UNIVERZITET SINGIDUNUM
BEOGRAD
DEPARTMAN ZA POSLEDIPLOMSKE STUDIJE

DOKTORSKA DISERTACIJA

XOR DETEKTOR KONFLIKTNIH ODLUKA O ANOMALIJAMA U RAČUNARSKIM MREŽAMA

Mentor:
Prof. dr Marko Tanasković

Student:
Danijela Protić
Broj indeksa: 2021/480140

Beograd, 2023. godine

SINGIDUNUM UNIVERSITY
BELGRADE
DEPARTMENT FOR DOCTORAL STUDIES

DOCTORAL THESIS

XOR-BASED DETECTOR OF CONFLICTING DECISIONS ON ANOMALIES IN THE COMPUTER NETWORK

Mentor:
Prof. dr Marko Tanasković

Student:
Danijela Protić
Index number: 2021/480140

Belgrade, 2023

Mentor: Prof. dr Marko Tanasković
Univerzitet Singidunum
Beograd

Članovi komisije:

1. Prof. dr Marina Marjanović-Jakovljević
   Univerzitet Singidunum
   Beograd

2. Prof. dr Petar Spalević
   Univerzitet Singidunum
   Beograd

Datum odbrane:

APSTRAKT

Detekcija anomalija je prepoznavanje sumnjivog ponašanja računarske mreže koje se izvodi poređenjem nepoznatog sadražaja sa statističkim modelom normalnog mrežnog saobraćaja. Binarni klasifikatori bazirani na nadgledanom mašinskom učenju pokazali su se dobrim rešenjem ovakvih modela. U radu je prikazano pet tipova binarnih klasifikatora: k-najbližih suseda, k-najbližih suseda sa težinskim koeficijentima, stabla odlučivanja, modeli vektora potpore i feedforward neuronska mreža.

Osnovni problem kod nadgledanog mašinskog učenja je veliki broj podataka koji je neophodan da bi klasifikatori bili precizni. Da bi bilo smanjeno vreme za trening klasifikatora, uz minimalnu degradaciju tačnosti modela, u ovom radu izvedeno je preprocesuiranje koji se sastoji od dve faze: izbor numeričkih atributa i skaliranje atributa u simetrične granice. Razvijena je nova metoda bazirana na primeni tangens hiperboličke funkcije i damping strategije Levenberg-Marquardt algoritma. Za prikaz pozitivnog uticaja ovakvog preprocesuiranja na vreme obučavanja i tačnost klasifikatora korišćena je Kyoto 2006+ baza podataka koja je jedina javno dostupna baza realnog mrežnog saobraćaja namenjena isključivo za istraživanje detekcije anomalija u računarskim mrežama.

Za izabrane klasifikatore, najveću brzinu procesuiranja ima feedforward neuronska mreža dok se model k-najbližih suseda s težinskim koeficijentima pokazao najpreciznijim. Pretpostavka je da, kada klasifikatori rade simultano, oni treba ili da detektuju anomaliju ili normalni mrežni saobraćaj što s vremena na vreme nije slučaj, dolazi do različite procene o anomaliji, tj. nastaje konflikt. Detektor konflikta je blok koji izvodi logičku ekskluzivno ili (XOR) operaciju nad izlazima klasifikatora. Ukoliko su oba klasifikatora istovremeno detektovala anomaliju ili prepoznala saobraćaj kao normalan, njihova odluka je da konflikta nema. U suprotnom detektovan je konflikt. Broj detektovanih konflikata obezbeđuje mogućnost dodatne detekcije promena u ponašanju računarske mreže.

ABSTRACT

Anomaly detection is the recognition of suspicious computer network behavior by comparing unknown network traffic to a statistical model of normal network behavior. Binary classifiers based on supervised machine learning are good candidates for normality detection. This thesis presents five standard binary classifiers: the k-nearest neighbors, weighted k-nearest neighbors, decision trees, support vector machines and feedforward neural network.

The main problem with supervised learning is that it takes a lot of data to train high-precision classifiers. To reduce the training time with minimal degradation of the accuracy of the models, a two-phase pre-processing step is performed. In the first phase, numeric attributes are selected to reduce the dataset. The second phase is a novel normalization method based on hyperbolic the tangent function and the damping strategy of the Levenberg-Marquardt algorithm. The Kyoto 2006+ dataset, the only publicly available data set of real-world network traffic intended solely for anomaly detection research in computer networks, was used to demonstrate the positive impact of such pre-processing on classifier training time and accuracy.

Of all the selected classifiers, the feedforward neural network has the highest processing speed, while the weighted k-nearest neighbor model proved to be the most accurate. The assumption is that when the classifiers work concurrently, they should detect either an anomaly or normal network traffic, which occasionally is not the case, resulting in different decision about the anomaly, i.e. a conflict arises. The conflicting decision detector performs a logical exclusive OR (XOR) operation on the outputs of the classifiers. If both classifiers simultaneously detected an anomaly or recognized traffic as normal, their decision was no conflict had occurred. Otherwise a conflict is detected. The number of conflicts detected provides an opportunity for additional detection of changes in computer network behavior.

ZAHVALNICA

Ovim putem zahvaljujem se prof. dr Marku Tanaskoviću, mentoru mojih doktorskih studija i izrade ove disertacije. Korektnost, ekspeditivnost, ljubaznost i stručnost profesora Tanaskovića su izuzetni. Zahvaljujem se na komentarima i članovima komisije koji su učestvovali u korekciji rada.

Imam tu sreću da sam ja nekome i da je meni neko kćerka, sestra, osoba prirasla srcu, najbolji prijatelj i roditelj. Moja mala grupa za podršku zovu se (redom od najmlađe do najstarijeg): Ana Arsenijević, Lola Tomić, Vladimir Antić, Bojana Stanković Todić i Radojka i Dobrosav Protić. Posebno se zahvaljujem prof. dr Miomiru Stankoviću, izuzetnom čoveku, humanisti, stručnjaku i pravom gospodinu, na podršci ne samo u izradi ove doktorske disertacije, nego i na putu do njene realizacije, koji nije uvek bio lak.

Pozdravljam i sve bivše i sadašnje članove kolektiva Centra za primenjenu matematiku i elektroniku, izuzetnu grupu stručnih i disciplinovanih ljudi, koji su uticali na moj profesionalni razvoj. Prof. dr Loveleen Gaur sa Amity univerziteta u Indiji je zvezda u podršci izrade mojih naučnih radova u poslednjih godinu dana. Moj naučno istraživački rad takođe su podržali kolega i prijatelj Muhamed Šadić iz Ostina u Teksasu, mr Nebojša Gaćeša, urednik Vojnotehničkog glasnika, i Diana Orlić, koordinator za doktorske studije Univerziteta Singidunum u Beogradu. U retkim trenucima opuštanja, veliki značaj u mom životu imali su i ljubimac Cilka, plesna škola Cubanito iz Beograda i Nikolija Vujović @kućna.devojčica.

Hvala svima.

Danijela

SADRŽAJ

SPISAK SKRAĆENICA

ACC – Accuracy
AIDS – Application-based Intrusion Detection System
AIS – adaptivni imuni sistem
ARS – Adaptive Response System
BIS – biološki imuni sistem
BP – Back Propagation
DNS – Domain Name System
DoS – Denial of Service
DdoS – Distrubutet Denial of Service
DT – Decision Tree
FN – False Negative
FP – False Positive
FPR – False Positive Rate
FNN – Feedforward Neural Network
FNR – False Negative Rate
GA – Genetic Algorithm
GD – Gradient Descent
GN – Gaus-Njutnov algoritam
HIDS – Host Intrusion Detection System
IB – informaciona bezbednost
ID3 – Iterative Dichotomizer 3
IDS – Intrusion Detection System
IIS – Innate Imune System
IKS – informaciono komunikacioni sistem
IP – Internet Protocole
IT – informacione tehnologije
k-NN – k-Nearest Neighbour
LDA – Linear Discriminant Analysis
LM – Levenberg-Marquardt
LR – Linear Regression
LS – Least Squares
MLP – Multi Layer Perceptron
MM – Min-Max

SPISAK SKRAĆENICA - NASTAVAK

MSE – Mean Squares Error
NB – Naive Bayes
NPV – Negative Prediction Value
NIDS – Network Intrusion Detection System
NLP – Natural Language Processing
OS – operativni sistem
PCA – Principal Component Analysis
PPV – Positive Prediction Value
RF – Random Forest
SSH – Secure SHell
SVM – Support Vector Machines
TH – tangens hiperbolički
TN – True Negative
TNR – True Negative Rate
TP – True Positive
TPR – True Positive Rate
VIS – veštački imuni sistem
wk-NN – weighted k-Nearest Neighbour
XOR – Exclusive OR

SPISAK SLIKA

SPISAK TABELA

1. UVOD

Brz razvoj računarskih mreža unazad nekoliko decenija doprineo je ekspanziji raznovrsnih malicioznih napada na osetljive podatke. Kao rezultat, sistemi za detekciju napada na računarske mreže postali su nezaobilazan alat za svakoga čiji je cilj zaštita informacija. Međutim, probleme u procesuiranju, čuvanju i prenosu podataka ne moraju da izazovu isključivo maliciozni napadač, nego i ljudske greške, nepoznati poremećaji u mrežnim sistemima, pad napona u mreži, atmosferske promene i, u opštem slučaju, anomalije. Pojava anomalije u mrežnom saobraćaju je evidentna u svakodnevnom radu računarskih mreža.

Detekcija anomalije podrazumeva formiranje statističkog modela normalnog ponašanja računarske mreže na osnovu kojeg se određuje da li nepoznati saobraćaj pripada klasi normalnog ili klasi abnormalnog ponašanja.

Modeli nadgledanog mašinskog učenja pokazali su se dobrim kandidatima za procenu anomalije. U ovom istraživanju izvršen je izbor pet modela nadgledanog mašinskog učenja koji su korišćeni kao binarni klasifikatori: model k-najbližih suseda, model k-najbližih suseda sa težinskim koeficijentima, stabla odlučivanja, vektori oslonca i feedforward neuronska mreža. Međutim, neophodan je veliki broj podataka za obučavanje modela, što traje dug vremenski period.

Da bi bilo smanjeno vreme procesuiranja, uz minimalnu degradaciju tačnosti modela, u radu je izvedeno preprocesuiranje kojeg čine izbor relevantnih numeričkih atributa i skaliranje izabranih atributa u simetrične granice, uz primenu tangens hiperboličke funkcije i damping strategije Levenberg-Marquardt algoritma. Za prikaz pozitivnog uticaja ovakvog preprocesuiranja na vreme evaluacije i tačnost klasifikatora korišćena je Kyoto 2006+ baza podataka snimljena na pet računarskih mreža unutar i izvan Univerziteta u Kyoto-u, u periodu od 10 godina, od 2006. do 2015. godine. Ova baza je jedina javno dostupna baza realnog mrežnog saobraćaja namenjena isključivo za istraživanje detekcije anomalija u računarskim mrežama.

Rezultati eksperimenata pokazali su da najveću brzinu procesuiranja ima feedforward neuronska mreža a, iako svi klasifikatori pokazuju izuzetno visoku tačnost, model k-najbližih suseda s težinskim koeficijentima se pokazao najpreciznijim u smislu maksimalne tačnosti i minimalnog broja lažnih alarma. Iz tog razloga su ova dva klasifikatora iskorišćena kao osnov za detekciju zasnovanoj na logičkoj operaciji ‘ekskluzivno ili’, odnosno XOR detekciju.

Kada klasifikatori rade u paraleli na istom mestu u računarskoj mreži oni bi istovremeno trebali ili da detektuju anomaliju ili da mrežni saobraćaj smatraju normalnim. Međutim, s vremena na vreme, dolazi do različite procene o anomaliji, odnosno normalnom saobraćaju, tj. nastaje konflikt. Detektor konflikta je baziran na primeni bitske operacije XOR na izlaze, odnosno odluke klasifikatora. Ukoliko su oba klasifikatora istovremeno detektovala anomaliju ili prepoznala saobraćaj kao normalan, njihova odluka je da konflikta nema. U suprotnom detektovan je konflikt. U okviru ove disertacije prikazane su mogućnosti poboljšanja tačnosti i procesnog vremena, i mogućnosti dodatne detekcije anomalija koja se izvodi XOR detekcijom.

Tekst disertacije prati navedenu problematiku tako da je nakon uvodnog dela prvo dat pregled literature po oblastima, zatim poreklo detekcije anomalija i osnove izbora binarnih klasifikatora. Nakon toga, prikazane su metode selekcije i skaliranja atributa. Nova metoda, skaliranjem tangens hiperboličkom funkcijom i upotreba Levenberg-Marquardt

algoritma, prva je od naučnih doprinosa teze. Drugi doprinos, XOR detektor konfliktnih odluka o anomalijama, opisan je u poglavlju koje sledi nakon poglavlja koje opisuje skaliranje.

Disertacija je organizovana u trinaest poglavlja, koja su međusobno povezana u celinu. Prvo poglavlje je uvod u kojem je ukratko izložen problem koji će biti razmatran u radu. Uvod sadrži opšte podatke o značaju i primeni detekcije anomalija u računarskoj mreži i prikaz detektora konfliktnih odluka o anomalijama u računarskim mrežama koji je baziran na primeni XOR operacije. Predmet, ciljevi, hipoteze i metodološki pristup predstavljaju poseban deo uvodnog razmatranja.

Drugo poglavlje je pregled relevantne literature. Korišćene su javno dostupne baze podataka, rezultati teorijskih i praktičnih istraživanja u naučnim publikacijama, i Internet pretraživači. Spisak referenci dat je u posebnom poglavlju na kraju disertacije.

U okviru trećeg poglavlja definisani su pojmovi vezani za detekciju anomalija. Opisana je relacija između biološkog i veštačkog imunog sistema. Prikazane su osnovne karakteristike anomalija i detekcije anomalija u računarskim mrežama i dat je opis sistema za detekciju anomalija.

Četvrto poglavlje daje uvid u sisteme za detekciju malicioznih napada na računarske mreže, i opisuje uticaj ljudskih grešaka, i različitih problema u funkcionisanju informaciono-komunikacionih sistema.

Peto poglavlje daje prikaz mašinskog učenja, s posebnim naglaskom na sledećih pet modela nadgledanog mašinskog učenja: model k-najbližih suseda, model k-najbližih suseda sa težinskim koeficijentima, model vektora potpore, stabla odlučivanja i feedforward neuronska mreža.

Šesto poglavlje opisuje baze podataka, selekciju i skaliranje atributa, sa posebnim osvrtom na binarnu klasifikaciju u detekciji anomalija, kada su binarni klasifikatori modeli nadgledanog mašinskog učenja. Opisan je proces binarne klasifikacije koji uključuje preprocesuiranje, obučavanje klasifikatora i postprocesuiranje sa vizuelizacijom. U okviru preprocesuiranja posebno su opisani: izbor numeričkih atributa, skaliranje tangens hiperboličkom normalizacijom i Levenberg-Marquardt algoritam.

Akcenat sedmog poglavlja je na detektoru konfliktnih odluka o anomalijama baziranom binarnim klasifikatorima visoke tačnosti i btiskoj operaciji ‘ekskluzivno ili’ (XOR).

Osmo poglavlje prikazuje eksperimentalne rezultate. Opisana je selekcija numeričkih atributa iz Kyoto 2006+ baze podataka, sa rezultatima koji potvrđuju da je moguće, smanjenjem atributa sa ukupno 24 na 9 numeričkih atributa, postići smanjenje vremena procesuiranja za više od dva puta, sa malom degradacijom tačnosti, za sve izabrane klasifikatore. U sledećoj fazi eksperimenata prikazane su prednosti skaliranja hiperboličkim tangensom u odnosu na Z-score standardizaciju, Min-Max normalizaciju u granice [0,1] i Min-Max normalizaciju u granice [-1,1]. Proces izbora i evaluacije klasifikatora takođe je prikazan u ovom delu eksperimenata. Posebno je prikazana struktura XOR detektora sa obrazloženjem korišćenja najbržeg i najpreciznijeg klasifikatora u detekciji konfliktnih odluka o anomalijama. Prikazani su rezultati procene ukupnog broja kontradiktornih odluka o anomalijama.

Zaključak rada daje kratak prikaz dobijenih rezultata, osvrt na dokaz postavljenih hipoteza i moguće pravce razvoja u ovoj oblasti. Na kraju rada dat je spisak korišćenih referenci.

Rad, takođe, sadrži tri priloga u kojem su detaljno prikazani: afina transformacija, softverski kodovi za evaluaciju pet binarnih klasifikatora, kao i gradijentna, Gaus-Njutnova i Levenberg-Marquardt-ova metoda.

1.1. Predment, hipoteze, ciljevi i metode istraživanja

U tekstu koji sledi dat je opis predmeta istraživanja, određene su tri hipoteze, navedeni ciljevi istraživanja i prikazane metode koje su korišćene u toku naučno-istraživačkog procesa, kao i metode post-procesuiranja i vizuelizacije dobijenih rezultata.

1.1.1. Predmet istraživanja

Predmet istraživanja ove disertacije je detekcija konfliktnih odluka o anomalijama u računarskim mrežama. Prednost u detekciji anomalija je prepoznavanje nepoznatog malicioznog napada dok je osnovni problem veliki broj podataka neophodan za evaluaciju statističkog modela normalnog ponašanja računarske mreže. Binarni klasifikatori tipa k-najbližih suseda, k-najbližih suseda sa težinskim koeficijentima, stabla odlučivanja, modeli vektora potpore i feedforward neuronska mreža pokazali su se kao dobro rešenje za ovaj problem. Predmet istraživanja su i nove metode izbora i skaliranja atributa, kojima je moguće smanjiti brzinu procesuiranja uz minimalnu degradaciju tačnosti.

1.1.2. Hipoteze istraživanja

Osnovna hipoteza istraživanja konfliktnih odluka o anomalijama u računarskim mrežama glasi:

> H: *„XOR detekcijom konfliktnih odluka o anomalijama moguće je povećati stepen prepoznavanja malicioznosti i grešaka u računarskim mrežama.“*

U skladu sa osnovnom hipotezom izvedene su dve sporedne hipoteze:

> H1: *„Izborom skaliranja tipa hiperboličkog tangensa i damping strategije Levenberg-Marquardt algoritma moguće je značajno smanjiti vreme procesuiranja uz minimalnu degradaciju tačnosti binarnih klasifikatora.“*

> H2: *„Nelinearni binarni klasifikatori sa težinskim koeficijentima pokazuju najbolje karakteristike za primenu u XOR detekciji konfliktnih odluka o anomalijama u računarskim mrežama.“*

1.1.3. Ciljevi istraživanja

Osnonvni cilj istraživanja je formirati detektor konfliktnih odluka o anomalijama koji je baziran na XOR operaciji primenjenoj na vrednostima izlaza binarnih klasifikatora. Da bi ovaj cilj bio ostvaren, prvo je potrebno izvršiti preprocesuiranje izborom numeričkih atributa iz Kyoto 2006+ baze podataka. Nakon toga je potrebno skalirati izabrane atribute tangens hiperboličkom prenosnom funkcijom sa pravilima koja važe za damping strategiju Levenberg-Marquardt algoritma. Cilj je, u zavisnosti od tačnosti i vremena procesuiranja, izabrati binarne klasifikatore bazirane na nadgledanom mašinskom učenju koji pokazuju najveću tačnost i najkraće vreme procesuiranja, za korišćenje u XOR detekciji. Na kraju, cilj

je dokazati da XOR detektor konfliktnih odluka o anomalijama predstavlja značajan alat u dodatnoj detekciji potencijalnih malicioznosti ili grešaka u računarskim mrežama.

1.1.4. Primenjene metode istraživanja

U toku naučno-istraživačkog rada primenjen je niz metoda kako bi bili zadovoljeni metodološki zahtevi za pouzdanost, objektivnost, sistematičnost i ponovljivost rezultata. U skladu sa izabranom problematikom istraživanja, definisanim ciljevima i postavljenim hipotezama, radi definisanja naučnih i stručnih zaključaka i pronalaženja mogućih rešenja, prvo je izvršeno temeljno istraživanje javno dostupnih referenci i baza podataka.

Metoda kompleksnog posmatranja i analiza sadržaja primenjena je prilikom obrade rezultata istraživanja u onim delovima predložene teme koji se odnose na karakteristike i primenu prikazanih binarnih klasifikatora, koji su dostupni u naučnim referencama.

Kvantitativna istraživanja i analiza korišćene su za prikaz istraživačkih procedura, kontrolu faktora koji imaju uticaj na tok istraživačkog procesa i obradu podataka. Prikazan je izbor Kyoto 2006+ baze podataka, na osnovu komparativne analize sa 15 drugih baza podataka koje se koriste za evaluaciju sistema za detekciju napada na računarske mreže. Statističke metode normalizacije i standardizacije korišćene su da se pokaže pozitivan uticaj hiperboličkog tangensa i damping strategije na skaliranje atributa.

Eksperimentima su utvrđene karakteristike grupa klasifikatora i, na osnovu parametara koji su dostupni u programskom paketu MATLAB: Classification Learner, izabrano je pet partikularnih modela za izvođenje eksperimenata. Na osnovu rezultata eksperimenata primenjene su metode analize dobijenih klasifikatora i izvršen je izbor najpovoljnijih rešenja za XOR detekciju. Za svaku od eksperimentalnih faza izvedena je vizuelizacija sa pratećim komentarima. Rezultati su prikazani tabelarno i grafički.

1.2. Očekivani naučni doprinos

U ovoj disertaciji prikazani su brojni rezultati istraživanja detekcije anomalija, binarne klasifikacije, preprocesuiranja podataka i hibridnih modela za poboljšanu detekciju anomalija. Ovi rezultati su upotrebljeni kao polazna osnova za sprovedena istraživanja i služe kao osnova za dalja istraživanja i produbljivanje znanja o ovoj temi.

Disertacija sadrži predlog i koncept rešenja detekcije kontradiktornih odluka o anomalijama u računarskim mrežama koji je baziran na XOR detektoru čije ulaze čine odluke o anomalijama sa izlaza dva binarna klasifikatora, sa težinskim koeficijentima, koji rade u paraleli i odlučuju o tipu mrežnog saobraćaja. Izlazni parametar ovog detektora je „1“ ukoliko je detektovana konfliktna odluka, u suprotnom, izlaz ima vrednost „0“. Cilj ove detekcije je skrenuti pažnju donosiocu odluka da je došlo do konflikta i da postoji mogućnost nedetektovanog malicioznog napada na računarsku mrežu.

Da bi ovakav sistem mogao da radi bilo je neophodno izvesti preprocesuiranje atributa. Doprinos disertacije biće izbor numeričkih atributa, kojima će biti obezbeđeno skaliranje u simetrične granice, tako da atributi ne utiču jedni na druge zbog različitih skala. Razvijena je nova metoda skaliranja u granice određene tangens hiperboličkom prenosnom funkcijom u blizini prevojne tačke, zbog čega damping strategija Levenberg-Marquardt algoritma znatno ubrzava obučavanje klasifikatora.

Tema doktorske disertacije je aktuelna a doprinos istraživanja je i razvoj novog matematičkog metoda za skaliranje numeričkih atributa i razvoj detektora konfliktnih odluka o anomalijama u računarskim mrežama.

U skladu sa predmetom i ciljem rada, postavljenim hipotezama i metodologijom istraživanja, očekivano je da će rezultati ovog naučno-istraživačkog rada dati doprinos ukupnom napretku nauci u oblasti zaštite računarskih mreža. Očekivani ishod sprovedenog istraživanja je i da će postavljene hipoteze biti dokazane ili opovrgnute.

2. PREGLED LITERATURE

Problemi koji, u poslednjim decenijama, prate skladištenje, procesuiranje i prenos podataka, postali su značajan istraživački zadatak i veliki izazov u svakodnevnoj poslovnoj praksi kako sa stanovišta informatičke pismenosti korisnika računarskih mreža, tako i sa aspekta tehnika zaštite informacija. U tekstu koji sledi prikazan je pregled referenci četiri oblasti koje obuhvataju ovaj istraživački izazov, a čiji je osnovni zadatak da pripremi čitaoca disertacije na: (1) način izbora baza podataka na kojima će biti izvršeno istraživanje teme ovog rada, (2) tipove smanjenja dimenzionalnosti baza podataka kroz selekciju atributa, (3) skaliranje atributa s ciljem da oni međusobno ne utiču jedni na druge i (4) klasifikaciju, kao proces, s posebnim osvrtom na binarnu klasifikaciju u detekciji anomalija u ponašanju računarske mreže.

2.1. Izbor podataka za evaluaciju modela

Javno dostupne baze podataka, namenjene za detekciju napada na računarske sisteme, mogu se podeliti u dve osnovne grupe. Prvoj, mnogobrojnijoj grupi, pripadaju simulacije realnog mrežnog saobraćaja, koje su namenjene za detekciju partikularnih napada. Drugu, manju grupu, čine baze podataka snimljenog, realnog mrežnog saobraćaja, namenjene za detekciju kako partikularnih napada, tako i anomalija u računarskoj mreži. U praksi su najčešće korišćene sledeće baze podataka: ADFA-LF, ADFA-WD, AWID, CAIDA, CIC-IDS-2017, CIDDS-001, CSE-CIC-2018, DARPA, IRSC, ISCX 2012, KDD Cup '99, Kyoto 2006+, NSL-KDD, UGR'16 i UNSW-NB15. Osnovne karakteristike ovih baza prikazane su Tabelom 1. Parametri koji su prikazani su: naziv baze, vreme kada je ona generisana, vrste napada, broj atributa, tip saobraćaja i sadržaj baze (Protić, et al., 2022a).

Tabela 1. Opis najčešće korišćenih baza podataka

Baza	Nastanak	Vrste napada	Broj atributa	Saobraćaj	Sadržaj baze
ADFA	2014	Brute force, Java/Linux meterpreter, C100 webshell	26	Hibridni	Linux/Windows OS system call
AWID	2015	Wi-fi 802.11 attacks	156	Simulirani	Wireless LAN saobraćaj.
CAIDA	2007	Distributed Denial of Service	Nije prikazano	Hibridni	Komercijalni backbone linkovi sa high-speed monitora.
CIC-IDS-2017	2017	Botnets, DDoS, Goldeneye, Hulk, HTTP	80+	Simulirani	5 dana packet-based mrežnog saobraćaja.
CIDDS-001	2017	DoS, Bruteforce, Ping/Port Scan	14	Simulirani	4 nedelje saobraćaja OpenStack i Ext. servers.
CSE-CIC-2018	2018	FTP/SSH potator, Dos, DDoS, Web attacks, $1^{st}/2^{nd}$ level infiltration, botnet.	80+	Simulirani	10 dana mrežnog saobraćaja
DARPA	1998-1999	DoS, R2L, U2R, probe	41	Simulirani	7 nedelja packet-based saobraćaja.
IRSC	2015	DoS, R2U, surveillance	Nije dostupno	Hibridni	Mreža Univerziteta Sudan.
ISCX 2012	2012	Infiltrating, DDoS, HTTP, SSH	20	Simulirani	7 dana packet-based saobraćaja.
KDD Cup '99	1998	Denial of Service, R2L, U2R, probing	42	Simulirani	5 nedelja packet-based saobraćaja
Kyoto 2006+	2006-2015	Port scan, malware, shellcode, DoS	24	Realni	10 godina realnog mrežnog saobraćaja.
NSL-KDD	1998	Denial of Service, R2L, U2R, probing	42	Simulirani	KDD-Cup '99 bez redundantnih zapisa i duplikata u test skupu.
UGR'16	2016	Denial of Service, Portscans Botnet	41	Hibridni	4 meseca network traces u tier-3 ISP.
UNSW-NB15	2015	Contemporary attacks behavior	49	Hibridni	31 sat tcpdump traces

Od 15 baza podataka koje su prikazane u prethodnom delu teksta, za eksperimente prikazane u ovoj disertaciji korišćena je Kyoto 2006+ baza podataka iz dva osnovna razloga: 1) sadržaj baze su realni podaci i 2) baza je namenjena isključivo za detekciju anomalija.

2.2. Selekcija atributa

U zavisnosti od toga da li trening skup ima labele ili ne, selekcija atributa može da bude nadgledana (supervised, engl.) (Song et al., 2007; Weston et al., 2003) nenadgledana (unsupervised, engl.) (Dy, Brodley, 2007; Mitra et al., 2002) ili polu-nadgledana (semi-supervised, engl.) (Zhao, Liu, 2007; Xu et al., 2010). Nenadgledana selekcija koristi podatke u bez da postoji merena relevantnost atributa. U nadgledanoj selekciji, indukcioni algoritam predstavljen je trening skupom, sa instancama koje su opisane skupom atributa i odgovarajućim labelama klasa. Značaj atributa određuje se na osnovu informacija koje nose labele. Međutim, da bi bila postignuta visoka tačnost, neophodan je veliki skup podataka, što utiče na vreme procesuiranja (procesno vreme). Nadgledani izbor atributa može se dalje podeliti na filter, wrapper i embedded izbor (Prokodi, 2014). Filter metod koristi klasifikaciju za izbor atributa, nezavisno od klasifikatora koji će biti korišćeni. Ovim metodom bira se podskup atributa pre klasifikacije, u fazi preprocesuiranja. Filter metod baziran je na opštim karakteristikama podataka i određuje atribute bez da uključi trening algoritam (Artur, 2021; Osniaye et al., 2019). Wrapper metod zahteva predeterminisani algoritam obučavanja i koristi ga kao crnu kutiju (black box, engl.) na nekoliko promenljivih u zavisnosti od njihovog uticaja na snagu predikcije. Performanse algoritma određuju evaluacioni kriterijum (Khammashi, Krichen, 2017; Umar et al., 2020). Predefinisani trening algoritam odrediće kvalitet izabranih atributa. Najpoznatije wrapper metode su genetički algoritam (GA), rekurzivna eliminacija atributa i sekvencijalna selekcija atributa. Embedded metod komubinuje efikasnost nenadgledanog metoda i selekciju kod nadgledanih metoda izbora atributa. Metod uči koji atribut obezbeđuje najveću tačnost klasifikacije, uvodi selekciju promenljivih u proces treninga i određuje relevantnost atributa analitički, iz funkcije cilja za model koji se obučava (Thakkar, Lohya, 2021; Almomani, 2020). Najpoznatije embedded metode su L1 i L2 regularizacija, i elastične mreže (Choudhury, 2021; Tang et al., 2021). Prednosti i mane sva tri metoda selekcije atributa prikazana su Tabelom 2 (Biwas et al., 2016; Rosely et al., 2019; Musher et al., 2017).

Tabela 2. Taksativni zbirni prikaz filter, wrapper i embedded metoda

Selekcija atributa	Prednosti	Mane
Filter	• Ne zavisi od procesnog vremena algoritma mašinskog učenja. • Mali rizik od overfitting-a. • Dobra sposobnost generalizacije. • Skalabilan.	• Ignoriše zavisnost atributa. • Ignoriše interakciju sa klasifikatorom.
Wrapper	• Procenjuje najbolje karakteristike podskupa atributa korišćenjem karakteristika trening algoritma. • Modeluje međuzavisnosti atributa. • Veća tačnost predikcije od Filter metoda.	• Viši rizik od overfitting-a. • Izbor zavisi od klasifikatora. • Veći resursi za proračune.
Embedded	• Kombinacija prethodnih metoda. • Redukuje rizik od overfitting-a. • Modeluje zavisnosti atributa.	• Problemi kod selekcije malog skupa atributa. • Izbor zavisi od klasifikatora.

Ahmed et al. (2011) istražuju tehnike skraćivanja vremena procesuiranja i predlažu algoritam selekcije atributa za redukciju memorije i ubrzavanje klasifikacionog algoritma. Uporedili su klasifikatore bazirane na neuronskim mrežama, modelima k-najbližih suseda (k-Nearest Neighbours, k-NN, engl.), stablima odlučivanja (Decision Trees, DT, engl.) i vektorima oslonca (Support Vector Machines, SVM, engl.). Klasifikacioni algoritmi su korišćeni da prepoznaju html, pdf, jpg, exe, gif, doc, txt, MP3, asp i xls fajlove. Autori pokazuju da veliki broj klasifikatora redukuje vreme procesuiranja za više od 50% i ima tačnost oko 90%, kada se koristi samo ~40% od ukupnog broja atributa. Maza et al. (2018) se koncentrišu na SVM, DT, k-means, veštački imuni sistem (VIS) i GA, i dolaze do zaključka da su algoritmi za selekciju atributa koji koriste filter pristup pokazali manje vreme procesuiranja od onih koji su bazirani na rastojanju, konzistentnosti, i sl. Zhao et al. (2018) pokazuju da korektna primena selekcije atributa može značajno da poboljša vreme procesuiranja i ukupne karakteristike klasifikacije. Različite taktike izbora atributa su korišćene da se smanji veličina skupa podataka, a izbor značajno zavisi od tipa klasifikacije. Suman et al. (2019) predlažu filter metod za izbor atributa u više-objektnoj optimizaciji da identifikuju relevantne atribute iz skupa neoznačenih podataka. Belavagi et al. (2016) biraju podskup optimalnih atributa uz šest različitih klasifikatora: DT, SVM, Random Forrest (RF, engl.), Adaboost i višeslojni perceptron (Multi-Layer Perceptron, MLP, engl.). Nakon nalaženja podskupa, 10-fold krosvalidacija je izvedena na jednom od podskupova. DT daje najbolje rezultate za KDD Cup ’99 i NSL-KDD baze podataka. Iz tog razloga, autori su koristili DT da odrede rezultate na Kyoto 2006+ bazi podataka. Tačnost (Accuracy, ACC engl.), ACC=99.6% i F-score=99.65%, su vrednosti dobijene eksperimentima. Perez et al. (2019) predlažu Principal Component Analysis (PCA, engl.) za redukciju dimenzionalnosti skupa podataka. Min-Max skaliranje je korišćeno za normalizaciju atributa u opseg [0,1]. Eksperimenti su izvedeni na k-NN, SVM i metodu ansambla (ensemble, engl.). Rezultati pokazuju prosečnu tačnost (60%), visoku preciznost (0.99) i nisku vrednost za F1-score (0.53). Rezultati takođe pokazuju da PCA ne unosi poboljšanja. Protic i Stankovic (2018) daju sumarni prikaz istraživanja uticaja izbora atributa i normalizacije numeričkih atributa, pokazuju da devet numeričkih atributa, skaliranih u opseg [-1,1], daju tačnost od preko 99% za k-NN, k-NN sa težinskim koeficijentima (weighted k-NN, wk-NN, engl.), DT i SVM modele, i zaključuju da wk-NN postiže najveću tačnost, a DT najkraće vreme procesuiranja (t_p). Isti autori (2020) istražuju i uticaj preprocesuiranja na feedforward neuronsku mrežu (Feedforward Neural Network, FNN, engl.) koja se koristi u binarnoj klasifikaciji. U poređenju sa prethodnim modelima, FNN se pokazala veoma preciznom s najkraćim vremenom procesuiranja. Najafabadi et al. (2016) su izdvojili atribute za analizu bezbednosti podataka i izabrali tri klasifikaciona modela: k-NN (k=5), DT (C5.4) i Naive Bayes (NB, engl.). Tabela 3 daje pregled referenci vezanih za metodologiju izbora atributa i izabrane klasifikacione algoritmime.

