

Otkrivanje anomalija mrežnog prometa metodom klasifikacije strojnim učenjem u hibridnoj programski definiranoj mreži

Fosić, Igor

Doctoral thesis / Doktorski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: <https://urn.nsk.hr/urn:nbn:hr:200:290107>

Rights / Prava: [In copyright](#)/[Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2025-03-14**

Repository / Repozitorij:

[Faculty of Electrical Engineering, Computer Science and Information Technology Osijek](#)



Sveučilište J. J. Strossmayera u Osijeku
Fakultet elektrotehnike, računarstva i
informatičkih tehnologija

Igor Fosić

Otkrivanje anomalija mrežnog prometa
metodom klasifikacije strojnim učenjem u
hibridnoj programski definiranoj mreži

Doktorska disertacija

Osijek, 2024.

Doktorski rad izrađen je na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija
Osijek

Mentor: prof. dr. sc. Drago Žagar

Doktorski rad ima 183 stranice

Doktorski rad br.:99

Mojoj obitelji...

SADRŽAJ

1. UVOD.....	1
1.1. Struktura rada	5
2. IZAZOVI KIBERNETIČKE SIGURNOSTI U SDN-MREŽNOM OKRUŽENJU	8
2.1. Tradicionalne mreže.....	8
2.2. Programski definirana mreža	8
2.3. Hibridna programski definirana mreža	12
2.4. Pregled istraživanja i rješenja kibernetičke sigurnosti u SDN-mrežnoj arhitekturi	14
3. STROJNO UČENJE I PRIMJENA ZA OTKRIVANJE UPADA U SDN-MREŽNOM OKRUŽENJU	20
3.1. Strojno učenje.....	21
3.1.1. Kategorije strojnog učenja	23
3.1.2. Nadzirano strojno učenje	24
3.1.3. Klasifikacija u strojnom učenju	26
3.1.4. Klasifikator stabla odluke	27
3.1.5. Klasifikator slučajne šume.....	28
3.1.6. Klasifikator stroja s potpornim vektorima	30
3.1.7. Naivni Bayesov klasifikator.....	31
3.1.8. Klasifikator K-najbližeg susjeda.....	33
3.2. Mjere uspješnosti klasifikacije	34
3.2.1. Matrica konfuzije	35
3.2.2. Mjera točnosti i pogreške klasifikacije	36
3.2.3. Uravnotežena točnost.....	36
3.2.4. Mjera preciznosti	37
3.2.5. Mjera opoziva	37
3.2.6. F-mjera i G-mjera	38
3.2.7. Područje ispod krivulje (AUC).....	39
3.2.8. Matthewsov koeficijent korelacije	41
3.2.9. Cohen kappa-koeficijent	42
3.3. Pregled primjene strojnog učenja u SDN-mrežama.....	42
4. PRIJEDLOG RJEŠENJA ZA OTKRIVANJE ANOMALIJA MREŽNOG PROMETA.....	48

4.1.	Odabir i prilagodba ulaznih podataka modela za otkrivanje anomalija mrežnog prometa	50
4.1.1.	Odabir i usporedba javno dostupnih referentnih skupova podataka za otkrivanje anomalija	52
4.1.1.1.	Cjeloviti skupovi podataka	53
4.1.1.2.	NetFlow-skupovi podataka.....	56
4.1.2.	Predobrada referentnih skupova podataka	58
4.1.2.1.	Brisanje suvišnih značajki	59
4.1.2.2.	Čišćenje neispravnih vrijednosti značajki	61
4.1.2.3.	Kodiranje kategorijskih značajki	63
4.1.2.4.	Usporedba kodiranja kategorijskih značajki.....	65
4.1.2.5.	Razdvajanje podataka za učenje i testiranje	66
4.1.2.6.	Skaliranje značajki.....	72
4.1.2.7.	Utjecaj algoritma skaliranja na točnost klasifikacije.....	74
4.1.3.	Odabir referentnog skupa podataka i klasifikatora	77
4.1.4.	Odabir hiperparametara klasifikatora	83
4.1.4.1.	Unakrsna validacija	83
4.1.4.2.	Rezultati odabira hiperparametara klasifikatora.....	86
4.1.5.	Prilagodba odabranog referentnog skupa podataka NetFlow-strukturi	91
4.1.5.1.	Smanjivanje broja značajki i prilagodba NetFlow-strukturi.....	95
4.1.5.2.	Usporedba i primjena algoritama smanjivanja broja značajki.....	98
4.2.	Prijedlog modela za otkrivanje anomalija mrežnog prometa primjenom strojnog učenja u hibridnoj programski definiranoj mreži	100
4.2.1.	Klasifikacija kombiniranog skupa podataka.....	101
4.2.2.	Rezultati otkrivanja anomalija u stvarnom vremenu	109
5.	PRIJEDLOG METODE VERIFIKACIJE MODELA ZA OTKRIVANJE ANOMALIJA U HIBRIDNOJ PROGRAMSKI DEFINIRANOJ MREŽI.....	117
5.1.	Pregled postojećih metoda verifikacije	119
5.2.	Prijedlog metode verifikacije modela za otkrivanje anomalija višestrukim NetFlow-zapisima mrežnog prometa.....	124
5.3.	Verifikacija predloženog modela za otkrivanje anomalija mrežnog prometa predloženom metodom višestrukih Netflow-zapisa.....	128
5.3.1.	Definiranje istraživačkog problema.....	128
5.3.2.	Određivanje svrhe istraživanja.....	129

5.3.3.	Definiranje konceptualnog modela	129
5.3.4.	Specifikacija ulaznih podataka	132
5.3.5.	Odabir alata za implementaciju simulacije	133
5.3.6.	Razvoj simulacijskog modela	133
5.3.7.	Provedba verifikacijskog eksperimenta	135
5.3.8.	Usporedba rezultata	138
5.3.9.	Verifikacija predloženog IDS/IPS-modela	140
5.3.10.	Dokumentiranje.....	141
6.	ZAKLJUČAK.....	143
	Literatura.....	147
	Popis slika	163
	Popis tablica	165
	Sažetak	167
	Abstract	168
	Životopis	169
	Prilozi.....	170
	Prilog A	170
	Prilog B	171
	Prilog C	172
	Prilog D	173
	Prilog E.....	176
	Prilog F	181

1. UVOD

U posljednjih nekoliko godina, eksplozivni rast korištenja interneta nametnuo je brzu evoluciju mrežnih sustava, kao što su računarstvo u oblaku i mrežna virtualizacija te značajan razvoj komunikacijskih tehnologija, uključujući Internet-stvari i pametne gradove. Sve te promjene dovele su do iznimno velikog porasta podatkovnog prometa kroz kompleksne mreže.

Jedan od ključnih elemenata za precizno upravljanje mrežnim sustavima identifikacija je mrežnog prometa. To omogućuje klasifikaciju tokova mrežnog prometa i pruža osnovu za njihovo učinkovito upravljanje. Postoji nekoliko široko korištenih pristupa za otkrivanje i predviđanje mrežnog prometa, kao što je pristup temeljen na ulaznom sučelju, duboka inspekcija paketa, proučavanje statističkih karakteristika toka mrežnog prometa i primjena strojnog učenja (engl. *machine learning – ML*) [1].

U današnjem suvremenom sigurnosnom okruženju, elementi kibernetičke sigurnosti postali su neizostavni te predstavljaju ključni dio svake poslovne i procesne mrežne strukture. Stalno se susrećemo s izvješćima i vijestima o kibernetičkim napadima, što povećava pritisak za implementacijom sigurnosnih rješenja. Zaštita informacija ima dugu povijest koja datira još od vremena kada su se fizički dokumenti čuvali pod ključem. S razvojem poslovnog svijeta i upotrebom računala, mrežna sigurnost postala je ključna za zaštitu elektroničke infrastrukture. Pojava interneta donijela je revoluciju, otvarajući nezamislive tehnološke mogućnosti, ali istovremeno stvarajući nove ranjivosti i potičući razvoj kibernetičke sigurnosti kao nove industrije. Evoluirajuća priroda sigurnosnih rizika predstavlja jedan od ključnih elemenata kibernetičke sigurnosti. S pojavom novih tehnologija i njihovim raznovrsnim načinima korištenja, neprestano se razvijaju i novi načini kibernetičkih napada. Praćenje tih čestih promjena i napretka u napadima te stalno ažuriranje metoda zaštite, predstavlja veliki izazov za sigurnosna rješenja. U rješavanju sve većeg broja i sofisticiranosti kibernetičkih prijetnji, automatizacija ima važnu ulogu. Korištenje umjetne inteligencije i strojnog učenja u mrežama s velikim protokom podataka mogu znatno poboljšati kibernetičku sigurnost. Ti napredni alati omogućuju brzo i precizno otkrivanje potencijalnih prijetnji, kao i pravovremenu reakciju i obranu od njih.

Razvojem mrežnih sustava, istovremeno su porasle i prijetnje mrežnoj sigurnosti. Kibernetički incidenti, kao što su napadi uskraćivanjem mrežnih usluga (engl. *Denial of Service - DoS*), otkrivanje korisničkih podataka ili zlonamjerni upadi u IT-sustav (engl. *Information Technology*), predstavljaju samo neke od primjera koji mogu izazvati ozbiljne poremećaje u IT-sustavima. Za osiguranje sigurnosti mreže, često se koriste antivirusni softver, vatrozidi i sustavi otkrivanja (engl. *Intrusion Detection System – IDS, Network Intrusion Detection System - NIDS*) i prevencije upada u mrežnu komunikaciju (engl. *Network Intrusion Prevention System - NIPS*), koji kontinuirano nadziru mrežni promet kako bi otkrili sumnjive i zlonamjerne aktivnosti [2]. Pouzdanost, točnost i brzina otkrivanja čimbenici su uspjeha sustava otkrivanja koji teži poboljšanjima u smanjenju lažnih alarma i povećanju točnosti otkrivanja. Kako bi se smanjila stopa lažnih alarma i povećala točnost otkrivanja prijetnji, koriste se različiti pristupi strojnog učenja [3].

U današnjem svijetu nemoguće je učinkovito implementirati tehnologiju kibernetičke sigurnosti bez integracije strojnog učenja. Istovremeno, strojno učenje zahtijeva sveobuhvatan pristup sirovim i obrađenim podacima kako bi bilo učinkovito. Ukratko, strojno učenje ima potencijal da pojednostavi, unaprijedi, smanji troškove i učini kibernetičku sigurnost učinkovitijom, ali to je moguće samo ako podatci korišteni u strojnom učenju pružaju cjelovitu sliku mrežnog okruženja.

