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	**Cieľ**
1-2	Analýza existujúcich riešení IDS (štruktúra, metódy, algoritmy).
3-5	Analýza metód spracovania veľkej množiny dát z prostredia počítačových sietí.
6-8	Analýza a zaúčanie sa do strojového učenia.
9-10	Odskúšanie získaných znalostí na vzorových príkladoch. Experimentovanie.
11-12	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 sieťových útokov a spôsob vyhodnocovania úspešnosti IDS. Taktiež sa nám podarilo dostatočne zanalyzovať sieťové 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	**Cieľ**
1-2	Zbieranie dát z prostredia počítačových sietí a ich predpríprava.
3-4	Návrh vlastného riešenia.
5-8	Implementácia zadania.
9-10	Analýza výsledkov vlastného riešenia. Odhalenie nedostatkov. Zapracovanie zmien do implementácie.
11-12	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, konkrétne logistická regresia, lineárny klasifikátor (SVM, logistická regresia) s výcvikom SGD, náhodný les a rozhodovací strom. Dosiahli sme vysoké miery presností odhalenia sieťových útokov. Počas vývoja prototypu sme čelili 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	**Cieľ**
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 odhaľovanie 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.

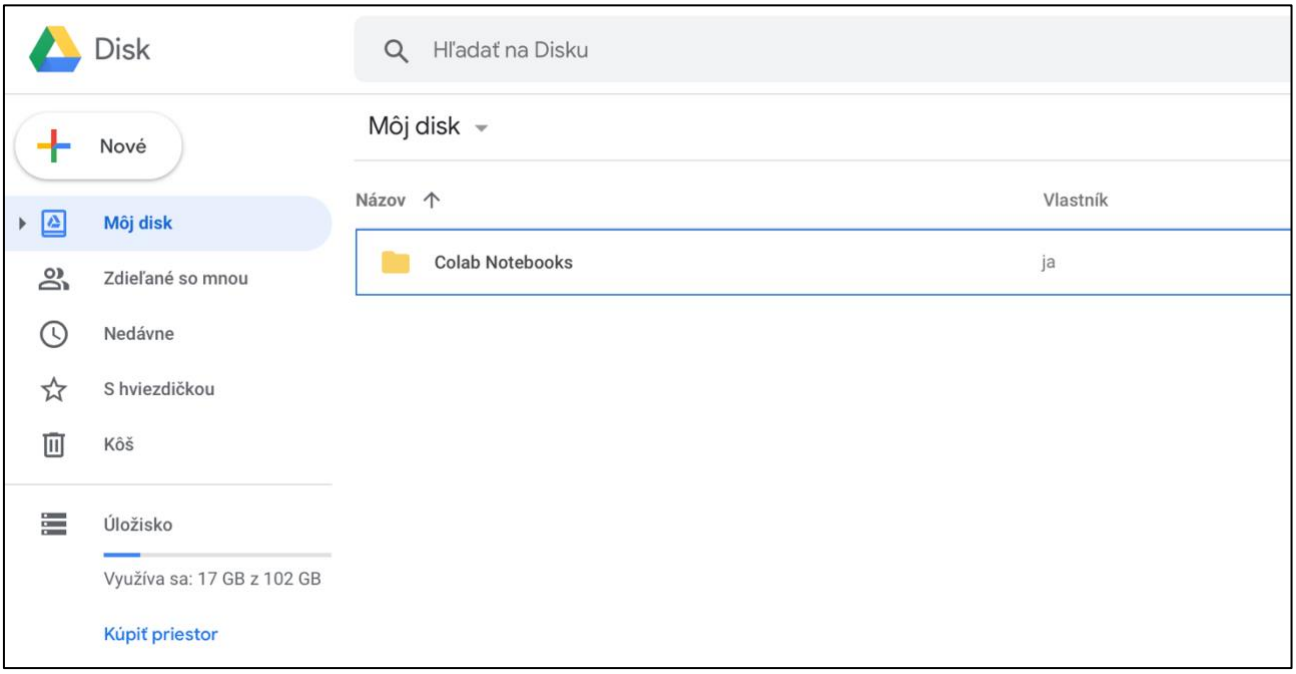
Príloha B: Technická dokumentácia

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 cloudová služba, ktorá integruje prostredie Jupyter Notebook.