Tabela 3. Izbor atributa i izabrani klasifikacioni algoritmi

Autori	Klasifikatori	Selekcija atributa	Metrika
Protic, Stankovic (2020)	FNN, SVM, DT, k-NN, wk-NN	9 numeričkih atributa, Min-Max	ACC, t_p
Ahmad, Aziz (2019)	SVM, k-NN	CFS, PSO	ACC, TPR, FPR
Suman et al. (2019)	DT, SVM, k-NN, MLP	NSGA-II	ACC, F1-score
Kalavadekar, Sane (2019)	DT	GA	ACC, F1-score
Perez et al. (2019)	k-NN, SVM, DT	PCA, Min-Max	ACC, F1-score
Salo (2019)	SVM, MLP	PCA, IG	ACC, F-measure,
Protic, Stankovic (2018)	SVM, DT, k-NN, wk-NN	9 numeričkih atributa, Min-Max	ACC, t_p
Zhao et al. (2018)	SVM	RPFMI	ACC
Mishra et al. (2016)	k-NN, DT	AUC, S2N, ROC	ACC, TPR, FPR
Ambusiadi, Nanda (2014)	DT, k-NN, SVM, ANN	FIMFS	ACC, F-measure, t_p

2.3. Skaliranje atributa: normalizacija i standardizacija

Skaliranje atributa je tehnika koja se primenjuje nakon njihove selekcije kako bi se smanjio međusobni uticaj atributa koji su različito skalirani. Naime, ukoliko su atributi različito skalirani, algoritam može da „smatra“ atribute sa većim amplitudama značajnijim od onih čije su maksimalne i/ili minimalne vrednosti znatno manje, iako ti atributi mogu da imaju značajniji uticaj na obučavanje modela. Metode skaliranja pripadaju jednoj od dve osnovne grupe: (1) normalizacija i (2) standardizacija. Različite opcije normalizacije i standardizacije razmatraju Obaid et al. (2019).

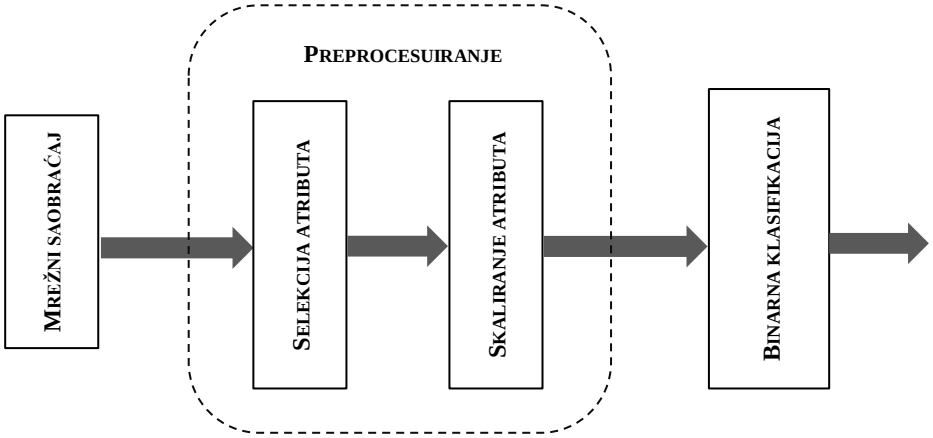
Normalizacijom se, uglavnom, atributi skaliraju tako da se vrednost instanci u atributu kreću u granicama [0,1] ili [-1,1] (Min-Max normalizacija). Na ovaj način, svaki od atributa postaje jednako značajan. Normalizacija je korisna u slučajevima kada skup podataka ne sadrži outlier-e i, kada se zna, da raspodela podataka nije Gausova (što može biti korisno kod neuronskih mreža ili algoritama najbližih suseda) (Maza, 2018). Treba imati u vidu da je neophodno, nakon klasifikacije, izvršiti denormalizaciju dobijenih podataka u osnovni opseg (postprocesuiranje). Min-Max normalizaciju primenjenu na atribute Kyoto 2006+ baze podataka opisuju Al-Imran i Ripon (2021) i Garcia et al. (2015). Obzirom da je ova baza neizbalansirana (instance normalnog saobraćaja i anomalije nisu jednako raspodeljene), normalizacija se primenjuje pre podele podataka na obučavajući i test skup.

Standardizacija je, na drugoj strani, tehnika skaliranja u kojoj su vrednosti skaliranih atributa takve da prate raspodelu verovatnoće. Treba napomenuti da standardizacija, za razliku od normalizacije, nema granični opseg. Z-score standardizacija je takva da atributi prate Gausovu raspodelu odnosno instance su centrirane oko nulte srednje vrednosti sa jediničnom standardnom devijacijom. Obzirom na činjenicu da Z-score standardizacija nema granični opseg, ne važi skaliranje u binarne granice tako da odstupanja mogu da dovedu do manje tačnosti ili povećanog broja lažno pozitivnih alarma.

2.4. Klasifikacija podataka i XOR detekcija

U opštem slučaju, proces klasifikacije se može podeliti na: (1) prikupljanje podataka, (2) preprocesuiranje (izbor i skaliranje atributa) i (3) klasifikaciju sa postprocesuiranjem (ukoliko je potrebno) (Slika 1). Algoritmi za klasifikaciju preslikavaju (mapiraju) ulazne podatke na željene izlaze i, za nepoznatu instancu, određuju kojoj od poznatih klasa pripada. U nadgledanom mašinskom učenju klasifikacija se odnosi na problem prediktivnog modelovanja u kojem se predviđa labela klase za datu instancu ulaznih podataka.

Većina problema u klasifikaciji zahteva predviđanje izlaznih promenjivih koje mogu biti kategoričke, numeričke ili statističke, na osnovu jednog ili više atributa, za formiranje statističkog/matematičkog modela koji povezuje skup atributa s odgovarajućim ciljevima. Nakon toga, model se koristi za predviđanje budućih, nepoznatih, izlaznih podataka, na osnovu odgovarajućih ulaza.



Slika 1. Proces binarne klasifikacije

U binarnoj klasifikaciji postoje dve međusobno isključive klase (1/0, true/false, da/ne, itd). U detekciji anomalija, jedna klasa odgovara labeli koja opisuje normalan saobraćaj u računarskoj mreži, dok druga labela označava klasu abnormalnog ponašanja mreže. Binarni klasifikatori bazirani na nadgledanom mašinskom učenju pokazali su se dobrim rešenjem za detekciju anomalije. Umar, Gupta i Arora (2021) daju pregled statističkih podataka o citiranosti publikacija od 2005. do 2020. godine, za različite implementacije mašinskog učenja i klasifikacije napada na računarske mreže. Radovi koji opisuju SVM (ukupno 1716) su najcitiraniji, sa ~27,000 citata u istraživanjima. Više od 2,000 publikacija o primeni neuronskih mreža, u istoj disciplini, ima oko 21,000 citata. DT modeli su citirani ~13,000 puta u 757 članaka, nakon čega slede FNN (252 publikacija i 5496 citatta), PCA (381 rad i 4279 citata), itd. Autori takođe pokazuju da zaštita od napada korišćenjem pristupa mašinskog učenja ima mnnogo veći broj citata od drugih pristupa. Prednosti SVM modela opisane su u radovima autora McCharty (2021) i Ring et al. (2019), karakteristike DT opisuju Protic i Stankovic (2020) i Pai et al. (2021), k-NN i wk-NN su prikazani u radovima Protic i Stankovic (2020a) i Ambusiadi et al. (2019), a veštačke neuronske mreže opisane su u radovima Pai et al. (2021) i Abiodun et al. (2018).

Primenjena pravila XOR operacije u detekciji anomalija retko se sreću u praksi ali postoje akademske reference koje se ovim problemom bave. Autori uglavnom istražuju osobine XOR pravila u detekciji prisustva ili odsustva determinističkih signala. Autori Sun et al. (2022) predstavljaju dva senzora za koje se smatra da nisu istog tipa ali imaju identične marginalne raspodele za komponente šuma. Prikazani koncept, kao i osnovne principe paralelnih proračuna u detekciji različitih odluka prikazuju i Stefan i Malita (2014). Cilj je uporediti sposobnost klasifikacije dva veoma precizna binarna klasifikatora koja rade u paraleli na istom mestu u računarskoj mreži. Decentralizovani sistem za detekciju sa paralelnom topologijom razmatra i Teodorescu (2017). Metod primenjen u realizaciji XOR detektora konfliktnih odluka koji je prikazan u ovoj disertaciji sadrži osnovne elemente svakog od pomenutih pristupa.

3. ULOGA IMUNOG SISTEMA U DETEKCIJI ANOMALIJA

Detekcija anomalija je proces u kojem je potrebno detektovati stanje koje se razlikuje od, unapred poznatog, normalnog stanja nekog sistema. U osnovi, anomalija je svako odstupanje od normalnosti a osnovni problem u detekciji anomalija je na koji način formirati model normalnog stanja sistema, na osnovu kojeg će biti određeno da li je neko novo stanje sistema normalno ili abnormalno. Da bi to bilo moguće, nephodno je poznavati biološki imuni sistem (BIS), koji u sebi sadrži prirodne detektore odstupanja od normalnog stanja organizma, na osnovu čega je moguće formirati veštački imuni sistem koji će moći dovoljno dobro da imitira pretpostavljenu karakteristiku biološkog imunog sistema, što će, posledično, omogućiti i detekciju anomalija u računarskoj mreži.

3.1. Biološki imuni sistem

Bilološki imuni sistem je decentralizovani, kompleksni, adaptivni sistem organa, ćelija i molekula koje štite organizam od patogena. Glavna uloga BIS-a je da odredi, odnosno napravi razliku između sopstvenih ćelija (self-cells, engl.) i ćelija koje to nisu (non-self cells, engl.). On prepoznaje, odgovara i pamti patogene s ciljem brže reakcije na pojavu istovetnog patogena u eventualnim ponovnim napadima na organizam. Biološki imuni sistem sastoji se od dva različita sistema koji deluju simultano i, nakon što se aktiviraju, mogu da se razvijaju, umnožavaju i stvaraju mehanizme za eliminaciju antigena (Timmis et al., 2008).

1. Urođeni imuni sistem[1] nastaje nakon rođenja, nasleđivanjem od majke, i nestaje u periodu od nekoliko dana do nekoliko nedelja. Ovaj sistem namenjen je da daje brz odgovor odmah nakon što prepozna izloženost napadaču (de Castro & Von Zuben, 2001).
2. Adaptivnom imunom sistemu[2] (AIS) potrebno je nekoliko nedelja da postane potpuno efektivan i efikasan. Adaptivni imuni sistem može biti humoralni i ćelijski (ćelijski-posredovani) imunitet.
    a. Humoralni imunitet targetira ekstracelularne mikroorganizme, a proces imunog odgovora je brz. Ćelije odgovorne za humoralni imunitet su B ćelije[3], limfociti koji nastaju u koštanoj srži i proizvode proteine – antitela specifična za svaki prepoznat patogen koji spolja napada organizam (antigen). Nakon aktivacije antigenom, B ćelije se dele na plazmu i antitelo koje se vezuje za antigen i na taj način ga uništava. B ćelije se razmnožavaju i proizvode ćelije plazme koje zatim luče antitela sposobna da se vezuju za antigene, i ubijaju ih, nakon čega se oni izlučuju iz organizma. B ćelije takođe proizvode memorijske ćelije koje služe da lakše bude prepoznat isti patogen, ukoliko do njegovog napada na organizam dođe u periodu koji sledi.
    b. Ćelijski imunitet targetira intracelularne mikroorganizme, ćelije tumora i ćelije transplantiranih organa. Ovaj imunitet je sporiji od humoralnog imuniteta i podrazumeva oslobađanje citokina (citotoksin), što ne uključuje stvaranje antitela nego formiranje T ćelija[4]. T ćelije potiču iz hematopoetskih matičnih

[1] Innate Immune System (IIS), engl.
[2] Adaptive Response System (ARS), engl.
[3] Od engleske reči bones (kost), prim. aut.
[4] Od engleske reči thymmus (žlezda timus), prim. aut

ćelija u koštanoj srži ali prolaze kroz žlezdu timus da bi sazrele (Broere et al., 2011). Timus se nalazi ispod vrha grudne kosti, aktivan je od rođenja, najviše se razvija u periodu puberteta, i zadržava funkcije, sa izvesnim slabljenjem, do kasnog životnog doba. Osnovna uloga T ćelija je ubijanje inficiranih ćelija izazivanjem apoptoze, odnosno ćelijske smrti. T ćelije razvijaju se tako da mogu da pripadaju jednoj od tri osnovne klase: (1) T-helper (poćnici) su neophodne za aktivaciju B ćelija, (2) T-killer (ubice) citotoksične ćelije se vezuju za strane napadače i (3) T-supressor (regulatori) ograničavaju delovanje drugih imunih ćelija da spreče autoimune bolesti. BIS takođe generiše biblioteku, odnosno bazu podataka, različitih memorijskih T ćelija specifičnih za antitela, čija je karakteristika da ostaju u telu dug vremenski period (nekoliko desetina godina) i reaguju na naknadno izlaganje istom antigenu.

3.2. Veštački imuni sistem: koncept i algoritmi

Veštački imuni sistem čini skup modela, tehnika i algoritama inspirisanih biološkim imunim sistemom, koji opisuju funkcije BIS-a, i niz drugih karakteristika i koncepta savremene imunologije, u opštem slučaju. Osnovna ideja VIS-a je formirati decentralizovani adaptivni sistem za dizajn i primenu matematičkih algoritama i tehnika koje koriste pojednostavljene modele raznovrsnih imunoloških procesa i funkcionalnosti. Koncept VIS-a može se primeniti u prepoznavanju oblika, detekciji anomalija, analizi podataka, klasifikaciji, klasteringu, mašinskom učenju, detekciji napada i upada u računarske sisteme, optimizaciji, pretraživanju, i u drugim oblastima koje su direktno ili indirektno izvedene iz karakteristika BIS-a. Za razliku od biološkog imunog sistema, veštački imuni sistem je projektovan tako da koristi optimizacione algoritme. Obzirom da je razdvajanje self/non-self ćelija (self-non-self discrimination, engl.) određeno antitelima koja se vezuju za antigene, ukoliko BIS ne može da ostvari ove funkcije, organizam može da reaguje autodestruktivno. Da bi rešio ovaj problem imuni sistem izvodi negativnu ili pozitivnu selekciju koje se, takođe, koriste kao algoritmi u VIS-u (Haag, 2007.).

3.2.1. Negativna selekcija

Negativna selekcija je mehanizam inspirisan disktriminatornim ponašanjem BIS-a koji detektuje strane antigene a ne reaguje na sopstvene ćelije. Osnovna ideja negativne selekcije, koju su prikazali Forrest et. al (1994), bazirana je na selekciji T ćelija kada one sazrevaju u timusu. T ćelije, koje nisu sazrele (formirane do kraja), a reaguju na sopstvene ćelije, imuni sistem eliminiše u procesu kontrolisanog ćelijskog odumiranja. Druge T ćelije sazrevaju, napuštaju timus, i kruže telom izvršavajći zaštitnu funkciju. U osnovi, algoritam negativne selekcije je detekcija anomalije koja se sastoji od dve faze: obučavanja i testiranja. Algoritam počinje definisanjem niza detektora koji opisuju normalno stanje sistema. Potom je potrebno generisati skup detektora koji prepoznaju isključivo komplement ovih vrednosti. Ovi detektori se kasnije primenjuju na nove podatke kako bi ih klasifikovali u one koji su sopstveni ili koji to nisu. U fazi obučavanja (trening faza) generiše se skup validnih detektora korišćenjem poznatih podataka, a druga (test faza) podrazumeva monitoring i evaluaciju onih detektora koji su bazirani na nepoznatim podacima (Ji, Dasgupta, 2007).

Trening faza

- Algoritam generiše niz detektora koji su komplementi poznatih podataka, po određenom algoritmu.
- Oni kandidati koji odgovaraju detektorima se eliminišu, a ostali postaju detektori.
- Faza treninga se završava kada su svi detektori takvi da detektuju poznate podatke.
- Detektori su takvi da detektuju one anomalije koje ukazuju na odstupanja od normalnog ponašanja (self-state, engl.).

Test faza

- Novi podaci (poznati i/ili nepoznati) se dovode na ulaz modela.
- Izvodi se poređenje nizova sa generisanim detektorima i procenjuje se kvalitet generisanog sistema.
- Ukoliko postoji odgovarajući detektor, podatak je non-self tipa.

Problemi koji se javljaju kod algoritama za negativnu selekciju su: (1) loše karakteristike u odnosu na statističke modele i (2) generisanje detektora u višedimenzionom prostoru.

3.2.2. Pozitivna selekcija

Ukoliko T ćelija ne prepoznaje sopstvene molekule, ona se testira na non-self molekulima. Ukoliko T ćelija nije mogla da ih prepozna, dolazi do apoptoze. Na ovaj način T ćelije koje reaguju sa non-self molekulima prežive. Ovaj proces je poznat kao pozitivna selekcija (Lakshmi, 2014). Ovim je omogućeno uklanjanje onih limfocita koji imaju neefektivne receptore, tako da efikasna antitela dobiju prostor da se dele i preživljavaju, što istovremeno održava i kontrolisanu veličinu njihove populacije (Haag, 2007). T ćelije koje prođu kroz timus ne reaguju na sopstvene ćelije ali reaguju na ćelije koje to nisu.

3.2.3. Klonalna selekcija

Teoriju klonalne selekcije razvio je Burnet 1959. godine. Teorija opisuje osnovu BIS-a odnosno imuni odgovor koji je moguć ukoliko B i T ćelije prepoznaju antigen na osnovu receptora, što je bazirano na komplementarnosti između antitela i antigena (de Castro, Timmis, 2002; Haag, 2007). Antitelo, takođe, može da prepozna drugo antitelo, ukoliko njihovi receptori odgovaraju jedan drugome. Ukoliko BIS ne razdvoji ova antitela, organizam može da napadne samog sebe. Iako se teorija klonalne selekcije odnosi i na B i na T ćelije ona je, u osnovi, inspirisana aktivacijom B ćelija na pojavu antigena. Kada antigen napadne organizam, B ćelije koje ga prepoznaju se naglo umnožavaju, i luče antitela. Ćelije su tako stimulisane da proizvode sopstvene klonove velikom brzinom a samoreaktivne ćelije se eliminišu (Aickelin, Dasgupta, 2005.). Ovakvo ponašanje odgovara obučavanju sa pojačavanjem (reinforcement learning, engl.), pri kojem sistem kontinualno poboljšava svoje mogućnosti da izvede određeni zadatak, u ovom slučaju da prepozna antigene (Sutton, Barto, 1998; de Castro, Von Zuben, 2001).

De Castro & Von Zuben su 2001. godine predstavili algoritam koji su nazvali CLONALG, koji sadrži prepoznavanje oblika i optimizaciju. Autori su ga opisali kao algoritam koji je u stanju da obučava skup ulaznih oblika „selekcijom, reprodukcijom i mutacijom skupa veštačkih imunih ćelija“. Ukazali su na dve bitne karakteristike pri sazrevanju B ćelija koje mogu biti korišćene u VIS-u. Prvo, nagli rast broja B ćelija proporcionalan je broju antigena na koji je potrebno uništiti, na osnovu mere koja ima naziv

afinitet (affinity, engl.), a odnosi se na sklonost ili relaciju ka kauzalnim vezama. Što je veći afinitet više je klonova proizvedeno. Drugo, mutacija antitela B ćelija je inverzno proporcionalna antigenu na koji se vezuje. Ovaj algoritam je, u osnovi, primer prepoznavanja oblika koji je sastavljen od formiranja antitela, kloniranja i hipermutacije, merenja afiniteta i selekcije. Po fazama algoritam ima sledeći oblik:

1. *Inicijalizacija* - Start algoritma je formiranje skupa oblika S koji odgovara skupu antigena. CLONALG potom proizvodi skup memorijskih antitela M, tako da odgovaraju elementima skupa S.
2. *Osnovna petlja* - Uporediti antigen i antitelo i izračunati afinitet za svako antitelo, u zavisnosti od sličnosti između antitela i antigena.
3. *Selekcija* - Skup antitela je izabran iz onog skupa antitetela koja imaju najveći afinitet sa antigenom.
4. *Kloniranje* - Skup izabranih antitela se klonira tako da bude proporcionalan afinitetu (viši afinitet – više antitela se klonira).
5. *Mutacija (affinity maturation)* - Mutirati inverzno proporcionalno one kolonove u zavisnosti od njihovog afiniteta (što manje odgovara to veći broj antitela mutira).
6. *Merenje klonalnog afiniteta* - Izračunaj vrednosti afiniteta za svako klonirano antitelo, u zavisnosti od sličnosti između kloniranog antitela i antigena.
7. *Izbor kandidata* - Antitela (originalna i klonirana) sa najvećim afinitetom se biraju tako za sledeću generaciju.
8. *Diverzitet* - Dodati broj novih slučajno generisanih antitela.
9. *Povratak u petlju* - Ponavljati proces do zadovoljenja predefinisanog kriterijuma.

3.2.4. Teorija imunih mreža

Jerneova teorija imunih mreža iz 1974. godine ukazala je i na one karakteristike BIS-a koje podrazumevaju da je BIS sastavljen od mreža međusobno povezanih ćelija i molekula koje prepoznaju jedne i druge i u onim slučajevima kada antigen nije prisutan u organizmu (de Castro, Von Zuben, 2001). Jerne je došao do zaključka da imuni sistem pokazuje određeno ponašanje i niz drugih aktivnosti koje su rezultat osobina kao što su tolerancija i memorija. Osnovu ovog principa čini tzv. idiotipska mreža (idiotypic network, engl.), u kojoj antitela postaju okidači za imuni odgovor ne samo u slučaju napada antigena već i u slučaju interakcije sa drugim antitelima. Imuni odgovor može biti pozitivan što dovodi do aktivacije ćelija i njihove deobe, ili negativan, što vodi toleranciji i supresiji imunog odgovora. Teoriju diskretnih immunih mreža predložili su Timmis i Neal (2001) a algoritam aiNet, koji odgovara teoriji imunih mreža je baziran na istim principima kao i CLONALG, i predložili su ga de Castro i Zeben (2002). Ovaj algoritam se koristi za redukciju podataka ili analizu klastera, i ima tri koraka (Ahmad, 2012):

1. *Inicijalizacija*: kreiranje inicijalne mreže iz podskupa antigena.
2. *Antigenska prezentacija:* za svaki antigenski oblik (pattern, engl.) uraditi sledeće:
    - 2.1. *Klonalna selekcija i interakcije u mreži:* za svaku ćeliju mreže, odrediti nivo njene stimulacije.
    - 2.2. *Metadinamika*: eliminisati one ćelije u mreži sa najnižom stimulacijom.
    - 2.3. *Klonalna ekspanzija*: izabrati one ćelije u mreži koje su najviše stimulisane i reprodukovati ih proporcionalno njihovoj stimulaciji.
    - 2.4. *Somatska hipermutacija*: mutirati svaki klon.
    - 2.5. *Konstrukcija mreže*: izabrati mutirane klonove i izvršiti integraciju.
3. *Ponavljanje ciklusa*: Ponoviti korak 2. do zadovoljenja zadatog kriterijuma.

3.2.5. Teorija opasnosti

Matzinger je 1994. godine predstavio novi model alarma/opasnosti. Ukazao je na mogućnost da imuni sistem ne treba da reaguje na razliku između self i non-self ćelija, već da reaguje na ono što je opasno ili nije za organizam. Osnovna ideja je da imuni sistem ne treba da reaguje na ono što nije sopstveno ali je bezopasno nego treba da reaguje na ono što je sopstveno a opasno. Po teoriji, okidač (trigger, engl.) za odziv imunog sistema je alarm – signal koji se generiše kada je detektovana opasnost. Aktivirane ćelije antigena u mogućnosti su da obezbede ko-stimulacioni signal za T-helper ćelije, koje potom kontrolišu adaptivni imuni odgovor. Signali za opasnost se emituju kroz sve ćelije u telu koje su bile oštećene napadom patogena.

3.2.6. Algoritam dendritičkih ćelija

Dendritičke ćelije su najmoćnija vrsta antigenskih ćelija koje, takođe, pronalaze nepoznate ćelije u organizmu, vare ih, i postavljaju njihove antigene na svojoj površini. Nakon toga se premeštaju u područje sa velikim brojem limfocita, kao što su limfni čvorovi i slezina. Na ovim mestima aktiviraju određene limfocite da na reakciju, tj. da napadnu date antigene u telu. U teoriji opasnosti, opasnost se meri štetom, odnosno uništavanjem ćelija koje su bile izložene signalima koji su poslati kada su ćelije umirale na neki neprirodan način (Aickelin, Cayzer, 2002). Ovi signali su detektovani dendritičkim ćelijama koje mogu biti: nezrele (immature, engl.), polu-zrele (semi-mature, engl.) ili zrele (mature, engl.), što određuje njihove funkcije. Dok ne sazri, ćelija prikuplja antigen, uz signal (bez)opasnosti iz lokalnog okurženja. Dendritička ćelija prikuplja i integriše signale iz svoje okoline i odlučuje da li je okolina opasna ili ne. U bezbednom okruženju, dendritička ćelija dobija polu-zreli oblik, prikazuje antigen T ćelijama, i inicira njihovu toleranciju. Ukoliko je okruženje opasno po organizam, dendritička ćelija sazreva i inicira T ćelije da postanu reaktivne na prisustvo antigena. Algoritam dendritičkih ćelija uvodi sledeće pojmove: signal opasnosti (danger signal, engl.), signal bezopasnosti (safe signal, engl.) i signal molekularnog oblika pavezanog sa patogenom (Pathogen associated molecular pattern signal, engl. – PAMP) koji odgovara signalu podataka u svakom trenutku izvršavanja algoritma. Na ovaj način je izbegnuto definisanje onoga što se smatra sigurnim. Istovremeno, neophodno je definisati pojmove kao što su opasnost, bezbednost i signali okruženja.

4. SISTEMI ZA ZAŠTITU OD NAPADA NA RAČUNARSKE MREŽE

Napadi na računarske mreže su maliciozne aktivnosti izvedene s ciljem kompromitacije bezbednosti mreže, njenih komponenti i integriteta, poverljivosti i dostupnosti podataka, pri čemu podaci mogu biti procesuirani (obrađivani), skladišteni i/ili prenošeni (Aissa, Guerroumi, 2016). Malicioznost ili slučajna greška može izazvati posledice po informacionu bezbednost (IB) tako da računarske mreže privremeno ili trajno stave van upotrebe. Svakodnevno otkrivanje novih ranjivosti ukazuje na potrebu za osmišljavanjem novih vidova sprečavanja napada na računarske mreže (Stanković et al., 2022).

U napadu na računarsku mrežu napadač napada jednu ili više komponenti informaciono-komunikacionih sistema (IKS) koje čine: (1) korisnik, (2) hardver, (3) softver, (4) linije veze (prenosni putevi) i (5) prateća oprema. Aktivnosti koje su izvori napada mogu nastajati pod dejstvom eksternih izvora čija je namera da obezbede pristup računarskoj mreži ili su u pitanju dejstva insajdera koji imaju mogućnost da pogrešno koriste, napadnu, promene ili ukradu informacije[5]. Značaj sistema za zaštitu od napada na računarsku mrežu (Intrusion Detection System, IDS, engl.) ogleda se, između ostalog, u tome da prepoznaje da li je napad eksterni ili interni.

Napadi na računarske mreže mogu biti klasifikovani u nekoliko grupa (Shen, 2012):

- zloupotreba (abuse, engl) (neautorizovane aktivnosti izvode autorizovani korisnici),
- izviđanje (reconnaisance, engl.) (uljez pokušava da odredi da li je sistem ili servis moguće iskoristiti u maliciozne svrhe),
- pokušaj upada (penetration attempt, engl.) (neautorizovanom aktivnošću napadač pokušava da ostvari pristup računarskim resursima),
- upad (neautorizovani korisnik uspešno pristupa računarskim resursima),
- trojanski konj (aktivni neautorizovani procesi),
- odbijanje pristupa (Denial of Service, DoS, engl.) (napad utiče na legitimne korisnike tako što ne mogu da pristupe računarskim resursima), itd.

Bez obzira na to gde je ili šta je izvor malicioznih aktivnosti, i kakav je način infiltracije u mrežu, postoje izvesni obrasci ponašanja u okviru jednog ciklusa malicioznog napada (Stallings, Brown, 2015). Obrazac čine:

- prikupljanje informacija,
- inicijalna kompormitacija mreže,
- sticanje privilegija,
- nastavak prikupljanja informacija,
- eksploatacija sistema i/ili narušavanje njegove strukture,
- održavanje privilegija i
- zametanje tragova (Rigaki, 2017).

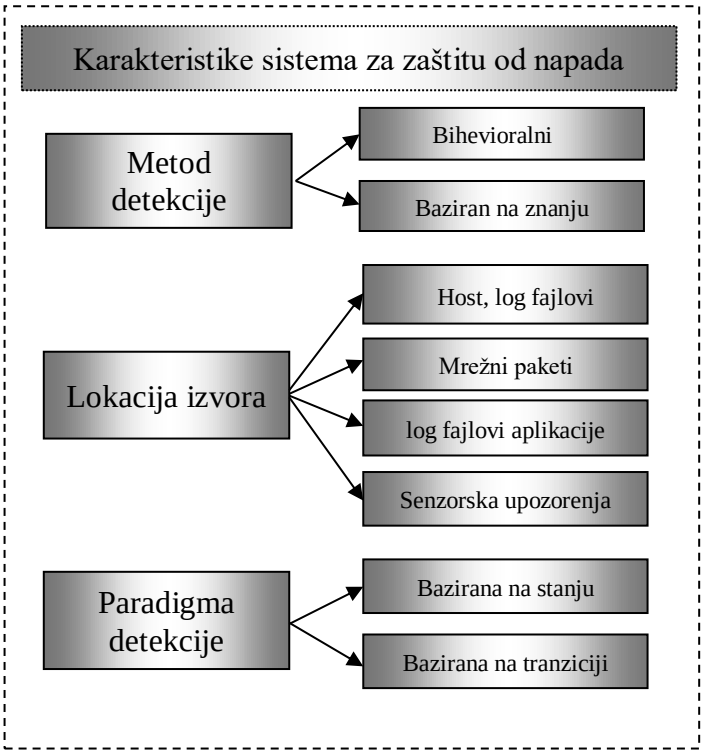
Iako su tipovi inicijalne faze kompromitacije raznovrsni, i teško je doći do privilegija, sve ostale faze su slične za svaki maliciozni napad. Treba imati u vidu da postoje situacije u kojima je teško razlikovati maliciozni napad od normalnih aktivnosti računarske mreže, tako da je jedno od osnovnih pitanja na koje treba odgovoriti je: Na koji način

[5] Insider treath (engl.)

razlikovati napad na računarsku mrežu od normalnog ponašanja mreže (Clayomb et al., 2014)? Odgovor na ovo pitanje leži kako u mogućnostima jednog sistema za zaštitu tako i u metodama koje ovaj sistem koristi da prepozna obrasce ponašanja napadača. Debar et al. (1999) uvode koncept klasifikacije IDS sistema, koji su prikazani Slikom 2.

1. Metod detekcije opisuje karakteristike analizatora. Ukoliko u IDS-u postoji tačno određen prag detekcije (anomalije), tj. ukoliko je poznat statistički model normalnog ponašanja mreže, onda je metod detekcije bihevioralan. Ukoliko sistem koristi prethodna znanja i poznate podatke o vrstama napada (partikularni napad), onda je u pitanju IDS baziran na znanju.
2. IDS baziran na lokaciji izvora razdvaja ulazne podatke koje treba da analizira (log fajlovi aplikacije ili hosta, senzorski alarmi, paketi mrežnog saobraćaja).

Paradigma detekcje opisuje mehanizam detekcije koji se koristi kod onih IDS-a koji mogu da procene stanje (sigurno/nije sigurno) ili prelazne pojave (sa bezbednog na nebezbedno ili obrnuto).



Slika 2. Sistemi za detekciju napada na računarsku mrežu

(Izvor: Debar et al., 1999)

Jedan od osnovnih problema kod instalacije IDS-a je gde ga treba postaviti u računarskoj mreži. U osnovi, rešenje ovog problema leži u topologiji mreže i namere korisnika. IDS može da bude postavljen na jedno ili više mesta, a postavljanje sistema zavisi i od toga da li se očekuje eksterni ili interni napad. Na primer, ukoliko je potrebno detektovati isključivo eksterne aktivnosti i ukoliko je veza sa Internetom ostvarena preko jednog rutera, najbolje je IDS postaviti između rutera i lokalne mreže. Na drugoj strani, ukoliko je potrebno detektovati internu malicioznu aktivnost onda je najbolje postaviti IDS u sve segmente mreže.

4.1. Funkcije sistema za detekciju napada na računarsku mrežu

Funkcije sistema za zaštitu od napada na računarske mreže moguće je klasifikovati u zavisnosti od mesta implementacije, tj. izvora podataka, kao što su host, mreža, aplikacija ili senzor (host-based, network-based, application-based, sensors-based, engl.) i po tipu indikatora na koje sistem reaguje, kao što je detekcija potpisa, anomalije ili njihova kombinacija (misuse/signature-based, anomaly-based, hybrid, engl).

4.1.1. Podela funkcija sistema po mestu implementacije

4.1.1.1. Host-based IDS

Sistemi za zaštitu od napada na računarsku mrežu implementirani na hostu (host-based IDS, HIDS, engl.) su tip IDS-a koji je implamentiran u računarske mreže (Debar, et al., 1999). HIDS je softverski proizvod koji se postavlja kao agent na jedan host s ciljem monitoringa i analize saobraćaja i svih sistemskih aktivnosti i aktivnosti vezanih za aplikacije u lokalnom sistemu (Hodo et al., 2017; Vijayarani, Sylviaa, 2015). HIDS se postavlja na one delove sistema koji su najranjiviji u smislu napada kao što je npr. web server. HIDS prikuplja invormacije sa sistemskih poziva (system calls, engl.), nadzora operativnog sistema (OS audit trails, engl.), iz log fajlova aplikacija i sistema (host system logs, application logs, engl.), s ciljem da detektuje aktivnosti napadača (Kajal, Devi, 2013). Ovaj sistem je monolitski (baziran je isključivo na reviziji ruta), a prikupljene informacije omogućuju da HIDS uoči patern pogrešnog korišćenja mreže na višem nivou apstrakcije, odnosno da uoči aktivnost koja nije vidljiva na prvi pogled.

Preciznost HIDS može biti takva da ukaže na aktivnosti napadača, komande koje je koristio, fajlove kojima je pristupao, itd. HIDS mogu biti reaktivni (aktiviraju alarm po njegovoj detekciji) ili proaktivni (osluškuju potencijalni napad na određeni host i reaguju kada do napada dođe, u realnom vremenu) (Rehman, 2003).

Prednost HIDS je što može da detektuje pretnje unutar mreže, skeniranjem saobraćaja pre slanja ili prijema podataka. Osnovni nedostatak je nadziranje isključivo jednog hosta. Primeri HIDS su Tripwire i OSSSEC (Mallissery et al., 2011; Kajal, Devi, 2013).

4.1.1.2. Network-based IDS

Network-based IDS (NIDS) detektuje napade na različitim sistemima u računarskoj mreži. Postavlja se na posebne tačke s ciljem analize niza paketa koji prolaze kroz linkove u mreži. NIDS koristi mrežni saobraćaj (žični ili bežični) da detektuje sumnjive aktivnosti ili nedozvoljeni pristup resursima mreže (Kajal, Devi, 2013). NIDS je moguće kategorizovati u tri grupe: monolitski (veliki, celi blokovi), hijerarhijski ili kooperativni (Kim, Bentley, 1999).

Kod monolitskog pristupa, jedan NIDS, postavljen na jednom serveru, nadzire softver koji je aktivan na više lokalnih hostova. Nadzirani lokalni hostovi šalju centralnom hostu zbirne, detaljne, hronološke, vremenski potpisane, sekvencijalne zapise o događajima (audit trails, engl.), i dalje glavnom serveru, koji nakon toga vrši njihovu analizu. Ovakav pristup je uglavnom prisutan kod malih računarskih mreža. Međutim, za skalabilnost, robusnost i prilagodljivost, ovaj sistem ima niz (kritičnih) nedostataka. Kako mreža raste, audit trails treba preneti od lokalnih hostova do centralnog servera, što može da degradira performanse mreže. Drugi problem je prekid rada centralnog servera, što može da ugrozi

celu mrežu. I, treće, da bi sistem radio, centralni setver mora da bude uniformno konfigurisan za različite zahteve lokalnih hostova.