Inteligentni sustavi u mrežnom okruženju zahtijevaju odgovarajuću podršku na mrežnoj razini. Tradicionalne mreže s distribuiranim upravljanjem otežavaju primjenu tehnika strojnog učenja. No, programski definirano, umrežavanje (engl. *Software Defined Network - SDN*) donosi novi pristup pružanju kibernetičke sigurnosti. Mogućnosti SDN-a, kao što su logički centralizirana kontrola, globalni pregled mreže, softverska analiza prometa i dinamičko ažuriranje pravila, olakšavaju primjenu tehnike strojnog učenja u ovom kontekstu.

Pored toga, tradicionalne računalne mreže obično se sastoje od uređaja za prosljeđivanje i filtriranje paketa, kao što su preklopnici, usmjerivači i vatrozidi. Svaki od tih uređaja ima ulogu u upravljanju širokim spektrom mrežnih protokola i zadržava lokalni pregled cijele mreže. Ta heterogenost u infrastrukturi i kompleksnost upravljanja mrežom čine sigurnosnu mrežnu politiku i optimizaciju performansi još složenijim zadatkom. Taj složeni skup uređaja i različitih protokola otežava održavanje i razvoj tradicionalnih IP-mreža. Svaki uređaj za prosljeđivanje ima vlastitu konfiguraciju i pravila koja se moraju pratiti kako bi se osigurala ispravna funkcionalnost i

sigurnost mreže. Postizanje dosljedne sigurnosne politike na svakom uređaju zahtijeva pažljivo upravljanje i koordinaciju. Sve to može dovesti do povećane ranjivosti na sigurnosne prijetnje i mogućih problema u performansama.

Paradigma programski definirane mreže (SDN) predstavlja inovativan pristup redizajniranju mrežne arhitekture. U ovom pristupu, upravljačka razina odvaja se od podatkovne razine, što omogućuje bolje upravljanje i kontrolu nad mrežom. Logički centralizirani kontroler igra ključnu ulogu u SDN-u jer održava globalni pogled na cijelu mrežu. SDN omogućuje da se upravljački softver odvoji od tradicionalnih mrežnih uređaja poput preklopnika i usmjerivača. Time se stvara fleksibilniji i dinamičniji način upravljanja mrežom. Upravljačka razina, koju kontrolira logički centralizirani kontroler, može donositi odluke o prosljeđivanju paketa na temelju globalnog pregleda cijele mreže. To omogućuje brže i učinkovitije donošenje odluka, što dovodi do poboljšanja performansi i sigurnosti mreže [2]. Takav pristup pruža iznimnu priliku za primjenu tehnika strojnog učenja.

U sklopu provedenog istraživanja, maksimalno su iskorištene prednosti SDN-arhitekture kako bi se razvio inovativan hibridni model. Taj model učinkovito koristi snagu strojnog učenja za otkrivanje anomalija u mrežnom prometu. U okviru programski definirane mreže (SDN), otvara se prostor za razvoj novih mrežnih aplikacija koje omogućavaju kvalitetan nadzor i učinkovito upravljanje mrežom. Integriranjem pristupa strojnog učenja uz postojeće sigurnosne funkcije kontrolera SDN-a, mogu se postići poboljšanja u sigurnosti mreže i nadzoru prometa. NIDS temeljen na strojnom učenju koristi naučene značajke iz mrežnog prometa kako bi identificirao potencijalne prijetnje i napade. Taj pristup još nije dovoljno istražen, posebno u kontekstu razvoja NIDS-sustava temeljenih na SDN-u i tehnikama strojnog učenja. Kombiniranje SDN-arhitekture s tehnologijama strojnog učenja može otvoriti nove mogućnosti za razvoj naprednih i inteligentnih sigurnosnih rješenja koja se mogu prilagoditi promjenjivim mrežnim uvjetima i prijetnjama, unaprjeđujući sigurnost mrežnih sustava.

Pristup primjene tehnika strojnog učenja na hibridnom SDN-sustavu, koji obuhvaća elemente kako tradicionalne tako i programski definirane mrežne arhitekture, predstavlja dodatni izazov. Integracija tradicionalnih i SDN-mrežnih uređaja zahtijeva inovativne i sofisticirane tehnike kako bi se osigurala sinergija među obama pristupima. Potencijalne prednosti predloženog modela, kao

što je bolja skalabilnost, učinkovitost i sigurnost mreže, osigurale su visoku točnost predviđanja različitih vrsta anomalija u stvarnom vremenu.

Zadatak je ovog istraživanja predložiti rješenje za hibridnu SDN-mrežu koja omogućuje:

- učinkovito otkrivanje anomalija mrežnog prometa uz pomoć strojnog učenja
- postizanje visoke točnosti predviđanja anomalija u smislu binarne klasifikacije mrežnog prometa u stvarnom vremenu
- zaštitu kritičnog dijela IT-sustava u hibridnoj SDN-mreži.

U disertaciji se predlaže nov i inovativan pristup kibernetičkoj sigurnosti, usmjeren na stvarne sigurnosne zahtjeve s jednostavnim, sistemskim, strukturiranim i potencijalno automatiziranim procesom kako bi se postiglo sveobuhvatno rješenje sigurnosti IT-sustava. Primjenjuju se tehnike strojnog učenja za identifikaciju i odgovor na zlonamjerni promet. Sustav se sastoji od dvaju ključnih dijelova: klasifikatora temeljenog na strojnom učenju i hibridne programski definirane mreže. Klasifikator se obučava razlikovati normalni i zlonamjerni promet koristeći referentni i stvarni skup podataka za oba tipa prometa. Nakon toga, uz prednosti hibridne SDN-mreže, sustav reagira na rezultate klasifikatora zaustavljanjem zlonamjernog prometa.

Predloženi model ima nekoliko prednosti u odnosu na postojeće sustave za otkrivanje upada na mreži jer je prilagodljiv za implementaciju i u tradicionalnom i u SDN-mrežnom okruženju, no postiže najveće prednosti i optimalne performanse kada se koristi u hibridnom SDN-okruženju, što je detaljno opisano u disertaciji. Osim toga, karakterizira ga brz odziv u stvarnom vremenu, čineći ga iznimno prikladnim za kritične sustave. Sposobnost trenutnog odgovora bitna je za pouzdan rad u situacijama gdje je nužno osigurati trenutne reakcije i visoku razinu pouzdanosti.

Uz postizanje visoke točnosti otkrivanja anomalija, predloženi se model ističe sposobnošću otkrivanja nepravilnosti u mrežnom prometu, koristeći smanjen broj ulaznih značajki. Njegova učinkovitost u preciznom identificiranju neželjenog prometa predstavlja značajan napredak, posebno s obzirom na smanjenje potrebnih ulaznih informacija.

Dodatno, predloženi model pokazuje visoku prilagodljivost s obzirom na korištenje različitih algoritama klasifikacije u području strojnog učenja, a sposobnost integracije s raznovrsnim algoritmima omogućuje prilagodbu modela specifičnim zahtjevima i karakteristikama ulaznih

podataka. Ta prilagodljivost pruža široki raspon mogućnosti za optimizaciju performansi i postizanje najboljih rezultata u različitim scenarijima primjene.

Model je postavljen u pasivni položaj u mrežnom okruženju, unatoč tomu što njegova funkcionalnost ostaje aktivna zahvaljujući iskorištenim mogućnostima SDN-a. Takav pristup omogućuje smanjenje potrebe za kopiranjem mrežnog prometa radi analize i otkrivanja anomalija. Istovremeno, izbjegava se unos dodatnog kašnjenja koje je često prisutno u sustavima i uređajima koji se nalaze u aktivnom mrežnom položaju. Na taj se način poboljšava ukupna učinkovitost sustava, dok se istodobno umanjuju potencijalni negativni učinci na brzinu i latenciju mrežnog prometa, što je posebno važno u dinamičnim i zahtjevnim mrežnim okolinama.

1.1. Struktura rada

Disertacija je organizirana na sljedeći način:

Poglavlje 2. istražuje izazove kibernetičke sigurnosti u kontekstu programski definiranih mreža (SDN) te prikazuje utjecaj prelaska sa statičnih sigurnosnih mehanizama, poput vatrozida i sustava za otkrivanje upada, na napredne strategije u SDN-okruženju. Dodatno se naglašava važnost komunikacije među slojevima s API-em (engl. *Application Programming Interface*) te ističe prednosti programabilnosti, agilnosti i centralnog upravljanja koje SDN pruža. Proučavaju se i hibridne SDN-mreže koje primjenjuju tradicionalne i SDN-uređaje kako bi pružile fleksibilnost i olakšale postupni prijelaz prema potpunoj SDN-arhitekturi. Opisuje kibernetičke napade u SDN-okruženju, s osvrtom na integraciju strojnog učenja s SDN-Fima kako bi se optimizirali sustavi za otkrivanje prijetnji. Razmatraju se različite tehnike, uključujući sustave za otkrivanje i suzbijanje prijetnji, otkrivanja DDoS-napada, upotrebu mamaca, IDSaaS-a (engl. *Intrusion Detection System as a Service*), tehnologije otkrivanja upada u IoT-u te okviri za unapređenje sigurnosti u podatkovnim centrima temeljenim na SDN-u.

U 3. poglavlju obrađen je pregled sustava za otkrivanje upada (IDS) temeljenih na strojnom učenju, s naglaskom na primjenu u programski definiranim mrežama (SDN). Pregled literature naglašava učestalo zanimanje za integraciju SDN-a i strojnog učenja radi očuvanja sigurnosti mreža. Prikazani su i objašnjeni tipični zadatci i podjela strojnog učenja te se detaljno bavi

nadziranim strojnim učenjem, analizirajući proces klasifikacije. Opisuje klasične klasifikatore poput stabla odluke, slučajne šume, stroja s potpornim vektorima, Naivnog Bayesa i K-najbližeg susjeda. Istražuju se različite metrike uspješnosti klasifikacije, uključujući matricu konfuzije, točnost, preciznost, opoziv, F-mjere i područje ispod krivulje AUC (engl. *Area under the Curve*). Opisuje se važnost pravilnog odabira metrike ovisno o karakteristikama problema klasifikacije i distribuciji klasa kako bi se osigurala pravilna evaluacija učinkovitosti klasifikatora.

