaly based network intrusion detection system. In: 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT) [online]. IEEE, 2017, 2017, s. 141-147 [cit. 2019-05-12]. DOI: 10.1109/ICEECCOT.2017.8284655. ISBN 978-1-5386-1205-7. Dostupné z: <http://ieeexplore.ieee.org/document/8284655/>
- [42] SANI, Yusuf, Ahmed MOHAMEDOU, Khalid ALI, Anahita FARJAMFAR, Mohamed AZMAN a Solahuddin SHAMSUDDIN. An overview of neural networks use in anomaly Intrusion Detection Systems. In: 2009 IEEE Student Conference on Research and Development (SCORED) [online]. IEEE, 2009, 2009, s. 89-92 [cit. 2019-05-12]. DOI: 10.1109/SCORED.2009.5443289. ISBN 978-1-4244-5186-9. Dostupné z: <http://ieeexplore.ieee.org/document/5443289/>
- [43] SAV File Format [online]. [cit. 2019-12-04]. Dostupné z: <https://whatis.techtarget.com/fileformat/SAV-Saved-date-file-for-SPSS-Statistical-Package-for-the-Social-Sciences>
- [44] SAXENA, Aumreesh Ku., Sitesh SINHA a Piyush SHUKLA. General study of intrusion detection system and survey of agent based intrusion detection system. In: 2017 International Conference on Computing, Communication and Automation (ICCCA) [online]. IEEE, 2017, 2017, s. 471-421 [cit. 2019-05-12]. DOI: 10.1109/ICCCA.2017.8229866. ISBN 978-1-5090-6471-7. Dostupné z: <http://ieeexplore.ieee.org/document/8229866/>
- [45] SCHAELICKE, Lambert, Thomas SLABACH, Branden MOORE a Curt FREELAND. Characterizing the Performance of Network Intrusion Detection Sensors. VIGNA, Giovanni, Christopher KRUEGEL a Erland JONSSON, ed. Recent Advances in Intrusion Detection [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, 2003, s. 155-172 [cit. 2019-05-12]. Lecture Notes in Computer Science. DOI: 10.1007/978-3-540-45248-5\_9. ISBN 978-3-540-40878-9. Dostupné z: [http://link.springer.com/10.1007/978-3-540-45248-5\\_9](http://link.springer.com/10.1007/978-3-540-45248-5_9)
- [46] SCHOONJANS, Frank. ROC curve analysis. MedCalc [online]. Belgium [cit. 2019-05-06]. Dostupné z: <https://www.medcalc.org/manual/roc-curves.php>

- [47] SENAPATI, Deepak. Grid Search vs Random Search. Medium [online]. 2018, 29.8.2018 [cit. 2019-12-06]. Dostupné z: <https://medium.com/@senapati.dipak97/grid-search-vs-random-search-d34c92946318>
- [48] SHAH, Tarang. About Train, Validation and Test Sets in Machine Learning. Towards Data Science [online]. 2017, 6.12.2017 [cit. 2019-12-05]. Dostupné z: <https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7>
- [49] SHAIKH, Raheel. Feature Selection Techniques in Machine Learning with Python. Towards Data Science [online]. 2018, 28.10.2018 [cit. 2019-12-04]. Dostupné z: <https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e>
- [50] SHEELA EVANGELIN PRASAD, S. N., M. V. SRINATH a Murtaza SAADIQUE BASHA. Intrusion Detection Systems, Tools and Techniques – An Overview. Indian Journal of Science and Technology [online]. 2015, 8(35) [cit. 2019-05-12]. DOI: 10.17485/ijst/2015/v8i35/80108. ISSN 0974-5645. Dostupné z: <http://www.indjst.org/index.php/indjst/article/view/80108>
- [51] SOHEILY-KHAH, Saeid, Pierre-Francois MARTEAU a Nicolas BECHET. Intrusion Detection in Network Systems Through Hybrid Supervised and Unsupervised Machine Learning Process: A Case Study on the ISCX Dataset. In: 2018 1st International Conference on Data Intelligence and Security (ICDIS) [online]. IEEE, 2018, 2018, s. 219-226 [cit. 2019-05-19]. DOI: 10.1109/ICDIS.2018.00043. ISBN 978-1-5386-5762-1. Dostupné z: <https://ieeexplore.ieee.org/document/8367767/>
- [52] THOMAS, Rajesh a Deepa PAVITHRAN. A Survey of Intrusion Detection Models based on NSL-KDD Data Set. In: 2018 Fifth HCT Information Technology Trends (ITT) [online]. IEEE, 2018, 2018, s. 286-291 [cit. 2019-05-13]. DOI: 10.1109/CTIT.2018.8649498. ISBN 978-1-5386-7147-4. Dostupné z: <https://ieeexplore.ieee.org/document/8649498/>
- [53] TIWARI, Mohit, Raj KUMAR, Akash BHARTI a Jai KISHAN. Intrusion Detection Systems. International Journal of Computer Science and Mobile Applications [online]. 2017, (5), 38-44 [cit. 2019-05-12]. ISSN 2320-8163. Dostupné z: [https://www.researchgate.net/publication/316599266\\_INTRUSION\\_DETECTION\\_SYSTEM](https://www.researchgate.net/publication/316599266_INTRUSION_DETECTION_SYSTEM)

[54] TPUs in Colab [online]. [cit. 2019-12-06]. Dostupné z: [https://colab.research.google.com/notebooks/tpu.ipynb#scrollTo=a\\_rjVo-RAoYd](https://colab.research.google.com/notebooks/tpu.ipynb#scrollTo=a_rjVo-RAoYd)

[55] The UNSW-NB15 Dataset Description [online]. 2018 [cit. 2019-05-14]. Dostupné z: <https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/>

[56] WARZYŃSKI, Arkadiusz; KOŁACZEK, Grzegorz. Intrusion detection systems vulnerability on adversarial examples. In: 2018 Innovations in Intelligent Systems and Applications (INISTA). IEEE, 2018. p. 1-4.

[57] WILLIAMS, Nigel, Sebastian ZANDER a Grenville ARMITAGE. A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification. ACM SIGCOMM Computer Communication Review [online]. 2006, 36(5) [cit. 2019-05-22]. DOI: 10.1145/1163593.1163596. ISSN 01464833. Dostupné z: <http://portal.acm.org/citation.cfm?doid=1163593.1163596>

[58] What is Correntropy [online]. [cit. 2019-12-03]. Dostupné z: <https://www.igi-global.com/dictionary/information-theoretic-learning/6003>

[59] XIA WEI-WEI a WANG HAI-FENG. Prediction model of network security situation based on regression analysis. In: 2010 IEEE International Conference on Wireless Communications, Networking and Information Security [online]. IEEE, 2010, 2010, s. 616-619 [cit. 2018-11-10]. DOI: 10.1109/WCINS.2010.5541853. ISBN 978-1-4244-5850-9. Dostupné z: <http://ieeexplore.ieee.org/document/5541853/>

[60] ZAMAN, Marzia a Chung-Horng LUNG. Evaluation of machine learning techniques for network intrusion detection. In: NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium [online]. IEEE, 2018, 2018, s. 1-5 [cit. 2019-05-21]. DOI: 10.1109/NOMS.2018.8406212. ISBN 978-1-5386-3416-5. Dostupné z: <https://ieeexplore.ieee.org/document/8406212/>

[61] MOUSTAFA, Nour a Jill SLAY. The Significant Features of the UNSW-NB15 and the KDD99 Data Sets for Network Intrusion Detection Systems. In: 2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS) [online]. IEEE, 2015, 2015, s. 25-31 [cit. 2020-05-03]. DOI: 10.1109/BADGERS.2015.014. ISBN 978-1-4673-8944-0. Dostupné z: <http://ieeexplore.ieee.org/document/7809531/>



## Príloha A: Plán práce

Tabuľka A.30 uvádza plán práce v rámci prvej etapy riešenia diplomovej práce.

Tabuľka A.30 – Plán práce k DP I

DP I	
Týždeň	Ciel'
<b>1-2</b>	Analýza existujúcich riešení IDS (štruktúra, metódy, algoritmy).
<b>3-5</b>	Analýza metód spracovania veľkej množiny dát z prostredia počítačových sietí.
<b>6-8</b>	Analýza a zaúčanie sa do strojového učenia.
<b>9-10</b>	Odkúšanie získaných znalostí na vzorových príkladoch. Experimentovanie.
<b>11-12</b>	Úprava a dokončenie dokumentácie.

Jednotlivé body vyššie uvedeného plánu boli mierne náročné na vypracovanie. Podarilo sa nám zanalyzovať rôzne aspekty vybranej problémovej oblasti.

Zanalyzovali sme IDS systémy, ich architektúru, existujúce riešenia zaoberajúce sa odhalením siet'ových útokov a spôsob vyhodnocovania úspešnosti IDS. Taktiež sa nám podarilo dostatočne zanalyzovať siet'ové útoky, dátové množiny a strojové učenie. V prvej etape diplomovej práce sme taktiež stihli aplikovať získané znalosti o strojovom učení na dátovej množine NSL-KDD.

Navrhli sme vlastné riešenie na základe preštudovania danej problematiky. Stanovenie podmienok na prvú etapu sa nám podarilo splniť a dokument sa tiež úspešne upravil do finálnej podoby na základe podmienok vedúceho práce.

Tabuľka A.31 uvádza plán práce v rámci druhej etapy riešenia diplomovej práce.

Tabuľka A.31 – Plán práce k DP II

DP II	
Týždeň	Ciel'
<b>1-2</b>	Zbieranie dát z prostredia počítačových sietí a ich predpríprava.
<b>3-4</b>	Návrh vlastného riešenia.
<b>5-8</b>	Implementácia zadania.
<b>9-10</b>	Analýza výsledkov vlastného riešenia. Odhalenie nedostatkov. Zapracovanie zmien do implementácie.
<b>11-12</b>	Úprava a dokončenie dokumentácie. Príprava na obhajobu DP II.

Jednotlivé body vyššie uvedeného plánu práce k DP II boli mierne náročné na vypracovanie. Počas práce na druhej etape sme sa stretli s viacerými problémami, ktoré boli časovo náročné na vyriešenie.

Podarilo sa nám stiahnuť rozsiahlu množinu dát UNSW-NB15, nad ktorou sme aplikovali rôzne metódy predspracovania dátovej množiny. Podarilo sa nám predspracovať dátové množiny rôznymi spôsobmi. Nad predspracovanými dátovými množinami sme aplikovali metódy strojového učenia, konkrétnie logistická regresia, lineárny klasifikátor (SVM, logistická regresia) s výcvikom SGD, náhodný les a rozhodovací strom. Dosiahli sme vysoké miery presnosti odhalenia sietových útokov. Počas vývoja prototypu sme čeliли rôznym nedostatkom, ktoré sa ale podarilo vyriešiť a zapracovať.

Počas práce na druhej etape sme pracovali aj na zdokonalení návrhu vlastného riešenia, ktorý podrobne opisujeme vyššie v tomto dokumente.

Stanovenie podmienok na druhú etapu sa nám podarilo splniť a dokument sa tiež úspešne upravil do finálnej podoby na základe podmienok vedúceho práce.

Tabuľka A.32 uvádza plán práce v rámci tretej etapy riešenia diplomovej práce.