Hijerarhijski pristup nadomešta nedostatke monolitskog pristupa jer je dizajniran tako da radi na velikim računarskim mrežama (nekoliko hiljada hostova). Po ovom pristupu definisan je određen broj hijerarhijskih (pod)oblasti za nadzor a svaki IDS nadzire jednu oblast tako da, umesto da se podaci šalju jednom hostu, prvo se izvodi lokalna analiza, pa se rezultat šalje na prvi viši hijerarhijski nivo. Skalabilnost je, na ovaj način, postignuta raspodelom na lokalne podoblasti. Međutim, mogu se pojaviti drugi problemi, kao što je na primer problem promene lokalnih hostova ukoliko dođe do promene topologije mreže. Pored toga, ukoliko je napadnut host višeg hijerarhijskog nivoa, moguće je da svi hostovi na nižim nivoima budu ugroženi.

Kooperativni pristup je baziran na distribuiranju odgovornosti jednog centralnog servera na niz kooperativnih IDS-ova. Svaki od njih je odgovoran za nadzor malog dela hosta, a oni međusobno komuniciraju i funkcionišu udruženo (kooperacija), čime prave koherentni sistem koji odlučuje globalno. Za razliku od hijerarhijskog pristupa, ovde nema hijerarhije između distribuiranih lokalnih hostova. Međutim, javlja se problem održavanja i efikasnosti jer postoji mnogo komunikacionih mehanizama, mehanizama monitoringa, pad sistema je moguć i slično. Prednost ovog IDS-a je postojanje jednog sistema koji vrši nadzor nad celom mrežom, štedeći vreme i cenu jer se ne postavlja na svaki host.

Glavna mana NIDS je njegova osetljivost na napad sa računarske mreže koju nadzire (Chandola et al., 2009). Najpoznatiji NIDS su Cisco Secure IDS, BRO, SNORT, Dragon, itd.

4.1.1.3. Application-based IDS

Application-based IDS (AIDS) je podskup HIDS-a, koji analizira događaje u okviru applikacija, prateći interakciju korisnik-aplikacija. AIDS prati one aplikacije koje imaju pristup kriptozaštićenim podacima, jer su krajnje tačke u transakcijama, a informacije su tipa otvorenog teksta[6]. Uobičajeno, AIDS se postavlja između web servera i sistema za upravljanje bazama podataka (database management sistem, engl.), i nadzire SQL protokol. AIDS je sličan antivirusu koji je dizajniran specijalno da nadzire mail server (Shimonsky, 2002).

4.1.1.4. Uporedni prikaz HIDS, NIDS i AIDS

Osnovna prednost HIDS-a nad drugim detektorima napada je da je on ekstremno snažan alat za analizu potencijalnog napada koji može da tačno signalizira korake napadača, korišćene komande, fajlove, upite ka sistemu i slično. Na ovaj način HIDS obezbeđuje detaljnije i relevantnije informacije od NIDS. Na drugoj strani, njegova osnovna mana je što mora da bude instaliran na svakom hostu.

NIDS može da detektuje neke od napada koji koriste mrežu, i dobri su u detekciji napada tipa neautorizovanog pristupa. NIDS ne zahtevaju modifikaciju ili nove servere i hostove, lako se instaliraju i postavljaju u mrežu na mesto na kojem treba da prate saobraćaj. Mana ovog sistema je da može da prati saobraćaj u mreži isključivo na segmentu na koji je postavljen i ne može da detektuje napad na drugim delovima mreže (lokalizovana vizija), što je posebno nepovoljno kod brzih ethernet mreža. Ovakav sistem ima problem i kada je potrebno nadzirati deo mreže u kojem je protok najveći.

[6] Otvoreni tekst je izraz koji opisuje podatke koji nisu zaštićeni.

AIDS, kao posebna podvrsta HIDS-a, ima sve prednosti i mane koje ima HIDS ali, obzirom da je AIDS aktivan kod kriptozaštićenih podataka i koristi odgovarajuće aplikacije za zaštitu, ranjiviji je na napad i troši više resursa.

Prednosti i mane Host-based, Network-based i Application-based sistema za zaštitu podataka, prikazane su Tabelom 4.

Tabela 4. HIDS, NIDS i AIDS – osnovne prednosti i mane

Implementacija	Prednosti	Mane
HIDS	Detaljno nadzire aktivnosti hosta. Identifikuje korisnika koji izvodi napad. Detektuje napade koje ne detektuje NIDS. Može da koristi enkripcione servise za ispitivanje zaštićenog saobraćaja, podataka, aktivnosti i memorije. Radi bez ograničenja na switch-based mrežama.	Podatke prikuplja na jednom hostu, upis u log fajlove ili izveštavanje usporava mrežni saobraćaj. Kompromitacija hosta može da obezbedi izvođenje DoS napada. Veliko procesno vreme, memorija i drugi resursi.
NIDS	Nadzire veliku mrežu ukoliko se postavi na nekoliko čvorova u mreži. Pasivan uređaj, ne utiče na rad mreže. Često ga napadač ne detektuje. Lak za instaliranje.	Ne može da analizira saobraćaj na velikim mrežama; može da previdi napad kod visokog saobraćaja. Ne može efikasno da nazire switch-based (high-speed) mreže. Ne može da analizira zaštićene podatke. Zahteva manuelno angažovanje administratora.
AIDS	Koncentrisan na aplikaciju. Detektuje napad analizom log fajlova. Prate neautorizovan pristup. Rade sa kriptozaštićenim podacima korišćenjem application-based encryption/decryption servisa.	Ranjiviji na napad od HIDS. Troše značajne resurse aplikacije ili hosta.

(Izvor: Shimonsky, 2002)

4.1.2. Podela funkcija po tipu događaja na koje sistem reaguje

4.1.2.1. Sistem za detekciju potpisa

Detekcija po najavi/nastanku (detection by appearance, engl.), odnosno detekcija potpisa (signature-based, engl.) ili detekcija pogrešnog korišćenja (misusse-based, engl.) bazirana je na skupu pravila koja se koriste da bi bilo provereno preklapanje mrežnog saobraćaja sa poznatim oblicima napada. Ukoliko je detektovano odstupanje od normalnog mrežnog saobraćaja generiše se alarm (Zhengbing et al., 2008). Ovo su sistemi koji koriste potpise[7] ili indikatore ekstrahovane iz prethodno poznatih napada, odnosno koriste raznovrsne tehnike da pronađu poklapanje sistemskih aktivnosti sa poznatim napadima čiji se potpisi nalaze u, za to namenjenim, bazama podataka. Potpise je potrebno manuelno generisati svaki put kada se pojavi novi napad tj. potreban je redovan update baze (Rigaki, 2014).

Signature-based IDS mogu se pojaviti u tri oblika, kao: (1) modeli stanja (state models/modelling, engl.), (2) ekspertski sistemi (expert systems, engl.) ili poklapanje nizova (string matching, engl.) U ovom tipu detekcije, napad prati dobro definisane paterne koji se odnose na slabosti sistema i aplikativnog softvera. Detekcija partikularnog napada zahteva specifično znanje o ponašanju koje podrazumeva određeni napad.

[7] Potpis (signature, engl.) je deo, oblik (patern) koji se nalazi unutar paketa sa podacima. Koristi se za detekciju više tipova napada, a nalazi se u na primer zaglavljima Inernet-Protokol (IP) adresa, zaglavljima transportnog sloja (TCP, UDP) i/ili u zaglavljima aplikacija (payload, engl.).

Osnovna prednost ovakvih sistema je nizak broj lažno pozitivnih detekcija, sve dok je napad poznat unapred. Takođe, sistem se lako instalira i koristi (Kumar, Sangwan, 2012).

Mane sistema su da ne može da detektuje nove/nepoznate napade, ukoliko njihovi potpisi nisu upisani u bazu poznatih napada. Takođe, postoji i problem napada koji su veoma slični (varijeteti jednog napada) ali ih detektor ne prepoznaje, iako su njihovi potpisi neznatno različiti (Kajal, Devi, 2013).

4.1.2.2. Sistem za detekciju anomalije

Detekcija anomalije u ponašanju mreže (anomaly detection, detection by behaviour, engl.) izvodi se skeniranjem ili paketa (packet-based, engl.) ili protoka (flow-based, engl.) u mrežnom saobraćaju. U prvom slučaju, detektor analizira mrežni saobraćaj paket po paket tražeći anomaliju. Međutim, ovo rešenje postaje nepraktično kada broj paketa raste. U tom slučaju koristi se odabiranje (sampling, engl.), tako da detektor proverava samo deo paketa, koji se generišu u određenim vremenskim intervalima (slotovi) i na tačno određenim delovima računarske mreže. Određivanje vremenskih slotova treba da bude pažljivo izvedeno jer, ukoliko je slot prevelik, moguće preskočiti anomaliju, dok se za uske vremenske slotove pojavljuje granularnost, odnosno, rezultat dobijen u slotu ne razlikuje se mnogo u odnosu na skeniranje svih paketa (Al-Haj Baddar et al., 2014). Kod detekcije anomalija koje su bazirane na protoku, detektor snima saobraćaj u mreži između određenih čvorova. I kod ovog tipa detekcije, ukoliko je intenzitet saobraćaja visok, moguće je koristiti vremenske slotove.

Oba tipa prethodno navedenih detektora su online tipa. Međutim, moguće ih je upotrebljavati na podacima koji se prikupljaju, beleže i analiziraju kasnije. Takođe, oba rešenja su primenljiva kako na mrežnom, tako i na aplikativnom nivou.

U osnovi se detekcija anomalije bazira na analizi ponašanja računarske mreže, tako da je IDS moguće posmatrati kao trokoračni sistem (Mittal, 2016):

- IDS je treniran u zavisnosti od računarske mreže i zahteva korisnika,
- sistem procenjuje anomalije na osnovu predefinisanih uslova i
- podnošenje zahteva za administratorsku podršku kod pojava koje su nelogične u odnosu na karakteristike obučenog sistema, radi update-a, kasnije analize, promene strukture mreže, promene korisnika, i slično.

U prvom koraku, detektor prati rad računarske mreže u određenom vremenskom periodu (najčešće 24/7 nekoliko dana uzastopno). Na ovaj način sistem procenjuje tipično ponašanje računarske mreže, a kompanija koja formira sistem za detekciju anomalije stavlja ekspertima na uvid i raspolaganje svoju računarsku mrežu i ponašanje svih korisnika (radno vreme, lozinke, aplikacije, itd) i sistema (OS, protok, komunikaciju sa spoljašnjim okruženjem, i sl.).

Detektor sam gradi prag koji će koristiti kao meru za razlikovanje normalnog i sumnjivog ponašanja mreže, prema predefinisanoj metodi obučavanja. Nakon što je određen statistički model normalnog ponašanja mreže, sa stanovišta detektora, svako drugo ponašanje se smatra anomalijom. Odrediti normalno ponašanje mreže može se izvesti samoobučavanjem, na osnovu primera, ili programiranjem, koje izvodi stručna osoba, prema postavljenim zahtevima klijenta.

Osnovna prednost ovih detektora je mogućnost detekcije nepoznatog napada (Modi et al., 2013; Protic, Stankovic, 2020; Marković et al., 2020), dok su osnovne mane generisanje velikog broja lažno pozitivnih alarma i veliki obim podataka obučavajućeg skupa (Protic et al., 2022).

4.1.2.3. Prednosti i mane signature-based i anomaly-based IDS

Uporedni prikaz prednosti i mana detektora baziranih na potpisu i detektora anomalije u računarskoj mreži dat je Tabelom 5.

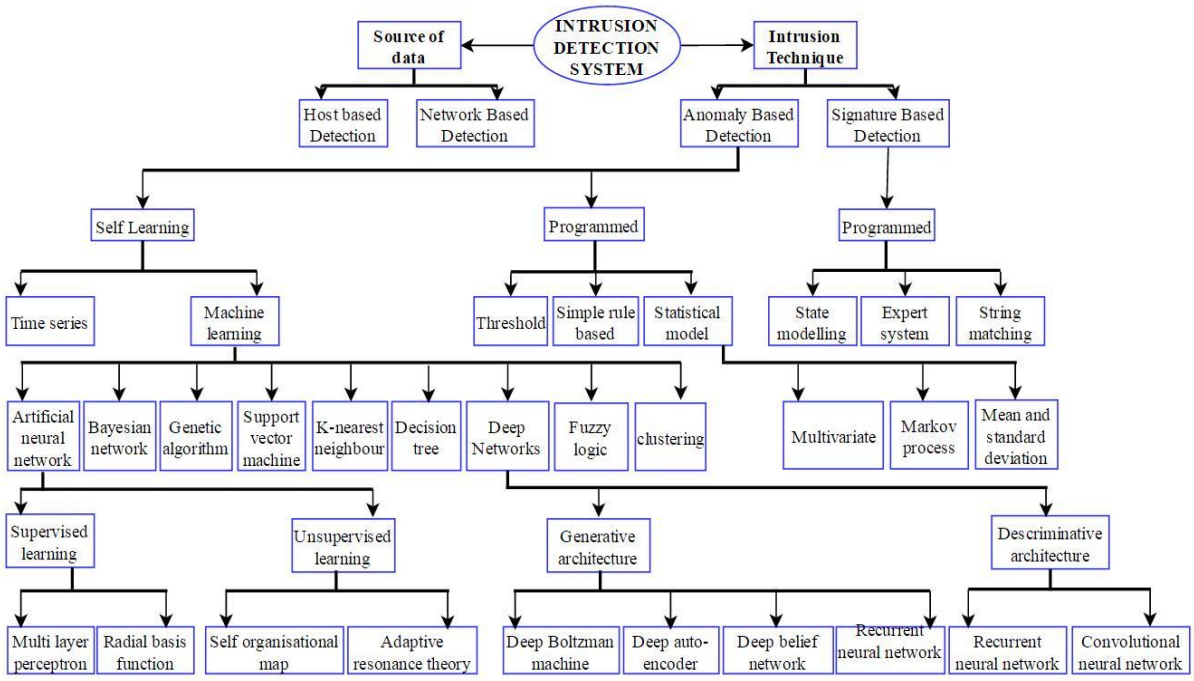
Tabela 5. Uporedni prikaz signature-based i anomaly-based IDS

Indikator	Prednosti	Mane
Potpis	Ispituje prenos podataka, aktivnost, transakcije i ponašanje tražeći poznate malicioznosti. Zahteva pristup bazi potpisa poznatih napada, radi poređenja potpisa.	Čest update baze potpisa poznatih napada. Neprepoznavanje nepoznatih napada.
Anomaija	Ispituje prenos podataka, aktivnost, transakcije i ponašanje tražeći anomalije na mreži ili u sistemu koje mogu da budu indikatori napada. Osnovni princip rada detektora: napad je sve što nije normalan saobraćaj u mreži. Može da odredi odstupanje sistema od statističkog modela normalnog ponašanja. Detektuje nepoznat napad.	Mogućnost česte promene normalnog ponašanja mreže i veliki broj lažno pozitivnih alarma. Intenzivno analitičko ponašanje može da izazove preopterećenje mreže na kojoj radi. Potrebno je određeno vreme za kreiranje statistički značajne margine koja razdvaja normalno ponašanje od anomalije.

(Izvor: Shimonsky, 2002)

4.2. Struktura sistema za detekciju napada na računarsku mrežu

Istraživanje teme strukture sistema za detekciju napada na računarske mreže, postalo je izuzetno aktuelno poslednjih decenija, zbog ubrzanog razvoja računarskih sistema, pada cena komponenti i sve dostupnijeg brzog prenosa i procesuiranja podataka. Hodo et al. (2017) daju prikaz strukture sistema za zaštitu računarskih mreža, kao na Slici 3.



Slika 3. Klasifikacija IDS-a

(Izvor: Hodo et al., 2017)

Autori prvu podelu u ovim sistemima izvode prema izvoru podataka (host-based, network-based) i prema tehnikama detekcije napada na mrežu (anomaly-based, signature-based), što je prikazano u prethodnom delu teksta, a potom tehnike napada dele na sledeći način:

- detekcija anomalija definiše normalne aktivnosti a sve drugo se smatra anomalijom, dok se obučavanje se izvodi:
  - samoobučavanjem (na osnovu vremenskih serija posmatranih procesa) ili
  - programiranjem (model praga za minimizaciju broja lažno negativnih alarma, model jednostavnih pravila po principu okidača i statistički modeli srednje vrednosti, odstupanja, korelacije i sl.) i
- detekcija potpisa koja definiše skup pravila korišćenih za proveru poklapanja oblika koja se izvodi:
  - modelovanjem stanja (prepoznavanje napada preko velikog broja različitih stanja u konačnom skupu svih stanja) ili
  - ekspertskim sistemima (sadrže skup pravila koja se koriste za opis scenarija napada poznatih sistemu tipa ulančavanja) i poklapanjem nizova (slično ekspertskim sistemima ali rade tako da nepoklapanje ne izaziva istovremeno i proces donošenja odluke o napadu).

5. BINARNI KLASIFIKATORI BAZIRANI NA NADGLEDANOM MAŠINSKOM UČENJU

Mašinsko učenje je podoblast računarskih nauka, koja podrazumeva metode kojima računari dobijaju mogućnost da uče bez da su eksplicitno programirani. U osnovi, to je proces konstrukcije algoritama, koji se ne formiraju programskim kodom, već modeli nastaju kao rezultat predikcije i odlučivanja, na osnovu dostupnih podataka. Mitchell (1997) definiše mašinsko učenje na sledeći način: „Za računarski program se kaže da uči iz iskustva E (experience, engl.) u odnosu na klasu zadataka T (tasks, engl.) i mera određenih karakteristika P (properties, engl.), ukoliko njegove karakteristike po zadacima iz T, ako su merene sa P, budu poboljšane iskustvom E.“ U biti, mašinsko učenje je proces automatizacije programiranja koji omogućava samoprogramiranje i smanjuje problem pisanja softvera. Iz navedenih razloga, mašinsko učenje je u relaciji sa statističkim proračunima fokusiranim na predikciju, optimizaciju i analizu podataka. U ovim slučajevima važi generalizacija, tj. mogućnost da model bude obučen tako da daje ispravne rezultate i u slučaju novih/nepoznatih ulaznih parametara. Problem koji se javlja određuje međuzavisnost kompleksnosti, efektivnosti i efikasnosti, odn. smanjenje kompleksnosti na račun efektivnosti ili efikasnosti modela, i obrnuto. Osnovni razlog zbog kojeg je povoljno koristiti mašinsko učenje je što su primenjeni algoritmi adaptivni u dinamičkom okruženju, optimizacija zadatih kriterijuma se izvodi na osnovu uzorka ili prethodnih iskustava, a zaključivanje se uglavnom izvodi statistički.

5.1. Tipovi mašinskog učenja

Po stilu obučavanja modela, Milosavljević (2017) deli mašinsko učenje u četiri grupe:

1. nadgledano ili induktivno učenje (supevised learning, engl.),
2. nenagledano učenje ili samoobučavanje (unsupervised learning, engl.),
3. semi-induktivno učenje (semi-supervised learning, engl) i
4. obučavanje sa pojačavanjem (reinforcement learning, engl.).

Nadgledano mašinsko učenje podrazumeva evaluaciju modela na osnovu poznatog trening skupa u kojem su instancama dodeljene labele. Kod ovako treniranih modela, nepoznatim instancama se dodeljuju poznate labele. Obučavajući podaci istovremeno sadrže i željeni izlaz. Induktivno učenje se često koristi u predikciji, ekstrakciji znanja na osnovu poznatih pravila, kompresiji, i slično.

Algoritmi za nenadgledano mašinsko učenje treniraju modele na osnovu neoznačenih instanci, a podaci za trening modela ne sadrže željeni izlaz. Samoobučavanje znači učenje iz koncepta odgovora na pitanje kakvo je ponašanje sistema koje je se može nazvati normalnim. Ovim učenjem moguće je izvoditi klasterovanje, a upotreba je česta kod kompresije slike tj. kvantizacije boja u kompresiji, u bioinformatici, itd.

Semi-induktivno učenje je slično induktivnom učenju ali podrazumeva da pored označenih instanci u procesu evaluacije modela dominiraju instance koje ne sadrže labele. Ulazni podaci su kombinacija instanci sa i bez labela, postoji željeni problem predikcije ali model mora da nauči strukture kako bi organizovao podatke i obezbedio adekvatnu predikciju. Primeri problema koji mogu biti rešeni na ovaj način su klasifikacija ili regresija.

Učenje sa pojačavanjem je učenje strategije na osnovu sekvenci izlaza. Nije na raspolaganju informacija o izlazu, već zakasnela nagrada. Karakteristični primeri za ovaj vid učenja su igre, višestruki agenti u delimično observabilnom ambijentu (okruženju), roboti u lavirintu, i slično.

Po pripadnosti funkcionalnim grupama, algoritmi mašinskog učenja mogu biti kvantizacija vektora obučavanja bazirana na neuronskim mrežama, instance-based metode i slično.

Po metodama sa istim imenima koje opisu problem ili klasu, mašinsko učenje može biti regresija, klastering, i dr.

Po sličnosti u smislu njihovih funkcija algoritme mašinskog učenja je moguće grupisati u odnosu na način na koji rade (neuronske mreže, stabla odlučivanja)[8] (Brownlee, 2013):

1. Regresioni algoritmi predstavljaju modelovanje relacija između promenljivih koje su iterativno određene korišćenjem merenja greške predikcije. Najpoznatije regresione metode su: (Ordinary) Least Squares regresija, Linearna regresija, Logistička regresija, Stepwise regresija, Multivariate Adaptive Regression Splines i Locally Estimated Scaterplot Smoothnig.
2. Instance-based algoritmi rešavaju problem odlučivanja na instancama obučavajućeg skupa. Kod ovakvih modela tipično je formiranje baze podataka koja se upoređuje sa novim podacima korišćenjem mere sličnosti, tako da se nađe odgovarajući podatak i izvede predikcija. Najpoznatiji algoritmi ovog tipa su: k-NNN, Learning Vector Quantization, Self-Organizing Map i Locally Weighted Learning.
3. Algoritmi sa regularizacijom – ekstenzija drugih algoritama (tipično regresije) su regularizacioni algoritmi, bazirani na „kažnjavanju“, tako da model uklanjanjem određenih parametara smanjuje kompleksnost. Najpoznatiji algoritmi su: Ridge Regression, Least Absolute Shrinkage and Selection Operator, Elastic Net i Least-Angle Regression.
4. DT algoritmi – metod odlučivanja formira model odluka koje su zasnovane na stvarnim vrednostima atributa. Odluke se donose u strukturi tipa stabla dok god se predikcijom ne dobije zadovoljavajuća vrednost. Ovaj tip predikcije se uglavnom koristi kod klasifikacije ili regresije. Najpopularniji DT algoritmi su Classification and Regression Tree, Iterative Dichotomiser 3 (ID3), C4.5 i C5.0, Chi-squared Automatic Interaction Detection, Decision Qtump, M5, itd.
5. Bajesovi algoritmi primenjuju Bajesovu teoremu za probleme regresije i klasifikacije. Najpopularnaji algoritmi iz ova grupe su NB, Gausov NB, Multinomial NB, Averadged One-Dependence Estimators, Bayesian (Belief) Network itd.
6. Algoritmi za klasterovanje – opisuju klasu problema i klasu metoda. Klastering metode su uglavnom organizovane po pristupu modelovanja kao što su npr. model baziran na centru ili hijerarhijski model. Metode klasterovanja su: k-Means, k-Medians, Expectation Maximization i hijerarhijsko klasterovanje.
7. Algoritmi sa pridruženim pravilima obučavanja (association learning rule methods, engl.) – određuju relacije između promenljivih u okviru podataka. Na ovaj način moguće je otkriti značajna pravila u komercijalnim, multidimenzionalnim bazama podataka. Najpoznatiji algoritmi su: Apriori Algorithm i Eclat Algorithm.
8. Veštačke neuronske mreže su inspirisane strukturom i funkcijama centralnog nervnog sistema. One su klasa prepoznavanja oblika koja se uglavnom koristi u regresiji i klasifikaciji, ali im je primena velika i u drugim oblastima. Najpoznatiji

[8] Nazivi nekih algoritama u ovom poglavlju nisu prevedeni sa engleskog jezika, prim.aut.

algoritmi ove grupe su: MLP, Back-Propagation (BP), Hopfieldove mreže, Radial Basis mreže, itd.

9. Deep-Learning algoritmi su nadogradnja vaštačkih neuronskih mreža (veće i kompleksnije mreže), koje su vezane za probleme za semi-induktivnog učenja, kod skupova podataka koje sadrže mali broj označenih podataka. Algoritmi deep-learning-a su: Deep Boltzman Machine, Deep Belief Networks, Convolutional Neural Network i Stacked Aut-Encoders.
10. Algoritmi za smanjenje dimenzionalnosti – koriste inherentnu strukturu u podacima na nenadgledani način, ili zbirno opisuju podatke korišćenjem malog broja informacija. Ovo je korisno kada je potrebno da se pojednostave podaci koji tada mogu da budu korišćeni u nadgledanom obučavanju. Mnoge od ovih metoda su adaptibilne, pa mogu da postanu primenjljive u regresiji ili klasifikaciji. Algoritmi su: PCA, Principal Component Regression, Partial Least Squares (LS) Regression, Sammon Mapping, Multidimensional Scaling, Projection Pursuit, Linear Discriminant Analysis (LDA), Mixture Discriminant Analyisis, Quadratic Discriminant Analysis i Flexibile Discriminant Analysis.
11. Ensemble algoritmi – algoritmi ansambla su modeli koji se sastoje od grupe slabijih modela koji su nezavisno obučavani i čije su predikcije na neki način kombinovane, tako da se dobije konačna predikcija. Veoma je bitno odrediti, u ovim slučajevima, kakvi/koji su slabiji modeli koje treba koristiti u datom slučaju. Ova popularna tehnika ima sledeće primenjene algoritme: Boosting, Bagging (Bootstrapped Aggregation), AdaBoost, Stacked Generalization (blending), Gradient Boosting Machines, Gradient Boosting Regression Trees i Random Forest.

U nabrojanoj podeli nisu opisani algoritmi sa specijalnim funkcijama u procesu mašinskog učenja, kao što su: selekcija atributa (feature selection, engl.), algoritmi za procenu tačnosti (accuracy evaluation, engl.) ili merenje karakteristika (performance measures, engl.). Takođe nisu prikazani algoritmi koji pripadaju posebnim podoblastima mašinskog učenja tipa evolutivnih algoritama, computer-vision algoritama, natural language processing (NLP) algoritama, grafičkih modela i slično.

U tekstu koji sledi posebno su opisani modeli nadgledanog mašinskog učenja koji su korišćeni u ovom istraživanju kao binarni klasifikatori u detekciji anomalija u računarskim mrežama, i to:

- model k-najbližih suseda,
- model k-najbližih suseda sa težinskim koeficijentima,
- model vektora oslonca,
- stabla odlučivanja i
- feedforward neuronska mreža.

5.2. Obučavanje modela

Primena algoritama mašinskog učenja je deo procesa modelovanja u kom se parametri modela podešavaju tako da bude zadovoljen unapred zadati kriterijum. U tom slučaju je moguće prenosnu funkciju svakog modela opisati izrazom (1) (Protić, Milosavljević, 2005).

$$g(\boldsymbol{y}, \boldsymbol{\delta}) = \varepsilon \tag{1}$$

pri čemu su: $\boldsymbol{\delta}$ – ulazni vektor, $\boldsymbol{y}$ – izlazni vektor, ε – zaustavni kriterijum i g – prenosna funkcija.

Međutim, ovim algoritmima je, u određenom broju slučajeva, teško opisati neke vrste zadataka koji su ljudima u svakodnevnom životu lako rešivi (npr. prepoznavanje lica) tako da, i pored toga što je modele moguće trenirati, to ne znači da su oni lako interpretabilni (npr. zahtevaju pristup velikim bazama podataka). Kod modelovanja u nadgledanom mašinskom učenju, algoritmu su poznati ulazni podaci i željene vrednosti izlaza, čime se postiže evaluacija određenog modela. Kod nenadgledanog mašinskog učenja nisu dostupni izlazni podaci nego algoritam savladava dodelu u procesu učenja, uočavanjem zakonitosti koje su mu dostupne u podacima.

Dva se osnovna problema javljaju u procesu modelovanja:

1. overfitting (pretreniranost) i
2. underfiting (nedovoljna treniranost/podešenost).

U prvom slučaju model nauči da prepoznaje instance obučavajućeg skupa ali nije u mogućnosti da prepozna instance koje se veoma malo razlikuju od naučenog. Rešenje je da algoritam bude imun na pamćenje koje mu je dostupno iz celog obučavajućeg skupa.

U drugom slučaju, model ne uspeva da izvede aproksimaciju podataka iz trening skupa (bias-variance algoritam). Dolazi do preterane kompleksnosti modela i degradacije generalizacionih osobina algoritma.

Da bi ovi problemi bili izbegnuti, neophodno je voditi računa o načinu na koji će biti zaustavljen trening modela, odnoso treba voditi računa o pravilnom izboru zaustavnog kriterijuma.

5.3. Kriterijumi zaustavljanja

Pored izbora tipa modela, jedan od osnovnih problema kod njegovog obučavanja je i metrika koja određuje zaustavni kriterijum. Ovaj problem se rešava merenjem:

- sličnosti, korišćenjem Teorije vektorskih prostora (detaljno objašnjeno u Prilogu 1) ili
- distance (rastojanja), koja je opisana u tekstu koji sledi (Bjelica, 2009).

Ukoliko su poznati vektorski prostor Ω i vektori $\mathbf{x} = [x_1, x_2, \cdots, x_n]$, $\mathbf{y} = [y_1, y_2, \cdots, y_n]$ i $\mathbf{z} = [z_1, z_2, \cdots, z_n]$, onda je funkciju f, koja predstavlja meru distance, moguće interpretirati kao rastojanje između (njenih) argumenata. Odnosno, što su dva vektora sličnija, manje je rastojanje između njih u posmatranom vektroskom prostoru. Ukoliko važi sledeće (2) – (4):

$$f(\mathbf{x}, \mathbf{y}) \geq 0 \tag{2}$$

$$f(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow x = y \tag{3}$$

$$f(\mathbf{x}, \mathbf{y}) = f(\mathbf{y}, \mathbf{x}) \tag{4}$$

$$f(\mathbf{x}, \mathbf{y}) + f(\mathbf{x}, \mathbf{z}) \geq f(\mathbf{x}, \mathbf{z}) \tag{5}$$

moguće je primeniti niz metričkih funkcija kako bi bio zadovoljen kriterijum zaustavljanja. Četiri tipa metrike opisane su u tekstu koji sledi, s namerom da budu korišćene kao osnova za izvođenje prenosnih funkcija modela, metoda i eksperimenata prikazanih u radu.

1. Rastojanje Minkowskog je najčešće korišćena mera distance, data formulom (6):

$$f(\mathbf{x}, \mathbf{y}) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}, \quad p \in R \ \text{ i } \ p \geq 1 \tag{6}$$

Ukoliko je p = 1, distanca Minkowskog postaje Manhattan distanca. Metrika supremuma računa se ukoliko $p \to \infty$. Za p = 2, rastojanje Minkowskog postaje Euklidovo rastojanje, koje je prikazano izrazom (7):

$$f(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{7}$$

2. Mahalanisovo rastojanje je ponderisana varijanta Euklidovog rastojanja oblika:

$$f(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} w_i (x_i - y_i)^2} \tag{8}$$

pri čemu su w_i, i = 1, ..., n težinski koeficijenti.

3. Gausovo rastojanje je ima oblik (9):

$$f(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \mathrm{e}^{\left[-\frac{(x_i - y_i)^2}{2\sigma_i^2}\right]} \tag{9}$$

pri čemu su σ_i, i = 1, ..., n podesivi parametri.

5.4. Binarni klasifkatori

Klasifikacija predstavlja razvrstavanje nepoznate ulazne instance u jednu od predefinisanih klasa. U ljudskoj prirodi postoji sklonost automatske klasifikacije koja je delimično nasleđena a delimično se stiče (kupovina najboljeg proizvoda, analiza slike, dijagnostikovanje bolesti, i slično). Klasifikacija se zasniva na pronalaženju sličnosti sa poznatim objektima koji su elementi klasa sa zajedničkim karakteristikama. U tom smislu, proces klasifikacije se može podeliti u dve faze:

- izgradnja modela na osnovu karakteristika objekata čija je klasifikacija poznata i
- testiranje modela radi procene njegove tačnosti.

Tek nakon što je model obučen i testiran, i nakon što je utvrđeno da je nepoznati podatak zaista razvrstan u poznatu, unapred zadatu klasu, model je moguće smatrati dobrim klasifikatorom.

U nadgledanom mašinskom učenju su ulazni parametri algoritma obučavanja modela poznate instance trening skupa iz kojih model uči, i odgovarajuće labele poznatih izlaza. Cilj učenja je formirati model na osnovu kojih će novi, nepoznati ulazni podaci biti svrstani u odgovarajuće klase, prema unapred zadatom kriterijumu, koji određuje pravila dodele željenoj klasi. Ukoliko se ciljanom funkcijom[9] označi preslikavanje ulaza na izlaz onda se nadgledano mašinsko učenje može smatrati obučavanjem (matematičkog/statističkog) modela koji daje najbolju aproksimaciju ciljane funkcije. U procesu učenja, na osnovu poznatih ulazno-izlaznih podataka model se trenira tako da je njegova generalizacija[10]

[9] Poznato i kao funkcija cilja (prim. aut.).

[10] Generalizacija je proces u kome se, po Novakoviću (2013), znanje koje važi za neki skup slučajeva prenosi na određeni nad-skup (prim. aut.).

maksimalna (prenosna funkcija je najbolja aproksimacija ciljane funkcije) da bi, nakon toga, model bio testiran na nepoznatim podacima. Kod klasifikacije, ciljane klase su diskretne a atributi (features, engl.) su, u opštem slučaju, kategoričkog tipa.

U tekstu koji sledi detaljnije su prikazana su sledeći tipovi klasifikacije koji su primenjeni u radu:

- vektori oslonca,
- k-najbližih suseda,
- k-najbližih suseda sa težinskim koeficijentima,
- stabla odlučivanja i
- feedforward neuronska mreža.

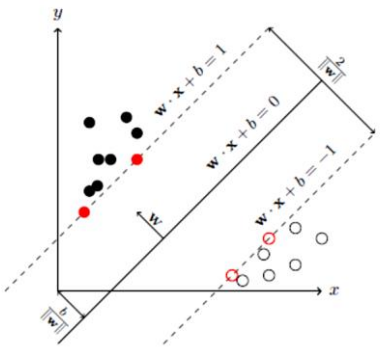
5.4.1. Vektori oslonca

Algoritmi bazirani na vektorima oslonca su algoritmi sa nadgledanim učenjem modela za klasifikacionu ili regresionu analizu, kod podataka velikog obima. Obučeni model deli ulazne podatke u jednu od klasa po principu binarne klasifikacije. Osnovna ideja SVM je naći hiper ravan u kojoj su podaci trenirani kao tačke u prostoru, koja deli podatke tako da su svi podaci iz date klase sa iste strane hiper ravni (Burgess, 1998; Dimic et al., 2017). Postoji izvesna mogućnost i da se SVM modeli koriste u nelinearnoj klasifikaciji preslikavanjem ulaza u višedimenzionalni prostor korišćenjem kernel funkcije. Softverski kod SVM algoritma prikazan je u Prilogu 2.