Poglavlje 4. istražuje sigurnosna rješenja kroz integraciju SDN-a, naglašavajući inovativan pristup podržan strojnim učenjem, s posebnom pažnjom na primjenu strojnog učenja za otkrivanje napada u SDN-mrežama. Detaljno su opisani struktura, odabir i prilagodba ulaznih podataka modela za otkrivanje anomalija u mrežnom prometu. To uključuje analizu i usporedbu javno dostupnih referentnih skupova podataka poput UNSW-NB15, CSE-CIC-IDS2018, i LUFLOW2021, te NetFlow-skupova podataka, kao i predobradu tih podataka kroz isključivanje suvišnih značajki, čišćenje neispravnih vrijednosti te kodiranje kategorijskih značajki, uključujući upotrebu labeliranja i *One-hot*-metode. Nadalje, poglavlje detaljno opisuje proces odabira i prilagodbe referentnih skupova podataka za otkrivanje anomalija, uključujući skaliranje značajki i analizu utjecaja algoritama skaliranja na točnost klasifikacije. Poseban naglasak stavljen je na odabir optimalnog skupa podataka i klasifikatora te detaljan proces odabira hiperparametara klasifikatora unakrsnom validacijom. Predstavljen je NFMIDS (engl. *NetFlow Machine-learning Intrusion Detection System*) model za otkrivanje kibernetičkih napada, koji se sastoji od ključnih elemenata kao što su algoritmi strojnog učenja, referentni skupovi podataka, stvarni mrežni podatci te integracija s hibridnom SDN-arhitekturom. Također su analizirani postojeći modeli za otkrivanje napada u stvarnom vremenu i pružene smjernice za daljnji razvoj i implementaciju predloženog modela za otkrivanje prijetnji u mrežnom prometu. Analiza rezultata istraživanja pruža uvid u učinkovitost modela u različitim scenarijima i okolnostima, zaključujući o njegovoj primjenjivosti.

Poglavlje se 5. usredotočuje na prijedlog metode verifikacije modela za otkrivanje anomalija u kontekstu hibridne programski definirane mreže. Pruža pregled postojećih metoda verifikacije koje su primjenjive u sličnim kontekstima. Predstavljena je i opisana nova metoda verifikacije temeljena na analizi višestrukih NetFlow-zapisa mrežnog prometa. U tom su procesu korišteni NetFlow-zapisi s trima preklopnocima postavljenim duž putanje od napadača prema žrtvama. Otkrivanje anomalija rezultata uspoređena je s otkrivanjima postojećeg NIDS-a u korporativnoj

mreži. Predložena metoda verifikacije analizirala je očekivane i stvarne vrijednosti vremena otkrivanja i trajanja kibernetičkih napada te status sučelja SDN-preklopnika. Verifikacija je provedena u stvarnom vremenu, u hibridnom SDN-okruženju, uz prisustvo pozadinskog prometa i bez njega. Rezultati verifikacije pokazali su uspješno otkrivanje kibernetičkih napada u dvjema od triju konfiguracija NetFlow-izvoza, uz niski postotak lažno-pozitivnih rezultata za svaku konfiguraciju.

Poglavlje 6. zaključuje disertaciju i raspravlja o sljedećim koracima kao nastavak promatranog istraživanja.

2. IZAZOVI KIBERNETIČKE SIGURNOSTI U SDN-MREŽNOM OKRUŽENJU

Kako se razvijaju nove tehnologije i uređaji, koncept umrežavanja mijenja se s njima. Stare, tradicionalne mrežne paradigme pokazale su se previše statičnima i teškima za promjenu, stoga je nastala paradigma programski definirane mreže kako bi se bolje nosila s rastućim prometom podataka, virtualizacijom mreže i mobilnošću korisnika.

2.1. Tradicionalne mreže

U tradicionalnim mrežama, karakteristike uređaja čvrsto su vezane uz proizvođača, što otežava dodavanje novih funkcionalnosti i zahtijeva rad s mnogo različitih protokola implementiranih u različit hardver. Sigurnosni mehanizmi, pravila usmjeravanja i prosljeđivanja, također su čvrsto povezani za tip i proizvođača samog fizičkog uređaja. Promjene u mreži zahtijevaju ažuriranje susjednih uređaja, što postaje još složenije ako su uređaji raspoređeni na velikom području. Složenost mreža otežava primjenu postojećeg pristupa, sigurnosti, QoS-a i drugih funkcionalnosti [4], [5]. Sigurnost se u tradicionalnim mrežama često temelji na vatrozidima, sustavima za otkrivanje upada (IDS), sustavima za sprječavanje upada (IPS), kontroli i upravljanju pravilima pristupa. Postojećim statičkim sigurnosnim mehanizmima nedostaje prilagodljivost i skalabilnost. Nedostatak učinkovitih sigurnosnih mehanizama u tradicionalnim mrežama može dovesti do nedovoljne sigurnosti mreže [6], [7], [8].

2.2. Programski definirana mreža

Virtualne mreže nisu novost i postojale su u mnogim oblicima tijekom godina kao MPLS, VPN, ATM, Frame Relay i VLAN. SDN se pojavio kao nova paradigma umrežavanja koja razdvaja mrežni upravljački sloj od podatkovnog sloja čija je svrha odvajanje od specifičnih hardverskih tehnologija te značajno povećava agilnost mreže. Programski definirane mreže stvorene su za automatizaciju i radikalno pojednostavljenje upravljanja računalnim mrežama uz značajno smanjenje pogrešaka, što je slučaj kod ručnog rada. Omogućuje mrežama da se izravno

povezuju s aplikacijama aplikacijskim programskim sučeljima API-a, poboljšavajući izvedbe te stvarajući pokretljivost i dinamičnu mrežnu arhitekturu koja se po potrebi može mijenjati. Kontroleri mogu dinamički rekonfigurirati mrežu kako bi izbjegli zagušenja, implementirati nove usluge, dodali virtualnu infrastrukturu itd. [9], [10].

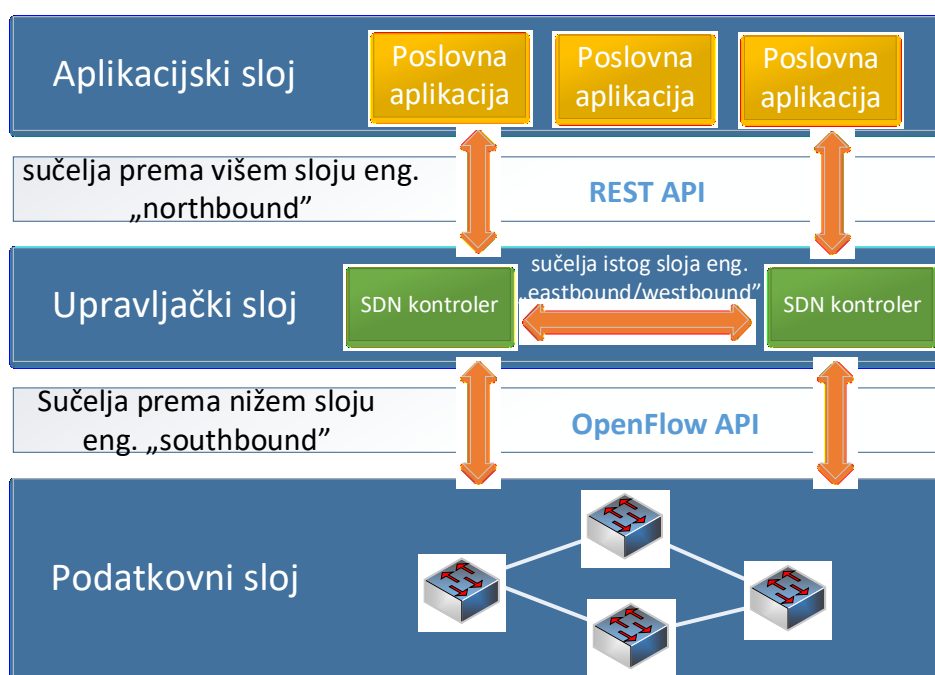
Programski definirana mreža omogućuje programabilnu kontrolu mreže, što olakšava prilagođavanje novim zahtjevima i dodavanje novih funkcionalnosti bez utjecaja na izvedbe, pouzdanost i korisničko iskustvo [11]. Novijim istraživanjima predlažu se mehanizmi temeljeni na programski definiranim mrežama kako bi se osigurala veća pokretljivost, dinamična programabilna funkcionalnost i smanjili operativni troškovi. SDN ima za cilj omogućiti dizajn dinamičnih i programabilnih sigurnosnih kontrola koje rade u stvarnom vremenu s malo ljudske interakcije ili bez nje, kako bi pružile pristup resursima legitimnim korisnicima, zaštitile sustave od napada i umanjile štetu u slučaju napada. Međutim, pojavom novih kontrola i mogućnosti u SDN-u, također se pojavljuju i nove vrste napada koje nisu bile prisutne u tradicionalnim mrežama [12]. SDN kao paradigma mrežne arhitekture, omogućuje dinamičko prilagođavanje mrežnog okruženja trenutnim zahtjevima ili potrebama korisnika i aplikacija te u velikoj mjeri pojednostavljuje upravljanje i proširivanje mreže. Dodatna je prednost mogućnost korištenja mrežnih komponenti različitih proizvođača koji podržavaju protokole prilagođene takvoj arhitekturi, bez potrebe poznavanja rada samih uređaja, jer se cjelokupnim mrežnim okruženjem upravlja putem kontrolera.

Neke od glavnih karakteristika SDN-arhitekture prema [13]:

- Programabilno upravljanje - kontrola i konfiguracija mreže izravno se programira jer je funkcija prosljeđivanja paketa odvojena od funkcije kontrole te omogućuje vrlo brzo konfiguriranje, upravljanje, osiguravanje i optimiziranje mrežnih resursa pomoću automatiziranih programa.
- Agilnost - izdvajanje kontrole pri prosljeđivanju paketa omogućuje dinamičko podešavanje protoka prometa na cijeloj mreži.
- Centralno upravljanje - mrežna inteligencija je centralizirana u kontrolerima koji imaju globalni pogled na mrežu.
- Otvoreni standardi - SDN se temelji na otvorenim standardima što pojednostavljuje dizajn i rad mreže jer kontroleri koriste iste protokole umjesto onih specifičnih za proizvođača.