Tabuľka A.32 – Plán práce k DP III

DP III	
Týždeň	Ciel'
1-4	Korekcia chýb návrhu na základe pripomienok od vedúceho projektu. Implementácia ďalších častí návrhu z druhej etapy. Implementácia ďalších metód strojového učenia, hlavne neurónovej siete.
5-6	Testovanie a zhodnotenie výsledkov implementovaného nástroja.
7-8	Finálne úpravy v projekte. Kontrola a odhalovanie chýb.
9-10	Dokončenie dokumentácie.
11-12	Príprava na obhajobu DP III.

Jednotlivé body vyššie uvedeného plánu práce k DP III boli pomerne náročné na vypracovanie. Počas práce na tretej etape sme sa stretávali s problémami iného charakteru ako predtým a ich odstránenie bolo časovo náročnejšie. Tieto problémy predstavovali odstraňovanie chýb v programových moduloch a testovanie. Keďže sme pracovali s rozsiahlymi dátovými množinami tak pretestovanie niektorých zmien trvalo niekedy aj viac hodín. Stretávali sme sa najviac s limitáciami vývojového prostredia čo viedlo k zakúpeniu Pro verzie Google Colab.

Vykonali sme zásadné zmeny v implementácii, ktoré viedli k lepším trénovaniam a modelom klasifikačných metód strojového učenia. V rámci tretej etapy sme implementovali chýbajúce časti programových modulov, ktoré sme nestihli v druhej etape. Implementácia porovnania metód strojového učenia pridáva na hodnote programového modulu strojového učenia. Implementácia tejto vlastnosti pritom nebola náročná. Súčasťou ďalšej implementácie bola aj implementácia ďalších metód strojového učenia – neurónových sietí ako LSTM, hlboká neurónová sieť, rekurentná neurónová sieť či perceptron.

Ďalej sme vykonali sériu predspracovaní dátových množín rôznymi spôsobmi a následne sme natrénovali inteligentné metódy strojového učenia na týchto predspracovaných dátových množinách. Dokázali sme tak vytvoriť dostatočné množstvo dát pre účely overenia nášho riešenia.

V poslednej časti tretej etapy sme doplnili dokumentáciu o nové poznatky, opravili chyby v dokumentácii a na záver sme pripravili diplomovú prácu na odovzdanie.



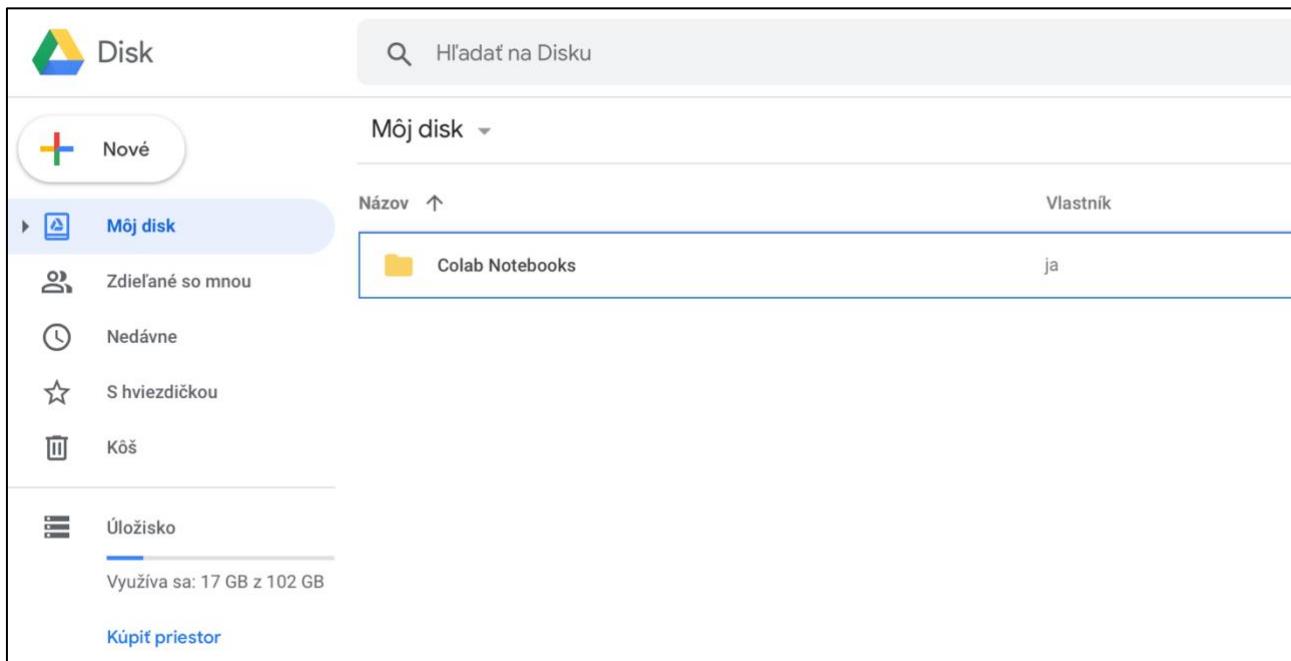
## Príloha B: Technická dokumentácia

Nižšie v tejto kapitole sa venujeme podrobnej dokumentácii implementovaných programových modulov na predspracovanie dátovej množiny a strojového učenia. Opíšeme vývojové prostredie Google Colab a adresárovú štruktúru úložiska Google Drive.

### B.1 Používateľská príručka pre Google Colab

Jednou z minimálnych požiadaviek pre bezproblémový beh programového modulu na predspracovanie dátovej množiny a programového modulu strojového učenia je stabilné internetové pripojenie a používateľský účet od spoločnosti Google. Programové moduly je možné spustiť v ľubovoľnom internetovom prehliadači v prostredí Google Colab. Za účelom ukladania výstupov z programových modulov je potrebný dostatok voľného priestoru v úložisku Google Drive. Pre bežnú analýzu, predspracovanie dátovej množiny a strojové učenie je voľne dostupných 15GB úložného priestoru od Google postačujúci. Pre prístup k súborovej štruktúre v úložnom priestore Google Drive je potrené sa prihlásiť cez Google používateľský účet. Google Colab je taktiež voľne dostupná clouдовá služba, ktorá integruje prostredie Jupyter Notebook.

Nasledujúci obrázok zobrazuje prostredie Google Drive.

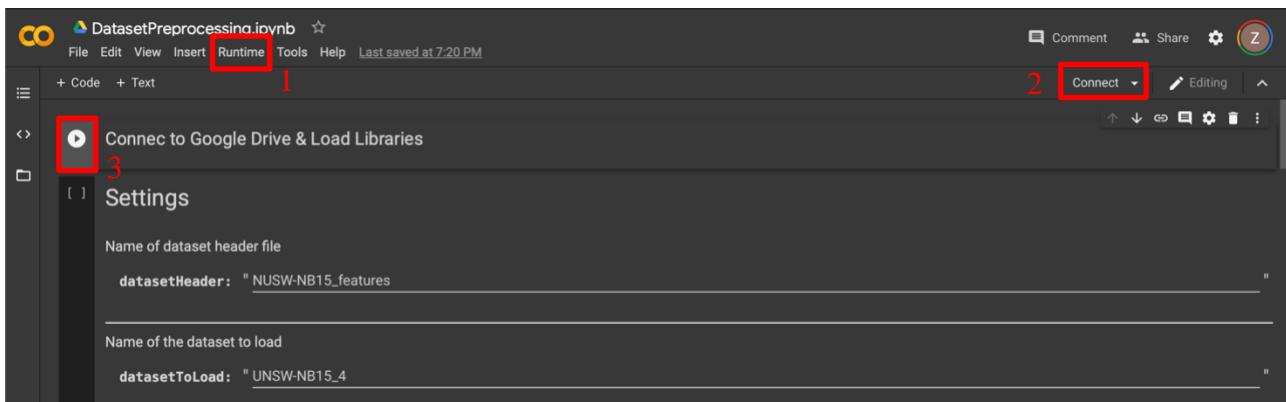


Obrázok 41 – Google Drive úložný priestor

Adresárová štruktúra Google Drive musí zodpovedať nasledovnej štruktúre a jednotlivé adresáre musia obsahovať nižšie spomenuté súbory:

- \Colab Notebooks – koreňový adresár
  - \Dataset – adresár s dátovou množinou
    - **NUSW-NB15\_features.csv**
    - **UNSW-NB15\_1.csv**
    - **UNSW-NB15\_2.csv**
    - **UNSW-NB15\_3.csv**
    - **UNSW-NB15\_4.csv**
  - \Documentation – adresár pre správy jednotlivých programových modulov
    - \DatasetPreprocessing – adresár pre správy programového modulu na predspracovanie dátovej množiny
    - \MachineLearning – adresár pre správy programového modulu strojového učenia
  - \MachineLearningModels – adresár pre modely strojového učenia
  - \Notebooks – adresár pre Google Colab Notebooky
    - \Lib – adresár knižníc
      - **Config.py**
      - **DatasetPreprocessingController.py**
      - **MachineLearningController.py**
    - **DatasetPreprocessing.ipynb** – programový modul na predspracovanie dátovej množiny
    - **MachineLearning.ipynb** – programový modul strojového učenia
  - \PreprocessedDatasets – adresár pre predspracované dátové množiny
  - \Resources – adresár so zdrojmi
    - **GeoLite2-Country.mmdb**

Jednotlivé programové moduly je možné spustiť z adresára \Colab Notebooks\Notebooks. Spustením jedného z programových modulov s príponou *.ipynb* sa otvorí nová záložka v internetovom prehliadači s prostredím Google Colab. Nasledujúci obrázok zobrazuje prostredie Google Colab.



Obrázok 42 – Prostredie Google Colab

V prostredí Google Colab je potrebné nastaviť v časti *Runtime/Change runtime type* (vid. obrázok č. 42 číslo 1):

- Typ runtime na **Python 3**
- Hardvérový urýchľovač na **TPU**
- Tvaru runtime na **High-RAM** pre Google Colab Pro

Kliknutí na tlačidlo Connect vyznačený číslom 2 na obrázku č. 42 sa pripojíte k hostovanému počítačovému modulu. Obrázok nižšie znázorňuje úspešné pripojenie k počítačovému modulu od spoločnosti Google.