Nasledujúci obrázok zobrazuje prostredie Google Drive.

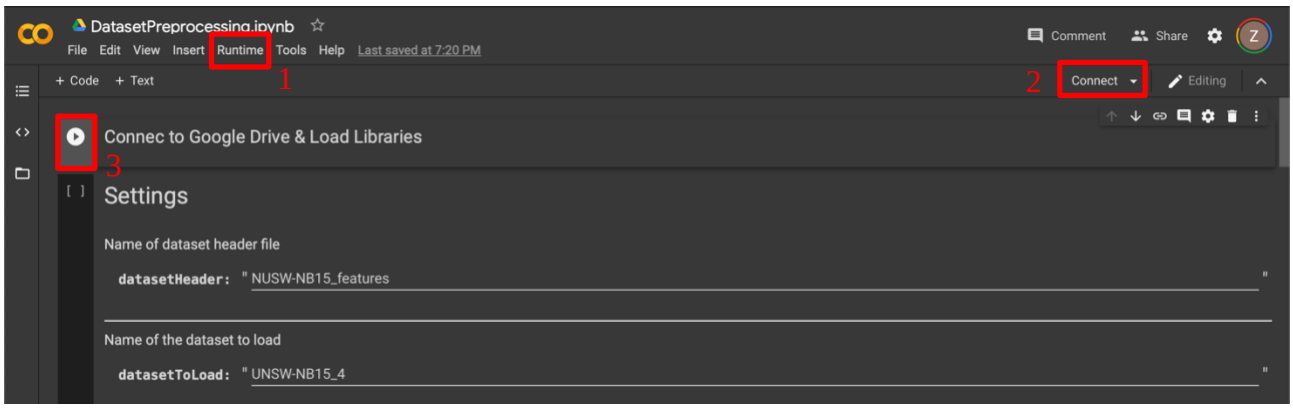


Obrázok 41 – Google Drive úložný priestor

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.

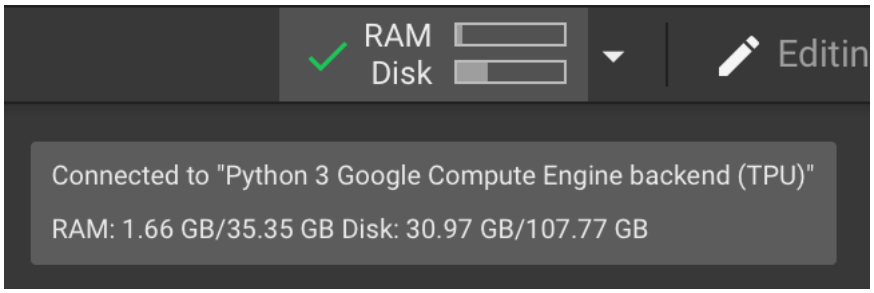


Obrázok 42 – Prostredie Google Colab

V prostredí Google Colab je potrebné nastaviť v časti *Runtime/Change runtime type* (viď. 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 hosťovanému počítačovému modulu. Obrázok nižšie znázorňuje úspešné pripojenie k počítačovému modulu od spoločnosti Google.