Cilj linearne SVM klasifikacije je da, nakon što je model obučen, nepoznate podatke pravilno, linearno podeli u klase. Ukoliko se pretpostavi da se jedan podatak može predstaviti kao vektor od *d* tačaka, cilj je formirati model koji će razdvojiti ove tačke sa $(d-1)$ - dimenzionalnom hiper ravni. Linearnu klasifikaciju moguće je izvesti primenom čvrste (krute) ili meke margine

Kruta margina

Slikom 4 prikazana je optimalna hiper ravan sa maksimalnom marginom koja razdvaja podatke u dve klase. Ukoliko važi linearna razdvojivost podataka, onda je u trening fazi neophodno odrediti optimalnu hiper ravan sa maksimalnom marginom (širinom razdvajanja) između klasa, koja iznosi $\rho = \frac{2}{\|w\|}$. Ovakav postupak smanjuje generalizacionu grešku klasifikatora i povećava linearnu razdvojivost. Kada je, jednačinom ravni, određena takva ravan dobija se model na osnovu koga se određuje kojoj klasi pripada nova instanca. Granica odlučivanja linearnog klasifikatora tada je prava $\mathbf{w} \cdot \mathbf{x} + b = 0$.



Slika 4. Hiper ravan sa maksimalnom marginom koja razdvaja podatke u dve klase

(Izvor: Nedeljković, 2015)

Ukoliko je trening skup oblika $\boldsymbol{x}(x_i, y_i)$, $x_i \in R^d$, $y_i \in \{-1, +1\}$ u d-dimenizionalnom prostoru u kome svaka instanca ima d atributa, tada je izrazom (10)

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b = \sum_{\boldsymbol{i}} \boldsymbol{w_i x_i} + b, \tag{10}$$

moguće opisati hiper ravan, pri čemu su: $\boldsymbol{w}$ normala hiper ravni a $\frac{b}{\|w\|}$ vertikalna udaljenost hiper ravni od koordinatnog početka.

Cilj SVM metode je odrediti hiper ravni najbližih instanci obe klase čime se izbor parametara **w** i b određuje na sledeći način:

$$x_i \boldsymbol{w} + b \geq +1, \ \forall \mathrm{y}_i = +1 \tag{11}$$

$$x_i \boldsymbol{w} + b \leq -1, \ \forall \mathrm{y}_i = -1 \tag{12}$$

Kombinacijom izraza (11) i (12) dobija se sledeće (13):

$$y_i(x_i \boldsymbol{w} + b) - 1 \geq 0, \forall i, 1 \leq i \leq n. \tag{13}$$

Izbor hiper ravni maksimalno udaljene od vektora oslonca određuje se maksimizacijom margine, odnosno potrebno je pronaći $min\|\boldsymbol{w}\|$ takav da važi izraz (13) (Nedeljković, 2015).

Meka margina

Ukoliko podaci nisu linearno razdvojivi koristi se nelinearni SVM baziran na ideji da se ulazni vektorski prostor preslika na višedimenzionalni vektorski prostor u kome je trening skup linearno razdvojiv. U izraze (11) i (12) uvodi se mala ne-negativna vrednost ζ_i takva da ublažava uslove linearnog SVM metoda, tako da važi sledeće:

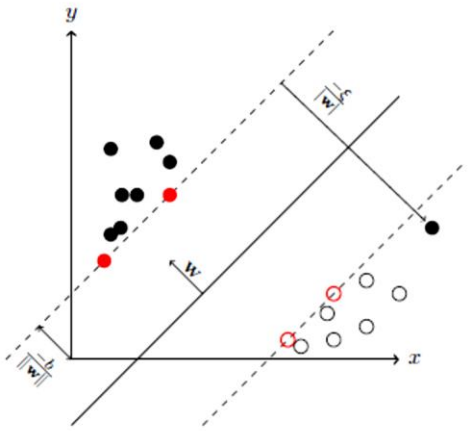
$$x_i \boldsymbol{w} + b \geq +1 - \zeta_i, \ \forall \mathrm{y}_i = +1 \tag{14}$$

$$x_i \boldsymbol{w} + b \leq -1 + \zeta_i, \ \forall \mathrm{y}_i = -1 \tag{15}$$

Kombinacija izraza (14) i (15) daje formulu za izračunavanje meke margine (16):

$$y_i(x_i \boldsymbol{w} + b) - 1 + \zeta_i \geq 0, \ \zeta_i \geq 0, \forall i, 1 \leq i \leq n \tag{16}$$

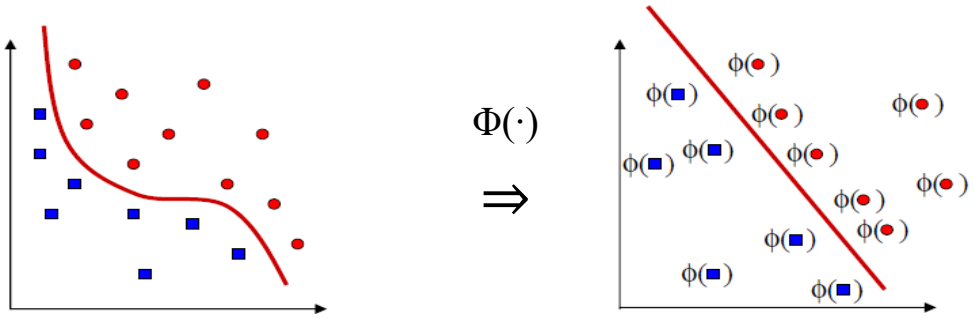
Slikom 5 ilustrovan je primer hiper ravni sa dve linearno nerazdvojive klase. Vidljiva je instanca sa pogrešne strane hiper ravni zbog koje prostor nije linearno razdvojiv. Mera rastojanja te instance od pripadajućeg vektora oslonca iznosi ζ_i (Nedeljković, 2015).



Slika 5. Hiper ravan sa mekom marginom i dve linearno nerazdvojive klase

(Izvor: Nedeljković, 2015)

Postoji način da se SVM koristi i za nelinearnu klasifikaciju primenom tzv. kernela na hiper ravni sa maksimalnom marginom. Na ovaj način algoritmu je obezbeđeno da traži hiper ravan sa maksimalnom marginom u transformisanom prostoru atributa. Primer preslikavanja linearno nerazdvojivog ulaznog prostora u novi prostor prikazan je Slikom 6.



Slika 6. Primer transformacije ulaznog prostora korišćenjem kernel funkcije Φ(·)

(Izvor: Nedeljković, 2015)

Transformacije mogu biti nelinearne, transformisani prostori višedimenzionalni, a originalni prostor nelinearan. U ovom slučaju generalizaciona greška raste iako postoji dovoljan broj instanci da algoritam precizno radi. Postoji niz kernel funkcija koje se mogu koristiti u primeni nelinearnog SVM metoda, kao što su: hiperbolički tangens, polinomijalne funkcije, Gausova Radial Basis funkcija, i druge.

5.4.2. k-najbližih suseda

Algoritmi zasnovani na najbližim susedima su algoritmi nadgledanog mašinskog učenja koji se mogu koristiti kako za primenu u klasifikaciji, tako i u regresiji. Algoritam čuva instance trening skupa i klasifikuje novu instancu na osnovu sličnosti. Ovo su „lenji“ algoritmi (lazy learners, engl.) jer ne uče iz trening skupa, nego čuvaju sve njegove instance, i na osnovu njih klasifikuju test skup. Klasifikacija se izvodi na osnovu većine (glasova) najbližih suseda svake tačke. Princip koji stoji iza metoda najbližih suseda je pronalaženje unapred definisanog broja obučavajućih instanci koje se nalaze na najmanjoj udaljenosti nove instance i predviđanje labele klase kojoj pripada. Rastojanje generalno može biti bilo

koja metrička mera, ali je Euklidovo rastojanje najčešći izbor. Ukoliko trening skup ima oblik prikazan izrazom (17)

$$L = \{(y_i), (x_i), i = 1, \cdots n_L\} \tag{17}$$

pri čemu su elementi klase $y_i \in \{1, \cdots, c\}$, a vektor $\mathbf{x}_i' = [x_{i1}, x_{i2}, \cdots x_{ip}]$ predstavlja vektor prediktora. Određivanje najbližeg suseda bazirano je na proizvoljnoj funkciji rastojanja $d(\cdot,\cdot)$ tako da se za svaku novu opservaciju (y, x) najbliži sused ($y_{(1)}$, $x_{(1)}$) u okviru obučavajućeg skupa određuje izrazom (18):

$$d(x, x_{(1)}) = \min_i \left(d(x, x_{(i)})\right) \tag{18}$$

a klasa najbližih suseda $\hat{y} = \mathrm{y}_{(1)}$ se izabere kao predikcija vrednosti y. Oznake $x_{(j)}$ i $y_{(j)}$ opisuju j-tog najbližeg suseda od x i pripadnost njegovoj klasi, respektivno (Hechenbichler, Schliep, 2004).

k-NN algoritam je jedna od najjednostavnijih tehnika koja se koristi u oblasti statističke diskriminacije. U pitanju je neparametarski metod kod koga se novi podaci pridružuju onoj klasi koja im je najbliža. k-NN čuva sve instance trening skupa u n-dimenzionalnom prostoru. Sličnost je određena distancom, a predviđanja nove k-te instance izvode se pretragom celog trening skupa sa k najbližih suseda i sumiranjem izlazne promenljive za ove slučajeve. Rang lista se izračunava prostom većinom glasova za svaki tačku. Podatak x_i iz vektora atributa poznatih instanci $\mathbf{x}_i = [x_1, \cdots x_k]$ nosi labelu y_i vektora atributa poznatih opservacija $\mathbf{y}_i = [y_1, \cdots y_k]$, a trening skup je označen kao $\{(x_i, y_i), i = 1, \cdots, k\}$ (Protić, Stanković, 2022).

Na žalost, ne postoji poseban metod za pronalaženje najbolje vrednosti k. Mala vrednost k može izazvati outlier-e ili uticati na porast šuma u modelu. Velika vrednost k je, u suštini, dobra ali su proračuni dugotrajni, a zahtevana memorija velika. Ukoliko se poredi sa drugim algoritmima ovo je, istovremeno, i njegova najveća mana. k-NN algoritam daje najbolje rezultate kada je broj atributa manji od 20 i na raspolaganju je velik obučavajući skup (© 1994-2016 The MathWorks, Inc.). U tom slučaju je obučavanje modela brzo i moguće je modelovanje kompleksnih prenosnih funkcija. Istovremeno, treba voditi računa da je odziv relativno spor i da irelevantni atributi unose dodatnu grešku. Programski kod k-NN algoritma prikazan je u Prilogu 2.

5.4.3. k-najbližih suseda sa težinskim koeficijentima

Ekstenzija težinskh koeficijenata opisuje ideju k-NN algoritma po kojoj, observacijama iz trening skupa koje su najbliže observaciji (y, x) treba dodeliti težinske koeficijente koji penalizuju najudaljenije susede tako da oni koji su bliži imaju veći uticaj na predikciju. Iz tog razloga je neophodno da prvi korak wk-NN algoritma bude transformacija rastojanja na kojima je baziran klasičan k-NN algoritam u mere sličnosti koje odgovaraju ovoj ideji. Bira se odgovarajuća kernel[11] funkcija $\Phi(\cdot)$, rastojanja d, tako da važe izrazi (19) – (21), odnosno:

[11] Kernel (jezgro, engl.) je funkcija koja predstavlja vezu između hardverskih komponenti i softvera (prim. aut.).

$$\Phi(d) \geq 0, \forall d \in R \tag{19}$$

$$\Phi(d) = max\big(K(d)\big) \Leftrightarrow d = 0 \tag{20}$$

$$\Phi(d) \text{ monotono opada za } d \rightarrow \pm\infty \tag{21}$$

Postoji niz poznatih kernel funkcija, kao što su npr. inverzna $\left(\frac{1}{|d|}\right)$, Gausova $\left(\frac{1}{\sqrt{2\pi}} e^{-\frac{d^2}{2}}\right)$, itd. Svaka od njih ima definisan ili prozor (ukoliko odstupanja od maksimalne vrednosti postanu 0) ili parametar disperzije (ukoliko su vrednosti veće od $d \in R$). Pre početka, obe vrednosti biraju se u zavisnosti od rastojanja od prvog suseda $x_{(k+1)}$, i više se ne menjaju u toku rada. Izbor se izvodi implicintno, standardizacijom svih drugih rastojanja u odnosu na rastojanje od (k+1)-vog suseda, na sledeći način:

$$D\big(x, x_{(i)}\big) = \frac{d(\mathrm{x}, \mathrm{x}_i)}{d(\mathrm{x}, \mathrm{x}_{k+1})}, \text{за } i = 1, \cdots k \tag{22}$$

Ove, standardizovane vrednosti uvek se nalaze u intervalu [0,1]. U određenim implementacijama na ovu vrednost se dodaje malo $\varepsilon > 0$ kako bi bila izbegnuta mogućnost da neki od najbližih suseda imaju isto rastojanje kao i (k+1)-vi sused, čime vrednosti kernela postaju 0, a vrednost rastojanja D=1, pa se gubi njihov uticaj na trening (Hechenbichler, Schliep, 2004). Softverski kod wk-NN algoritma dat je u Prilogu 2.

5.4.4. Stabla odlučivanja

Kao i svi drugi klasifikatori, i stabla odlučivanja klasifikuju podatke nepoznatog skupa u klase određena u procesu treninga modela, u kojem se korišćenjem definisanih kriterijuma biraju najbolji atribut svakog čvora stabla tokom njegovog formiranja (Sebastiani, Fabrizio, 2002). Stabla odlučivanja su jedan od najpopularnijih pristupa za klasifikaciju podataka u različitim oblastima kao što su na primer statistika, prepoznavanje obrazaca, mašinsko učenje, pretraživanje podataka i slično (Pejić, 2017). Stablo odlučivanja sadrži:

- koren – čvor bez ulaznih grana,
- unutrašnje čvorove koji imaju izlazne grane i
- listove – terminalni čvorovi (čvorovi odlučivanja).

DT predikcija zasnovana je na principu rekurzivne podele praćenjem odluka od korena ka listovima, do poslednjeg čvora. Unutrašnji čvor deli ulazne podatke prema diskretnim funkcijama ulaznih vrednosti atributa, na dva ili više podprostora i, najčešće, svaki podatak uzima u obzir jedan atribut, tako da je prostor podataka podeljen prema vrednosti posmatranog atributa. Istovremeno, svaki list stabla pridružen je jednoj klasi i predstavlja najbolju ciljnu vrednost. Dodeljivanje podataka klasi vrši se prolaskom kroz stablo, od korena ka listu. Metodom grananja ilustruje se mogući rezultat odlučivanja, pri čemu čvorovi odgovaraju atributima, grane pravilima odlučivanja a listovi izlazima (Protić, Stanković, 2020).

Atributi se u najvećem broju DT algoritama biraju tako da se koristi grananje u smeru odozgo na dole, na sledeći način:

- odabrati atribut za čvor stabla (kreirati granu za svaku vrednost atributa),
- podeliti instance na podskupove (jedan podskup za svaki atribut),
- rekurzivno ponavljati svaku granu i
- zaustaviti algoritam.

Složenost stabla kontroliše se pomoću zaustavnog kriterijuma, koji ima uticaj na njegovu tačnost. Na početku formiranja stabla vrši se izbor atributa čija vrednost najbolje deli raspoložive primene. Nakon toga, svaki čvor u stablu predstavljaće test nekog atributa a svaka grana koja polazi iz tog čvora odgovaraće jednoj od njegovih mogućih vrednosti. Na ovaj način, ukoliko svi atributi pripadaju jednoj klasi, stablu se dodaje novi list. Stablo donosi odluke ili na osnovu multivarijabilnih testova (više atributa i ulaza odjednom) ili na osnovu monovarijabilnih testova (test jednog atributa). Testovi mogu imati više ishoda a, u slučaju binarnog testa, radi se o binarnom stablu odlučivanja. Atributi mogu biti numerički ili kategorički a podatke je moguće deliti u više klasa. Ukoliko je ulaz u stablo binaran a atributi pripadaju dvema klasama, onda je u pitanju Bulovo stablo odluke.

Metode stabla odlučivanja robusne su na šum u ulaznim podacima, tj, greške u klasifikaciji su male za instance trening skupa i šum malo utiče na greške u vrednostima atributa koje opisuju ove instance. Metode stabla se mogu koristiti i ukoliko neki od instanci trening skupa ima nepoznate vrednosti.

Kod proračuna stabla koristi se Šenonova teorija informacija, i računa se entropija po formuli:

$$H(S) = H(p,n) = -\sum_{i=1}^{k} p_i \log_2 p_i = -p_+ \log_2 p_+ - p_- \log_2 p_- \tag{23}$$

pri čemu su S – skup svih primera u skupu podataka, k – broj primera u skupu podataka, p_i – verovatnoća događaja, $p+ = p/(p+n)$ – proporcija pozitivnih primera u S, $p_- = n/(p+n)$ – proporcija negativnih primera u S. Informaciona dobit $Gain(A,S)$ atributa A nad skupom instanci S predstavlja količinu informacija koja se dobija poznavanjem vrednosti atributa A i, istovremeno, predstavlja razliku entropije pre grananja i nakon grananja atributom A.

$$Gain(A,S) = H(S) - \sum_{i=1}^{v} \frac{|S_j|}{|S|} \cdot H(S_j) = H(S) - H(A,S) \tag{24}$$

pri tome S_j obeležava podskup od S za koji atribut A ima vrednost j. Vrednost $Gain(A,S)$ je informacija o vrednosti ciljne funkcije za poznatu vrednost atributa A. Najbolji atribut je onaj koji ima najveći informacioni dobitak. Programski kod ID3 algoritma (Mitchell, 1997) dat je u Prilogu 2.

5.4.5. Feedforward neuronska mreža

Feedforward neuronska mreža je jedna od najjednostavnijih i najbržih struktura za klasifikaciju koja je bazirana na propagaciji unazad, čime proizvodi rezultat baziran na verovatnoći predikcije i pragu klasifikacije. FNN sadrži slojeve povezanih neurona, tako da neuroni ulaznog sloja prihvataju ulazne podatke u FNN, a svi sledeći slojevi imaju vezu sa prethodnim slojem. Izlazni sloj proizvodi izlazne podatke tako da budu u željenoj relaciji sa ulazima u mrežu. Prenosna funkcija FNN sa jednim skrivenim slojem data je izrazom (25).

$$y_i(\mathbf{w},\mathbf{W}) = F_i\left(\sum_{j=1}^{q} W_{ij} f_j\left(\sum_{l=1}^{m} w_{ij} z_l + w_{j0}\right) + W_{f0}\right) \tag{25}$$

pri čemu su: z_l ulazi, y_i izlazi, $\mathbf{w}$ i $\mathbf{W}$ – težinske matrice ulaznog i izlaznog sloja, f_j i F_i, prenosne funkcije skrivenog i izlaznog sloja, m – broj elemenata ulaznog sloja, q – broj elemenata skrivenog sloja, a w_{j0} i W_{f0} – biasi.

U opštem slučaju, kada se primenjuje nelinearni optimizacioni algoritam, što važi i za FNN sa jednim skrivenim slojem, neophodno je izabrati odgovarajući kriterijum zaustavljanja. Mogući kriterijumi su:

- zaustavljanje nakon unapred izabranog broja iteracija algoritma,
- zaustavljanje nakon što kriterijumska funkcija (greške) dostigne maksimalnu vrednost ili
- zaustavljanje ukoliko relativna promena funkcije greške padne ispod zadate vrednosti (Bishop, 1995).

Za FNN opisanu izrazom (25) ulazni signal se propagira ka izlazu kroz slojeve neurona u mreži, do izlaza. Kod gradijentnih algoritama (Gradient Descent, GD, engl.) na osnovu prvog izvoda funkcije greške, težinski parametri se podešavaju iterativnim putem, unazad do ulaznog sloja. Podešavanje se izvodi po najvećem padu negativne vrednosti gradijenta, do pronalaženja minimuma funkcije greške (Silva et al., 2008). Negativna osobina ovog algoritma je da je, po pravilu, spor, može da se zaustavi pri dostizanju lokalnog umesto globalnog minimuma, ili da dođe do cik-cak kretanja po pravcu negativnog gradijenta. Na drugoj strani, ukoliko je funkcija greške računata po metodi minimizacije kvadratne funkcije greške aproksimirane u blizini optimalnog rešenja na osnovu Tejlorovog reda sume kvadratne funkcije greške, moguće je ubrzati proces minimizacije i optimizacija postaje bolja. Ova vrsta algoritma poznata je kao Gaus-Njutnov (GN) algoritam (Riecke et al., 2009; Shahin, Pitt, 2012).

5.4.6. Uporedne karakteristike algoritama za klasifikaciju

U nadgledanom mašinskom učenju, binarna klasifikacija je proces kojim se skup podataka deli na dve klase, merenjem atributa. Karakteristike od značaja su trening algoritam, zaustavni kriterijum, brzina predikcije, veličina memorije, interpretabilnost i fleksibilnost modela. MATLAB Classification Learner korišćen je kao osnova za opis ovih parametara.

Classification Learner sadrži podesive modele koji se treniraju optimizacijom tzv. hiperparametara. U zavisnosti od hiperparametara, modeli automatski optimizuju njihove vrednosti korišćenjem Bajesove optimizacije, koja minimizuje gubitke modela, u zavisnosti od izabrane opcije validacije[12]. U tekstu koji sledi treba uzeti u obzir i brzinu algoritma i potrošnju memorije, za koje važi sledeće

- Vrednosti brzine algoritma:
  - Brz algoritam – do 0.01s;
  - Algoritam srednje brzine – od 0.01s do 1s;
  - Spor algoritam – od 1s do 100s (i više, ukoliko se parametri zadaju manuelno);

[12] Classification learner. [Online] Dostupno na: https://uk.mathworks.com/help/stats/classification-learner-app.html Preuzeto 28.11.2022. godine

- Vrednosti potrošnje memorije:
  - Mala potrošnja – do 1MB;
  - Srednja potrošnja – od 1MB do 10MB;
  - Velika potrošnja – do 10MB do 100MB (i više, ukoliko se parametri zadaju manuelno);

1. DT je jedan od najefektivnijih klasifikatora, ali njegove karakteristike gotovo uopšte ne zavise od selekcije i skaliranja atributa. Hiperparametri koje je moguće optimizovati kod DT modela su maksimalni broj podela i kriterijum podele. Stabla odlučivanja su laka za interpretaciju, brza u predikciji i koriste malo memorije, ali imaju malu tačnost predikcije. Takođe, za stabla malih dimenzija postoji rizik od overfitting-a pa treba kontrolisati dubinu grananja maksimalnim brojem podela. Fleksibilnost modela raste sa povećanjem ovog broja. Tabelom 6 prikazane su karakteristike stabla odlučivanja.

Tabela 6. Classification Learner: DT model

Tip DT	Brzina predikcije	Potrošnja memorije	Interoperabilnost	Flaksibilnost
Simple	Brz	Mala	Laka	Niska (maksimalni broj podela = 4)
Medium	Brz	Mala	Laka	Srednja (maksimalni broj podela = 20)
Complex	Brz	Mala	Laka	Visoka (maksimalni broj podela = 100)

(Izvor: The MathWorks)

U eksperimentima prikazanim u disertraciji, softver pretražuje logaritmički skalirane cele brojeve iz opsega $[1, max(2, n-1)]$, pri čemu je n broj instanci i, takođe, pretražuje po Ginijevom diverziti indeksu, Twoingovom pravilu i maksimalnoj redukciji odstupanja. U ovom slučaju, maksimalni broj podela je 20.

2. k-NN identifikuje uzorak na osnovu k suseda. Kada je k malo, model ima tendenciju overfitinga. Za veliko k instance mogu da budu pogrešno klasifikovane. wk-NN algoritam je modifikovan k-NN algoritam, kojeg je predložio Dudani (1976). U odlučivanju, instanca prvog suseda dobija najveći težinski koeficijent (1), dok instanca k dobija najmanju vrednost težinskog koeficijenta (0). U eksperimentima su hiperparametri koji mogu biti podešavani sledeći: broj suseda, mera distance, težinski koeficijenti i binarna podela (tačno/netačno, 1/0, itd). Broj suseda se, po defaultu, ovde bira tako da bude $k \leq 10$. Softver pretražuje logaritamski skalirane cele brojeve iz opsega $[1, max(2, round\,(n/2))]$, pri čemu je n broj instanci. Mera rastojanja je Euklidova a težine se računaju kao inverzno kvadratno rastojanje. Tabelom 7 prikazane su karakteristike koje Classification Learner nudi za Nearest Neighbour klasifikatore.

Tabela 7. Classification Learner: Nearest Neighbours klasifikatori

Tip NN	Brzina predikcije	Potrošnja memorije	Interpretabilnost	Fleksibilnost
Fine k-NN	Srednja	Srednja	Teška	Detaljno razdvajanje; k = 1
Medium k-NN	Srednja	Srednja	Teška	Srednje razdvajanje; k = 10
Coarse k-NN	Srednja	Srednja	Teška	Coarse razdvajanje; k = 100
Cosine k-NN	Srednja	Srednja	Teška	Srednje razdvajanje; k = 10; tip cosine
Cubic k-NN	Spora	Srednja	Teška	Srednje razdvajanje; k = 10; tip cubic
Weighted k-NN	Srednja	Srednja	Teška	Srednje razdvajanje; k = 10; tip distance weight

(Izvor: The MathWorks)

U eksperimentima su izabrani Medium k-NN model i njegova ponderisana vrednost wk-NN model jer su relativno jednostavni i nema velikih zahteva prema memoriji.

Na drugoj strani, svaki korak klasifikacije zahteva procesorsko vreme koje je, kada je obučavajući skup velik, duže od ostalih klasifikacionih algoritama.

3. Gausov SVM nudi brzu klasifikaciju, a hiperparametri koje je moguće optimizovati su: kernel funkcija, kernel skala, multiklasni metod i standardizacija podataka. Tabelom 8 prikazane su opcije Classification Learner-a za SVM model.

Tabela 8. Classification Learner: SVM model

Tip SVM	Brzina predikcije	Potrošnja memorije	Interpretabilnost	Fleksibilnost
Linear	Binarni: Brza Multiklasni: Srednja	Srednja	Laka	Niska (linearna separacija)
Quadratic	Binarni: Brza Multiklasni: Spora	Binarni: Srednja Multiklasni: Velika	Teška	Srednja
Cubic	Binarni: Brza Multiklasni: Spora	Binarni: Srednja Multiklasni: Velika	Teška	Srednja
Fine Gaussian	Binarni: Brza Multiklasni: Spora	Binarni: Srednja Multiklasni: Velika	Teška	Visoka (opada skaliranjem kernela)
Medium Gaussian	Binarni: Brza Multiklasni: Spora	Binarni: Srednja Multiklasni: Velika	Teška	Visoka (opada skaliranjem kernela)
Coarse Gaussian	Binarni: Brza Multiklasni: Spora	Binarni: Srednja Multiklasni: Velika	Teška	Niska

(Izvor: The MathWorks)

U ovoj disertaciji prikazani su eksperimenti sa Gausovom kernel funkcijom a softver pretražuje pozitivne logaritmički skalirane vrednosti iz opsega $[0.001, 1000]$. Korišćen je One-vs-One metod a podaci su standardizovani pretraživanjem između vrednosti tačno (true, engl.) i netačno (false, engl.). Broj kernela je $\sqrt{P}/4$, pri čemu je P broj prediktora.

4. U eksperimentima je korišćena i FNN sa jednim skrivenim slojem, strukture 9 neurona u ulaznom sloju, 9 neurona u skrivenom sloju i jedan izlazni neuron (9-9-1), sa prenosnim funkcijama neurona oblika hiperboličkog tangensa. Osnovni problem kod FNN je da, za drastično različito skalirane atribute ulaznih podataka, model može da divergira, bude pretreniran (overestimated, engl.) ili nedovoljno treniran (underestimated, engl.), ili da ignoriše neke parametre. U eksperimentima prikazanim u ovom radu, rezultati su generisani na osnovu Levenberg-Marquardt (LM) algoritma obučavanja, algoritma podešavanja srednje kvadratne greške (Mean Squared Errors, MSE, engl.), sa sledećim default vrednostima: maksimalni broj iteracija $= 1000$, amplituda gradijenta $\leq 10^{-5}$ i broj validacija $= 6$.

Tabelom 9 prikazane su karaktersitike klasifikatora prikazanih u prethodnom delu teksta. Treba uzeti u obzir i da je, za evaluaciju modela, skup instanci podeljen tako da je 70% instanci korišćeno za trening modela, dok je po 15% instanci korišćeno za validaciju i testiranje.

Tabela 9. Karakteristike algoritama primenjenih u eksperimentima

	DT	SVM	k-NN	wk-NN	FNN
Trening algoritam	Algoritam stabla	Srednje Gausov	Euklidov	Euklidov	LM
Zaustavni kriterijum	Max broj podela = 20	sqrt(P)/4	k=10	k=10; Broj instanci trening skupa	min MSE
Brzina predikcije	Visoka	Visoka	Srednja	Srednja	Visoka
Zauzetost memorije	Mala	Srednja	Srednja	Srednja	Srednja/Velika
Interpretabilnost	Laka	Teška	Teška	Teška	Teška
Fleksibilnost	Srednja	Srednja	Teška	Teška	Srednja/Velika

(Izvor: The MathWorks)

6. SELEKCIJA I SKALIRANJE ATRIBUTA

U nadgledanom mašinskom učenju, binarna klasifikacija je proces koji deli skup podataka u dve različite klase, normalnog saobraćaja i anomalije, merenjenjem uticaja relevantnih atributa (Brownlee, 2020). Popularni binarni klasifikatori su, pored pet navedenih u prethodnim poglavljima, linearna regresija (LR), koja koristi logističku funkciju za modelovanje verovatnoće da će se neki događaj desiti (Rice, 2013), NB algoritam koji klasifikuje podatke na osnovu prethodnih rezultata, veoma je brz i ne zahteva ekstenzivan trening skup, i drugi (Nawir et al., 2018). Cilj binarne klasifikacije je formirati prenosnu funkciju koja će minimizovati pogrešnu klasifikaciju instance (Protic, Stankovic, 2022). Proces binarne klasifikacije čine izbor baze podataka, preprocesuiranje sa selekcijom i skaliranjem atributa, i sama binarna klasifikacija. Parametri binarnih klasifikatora koji su korišćeni u eksperimentima (podesivi hiperparametri objašnjeni u prethodnom delu teksta) su: algoritam obučavanja, zaustavni kriterijum, brzina predikcije, vreme procesuiranja podataka, zauzetost memorije, interpretabilnost i fleksibilnost modela. Izabrani parametri direktno utiču na tačnost rezultata i preciznost modela.

6.1. Kyoto 2006+ baza podataka

Kyoto 2006+ je javno dostupna baza podataka nastala snimanjem realnog mrežnog saobraćaja u periodu od 10 godina, na različitim računarskim mrežama unutar i izvan Kyoto univerziteta. Prva verzija baze snimljena je u periodu od 2006. do 2009. godine. Korišćeno je ~350 honeypot-a (zamke) (Solaris 8 for Intel, Windows XP (no patch, SP2, fully patched), Nepenthes, i drugi), uključujući i dva darknet senzora sa ~300 nekorišćenih IP adresa, e-mail servera, web crawler-a i drugih mehanizama za detekciju napada (Tabela 10) (Sing et al., 2015; Song et al., 2011).

Tabela 10. Pristupne tačke, senzori i sistemi unutar i van Kyoto Univerziteta

PRISTUPNE TAČKE, SENZORI I SISTEMI	
Honeypots	Solaris 8 za Intel
	Windows XP (no patch, SP2, fully patched)
	Nepenthes
	Drugo
Darknet sensors	Darknet sensors (za detekciju softvera, konfiguracije ili autorizacije koji koriste nestandardne komunikacione protokole ili portove)
Drugi sistemi	Mail server (za prikupljanje podataka iz e-mail-ova)
	Web crawler (razvijen u NTT Information Sharing Platform Laboratories)
	Windows XP (za malware aktivnosti)

(Izvor: Singh et al., 2015)

Ovaj deo baze sadrži oko milijardu instanci podataka (i normalnog i abnormalnog sadržaja). U toku perioda opservacije snimljeno je ~50 miliona sesija normalnog saobraćaja, ~43 miliona sesija poznatih napada i ~426 hiljada sesija nepoznatih napada. Nova verzija baze sadrži ~20 GB dodatnih podataka prikupljenih od 2009. do 2015. godine. Bazu čine dnevni zapisi koji se sastoje od 14 statističkih atributa preuzetih iz KDD Cup ’99 baze i 10 atributa protoka (IP adrese, portovi, trajanje konekcije) koje su autori dodali a namenjeni su za evaluaciju sistema za zaštitu podataka baziranih na anomalijama (Protić, 2018). Atribut

Label služi za detekciju anomalije. Ovaj atribut ima tri moguće vrednosti: 1 – obeležava normalan saobraćaj, -1 – obeležava poznati napad i -2 – obeležava nepoznat napad. Međutim, obzirom da je broj nepoznatih napada sporadičan u Kyoto 2006+ bazi podataka (~0.7%), u ovom istraživanju je i poznatom i nepoznatom napadu dodeljena ista labela koja ima vrednost -1.

Na osnovu 41 originalnog atributa iz KDD Cup ’99 baze, autori su izdvojili atribute sa honeypot-ova ignorišući one atribute koji sadrže redudantne podatke. Pored toga, izbačeni su podaci koje autori nisu smatrali značajnim za evaluaciju modela za zaštitu od napada na računarske mreže baziranih na anomalijama (karakteristike i opisi pojedinih napada). Takođe, izbačeni su svi oni atributi sadržaja koji su nepotrebni za istraživanje NIDS i mogu da utiču na povećanje vremena obrade podataka (npr. ekstrakcija irelevantnih atributa). Nakon svega, preostalo je 14 atributa. Na drugoj strani, autori su dodali 10 atributa koji su omogućili evaluaciju modela i njihovo testiranje. Tabela 11 prikazuje karakteristike Kyoto 2006+ baze podataka.

Tabela 11. Kyoto 2006+ baza podataka

R. br.	Atribut	Opis
1	Duration	Connection duration [s].
2	Service	Type of connection service.
3	Source bytes	# B sent by source IP address.
4	Destination bytes	# B sent by destination IP address.
5	Count	# of connections with same source/destination IP addresses to those of current connection in past 2s.
6	Same_srv_rate	% of connections to the same service in the feature Count
7	Serror_rate	% of connections that have 'SYN' errors in the feature Count.
8	Srv_error_rate	% of connections that have 'SYN' errors in Srv_count in past 2s.
9	Dst_host_count	Source/destination IP addresses are the same as the current connection (among past 100).
10	Dst_host_srv_count	The number of connections whose service type is also the same to that of the current connection.
11	Dst_host_same_src_port_rate	% of connections whose source port is the same to that of the current connection in Dst_host_count.
12	Dst_host_serror_rate	% of connections that have 'SYN' errors in Dst_host_count.
13	Dst_host_srv_serror_rate	% of connections that have 'SYN' errors in Dst_host_srv_count.
14	Flag	The state of the connection at the time of connection was written.
15	IDS_detection	Reflects if IDS triggered an alert for the connection.
16	Malware_detection	Indicates malware.
17	Ashula_detection	Shellcode and exploit codes were in the connection.
18	Label	Indicates of an attack.
19	Source_IP_Address	Source IP address used in the session.
20	Source_Port_Number	Session's source port number.
21	Destination_IP_Address	Also sanitized.
22	Destination_Port_Number	Session's destination port number.
23	Start_time	Start of the session.
24	Duration	Session duration.