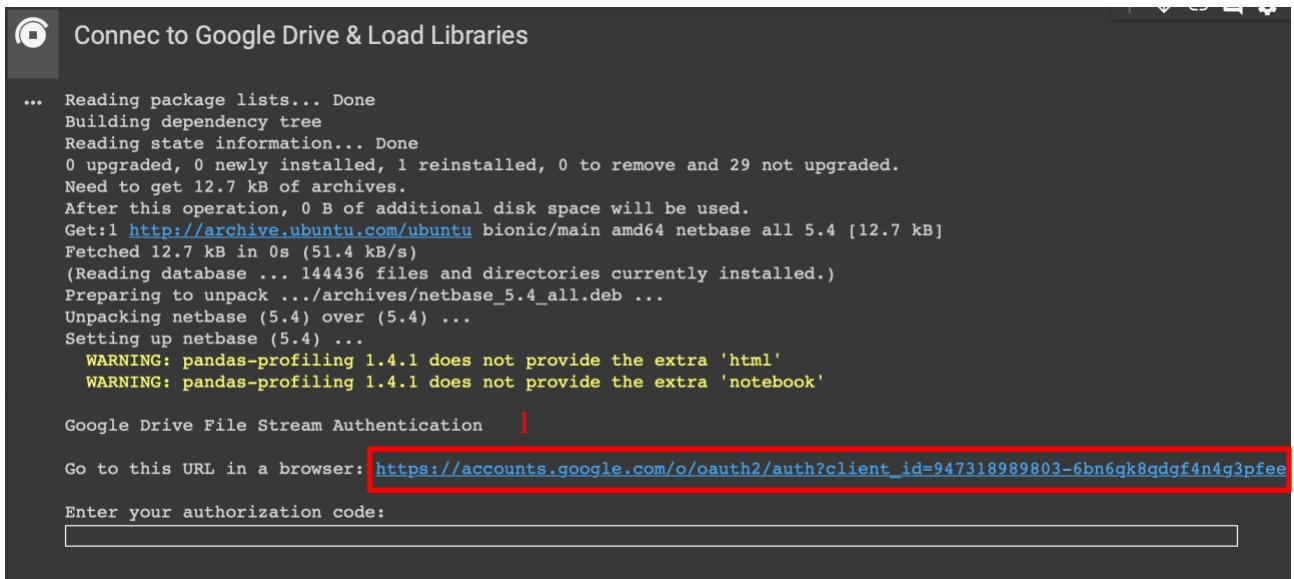


Obrázok 43 – Počítačový modul Google

Prístup k adresárovej štruktúre Google Drive je možný aj cez prostredie Google Colab. Adresárová štruktúra je dostupná po overení používateľa pomocou Google Drive File Stream Authentication. Pre overenie je potrebné spustiť prvú bunku s názvom " Connec to Google Drive & Load Libraries", každého programového modulu zvlášť.

Jednotlivé bunky notebooku je možné spúšťať samostatne kliknutím na tlačidlo 3 vyznačené na obrázku č. 42, alebo spustiť všetky bunky naraz cez *Runtime/Run all*. Ak chceme zastaviť bežiaci proces (bunku) tak v časti *Runtime/Interrupt execution* vyvoláme prerušenie vykonávania bežiaceho procesu. Prerušenie procesu je možné vykonať aj cez kontextové menu pravým klikom myši na bunku, ktorá aktuálne beží.

Po pustení bunky č. 3 na obrázku č. 42 sa v časti výstup pre bunku zobrazí textové pole s názvom "Enter your authorization code" kam je potrebné skopírovať unikátny overovací kľúč. Tento kľúč sa vygeneruje po autentifikácii používateľa kliknutím na odkaz nad textovým poľom. Postup overenia zobrazujú obrázky č. 44 až 46.



```
Connec to Google Drive & Load Libraries

...
Reading package lists... Done
Building dependency tree
Reading state information... Done
0 upgraded, 0 newly installed, 1 reinstalled, 0 to remove and 29 not upgraded.
Need to get 12.7 kB of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 netbase all 5.4 [12.7 kB]
Fetched 12.7 kB in 0s (51.4 kB/s)
(Reading database ... 144436 files and directories currently installed.)
Preparing to unpack .../archives/netbase_5.4_all.deb ...
Unpacking netbase (5.4) over (5.4) ...
Setting up netbase (5.4) ...
WARNING: pandas-profiling 1.4.1 does not provide the extra 'html'
WARNING: pandas-profiling 1.4.1 does not provide the extra 'notebook'

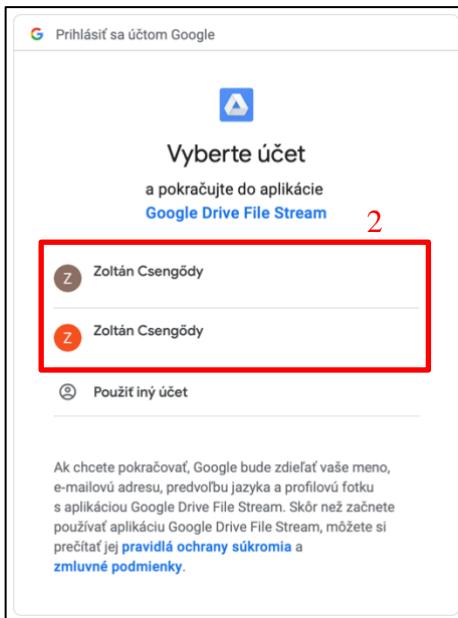
Google Drive File Stream Authentication [1]

Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client\_id=947318989803-6bn6qk8qdgf4n4g3pfef
[2]

Enter your authorization code:
[ ]
```

Obrázok 44 – Autentifikácia – krok 1

Používateľ musí vybrať jeden zo svojich Google účtov.



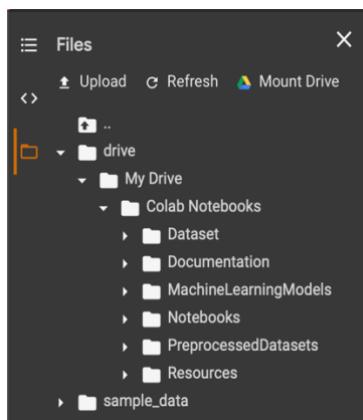
Obrázok 45 – Autentifikácia – krok 2

Používateľ následne musí potvrdiť prístup Google Drive File Stream k používateľskému účtu. Následne sa zobrazí obrazovka s unikátnym kľúčom, ktorý je potrebný skopírovať do textového poľa spomenutý vyššie a potvrdiť stlačením klávesy Enter.



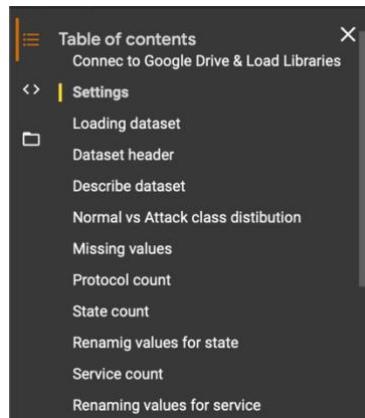
Obrázok 46 – Autentifikácia – krok 3

Po autentifikácii používateľa je prostredie Google Colab pripravené, aby programové moduly mohli byť spustené. Taktiež máme prístup k adresárovej štruktúre Google Drive. Nasledujúci obrázok č. 47 znázorňuje ľavé bočné menu Google Colab pre prístup ku Google Drive adresárovej štruktúre.



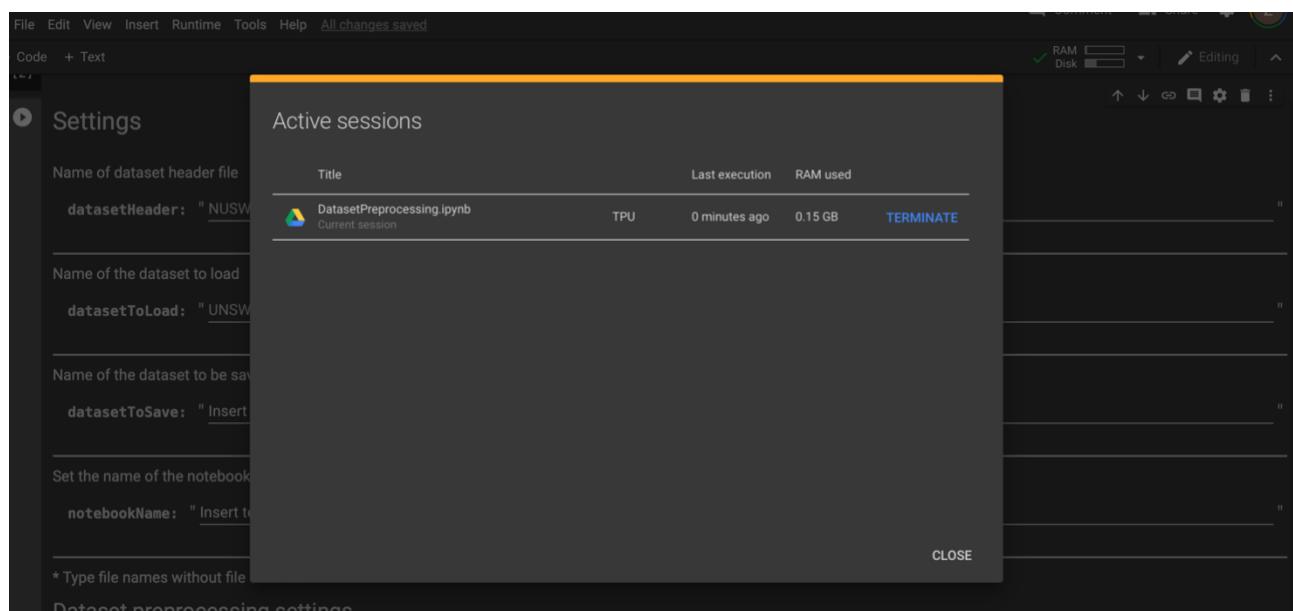
Obrázok 47 – Google Drive adresárová štruktúra v prostredí Google Colab

Každý programový modul má svoj vlastný obsah programových buniek, pre rýchli prístup k jednotlivým bunkám. Obsah je znázornený na nasledujúcom obrázku č. 48.



Obrázok 48 – Obsah programového modulu

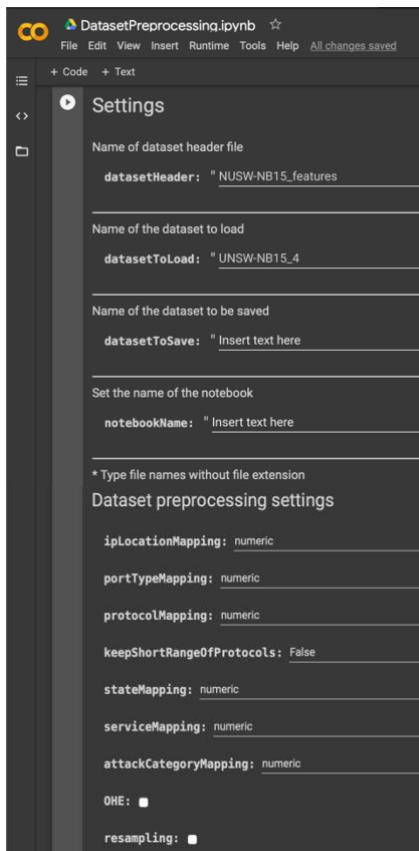
Ak sa používateľ pripojí k počítačovému modulu cez jeden programový modul v prostredí Google Colab a chce sa pripojiť aj s druhým programovým modulom k počítačovému modulu, tak túto možnosť Google nepodporuje. Používateľský účet môže byť pripojený naraz iba k jednému počítačovému modulu. Ak sa používateľ chce pripojiť k počítačovému modulu z druhého programového modulu, tak najskôr musí ukončiť spojenie s počítačovým modulom pre prvý programový modul, ku ktorému je pripojený a zopakovať kroky pripojenia k počítačovému modulu ešte raz pre druhý programový modul. Zrušenie spojenia je cez položku *Runtime/Manage sessions* v hlavnom menu v prostredí Google Colab. Ukončenie spojenia s počítačovým modulom znázorňuje obrázok nižšie.