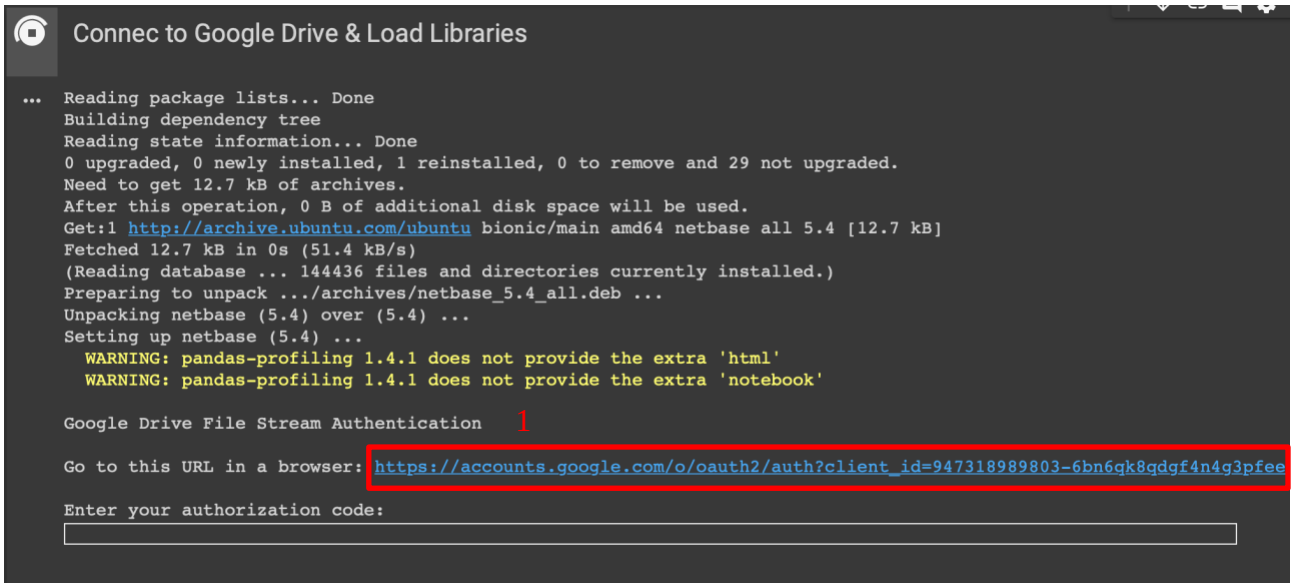


Obrázok 43 – Počítačový modul Google

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.

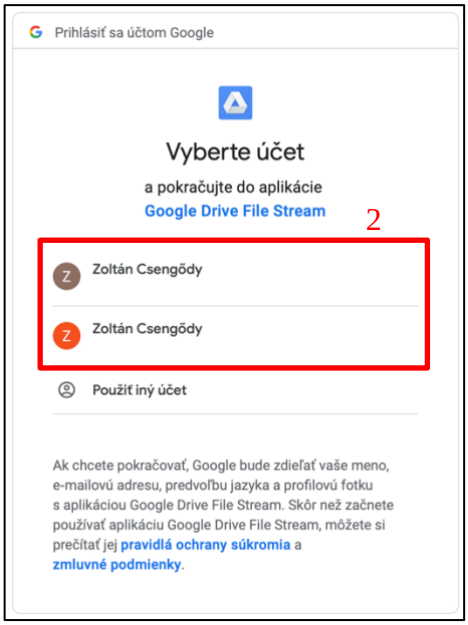
Jednotlivé bunky notebooku je možné spúšťať samostatne kliknutím na tlačidlo 3 vyznačené na obrázku č. 42, alebo spustiť všetky bunky naraz cez *Runtime/Run all*. Ak chceme zastaviť bežiaci proces (bunku) tak v časti *Runtime/Interrupt execution* vyvoláme prerušenie vykonávania bežiaceho procesu. Prerušenie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



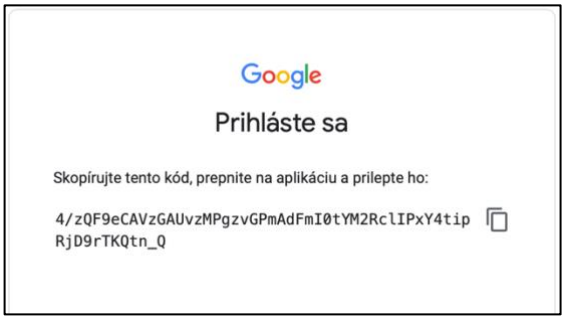
Obrázok 44 – Autentifikácia – krok 1

Používateľ musí vybrať jeden zo svojich Google účtov.



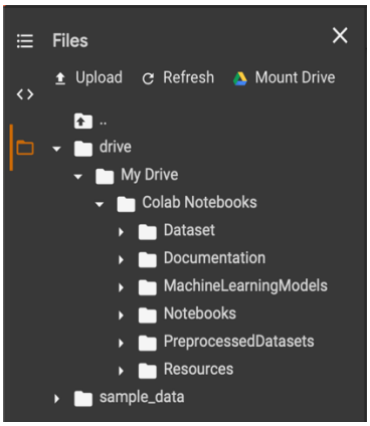
Obrázok 45 – Autentifikácia – krok 2

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.