Arhitektura modela SDN-mreže kao što je prikazano na slici 2.1. sastoji se od triju različitih slojeva koji se povezuju API-em:

- Aplikacijski se sloj sastoji od aplikacija krajnjeg korisnika koji koriste SDN-komunikacijske usluge.
- Upravljački sloj pruža konsolidiranu upravljačku funkcionalnost koja nadzire ponašanje prosljeđivanja paketa.
- Podatkovni sloj sastoji se od mrežnih elemenata i uređaja koji omogućuju prosljeđivanje paketa.



Slika 2.1. Arhitektura programski definirane mreže

U takvoj arhitekturi programski definirane mreže slojevi se povezuju API-em te se za sučelje određenog sloja prema nižem sloju koristi engleski naziv "*southbound*", dok se naziv "*northbound*" koristi za sučelje određenog sloja prema višem sloju.

Aplikacijski sloj čine krajnji korisnici i aplikacije koji se koriste SDN-mrežnim uslugama. Aplikacija može kontroleru podnijeti zahtjev za određenim promjenama u konfiguraciji i radu mreže. Zahtjevi za upravljanje mrežnom infrastrukturom odvijaju se preko „*northbound*“ API-sučelja prema kontroleru i tim se sučeljima osigurava apstraktni pogled na mrežu. Jedan od često korištenih API-a je REST API (engl. *REpresentational State Transfer API*).

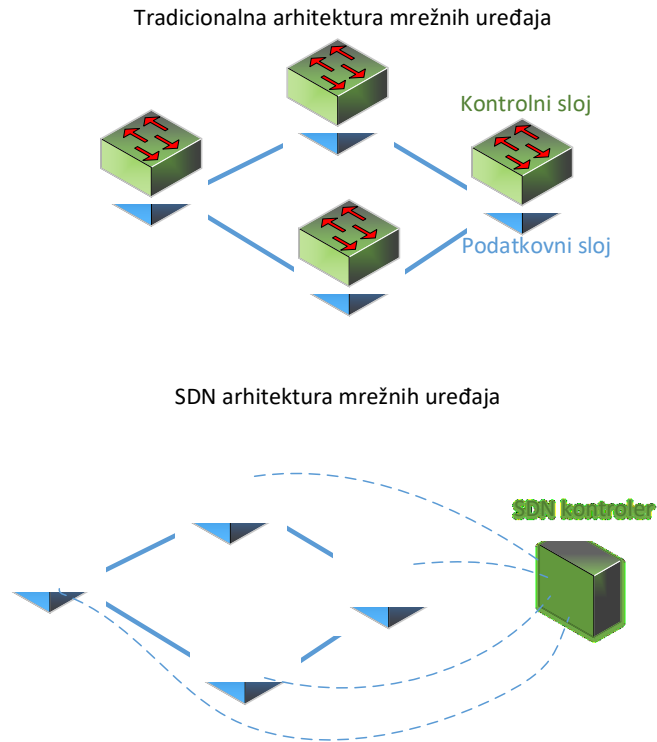
SDN-kontroler u upravljačkom sloju uglavnom je odgovoran za dva zadatka. Jedan je prevesti zahtjeve aplikacijskog sloja u podatkovni sloj, a drugi je dati apstraktni model fizičke mreže aplikacijskom sloju. Kontrolni se sloj često naziva mrežnim operativnim sustavom jer sadrži logiku upravljanja mrežom i pruža aplikacijskom sloju apstraktni prikaz globalne mreže. U SDN-okruženju kontroler koristi API-e za komunikaciju s aplikacijskim slojem, podatkovnim slojem i drugim kontrolerima. U distribuiranoj arhitekturi kontrolera, oni međusobno komuniciraju koristeći tzv. "eastbound"/"westbound"-API-e koji nisu toliko često korišteni za razliku od "northbound" i "southbound" API-a.

Podatkovni se sloj se sastoji od uređaja za prosljeđivanje paketa i to je glavna uloga tog sloja koji pruža vrlo učinkovite mehanizme za prosljeđivanje. Komunikacija upravljačkog i podatkovnog sloja odvija se tzv. "southbound"-API sučeljem i protokolima poput OpenFlow, CLI/SNMP, Netconf/YANG, ali detalji protokola obično su skriveni od korisnika API-a jer se korisnik ne mora brinuti o specifičnim protokolima koji implementiraju traženu mrežnu aktivnost [14].

Ključne komponente SDN-a su API-i koje ga čine moćnim alatom za upravljanje mrežom i rad sa značajkama kao što su programabilnost, neovisnost o protokolu, mogućnost izmjene mrežnih parametara prema potrebi. Upravljački sloj koristi API-e za nadgledanje, upravljanje i olakšavanje komunikacije svih ostalih slojeva. Jedna od vrijednosti SDN-a leži u činjenici da se API koristi na otvoren, neutralan i interoperabilan način [15], [16], [17], [18], [19].

Na slici 2.2. prikazana je usporedba tradicionalnih mrežnih uređaja i SDN-uređaja. Na gornjoj strani slike prikazani su tradicionalni mrežni uređaji, koji se temelje na čvrstim, zatvorenim hardverskim sustavima. To uključuje klasične preklopnike, usmjerivače koji zahtijevaju ručnu konfiguraciju i upravljanje, a samim time su i manje prilagodljivi.

Na donjoj strani slike prikazani su SDN-uređaji koji koriste fleksibilan i dinamičan pristup upravljanju mrežom putem softverskih kontrolera. Slikom 2.2 ističu se glavne prednosti SDN-a u usporedbi s tradicionalnim mrežama, uključujući veću agilnost, bolje iskorištavanje resursa i olakšano upravljanje mrežom. Ta usporedba opisuje evoluciju mrežne tehnologije prema SDN-u kao modernom pristupu upravljanju mrežama.



Slika 2.2. Usporedba tradicionalnih i SDN-mrežnih uređaja

2.3. Hibridna programski definirana mreža

Mreža koja sadrži kombinaciju SDN i tradicionalnih mrežnih uređaja obično se naziva hibridnom SDN-mrežom te nudi niz prednosti i predstavlja prijelazni korak ka potpunom usvajanju SDN-a. Hibridna SDN-mreža spaja tradicionalno umrežavanje i SDN-protokole koji djeluju u istom okruženju te omogućava uvođenje novih tehnologija i protokola u tradicionalna okruženja bez potpune rekonfiguracije mrežne arhitekture. Potpuna promjena tradicionalne mreže u programski definiranu mrežu bez ikakvog oblika prilagodbe predstavlja rizik koji može utjecati na izvedbu i sigurnost IT-sustava. Nadalje, potrebna su velika financijska sredstva za sastavnice SDN-mreže, a nadogradnja relativno novih tradicionalnih uređaja mreže smatra se neisplativom [20]. Prema [21], [22], [23], [24] hibridne SDN-mreže nude niz prednosti:

- Olakšavaju financijske troškove jer je implementacija potpune SDN-mreže vrlo skupa i potrebna su dodatna ulaganja za edukaciju, projektiranje, konfiguriranje i rad na SDN-mreži.

- Mogu se koristiti za iskorištavanje nekih prednosti SDN-paradigme bez implementacije pune SDN-mreže. Pristupna mreža može koristiti naslijeđene, tradicionalne uređaje, dok distribucijska mreža može koristiti SDN-uređaje. Hibridna se SDN-mreža može koristiti za obradu i prosljeđivanje većine paketa u pristupnoj mreži naslijeđenim, tradicionalnim uređajima, dok se SDN-uređaji koriste u distribucijskoj mrežnoj razini kako bi iskoristili prednosti SDN-a. Da bi osigurali nesmetan i kontroliran prijelaz, dobro je na početku implementirati SDN samo za mali dio nekritičnog prometa.
- SDN omogućuje finu granulaciju kontrole protoka podataka. Ako je potrebna kontrola samo za mali dio mreže, može se implementirati hibridna SDN-mreža, dok ostatak mreže koristi tradicionalno umrežavanje.
- Tradicionalni protokoli usmjeravanja vrlo su učinkoviti za neke zadatke poput povezivanja kontrolera SDN-a kako bi kontrolirali različite dijelove mreže. Hibridna SDN-mreža može se primijeniti kako bi se oslobodio kontroler od zadataka koji se učinkovito mogu odraditi s tradicionalnim protokolima usmjeravanja.
- Hibridna SDN-mreža olakšava prijelaz s naslijeđenih, tradicionalnih na SDN-mrežne uređaje gdje se može raditi postupno raspoređivanje sve više i više SDN-uređaja i procijeniti utjecaj zamjene tradicionalnih uređaja.
- Hibridnom mrežom rješava se povezivanje dviju odvojenih SDN-mreža preko tradicionalnih mrežnih uređaja.

Nekoliko je mogućih hibridnih SDN-modela koji se opisuju u literaturi [21], [25]:

- Model temeljen na topologiji gdje je mreža podijeljena u zone tako da svaki čvor pripada samo jednoj, tradicionalnoj ili SDN-zoni.
- Model temeljen na servisu gdje su usluge podijeljene za tradicionalni i SDN-dio mreže. Da bi se implementirale neke usluge, poput prosljeđivanja na razini cijele mreže, određeni čvorovi mogu pripadati objema paradigmama.
- Model temeljen na klasifikaciji i podjeli prometa u tradicionalni i SDN-nadzorni promet.
- Integrirani model gdje je SDN odgovorna za sve mrežne usluge, a koristi tradicionalne protokole kao sučelje za prosljeđivanje paketa te tako može nadzirati put prosljeđivanja umetanjem pažljivo odabranih ruta u sustav usmjeravanja ili podešavanjem postavki protokola.

Hibridna SDN-mreža predstavlja mogući način migracije tradicionalne mreže na potpunu SDN-mrežnu arhitekturu.

2.4. Pregled istraživanja i rješenja kibernetičke sigurnosti u SDN-mrežnoj arhitekturi

Pregled istraživanja o kibernetičkim napadima s pogleda SDN-mrežne arhitekture, uključuje različite vidove poput prijetnji, sigurnosnih izazova, rješenja i trenutnih istraživanja o prijetnjama relevantnim za SDN arhitekturu. Proučena istraživanja pokazuju kako SDN mijenja tradicionalni model sigurnosti mreža i kako se prilagođava novim potrebama.