Obrázok 49 – Ukončenie spojenia s počítačovým modulom

## B.2 Používateľská príručka pre programový modul na predspracovanie dátovej množiny

V tejto kapitole opíšeme použitie programového modulu na predspracovanie dátovej množiny. Nasledujúci obrázok zobrazuje nastavenia pre programový modul na predspracovanie dátovej množiny.



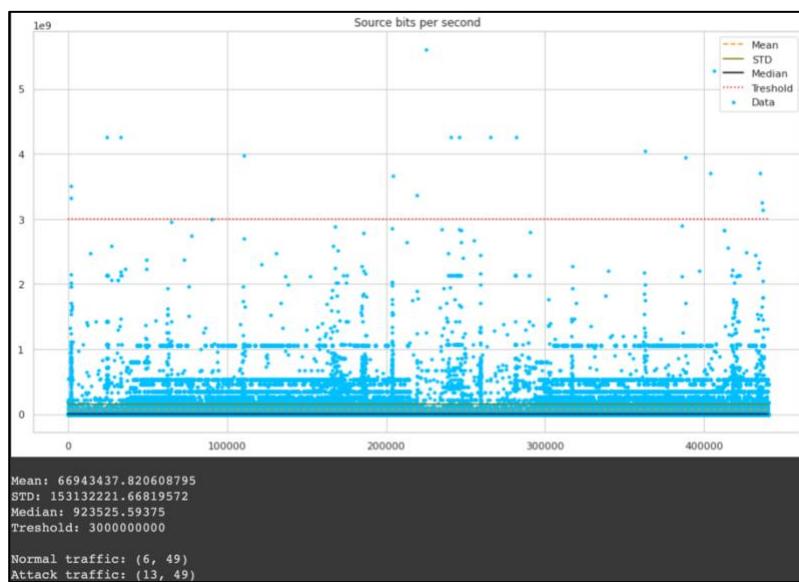
Obrázok 50 – Nastavenie programového modulu na predspracovanie dátovej množiny

Vysvetlenie nastavení:

- **datasetHeader** – názov CSV súboru pre hlavičku dátovej množiny UNSW-NB15
- **datasetToLoad** – názov dátovej množiny na načítanie
- **datasetToSave** – názov predspracovanej dátovej množiny
- **notebookName** – názov správy
- Spôsob spracovania jednotlivých atribútov dátovej množiny UNSW-NB15. Spôsoby sú dva: numerické mapovanie (numeric) kategorických hodnôt alebo použitie metódy One-Hot Encoding (nominal).
  - **ipLocationMapping** – spôsob spracovania zdrojovej a cielovej IP adresy

- **portTypeMapping** – spôsob spracovania typu portu
- **protocolMapping** – spôsob spracovania atribútu *proto*
- **keepShortRangeOfProtocols** – skrátené mapovanie protokolov. Skrátené protokoly sú: *tcp*, *udp*, *arp*, *ospf*, *icmp*, *gre*, *sctp*. Ostatné menej používané protokoly sa označia príznakom "other".
- **stateMapping** – spôsob spracovania atribútu *state*
- **serviceMapping** – spôsob spracovania atribútu *service*
- **attackCategoryMapping** – spôsob spracovania atribútu *attack\_cat*
- **OHE** – použitie metódy One-Hot Encoding. One-Hot Encoding sa použije pri spracovaní kategorických atribútov.
- **resampling** – použitie prevzorkovania

Pre atribúty: *dur*, *sbytes*, *dbytes*, *sttl*, *ttl*, *sload*, *dload*, *spkts*, *dpkts*, *smeansz*, *dmeansz*, *trans\_depth*, *sintpkt* a *dintpkt* používateľ môže zvoliť prahovú hodnotu (threshold), nad ktorou sa má spočítať distribúcia normálnej a útočnej siet'ovej premávky. Pomocou prahovej hodnoty vieme zistíť zastúpenie útočnej siet'ovej premávky voči normálnej. Prahouvá hodnota sa zobrazí na grafe distribúcie hodnôt pre daný atribút ako červená trhaná oddel'ovacia horizontálna čiara. Nasledujúci obrázok znázorňuje takýto príklad.

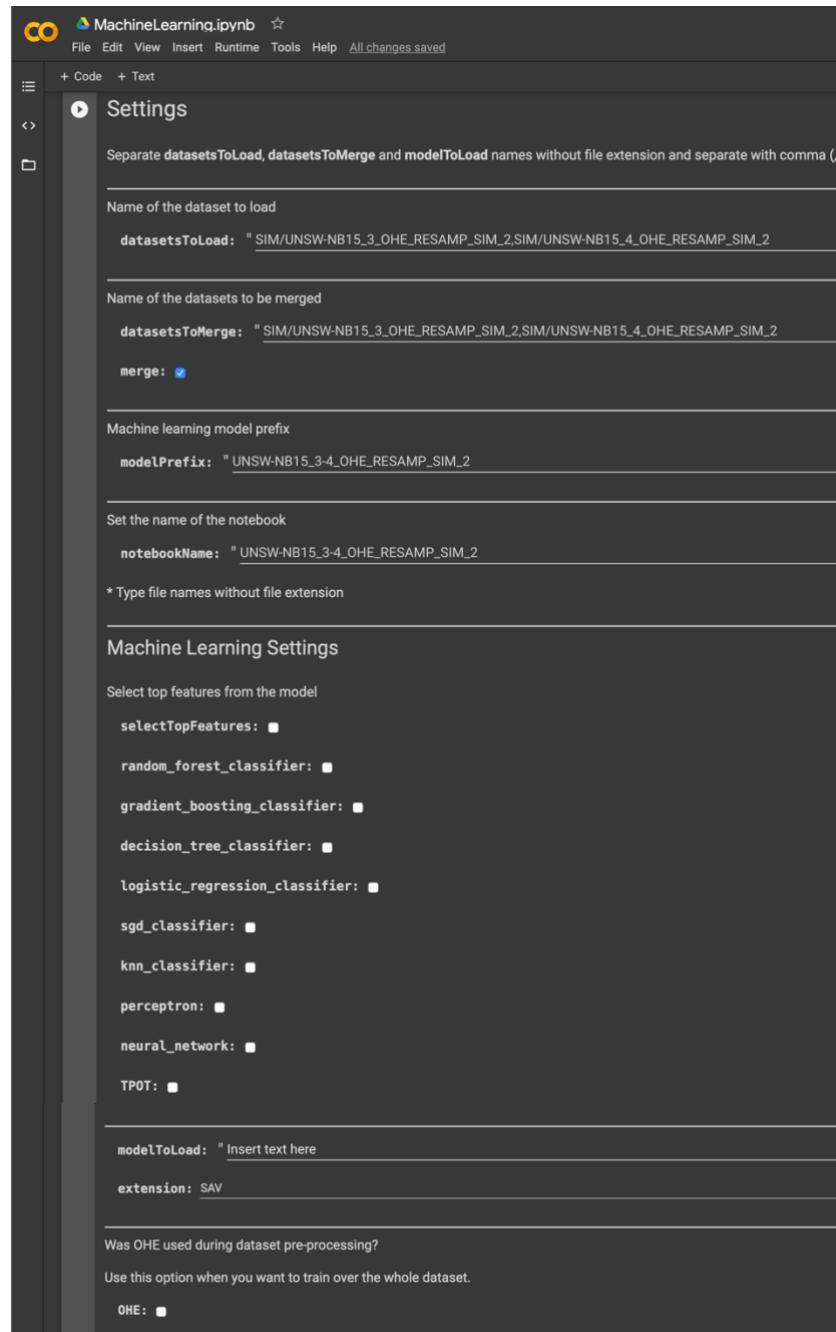


Obrázok 51 – Prahouvá hodnota - threshold

Z obrázku vyššie vieme vyčítať, že nad prahovou hodnotou 3 000 000 000 pre atribút *sload* sa nachádza 13 hodnôt pre útočnú siet'ovú premávku a 6 hodnôt pre normálnu siet'ovú premávku.

## B.2 Používateľská príručka pre programový modul strojového učenia

V tejto kapitole opíšeme použitie programového modulu pre strojové učenie. Nasledujúci obrázok zobrazuje nastavenia pre programový modul strojového učenia.

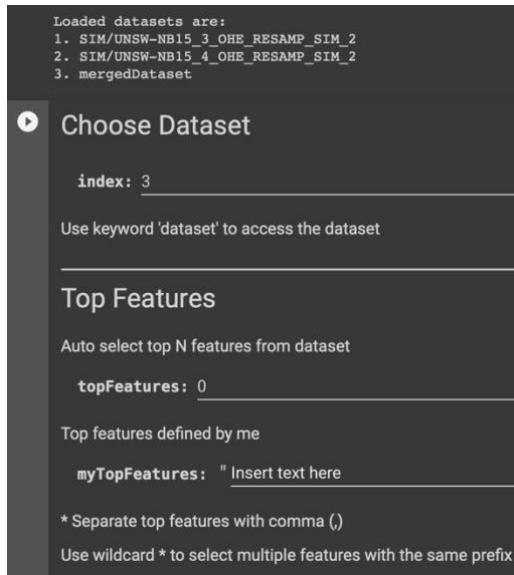


Obrázok 52 – Nastavenie programového modulu pre strojové učenie

## Vysvetlenie nastavení:

- **datasetsToLoad** – názov predspracovanej dátovej množiny na načítanie
- **datasetsToMerge** – zoznam názvov predspracovaných dátových množín určené na zlúčenie
- **merge** – voľba či sa majú dátové množiny zlúčiť
- **modelPrefix** – prefix pre model strojového učenia
- **notebookName** – názov správy
- **selectTopFeatures** – výber najlepších atribútov z modelu strojového učenia pomocou metódy *SelectFromModel* z knižnice *sklearn.feature\_selection*
- Zoznam metód strojového učenia
  - **random\_forest\_classifier** – Náhodný les
  - **gradient\_boosting\_classifier** – XGradient Boosting
  - **decision\_tree\_classifier** – Rozhodovací strom
  - **logistic\_regression\_classifier** – Logistická regresia
  - **sgd\_classifier** – SGD klasifikátor
  - **knn\_classifier** – K-nearest neighbors
  - **perceptron** – Perceptron
  - **neural\_network** – Neurónová siet
    - Long short-term memory (LSTM) neurónová siet'
    - Rekurentná neurónová siet' (Recurrent Neural Network – RNN)
    - Hlboká neurónová siet' (Deep Neural Network – DNN)
- **TPOT** – použitie automatizovaného nástroja strojového učenia TPOT
- **modelToLoad** – názov modelu strojového učenia na načítanie
- **extension** – prípona modelu strojového učenia na načítanie
- **OHE** – ak bola použitá metóda One-Hot Encoding počas predspracovania dátovej množiny, tak optimalizácia hyperparametrov sa vykoná s menej iteráciami a počtom krízových validácií

V prípade načítania viacerých predspracovaných dátových množín používateľ musí vybrať, ktorý chce použiť. Z obrázka vyššie môžeme vyčítať dva názvy predspracovaných dátových množín. Tieto dve predspracované dátové množiny sú zároveň určené na zlúčenie. Obrázok nižšie znázorňuje dve načítané predspracované dátové množiny a jednu zlúčenú dátovú množinu. Používateľ zadánim hodnoty indexu (viď. obrázok č. 53) zvolenej dátovej množiny vyberie dátovú množinu pre ďalšie použitie.