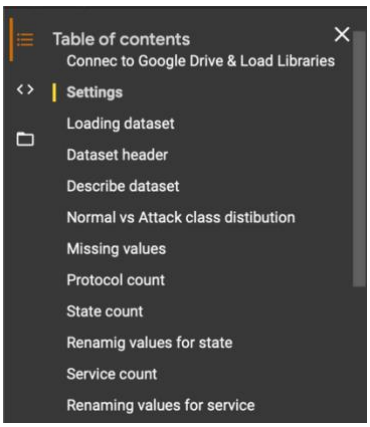
Obrázok 46 – Autentifikácia – krok 3

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.



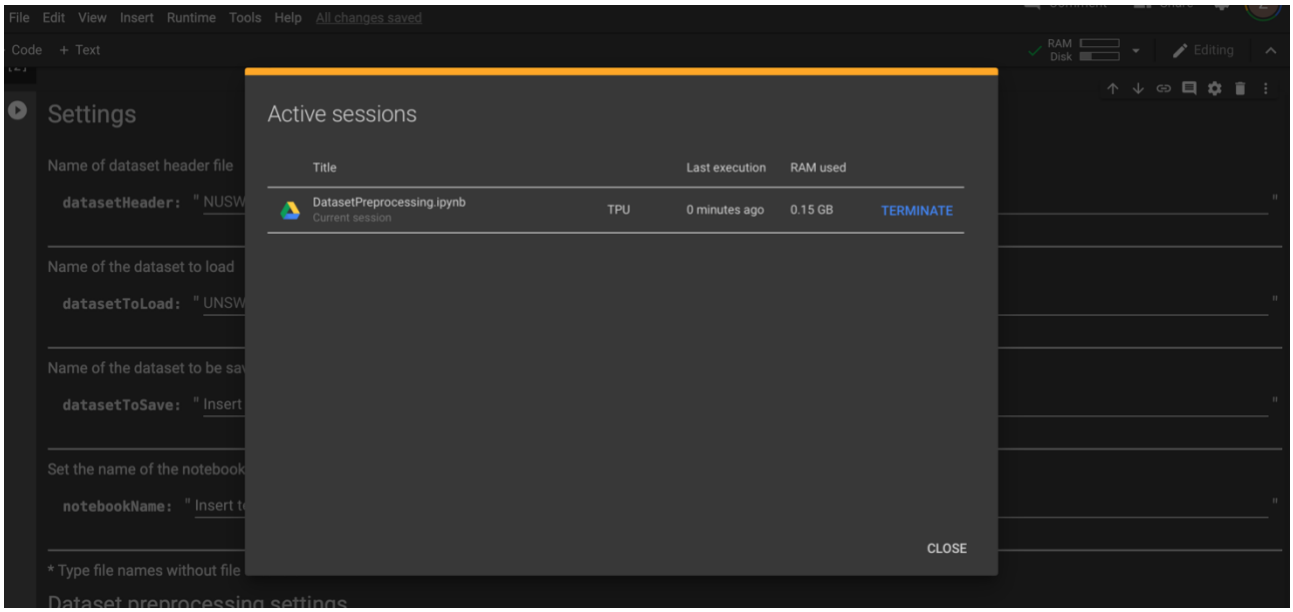
Obrázok 47 – Google Drive adresárová štruktúra v prostredí Google Colab

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

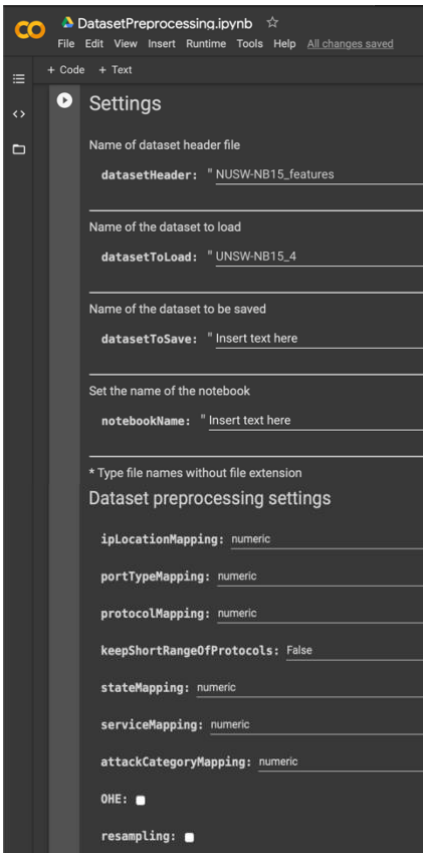
Ak sa používateľ pripojí k počítačovému modulu cez jeden programový modul v prostredí Google Colab a chce sa pripojiť aj s druhým programovým modulom k počítačovému modulu, tak túto možnosť Google nepodporuje. Používateľský účet môže byť pripojený naraz iba k jednému počítačovému modulu. Ak sa používateľ 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