Autori u [26] opisuju sustav namijenjen otkrivanju i suzbijanju prijetnji na mreži u SDN-okruženju. Sustav samostalno nadzire promet i primjenjuje protumjere na korištenim uređajima kako bi održao dostupnost usluga mreže. U tu svrhu, koristi analizu prometa s više značajki za profiliranje normalne upotrebe mreže te normalni profil, koristi se za identifikaciju neobičnih uzoraka prometa, a politika obrane odabire se prema otkrivenim anomalijama. Predloženi sustav je razvijen korištenjem testnog prometa koji simulira DDoS i skeniranje porta. Sustav identificira sučelja koja su napadači koristili i koristi tu informaciju za pokretanje definiranih sigurnosnih rutina na preklopnicima i tako uspostavlja ispravno funkcioniranje mreže. Performansa predloženog sustava promatrana je u odnosu na vrijeme izvršavanja i korištenje resursa sustava, uključujući upotrebu CPU-a SDN-kontrolera i veličinu tablice protoka preklopnika.

Napredak interneta, tehnologije u medicini i prije svega nesigurna priroda zastarjelih zdravstvenih sustava, povećavaju širinu kibernetičkih napada. U radu je [27] usmjerena pažnja na protokole u industrijskim sustavima kao i u sektoru zdravstva. Predstavljen je kvantitativni model prijetnji koji procjenjuje ozbiljnost mogućih kibernetičkih napada u vezi s odgovarajućim naredbama promatranog protokola. Predložen je sustav koji kombinira strojno učenje (ML) i programski definirane mreže (SDN) kako bi se otkrili i ublažili kibernetički napadi modificiranjem i dodavanjem tablica protoka SDN-preklopnika koristeći OpenFlow. Otkrivanje upada temelji se na klasifikatoru koji koristi statistike mrežnog protoka TCP/IP i statistike toka podataka IEC 60 870-5-104 protokola. Rezultati evaluacije pokazali su učinkovitost predloženog IDPS-a mjerenjem uspješnosti klasifikacije.

U radu je [28] predloženo sigurnosno rješenje koje otkriva DDoS napade prema SDN-kontroleru korištenjem odgovarajućih ML-algoritama. SDN, temeljen na prilagodljivom mehanizmu propusnosti, imao je važnu ulogu u unaprjeđenju sigurnosti SDN-kontrolera, gdje je XGBoost-algoritam korišten za sprječavanje DDoS-napada. U analizi su korišteni CICDDoS2019 i NSL-KDD-skup podataka za ML-algoritme kako bi se poboljšalo sigurnosno rješenje s prilagodljivim mehanizmom propusnosti. Postavljena su tri profila za usporedbu rezultata gdje XGBoost ima potpunu iskoristivost CPU-a i drugih izvora uz minimalne troškove energije i složenost sustava. Korištenje tog pristupa s algoritmom prilagodljive propusnosti i ML-tehnikom, poboljšava se točnost predloženog rješenja i smanjuje omjer odbačenih paketa.

Prema istraživanju u radu [29], autori predlažu novi način implementacije sustava za sprječavanje upada temeljenog na SDN/OpenFlow arhitekturi. U predloženom rješenju smanjuju se troškovi implementacije i održavanja SDN-mreže gdje se SDN-kontroler koristi za racionalno raspoređivanje protoka podataka. U usporedbi s implementacijom IPS-a u tradicionalnim mrežama, predloženo rješenje omogućava unificirano raspoređivanje sigurnosnih pravila u cijeloj mreži. Postiže se jedinstvena učinkovitost u smislu sigurnosti od kibernetičkih napada i koordinacije IPS-ova u cijeloj mreži. SDN-kontroler koristi prednosti fleksibilnog raspoređivanja protoka kako bi se poboljšala iskoristivost nekih neiskorištenih IPS-ova i smanjila opterećenja nekih preopterećenih IPS-ova.

Pristup SDN-arhitekture s mamcima predstavljeni su u istraživanjima [30] i [31]. Mamci su vrsta aktivne obrambene sigurnosne tehnologije i to je mjesto za koje se očekuje da će biti napadnuto. Ti sustavi mogu simulirati veliku i stvarnu mrežu kako bi privukli napadače i preusmjerili upade na mamce te tako osigurali vrijeme za daljnju analizu. Takav pristup u SDN-u poboljšava nedostatke postojećih tehnologija podmetanja mamaca, čiji su mehanizmi uočljivi i napadači ih mogu lako otkriti. SDN-kontroler korisnicima omogućuje konfiguriranje vlastitih pravila upravljanja mrežnim podacima, koja će na temelju upozorenja proslijediti ili preusmjeriti promet na odgovarajuće mamce. SDN-kontroler pruža simulaciju topologije mreže u hibridnoj mreži i osigurava preciznu kontrolu podataka tijekom cijele migracije zlonamjernog prometa, čime se poboljšavaju nedostaci u tehnologiji kontrole protoka u tradicionalnoj mreži s mamcima.

Neki IDS-ovi strukturirani su kao usluga (engl. *Intrusion Detection System as a Service - IDSaaS*) u SDN-u koja nastoji otkriti i zaustaviti ulazak zloćudnog prometa i korištenje mrežnih

elemenata u složenijim okruženjima i ispitivanje s realističnijim scenarijima. Prednost IDSaaS-strategije proizlazi iz općeg trenda hijerarhijskog upravljanja mrežom, odnosno, ako SDN-kontroler nije pravilno osiguran, klijentski čvorovi (pojedinačni elementi ili mreže koji ovise o uslugama kontrolera) postaju izloženiji napadima i anomalijama u podatkovnom prometu. IDSaaS ima sposobnost postizanja dosljednosti u implementaciji pravila IDS-a na više SDN-mreža. Implementacija IDSaaS-a za SDN rezultira i uštedama troškova i vremena pri instalaciji IDS-a za sve pojedine mreže [32], [33].

Predloženi pristup u [34] predstavlja okvir za otkrivanje DDoS-napada, koristeći statističke parametre u okviru SDN-arhitekture. Predloženi okvir rješava problem vezan uz DDoS-napad i SDN-arhitekturu i ograničenje IDS-a u brzim mrežama. Osim toga, predloženo rješenje za otkrivanje upada pruža dobru pretpostavku napada s visokom učinkovitošću otkrivanja. Eksperimentalni rezultati pokazuju da je predloženi okvir uspješan u postizanju ravnoteže među učinkom otkrivanja i učinkovitošću okvira u brzom mreži. Osigurava nisku stopu pogreške uz visoku stopu i snagu otkrivanja. Nadalje, pruža ekonomičan pristup bez korištenja vanjske hardverske opreme izvan SDN-a.

Istraživanje [35] razmatra otkrivanje upada u IoT-u, gdje je predložen mehanizam otkrivanja upada dubokim učenjem uz korištenje SDN-a kako bi onemogućili rastuće kibernetičke prijetnje u IoT-u. Predloženi sustav pruža jedinstvenu sposobnost otkrivanja raznovrsnih potencijalnih sigurnosnih prijetnji, uključujući DOS, DDOS, MITM, botnet-napade, infiltracijske napade, napade grubom silom, napade skeniranja itd. Izvedbe predloženog modela razvijene su ML-metrikama, uključujući točnost, preciznost, odziv, F1-ocjenu. U validaciji, predloženi okvir uspoređen je s dvama referentnim klasifikatorima u IoT-okruženjima.

U radu [36] predstavljen je okvir za poboljšanje sigurnosti u podatkovnim centrima temeljenim na SDN-u. Okvir se oslanja na uvođenje upravljačke jedinice koja se sastoji od sigurnosnog agenta i internog poslužitelja dnevničkih zapisa koji se prikupljaju s različitih sigurnosnih uređaja. Sigurnosni agent analizira takve zapise kako bi zaustavio napadače SDN-kontrolerima. Izrađen je i implementiran koncept koji se izvodi na emulacijskom okruženju.

Jedan su od najčešćih napada u mreži napadi uskraćivanjem usluge (DoS) i distribuirani DoS (DDoS)-napadi. U istraživanju [37] pokazano je da određene značajke SDN-a mogu biti korištene kako bi se uočili i spriječili upadi te gotovo trenutačno odbacili pakete kad se otkrije

napad. Implementiran je i testiran IDPS temeljen na anomalijama mrežnog prometa. Korištena su dva tipa algoritama za praćenje komunikacije temeljenih na brzinama protoka podataka: CB-TRW i RL. Predstavljena je i tehnika otkrivanja skeniranja portova. Prema rezultatima eksperimenata u SDN-okruženju, IDPS se pokazao sposobnim za otkrivanje Nmap-skeniranja portova i različitih vrsta DoS-napada. Dio sustava za otkrivanje upada uključivao je upozorenja SDN-kontrolera o prisutnosti napada, prikazujući koji je algoritam otkrio napad, vrijeme napada i obavijest o prirodi napada. Dio sustava za sprječavanje upada automatski je stvarao i slao promjene toka mrežnim preklopniciima kako bi odbacio pakete napadača. Dodatna ispitivanja također su provedena korištenjem stvarnog mrežnog prometa kako bi se mjerio odnos lažno pozitivnih rezultata, postavki praga algoritma i korištenja CPU-a.

Istraživanje u [38] predstavlja rješenje za sprječavanje napada skeniranja portova. Korištenjem statistika mrežnog toka prikupljenih u SDN-mreži, otkriveni su tokovi skeniranja portova i osigurana je sigurnost mreže ažuriranjem OpenFlow-pravila usmjeravanja podataka. Rezultati, prikazani u eksperimentalnoj evaluaciji, pokazuju učinkovitost u otkrivanju zlonamjernih tokova što rezultira odsustvom lažno pozitivnih i vrlo malim brojem lažno negativnih rezultata.

U radu [39] je provedena analiza sigurnosnih ispitivanja sustava koja se odnosi na trajanje zaustavljanja otkrivenih napadača. Sustav omogućuje slanje informacija o toku mrežnog prometa putem REST API-a u Ryu kontroler za zaustavljanje i uklanjanje pravila s određenog uređaja. Na temelju rezultata ispitivanja svih scenarija, SDN-mreža opremljena adaptivnim IPS-om ima sposobnost otkrivanja kibernetičkih napada te može zaustaviti napadača u trajanju ovisno o učestalosti i vrsti izvršenih napada, uz minimalno dodavanje dodatnog vremena za izvođenje fuzzy algoritma u procesu otkrivanja.