(Izvor: Singh et al., 2015)

IDS Bro, signature & behavior analysis framework, korišćen je kao softver za konverziju saobraćaja baziranog na prenosu paketa u format oblika sesije. IDS Bro je sistem sa visokom preciznošću za analizu mrežnog saobraćaja tipa: hypertext transfer protocol (http), Domain Name System (DNS), Secure SHell (SSH) komunikacioni protokol i nepoznato ponašanje računarske mreže. Bro event engie prihvata IP pakete i vrši njihovu konverziju u događaje. Zahvaljujući tome, IDS Bro je prilagođen monitoringu mreže, analizi protokola i statusnim informacijama aplikacionog sloja, u realnom vremenu. Policy scrypt interpreter nakon toga generiše izlaz (Hardesty, 2017). Tabelom 12 dat je primer jedne instance formata sesije, koji sadrži elemente 24 atributa prikazanih u Tabeli 11.

Tabela 12. Primer sesije iz Kyoto 2006+ baze podataka

Redni broj	Vrsta atributa	Vrednost
1	Statistički	0.52
2	Kategorički	smtp
3	Statistički	3333
4	Statistički	244
5	Numerički	1.00
6	Numerički	1.00
7	Numerički	0.00
8	Numerički	0.00
9	Numerički	6.00
10	Numerički	99.00
11	Numerički	0.00
12	Numerički	0.00
13	Numerički	0.00
14	Statistički	SF
15	Za dalju analizu	0
16	Za dalju analizu	0
17	Za dalju analizu	0
18	Numerički	1
19	Kategorički	fdfd:c3e9:3c9c:264d:052b:4470:1f85:3407
20	Kategorički	41339
21	Kategorički	fdfd:c3e9:3c9c:9f52:7d2e: 27ee:079e:0f3f
22	Kategorički	25
23	Kategorički	00:00:36
24	Za dalju analizu	0.523710

Tabela 12 prikazuje četiri tipa atributa: statističke, kategoričke, numeričke i atribute za dodatne analize sistema za zaštitu od napada, koji su bazirani na detekciji anomalija.

6.2. Preprocesuiranje

Mnogi istraživački izazovi u analizi podataka, vizuelizaciji i razumevanju, u direktnoj su vezi sa dostupnošću i primenjivosti višedimenzionalnih podataka. Preprocesuiranje pre klasifikacije može u mnogome da pomogne da se ovi izazovi reše. Selekcija atributa često se koristi u praksi pre klasifikacije, s namerom da se smanji dimenzionalnost skupa podataka (Jie et al., 2018). Takođe, skaliranje podataka može biti primenjeno pre klasifikacije. Ukoliko skaliranje nije izvedeno, algoritam baziran na mašinskom učenju ima tendenciju da većim vrednostima dodeljuje veći značaj, a male vrednosti tretira kao manje značajne, bez obzira na njihovu relevantnost. Iz tog razloga, atributi treba da budu skalirani u iste (ili veoma slične) opsege, a preciznost klasifikatora treba da bude merena istom metrikom, za sve klasifikatore (Ulah, Barbar, 2019; Luque et al., 2019).

6.2.1. Izbor atributa

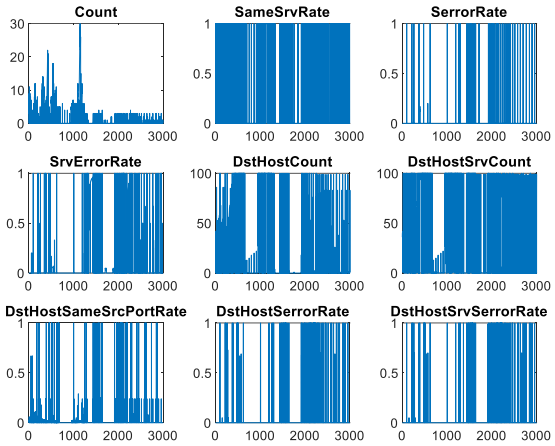
Izbor, odnosno selekcija atributa je prva faza preprocesurianja, pre klasifikacije. Cilj izbora atributa je pronaći relevantni podskup koji će redukovati kompleksnost modela, minimizovati generalizacionu grešku, obezbediti bolju predikciju i, sve u svemu, ubrzati evaluaciju modela. Najbitniji razlog selekcije atributa je izbeći da visok saobraćaj u mreži degradira karakteristike modela, zbog redudantnih podataka. Algoritmi za selekciju dizajnirani su tako da redukuju skup podataka i ubrzaju klasifikator bez da znatno utiču na njegove performanse.

Osnovni problem Kyoto 2006+ baze je njena veličina, odnosno broj instanci koji je dostupan za istraživanja. Po Rahman, Islam (2012 i 2015), literatura ukazuje na činjenicu da korisnici koji imaju dovoljno znanja u oblasti selekcije atributa mogu, u zavisnosti od kriterijuma odbacivanja, proceniti efikasnost atributa, bez korišćena automatizovanih mehanizama (algoritama) za redukciju veličine baze podataka. Ovakav pristup iskorišćen je i u ovom radu. U eksperimentima je ovaj problem složenosti rešen uklanjanjem svih redudantnih i irelevantnih atributa (kategorički atributi, statistički atributi koji se odnose na trajanje sesije i atributi namenjeni daljim analizama). Kao rezultat, devet numeričkih atributa, koji su prikazani u Tabeli 13, je odabrano za istraživanje.

Tabela 13. Numerički atributi relevantni za trening klasifikatora

Redni broj	Atribut
1	Count
2	Same_srv_rate
3	Serror_rate
4	Srv_error_rate
5	Dst_host_count
6	Dst_host_srv_count
7	Dst_host_same_src_port_rate
8	Dst_host_serror_rate
9	Dst_host_srv_serror_rate

Atribut *Label* iskorišćen je za detekciju anomalije. Kada je *Label* = 1, detektovano je normalno ponašanje mreže. Ukoliko je *Label* = −1, detektovana je anomalija. Problem koji je evidentan i nakon izbora atributa je da atributi koji nisu skalirani imaju različite vrednosti jedni u odnosu na druge. Slikom 7. dat je primer 3000 instanci izabranih atributa.



Slika 7. Instance izabranih, neskaliranih atributa

6.2.2. Skaliranje atributa

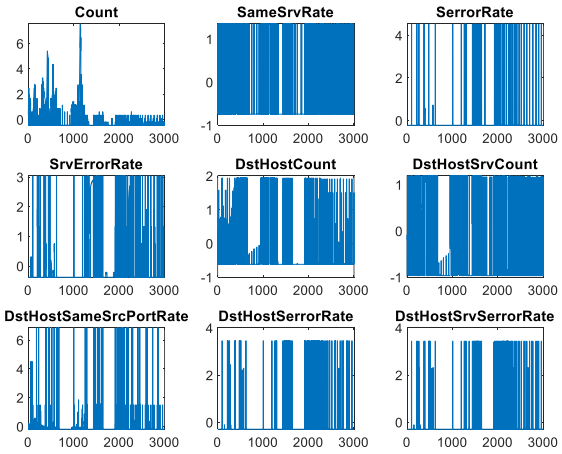
Skaliranje atributa je tehnika koja se primenjuje pre klasifikacije da bi nezavisni, relevantni atributi bili skalirani u isti opseg. U ovom delu rada prikazane su karakteristike Z-score standardizacije, dva tipa poznatih Min-Max normalizacija u opsege [0,1] i [−1,1], kao i nova metoda normalizacije bazirane na funkciji hiperbolički tangens ($tanh(x)$) i damping strategiji Levenberg-Marquardt algoritma.

6.2.2.1. Z-score standardizacija

Z-score standardizacija je strategija skaliranja podataka na takav način da atributi budu reskalirani da imaju nultu srednju vrednost i jediničnu standardnu devijaciju, i da za njih važi Gausova raspodela. Z-score standardizacija je jedna od najbitnijih metoda skaliranja za algoritme mašinskog učenja kao što su SVM, LR ili k-NN. Ukoliko vektor X sadrži n instanci $x(i)$, $i = 1,2,\ldots,n$, onda je Z-score standardizacija opisana formulom (26)

$$x(i)_{Z-score} = \frac{x(i)-mean(X)}{std(X)} \tag{26}$$

pri čemu je $x(i)_{\text{Z-score}}$ skalirana instanca, dok *mean(X)* i *std(X)* određuju srednju vrednost i standardnu devijaciju vektora X. Slikom 8 prikazane su vrednosti instanci sa Slike 7, skalirane Z-score standardizacijom.



Slika 8. Atributi skalirani Z-score standardizacijom

Treba znati da je, za povratak na osnovno skaliranje atributa, neophodno izvršiti postprocesuiranje, na sledeći način. Ukoliko se jednačina (26) posmatra kao funkcija oblika $f(x) = \frac{x-\mu}{\sigma}$, pri čemu je μ srednja vrednost a σ varijansa, onda se povratak na originalnu izvodi inverznom funkcijom $f^{-1}(x) = \sigma x + \mu$ tako da važi $x = f^{-1}(f(x))$, za prediktovane vrednosti.

6.2.2.2. Min-Max normalizacija atributa

Min-Max normalizacija je najčešće korišećna alternativa koja pomaže da se izbegnu efekti uticaja outlier-a obezbeđujući manju standardnu devijaciju izlaza. Normalizacija instance $x(i)$ datog atributa, nad intervalom $[a, b], \forall a, b \in R$ data je jednačinom (27):

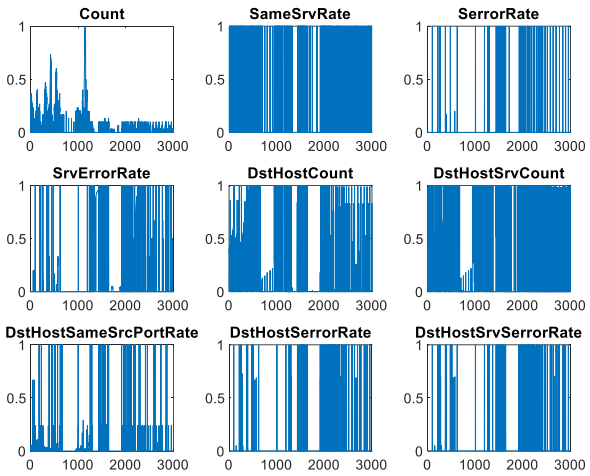
$$x(i)_{new} = a + \frac{(x(i)-x_{Min})(b-a)}{x_{Max}-x_{Min}} \tag{27}$$

pri čemu su $x(i)_{new}$, x_{Max} i x_{Min} normalizovana instanca, maksimalna i minimalna vrednost neskaliranog atributa, respektivno.

Normalizacija se najčešće izvodi na magnitudama vrednosti koje se koriste u algoritmima kao što su na primer k-NN i wk-NN, kada je neophodno izračunati rastojanja, ili u regresionim algoritmima, kada je potrebno odrediti koeficijente. Min-Max normalizacija u granice [0,1] ($MM_{[0,1]}$) data je izrazom (28)

$$x(i)_{normalized[0,1]} = \frac{x(i)-x_{min}}{x_{max}-x_{min}} \tag{28}$$

pri čemu je $x(i)_{normalized[0,1]}$ instanca skalirana u opseg [0,1]. Slikom 9. prikazani su atributi skalirani u granice [0,1].

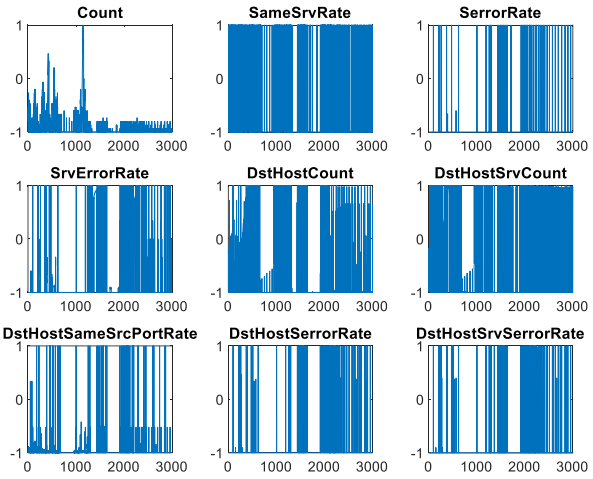


Slika 9. Atributi skalirani u granice [0,1]

Min-Max normalizacija u granice $[-1,1]$ ($MM_{[-1,1]}$) data je formulom (29):

$$x(i)_{normalized[-1,1]} = \frac{x(i) - \frac{x_{max}+x_{min}}{2}}{\frac{x_{max}-x_{min}}{2}} \tag{29}$$

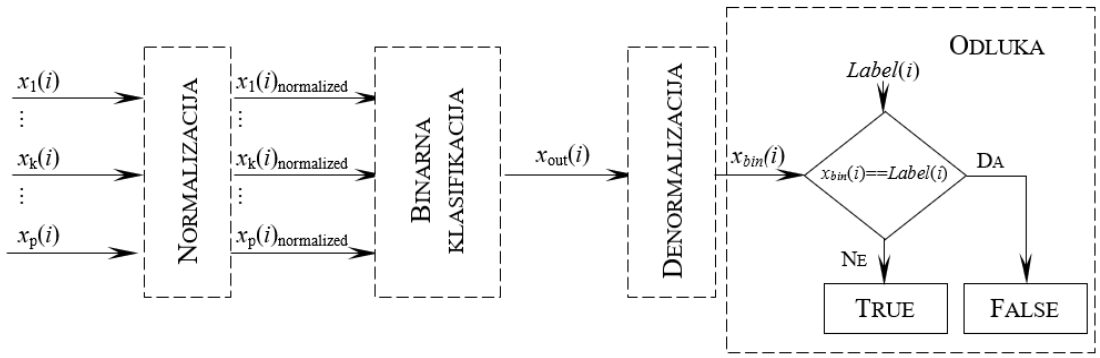
pri čemu je $x(i)_{normalized[-1,1]}$ instanca skalirana u opseg $[-1,1]$. Slikom 10. prikazani su atributi normalizovani u opseg$[-1,1]$.



Slika 10. Atributi normalizovani u granice [-1,1]

6.2.2.4. Denormalizacija i odlučivanje o izlaznoj labeli

U oba prethodna slučaja, Min-Max normalizacija se izvodi pre binarne klasifikacije. Međutim, šta se smatra pozitivnim a šta negativnim rezultatom mora biti striktno predefinisano radi boljeg tumačenja dobijenih rezultata, kasnije. Normalizacija, binarna klasifikacija i prezentacija dobijenih rezultata, prikazane su Slikom 11. U ovom slučaju atribut sadrži instance x_k(i), i=1,…, p, k=1,…, n, pri čemu su p i n broj atributa i ukupan broj instanci, respektivno.



Slika 11. Normalizacija, binarna klasifikacija, denormalizacija i donošenje odluka o anomaliji

Rezultat obe Min-Max normalizacije su instance $x_i(k)_{\text{normalized}}$, dok je rezultat binarne klasifikacije $x_{out}(i)$, koji nije obavezno ograničen u opseg atributa $Label$. Iz tog razloga, denormalizacijom se generiše nova binarna vrednost koja predstavlja pozitivan ili negativan rezultat klasifikacije. Obzirom da je (dobijeni) rezultat denormalizacije, $x_{bin}(i)$ isključivo pozitivan ili negativan moguće ga je porediti sa stvarnom vrednošću i-te instance ($Label(i)$).

1. Ukoliko je $MM_{[0,1]}$ normalizacija korišćena za skaliranje, proces denormalizacije ima dva koraka:
   - $ukoliko\ je\ x_{out}(i) \geq 0.5, x_{bin}(i) = 1,$
   - $u\ suprotnom\ x_{bin}(i) = -1$
2. U slučaju da je $MM_{[-1,1]}$ normalizacija bila primenjena za skaliranje, koraci denormalizacije su sledeći:
   - $ukoliko\ je\ x_{out}(i) \geq 0, x_{bin}(i) = 1,$
   - $u\ suprotnom\ x_{bin}(i) = -1$
3. Ukoliko su atributi skalirani Z-score standardizacijom, neophodno je primeniti sledeće korake da se odredi $x_{bin}(i)$:
   - $izvesti\ jednu\ od\ Min - Max\ metoda\ skaliranja$
   - $odrediti\ x_{bin}(i)\ u\ zavisnosti\ od\ izabrane\ normalizacije$
4. Na kraju, rezultat klasifikacije se smatra pozitivnim (TRUE) ukoliko je $x_{bin}(i) = Label(i)$, u suprotnom rezultat se smatra negativnim (FALSE).

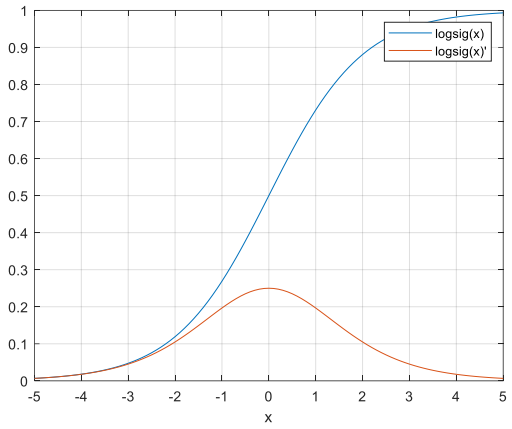
6.2.2.5. Tangens-hiperbolička normalizacija: primena Levenberg-Marquardt algoritma za ubrzavanje obučavanja klasifikatora

Min-Max normalizacija se, uobičajeno, koristi za trening neuronskih mreža jer atributi, skalirani na ovaj način, ubrzavaju trening i konvergencija je brža. Najčešće korišćene aktivacione funkcije neurona u mreži su funkcije u obliku slova *S*, logistička sigmoidalna funkcija ($logsig(x)$) i hiperbolički tangens. (Wang et al., 2019). Osnovna razlika između ove dve funkcije je njihov opseg. Vrednosti funkcije $tanh(x)$ ograničene su u opseg $[-1,1]$, dok su vrednosti $logsig(x)$ funkcije ograničene u opseg $[0,1]$ (Esfetanaj, 2018.). Tabela 14 prikazuje formule za obe funkcije i njihove prve izvode.

Tabela 14. Funkcije *tanh*(*x*), *logsig*(*x*) i njihovi prvi izvodi

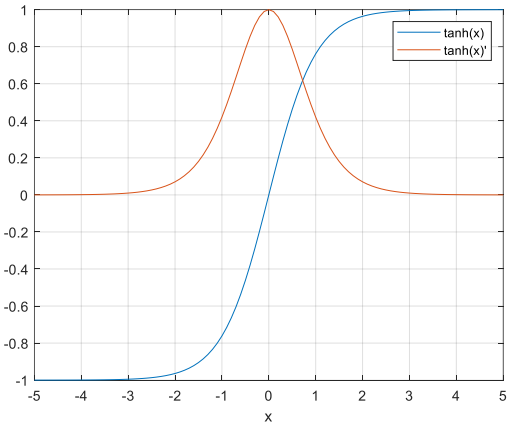
$f(x)$	$df(x)/dx$
$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$1 - tanh(x)^2$
$logsig(x) = \frac{1}{1 + e^{-x}}$	$logsig(x)(1 - logsig(x))$

U opštem slučaju, centrirana ne-nulta funkcija ograničava pomeranje parametara po prostoru u određenom pravcu. Le Cun et al. (1992) objašnjavaju bržu konvergenciju kada je srednja vrednost svake ulazne promenljive nad trening skupom blizu nulte vrednosti, i daju primer ekstremne situacije ukoliko su svi ulazi pozitivni. Kada se ovo desi, sve nove vrednosti težina (updates, engl.) neurona ulaznog sloja mogu samo da rastu ili opadaju, istovremeno i u istom pravcu. Za aktivacione funkcije kao što je $logsig(x)$, čiji je izlazni opseg [0,1] a centrirana je oko 0.5, ukoliko težinski parametar mora da promeni pravac, to može da radi u cik-cak maniru, što je neefikasno i veoma sporo. Slikom 12 prikazana je funkcija $logsig(x)$ i njen prvi izvod.



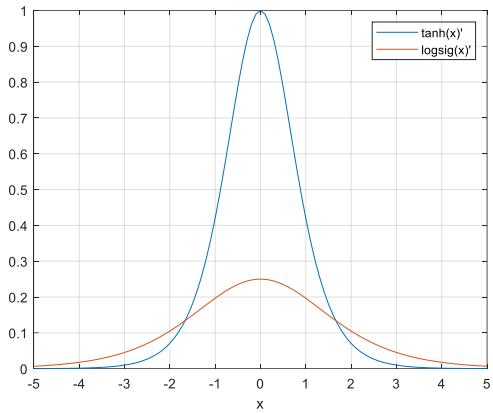
Slika 12. Funkcija logsig i njen prvi izvod

Na drugoj strani, funkcija $tanh(x)$ je centrirana oko nule sa izlaznim vrednostima koje se nalaze u opsegu $[-1,1]$. Prvi izvod ove funkcije ima znatno veću vrednost za $x = 0$ od funkcije $logsig(x)$. Slikom 13. prikazane su funkcija $tanh(x)$ i njen prvi izvod.



Slika 13. Funkcija tanh i njen prvi izvod

Uporedni prikaz prvih izvoda funkcija $logsig(x)$ i $tanh(x)$ prikazan je Slikom 14.



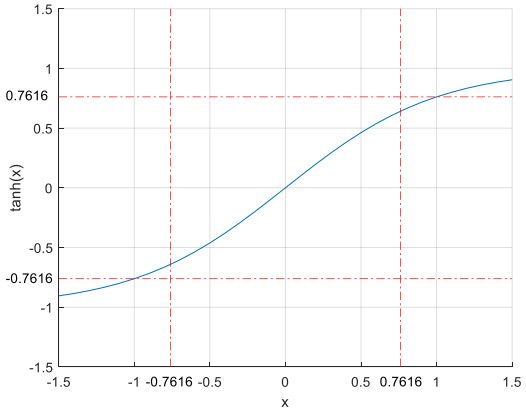
Slika 14. Uporedni prikaz funkcija *logsig*(*x*) i *tanh*(*x*)

U ovom radu prikazan je nov pristup normalizaciji tangens hiperboličkom funkcijom (TH pristup), obzirom na njeno svojstvo da je pad gradijenta prvog izvoda ciljane funkcije veoma brz. Funkcija gradijenta ima velik pad u okolini tačke $x = 0$, kada je rešenje funkcije cilja blizu optimalnog rešenja dobijenog iterativnim putem, tj. $tanh(x)' = 1 - tanh(x)^2$, $x = 0$, odnosno $tanh(x)'_{x \to \pm 0} \approx 1$. Ovaj pristup obezbediće ubzavanje svih algoritama koji su bazirani na gradijentnom metodu.

Pored toga, u blizini optimalnog rešenja, primenjen je algoritam baziran na Gaus-Njutnovom algoritmu, odnosno na aproksimaciji funkcije cilja drugim izvodom funkcije dobijenim Tejlorovim redom. Naime, u blizini optimalnog rešenja gradijentna metoda usporava i može doći do cik-cak kretanja. Gaus-Njutnov algoritam (drugog reda) traži prevojnu tačku i na taj način preskače lokalne minimume, ubrzava pretraživanje i, u opštem slučaju, ubrzava proces nalaženja globalnog minimuma dobijenog iterativnim putem.

Levenberg-Marquardt algoritam kombinuje ove dve aproksimacije tzv. damping strategijom (Levenberg, 1944; Marquardt, 1965). Kada je rešenje daleko od optimalnog, algoritam pretrage je gradijentan jer je precizniji. U blizini optimalnog rešenja, algoritam postaje Gaus-Njutnov jer je brži. Prebacivanje sa jednog na drugi vid pretrage izvodi se damping faktorom koji može biti predefinisan (konstantna vrednost koju zadaje korisnik) ili se menjati sa svakom iteracijom.

Obzirom da je deo funkcije $tanh(\pm 1) \approx \pm 0.7616$, funkcija se može se smatrati kvazi-linearnom u ovom opsegu (Slika 15).



Slika 15. Kvazi-linearan deo funkcije tanh(x)

Na osnovu svega navedenog, TH normalizacija se izvodi tako da se, na instance skalirane Min-Max normalizacijom, primenjuje izraz (30).

$$x(i)_{TH} = tanh\left(\frac{x(i)-\frac{x_{Max}+x_{Min}}{2}}{\frac{x_{Max}-x_{Min}}{2}}\right) \tag{30}$$

pri čemu je $x(i)_{TH}$ instanca skalirana TH normalizacijom. Ovaj korak u preprocesuiranju se može smatrati finim podešavanjem pre treninga modela. Takođe, bitno je razumeti da, zbog ograničenja instanci u opseg $[tanh(-1), tanh(1)] \approx [-0.7616, 0.7616]$, proizvod ovih vrednosti i maksimalnih vrednosti težinskih koeficijenata kod primenjenih modela, ne može imati vrednosti ± 1, što će obezbediti dodatno ubrzavanje LM algoritma. Takođe, biće izbegnuta saturacija FNN na početku treninga (Protic, Stankovic, 2021). Detaljan prikaz GD, GN i LM algoritma, sa odgovarajućim aproksimacijama, dat je u Prilogu 3. U delu rada koji sledi opisani su osnovni elementi metoda i optimizacionog kriterijuma, kao i način podešavanja damping faktora.

Levenberg-Marquardt algoritam

Kao što je prikazano u prethodnom delu teksta, LM algoritam je nelinearan interativni metod koji kombinuje GD algoritam minimizacije funkcije cilja iterativnom procedurom baziranom na veličini koraka pretraživanja i pravcu pretraživanja određenom negativnim gradijetnom, i GN algoritam koji pojednostavljuje proračun inverzne Hesijanove matrice korišćenjem vektorskog proizvoda Jakobijanovih matrica, kada je funkcija greške približno kvadratna u blizini optimalnog rešenja.

Osnovna ideja LM algoritma je izvesti afinu transformaciju funkcije $f: R \to R$ u blizini trenutne tačke *p*, koja je bazirana aproksimaciji Tejlorove funkcije (Wang et al., 2019a; Xue et al., 2020), tako da važi (31):

$$\hat{f}(x;p) = f(p) + Df(p)(x-p) \tag{31}$$

pri čemu je $Df(p)(x-p)$ Jakobianova matrica (**J**) parcijalnih izvoda funkcije *f*. LM algoritam aproksimira problem na taj način da, za trenutnu vrednost rešenja, važi da je $\hat{f}(x;p) = f(x)$ i $x \approx p$.

Neka su $x^{(1)}, x^{(2)}, \ldots x^{(l)}$ vrednosti potencijalnih rešenja. U iteraciji *k*, vrednost $x^{(k+1)}$ bira se tako da minimizuje oba izraza iz formule (32):

$$\left\|\hat{f}\left(x;x^{(k)}\right)\right\|^2 + \lambda^{(k)}\left\|x - x^{(k)}\right\|^2, \ \lambda^{(k)} > 0 \tag{32}$$

Parametar $\lambda^{(k)}$ je damping faktor (faktor odbacivanja) koji varira sa veličinom koraka i čiju vrednost je moguće prilagoditi.

Neka je $x^{(k+1)}$ rešenje problema. U tom slučaju $x^{(k)}$ predstavlja stacionarnu tačku ukoliko je $x^{(k+1)} \approx x^{(k)}$, i može biti određen na sledeći način:

$$x^{(k+1)} = x^{(k)} - \left(Df\left(x^{(k)}\right)^{\mathrm{T}} Df\left(x^{(k)}\right) + \lambda^{(k)}\mathrm{I}\right)^{-1} Df\left(x^{(k)}\right)^{\mathrm{T}} f\left(x^{(k)}\right), \ \lambda^{(k)} > 0 \tag{33}$$

pri čemu optimalni uslovi važe ako i samo ako je $Df\left(x^{(k)}\right)^{\mathrm{T}} f\left(x^{(k)}\right) = 0$.

U iteraciji k, parametar $\lambda^{(k)}$ može biti podešen na sledeći način:

→ Ukoliko $\lambda^{(k)}$ ima veliku vrednost, onda je $x^{(k+1)}$ veoma blizu $x^{(k)}$, što usporava proces, pa $\lambda^{(k)}$ treba uvećati (uobičajeno je da $\lambda^{(k+1)} = 2\lambda^{(k)}$).

→ Ukoliko $\lambda^{(k)}$ ima malu vrednost, $x^{(k+1)}$ je daleko od $x^{(k)}$, i aproksimacija je loša što znači da $\lambda^{(k)}$ treba da bude umanjena (uobičajeno je da $\lambda^{(k+1)} = 0.5\lambda^{(k)}$).

Vrednost $x^{(k+1)}$ može da bude određena po LM formuli, na sledeći način:

$$x^{(k+1)} = x^{(k)} - \left(\mathbf{H} + \lambda^{(k)}\mathbf{I}\right)^{-1}\mathbf{J}^{\mathrm{T}} f\left(x^{(k)}\right) = x^{(k)} - \left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda^{(k)}\mathbf{I}\right)^{-1}\mathbf{J}^{\mathrm{T}} f\left(x^{(k)}\right) \quad (34)$$

pri čemu su $\mathbf{H} = \mathbf{J}^{\mathrm{T}}\mathbf{J}$ Hesijanova matrica, a $\mathbf{I}$ predstavlja identičnu matricu.

→ U slučajevima kada $\lambda^{(k)} \to \infty$, važi da je $\mathbf{H} + \lambda^{(k)}\mathbf{I} \approx \mathbf{I}$ i LM algoritam se ponaša kao GD algoritam.

→ U suprotnom, kada $\lambda^{(k)} \to 0$, LM algoritam se ponaša kao GN algoritam jer je $x^{(k)}$ blizu optimalnog rešenja.

Alternacija između GD i GN algoritama poznata je kao damping strategija. Na ovaj način izbegnut je uticaj veoma velikih ili veoma malih vrednosti gradijenata i skraćeno je vreme procesuiranja.

7. KONFLIKTNE ODLUKE O ANOMALIJAMA: BINARNA KLASIFIKACIJA I XOR DETEKTOR

Preprocesuiranje, u opštem slučaju, smanjuje broj atributa za trening klasifikatora i skalira relevantne atribute, čime evaluacija klasifikatora postaje brža, a njegova tačnost može da poraste ili da neznatno degradira. U prethodnom delu rada prikazana je metodologija selekcije 9 numeričkih od ukupno 24 atributa iz Kyoto 2006+ baze podataka. Prikazane su različite metode skaliranja, s posebnim osvrtom na novu TH normalizaciju. Izabrano je pet binarnih klasifikatora baziranih na nadgledanom mašinskom učenju, čije su karakteristike detaljno prikazane u prethodnim poglavljima.

Cilj XOR modela nije smanjiti procesno vreme ili direktno povećati tačnost sistema za detekciju anomalije, nego je cilj generisati dodatne alarme koji bi upozorili donosioca odluka o potencijalno malicioznom napadu, koji nije bilo moguće utvrditi jednim binarnim klasifikatorom. Osnovu XOR detektora čini primena logičke operacije ekskluzivno ili koja je prikazana Tabelom 15. Karakteristika ove logičke operacije je da, za isti tip ulaznih vrednosti, generiše izlaznu vrednost '0', dok je ova veličina '1' ukoliko se ulazne vrednosti razlikuju.

Tabela 15. XOR operacija

XOR	0	1
0	0	1
1	1	0

Za formiranje XOR detektora neophodno je da postoje najmanje dva klasifikatora visoke tačnosti koji simultano rade u paraleli. Trening, validacija i testiranje klasifikatora izvode se tako da se poznate, skalirane instance dovode na ulaz modela, koji se obučava do ispunjenja predefinisanih zaustavnih kriterijuma. Klasifikatori se zatim postavljaju na isto mesto u računarskoj mreži, i puštaju u rad u realnom vremenu. Detekcija saobraćaja izvodi se na triger, po prijemu podataka.

Ideja u radu je da se, kao binarni klasifikatori, koriste jedan klasifikator visoke tačnosti, sa brzim vremenom procesuiranja i jedan izuzetno precizan klasifikator, čije je vreme procesuiranja dugo. Pored toga, oba klasifikatora treba da imaju težinske koeficijente i koriste kvadratni tip greške u odlučivanju. Iz tog razloga, izbor je pao na izuzetno brzu FNN i veoma precizan wk-NN model.

1. wk-NN model optimizuje uticaj najbližih suseda tako što instance u obučavajućem skupu, koje su najbliže novoj instanci, dobijaju veću vrednost težinskih parametara kod odlučivanja, rastojanja se transformišu u mere sličnosti, odnosno težinske koeficijente w_i, na sledeći način:

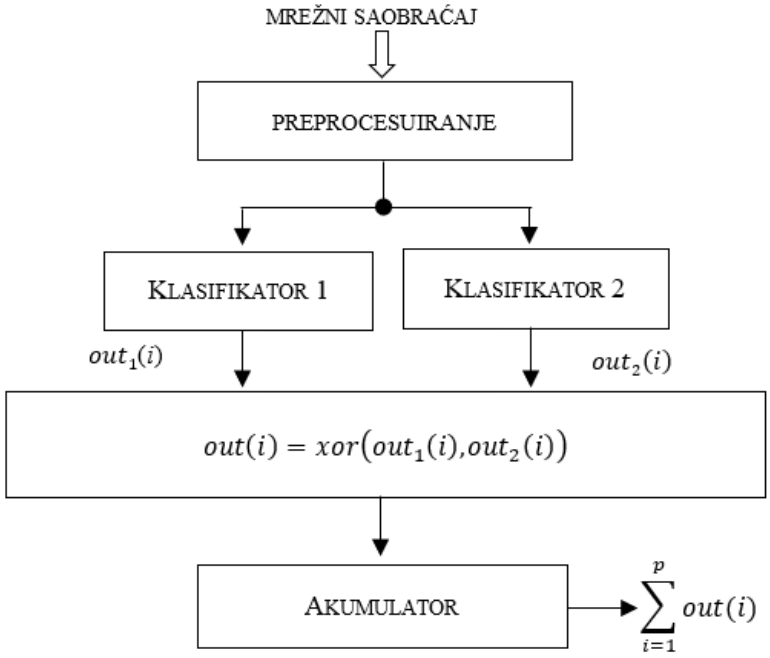
$$d_w(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{s} w_i^2 (x_{is} - y_{is})^2} \tag{35}$$

pri čemu su $d_w(\boldsymbol{x}, \boldsymbol{y})$, x_{is} i y_{is}, i w_i distanca, susedi i težinski koeficijenti, respektivno, a vektor težina se računa po formuli:

$$w = \frac{1}{d_w(\boldsymbol{x},\boldsymbol{y})^2} \tag{36}$$

2. Struktura i prenosna karakteristika FNN opisane su formulom (25), pri čemu treba naglasiti da su težine definisane kao vrednost hiperboličkog tangensa ulaza, odnosno izlaza iz prethodnog sloja u mreži, a da je struktura FNN 9 ulaza, 9 neurona skrivenog sloja i 1 izlazni neuron.

Međutim, bez obzira na njihovu tačnost, kada klasifikatori rade simultano i u paraleli povremeno se dešava da jedan detektuje anomaliju dok drugi procenjuje da je ponašanje računarske mreže normalno, i obrnuto. XOR detektor, prikazan u ovoj disertaciji, dizajniran je tako da poredi odluke dva binarna klasifikatora tako što primenjuje bitsku operaciju XOR na procenjene labele. U konfliktnim odlukama nebitno je koji od klasifikatora detektuje anomaliju a koji klasifikuje saobraćaj u računarskoj mreži kao normalan, bitno je da su odluke oprečne. Konceptualni dizajn XOR detektora prikazan je Slikom 16.