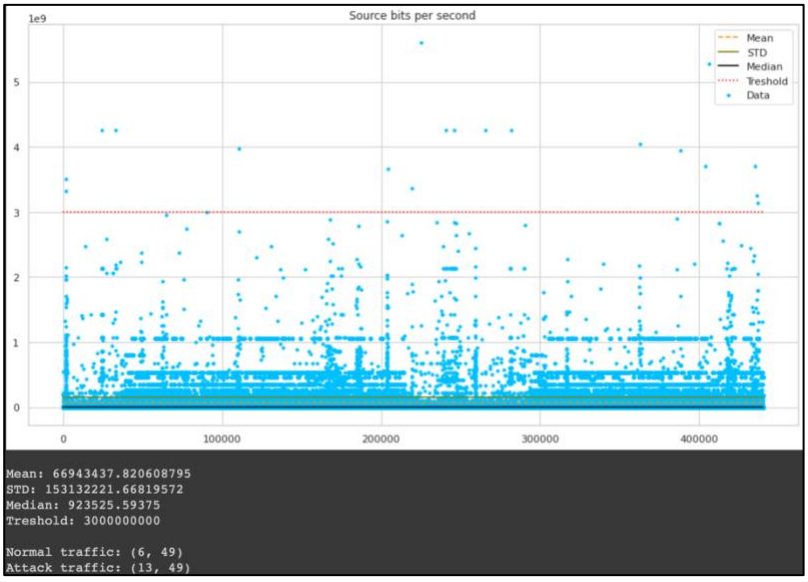
Vysvetlenie nastavení:

- **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 cieľovej 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*
- **arrackCategoryMapping** – 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*, *dttl*, *sload*, *dload*, *spkts*, *dpkts*, *smeansz*, *dmeansz*, *trans_depth*, *sintpkt* a *dintpkt* používateľ môže zvoliť prahovú hodnotu (treshold), nad ktorou sa má spočítať distribúcia normálnej a útočnej sieťovej premávky. Pomocou prahovej hodnoty vieme zistiť zastúpenie útočnej sieťovej premávky voči normálnej. Prahová hodnota sa zobrazí na grafe distribúcie hodnôt pre daný atribút ako červená trhaná oddeľovacia horizontálna čiara. Nasledujúci obrázok znázorňuje takýto príklad.