Tablica 2.1. Pregled literature s aspekta kibernetičkih napada i SDN-mrežne arhitekture

#	godina	SDN	stvarno vrijeme	stvarna testna okolina	problem istraživanja
[26]	2018.	✓	✗	✗	sigurnost IT-sustava od DDos i skeniranja portova.
[27]	2022.	✓	✗	✓	sigurnost zdravstvenog IT-sustava.
[28]	2021.	✓	✗	✗	sigurnost SDN-kontrolera i sprječavanje DDoS-napada.
[29]	2013.	✓	✗	✗	smanjivanje troškova implementiranja i održavanja SDN-mreže, poboljšavanje iskoristivosti IPS-sustava u SDN-okruženju.
[30]	2019.	✓	✗	✗	preusmjeravanje neželjenog prometa na mamce.
[31]	2017.	✓	✗	✗	preusmjeravanje neželjenog prometa na mamce.
[32]	2017.	✓	✓	✓	IDS kao usluga u SDN-u.
[33]	2017.	✓	✓	✓	IDS kao usluga u SDN-u za mobilne mreže.
[34]	2021.	✓	✗	✗	otkrivanje DDoS-napada
[35]	2022.	✓	✗	✗	otkrivanje kibernetičkih napada u IoT-okruženju.
[36]	2016.	✓	✗	✗	sigurnost u podatkovnim centrima.
[37]	2019.	✓	✓	✗	otkrivanje DoS-napada i skeniranja portova u SDN-mreži.
[38]	2018.	✓	✗	✗	otkrivanje skeniranja portova na uređajima u SDN-mreži.
[39]	2018.	✓	✗	✗	adaptivni IPS u svrhu obrane od DoS-napada.

Provedena istraživanja odražavaju neprekidno zanimanje za SDN i sigurnost općenito. Sve analizirane studije uključuju implementiranje SDN-a, što ukazuje na opću prihvaćenost

tehnologije u sektoru kibernetičke sigurnosti. Primjena strojnog učenja nije česta, uglavnom se pojavljuje u kontekstu otkrivanja prijetnji. Tek nekoliko studija naglašava potrebu za obradom u stvarnom vremenu, što je bitno u okviru sigurnosnih rješenja koja zahtijevaju brze reakcije. Neki radovi koriste stvarne testne okoline, posebno oni koji se bave mobilnim mrežama i SDN-uslugama. Raznoliki problemi istraživanja obuhvaćaju širok spektar tema, uključujući sigurnost od DDoS-napada, sprječavanje skeniranja portova, poboljšavanje iskoristivosti IPS-sustava u SDN-okruženju, preusmjeravanje neželjenog prometa te otkrivanje prijetnji u različitim kontekstima. Unatoč nedostatku primjene strojnog učenja u većini radova, učestalost SDN-tehnologije ukazuje na važan princip u pristupu sigurnosti mreže. Uvođenje stvarnih testnih okolina doprinosi valjanosti rezultata, a raznolikost istraživačkih pitanja naglašava složenost izazova kibernetičke sigurnosti u kontekstu SDN-a. Postizanje ravnoteže između učinkovitosti obrambenih tehnika i minimalne potrošnje resursa također predstavlja izazov, s obzirom na pritisak koji kibernetički napadi vrše na ciljane resurse. Potrebno je osigurati obrambeni mehanizam koji minimalno opterećuje resurse sustava, dok istovremeno uspješno umanjuje utjecaj kibernetičkih napada.

Napadači koriste sofisticirane metode za izvođenje napada visoke složenosti, stoga je potrebno razviti model koje omogućuje brze reakcije u sigurnosnom kontekstu SDN-a kako bi se ostvarila obrada u stvarnom vremenu. Također, treba istražiti kako integrirati strojno učenje u različite vidove sigurnosti mreže ne samo u otkrivanju već i u onemogućavanju kibernetičkih prijetnji, posebice kritičnih IT-sustava. Testiranje u stvarnoj testnoj okolini doprinijet će dobivanju valjanijih rezultata istraživanja, dok će istovremeno pronaći optimalno rješenje koje minimizira utjecaj obrambenih tehnika na resurse sustava, uz istodobno učinkovitu zaštitu od kibernetičkih napada.

3. STROJNO UČENJE I PRIMJENA ZA OTKRIVANJE UPADA U SDN-MREŽNOM OKRUŽENJU

Zanimanje za sustave za otkrivanje upada (IDS) temeljene na strojnom učenju ubrzano raste tijekom zadnjih godina, a njihova je svrha otkrivanje upada kroz analizu mrežnih podataka. Ključan aspekt tih sustava je opsežno prikupljanje mrežnih podataka te primjena rudarenja podataka kako bi se izgradili modeli koji precizno bilježe normalno ponašanje i zlonamjerne upade u mrežnom prometu. Postoji više metoda za otkrivanje anomalija koje koriste različite pristupe analiziranju podataka, uključujući statističke metode i algoritme nadziranog, polunadziranog, podržanog i nenadziranog strojnog učenja. Primjena programski definirane mreže (SDN) kao tehnologije za mreže predstavlja inovativan pristup koji omogućuje razvoj sigurnosnih rješenja prilagođenih mrežnoj programabilnosti i fleksibilnosti, uz integraciju podrške za strojno učenje. Prilagodba sustava dinamičkim mrežnim uvjetima i stanjima otvara nove mogućnosti za upravljanje i konfiguraciju mreže [40].

Novi pristup otkrivanju upada uključuje i niz izazova na koje treba obratiti pažnju kako bi se postigli ciljevi kibernetičke sigurnosti i riješili sljedeći problemi:

- **Brzina otkrivanja u stvarnom vremenu:** potrebno je osigurati brzo otkrivanje zlonamjernog dolaznog prometa u stvarnom vremenu kako bi se što prije reagiralo na moguće prijetnje.
- **Prilagodba dinamičnoj mrežnoj promjeni:** IDS-sustav mora biti sposoban brzo se prilagoditi čestim promjenama stanja u mreži kako bi zadržao visoku učinkovitost.
- **Reakcija na rijetke i nepoznate napade:** potrebno je osigurati da se sustav može uspješno nositi s rijetkim i nepoznatim napadima uz minimalnu nesigurnost u obradi odluka.
- **Primjenjivost tehnika strojnog učenja u SDN-u:** potrebno je istražiti koje su tehnike strojnog učenja primjenjive i najučinkovitije u kontekstu programski definiranih mreža.
- **Odabir algoritma i relevantnih značajki klasifikatora:** bitno je odabrati najrelevantnije algoritme i značajke klasifikatora strojnog učenja za precizno otkrivanje napada.

- **Smanjenje broja značajki:** osigurati smanjenje broja značajki u skupu podataka kako bi se postigla brza obrada uz visoku točnost otkrivanja.
- **Smanjenje nesigurnosti u odlukama:** minimizirati nesigurnost u obradi odluka kako bi se osiguralo pouzdano otkrivanje.

SDN stvara nove mogućnosti za implementaciju učinkovitijih metoda otkrivanja upada. Zbog otvorenosti platformi koje podržavaju tehnologije SDN-a, moguće je koristiti postojeće mehanizme i protokole za prikupljanje podataka koji se mogu koristiti kao izvor podataka za algoritme otkrivanja upada. Programabilnost SDN-a i strojno učenje pružaju vrlo učinkovitu tehniku obrane od kibernetičkih napada primjenjivu u raznim područjima. Ubrzanim razvojem na području komunikacijske povezanosti ubrzava se stopa automatizacije, ali u isto vrijeme predstavlja mnoge sigurnosne izazove, stoga sustavi za otkrivanje i sprječavanje upada imaju zadaću unaprijediti iskorištenost resursa, povećati performanse, skalabilnost i sigurnost kritičnih sustava u stvarnom vremenu. Kombinacija strojnog učenja i SDN-a jača obrambene mehanizme, čineći ih prilagodljivima i učinkovitima u suočavanju s dinamičnim kibernetičkim prijetnjama.

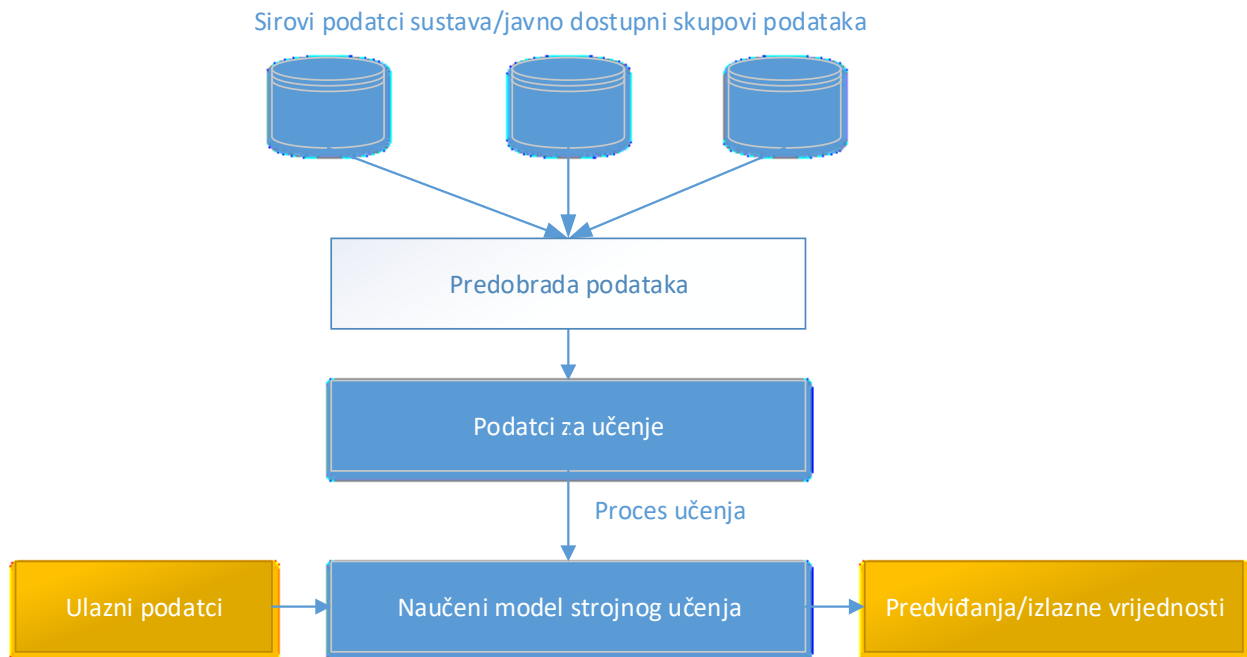
3.1. Strojno učenje

Arthur Samuel, pionir proučavanja umjetne inteligencije, 1959. godine opisao je strojno učenje kao "znanstvenu disciplinu koja računalima daje mogućnost učenja bez izričitog programiranja". Osnovni rad Alana Turinga iz 1950. godine uveo je referentni standard za pokazivanje inteligencije stroja takav da stroj mora biti inteligentan i reagirati na način koji se ne razlikuje od ljudskog bića. Tehničku definiciju dao je Tom M. Mitchell 1997. godine koja glasi: „Za računalni se program kaže da uči iz iskustva E s obzirom na neku klasu zadataka T i mjeru izvedbe P , ako se izvedba na zadacima u T , mjerena s P , poboljšava iskustvom E . " [52].