Slika 16. Dizajn XOR detektotra

Prediktovane vrednosti oba klasifikatora su odluke o labeli, tj. detekcija ili anomalije ili normalnog mrežnog saobraćaja. Ukupan broj konfliktinh odluka računa se po formuli (37)

$$sum_{out} = \sum_{i=1}^{data\_size} xor(out_1(i), out_2(i)) \quad (37)$$

pri čemu su $out_1(i)$ i $out_2(i)$ rezultati klasifikacije za instance i=1,..., *data_size*, a $out(i)=xor(out_1(i),out_2(i))$ je konfliktna odluka različitih detekcija, na ukupnom broju (*data_size*) instanci saobraćaja. Detekcija konflikata izvodi se na sledeći način:

- generisanje trigera o prijemu instance,
- selekcija 9 numeričkih atributa,
- skaliranje,
- detekcija anomalije/normalnog saobraćaja,
- XOR operacija nad detektovanim vrednostima,
- procena konfliktne odluke,
- generisanje upozorenja o konfliktu.

Prednosti akumulator-registra su, u ovom slučaju, kraće instrukcije i manje memorije neophodne za izvođenje operacija. Svaki sledeći izlaz $out(i)$ dodaje se na prethodne vrednosti u akumulatoru. Druga prednost je što svaki ciklus za izvršavanje instrukcija traje kraće jer akumulator smanjuje fetching instrukcija iz memorije. Broj instanci nad kojima se izvodi odlučivanje o konfliktu odrediće ukupnu dužinu akumulatora (p). Važno je napomenuti da je, u opštem slučaju $data\_size \neq p$.

Obzirom da broj odluka varira, i zavisi od zahteva korisnika za bezbednost podataka, osetljivost XOR detektora na donošenje odluka takođe zavisi od korisnika, a ne isključivo od tehničkog izvođenja komponenti klasifikatora, akumulatora ili alarma. Pitanje je kolika je granularnost neophodna kompaniji u mehanizmima zaštite podataka. Neophodno je razumeti da, iako XOR detector pomaže u donošenju odluka, na ovaj način nije moguće predvideti verovatnoću nastanka određenog događaja. Njegova osnovna namena je obezbediti dodatna upozorenja onima koji donose odluke.

8. EXPERIMENTALNI REZULTATI

Eksperimenti, prikazani u ovom poglavlju, potvrđuju polazne pretpostavke i daju uvid u rezultate istraživanja koja su prikazana u prethodnim poglavlja. Korišćeni su MATLAB Classification Learner i Neural Network Toolbox, dnevni zapisi iz Kyoto 2006+ baze podataka, Intel(R) Core(TM) i7-2620M CPU 2.70GHz processor sa 16GB RAM instalirane memorije i Winndows operativni sistem. Eksperimenti su izvedeni u periodu od 2016. do 2021. godine, a rezultati su sukcesivno objavljivani u naučnim radovima, počevši od 2018. godine, zaključno sa krajem 2022. godine.

8.1. Procena efikasnosti i efektivnosti klasifikacije

Ocena kvaliteta i efektivnost predikcije binarnih klasifikatora ne izvodi se analitički jer je, u tom slučaju, neophodno poznavati formalne specifikacije problema koje sistem treba da reši. Iz tog razloga je, radi poređenja modela, potrebno obučavati i testirati klasifikatore na istom skupu podataka i sa istim atributima. Mere kvaliteta klasifikatora i efektivnost predikcije za pozitivne vrednosti (1) i negativne vrednosti (0), mogu se odrediti na osnovu podataka iz konfuzione matrice. Osnovna prednost konfuzione matrice je što je moguće lako odrediti da li model greši u predikciji, i koliko. Ukoliko P (Positive, engl.) predstavljaju tačne, a N (Negative, engl.) predstavljaju pogrešno procenjene vrednosti, onda je klasifikacione rezultate moguće porediti sa odgovarajućim labelama i karakteristike modela je moguće meriti u odnosu na tačno i lažno pozitivne i negativne rezuultate, na sledeći način:

- *TP* (*True Positive*, engl.) – korektno klasifikovano normalno ponašanje mreže (tačno pozitivni);
- *TN* (*True Negative*, engl.) – određuje korektno identifikovane negativne rezultate (tačno negativni);
- *FP* (*False Positive*, *False Alarm*, *Type 1 error*, engl.) – označava normalno ponašanje mreže pogrešno klasifikovano kao anomalija;
- *FN* (*False Negative*, *Type 2 error*, engl.) – označava abnormalno ponašanje mreže pogrešno dodeljeno klasi normalnog saobraćaja;

Jednačine (38) – (46) opisuju veličine koje se mogu izračunati na osnovu konfuzione matrice (Ambedkar, Babu, 2015).

- *True Negative Rate* (*TNR*, *Specifity*, *Selectivity*) – deo pozitivnih uzoraka koje je model tačno klasifikovao:

$$TNR = Selectivity = Specificity = \frac{TN}{N} = \frac{TP}{FP+TN} = 1 - FPR \tag{38}$$

- *True Positive Rate* (*Sensitivity*, *TPR*, *Recall*, engl.) – fokusira se na *FN* rezultate i određuje koliko je rezultata od svih prediktovanih pozitivnih korektno prediktovano:

$$TPR = Recall = Sensitivity = \frac{TP}{P} = \frac{TP}{TP+FN} = 1 - FPR \tag{39}$$

- *False Negative Rate* (*FNR*, *Miss Rate*, engl.) – deo pozitivnih uzoraka prediktovanih u negativnu klasu:

$$FNR = \frac{FN}{P} = \frac{FN}{TP+FN} = 1 - TPR \tag{40}$$

- *False Positive Rate* (*FPR*, *Fall-out*, engl.) – deo negativnih uzoraka prediktovanih u klasu pozitivnih:

  $$FPR = \frac{FP}{N} = \frac{FP}{TN+FP} = 1 - Specifity \quad (41)$$

- *Precision* (*Positive Predictive Value*, engl.) – odnosi se na FP i određuje koliko je, od ukupno prediktovanih pozitivnih instanci, ustvari pozitivno:

  $$Precision = PPV = \frac{TP}{TP+FP} \quad (42)$$

- *Negative Prediction Value* (*NPV*)

  $$NPV = \frac{TN}{TN+FN} \quad (43)$$

Visoka vrednost *Recall* a niska vrednost *Precision* znači da je veći deo primera dobro propoznat ali postoji veliki broj lažno pozitivnih primera (niska vrednost *FN*), dok niska vrednost *Recall* a visoka vrednost *Precision* pokazuje da je propušten veliki broj pozitivnih primera, a oni koji su prediktovani kao pozitivni zaista i jesu pozitivni (niska vrednost *FP*). Kako bi bio smanjen uticaj ekstremnih vrednosti uvodi se termin F_β, koji predstavlja otežanu harmonijsku srednju vrednost, kao u formuli (44):

- $$F_\beta = \frac{1}{\beta\frac{1}{Precision}+(1-\beta)\frac{1}{Recall}} \quad (44)$$

  Kada je $\beta = 1/2$, F_β is je poznat i kao *F1-score* (F_1); u ovom slučaju *Precision* i *Recall* imaju isti značaj i tada je:

- $$F_1 = \frac{2*Recall*Precision}{Recall+Precision} \quad (45)$$

  koji predstavlja harmonijsku srednju vrednost umesto aritmetičke i na taj način je uvek bliža manjoj vrednosti ili *Recall* ili *Precision*. Međutim, ova veličina je teška za interpretaciju, pa nije poznato šta klasifikator minimizuje *Precision* ili *Recall*. Tako da je potpuni uvid u rezultat moguće steći ukoliko se vrednost F_1 kombinuje sa nekom od drugih metrika, kao što je na primer tačnost (Protić et al., 2022).

- Accuracy (Classification rate, engl.) je izraz koji opisuje tačnost modela:

  $$Accuracy = \mathrm{ACC} = \frac{TP+TN}{P+N} = \frac{TP+TN}{TP+TN+FP+FN} \quad (46)$$

  Treba imati u vidu da se tačnost odnosi na greške za obe klase tako da, u zavisnosti od problema istraživanja, tačnost ne treba da bude jedina mera kvaliteta modela. Treba, na drugoj strani, da bude što veća, jer predstavlja odnos korektno klasifikovanih instanci sa njihovim ukupnim brojem.

Za predikciju nad skupom modataka, MATLAB ceo skup deli na tri dela: trening skup se koristi za obučavanje modela, validacioni skup se koristi da se odredi kada treba prekinuti obučavanje, a test skup se koristi da se izračunaju greške modela na nepoznatim podacima. Pored svega navedenog, kao veličina za uporedni prikaz biće korišćeno i vreme procesuiranja koje predstavlja ukupno vreme treninga i testiranja modela.

8.2. Selekcija atributa

Za potrebe eksperimenata izvedene su višestruke redukcije Kyoto 2006+ baze podataka, s ciljem da se odredi na koji način i u kojoj meri selekcija relevantnih atributa utiče na povećanje tačnosti i smanjenje vremena procesuiranja binarnih klasifikatora. Redukcije su izvedene selekcijom atributa u nekoliko faza:

- odbačeni su svi kategorički atributi i
- odbačeni su svi atributi namenjeni daljim analizama, osim atributa *Label*, koji služi za detekciju tipa saobraćaja.

Nakon selekcije izvedeno je skaliranje relevantnih atributa. U prvoj fazi izabrano je 17 atributa, odnosno obačeni su kategorički atributi koji opisuju vreme, datum, vrstu greške, tip protokola i IP source i destination adrese (2, 19-23). U drugoj fazi odbačeni su svi nenumerički atributi, tako da je devet numeričkih atributa preostalo za validaciju modela (Tabela 16).

Tabela 16. Numerički atributi iz Kyoto 2006+ baze podataka

Redni broj	Atribut	Vrednost/Opseg
5.	Count	{0, 1, 2, 3, 4, 5, 6, 7}
6.	Same_srv_rate	{0, 1}
7.	Serror_rate	{0, 0.5, 1}
8.	Srv_serror_rate	[0, 1]
9.	Dst_host_count	[0, 100]
10.	Dst_host_srv_count	[0, 100]
11.	Dst_host_same_src_port_rate	{-1, 0, 1}
12.	Dst_host_serror_rate	{0, 0.5, 1}
13.	Dst_host_srv_serror_rate	{0, 1}
18.	Label	{0, 1}

Na ovaj način poboljšane su karakteristike klasifikatora opisanih u prethodnom delu teksta, evidentna je ušteda memorijskog prostora a rezultate je lakše tumačiti. Tačnost i vreme procesuiranja za 17 i 9 atributa obeleženi su kao ACC-17, t_p-17, ACC-9 i t_p-9, pri čemu je $t_p = t_{train} + t_{test}$, zbir vremena potrebnog za trening i testiranje modela, respektivno. U delu eksperimenata koji se odnosi na nenumeričke atribute nije korišćena FNN jer ulazni parametri u ovu strukturu moraju da budu isključivo brojevi. Kod treninga sa 9 atributa, izvršena je Min-Max normalizacija u granice [-1,1]. Rezultati dobijeni na 10 dnevnih zapisa iz Kyoto 2006+ baze su prikazani Tabelom 17. Broj instanci u jednom dnevnom zapisu kreće se u opsegu od ~57,300 do ~158,600.

Rezultati pokazuju visoku tačnost modela, pri čemu DT daje najbolje rezultate kad atributi nisu normalizovani. U ovom slučaju tačnost varira od 99.4% do 99.8%. wk-NN ima najveću vrednost kada su podaci normalizovani (99.5%), nakon čega slede DT (99.3%), SVM (98.3%) i k-NN (98.2%). Evidentna je mala degradacija tačnosti ukoliko se 9 numeričkih atributa koristi za evaluaciju modela. Na drugoj strani, znatna je ušteda u vremenu, koje je do četiri puta brže u odnosu na vreme neophodno za trening klasifikatora kada je 17 atributa smatrano relevantnim (Protic, Stankovic, 2018).

Tabela 17. Tačnost i vreme procesuiranja za 9 i 17 atributa iz Kyoto 2006+ baze

Dan	Broj instanci	Model	ACC-9	t_p-9	ACC-17	t_p-17
1	158572	k-NN	98.3%	275.72s	99.0%	1000.8s
		wk-NN	98.4%	277.32s	99.1%	1019.15s
		SVM	98.1%	449.35s	98.4%	467.7s
		Tree	97.2%	3.8452s	98.4%	14.241s
2	129651	k-NN	91.8%	175.84s	98.8%	695.88s
		wk-NN	91.8%	173.32s	99.0%	691.08s
		SVM	98.3%	254.32s	98.4%	304.56s
		Tree	97.3%	3.3104s	99.7%	9.4989s
3	128740	k-NN	98.2%	193.82s	98.6%	682.07s
		wk-NN	98.1%	194.81s	98.8%	690.58s
		SVM	97.8%	280.82s	97.9%	379.61s
		Tree	97.2%	3.3033s	99.8%	9.5367s
4	136625	k-NN	99.3%	194.83s	99.5%	782.1s
		wk-NN	99.4%	194.23s	99.7%	788.11s
		SVM	99.1%	217,32s	99.3%	234.59s
		Tree	98.3%	8.3169s	99.7%	10.001s
5	90128	k-NN	99.0%	101.28s	98.5%	731.2s
		wk-NN	99.1%	101.753s	99.6%	744.15s
		SVM	99.0%	86.283s	99.3%	230.33s
		Tree	98.4%	2.2308s	99.7%	10.855s
6	93999	k-NN	96.5%	109.25s	99.4%	354.09s
		wk-NN	96.5%	108.77s	99.5%	351.55s
		SVM	98.0%	111.83s	98.4%	149.03s
		Tree	97.5%	2.2613s	99.5%	6.9921s
7	80807	k-NN	98.8%	91.25s	99.4%	285.77s
		wk-NN	98.8%	91.267s	99.5%	285.25s
		SVM	97.9%	227.28s	98.1%	125.25s
		Tree	98.9%	2..2615s	99.4%	6.2339s
8	57278	k-NN	99.6%	42.704s	99.3%	77.224s
		wk-NN	99.3%	43.235	99.3%	77.121s
		SVM	99.2%	33.754s	99.1%	31.211s
		Tree	99.3%	1.743s	99.4%	3.7448s
9	58317	k-NN	99.1%	31.714s	99.3%	133.92s
		wk-NN	99.2%	31.738s	99.4%	134.4s
		SVM	99.1%	34.234s	99.2%	36.907s
		Tree	98.9%	1.7482s	99.5%	4.4372s
10	57278	k-NN	99.4%	43.734s	99.6%	129.99s
		wk-NN	99.5%	43.272s	99.6%	130.88s
		SVM	99.2%	30.239s	99.3%	37.894s
		Tree	99.4%	1.7489s	99.7%	4.4535s

Kada se, za 9 numeričkih atributa u trening uključi i FNN, dobijaju se rezultati prikazani Tabelom 18. Najveću tačnost imaju wk-NN model (99.5%) i k-NN model (99.36%) koji su, istovremeno, i najsporiji. FNN ima najkraće vreme procesuiranja, a najviša tačnost koju postiže je 92.21%. Veliku brzinu, kao i u prethodnom slučaju, ima i DT model. SVM ima, u proseku, nižu tačnost i duže vreme procesuiranja, ukoliko se poredi sa ostalim modelima (Protic, Stankovic, 2020).

Tabela 18. Tačnost i vreme procesuiranja za binarne klasifikatore i 9 atributa

Dan	Broj instanci	ACC-9 [%] t_p-9 [s]	FNN	k-NN	wk-NN	SVM	DT
1	158572	ACC-9	98.8	98.3	98.4	98.1	97.2
		t_p-9	26	275.72	277.32	449.35	3.8452
2	129651	ACC-9	97.67	91.8	91.8	98.3	97.3
		t_p-9	20	175.84	173.32	254.3	3.3104
3	128740	ACC-9	98.32	98.2	98.1	97.8	97.2
		t_p-9	8	193.82	194.81	280.82	3.3033
4	136625	ACC-9	99.21	99.3	99.4	99.1	98.3
		t_p-9	20	194.83	194.23	217.32	8.3169
5	90129	ACC-9	98.99	99.0	99.1	99.0	98.4
		t_p-9	11	101.28	101.753	86.283	2.2308
6	93999	ACC-9	98.12	96.5	96.5	98.0	97.5
		t_p-9	7	109.25	108.77	111.83	2.2613
7	81807	ACC-9	98.3	98.8	98.8	97.9	98.9
		t_p-9	10	91.25	91.26	227.28	2..2615
8	57278	ACC-9	99.14	99.36	99.3	99.2	99.3
		t_p-9	2	42.704	43.235	33.754	1.743
9	58317	ACC-9	98.97	99.1	99.2	99.1	98.9
		t_p-9	3	31.714	31.738	34.234	1.7482
10	57278	ACC-9	99.2	99.4	99.5	99.2	99.4
		t_p-9	2	43.734	43.272	30.239	1.2901

8.3. Primena hiperboličkog tangensa: prednosti u odnosu na druge metode skaliranja

U ovom podpoglavlju dokazan je pozitivan uticaj TH normalizacije na vreme procesuiranja, uz minimalnu degradaciju tačnosti modela. Za evaluaciju modela korišćeno je ~57,300 instanci, pri čemu je skup podeljen u tri dela, tako da je 70% podataka korišćeno za trening a po 15% za validaciju i testiranje.

Z-score standardizacija korišćena je za skaliranje atributa tako da instance budu centrirane oko srednje vrednosti 0, sa jediničnom standardnom devijacijom. Atributi nisu skalirani u fiksni opseg a trening modela izveden je instancama sa Gausovom raspodelom. Svi modeli su pokazali visoku tačnost, sa izuzetkom DT modela (Tabela 19).

Tabela 19. Uticaj Z-score standardizacije na evaluaciju modela

Model	ACC	t_p [s]	FPR	FNR	Precision	Recall	F1-score
k-NN	0.9943	102.35	0.002906	0.007039	0.997090	0.986960	0.991990
wk-NN	0.9948	103.25	0.002902	0.006401	0.997098	0.988147	0.992590
DT	0.9889	2.25	0.002091	0.007913	0.997909	0.985099	0.991463
SVM	0.9922	36.28	0.003125	0.010247	0.996875	0.980956	0.988850
FNN	0.9943	12.00	0.000996	0.014500	0.998100	0.985500	0.991761

Da bi bili izbegnuti problemi različitih skala primenjena je Min-Max normalizacija u opseg [0,1]. Ukoliko se uporedi sa prethodnim rezultatima, uočljivo je da ovaj metod skaliranja doprinosi sveukupnim karakteristikama klasifikatora (Tabela 20).

Tabela 20. Uticaj Min-Max normalizacije [0,1] na evaluaciju modela

Model	ACC	t_p [s]	FPR	FNR	Precision	Recall	F1-score
k-NN	0.9945	107.35	0.002656	0.007039	0.997344	0.986960	0.992125
wk-NN	0.9948	102.73	0.002803	0.006428	0.997197	0.988098	0.992627
DT	0.9947	2.79	0.001911	0.008389	0.998089	0.984428	0.991211
SVM	0.9915	36.99	0.003172	0.009718	0.996828	0.981948	0.989332
FNN	0.9953	11	0.001300	0.011000	0.997600	0.989000	0.993280

Min-Max normalizacija u opseg [-1,1] korišćena je da se reše problemi veoma velikih ili veoma malih izvoda. Međutim, i pored toga, nisu vidljivi značajni uticaji na karakteristike klasifikatora (Tabela 21).

Tabela 21. Uticaj Min-Max normalizacije u granice [-1,1] na evaluaciju modela

Model	Accuracy	t_p [s]	FPR	FNR	Precision	Recall	F1-score
k-NN	0.9941	103.2	0.003058	0.007386	0.997343	0.986502	0.991980
wk-NN	0.9958	105.3	0.002844	0.006321	0.997647	0.988147	0.992874
DT	0.9941	6.26	0.001909	0.008018	0.997940	0.985030	0.991443
SVM	0.9917	43.39	0.004379	0.010333	0.996725	0.981255	0.988893
FNN	0.9931	12.12	0.001560	0.011039	0.998034	0.982872	0.990415

TH skaliranje je primenjeno zbog tri značajne osobine:

- atributi su skalirani u isti, fiksni opseg, sa granicama ± 0.7616,
- funkcija $tanh(x)$ ubrzava trening i
- izbegnuto je preskakanje globalnog minimuma i/ili cik-cak kretanje po pravcu negativnog gradijenta.

Kada se poredi sa drugim metodama skaliranja, uočava se, osim za SVM, znatna redukcija vremena procesuiranja koja, za nearest neighbour modele, iznosi i više od dva puta (Tabela 22).

Tabela 22. Uticaj TH normalizacije na evaluaciju klasifikatora

Model	Accuracy	t_p [s]	FPR	FNR	Precision	Recall	F_1-score
k-NN	0.9930	56.101	0.002054	0.015549	0.995938	0.984535	0.990110
wk-NN	0.9940	56.335	0.002246	0.001961	0.995504	0.987807	0.991640
DT	0.9940	2.49	0.002059	0.008072	0.997292	0.985651	0.991747
SVM	0.9910	26.882	0.003204	0.015500	0.993670	0.980372	0.986940
FNN	0.9936	5	0.001300	0.015700	0.997500	0.984300	0.988862

Na osnovu prethodno prikazanih rezultata eksperimenata moguće je zaključiti da postoji opravdanost primene normalizacije tipa hiperboličkog tangensa i damping strategije Levenberg-Marquardt algoritma, s ciljem značajnog smanjenja vremena procesuiranja i male degradacije tačnosti, odnosno opravdanost hipoteze H1.

Tabelom 23 dat je zbirni pregled rezultata koji ukazuju na nekoliko zajedničkih karakteristika, za primenjene modele i tehnike skaliranja.

1. DT model ne reaguje na skaliranje, bez obzira koja od metoda skaliranja se koristi.
2. Rezultati za nearest neighbour klasifikatore su slični, sa neznatno većom tačnošću i vremenom procesuiranja kod wk-NN modela.
3. Z-score standardizacija nema značajnog pozitivnog efekta u odnosu na tri prikazane normalizacije atributa.
4. Podaci koji ukazuju na moguće negativne uticaje neizbalansiranosti Kyoto 2006+ baze podataka (*Precision, Recall, F*1*-score*) pokazuju da nema efekta balansiranosti sadržaja baze na evaluaciju prikazanih klasifikatora.
5. TH normalizacija smanjuje vreme evaluacije modela nekoliko puta u odnosu na ostale tehnike skaliranja.
6. Rezultati podržavaju pretpostavku da je opravdano korišćenje TH normalizacije, kako bi bio ubrzan trening modela.
7. Osim SVM, svi modeli imaju visoku tačnost i *F*1-score (wk-NN model ima najveću tačnost klasifikacije).

Tabela 23. Tačnost, vreme procesuiranja i F1-score za četiri metode skaliranja i binarne klasifikatore

	TH			**Min-Max [0,1]**			**Min-Max [-1,1]**			**Z-score**		
	ACC	tp [s]	F1	ACC	tp [s]	F1	ACC	tp [s]	F1	ACC	tp [s]	F1
FNN	0.9936	5	0.9889	0.9953	11	0.9933	0.9931	12	0.9904	0.9943	12	0.9917
DT	0.9940	2.5	0.9918	0.9947	2.8	0.9912	0.9941	6.3	0.9914	0.9889	2.2	0.9915
SVM	0.9910	26.9	0.9869	0.9915	36.9	0.9893	0.9917	43.4	0.9889	0.9922	36.3	0.9888
k-NN	0.9930	56.1	0.9901	0.9945	107.4	0.9921	0.9941	103.2	0.9920	0.9943	102.4	0.9911
wk-NN	0.9940	56.3	0.9916	0.9948	102.7	0.9926	0.9958	105.3	0.9929	0.9948	103.2	0.9926

Zbirni rezultati i opis zajedničkih karakteristika modela ukazuju da je, za potrebe formiranje XOR modela, kojeg čine najbrži i najprecizniki binarni klasifikator, opravdano korišćenje binarnih klasifikatora sa težinskim koeficijenata, odnosno opravdana je hipoteza H2.

8.4. XOR detekcija i procena konfliktnih odluka o anomalijama

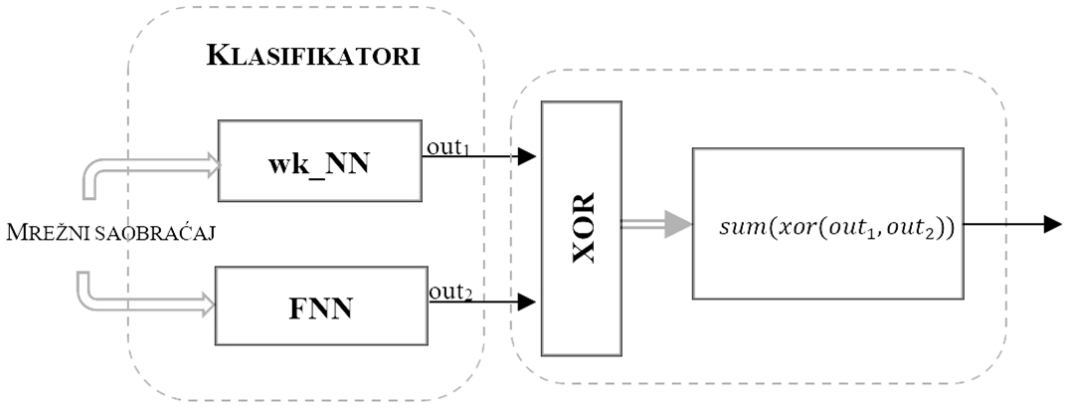
Prezentacija izlaza iz klasifikatora izuzetno je bitan aspekt u detekciji anomalija. Kada podaci o mrežnom saobraćaju pripadaju jednoj od klasa, koristi se tehnika dodele binarne labele nepoznatoj instanci. Međutim, bez obzira na tačnost klasifikatora, s vremena na vreme on greši u detekciji. U ovoj disertaciji predstavljen je novi model koji obezbeđuje da kontradiktorne odluke u detekciji anomalija između dva klasifikatora koji rade u paraleli budu okidač alarma konflikta u odluci. Naime, ukoliko je neophodno zaštititi veoma osetljive podatke, visok procenat konfliktnih odluka pomoći će u podizanju dodatnih upozorenja donosiocima odluka o mogućim napadima na računarsku mrežu koje je teško detektovati drugim sistemima za zaštitu podataka.

Ideja je primeniti bitsku XOR operaciju na izlaze iz klasifikatora, u XOR bloku čija funkcija je, ustvari, dvojaka:

- Prvo, XOR blok detektuje sličnosti/razlike u odlukama binarnih klasifikatora, za svaki događaj, odnosno za pojavu instance sa podacima.
- Na drugoj strani, zbir svih dobijenih vrednosti je, u vidu kumulativne sume, zabeležen u akumulatorskom registru, čija je dužina promenljiva, i može se biti npr. dnevni zapis mrežnog saobraćaja, ili može biti fiksne dužine (npr. 1kb).

U eksperimentima prikazanim u ovom delu rada, broj instanci na dnevnom nivou je bio referenca za procentualnu vrednost broja konfliktnih odluka.

U prethodnom delu teksta opisan je princip rada XOR detektora, uz dijagram toka, korake algoritma, i odgovarajuću matematičku podršku, za dva binarna klasifikatora. U eksperimentalnom delu opisano je i primenjeno konkretno rešenje bazirano na brzoj FNN i veoma preciznom ali sporijem wk-NN modelu. Konceptualni dizajn ovog modela prikazan je Slikom 17.



Slika 17. XOR detektor

Oba klasifikatora, bez obzira na metod koji koriste za trening, su prvo terenirana na istom skupu instanci. Trening FNN je izveden LM algoritmom, i formirana je prenosna funkcija na osnovu podataka iz trening skupa. wk-NN model traži *k* najbližih suseda i donosi odluku na osnovu prethodnih iskustava, koristeći ponderisanje bazirano na proračunu srednje kvadratne greške i inverzije kvadrata distance između suseda.

Razlika između ova modela je u načinu (vremenu) na koji algoritam vrši apstrakciju iz podataka. FNN uči sa prvim instancama trening skupa, trening je sporiji, a klasifikacija nepoznate instance je brza. wk-NN memoriše trening skup, što je brzo, ali poređenje nepoznate instance sa poznatim podacima dugo traje, odnosno klasifikacija je spora.

Da bi bila demonstrirana funkcionalnost prikazanog detektora izvedeni su eksperimenti nad instancama šest nasumično odabranih dana iz Kyoto 2006+ baze podataka. Svaki dnavni zapis podeljen je tako da je dve trećine instanci korišćeno za evaluaciju modela, a jedna trećina je korišćena za proračun kumulativne sume. Dobijeni rezultati koji su prikazani Tabelom 24 ukazuju na činjenicu da su konfliktne odluke evidentne i da ih je moguće detektovati.

Tabela 24. Detekcija različitih odluka o anomalijama

Broj instanci	Različite odluke	Različite odluke [%]
57270	1160	6.08
57280	460	2.41
58300	100	0.51
57278	1157	6.06
58335	98	0.5
57279	456	2.39

Viši procenat različitih odluka o anomalija ukazuje na nivo nesigurnosti koji svaki pojedinačni klasifikator unosi prilikom detekcije, kao i na nedovoljno poznavanje događaja koji se dešavaju u računarskoj mreži ali ih nije moguće detektovati jednim klasifikatorom, niti su isključivo vezani za odlučivanje klasifikatora.

Rezultati pokazuju i da broj različitih odluka ne zavisi od veličine dnevnog zapisa. Nesigurnost u odlučivanju može biti izazvana i greškom u podacima, rezidualnim greškama modela, nepoznatim/neodređenim malicioznim napadima, i slično.

Na osnovu dobijenih rezultata moguće je dokazati i opravdanost osnovne hipoteze da je XOR detekcijom konfliktnih odluka o anomalijama moguće povećati stepen prepoznavanja malicioznosti i/ili grešaka u računarskim mrežama.

Neophodno je razumeti da XOR detektor zaista pomaže u detekciji ovih događaja, ali da njegova namena nije predikcija verovatnoće ovakvih događaja. On je alat za podizanje alarma o konfliktu, a kriterijum za donošenje odluka moguće je izabrati na različite načine, u zavisnosti od osetljivosti podataka, primenjenih tehnologija, zakonske prakse itd.

9. ZAKLJUČAK

Detekcija anomalija u kompleksnim računarskim mrežama je jedan od najvećih istraživačkih izazova današnjice. Anomalije, koje ne registruje sistem za zaštitu od napada na računarsku mrežu, mogu da izazovu ozbiljne posledice po poverljivost, integritet ili raspoloživost podataka.

Sistemi za zaštitu informacija koji su bazirani na detekciji anomalija prate ponašanje računarske mreže s ciljem da detektuju neregularnosti u obrascu koji je određen statističkim modelom poznatog, normalnog ponašanja mreže. Ovi sistemi traže varijacije u mrežnom saobraćaju koje mogu da ukažu na neregularnosti izazvane različitim tipovima malicioznosti.

Binarni klasifikatori bazirani na nadgledanom mašinskom učenju koriste se da prepoznaju normalno ponašanje računarske mreže, identifikuju abnormalnosti i aktiviraju odgovarajuće alarme. U radu je prikazan rad pet binarnih klasifikatora koji su bazirani na nadgledanom mašinskom učenju: k-najbližih suseda, k-najbližih suseda sa težinskim koeficijentima, stabla odlučivanja, vektori oslonca i feedforward neuronska mreža.

Da bi bio kreiran model normalnog ponašanja računarske mreže u kompleksnim sistemima neophodan je veliki broj instanci, što trening modela čini dugotrajnim, i zahtevnim u smislu zauzetosti memorijskih resursa. Iz tog razloga se primenjuje selekcija atributa kojom se skup podataka redukuje tako da se isključivo relevantni podaci koriste za evaluaciju modela. Pored toga, da se smanji međusobni uticaj atributa koji su različito skalirani, moguće je primeniti različite metode skaliranja, kao što su normalizacija ili standardizacija. U ovom radu prikazana je metoda selekcije devet numeričkih atributa iz Kyoto 2006+ baze realnih podataka snimljenog mrežnog saobraćaja i namenjene isključivo za istraživanje detekcije anomalija. Takođe, pored Z-score standardizacije i dve Min-Max normalizacije, u granice [0,1] i [-1,1], prikazana je i nova metoda tangens hiperboličke normalizacije sa ubrzanim optimizacionim procesom koji je baziran na Levenberg-Marquardt algoritmu.

Iako se pokazuje da su svi prezentovani klasifikatori izuzetno tačni u proceni, FNN se pokazala najbržim modelom. Na drugoj strani, wk-NN model je značajnije sporiji, ali izuzetno tačan. Iako su bazirani na različitim metodama obučavanja, obe metode koriste ponderisanje koeficijenata i proračunu kvadratnih grešaka u treningu.

Dokazano je da, i pored visoke tačnosti, jedan model može da klasifikuje mrežni saobraćaj kao normalan, dok drugi detektuje anomaliju, odnosno odluka je različita. Za podizanje dodatnih alarma o konfliktu koristi se XOR detektor kojeg čine dva binarna klasifikatora, blok za izvođenje bitske XOR operacije nad izlazima klasifikatora, i akumulator, registar za izračunavanje kumulativne sume ukupnog broja konfliktnih odluka.

Eksperimentalni rezultati potvrđuju pretpostavke da tangens hiperbolička normalizacija znatno redukuje vreme procesuiranja, u odnosu na druge metode skaliranja. Levenberg-Marquardt algoritam ubrzao je trening modela zbog činjenice da se tehnika pretrage po padu negativnog gradijenta u blizini optimalnog rešenja nelinearnog problema, menja mnogo bržim Gaus-Njutnovim algoritmom. Za klasifikatore tipa najbližih suseda ovo smanjenje iznosi više od dva puta.

Dokazano je i da je moguće detektovati do ~6% konfliktnih odluka o anomalijama, pri čemu treba konstatovati da broj detektovanih anomalija ne zavisi od broja instanci u dnevnom zapisu, a da procenat različitih odluka predstavlja indikator nesigurnosti u

klaisifikaciji i nedovoljno znanje, odnosno poznavanje događaja koji je registrovan kao konflikt ali nije izazvan odlukama klasifikatora. Neodređenost, odnosno nesigurnost u dobijenim rezultatima može biti izazvana greškama u podacima, rezidualnim greškama u modelu, neidentifikovanim malicioznim napadima i slično.

Treba naglasiti da, iako XOR detektor koji je prikazan u ovom radu, može da pomogne rešavanju opisanih problema, njegova osnovna namena nije da izvrši predikciju pojave nepoznatog događaja. Njegova namena je da obezbedi informaciju, alarm ili upozorenje o pojavi konflikta onim autoritetima koji će doneti odluke u zavisnosti od sopstvenih potreba za zaštitu osetljivih podataka.