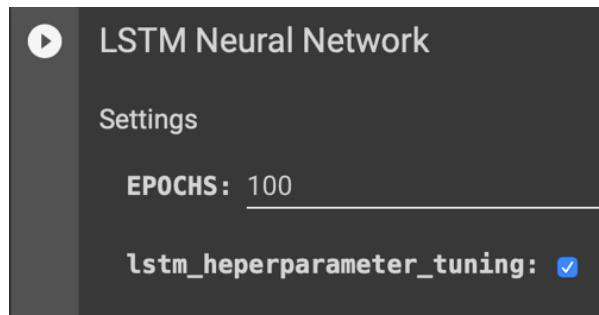
Obrázok 53 – Načítanie predspracovanej dátovej množiny

V prípade zadania celočíselnej hodnoty pre nastavenie *topFeatures* sa z vybranej dátovej množiny vyberie N najlepších atribútov pomocou metódy *SelectKBest* z knižnice *sklearn.feature\_selection*.

Ak používateľ zadá zoznam atribútov ako reťazec pre nastavenie *myTopFeatures*, tak z vybranej dátovej množiny sa vyberú používateľom zadefinované atribúty. Atribúty, ktoré používateľ zadá musia existovať vo vybranej predspracovanej dátovej množine.

V prípade logistickej regresie výber najlepších atribútov má osobitný prístup, pretože jedine v prípade tohto klasifikátora sme sa stretli s fenoménom multikolinearita. Ak má nastavenie *enableFeatureSelection* hodnotu True, tak výber najlepších atribútov z načítanej predspracovanej dátovej množiny je povolený. Výber najlepších atribútov pre logistickú regresiu sa realizuje pomocou metódy *RFE* z knižnice *sklearn.feature\_selection*. V prípade výskytu fenoménu multikolinearita je potrebné odobrať zo zoznamu najlepších atribútov ďalšie atribúty zadáním číselného rozmedzia poriadia atribútov pre nastavenie *rangeToRemove* a opakovať proces pokiaľ sa fenomén už nevyskytne.

Optimalizácia hyperparametrov pre neurónové siete na nastavuje zvlášť pre každý typ neurónovej siete, pretože každý typ neurónovej siete má inú množinu parametrov. Taktiež zvlášť sa nastavuje aj počet epoch pre jednotlivé neurónové siete. Obrázok č. 54 zobrazuje nastavenie LSTM neurónovej siete s optimalizáciou hyperparametrov a s počtom epoch 100.



Obrázok 54 – Nastavenie LSTM neurónovej siete

Optimalizácia hyperparametrov pre ostatné neurónové siete RNN a DNN sa vykonáva podobne ako pre LSTM neurónovú sieť iba s inou množinou parametrov. Povolenie optimalizácie hyperparametrov a nastavenie epoch pre RNN a DNN je rovnaké ako pre LSTM neurónovú siet'.



# Príloha C: Článok na konferenciu IIT.SRC 2020

## Data analysis and security risk identification in computer networks

Bc. Zoltán CSENGÖDY\*

Slovak University of Technology in Bratislava  
Faculty of Informatics and Information Technologies  
Ilkovičová 2, 842 16 Bratislava, Slovakia  
[csengody4@gmail.com](mailto:csengody4@gmail.com)

**Abstract.** This work is devoted to research in the field of detection of computer attacks by machine learning methods. The aim of this paper is to create a program module, which in an appropriate way documents the selected machine learning algorithms. The main motivation is to create a unified analysis of the effects of the different settings of the classification algorithms and the different methods of pre-processing the selected datasets to the results of network attack detection. Part of this work is a custom design solution, which stems from the fact that the main drawback of using machine learning algorithms is insufficient documentation of the use, creation of architecture and parameter settings. In this work, we look for anomalies in network traffic and methods intended to detect them. Part of the work is also a suitable pre-processing of the selected dataset and revealing the dependencies between its attributes, which have a significant impact on the detection of attacks.

**Keywords:** artificial intelligence, data analysis, dataset, computer network, security, computer attack.

### 1 Introduction

With the development of Internet technologies, computer networks gradually change people's lives and increasingly facilitate the way people work. The development of this area is very fast, and therefore increasingly vulnerable to cyber-attacks. With the development of this technology area, new ways and types of attacks are coming.

Since the computer network can be open (freely available) and shared internationally, the data transmitted there is not secure. It is therefore necessary to introduce technical measures to ensure data protection in a network environment. Collection of data in a computer network can greatly help in detecting network attacks and assist in

---

\* Master study programme in field: Informatics  
Supervisor: Ing. Rudolf Grežo, Institute of Computer Engineering and Applied Informatics,  
Faculty of Informatics and Information Technologies STU in Bratislava

network management. By monitoring, testing, controlling and evaluating in real time, network administrators are able to obtain information about network system performance, evaluate quality of service (QoS), and detect network failure.

This problem needs to be solved in order to prevent malicious attacks through prediction based on data analysis. To address this issue, it is important to design, develop and implement security methods to prevent attacks. Early detection of malicious activity will provide better protection for the network from future trends in this area that bring new, complex and more sophisticated attacks. It also provides a cost-effective strategy in the case of a new attacks.

The first part of this work is devoted to the analysis of data from the computer network environment. In order to identify security risks and attack traffic, it is necessary to analyze existing algorithms and processes aimed at processing this type of data. Our aim is to analyze the current state of the problems and methods applicable to the analysis of data from the computer network environment.

In the second part of this work we deal with the design of our own solution. We will design two program modules. The first program module is designed to process large datasets and the second program module is designed to apply machine learning algorithms to the selected pre-processed dataset. Finally, we perform experimental tests and evaluate the success of classification of selected machine learning methods based on the results.

## 2 State of the art

Assuming that the attacker's behavior on the network is different from the normal behavior of the user, we can identify such network traffic as offensive. By examining anomalies in the network, it is possible to detect both known and unknown types of attacks. Network intrusion detection systems are implemented as a second line of defense in addition to user authentication and other security mechanisms.

In work of [2], authors argue that network anomalies are detectable through machine learning, which reveals two main areas / categories of anomalies' impact on the computer network. Anomalies according to [2] may affect network performance and security.

Our main goal is to address anomalies that cause security risks in the network, namely anomalies caused by malicious activities. Malicious network activities can have different types, such as point anomalies, contextual anomalies, or collective anomalies.

According to [4], most of the approaches focus on the problem of making the machine learning model explainable. [4] note, that many of the decisions that affect the model's predictive behavior are made during the data pre-processing phase and are encoded as a specific data transformation steps as part of pre-learning phase. Author of the article undertook a research into the impact of how dataset pre-processing methods can affect the outcome of the machine learning method prediction. In particular article the following open points are analyzed: understanding the sensitivity of outcomes of certain datapoints to different pre-processing techniques, detecting whether a pre-processing step is increasing or decreasing bias for particular vulnerable population groups

and increasing the transparency of the whole machine learning process by analysing any of the specific data pre-processing steps, thus enabling to create better models. Regarding to this problematic area, [13] also measures the impact of various pre-processing methods on the performance metrics for certain classifiers.

Experiments were performed on the UNSW-NB15 dataset. Moustafa and Slay [10] have designed a network forensic scheme where the first step intercepts network packets through sniffing and stores the intercepted packets in a database to facilitate the investigation of attacks. The second step selects important features and removes foreign and redundant information that could negatively affect the detection of attacks. Important features / attributes are selected using chi-square statistics. The third step examines attacks and their origins using a new correntropy-variation technique.

The proposed technique was compared with three state-of-the-art approaches: Filter-based Support Vector Machine, Multivariate Correlation Analysis and Artificial Immune System. By comparison, the network forensic scheme is better in terms of accuracy and frequency of false alarm. The forensic technique provides the best results by estimating the correntropy values for normal and test samples and subsequently identifying samples that are more than two standard deviations from the average of normal samples as attacks.

In our case, based on the detailed analysis, selected machine learning methods will be implemented on the selected data set - UNSW-NB15. Based on the official site [14] the UNSW-NB15 dataset is an unmodified network packet data file created by IXIA PerfectStorm at the Cyber Range laboratory at the Australian Cyber Security Center (ACCS). It was created to generate a common hybrid of real modern activities and synthetic simultaneous attack behavior.

The *tcpdump* tool is used to capture packet network traffic. This tool was used to capture 100 GB of raw data traffic. The UNSW-NB15 dataset has 9 types of attacks: *Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode* and *Worms*. The dataset consists of 2 540 044 records and 49 class-labeled attributes.

The following figure shows chunk of first rows of the selected dataset. First six rows represent normal traffic and the last row represents attack traffic categorized as *Exploits*. In the dataset normal traffic is labeled with label 0 and attack traffic with label 1.

	srcip	sport	dstip	dsport	proto	state	dur	abytes	dbytes	sttl	dttl	sloss	dloss	service	Siload	Dload	Spkts	Dpkts	swin
0	59.166.0.9	7045	149.171.126.7	25	tcp	FIN	0.201886	37552	3380	31	29	18	8	smtp	1.459438e+06	1.307669e+05	52	42	256
1	59.166.0.9	9685	149.171.126.2	80	tcp	FIN	5.864748	19410	1087890	31	29	2	370	http	2.640454e+04	1.481983e+06	364	746	256
2	59.166.0.2	1421	149.171.126.4	53	udp	CON	0.001391	146	178	31	29	0	0	dns	4.198418e+05	5.119820e+05	2	2	0
3	59.166.0.2	21553	149.171.126.2	25	tcp	FIN	0.053948	37812	3380	31	29	19	8	smtp	5.503374e+06	4.893601e+05	54	42	256
4	59.166.0.8	45212	149.171.126.4	53	udp	CON	0.000953	146	178	31	29	0	0	dns	6.128017e+05	7.471144e+05	2	2	0
5	59.166.0.0	59922	149.171.126.8	6881	tcp	FIN	8.631166	2500	1094768	31	29	38	390	-	2.3186429e+04	1.013311e+06	446	858	256
6	173.45.176.0	49562	149.171.126.12	80	tcp	FIN	0.189983	13304	268	254	252	6	1	http	5.291020e+05	9.439423e+03	18	6	256

Fig. 1. Sample data

### 3 Data pre-processing and analysis

According to Bhardwaj [1], raw data is highly sensitive to noise, missing values and inconsistent distribution. The quality of the data greatly affects the results of the classification methods. Data processing is one of the most critical step in the data mining process that deals with the preparation and transformation of the original dataset.

Methods for data pre-processing are as follows:

**Data cleaning** is a method based on adding missing values, smoothing data noise, identifying and removing outliers, and addressing irregularities.

Missing values are still found in the real-world datasets. Their correction is crucial because no model can handle missing data, often referred to as NULL or NaN. Removing entire columns or rows from a dataset can result in the loss of valuable data, and therefore such a method of processing missing values is not appropriate.

The following methods are used to add missing values: skip the tuple, if the class designation is missing; adding values manually; adding a global constant (eg "unknown"); adding the average of all values; adding a diameter to a particular class; adding the most likely value or predicting missing values based on existing values within a given attribute.