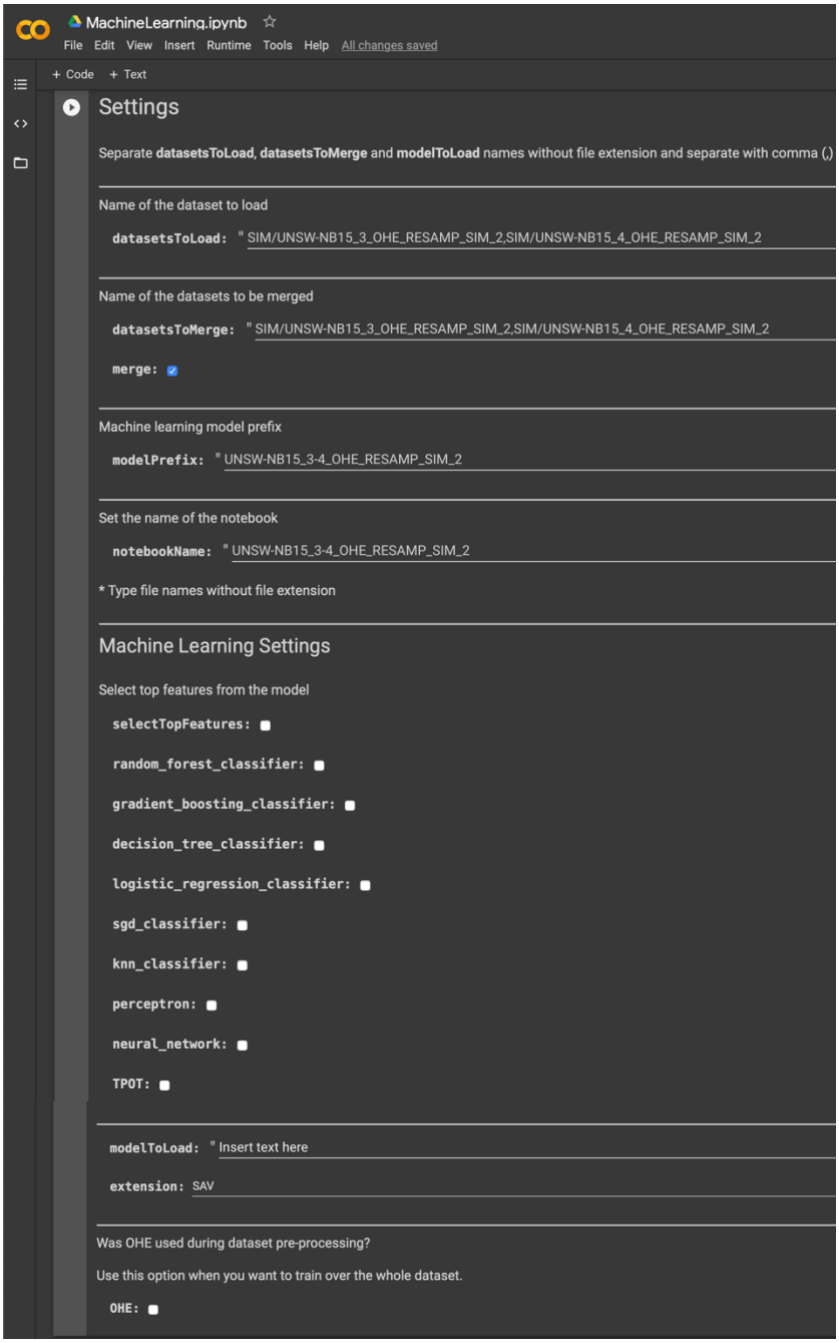


Obrázok 51 – Prahová hodnota - treshold

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

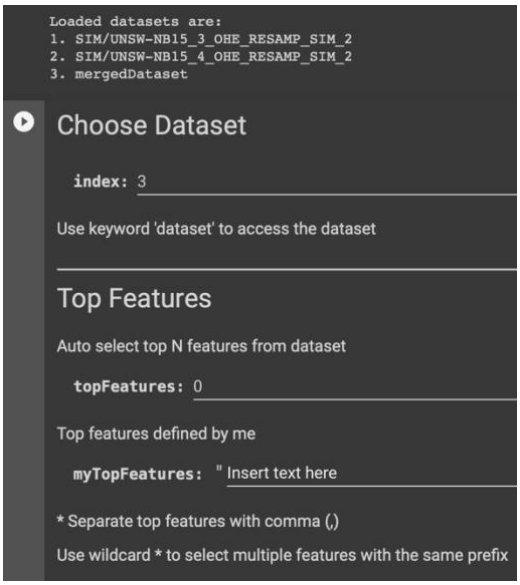


Obrázok 52 – Nastavenie programového modulu pre strojové učenie

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 sieť
        - Long short-term memory (LSTM) neurónová sieť
        - Rekurentná neurónová sieť (Recurrent Neural Network – RNN)
        - Hlboká neurónová sieť (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 krížových 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 zadaním hodnoty indexu (viď. obrázok č. 53) zvolenej dátovej množiny vyberie dátovú množinu pre ďalšie použitie.