10. REFERENCE

1. Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, k. V., Mohamed, N. A., Arshad, N. 2018. State-of-the-art in artificial neural network applications: A survey. *Heliyon* 4(11) e00938. https://doi.org/10.1016/j.heliyon.2018.e00938
2. Ahmad, T., Aziz, M. N. 2019. Data preprocessing and feature selection for machine learning intrusion detection systems. *ICIC Express Letters* 13(2), pp. 93-101. http://doi.org/10.24507/icicel.13.02.93
3. Ahmed, I., Shin, H., Hong, M. 2011. Fast Content-Based File Type Identification. Project: Digital Forensics. http://doi,org/10.1007/978-3-642-24212-0_5
4. Aickelin, U., Cayzer, S. 2002. The Danger Theory and Its Application to Artificial Immune Systems. *Proceedings of the 1st Internal Conference on ARtificial Immune Systems (ICARIS-2002)*. Canterbury. UK, pp. 141-148.
5. Aickelin U., Dasgupta, D. 2005. Artificial Immune Systems. In E. Burke and G. Kendal (Eds.) *Introductory Tutorials in Optimization, Decision Support and Search Methodology*. Kluwer.
6. Aissa, N.B., Guerroumi, M. 2016. Semi-Supervised Statistical Approach for Network Anomaly Detection. *Procedia Computer Science* 83, pp. 1090-1095.
7. Al-Haj Baddar, S., Merlo, A., Migliardi, M. 2014. Anomaly Detection in Computer Networks. *Journal of Wireless Mobile Networks, Ubiquitos Computinf and Dependable Applications* 5(4), pp. 29-64.
8. Al-Imran, M., Ripon, S.H. 2021. Network Intrusion Detection: An analytical assessment using deep learning and state-of-the-art machine learning models. *Int. J. Comput. Intell. Syst.* 14(200). https://doi.org/10.1007/s44196-021-00047-4
9. Almomani, O. 2020. A Feature Selection Model for Network Intrusion Detection System Based on PSO, GWO, FFA and GA Algorithms. *Symmetry* 12(6), 1046 (2020). https://doi.org/10.3390/sym12061046
10. Ambedkar, Ch. & Babu V. K. 2015. Detection of Probe Attacks Using Machine Learning Techniques. *International Journal of Research Studies in Computer Science and Engineering* 2(3), pp. 25-29.
11. Ambusaidi, M. A., Nanda, P. 2014. Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE Transactions on Computers*, pp. 1-13 (November 2014).
12. Ambusaidi, M. A., He, X, Nanda, P., Tan, Z. 2016. Building an Intrusion Detection System Using a Filter-Based Feature Selection Algorithm. *IEEE Transaction on Computers* 65(10), pp. 2986-2998.
13. Aamir, M., Zaidi, S. M. A. 2019. DDoS Attack detection with feature engineering and machine learning, the framework and performance evaluation. *Int. J. Inf. Secur*. 18, pp. 761-785. https://doi.org/10.1007/s10207-019-00434-1
14. Artur, A. 2021. Review the performance of the Bernoulli Naive Bayes Classifier in Intrusion Detection Systems using Recursive Feature Elimination with Cross-validated Selection of the Best Number of Features. *Procedia Computer Science* 190, pp. 564-570.

15. Belavagi, M.C., Munyal, B. 2016. Performance Evaluation of Supervised Machine Learning Algorithms for Intrusion Detection. *Procedia Computer Science* 89, pp. 117-123.
16. Biswas, S.K., Bordoloi, M., Purkayastha, B. 2016. Review of Feature Selection and Classification using Neuro-Fuzzy Approaches. *International Journal of Applied Evolutionary Computation* 7(4), pp. 28-44 http://dx.doi.org/10.4018/IJAEC.2016100102
17. Bjelica, M. 2009. Funkcije za kvanitfikovanje uzajamne sličnosti telekomunikacionih servisa. 17. telekomunikacioni forum TELFOR 2009. Beograd, novembar 24 – 26., pp. 914-918.
18. Broere, F., Apasov, S.G., Sitkovsky, M.V., van Eden, W. 2011. T cell subsets and T cell-mediated immunity. In Nijkamp, F.P., Parhman, M.J. (Eds). *Principles of immunopharmacology*. Springer. ISBN: 978-3-0346-0135-1. pp. 16-27.
19. Brownlee, J. 2013. A Tour of Machine Learning Algorithms. [Online] Dostupno na: http://www. machinelearningmastery.com Preuzeto: 23.03.2017. godine.
20. Burgess, M. 1998. *Computer Immunology*, 12th USENIX Conference on System Administration, Boston, MA, USA, 06 – 11 Dec. 1998, pp. 283-298.
21. Burnet, F.M. 1959. *The Clonal Selection Theory of Acquired Immunity*. Vanderbilt University Press. Nashville. Tennessee. USA.
22. Chandola, V., Banerjee, A., Kumar, V. 2009. Anomaly detection. *ACM Comput. Surv.*, 41(3), pp. 1-58.
23. Choudhury, A. 2019. *What are feature selection techniques in machine learning*. [Online] Dostupno na: https://analyticsindiamag.com/what-are-feature-selection-techniques-in-machine-learning/. Preuzeto 10.10.2021. godine
24. Clayomb, W.R., Legg, P.A., Gollmann, D. 2014. Guest editorial: Emerging trends in research for insider threat detection. *JoWUA* 5(2), pp. 1-6.
25. Debar, H., Dacier, M., Wespi, A. 1999. Towards a taxonomy of intrusion-detection systems. *Comput. Networks* 31(8), pp.805-822.
26. de Castro, L.N., Von Zuben, F.J. 2001. aiNet: An artificial immune network for data analysis. In: H.A. Abbas, R.A. Sarker. and C. Newton (Eds.) *Data Mining: A Heuristic Approach*. Idea Group Publishing, USA.
27. de Castro L.N., and Timmis, J. 2002. *Artificial Immune Systems: A New Computational Intelligence Approach.* Springer Publishing.
28. Dimic, G., Rancic, D., Spalevic, P., Plecic, K. 2017. Comparative study: Feature selection methods in the blended learning environment. *Facta Universitatis, Series: Automatic Control and Robotics* 16(2), pp. 95-116, https://doi.org/10.22190/FUACR1702095D
29. Dudani, S.A. 1976. The distance-weighted k-Nearest-Neighbor rule, *IEEE Transactions on Systems, Man, and Cybernetics SMC* 6, pp. 325-327. https://doi.org/10.1109/TSMC.1976.5408784
30. Dy, J.G., Brodley, C.E. 2005. Feature selection for unsupervised learning. *Journal of Machine Learning Research* 5, pp. 845-889.
31. Esfetanaj, N. N., Nojavan, S. 2018. Chapter 4: The Use of Hybrid Neural Networks, Wavelet Transform and Heuristic Algorithm of WIPSO in Smart

Greed to Improve Short-Term Prediction of Load, Solar Power, and Windy Energy." In Kazem Zare and Sayyad Nojavan (eds.) *Operation of Distributed Energy Resources in Smart Distribution Networks*. Elsevier Inc, Academic Press, pp. 75-100. https://doi.org/10.1016/C2017-0-02272-3

32. Ferryian, A., Thamrin, A.H., Takeda, K., Murai, J. 2021. Generating network intrusion detection dataset based on real and encrypted synthetic attack traffic. *MDPI Applied Sciences* 11, pp. 2-17.

33. Forrest, S., Perelson, A.S., Allen, L. and Cherukuri, R. 1994. Self-nonself discrimination in a computer. Research in Security and Privacy. *Proceedings of the 1994 IEEE Computer Society Symposium* 16th June 1994. pp. 2002-2012.

34. Garcia, S., Luengo, J., Herera, F. 2015. Data Preparation Basic Model., In: *Data Preprocessing in Data Mining*. Intelligent System Reference Library 72, Springer, Cham, pp. 39-57. https://doi.org/10.1007/978-3-319-10247-4_3.

35. Haag, C.R. 2007. *An Artificial Immune System-Inspired Multiobjective Evolutionary Algorithm with Application to the Detection of Distributed Computer Network Intrusions.* AFIT/GCS/ENG/07-05. Air Force Institute of Technology. Wright-Peterson Air Force Base. Ohio.

36. Hardesty, L. 2017. Explained: neural networks. MIT News on campus and around the world. *Int. J. Netw. Secur. Appl.* 12(1), pp. 1-18. [Online] Dostupno na: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414. Preuzeto: 11.07.2021. godine

37. Hechenbichler, K., Schliep, K. 2004. Weighted *k*-Nearest-Neighbor Techniques and Ordinal Classification. *Sonderforschungsbereich* 386, Paper 399, Institut fur Statistik Sonderforschungsbereich 386. Ludwig-Maximilian Universitat. Munchen. [Online] Dostupno na: http://epub.bu.uni-muenchen.de/. Preuzeto 29.10.2018. godine.

38. Hodo, E., Bellekens, X., Hamilton, A., Tachtatzis, C., Atkinson, R. 2017. *Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey*. University of Strathclyde & University of Abertay Dundee. pp. 1-43.

39. Ji, Z., Dasgupta, D. 2007. Revisiting Negative Selection Algorithms. *Evolutionary Computations* 15(2), pp. 223-251.

40. Jie, C., Jiawei, L., Shulin, W., Sheng, Y. 2018. Feature selection in machine learning: A new perspective. *Neurocomputing* 300(26), pp. 70-79. https://doi.org/10.1016/j.neucom.2017.11.077

41. Kajal, R., Devi, S. 2013. Intrusion Detection Systems: A Review. *Journal of Network and Information Security* 1(2), pp. 15-21.

42. Kalavadekar, P. N., Sane, S. S. 2019. Building an Effective Intrusion Detection System using combined Signature and Anomaly Detection Techniques. International *Journal of Innovative Technology and Exploring Engineering* 8(10), pp. 429- 435.

43. Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J. 2019. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, pp. 2-20.

44. Khammassi, C., Krichen, S. 2017. A GA-LR wrapper approach for feature selection in network intrusion detection. *Computers and Security* 70, pp. 255-277. https://doi.org/10.1016/j.cose.2017.06.005

45. Kim, J., Bentley, P. 1999. *An Artificial Immune Model for Network Intrusion Detection*. Universiy College London. pp. 1-7.

46. Kumar, S., Gupta, S., Arora, S. 2021. Research trends in network-based intrusion detection systems: a review. *IEEE Access* 9, pp. 157761-157779. https://doi.org/10.1109/ACCESS.2021.3129775

47. Kumar, V., Sangwan, O.P. 2012. Signature Based Intrusion Detection Using SNORT. *International Journal of Computer Applications & Information Technology* 1(3), pp. 36-41.

48. Levenberg, K. 1944. A method for the solution of certain problems in least squares. *Quarterly of Applied Mathematics* 5, pp. 164-168.

49. Luque, A., Carrasco, A., Martin, A., de las Heras, A. 2019. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition* 91, pp. 216-239.

50. Mallissery, S., Prabhu, J., Ganiga, R. 2011. Survey on intrusion detection methods. *3rd International Conferece on Advantages in Recent Technologies in Communication and Computing (ARTCom 2011)*, pp. 224-228.

51. Markovic, V.S., Njegus, A., Marjanovic, M. 2020. Anomalies detection in the application logs using Kohonensom som machine learning argorithm. *International Scientific Conference on Information Technology and Data Related Research, SINTEZA 2020*, pp. 275-282.

52. Marquardt, D. 1963. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal in Applied Mathematics* 11(2), pp. 431-441.

53. Matzinger, P. 1994. Tolerance, danger, and the extended family. *Annu. Rev. Immunol.* 12, pp. 991-1045.

54. Maza, S., Touahria, M. 2018. Feature Selection Algorithms in Intrusion Detection System: A Survey. KSII Transactions on Internet and Information Systems 12(10), pp. 5079- 5099. http://dx.doi.org/10.3837/tiis.2018.10.024

55. McCarthy, R. 2014. Network analysis with the Bro security monitor. [Online] Dostupno na: https://www.admin-magazine.com/Archive/2014/24/Network-analysis-with-the-Bro-Network-Security-Monitor Preuzeto: 2109.2022. godine.

56. Milosavljević, M. 2017. *Introduction to Machine Learning*. [Online] Dostupno na: http://www. predmet.singidunum.ac.rs. Preuzeto: 22.03.2017. godine.

57. Mishra, A., Cheng, A. M. K., Zhang, Y. 2020. Intrusion Detection Using Principal Component Analysis and Support Vector Machines. *2020 IEEE 16th International Conference on Control and Automation* (ICCA), pp. 907-912.

58. Mitchell, T. 1997. *Machine Learning*. McGraw Hill. ISBN 0-07-042807-7.

59. Mitra, P., Murthy, C.A., Pal, S. 2002. Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, pp. 301-312.

60. Mittal, N.K. 2016. A survey on Wireless Sensor Network for Community Intrusion Detection Systems. *3rd International Conference on Recent Advances in Information Technology (RAIT)*, pp. 107-111.

61. Modi, C., Patel, D., Borisaniya, H., Patel, A., Rajaran, M. 2013. A survey of intrusion detection techniques in Cloud. *J. Netw. Comput. App.* 36(1), pp.42-57.

62. Mukrimah Nawir, M., Amir, A., Lynn O.,, Yaakob, N., Badlishah Amad, R. 2018. Performances of Machine Learning Algorithms for Binary Classification of Network Anomaly Detection System. *J. Phys.: Conf. Ser.* 1018.

63. Musheer, R.A., Verma, C.K., Srivastava, N.. 2017. Dimension reduction methods for microarray data: a review. *AIMS Bioengineering* 4(2), pp. 179-107. http://dx.doi.org/10.3934/bioeng.2017.1.179

64. Najafabadi, M.N., Khoshgoftaar, T.M., Selyia, N. 2016 Evaluating Feature Selection Methods for Network Intrusion Detection with Kyoto data. *International Journal of Reliability, Quality and Safety Engineering* 23(1). http://dx.doi.org/10.1142/S0218539316500017

65. Novaković, J. 2013. Rešavanje klasifikacionih problema mašinskog učenja. *Reinženjering poslovnih procesa. Knjiga 4*, pp. 3-197.

66. Obaid, H. S., Dheyab, A., Sabry, S. S. 2019. The Impact of Data Pre-Processing Techniques and Dimensionality Reduction on the Accuracy of Machine Learning," *9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference* (IEMECON), pp. 279-283. https://doi.org/10.1109/IEMECONX.2019.8877011

67. Osnaiye, O. Ogundile, O., Aina, F., Peniola, A. 2019. Feature Selection for Intrusion Detection System in cluster-based heterogeneous wireless sensor networks. *Facta Universitatis, Series: Electronics and Energetics* 32(2), pp. 315-330. http://doi.org/10.2298/FUEE1902315O

68. Pai, V., Devidas, B., Adesh, N. D. 2021. Comparative analysis of machine learning algorithms for intrusion detection. *IOP Conf. Series: Material Science and Engineering* 1013, pp. 1-7.

69. Pejić, B. 2017. Primena algoritma stabla odlučivanja u prepoznavanju ponašanja i zdravstvenih rizika kod starijih osoba. Univerzitet u Beogradu. Matematički fakultet.

70. Perez, D. Alonso, S., Moran, A., Prada, M. A., Fuentes, J. J., Domingez, M. 2019. Comparison of Network Intrusion Detection Performance Using Feature Representation. In: Macintyre, J., Illadis, L., Maglogoiannis, I., Jayne C. (eds.) Engineering Applications of Neural Networks. EANN 2019. *Communications in Computer and Information Science* (1000), Springer, Cham. https://doi.org/10.1007/978-3-030-20257-6_40

71. Porkodi, R.. 2014. Comparison on filter-based feature selection algorithms: An overview. *International Journal of Innovative Research in Technology and Science* 2(2), pp. 108-113.

72. Protić, D. 2018. Review of KDD CUP '99, NSL-KDD and Kyoto 2006+ Datasets. *Vojnotehnički glasnik/Military Technical Courier* 66(3), pp.580-595. https://doi.org/10.5937/vojteh66-16670

73. Protic, D., Gaur, L., Stankovic, M., Rahman, M. A. 2022. Cybersecurity in smart cities: Detection of opposing decisions on anomalies in the computer network behavior. *Electronics* 11(3718), pp. 1-16. https://doi.org/10.3390/electronics11223718

74. Protić, D., Milosavljević, M. 2005. Generalizaciona svojstva različitih klasa linearnih i nelinarnih modela govornog signala. *Zbornik radova Festivala informatičkih dostignuća INFOFEST*, Budva, pp. 248-257.

75. Protic, D., Stankovic, M. 2018. Anomaly-Based Intrusion Detection: Feature Selection and Normalization Instance to the Machine Learning Model Accuracy. *European Journal of Engineering and Formal Sciences* 1(3), pp. 43-48.

76. Protić, D., Stanković, M. 2020. Detection of Anomalies in the Computer Network Behaviour. 5th International Conference on Engineering and Formal Science. 24-25 January 2020, Brussels. *European Journal of Engineering and Formal Sciences* 4(1) 2020, pp. 7-13. ISBN 978-164786089-9,. https://doi.org/10.26417/ejef.v4i1.p7-13

77. Protic, D., Stankovic, M. 2020a. A Hybrid Model for Anomaly-Based Intrusion Detection in Complex Computer Networks. *21st International Arab Conference on Information Technology*, 6th of October 2020, Giza, Egypt, pp. 1-8. https://doi.org/10.1109/acit50332.2020.9299965

78. Protić, D., Stanković, M. 2021 The q-Levenberg-Marquardt Method for Unconstrained Nonlinear Optimization, pp.1-5. http://arxiv.org/abs/2107.03304

79. Protić, D. Stanković, M., Antić, V. 2022. Anomaly-Based Intrusion Detection Systems in Computer Networks: Feedforward Neural Networks and Nearest Neighbour Models as Binary Classifiers. *Proceedings of Conference on Computational Intelligence for Engineering and Management Applications – CIEMA 2022*, Proceedings by Springer LNEE Series (Indexed in Scopus). [Online] March 26-27, 2022.

80. Protić, D. Stanković, M., Antić, V. 2022a. WK-FNN design for detection of anomalies in the computer network traffic. Facta Universitatis, Series: Electronics and Energetics 35(2), pp. 269-282. https://doi.org/10.2298/FUEE2202269P

81. Rahman, M. A., Islam, M. Z. 2012. CRUDAW: A novel fuzzy technique for clustering records following user defined attribute weights. *Proceedings of 10th Australasian Data Mining Conference* 134, pp. 27-42.

82. Rahman, M. A., Islam, Z. 2015. AWST: A novel attribute weight selection technique for data clustering. *Proceedings of 13th Australasian Data Mining Conference*, Sidney, Australia, pp. 51-58.

83. Rice, D. M. 2013. Causal Reasoning, in Calculus of Thought: Neuromorphic Logistic Regression in Cognitive Machines, Academic Press Inc.

84. Rehman, R.U. 2003. *Intrusion Detection Systems with Snort: What is Intrusion Detection*. Prentice Hall PTR. Upper Saddle River, New Jersey 07458, pp. 5-10.

85. Riecke, L., Esposito, F., Bonte, M., Formisano, E. 2009. Hearing Illusory Sounds in Noise: The Timing of Sensory-Perceptual Transformations in Auditory Cortex. *Neuron* 64(4), pp. 550-561. https://doi.org/10.1016/j.neuron.2009.10.016

86. Rigaki, M. 2017. *Adversarial Deep Learning Against Intrusion Detection Classifiers*. Lulea University of Technology, pp. 1-33.

87. Ring, M., Wunderlich, S., Scheuring, D., Landes, D., Hotho, A. 2019. A Survey of Network-based Intrusion Detection Data Sets. arXiv:1903.02460v2 [cs.CR] 6 Jul 2019, pp. 1-17.

88. Rosely, N. F, Sallehuddin, R., Zain, A.M. 2019. Overview Feature Selection *Algorithm. Journal of Physic: Conference Series* 1192, 012068. http://dx.doi.org/10.1088/1742-6596/1192/1/012068

89. Salo, F. 2019. Towards Efficient Intrusion Detection using Hybrid Data Mining Techniques. Western University, Electronic Theses and Dissertations Repository, 6363.

90. Sebastiani, F., Fabrizio. 2002. Machine learning in automated text categorization. *ACM Comput. Surv.* 34(1), pp. 1-47.

91. Shahin, A.J., Pitt, M.A. 2012. Alpha activity marking word boundaries mediates speech segmentation. *European Journal of Neuroscience* 36(12), pp. 3740-3748. https://doi.org/10.1111/ejn.12008

92. Shimonsky, R. 2002. *What You Need to Know About Intrusion Detection System?*

93. SIGKDD - KDD Cup, KDD Cup 1999. 2018. Computer network intrusion detection. [Online] Dostupno na: www.kdd.org Preuzeto 21.09.2022).

94. Silva, L.M., de Sa, J.M., Alexandre, L.A. 2008. Data classification with multilayer perceptrons using generalized error function. *Neural Networks* 21(9), pp.1302-1310. https://doi.org/10.1016/j.neunet.2008.04.004

95. Singh, R., Kumar, H., Singla, R.K. 2015. An intrusion detection system using network traffic profiling and online sequential extreme learning machine. *Expert Systems with Applications* 42(22), pp.8609-8624.

96. Serkani, E., Gharaee, H., Mohammadzadeh, N. 2019. Anomaly detection using SVM as classifier and DT for optimizing feature vectors. *ISeCure* 2019 11(2), pp. 159-171.

97. Shen, J. 2012. *Network Intrusion Detection by Artificial Immune System*. Electrical and Computer Engineering College of Science. Engineering and Health RMIT University. pp.1-63.

98. Soltani, M., Siavoshani, M. J., Jahangir, A. H. 2021. A content-based deep intrusion detection system. *Int. J. Inf. Secur*. https://doi.org/10.1007/s10207-021-00567-2

99. Song, L., Smola, A., Gretton, A., Borgwardt, K., Bedo, J. 2007. Supervised feature selection via dependence estimation. *Proceedings on 24th International Conference on Machine Learning*, pp. 823-830. https://doi.org/10.1145/1273496.1273600

100. Song, J., Takakura, H., Okabe, Y., Eto, M., Inoue, D., Nakao, K. 2011. Statistical Analysis of Honeypot Data and Building of Kyoto 2006+ Dataset for NIDS Evaluation. *Proc. 1st Work-shop on Building Anal. Datasets and Gathering Experience Returns for Security*. Salzburg, April 10-13 2011, pp. 29-36.

101. Stallings, W., Brown, L. .2015. *Computer Security: Principles and Practice*, 3rd ed. Pearson.

102. Stankovic, M., Antonijevic, M., Bacanin, N., Zivkovic, M., Tanaskovic, M., Jovanovic, D. 2022. Feature selection by Hybrid Artificial Bee Colony Algorithm for Intrusion Detection. *2022 International Conference on Edge Computing and Applications*, 13-15 October 2022, https://doi.org/10.1109/ICECAA55415.2022.9936116

103. Stefan, G. M., Malita, M. 2014. Can one-chip parallel computing be liberated from ad hock solutions? A computational model-based approach and its

implementation, *Advances in Information Science and Applications* 2, pp. 582-597.

104. Suman, C., Tripathy, S., Saha, S. 2019. Building an Effective Intrusion Detection Systems using Unsupervised Feature Selection in Multi-objective Optimization Framework. arXiv:1905.06562v1 [cs.NE].

105. Sun, X., Yagnik, S., Viswanathan, R., CAO, L. 2022. Performance of XOR rule for decentralized detection of deterministic signal in bivariate Gaussian noise. *IEEE Access* 10, pp. 8092-8102. https://doi.org/10.1109/ACCESS.2022.3243105

106. Sutton, R. S., Barto, A. G. 1998. *Reinforcement Learning an Introduction*. A Bradford Book.

107. Tang, J., Alelyani, S., Liu, H. 2014. Feature selection for classification: A review. CRC Press, pp. 1-29. Dostupno na: http://www.math.chalmers.se/Stat/Grundutb/GU- /MSA220/S18/featselect.pdf. Preuzeto: 12.12.2021. godine.

108. Teodorescu, H. N. L. 2017. Sensors based on nonlinear dynamic systems – A survey, *International Conference on Applied Electronics* (AE), pp. 1-10. https://doi.org/10.23919/AE.2017.8053572

109. Thakkar, A., Lohiya, R.A. 2021. A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions. *Artif Intell Rev*. https://doi.org/10.1007/s10462-021-10037-9

110. Timmis, J., Hone, A., Stibor, T., Clark, E. 2008. Theoretical advances in artificial immune systems. *Theoretical Computer Science* 403, pp. 11-32.

111. Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., Lin, W.-Y. 2009. Intrusion detection by machine learning. *Expert Syst. Appl.* 36(10), pp. 11994-12000. https://doi.org/10.1016/j.eswa.2009.05.029

112. Ullah, F., Babar, M. A. 2019. Architectural Tactics for Big Data Cybersecurity Analysis Systems: A Review. *Journal of Systems and Software* 151, pp.81-118

113. Umar, M.A., Zhanfang, C., Liu, Y. 2020. Network Intrusion Detection Using Wrapper-based Decision Tree for Feature Selection. *Proceedings of the 2020 International Conference on Internet Computing for Science and Engineering*, pp. 5–13. https://doi.org/10.1145/3424311.3424330 16

114. Vijayarani, S., Sylviaa. S. M. 2015. Intrusion detection system – A study. *International Journal of Security, Privacy and Trust Management (IJSPTM)* 4(1), pp. 31-44.

115. Wang, Y., Liu, D., Huang, T. S. 2019. Chapter 6: Signal Processing." In: Zhangyang Wang, Yun Fu, Thomas S. Huang (eds.) *Computer Vision and Pattern Recognition, Deep Learning Through Sparse and Low-Rank Modeling*, Academic Press, pp. 121-142. https://doi.org/10.1016/B978-0-12-813659-1.00006-8

116. Wang, Z., Cai, L., Yf. Su, et al. 2019a. An Inexact Affine Scaling Levenberg-Marquardt Method Under Local Error Bound Conditions. *Acta Math. Appl. Sin. Engl.*, Ser. 35, pp. 830–844. https://doi.org/10.1007/s10255-019-0856-0.

117. Weston, J., Elisseff, A., Schoelkopf, B., Tipping, M. 2003. Use of the zero norm with linear models and kernel methods. *Journal of Machine Learning Research* 3, pp. 1439-1461.

118. Xu, Z. Jin, R. Ye, J., Lyu, M., King, I. 2010. Discriminative semi-supervised feature selection via manifold regularization. *IEEE Transactions on Neural Networks* 21(7), pp. 1033-1047.

119. Xue, X., Zhang, K., Tan, K. C., Feng, L., Wang, J., Chen, Z., Zhao, X., Zhang, L., Yao, J. 2020. Affine Transformation-Enhanced Multifactorial Optimization for Heterogeneous Problems. *IEEE Transactions on Cybernetics*, pp. 1-5. https://doi.org/10.1109/TCYB.2020.3036393.

120. Zhao, F., Zhao, J., Niu, X., Luo, S., Xin, Y. 2018. A Filter Feature Selection Algorithm Based on Mutual Information for Intrusion Detection. *Applied Sciences* 8(9) 1535, pp. 1-20. http://dx.doi.org/10.3390/app8091535

121. Zhao, Z., Liu, H. 2007. Semi-supervised feature selection via spectral analysis. *Proceedings of SIAM International Conference on Data Mining*, pp. 641-646. https://doi.org/10.1137/1.9781611972771.75

122. Zhengbing, H, Zhitang, L., Junqi, W. 2008. A Novel Network Intrusion Detection System (NIDS) Based on Signatures Search and Data Mining. *First International Workshop on Knowledge Discovery and Data Mining (WKDD 2008)*, pp. 10-16.

11. PRILOG 1: VEKTORSKI PROSTOR

Neka su dati: skup Ω i polje (F, +, ·). Skup Ω je vektorski prostor nad poljem F, ukoliko je zatvoren nad operacijom sabiranja (P.1) i množenja (P.2) elementima polja F:

$$\big((\forall \mathbf{x}, \mathbf{y}) \in \Omega\big)\ \mathbf{x} + \mathbf{y} \in \Omega \tag{P.1}$$

$$(\forall \alpha \in F)(\forall \mathbf{x} \in \Omega)\ \alpha\mathbf{x} \in \Omega \tag{P.2}$$

i, ukoliko za sve elemente **x**, **y**, **z** ∈ Ω, kao i za α i $\beta \in F$, važe aksiomi:

$$\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x} \tag{P.3}$$

$$\mathbf{x} + (\mathbf{y} + \mathbf{z}) = (\mathbf{x} + \mathbf{y}) + \mathbf{z} \tag{P.4}$$

$$(\exists \mathbf{0} \in \Omega)\ \mathbf{0} + \mathbf{x} = \mathbf{x} \tag{P.5}$$

$$(\exists(-\mathbf{x}) \in \Omega)\ \mathbf{x} + (-\mathbf{x}) = \mathbf{0} \tag{P.6}$$

$$1 \cdot \mathbf{x} = \mathbf{x} \tag{P.7}$$

$$\alpha(\beta\mathbf{x}) = (\alpha\beta)\mathbf{x} \tag{P.8}$$

$$\alpha(\mathbf{x} + \mathbf{y}) = \alpha\mathbf{x} + \alpha\mathbf{y} \tag{P.9}$$

$$(\alpha + \beta)\mathbf{x} = \alpha\mathbf{x} + \beta\mathbf{x} \tag{P.10}$$

onda se elementi vektorskog prostora nazivaju vektori, a elementi polja *F* skalari i, na osnovu aksioma P.3 - P.6, Ω predstavlja komutativnu ili Abelovu grupu u odnosu na operaciju sabiranja vektora. Aksiomi P.7 – P.10 odnose se na množenje vektora skalarom.

Za dati vektorski prostor Ω i polje *F*, preslikavanje π: Ω x Ω → *F*, za koje važi:

$$\pi(\mathbf{x} + \mathbf{y}, \mathbf{z}) = \pi(\mathbf{x}, \mathbf{z}) + \pi(\mathbf{y}, \mathbf{z}) \tag{P.11}$$

$$\pi(\alpha\mathbf{x}, \mathbf{y}) = \alpha\pi(\mathbf{x}, \mathbf{y}) \tag{P.12}$$

$$\pi(\mathbf{x}, \mathbf{y}) = \pi(\mathbf{y}, \mathbf{x}) \tag{P.13}$$

$$\mathbf{x} \neq \mathbf{0} \Rightarrow \pi(\mathbf{x}, \mathbf{x}) > 0 \tag{P.14}$$

naziva se skalarnim proizvodom vektora., a uobičajeni zapis skalarnog proizvoda ima sledeći oblik:

$$\pi(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y} = |\mathbf{x}||\mathbf{y}| cos\theta \tag{P.15}$$

pri čemu je || oznaka modula vektora, a θ ugao između njih (Bjelica, 2009).

12. PRILOG 2: SOFTVERSKI KODOVI ZA FNN, DT, k-NN, wk-NN I SVM ALGORITME

U ovom prilogu prikazani su softverski kodovi za pet modela mašinskog učenja, razvijeni u programskom paketu MATLAB.

12.1. Feedforward Neural Network

```
function [y1] = myNeuralNetworkFunction(x1)
% Generated by Neural Network Toolbox function genFunction on 06-Jan-2019 21:05:23.
% x = Qx9 matrix, input #1 and returns y = Qx1 matrix; Q is #instances.
% ===== NEURAL NETWORK CONSTANTS =====
% Input 1
x1_step1.xoffset = [-1;-1;-1;-1;-1;-1;-1;-1;-1];
x1_step1.gain = [1;1;1;1;1;1;1;1;1];
x1_step1.ymin = -1;
% Layer 1: bias & weights
b1 = [3.2707821601477183;-1.5940480979567007;1.5016084790279531;-2.2273197730653624;-1.8316675643416864;0.19256636530626425;2.0915891107963347;0.71472187866258219;-2.0673662707963389];
IW1_1 = [4.0280119740447704 0.85990564845071127 0.44822545087211857 0.093395132053410046 -0.48515878280957347 0.24024105502871854 -1.3334450275845258 0.63517124231777511 -0.90789169956379656;0.93867947192072465 0.83432054854242377 -0.017296156910559673 0.48716422110361957 0.26648689813242676 0.0038177278651953246 0.48772602349002647 0.46365475386116256 0.96139734337405935;0.028955792464273222 0.43381502954959461 -0.22125141053891031 -2.6349431204604485 0.23640865414444787 -5.8271930735732269 -0.1476784590854931 -0.74283302421919328 -1.1494378744674032;-3.8777614910107401 0.60681287395249806 3.1629496174268081 0.71610323781929197 1.0932533566739424 -1.6669976313105093 -3.4569073886221675 -0.73614378507643163 0.34188332102270769;-2.1133196854828986 2.3069382191381949 2.3643819468157332 -0.38330830165036711 3.2068257081576488 -3.1699355649657543 -0.26785392902542821 0.076853831058028954 -0.49215879120793421;0.35725235430769647 0.87441506471334263 1.3805679963147734 0.85862397046491989 0.67502310574750257 0.38639037273514121 0.69168868524229321 0.3232362888886478 0.27197037842919392;3.6149707502658446 -0.082865076987196473 -2.2222257838129091 -0.12768784003571285 1.2462899463663557 5.8004682953549347 -0.47377582021740205 -2.6003902559905678 -2.9815992117382057;-0.21552009536495689 0.28308115322538779 -1.2407261181697917 -0.1307156899617111 -0.28617039198778693 3.3788859290607389 0.43602646858862704 -
```

```
0.68071167120768095 0.9598536124188551 4;0.10859943973766251
0.13404923950123182 0.55173576060206675 -0.18528707450038739 -
0.10408074576933689 -0.40785071051537747 -0.4523118448086571
1.9533811849693639 0.32821913350510251];

% Layer 2: bias & weights

b2 = -1.9642326171461448;

LW2_1 = [5.3454059990345506 2.4956505109683822 4.8752909003539999 -
5.4208233855862069 -3.189710010673076 1.32217514435874 4.5482333082007544 -
4.1541700333695992 2.5549963678054723];

% ===== SIMULATION ========

% Dimensions

Q = size(x1,1); #instances

% Input 1

x1 = x1';

xp1 = mapminmax_apply(x1,x1_step1);

% Layer 1

a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*xp1);

% Layer 2

a2 = logsig_apply(repmat(b2,1,Q) + LW2_1*a1);

% Output 1

y1 = a2;

y1 = y1';

end

% Map Minimum and Maximum Input Processing Function

function y = mapminmax_apply(x,settings)

y = bsxfun(@minus,x,settings.xoffset);

y = bsxfun(@times,y,settings.gain);

y = bsxfun(@plus,y,settings.ymin);

end

% Sigmoid Positive Transfer Function

function a = logsig_apply(n,~)

a = 1 ./ (1 + exp(-n));

end

% Sigmoid Symmetric Transfer Function

function a = tansig_apply(n,~)

a = 2 ./ (1 + exp(-2*n)) - 1;

end
```

12.2. Decision Trees

```
function [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)


% Auto-generated by MATLAB Classification Learner on 10-Jan-2019 11:06:26

% Returns a trained classifier and its accuracy. This code recreates the % classification model

% Input:

    % trainingData: the training data of same data type as imported in the app (table or matrix).

% Output:

        % trainedClassifier: a struct containing the trained classifier. The struct contains various % % fields with information about the trained classifier.

        % trainedClassifier.predictFcn: a function to make predictions  on new data. It takes an input % of the same form as this training code (table or matrix) and returns predictions for the % % response. If you supply a matrix, include only the predictors columns (or rows).

        % validationAccuracy: a double containing the accuracy in  percent. In the app, the History % list displays this  overall accuracy score for each model.

% Use the code to train the model with new data. To retrain the classifier, call the function from the % command line with your original data or new  data as the input argument trainingData.

% To make predictions with the returned 'trainedClassifier' on new data T,
% use yfit = trainedClassifier.predictFcn(T)

% To automate training the same classifier with new data, or to learn how to programmatically train % classifiers, examine the generated code.

% Extract predictors and response; the code processes the data into the right shape for training the
% classifier.

% Convert input to table

inputTable = array2table(trainingData, 'VariableNames', {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9', 'column_10'});

predictorNames = {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9'};

predictors = inputTable(:, predictorNames);

response = inputTable.column_10;

isCategoricalPredictor = [false, false, false, false, false, false, false, false, false];

% Train a classifier

classificationTree = fitctree(...

   predictors, ...
```

```
    response, ...
    'SplitCriterion', 'gdi', ...
    'MaxNumSplits', 20, ...
    'Surrogate', 'off', ...
    'ClassNames', [0; 1]);


% Create the result struct with predict function
predictorExtractionFcn = @(x) array2table(x, 'VariableNames', predictorNames);
treePredictFcn = @(x) predict(classificationTree, x);
trainedClassifier.predictFcn = @(x) treePredictFcn(predictorExtractionFcn(x));
% Add additional fields to the result struct
trainedClassifier.ClassificationTree = classificationTree;
trainedClassifier.About = 'This struct is a trained classifier exported from Classification Learner R2016b.';
trainedClassifier.HowToPredict = sprintf('To make predictions on a new predictor column matrix, X, use: \n  yfit = c.predictFcn(X) \nreplacing "c" with the name of the variable that is this struct, e.g. "trainedClassifier". \n \nX must contain exactly 9 columns because this classifier was trained using 9 predictors. \nX must contain only predictor columns in exactly the same order and format as your training \ndata. Do not include the response column or any columns you did not import into \nClassification Learner. \n \nFor more information, see <a href="matlab:helpview(fullfile(docroot, "stats", "stats.map"), "appclassification_exportmodeltoworkspace")">How to predict using an exported model</a>.');
% Extract predictors and response. This code processes the data into the right shape for training the
% classifier.
% Convert input to table
inputTable = array2table(trainingData, 'VariableNames', {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9', 'column_10'});
predictorNames = {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9'};
predictors = inputTable(:, predictorNames);
response = inputTable.column_10;
isCategoricalPredictor = [false, false, false, false, false, false, false, false, false];
% Perform cross-validation
partitionedModel = crossval(trainedClassifier.ClassificationTree, 'KFold', 5);
% Compute validation accuracy
validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError');
```

% Compute validation predictions and scores

[validationPredictions, validationScores] = kfoldPredict(partitionedModel);

12.3. k-Nearest Neighbours

function [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)

% trainClassifier(trainingData) returns a trained classifier and its accuracy. This code recreates the % classification model trained in Classification Learner app.