In case that if data has some noise then the following methods are used to remove noise from data: linear or multi-linear regression; outlier values can be identified by a combination of computer and human control or they can be detected by clustering, where similar values are grouped.

**Data transformation** activity transforms data into an appropriate form for data mining methods.

Data transformation include the following methods: normalization i. transform data to -1.0 - 1.0 or 0.0 - 1.0 (also used term of standardization), data noise removal, data aggregation or generalization of data.

**Data reduction** is a process of reducing the volume or dimensions (number of attributes) of a dataset. Working with a reduced dataset is more efficient.

The following approaches lists some data reduction methods: applying aggregation operations to the data in the data cube construction, removing of irrelevant or redundant attributes, reducing the size of the dataset by compression, such as a wavelet transformation, replacing or estimating data with smaller alternative data, such as parametric methods or non-parametric methods (clustering, sampling or using histograms) or generating discretization and conceptual hierarchy, where raw data values for attributes are replaced by ranges or higher conceptual levels.

**Categorical data processing** is a process when categorical values which take on discrete values, such as color are pre-processed as a numerical values.

Categorical values are subdivided into two subclasses like **ordinal categorical attributes** which values can be sorted, for example: large, medium, small and **nominal categorical attributes** which values cannot be sorted.

The following methods are used to transform categorical attributes to numeric:

**Mapping** what is a dictionary specifying the mapping of categorical values to numerical values, **Label encoding** what transforms categorical text data into numeric, understandable numeric data for the model and **One-Hot Encoding** what takes columns that contain categorical values and creates additional columns based on unique values for numerical representation of categorical values. Newly created columns have values of 0 or 1, depending on which row has the given categorical value.

**Data Enrichment** according to Hinton [5], data enrichment is defined as combining third-party data from an external authorized source with an existing database to improve data. Hinton further states that data enrichment has one key condition, namely the timeliness of the data, which leads to a 24/7 data enrichment process.

#### 4 Classification algorithms

Machine learning algorithms are generally used to detect anomalies. These algorithms create a detection model or a prediction model in the learning phase using a training algorithm on the training data. This prediction model is then tested on new data in the testing phase. The input data needs pre-processing to be readable by machine learning algorithms. The input data represents observations or records, and each record is represented by an attribute.

The following list by Jain and Bhupendra [6] lists some classification algorithms to detect network attacks:

**Random Forest** is one of the tree classification algorithms. The main goal of this algorithm is to increase tree classifiers based on the forest concept. Random forest classifiers have an accepted degree of accuracy and can be implemented to process noise values of a dataset. There is no re-modification process during classification. When implementing this algorithm, the number of trees in the forest should be estimated, because each individual tree within the forest predicts the expected output. Then the voting technique is used to select the expected output. Random forest is slow in training, is prone to overtraining and is too simple for complex problems.

**Decision tree** is a tree-like model with unit tree structures that represent decision files. These files generate the rules that are used to classify the data.

**Support Vector Machine** is a new generation of learning algorithms. It is used for classification and regression. SVM is at the forefront of machine learning through consistent mathematical foundations of optimization and statistical learning theory. According to [9], SVM separates classes using a hyper plan and uses class-marked data in the training phase as well as other supervised grading learning algorithms. Although SVM is a binary classifier, it can also be used to classify multiple classes. Two different methods are used to classify multiple classes, one-vs-all and one-vs-one.

**Logistic regression** according to the author of the Internet article Navlani [11] is a statistical method for predicting binary classes. The resulting or target variable is dichotomous, with dichotomy meaning that there are only two possible classes. In our case, this is normal (0) or offensive traffic (1).

**Multi-Layer Perceptron** maps a set of input data to a suitable set of outputs. It is a graph consisting of multiple layers, where each layer is fully attached to the next. MLP employs redistribution (generalization), a controlled teaching technique for network training. It can classify data that is not linearly separable.

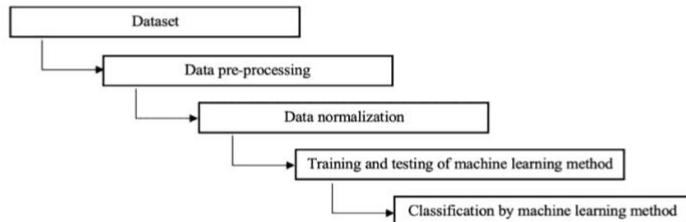
Authors Viet et. al [15] also used four out of our five machine learning classification algorithms which have been mentioned above.

## 5 Proposal

Based on a detailed analysis of the various dataset processing methods and classification algorithms mentioned in Chapter 4, which are commonly used to detect anomalous network traffic, we have decided to choose this direction that our work will take. The proposal of pre-processing dataset and then doing classification with selected algorithms is a widely used and standard approach for anomaly detection. Methods for data pre-processing are mentioned in Chapter 3, which will be used to pre-process the selected dataset. Methods used during the phase of data pre-processing are strongly dependent on the characteristics of the dataset, but in general the mentioned methods should fulfill the needs of dataset pre-processing before handing over to machine learning algorithms.

In the following chapters we will design two program modules. The first program module will serve to pre-process the dataset. The second program module will implement various machine learning methods to detect offensive / anomalous network traffic over the pre-processed dataset. This work will have a research character in terms of a deeper analysis of individual dataset pre-processing methods and machine learning algorithms, which have been discussed in detail in the individual chapter in this work. We implement machine learning methods over a selected dataset in order to compare the evaluation results and evaluate which method is best suited for what type of network intrusion issue. We will focus mainly on the detection of anomalies in network traffic and on methods that are designed to detect them. A major problem and drawback in this area is the appropriate setting of machine learning methods. Therefore, our work will be devoted to the research of appropriate parameter settings for selected methods.

In the following figure you can see the process of dataset processing and network attack classification.



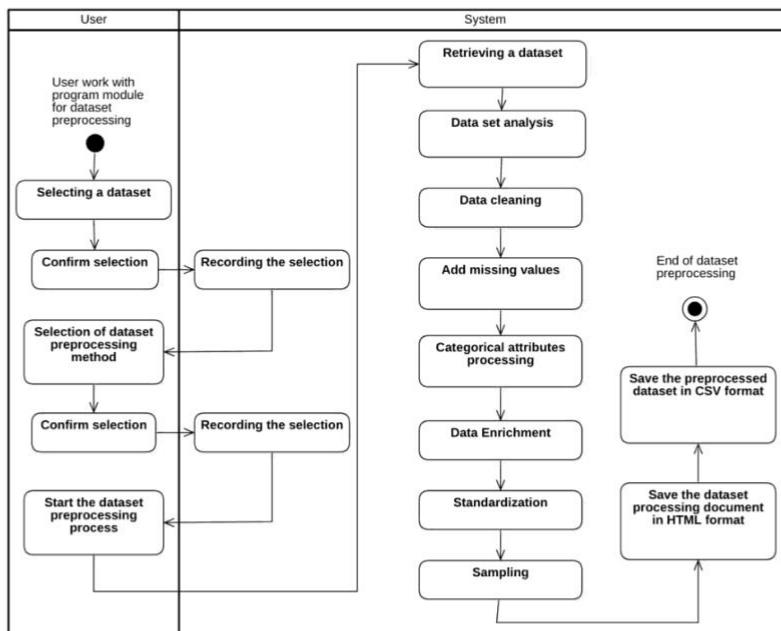
**Fig. 2.** Network Attack Classification Process

## 6 Dataset pre-processing module

The program module contains a sequence of steps necessary to prepare the data that will be later an input for the machine learning method in the form of a training, validation and test subset.

The user initiates the pre-processing by selecting a dataset and pre-processing method. When selecting the pre-processing method, the type of method to be applied to that pre-processing step is determined. Some pre-processing steps may be omitted. This does not apply to the steps necessary to pre-process the dataset to an acceptable form by the machine learning method. The system then performs pre-processing of the selected dataset by a sequence of pre-processing steps.

Activity diagram in figure no. 3 shows a process for processing selected dataset.



**Fig. 3.** Activity diagram of dataset pre-processing program module

### 6.1 Retrieving the dataset

The dataset of CSV format is loaded into the *DataFrame* data structure. Typically, datasets do not contain a header that represents the names of each attribute / column, and therefore it is necessary to read the attributes separately and set them as the dataset header.

## 6.2 Dataset analysis

Using the *describe* method from *pandas.DataFrame* library we display descriptive statistics of a dataset that summarize the central tendency, variance and shape of the distribution. This method does not take into account nominal attributes and values with NaN. From the description we can read the number of records, arithmetic mean, standard deviation, minimum, maximum value and quarters, with the fifty percent quarter being the median. This description is useful for detecting outliers.

Another method for data analysis is the correlation matrix, which in color and numeric determines the degree of correlation of attributes. Each cell in the table shows the correlation between the two attributes. The correlation matrix is mainly used to summarize data as an input to more advanced analysis. To be able to properly apply data cleaning methods in the next step, we first need to display the number of unique values for each attribute. This method reveals what values each attribute takes and how many. This approach is mostly applied to nominal attributes.

## 6.3 Cleaning the data

During the phase of data cleaning, we concluded that there were uncategorized denominations in the dataset that we either deleted or renamed. Furthermore, we found that the source and destination port values take a hexadecimal shape. We have converted these values to decimal. Regarding port numbers, we have found that some values exceed the dynamic port range, the maximum value of which is 65 535. There were only a few of these records and we could delete them based on a thorough analysis. Other attributes also included outliers, but after analyzing connection types and whether normal or offensive traffic, we concluded that outliers in certain attributes were high in anomalous network traffic, so we could not delete these records because of the reduction of the already low proportion of anomalous traffic to an even lower number.

## 6.4 Adding missing values

To add missing numerical values, we decided to use regression. We used the *RandomForestRegressor* method from *sklearn.ensemble* library because it uses averaging to improve prediction accuracy and has control over overtraining. Before we started training Random Forest Regressor, we optimized hyperparameters using two methods: *RandomizedSearch* and *GridSearch* from *sklearn.model\_selection* library. According to Senapatia [12], Random Search is a technique where random combinations of selected hyperparameters are used to find the best solution for a built model. In further states that the chance of finding an optimal hyperparameter in a random search is relatively higher due to a random search pattern. A matrix search versus a random hyperparameter search will try each combination of a default list of hyperparameter values and evaluate the model for each combination. After evaluating all combinations of hyperparameters, the best model is considered to have the highest accuracy. Our solution is to combine these two methods.

## 6.5 Pre-processing of categorical attributes

We decided to process categorical attributes in two ways. The first is to assign static numeric values to each categorical value. This is called mapping, when a dictionary is created that accurately maps categorical values to numerical values. Such a dictionary has the following form: `{'ftp': 1, 'smtp': 2, 'dns': 3, ...}`.

The second way to pre-process categorical attributes is the *One-Hot Encoding* method using `get_dummies` method from `pandas` library. Using appropriate method, we transformed the nominal attributes to numeric.