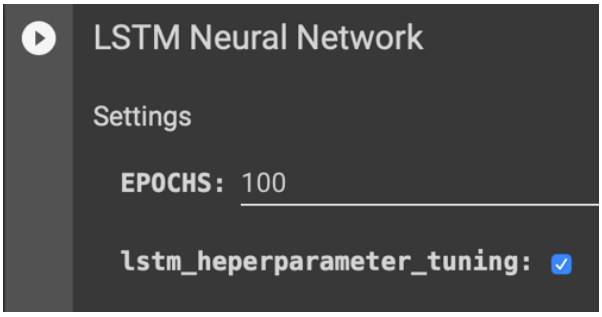
Obrázok 53 – Načítanie predspracovanej dátovej množiny

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 zadaním 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 sieť.

Príloha C: Článok na konferenciu IIT.SRC 2020

Data analysis and security risk identification in computer networks

Bc. Zoltán CSENGŐDY*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičová 2, 842 16 Bratislava, Slovakia
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	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	149.171.126.6	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

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 Perceptone 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.

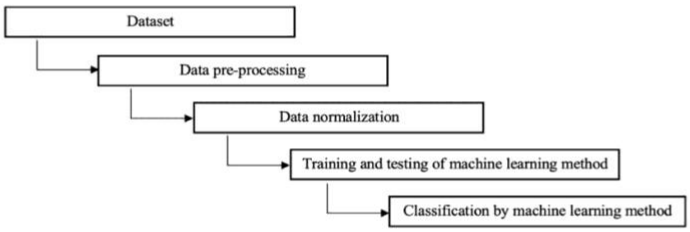


Fig. 2. Network Attack Classification Process

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.

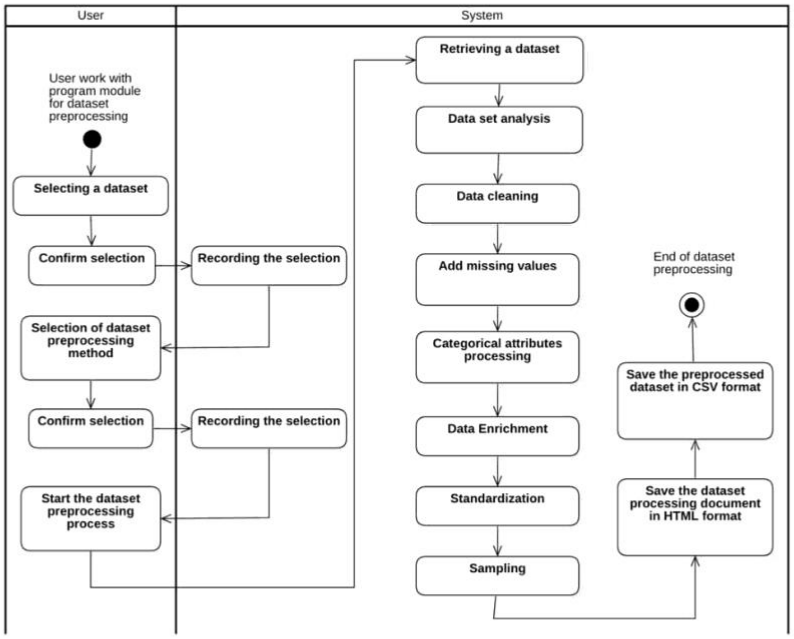


Fig. 3. Activity diagram of dataset pre-processing program module

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 geoip2.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.