% Input:

% trainingData: the training data of same data type as imported in the app (table or matrix).

% Output:

% trainedClassifier: a struct containing the trained classifier. The struct contains various % % fields with information about the trained classifier.

% trainedClassifier.predictFcn: a function to make predictions on new data. It takes an input % of the same form as this training code (table or matrix) and returns predictions for the % % response. If you supply a matrix, include only the predictors columns (or rows).

% validationAccuracy: a double containing the accuracy in percent. In the app, the History % list displays this overall accuracy score for each model.

% Use the code to train the model with new data. To retrain your classifier, call the function from % the command line with your original data or new data as the input argument trainingData.

% To automate training the same classifier with new data, or to learn how to programmatically train classifiers, examine the generated code.

% Extract predictors and response: processes the data into the right shape for training the classifier.

% Convert input to table

inputTable = array2table(trainingData, 'VariableNames', {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9', 'column_10'});

predictorNames = {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9'};

predictors = inputTable(:, predictorNames);

response = inputTable.column_10;

isCategoricalPredictor = [false, false, false, false, false, false, false, false, false];

% Train a classifier: code specifies all the classifier options and trains the classifier.

classificationKNN = fitcknn(predictors, response, 'Distance', 'Euclidean', 'Exponent', [], ...

'NumNeighbors', 10, 'DistanceWeight', 'Equal', 'Standardize', true, 'ClassNames', [0; 1]);

% Create the result struct with predict function

predictorExtractionFcn = @(x) array2table(x, 'VariableNames', predictorNames);

knnPredictFcn = @(x) predict(classificationKNN, x);

trainedClassifier.predictFcn = @(x) knnPredictFcn(predictorExtractionFcn(x));

% Add additional fields to the result struct

trainedClassifier.ClassificationKNN = classificationKNN;

trainedClassifier.About = 'This struct is a trained classifier exported from Classification Learner R2016b.';

trainedClassifier.HowToPredict = sprintf('To make predictions on a new predictor column matrix, X, use: \n yfit = c.predictFcn(X) \nreplacing "c" with the name of the variable that is this struct, e.g. "trainedClassifier". \n \nX must contain exactly 9 columns because this classifier was trained using 9 predictors. \nX must contain only predictor columns in exactly the same order and format as your training \ndata. Do not include the response column or any columns you did not import into \nClassification Learner. \n \nFor more information, see How to predict using an exported model.');

% Extract predictors and response% This code processes the data into the right shape for training the classifier.

% Convert input to table

inputTable = array2table(trainingData, 'VariableNames', {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9', 'column_10'});

predictorNames = {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9'};

predictors = inputTable(:, predictorNames);

response = inputTable.column_10;

isCategoricalPredictor = [false, false, false, false, false, false, false, false, false];

% Perform cross-validation

partitionedModel = crossval(trainedClassifier.ClassificationKNN, 'KFold', 5);

% Compute validation accuracy

validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError');

% Compute validation predictions and scores

[validationPredictions, validationScores] = kfoldPredict(partitionedModel);

12.4. weighted k-Nearest Neighbours

```
function [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)
% trainClassifier(trainingData) returns a trained classifier and its accuracy.
%  Input:
%      trainingData: the training data of same data type as imported in the app (table or matrix).
%  Output:
%      trainedClassifier: a struct containing the trained classifier. The struct contains various fields with % information about the trained classifier.
%      trainedClassifier.predictFcn: a function to make predictions on new data. It takes an input of % the same form as this training code (table or matrix) and returns predictions for the % % % response. If you supply a matrix, include only the predictors columns (or rows).
%      validationAccuracy: a double containing the accuracy in  percent.
% Auto-generated by MATLAB on 10-Jan-2019 11:28:04
% Extract predictors and response: processes the data into the right shape for training the classifier.
% Convert input to table
inputTable = array2table(trainingData, 'VariableNames', {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9', 'column_10'});
predictorNames = {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9'};
predictors = inputTable(:, predictorNames);
response = inputTable.column_10;
isCategoricalPredictor = [false, false, false, false, false, false, false, false, false];
% Train a classifier
classificationKNN = fitcknn(predictors, response, 'Distance', 'Euclidean', 'Exponent', [], ...
    'NumNeighbors', 10, 'DistanceWeight', 'SquaredInverse', 'Standardize', true, 'ClassNames', [0; 1]);
% Create the result struct with predict function
predictorExtractionFcn = @(x) array2table(x, 'VariableNames', predictorNames);
knnPredictFcn = @(x) predict(classificationKNN, x);
trainedClassifier.predictFcn = @(x) knnPredictFcn(predictorExtractionFcn(x));
% Add additional fields to the result struct
trainedClassifier.ClassificationKNN = classificationKNN;
trainedClassifier.About = 'This struct is a trained classifier exported from Classification Learner R2016b.';
```

trainedClassifier.HowToPredict = sprintf('To make predictions on a new predictor column matrix, X, use: \n yfit = c.predictFcn(X) \nreplacing "c" with the name of the variable that is this struct, e.g. "trainedClassifier". \n \nX must contain exactly 9 columns because this classifier was trained using 9 predictors. \nX must contain only predictor columns in exactly the same order and format as your training \ndata. Do not include the response column or any columns you did not import into \nClassification Learner. \n \nFor more information, see How to predict using an exported model.');

% Extract predictors and response: processes the data into the right shape for training the classifier.

% Convert input to table

inputTable = array2table(trainingData, 'VariableNames', {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9', 'column_10'});

predictorNames = {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9'};

predictors = inputTable(:, predictorNames);

response = inputTable.column_10;

isCategoricalPredictor = [false, false, false, false, false, false, false, false, false];

% Perform cross-validation

partitionedModel = crossval(trainedClassifier.ClassificationKNN, 'KFold', 5);

% Compute validation accuracy

validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError');

% Compute validation predictions and scores

[validationPredictions, validationScores] = kfoldPredict(partitionedModel);

12.5. Support Vector Machines

function [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)

% trainClassifier(trainingData) returns a trained classifier and its accuracy.

% Input:

% trainingData: the training data of same data type as imported in the app (table or matrix).

% Output:

% trainedClassifier: a struct containing the trained classifier. The struct contains various fields % with information about the trained classifier.

% trainedClassifier.predictFcn: a function to make predictions on new data. It takes an input of % the same form as this training code (table or matrix) and returns predictions for the response. % If you supply a matrix, include only the predictors columns (or rows).

```
%       validationAccuracy: a double containing the accuracy in percent.
% Auto-generated by MATLAB on 10-Jan-2019 11:40:03
% Extract predictors and response processes the data into the right shape for training
% Convert input to table
inputTable = array2table(trainingData, 'VariableNames', {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9', 'column_10'});
 predictorNames = {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9'};
predictors = inputTable(:, predictorNames);
response = inputTable.column_10;
isCategoricalPredictor = [false, false, false, false, false, false, false, false, false];
 % Train a classifier specifies all the classifier options and trains the classifier.
classificationSVM = fitcsvm(predictors, response, 'KernelFunction', 'gaussian', 'PolynomialOrder', ... [], 'KernelScale', 3, 'BoxConstraint', 1, 'Standardize', true, 'ClassNames', [0; 1]);
 % Create the result struct with predict function
predictorExtractionFcn = @(x) array2table(x, 'VariableNames', predictorNames);
svmPredictFcn = @(x) predict(classificationSVM, x);
trainedClassifier.predictFcn = @(x) svmPredictFcn(predictorExtractionFcn(x));
 % Add additional fields to the result struct
trainedClassifier.ClassificationSVM = classificationSVM;
trainedClassifier.About = 'This struct is a trained classifier exported from Classification Learner R2016b.';
trainedClassifier.HowToPredict = sprintf('To make predictions on a new predictor column matrix, X, use: \n  yfit = c.predictFcn(X) \nreplacing "c" with the name of the variable that is this struct, e.g. "trainedClassifier". \n \nX must contain exactly 9 columns because this classifier was trained using 9 predictors. \nX must contain only predictor columns in exactly the same order and format as your training \ndata. Do not include the response column or any columns you did not import into \nClassification Learner. \n \nFor more information, see <a href="matlab:helpview(fullfile(docroot, "stats", "stats.map"), "appclassification_exportmodeltoworkspace")">How to predict using an exported model</a>.');
% Extract predictors and response processes the data into the right shape for training the classifier.
% Convert input to table
inputTable = array2table(trainingData, 'VariableNames', {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9', 'column_10'});
 predictorNames = {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9'};
```

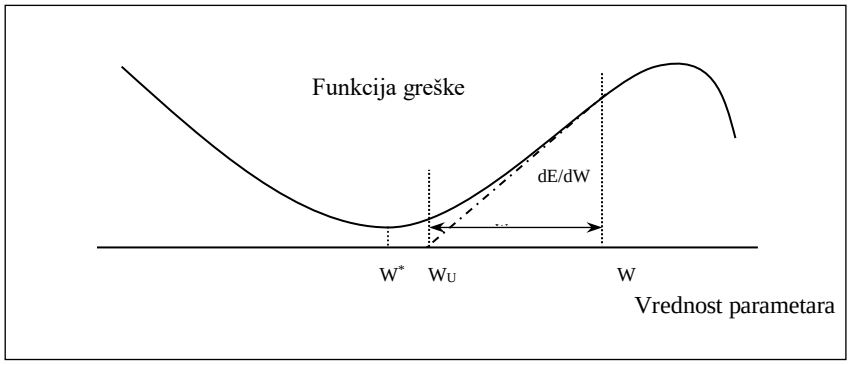
```
predictors = inputTable(:, predictorNames);
response = inputTable.column_10;
isCategoricalPredictor = [false, false, false, false, false, false, false, false, false];
 % Perform cross-validation
partitionedModel = crossval(trainedClassifier.ClassificationSVM, 'KFold', 5);
 % Compute validation accuracy
validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError');
 % Compute validation predictions and scores
[validationPredictions, validationScores] = kfoldPredict(partitionedModel);
```

13. PRILOG 3: GD, GN i LM algoritam

U slučajevima kada je neophodno minimizovati nelinearnu funkciju greške iterativnim putem, ne postoje metode kojima je moguće dostići (tačan) globalni minimum. Ovo je, takođe, karakterisitčno za neuornske mreže, koje koriste optimizaciju parametara umesto iterativnih metoda. Postoje i slučajevi u kojima funkcija greške ima dva globalna minimuma (ukoliko je npr. simetrična) ili da ima nekoliko lokalnih minimuma. Optimizacione tehnike podrazumevaju predobradu podataka, optimizaciju algoritama obučavanja i slično, s ciljem nalaženja najboljeg rešenja za postavljeni problem. U tekstu koji sledi prikazane su gradijenta metoda, Gauss-Njutnova aproksimacija i Levenberg-Marquardt algoritam, za rešavanje datog problema, većom preciznošću i brzinom.

13.1. Gradijentna metoda nalaženja optimalnog rešenja

Gradijentna metoda podrazumeva proračun promena parametara koji utiču na smanjenje funkcije greške, koja treba da bude diferencijabilna po svakom od parametara, tako da je moguće izračunati njen gradijentni vektor. Ukoliko se, tokom optimizacionog procesa, vrši pomeranje po pravcu negativnog gradijenta, moguće je pronaći optimalne vrednosti parametara koji lokalno minimizuju funkciju greške. Pri tome nije poznata veličina koraka kojom se vrši pomeranje po funkciji greške kako se do minimuma ne bi stizalo ili on ne bi bio preskočen. Princip pomeranja po paravcu negativnog gradijenta (gradient descent, steepest descent, engl.) prikazan je Slikom 18.



Slika 18. Promena parametara po principu pada negativnog gradijenta

W^* je parametar koji određuje globalni minimum, i njemu se teži. W je trenutna vrednost parametra, ΔW je trenutna vrednost promene parametra, a W_u je promenjena vrednost parametra.

13.2. Gaus-Njutnova aproksimacija

Jedan metod određivanja optimalne veličine koraka pri promeni funkcije greške bazira se na Gaus-Njutnovom metodu aproksimacije Tejlorovog niza, koji je prikazan sledećim izrazom:

$$E = E_0 + \left(\frac{\partial E}{\partial \boldsymbol{u}}\right)^T \delta \boldsymbol{u} + \frac{1}{2}\delta \boldsymbol{u}^T \frac{\partial^2 E}{\partial \boldsymbol{u}^2}\delta \boldsymbol{u} + \cdots \tag{13.1}$$

Gaus-Njutnova aproksimacija Tejlorovog niza prikazana je sledećim izrazom:

$$E \approx E_0 + \left(\frac{\partial E}{\partial \boldsymbol{u}}\right)^T \delta \boldsymbol{u} + \frac{1}{2}\delta \boldsymbol{u}^T \boldsymbol{H} \delta \boldsymbol{u} \tag{13.2}$$

pri čemu su: E - funkcija greške, E_0 - vrednost funkcije greške u tački aproksimacije, $\mathbf{u}$ – vektor parametara, $\delta\mathbf{u}$ – vektor odstupanja parametara $\mathbf{u}$ a $\mathbf{H}$ – Hessianova matrica (izrazi 13.3 – 13.5).

$$\boldsymbol{u} = [u_1, u_2, \ldots, u_n]^T \tag{13.3}$$

$$\frac{\partial E}{\partial \boldsymbol{u}} = \left[\frac{\partial E}{\partial u_1}, \frac{\partial E}{\partial u_2}, \ldots, \frac{\partial E}{\partial u_n}\right]^T \tag{13.4}$$

$$\mathbf{H} = \frac{\partial^2 E}{\partial \mathbf{u}^2} = \begin{bmatrix} \frac{\partial^2 E}{\partial u_1^2} & \frac{\partial^2 E}{\partial u_1 \partial u_2} & \cdots & \frac{\partial^2 E}{\partial u_1 \partial u_n} \\ \frac{\partial^2 E}{\partial u_2 \partial u_1} & \frac{\partial^2 E}{\partial u_2^2} & \cdots & \frac{\partial^2 E}{\partial u_2 \partial u_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 E}{\partial u_n \partial u_1} & \frac{\partial^2 E}{\partial u_n \partial u_2} & \cdots & \frac{\partial^2 E}{\partial u_n^2} \end{bmatrix} \tag{13.5}$$

pri čemu je n broj parametara. Minimum funkcije greške određen je diferenciranjem po vektoru parametara i izjednačavanjem izvoda sa nulom.

$$\delta \boldsymbol{u} = \boldsymbol{u}^* - \boldsymbol{u} = -\mathbf{H}^{-1}\frac{\partial E}{\partial \boldsymbol{u}} = 0 \tag{13.6}$$

$$\boldsymbol{u}^* = \boldsymbol{u} - \mathbf{H}^{-1}\frac{\partial E}{\partial \boldsymbol{u}} \tag{13.7}$$

Problem ove metode je proračun inverzne Hesianove matrice $\mathbf{H}^{-1}$, jer je broj operacija potreban za inverziju ove matrice ~ n^3 i traje vremenski dugo. Problem je moguće rešiti estimacijom inverzne Hesianove matrice i primenom jednog od pseudo-Njutnovih metoda.

13.3. Levenberg-Marquardt algoritam

Neka je $f: \mathrm{R}^n \to \mathrm{R}$ kontinualna funkcija, i neka je ulazni vector $\mathbf{x} = [x_1\ x_2\ \cdots x_n]^{\mathrm{T}}$. Cilj LM metoda je minimizovati Euklidovo rastojanje dato u obliku $\|f(x)\|^2$. Gradijent $\nabla f(\mathbf{x})$ i Hesijanova matrica $\mathbf{H}\big(f(\mathbf{x})\big)$ su vektor parcijalnih izvoda i matrica parcijalnih izvoda drugog reda funkcije $f(\mathbf{x})$, respektivno, tako da važi:

$$\nabla f(\mathbf{x}) = Df(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}\ \frac{\partial f(\mathbf{x})}{\partial x_2} \cdots \frac{\partial f(\mathbf{x})}{\partial x_n}\right],\ \mathbf{H}\big(f(\mathbf{x})\big) = \nabla^2 f(\mathbf{x}) = D^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1^2} & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_n^2} \end{bmatrix}_{nxn} \tag{13.8}$$

U višedimenzionalnom prostoru za svako $g_i = \frac{\partial f_j(\mathbf{x})}{\partial x_i}$ $(i = 1, \ldots, n\,;\ j = 1, \ldots m)$, gradijentna matrica parcijalnih izvoda iznosi $g_j(\mathbf{x}) = \nabla f_j(\mathbf{x}) = \left[g_{1,j}(\mathbf{x})\ g_{2,j}(\mathbf{x}) \cdots g_{n,j}(\mathbf{x})\ \right]^{\mathrm{T}}$, po svim dimenzijama. Jakobijanova matrica funkcije *f*, $\mathbf{J}\big(\boldsymbol{f}(\boldsymbol{x})\big)$, sadrži sve njene parcijalne prve izvode, odnosno

$$\mathbf{J}\big(f(\boldsymbol{x})\big) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_{2(\mathbf{x})}}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \frac{\partial f_m(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{bmatrix}_{nxm} \tag{13.9}$$

Sada je Hesianovu matricu moguće aproksimirati da važi: $\mathbf{H}\big(f(\mathbf{x})\big) = \mathbf{J}\big(\nabla f(\mathbf{x})\big)^{\mathrm{T}}$.

U rešavanju nelinearnih problema najmanjih kvadrada (non-linear least squares, NLS, engl.) ideja aproksimacije bazirana je na izvođenju afine transformacije funkcije *f* u blizini posmatrane tačke. Aproksimacija Tejlorovog niza funkcije f u tački p data je formulom (13.10).

$$\hat{f}(x; p) = f(p) + Df(p)(x - p) \tag{13.10}$$

Izraz $Df(p)(x - p)$ prestavlja Jakobijanovu matricu parcijalnih izvoda funkcije *f*. Ukoliko je $\hat{f}(x; p) = f(x)$ i x je blizu p, onda $\hat{f}(x; p)$ može da bude minimizovana korišćenjem linearne least-squares (LS) metoda da bi bilo određeno tačno rešenje.

LM je iterativna procedura kod koje je, u trenutnoj iteraciji, problem rešen tako da istovremeno važi $\hat{f}(x; p) = f(x)$ i $x \approx p$. Neka su vrednosti promenljive *x*, u iteracijama, određene kao $x^{(1)}, x^{(2)}, \ldots\ x^{(l)}$. U iteraciji *k*, afina transformacija funkcije *f* u nekoj tački *k* data je sledećim izrazom:

$$\hat{f}\big(x; x^{(k)}\big) = f\big(x^{(k)}\big) + Df\big(x^{(k)}\big)\big(x - x^{(k)}\big) \tag{13.11}$$

Ideja algoritma je izabrati $x^{(k+1)}$ da minimizuje izraz

$$\left\|\hat{f}\big(x; x^{(k)}\big)\right\|^2 + \lambda^{(k)}\left\|x - x^{(k)}\right\|^2,\ \ \lambda^{(k)} > 0 \tag{13.12}$$

Odnosno, cilj je minimizovati i prvi i drugi deo izraza (13.12). Ukoliko je minimizovan izraz $\left\|\hat{f}\big(x; x^{(k)}\big)\right\|^2$aproksimacija može biti razmatrana kao $\hat{f} \approx f$. Drugi cilj je odrediti koliko je $x^{(k)}$ udaljen od x. LM opisuje razmenu između ova dva cilje. Parametar

$\lambda^{(k)}$, poznat kao damping faktor, varira sa veličinom koroaka i određuje nivo razmene. Neka je $x^{(k+1)}$ rešenje LS problema (13.3)

$$min\left\{\left\|f\left(x^{(k)}\right)+Df\left(x^{(k)}\right)\left(x-x^{(k)}\right)\right\|^2+\lambda^{(k)}\left\|x-x^{(k)}\right\|^2\right\},\ \lambda^{(k)}>0 \tag{13.13}$$

tada je:

$$x^{(k+1)}=x^{(k)}-\left(Df\left(x^{(k)}\right)^{\mathrm{T}}Df\left(x^{(k)}\right)+\lambda^{(k)}\mathrm{I}\right)^{-1}Df\left(x^{(k)}\right)^{\mathrm{T}}f\left(x^{(k)}\right),\ \lambda^{(k)}>0 \tag{13.14}$$

Izraz $\left(Df\left(x^{(k)}\right)^{\mathrm{T}}Df\left(x^{(k)}\right)+\lambda^{(k)}\mathrm{I}\right)^{-1}$ predstavlja generalno rešenje koje uvek postoji obzirom da je $\lambda^{(k)}>0$. Tačka $x^{(k)}$ je stacionarna ako važi da je $x^{(k+1)}\approx x^{(k)}$. Optimalno rešenje važi samo za $Df\left(x^{(k)}\right)^{\mathrm{T}}f\left(x^{(k)}\right)=0$. U toku iterativne procedure $\lambda^{(k)}$ se podešava na sledeći način: ukoliko je $\lambda^{(k)}$ preveliko, onda je $x^{(k+1)}$ preblizu $x^{(k)}$, i proces je spor.U suprotnom slučaju $x^{(k+1)}$ je udaljen od $x^{(k)}$ i aproksimacija je loša. Mehanizam promene damping faktora određen je sledećim pseudokodom:

→ ukoliko važi da je $\left\|f\left(x^{(k+1)}\right)\right\|^2<\left\|f\left(x^{(k)}\right)\right\|^2$, $f\left(x^{(k+1)}\right)$ ima prihvatljiviju vrednost od trenutne, prihvati novu vrednost $x^{(k+1)}$ i redukuj $\lambda^{(k)}$,
→ u suprotnom povećaj $\lambda^{(k)}$ ali ne menjaj vrednost $x^{(k)}$; $x^{(k+1)}=x^{(k)}$.

Neka je $x^{(k+1)}$ rešenje gradijenta funkcije cilja (GD algoritam), tada važi:

$$x^{(k+1)}=x^{(k)}-\lambda^{(k)}Df\left(x^{(k)}\right),\ \lambda^{(k)}>0 \tag{13.15}$$

rešenje $x^{(k+1)}$ po GN algoritmu, u bilzini optimalne vrednosti dato je izrazom

$$x^{(k+1)}=x^{(k)}-\left(D^2f\left(x^{(k)}\right)\right)^{-1}Df\left(x^{(k)}\right)^{\mathrm{T}} \tag{13.16}$$

LM algoritam počinje iz inicijalne tačke $x^{(0)}$ i sa inicijalnim faktorom $\lambda^{(0)}$. Nakon toga se $x^{(k+1)}$ određuje po formuli

$$x^{(k+1)}=x^{(k)}-\left(\mathbf{H}+\lambda^{(k)}\mathbf{I}\right)^{-1}\mathbf{J}^{\mathrm{T}}f\left(x^{(k)}\right)=x^{(k)}-\left(\mathbf{J}^{\mathrm{T}}\mathbf{J}+\lambda^{(k)}\mathbf{I}\right)^{-1}\mathbf{J}^{\mathrm{T}}f\left(x^{(k)}\right) \tag{13.17}$$

pri čemu je $\mathbf{I}$ jedinična matrica a $\mathbf{H}=\mathbf{J}^{\mathrm{T}}\mathbf{J}$. Ukoliko $\lambda^{(k)}\to\infty$, onda je $\mathbf{H}+\lambda^{(k)}\mathbf{I}\approx\mathbf{I}$ i LM algoritam se ponaša kao GD algoritam. Ukoliko $\lambda^{(k)}\to 0$ onda se LM ponaša kao GN algoritam jer je vrednost $x^{(k)}$ približno jednaka optimalnoj vrednosti.

BIOGRAFIJA

Opšti podaci

Danijela Protić rođena je 1970. godine u Varaždinu. IV beogradsku gimnaziju završila je 1988. godine. Osnovne akademske studije završila je 1995. godine na Fakultetu tehničkih nauka Univerziteta u Novom Sadu, na odseku Elektrotehničke struke i računarstva, smer Elektronika i telekomunikacije, i stekla zvanje diplomiranog inženjera elektrotehnike. Magistarske studije završila je 2001. godine na Elektrotehničkom fakultetu Univerziteta u Beogradu i, odbranom magistarske teze pod nazivom „Analiza neuronskih modela vokala srpskog jezika“, stekla akademsko zvanje Magistar elektrotehničkih nauka – oblast telekomunikacije. Izbor u zvanje Stručni savetnik stekla je na osnovu odluke Naučnog veća Tehničkog opitnog centra, 2021. godine.

Više od dve decenije radi u Centru za primenjenu matematiku i elektroniku u Beogradu. U svom naučno – istraživačkom radu objavila je preko 30 naučnih i stručnih radova u časopisima internacionalnog i nacionalnog značaja. Učestvovala je na dva naučna projekta Matematičkog instituta SANU i na jednom projektu CPME. Prvi je autor na listi Vojnotehničkog glasnika po broju radova, čitanosti i citiranosti, a njeni radovi vidljivi su u bazama kao što su: MDPI, Web of Science, IEEEXplore, ResearchGate, Google Schoolar, Academia i dr. Član je IEEE asocijacije od 2022. godine.

Objavljeni naučni radovi

[1] Protic, D., Gaur, L., Stankovic, M., Rahman, M.A. Cybersecurity in smart cities: Detection of opposing decisions of anomalies in the computer network behavior. *Electronics*, Vol. 11, 3718, 2022. https://doi.org/10.3390/electronics11223718 (M22)

[2] Protic, D., Stankovic, M. Risk management: Machine learning approach to the computer network security. 19th International Conference “Man and Working Environment” Occupational and Environmental Safety, Engineering and Management – OESEM. November 24-27, 2022, Niš, Serbia. [Accepted for publication] (M33)

[3] Grdovic, M., Protic, D., Antic, V., Jovanovic, B. Screen reading: Electromagnetic information leakage from the computer monitor. *Vojnotehnički glasnik/Military Technical Courier*, Vol. 70, Issue 4, 2022, pp. 836-855. [Online] https://doi.org/10.5937/vojtehg70-38930 (M51)

[4] Protic, D., Stankovic, M., Antic, V. WK-FNN Design for Detection of Anomalies in the Computer Network Traffic. *Facta Universitatis, Series: Electronics and Energetics*, Vol. 35, No. 2, 2022, pp. 269-282. ISSN 0353-3670 (Print), ISSN 2217-5997. https://doi.org/10.2298/FUEE2202269P (M24)

[5] Protić, D. Stanković, M. Detection of anomalies in the computer networks: Feature scaling methodology. 1st Serbian International Conference on Applied Artificial Intelligence (SICAAI), Kragujevac, Serbia, May 19-20, 2022. (M33)

[6] Protić, D. Stanković, M., Antić, V. Anomaly-Based Intrusion Detection Systems in Computer Networks: Feedforward Neural Networks and Nearest Neighbour Models as Binary Classifiers. *Proceedings of Conference on Computational Intelligence for Engineering and Management Applications – CIEMA 2022*, Proceedings by Springer LNEE Series (Indexed in Scopus), March 26-27, 2022. (M33)

[7] Protić, D., Stanković, M. The q-Levenberg-Marquardt Method for Unconstrained Nonlinear Optimization, 2021, pp.1-5. [preprint] http://arxiv.org/abs/2107.03304

[8] Protić, D., Stanković, M. The q-Gauss-Newton Method for Unconstrained Nonlinear Optimization, 2021, pp. 1-18. [preprint] http://arxiv.org/abs/2105.12994
[9] Protić, D. Intrusion Detection Based on the Artificial Immune System. *Vojnotehnički glasnik/Military Technical Courier*, Vol. 68, Issue 4, 2021, pp. 790-803. ISSN 0042-8469. e-ISSN 2217-4753. https://doi.org/10.5937/vojtehg68-27954 (M53)
[10] Protić, D., Stanković, M. A Hybrid Model for Anomaly-Based Intrusion Detection in Complex Computer Networks. *21st IEEE International Arab Conference on Information Technology* (ACIT), 6th of October 2020, Giza, Egypt, pp. 1-8. (M33)
[11] Protić, D. Influence of Pre-processing on Anomaly Based Intrusion Detection. *Vojnotehnički glasnik/Military Technical Courier*, Vol. 68, Issue 3, 2020, pp. 598-611. ISSN 0042-8469. e-ISSN 2217-4753. https://doi.org/10.593/vojtehg68-27319 (M53)
[12] Protić, D., Stanković, M. Detection of Anomalies in the Computer Network Behaviour. 5th International Conference on Engineering and Formal Science. 24-25 January 2020, Brussels. *European Journal of Engineering and Formal Sciences*, Vol. 4, Issue 1, 2020, pp. 7-13. ISBN 978-164786089-9 https://doi.org/10.26417/ejef.v4i1.p7-13 (M53)
[13] Protić, D., Stanković, M. Anomaly-Based Intrusion Detection: Feature Selection and Normalization Influence to the Machine Learning Models Accuracy. *Proceedings of 4th International Conferemce on Engineering and Formal Science*, Amsterdam, Netherlands, 14-15 December 2018, pp. 46-51. ISBN 978-88-909700-4-7 https://doi.org/10.26417/ejef.v2i3.p101-106 (M33)
[14] Protić, D. Review of KDD, NSL-KDD and Kyoto 2006+ Datasets, *Vojnotehnički glasnik/Military Technical Courier*, Vol. 66, No. 3, 2018, pp. 580-596. ISSN 0042-8469. e-ISSN 2217-4753. https://doi.org/10.5937/vojtehg-66-16670 (M53)
[15] Protić, D. Critical Infrastructures: Threats, Vulnerabilities and Protection, *Vojnotehnički glasnik/Military Technical Courier*, Vol. 64, No. 3., 2016, pp. 812-832. ISSN 0042-8469. e-ISSN 2217-4753. https://doi.org/10.5937/vojtehg64-9986 (M52)
[16] Protić, D. *Gender and the Information Security Profession. Gender equality in defense system. Accomplishments and Trends. Thematic Collection of Articles*. Ministry of Defense of the Republic of Serbia, 2016, pp. 214-228. (M45)
[17] Protić, D. Neural Cryptography. *Vojnotehnički glasnik/Military Technical Courier*, Vol. 64, No. 2, 2016, pp. 483-495. ISSN 0042-8469. e-ISSN 2217-4753. https://doi.org/10.5937/vojtehg64-8877 (M52)
[18] Protić, D. Feedforward neural networks: The Levenberg-Marquardt optimization and the optimal brain surgeon prunning. *Vojnotehnički glasnik/Military Technical Courier*, Vol. 63, No. 3, 2015, pp.11-28. ISSN 0042-8469. e-ISSN 2217-4753. https://doi.org/10.5937/vojtehg63-7529 (M52)
[19] Protić, D. The impact of glottal signal to the prediction of speech. *Vojnotehnički glasnik/Military Technical Courier*, Vol. 63, No. 1, 2015, pp. 9-31. https://doi.org/10.5937/vojtehg63-6357 (M52)
[20] Protić, D. A Comparative Analysis of Serbian Phonemes: Linear and Nonlinear Models. *Vojnotehnički glasnik/Military Technical Courier*, Vol. 62, No. 4, 2014, pp. 7-37. ISSN 0042-8469. e-ISSN 2217-4753. https://doi.org/10.5937/vojtehg62-5170 (M52)
[21] Protić, D. AES i ARM procesori. *Vojnotehnički glasnik/Military Technical Courier*, Vol. 61, No. 4, 2013, pp. 180-197. ISSN 0042-8469. e-ISSN 2217-4753. https://doi.org/10.5937/vojtehg61-3256 (M52)
[22] Protić, D. Rodna ravnopravnost u IT službama. *Vojno delo*, Vol. 65, No. 4, 2013, pp.170-181. ISSN 0042-8426 (M51)

[23] Protić, D. Informaciona bezbednost: Standardi ili pravila. *Vojno delo*, Vol. 65, No. 1, 2013, pp. 113-150. ISSN 0042-8426 (M51)
[24] Protić, D. Strategija razvoja informacionog društva u Republici Srbiji do 2020. godine: Bezbednost i kritična infrastruktura, *Vojnotehnički glasnik/Military Technical Courier*, Vol. LX, No. 4, 2012, pp 82-101. ISSN 0042-8469. e-ISSN 2217-4753. https://doi-org/10.5937/vojtehg1204082P (M53)
[25] Protić, D., Antić, V. Smanjenje uticaja greške u krizi: modelovanje zasnovano na agentima. *Vojnotehnički glasnik/Military Technical Courier*, Vol. 60, No. 1, 2012, pp. 99-114. ISSN 0042-8469. e-ISSN 2217-4753. https://doi-org/10.5937/vojtehg1201099P (M53)
[26] Protić, D. The Emotional Influence in stock. *Communications in Dependability and Quality Management*, Vol. 11, No. 4, 2008, pp. 70–78. ISSN 1450-7196 (M63)
[27] Protić, D., Milosavljević M. NNARX Model of Speech Signal Generating System: Test Error Subject to Modeling Mode Selection. In *Proceedings of 25th International Conference on Microelectronics (MIEL)*, 14-17 May 2006, pp. 685–688. Available through IEEE Xplore under title: *Microelectronics, 2006 25th International Conference on*. ISBN 1-4244-0116-X (M33)
[28] Protić, D., Milosavljević M. NNARX model sistema za proizvođjenje govora – Greška kao rezultat izbora uslova modelovanja. *Zbornik radova XIII Telekomunikacionog foruma TELFOR*, 22-24 novembar, 2005, Beograd, Akademska misao. (M33)
[29] Protić, D., Milosavljević M. Modeli sistema za proizvođenje govora – Uticaj osnovne učestanosti na procenu parametara modela. *Zbornik radova XIII Telekomunikacionog foruma TELFOR*, 22-24 novembar, 2005, Beograd, Akademska misao. (M33)
[30] Protić, D., Milosavljević M. Pruning u procesiranju govornog signala. *Zbornik radova XIII Telekomunikacionog foruma TELFOR*, 22-24 novembar, 2005, Beograd, Akademska misao. (M33)
[31] Protić, D., Milosavljević, M. Generalizaciona svojstva različitih klasa linearnih i nelinearnih modela govornog signala. *Festival informatičkih dostignuća INFOFEST*, *Festivalski katalog*, Budva, 2005, pp. 248–257. (M33)
[32] Protić, D., Milosavljević, M. 2005. Poboljšanje predikcije govora korišćenjem elektroglotogramskog signala: Uporedna analiza. *Zbornik radova XLIX Konferencije ETRAN*, Vol. 3, 5-10 jun, 2005, Budva, pp. 226– 229. ISBN (Print) 86-80509-53-1 (M33)