Prema [53], tipični zadatci strojnog učenja mogu se opisati kao:

- **Klasifikacija ili kategorizacija:** uključuje popis zadataka ili problema za koje bi stroj trebao koristiti uzorke te svakom uzorku dodijeliti određenu kategoriju ili klasu.

- **Regresija:** vrsta zadatka koji obično izvode predviđanja na način da će rezultat biti stvarna numerička vrijednost za zadanu ulaznu vrijednost podatka.
- **Otkrivanje anomalija:** uključuje analizu zapisnika događaja, izvješća o transakcijama i drugih podatkovnih zapisa na način da se otkriju anomalije i obrasci/događaji koji se razlikuju od uobičajenog ponašanja.
- **Strukturirana bilješka:** uključuje provođenje analize na ulaznim podacima i dodavanje strukturiranih metapodataka kao bilješke izvornim podacima kako bi dodatno opisali odnose među elementima podataka.
- **Prijevod:** funkcije automatiziranog strojnog prevođenja ulaznih podataka koji pripadaju određenom jeziku na drugi jezik.
- **Klasteriranje ili grupiranje:** klasteri ili grupe se formiraju od uzoraka ulaznih podataka provjerom karakterističnih obrazaca, sličnosti i odnosa među ulaznim podacima.



Slika 3.1. Osnovni princip rada strojnog učenja [54]

Slika 3.1. prikazuje pojednostavljeni postupak strojnog učenja. Postupak počinje prikupljanjem ulaznih podataka iz različitih izvora, uključujući javno dostupne skupove podataka, senzore ili druge izvore sirovih podataka. Nakon prikupljanja, ti se podatci pripremaju za strojno učenje. Taj korak uključuje čišćenje podataka, uklanjanje nevažnih ili nepotpunih informacija,

skaliranje i kodiranje kategorijskih podataka kako bi se osigurala kvaliteta ulaznih podataka za model. Nakon pripreme podataka, odabire se odgovarajući model strojnog učenja koji će se koristiti za treniranje. Model se trenira prikupljenim podacima kako bi naučio njihove uzorke i relacije. Kada je model uspješno naučen, može se koristiti za predviđanje rezultata na novim ulaznim podacima. Ovisno o prirodi problema, to može biti jednostavan model poput linearne regresije, klasifikacije ili složeniji model poput neuronske mreže. Osnovni princip strojnog učenja leži u sposobnosti algoritama da automatski identificiraju obrasce iz ulaznih podataka i njihovih prošlih vrijednosti kako bi donosili predviđanja ili odluke kao izlazne vrijednosti. Nakon implementacije modela, važno je sustavno pratiti performanse kako bi se osigurala preciznost i pouzdanost u predikcijama. Algoritmi prilagođavaju svoje parametre za izgradnju optimalnih logičkih modela kako bi minimizirali razliku među stvarnim i predviđenim izlaznim vrijednostima. Osim toga, moguće je povremeno ponovno treniranje modela s novim podacima kako bi se osigurala njegova prilagodljivost promjenama u okolini ili problemu [54], [55].

3.1.1. Kategorije strojnog učenja

Općenito se metode strojnog učenja prema količini ljudskog nadzora u procesu učenja mogu podijeliti na nadzirano učenje (engl. *supervised learning*), nenadzirano učenje (engl. *unsupervised learning*), polunadzirano učenje (engl. *semi-supervised learning*) i podržano učenje (engl. *reinforcement learning*).

Nadzirano učenje: cilj je nadziranog učenja naučiti model iz označenih podataka koji omogućuju predviđanje budućih podataka. Izraz „nadzirani“ odnosi se na skup uzoraka gdje su izlazne oznake već poznate. Tako naučeno znanje može se koristiti za predviđanje izlaza za bilo koji novi uzorak ulaznih podataka koji je prethodno bio nepoznat ili neviđen tijekom procesa učenja modela. Dva najčešća zadatka nadziranog strojnog učenja su klasifikacija i regresija. U problemima klasifikacije predviđaju se diskretne vrijednosti, odnosno predviđa se najvjerojatnija kategorija, klasa ili oznaka za nove uzorke. U problemima regresije predviđa se vrijednost varijable kontinuiranog odziva, odnosno pokušava se zavisnost varijabli opisati neprekidnom funkcijom.

Nenadzirano učenje: pokušava se otkriti uzorke podataka s nerazvrstanim i neoznačenim podacima. Jedan je od uobičajenih zadataka nenadziranog učenja grupirati slične podatke odnosno

grupirati slične elemente u „klastere” ili svesti podatke na mali broj „važnih dimenzija”. Nenadzirano učenje više se bavi pokušajem izvlačenja korisnih informacija iz podataka, a ne predviđanjem ishoda.

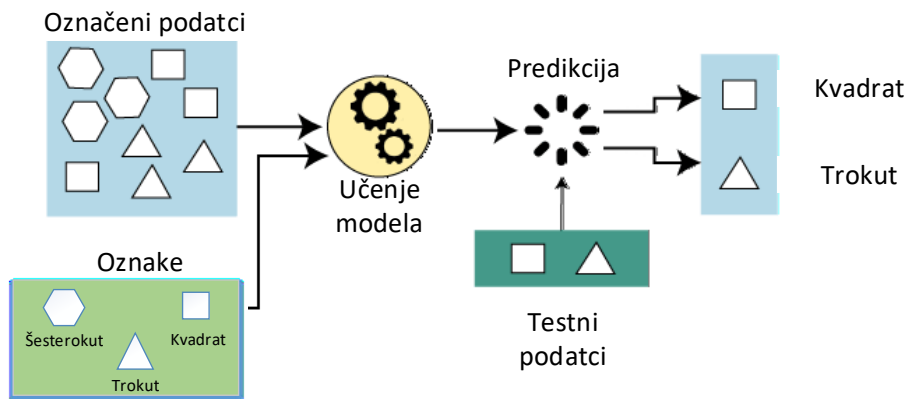
Polunadzirano učenje: konceptualno je smješteno između nadziranog i nenadziranog učenja, omogućuje iskorištavanje velike količine neobilježenih podataka u kombinaciji s tipično manjim skupovima označenih podataka. Obično polunadzirani algoritmi strojnog učenja pokušavaju poboljšati izvedbu jedne od prethodno spomenutih metoda korištenjem uzajamno povezanih informacija.

Podržano učenje: odnosi se na ciljno orijentirane algoritme koji uče kako postići složeni cilj ili maksimizirati vrijednost određene dimenzije u više iteracija. Ta metoda omogućuje strojevima i softverskim agentima automatsko određivanje idealnog ponašanja u određenom kontekstu kako bi se maksimizirale izvedbe. Potrebne su povratne informacije kako bi se naučilo koja je akcija najbolja u postizanju maksimalne vrijednosti [52], [53], [56].

3.1.2. Nadzirano strojno učenje

Nadzirano učenje nudi široku primjenjivost na različite probleme poput klasifikacije, regresije i otkrivanja anomalija, što ga čini izuzetno fleksibilnim za raznolike zadatke strojnog učenja. Razlozi za odabir nadziranog strojnog učenja uključuju jasno definirane ciljne izlazne vrijednosti (predviđanja) koje su ključne za klasifikaciju i otkrivanje anomalija. Taj pristup omogućuje precizno modeliranje uzoraka u podacima te stvara temelj za pouzdane predikcije. Također, nadzirano učenje olakšava kontrolu nad procesom učenja, omogućujući preciznu procjenu izvedbi modela, što je od vitalnog značaja za otkrivanje anomalija i klasifikaciju u stvarnim scenarijima primjene.

Nadzirano je strojno učenje grana strojnog učenja koja se temelji na razvoju modela za predviđanje budućih ishoda, a to postiže učenjem na temelju ulaznih i izlaznih parova podataka ili označenih podataka. Cilj je takvog modela stvaranje funkcije koja je dovoljno dobra u aproksimaciji i koja može predviđati izlazne vrijednosti novih ulaznih podataka [55]. Princip rada nadziranog učenja može se vizualizirati i razumjeti kroz sljedeći dijagram na slici 3.2.



Slika 3.2. Princip rada nadziranog strojnog učenja [55]

Problemi u nadziranom strojnom učenju mogu se grupirati u probleme regresije i probleme klasifikacije. U promatranom istraživanju uspoređene su različite vrste klasifikacijskih algoritama korištenih u svrhu otkrivanja anomalija mrežnog prometa, a prema [57], algoritmi klasifikacije nadziranog strojnog učenja mogu se podijeliti na:

- algoritme temeljene na logici (stablo odlučivanja, slučajna šuma)
- algoritme strojeva s potpornim vektorima (Strojevi potpornih vektora klasifikacije ili regresije)
- algoritmi temeljeni na statistici (Naivni Bayesovi algoritmi)
- algoritam lijenog učenja (K-najbliži susjed).

Na primjeru otkrivanja anomalija mrežnog prometa teži se postizanju cilja pronalaženja sustavnog načina predviđanja anomalija s obzirom na ulazni skup podataka. U smislu strojnog učenja, taj se cilj formulira kao zadatak izvođenja modela nadziranog strojnog učenja nad prikupljenim podacima, gdje model predviđa vrijednost izlazne varijable na temelju promatranih vrijednosti ulaznih varijabli. Pronalaženje odgovarajućeg modela temelji se na pretpostavci da se za izlaznu varijablu ne uzima nasumična vrijednost i da postoji povezani odnos među ulaznim i izlaznim vrijednostima [58].