## 6.6 Data enrichment

We have transformed the nominal attribute of the source and destination IP addresses into another nominal attribute, which represents the country of the IP address using the GeoLite2 database. Using One-Hot Encoding for IP addresses would make no sense because the range of IPv4 addresses is around 4.3 billion. Using the `geolite2` database library, we called a method to get the country code (ISO code) to which the IP address belongs. In the case of IP addresses that are not in the database, we have treated one of the flags: "Private", "Localhost", "Multicast" according to the range of IP addresses. Subsequently, the process of processing categorical attributes from the chapter above applies. This process is time-consuming in the case of transforming IP addresses of more than hundred thousand.

We have also enriched the dataset with two additional attributes that describe the port type. Based on the range of each port number, we can determine whether it is a well-known port, registered or private. Subsequently, the process of processing categorical attributes from the chapter above applies.

## 6.7 Standardization

For the standardization process we used the `StandardScaler` method from `sklearn.preprocessing` library, which normalizes the properties by removing the mean value and scaling the values according to the variance of the units. According to the author [7], not every dataset requires values to be normalized, this process is only necessary if the dataset has different ranges of data values. In other words, standardization is a systematic way to ensure that the data structure is suitable for general lookup without certain undesirable characteristics. Once standardized, the machine learning model is better able to "understand" data and achieve better results.

## 6.8 Data sampling

Data resampling is for purpose to better represent the minority classes so the classifier would have more samples to learn from (oversampling) or less samples to better differentiate minority class samples from the majority (undersampling).

The sampling process is implemented by two methods: *SMOTE* and *ADASYN* from `imblearn.over_sampling` library. These methods are interconnected with a pipeline. A

pipeline is used to use the output of one method as an input for another method. According to Lemaitre et al. [8] The standard *RandomOverSampler* method, compared to the *SMOTE* and *ADASYN* methods, samples by duplicating some of the original minority class samples. *SMOTE* and *ADASYN* generate new samples by interpolation. However, the samples used to interpolate / generate new synthetic samples vary. *ADASYN* focuses on creating samples next to original samples that are incorrectly classified using the *K-Nearest Neighbors* (KNN) classifier. While *SMOTE* does not distinguish between samples to be classified using the nearest neighbor rule.

## 7 Machine learning program module

Activity diagram for network attack detection using machine learning methods. In the figure below illustrates the process of applying machine learning methods to a set of pre-processed data that is an output from a dataset pre-processing program module.

The user initiates the process by selecting the pre-processed dataset and machine learning methods. In the case of several machine learning methods, they are compared after training the machine learning model. The output is a summary of machine learning methods, based on which we can easily deduce which method is the best. Next, a set of steps is required to prepare the pre-processed dataset for the model. It is important to note the hyperparameter optimization activity before training the model. This step is key to achieving better results in detecting anomalous network traffic. By tuning hyperparameters we can control the overall behavior of the machine learning model. Setting the optimal combination of parameters for machine learning model can lead to minimizing loss and achieving better results. In other words, hyperparameter tuning leads to the fact that the model can optimally solve the machine learning problem.

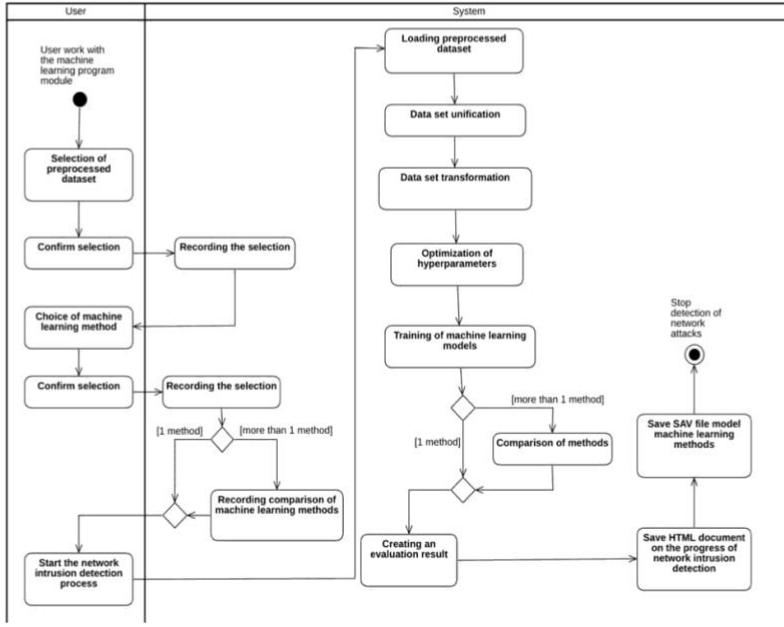


Fig. 4. Activity diagram of machine learning program module

### 7.1 Loading, unification and transformation of pre-processed dataset

The pre-processed dataset in CSV format is read into the *Set* data structure, with the individual items of the set representing the pre-processed dataset itself, which is stored in the *DataFrame* data structure. The pre-processed datasets are stored with the headers, so there is no need to separately read and set them as during data pre-processing phase.

If a user has loaded more than one dataset with the same pre-processing method and attributes, then these datasets can be merged, the choice is up to the user. It is important to note that if large pre-processed datasets are merged, RAM memory may be exhausted. Memory depletion can usually occur during the process of selecting hyperparameters or while training a machine learning model.

The pre-processed dataset is transformed into smaller datasets by design. By default, the *train\_test\_split* method of the *sklearn.model\_selection* library is used. This method determines the size of the validation sample as a percentage. The return value of the method is a training and validation sample of the data. To get a test sample of data, we need to apply the method twice. The samples are divided in the ratio of training sample: 60%, validation sample: 20%, test sample: 20%.

## 7.2 Training and evaluation of machine learning model

We chose the following intelligent machine learning methods: logistic regression and SGD classifier both from `sklearn.linear_model` library, decision tree from `sklearn.tree` library and random forest from `sklearn.ensemble` library. In addition to the SGD classifier, all of the above-mentioned machine learning methods are described in Chapter 4. The SGD (Stochastic Gradient Descent) classifier is also a linear classifier. It is a combination of SVM (binary classifier) and logistic regression with SGD training. Logistic regression has been used because it is a statistical method for predicting binary classes as well as normal and attack traffic is labeled in the selected dataset. Random forest has been chosen because in general provides high accuracy for classification tasks, is resistant to overfitting and is able to handle a large set of data with high dimensionality. We decided to use decision tree because of its robustness. It can handle non-scaled data and in general does not require less effort for data pre-processing.

We used the *RFE* (Recursive Feature Elimination) method from `sklearn.feature_selection` library to select a suitable set of attributes for logistic regression, which creates a ranking of individual attributes with recursive removal. We used a linear regression model for the estimator parameter. The method returns a list of attributes with the appropriate rating. Attributes whose p-value is less than 0.05 are selected. By convention p-value / probability value is commonly set to 0.05 in statistical hypothesis testing. For the SGD classifier, we used the *Nystroem* method from `sklearn.kernel_approximation` library, which constructs an approximate attribute map for any kernel using a subset of data as the basis. For decision tree and random forest methods, we used the *SelectFromModel* method from `sklearn.feature_selection` library, which is a meta-transformer for selecting the best attributes based on important weights, as a method for selecting the best attributes.

We used a combination of *RandomizedSearch* and *GridSearch* to select the optimal hyperparameters. The SGD classifier model uses the *bestFit* method from the `parfit` library [3], which is designed to parallel model learning and flexible machine learning model evaluation.

After selecting the best attributes of the pre-processed dataset, it is necessary to transform the training, validation, and test data sample based on the best attributes.

We decided to use the following metrics to evaluate the model's success:

- Accuracy, precision, recall, F1-score, support
- Standard 10-fold cross-validation
- Stratified 10-fold K-fold cross validation
- Confusion matrix
- Receiver operating characteristic curve (ROC) – Area under curve (AUC)

## 8 Evaluation

Based on pre-processed datasets, we applied selected methods of machine learning. The machine learning program module evaluated selected machine learning methods according to the tables below. Values are given for accuracy and F1-scores in percent for the test sample of data.

**Table 1.** Results of experimental tests – accuracy

Dataset	Logistic regression	SGD classifier	Decision tree	Random forest
dataset 4 MAP TOPF F	98.43	98.02	99.25	99.44
dataset 4 MAP TOPF T	98.34	<b>79.66</b>	98.85	98.95
dataset 4 MAP RESAMP_TOPF_F	98.90	98.65	99.40	99.67
dataset 4 MAP RESAMP_TOPF_T	98.85	<b>50.09</b>	98.93	99.26
dataset 4 OHE TOPF T	-	<b>72.55</b>	99.48	99.41
dataset 4 OHE RESAMP_TOPF_T	-	<b>49.56</b>	98.91	99.50

**Table 2.** Results of experimental tests – F1-score

Dataset	Logistic regression	SGD classifier	Decision tree	Random forest
dataset 4 MAP TOPF F	96	95	98	99
dataset 4 MAP TOPF T	96	<b>0</b>	97	98
dataset 4 MAP RESAMP_TOPF_F	99	99	99	100
dataset 4 MAP RESAMP_TOPF_T	99	<b>67</b>	99	99
dataset 4 OHE TOPF T	-	<b>10</b>	99	99
dataset 4 OHE RESAMP_TOPF_T	-	<b>63</b>	99	99

Table Explanation:

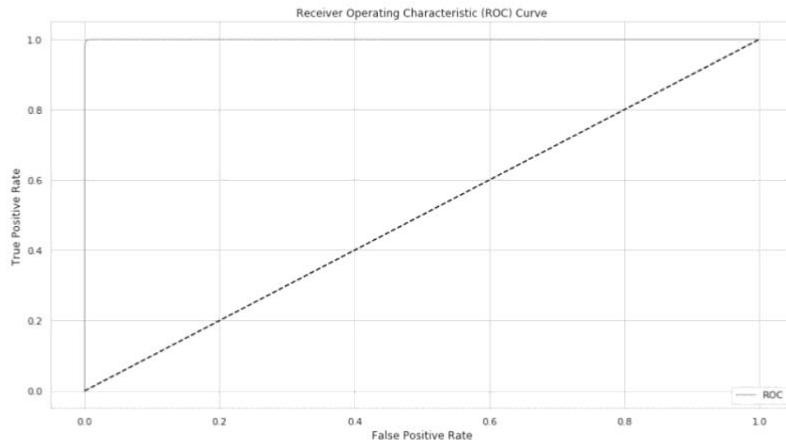
- MAP in the dataset name represents mapping and OHE represents One-Hot Encoding of categorical attributes. MAP datasets have 50 attributes and OHE datasets have 218 attributes.
- RESAMP in the dataset name indicates that the dataset is resampled. The sampled datasets have a record count of 702 296 over an unsampled dataset with a record count of 440 042.
- TOPF\_T / F in the dataset name represents a selection of the best attributes. For T (true) the best attributes were selected and for F (false) they were not selected.
- Bold values represent observations which have a large variation in the model's accuracy against the F1-score.
- Empty values were caused by the multicollinearity phenomenon

Based on experimental tests, it can be concluded that the most reliable method of machine learning is random forest, as it achieved very similar, constant and high accuracy

and F1-scores in each experimental test. The decision tree, compared to a random forest, reached a slightly less accurate classification and F1-score. The third correct classification method is logistic regression. In the case of pre-processed categorical attributes using the One-Hot Encoding method, we encountered the phenomenon of multicollinearity. Therefore, we were unable to determine the correctness and F1-scores for this method despite applying the gradual sampling method. The worst results were achieved by the SGD classifier. The results of this machine learning method have great fluctuations in the model's accuracy against the F1-score, and thus we can conclude that the model is not reliable.