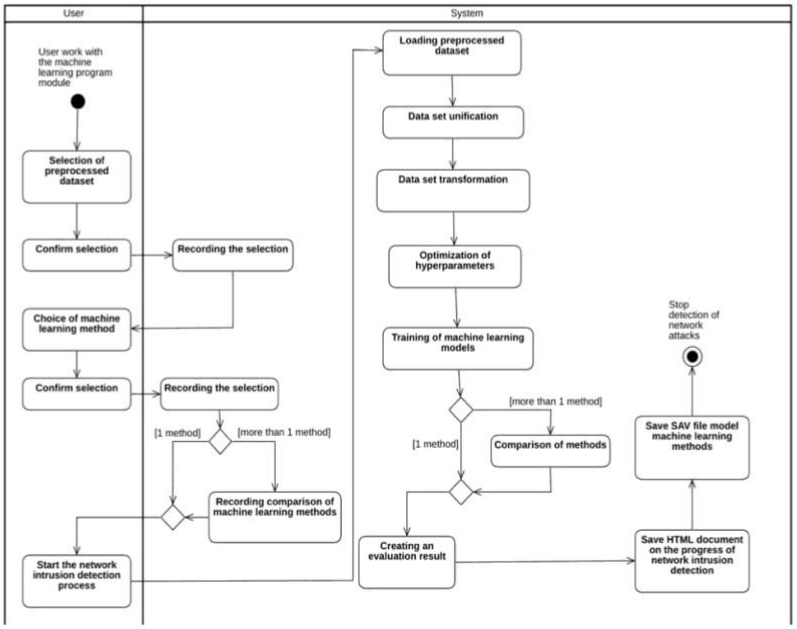


Fig. 4. Activity diagram of machine learning program module

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 it's robustness. It can handle not scaled data and in general does 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	**79.66**	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	**50.09**	98.93	99.26
dataset_4_OHE_TOPF_T	-	**72.55**	99.48	99.41
dataset_4_OHE_RESAMP_TOPF_T	-	**49.56**	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	**0**	97	98
dataset_4_MAP_RESAMP_TOPF_F	99	99	99	100
dataset_4_MAP_RESAMP_TOPF_T	99	**67**	99	99
dataset_4_OHE_TOPF_T	-	**10**	99	99
dataset_4_OHE_RESAMP_TOPF_T	-	**63**	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%.

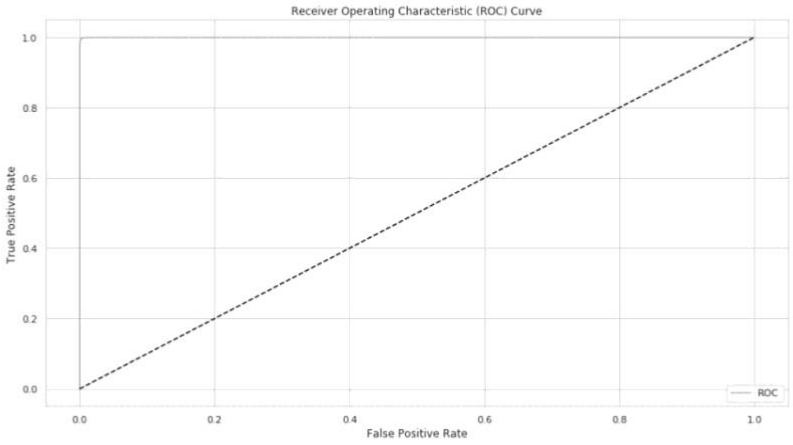


Fig. 5. ROC curve of the best random forest model

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.

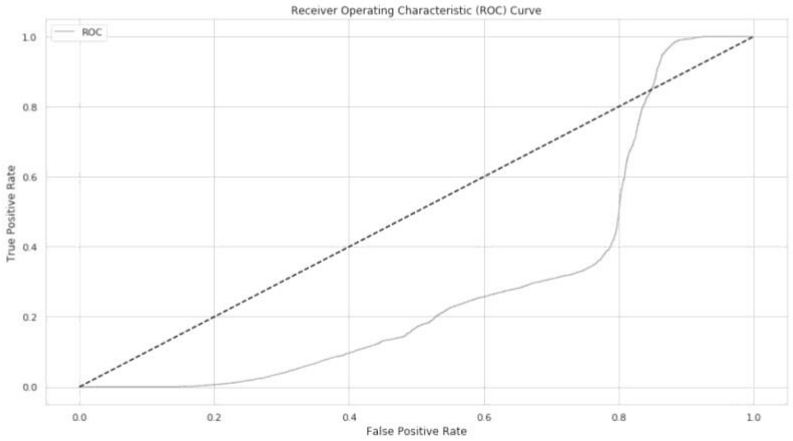


Fig. 6. ROC curve of SGD classifier

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
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=IwAR3lOtF1cBM08sGB8yzgj3evwzK00Nj8Af03oMttUmbpv9pd4Fh2J0tiZHA
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
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
    - DP_ZoltanCsengody.docx
    - DP_ZoltanCsengody.pdf
    - DP_prilohy_ZoltanCsengody.pdf
- \IITSRC – Adresár s článkom na konferenciu IIT.SRC 2020. Adresár obsahuje súbory vo formáte .docx a .pdf
    - IITSRC_ZoltanCsengody.docx
    - IITSRC_ZoltanCsengody.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 prílohami
    - \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