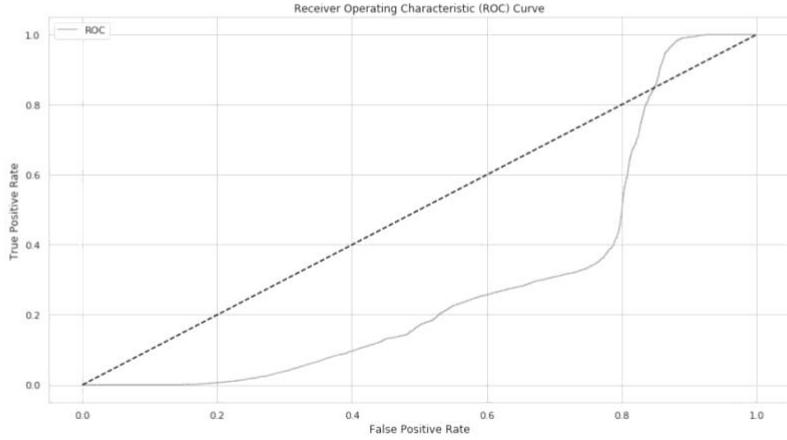
We also found that selecting the best attributes results in a slight decrease in the accuracy of the model evaluation. Another finding is that there is not much difference between the different dataset pre-processing methods. Machine learning that was taught on a dataset by a pre-processed resampling method performed better.

In the following figure no. 5 is the ROC curve for the best random forest model with a 99.67% accuracy, an F1-score of 100% and an AUC of 99.99%.



**Fig. 5.** ROC curve of the best random forest model

In the following figure no. 6 you can see the ROC curve of the SGD classifier with a success rate of 79.66%, an F1-score of 0% and an AUC of 29.01%. Compared to a random forest, it can be concluded that the SGD classifier model in this case could not recognize normal network traffic from anomalous.



**Fig. 6.** ROC curve of SGD classifier

## 9 Conclusion and future work

We designed two program modules. The first program module is designed to process large datasets and the second program module is designed to apply machine learning methods to a selected pre-processed dataset.

We have also described in detail how to implement individual parts of the program modules. We managed to download a large dataset NUSW-NB15, over which we applied various methods of dataset pre-processing. We were able to pre-process datasets in different ways. We applied machine learning methods over pre-processed datasets, namely: logistic regression, linear classifier (SVM, logistic regression) with SGD training, random forest and decision tree.

In the last part of this work we evaluated the proposed solution. We performed experimental tests and based on the results of evaluation of the success of classification of selected machine learning methods we chose the best machine learning method. We have achieved high accuracy detection rates for network attacks. The best method based on the results of the experimental tests is a random forest, as it achieved very similar, constant and high accuracy values in each experimental test. Based on experimental tests, we also conclude that machine learning methods like logistic regression and SGD classifier are not suitable for certain types of dataset pre-processing.

Our work will continue to complete the proposed program modules. We also plan to implement a neural network to detect network attacks. We see a great challenge in understanding the concept of neural networks, their architecture, optimizing hyperparameters and interpreting the results. Our next goal is to combine the different machine learning methods mentioned in this work. In conclusion, we would like to verify the success of evaluating the best machine learning model on simulated network traffic.

## References

1. BHARDWAJ, Anshu. Data Preprocessing Techniques for Data Mining. Data Mining Techniques and Tools for Knowledge Discovery in Agricultural Datasets [online]. New Delhi: Division of Computer Applications Indian Agricultural Statistics Research Institute (ICAR), s. 139-144 [cit. 2019-05-22]. Available from: [http://iasri.res.in/ebook/win\\_school\\_aa/notes/Data\\_Preprocessing.pdf](http://iasri.res.in/ebook/win_school_aa/notes/Data_Preprocessing.pdf)
2. BHATTACHARYYA, Dhruba Kumar a Jugal Kumar KALITA. Network Anomaly Detection [online]. Chapman and Hall/CRC, 2013 [cit. 2019-05-12]. DOI: 10.1201/b15088. ISBN 9780429166877.
3. CARPENTER, Jason. Parfit — quick and powerful hyper-parameter optimization with visualizations. Medium: ML Review [online]. 2017, 26.11.2017 [cit. 2019-12-07]. Dostupné z: <https://medium.com/mlreview/parfit-hyper-parameter-optimization-77253e7e175e>
4. GONZALEZ ZELAYA, Carlos Vladimiro. Towards Explaining the Effects of Data Preprocessing on Machine Learning. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE) [online]. IEEE, 2019, 2019, s. 2086-2090 [cit. 2020-04-16]. DOI: 10.1109/ICDE.2019.00245. ISBN 978-1-5386-7474-1. Dostupné z: <https://ieeexplore.ieee.org/document/8731532>
5. HINTON, Todd. Introduction to Data Preprocessing in Machine Learning. Redpoint Global [online]. 2018, 10.7.2018 [cit. 2019-12-03]. Available from: <https://www.redpointglobal.com/blog/what-is-data-enrichment/?fbclid=IwAR3lOtF1cBM08sGB8yzgj3evwzK00Nj8Af03oMttUm-bpv9pd4Fh2J0tIZHA>
6. JAIN, Anurag; BHUPENDRA, L. Classifier selection models for intrusion detection system (IDS). Informatics Engineering, an International Journal (IEIJ), 2016, 4.1: 1-11.v
7. JAITLEY, Urvashi. Why Data Normalization is necessary for Machine Learning models. Medium [online]. 2018, 8.10.2018 [cit. 2019-12-04]. Available from: <https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029>
8. LEMAITRE, G., F. NOGUEIRA, D. OLIVEIRA a C. ARIDAS. Over-sampling. Imbalanced-learn [online]. 2017 [cit. 2019-12-06]. Available from: [https://imbalanced-learn.readthedocs.io/en/stable/over\\_sampling.html](https://imbalanced-learn.readthedocs.io/en/stable/over_sampling.html)
9. MEHMOOD, Tahir a Helmi B Md RAIS. Machine learning algorithms in context of intrusion detection. In: 2016 3rd International Conference on Computer and Information Sciences (ICCOINS) [online]. IEEE, 2016, 2016, s. 369-373 [cit. 2019-05-13]. DOI: 10.1109/ICCOINS.2016.7783243. ISBN 978-1-5090-2549-7. Available from: <http://ieeexplore.ieee.org/document/7783243/>
10. MOUSTAFA N., SLAY J. (2018) A Network Forensic Scheme Using Correntropy-Variation for Attack Detection. In: Peterson G., Shenoi S. (eds) Advances in Digital Forensics XIV. DigitalForensics 2018. IFIP Advances in Information and Communication Technology, vol 532. Springer, Cham
11. NAVLANI, Avinash. Understanding Logistic Regression in Python. DataCamp [online]. 2018, 7.9.2018 [cit. 2019-12-03]. Available from: <https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python>
12. SENAPATI, Deepak. Grid Search vs Random Search. Medium [online]. 2018, 29.8.2018 [cit. 2019-12-06]. Available from: <https://medium.com/@senapati.dipak97/grid-search-vs-random-search-d34c92946318>

13. S. F. Crone, S. Lessmann, and R. Stahlbock, "The impact of pre- processing on data mining: An evaluation of classifier sensitivity in direct marketing," European Journal of Operational Research, vol. 173, no. 3, pp. 781–800, 2006.
14. The UNSW-NB15 Dataset Description [online]. 2018 [cit. 2019-05-14]. Dostupné z: <https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/>
15. Viet H.N., Van Q.N. , Trang L.L.T.,Nathan S. :Using Deep Learning Model for Network Scanning Detection, ICFET '18, June 25–27, page no, 117,ACM (2018)



# Príloha D: Opis digitálnej časti práce

Evidenčné číslo práce v informačnom systéme: FIIT-182905-74582

Obsah digitálnej časti práce (archív ZIP):

- \DiplomovaPraca – Elektronická verzia diplomovej práce vo formáte .docx a .pdf
  - xcsengody.docx
  - xcsengody.pdf
- \IITSRC – Adresár s článkom na konferenciu IIT.SRC 2020. Adresár obsahuje súbory vo formáte .docx a .pdf
  - xcsengodyIITSRC.docm
  - xcsengodyIITSRC.pdf
- \Notebooks – Projektový adresár programových modulov pre predspracovanie dátovej množiny a strojové učenie. Adresár obsahuje súbory vo formáte .ipynb
  - DatasetPreprocessing.ipynb
  - MachineLearning.ipynb
  - \lib – Adresár vlastných knižníc. Adresár obsahuje súbory vo formáte .py
    - Config.py
    - DatasetPreprocessingController.py
    - MachineLearningController.py
- \Prilohy – Adresár s prílohmi
  - \KorelacneMatice – Adresár s korelačnými maticami. Adresár obsahuje obrázky vo formáte .png
    - KM\_UNSW-NB15\_3-4\_MAP\_RESAMP\_SIM.png
    - KM\_UNSW-NB15\_3-4\_MAP\_RESAMP\_SIM\_2.png
    - KM\_UNSW-NB15\_3-4\_MAP\_SIM.png
    - KM\_UNSW-NB15\_3-4\_MAP\_SIM\_2.png
    - KM\_UNSW-NB15\_3-4\_OHE\_RESAMP\_SIM.png
    - KM\_UNSW-NB15\_3-4\_OHE\_RESAMP\_SIM\_2.png
    - KM\_UNSW-NB15\_3-4\_OHE\_SIM.png
    - KM\_UNSW-NB15\_3-4\_OHE\_SIM\_2.png
  - \Spravy – Adresár s výstupnými správami pre programové moduly vo formáte .html
    - dataset\_3\_MAP\_2020-ProfileReport.html
    - dataset\_3\_MAP\_2020.html

- UNSW-NB15\_3-4\_MAP\_SIM.html
- \Modely – Adresár s modelmi strojového učenia vo formáte .sav, .h5 a .txt
  - UNSW-NB15\_3-4\_MAP\_SIM\_DecisionTree.sav
  - UNSW-NB15\_3-4\_MAP\_SIM\_DNN.h5
  - UNSW-NB15\_3-4\_MAP\_SIM\_LogisticRegression.sav
  - UNSW-NB15\_3-4\_MAP\_SIM\_LSTM.h5
  - UNSW-NB15\_3-4\_MAP\_SIM\_Perceptron.sav
  - UNSW-NB15\_3-4\_MAP\_SIM\_RandomForest.sav
  - UNSW-NB15\_3-4\_MAP\_SIM\_RNN.h5
  - UNSW-NB15\_3-4\_MAP\_SIM\_SGDClassifier.sav
  - UNSW-NB15\_3-4\_MAP\_SIM\_XGradientBoosting.sav
  - UNSW-NB15\_3-4\_MAP\_SIM\_Models.txt
- \GeoLiteDatabaza – Adresár s geolokačnou databázou IP adries vo formáte .mmdb
  - GeoLite2-Country.mmdb

Názov odovzdaného archívu: DP\_prilohy\_digital\_ZoltanCsengody.zip