U nadziranom učenju promatramo ulaznu varijablu (vektor značajki) $X \in R^d$ koja predstavlja poznate informacije i izlaznu varijablu (oznaku) Y koja predstavlja nepoznate informacije koje želimo predvidjeti. Cilj je predvidjeti Y temeljem X . U praksi, skup pravila za predviđanje izveden je parametriziranim funkcijama $f(w, *): R^d \rightarrow R^k$ gdje $w \in \Omega$ predstavlja parametar modela koji

se može naučiti temeljem podataka za učenje. Na primjer, za problem klasifikacije, gdje je $Y \in \{1, \dots, k\}$, predviđa se Y sljedećim pravilom predviđanja prema funkciji $f(w, x) = [f_1(w, x), \dots, f_k(w, x)] \in R^k$.

Kvaliteta predviđanja mjeri se funkcijom gubitka $L(f(X), Y)$: što je gubitak manji, to je točnost predviđanja bolja.

Nadzirani pristup strojnom učenju uključuje upotrebu označenih podataka kako bi se algoritmu pružile jasne smjernice ili primjeri tijekom učenja. Označeni podatci sadrže ulazne podatke $S_n = [(X_1, Y_1), \dots, (X_n, Y_n)]$ povezane s odgovarajućim izlaznim vrijednostima, čime se omogućuje modelu da uči od postojećih informacija i prilagodi se temeljem dobivenih podataka. Nastoji se procijeniti vrijednost $\hat{w} \in \Omega$ temeljem promatranog ulaznog skupa S_n . Algoritam nadziranog učenja A koristi skup podataka za učenje S_n kao ulaz te generira funkciju $f(\hat{w}, *)$ gdje je $\hat{w} = A(S_n) \in \Omega$. Nadzirano učenje često se koristi u zadacima poput klasifikacije i regresije, gdje je cilj predvidjeti odgovarajuću klasu ili numeričku vrijednost temeljem ulaznih podataka [59].

3.1.3. Klasifikacija u strojnom učenju

Klasifikacija u strojnom učenju koristi podatke i algoritme kako bi podatkovni uzorci bili svrstani u različite kategorije ili klase. Izbor najboljeg algoritma za klasifikaciju strojnog učenja prema [60] ovisi o raznim čimbenicima, uključujući:

- strukture podataka: ukupna veličina skupa podataka, broj značajki i raspodjela klasa
- složenost problema: jednostavniji algoritmi poput logističke regresije mogu imati dobre rezultate za jednostavnije probleme, dok se složeniji problemi bolje rješavaju algoritmima poput stabla odlučivanja ili umjetne neuronske mreže
- kvaliteta podataka: neadekvatne vrijednosti i korelacija među značajkama
- vrijeme izvođenja: vrijeme potrebno za izvođenje u sustavu kritičnih i nekritičnih procesa.

Za postizanje što boljih rezultata klasifikacije, potrebno je testirati različite algoritme i izabrati onaj koji najbolje funkcionira za određeni problem i skup podataka. Poželjna je i primjena metoda kao što su unakrsna validacija i izolirana provjera kako bi se izbjegli problemi prekomjernog

prilagođavanja (engl. *overfitting*) skupu za učenje ili nedovoljne kompleksnosti (engl. *underfitting*).

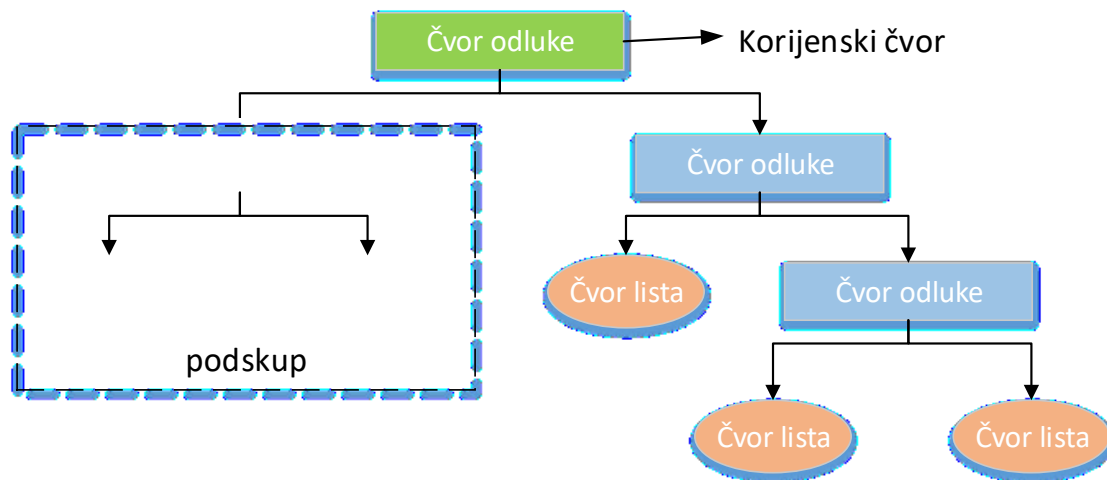
Klasifikacija se podataka može izvoditi kako s nestrukturiranim tako i sa strukturiranim skupovima podataka. Elementi izlaznog skupa podataka klasificiraju se prema nekoj danoj oznaci ili kategoriji. Neki pojmovi koji se često koriste u klasifikaciji prema [57] su:

- klasifikator - algoritam koji uči iz skupa za učenje i zatim dodjeljuje oznake (klase) novim ulaznim podacima
- klasifikacijski model - zaključuje valjano mapiranje iz skupa podataka za učenje i predviđa klasu uz pomoć funkcije mapiranja za novi unos podataka
- značajka ili svojstvo - parametar pronađen u zadanom skupu podataka koji može u dovoljnoj mjeri pomoći u izgradnji točnog prediktivnog modela.

Najpoznatiji i najučestaliji klasifikatori za otkrivanje anomalija, koji će biti detaljnije opisani u narednim poglavljima, a dio su *Scikit-learn*-biblioteke otvorenog koda za strojno učenje u programskom jeziku *Python*, uključuju: klasifikator stabla odluke, slučajnu šumu, stroj s potpornim vektorima, Naivni Bayesov klasifikator i klasifikator K-najbližih susjeda.

3.1.4. Klasifikator stabla odluke

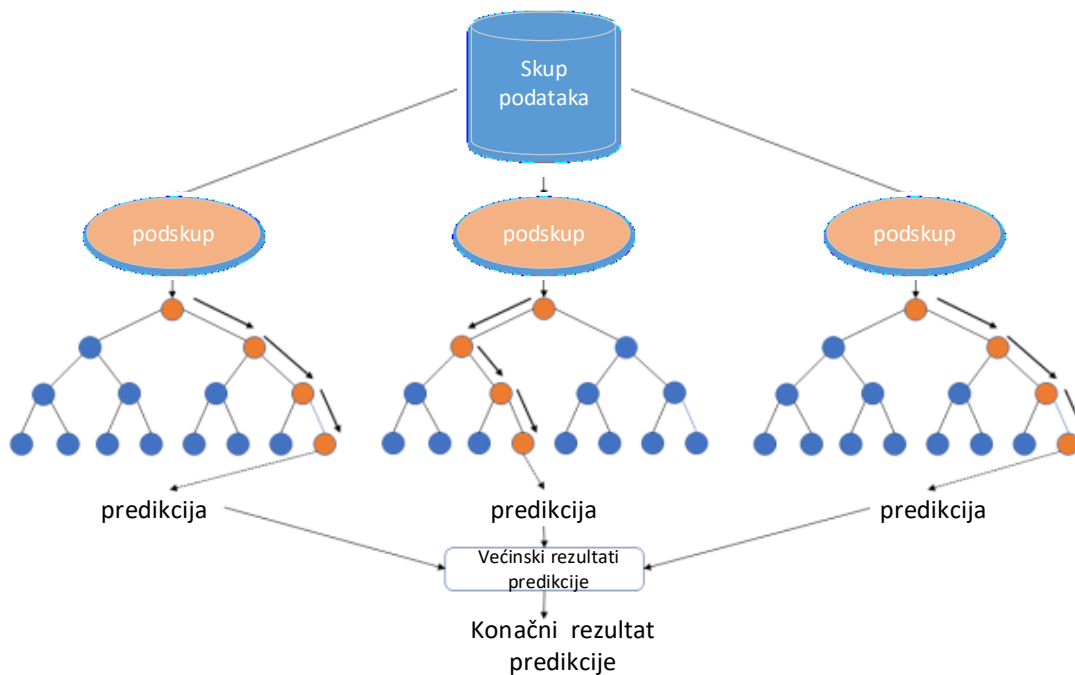
Stablo odluke (engl. *Decision Tree*) je klasifikator temeljen na logici u nadziranoj metodi strojnog učenja koji se koristi i za klasifikaciju i za regresiju. Cilj je stvoriti model koji predviđa vrijednost ili klasu izlazne varijable učenjem jednostavnih pravila odlučivanja zaključenih iz značajki skupa podataka [61]. Klasifikator stabla odluke razdvaja skup podataka na sve manje i manje podskupove temeljem različitih kriterija. Strukturiran je prema stablu, gdje unutarnji čvorovi predstavljaju značajke skupa podataka, grane predstavljaju pravila odluke, a svaki list predstavlja rezultat. Za podjelu skupa podataka koriste se različiti kriteriji razvrstavanja, a broj podataka će se smanjivati sa svakim dijeljenjem. U stablu odluke postoje dva čvora, a to su tzv. čvor odluke i čvor lista, kao što je prikazano slikom 3.3. Čvorovi odluka koriste se za donošenje bilo koje odluke i imaju više grana, dok su čvorovi lista izlazne vrijednosti tih odluka i ne sadrže daljnje grane. Stablo odluke klasificira elemente skupa podataka razvrstavajući ih "niz stablo" od korijenskog čvora do čvora lista [62], [63].



Slika 3.3. Princip klasifikacije stablom odluke [62]

3.1.5. Klasifikator slučajne šume

Klasifikator Slučajne šume sastoji se od više stabala odluke, svako izgrađeno na različitom podskupu skupa podataka, s ciljem poboljšanja točnosti predviđanja originalnog skupa podataka. Umjesto oslanjanja samo na jedno stablo odluke, taj klasifikator koristi predviđanja svakog stabla i zatim temeljem većine glasova donosi konačni rezultat predviđanja [55], [64].



Slika 3.4. Prikaz rada klasifikatora slučajne šume [65]

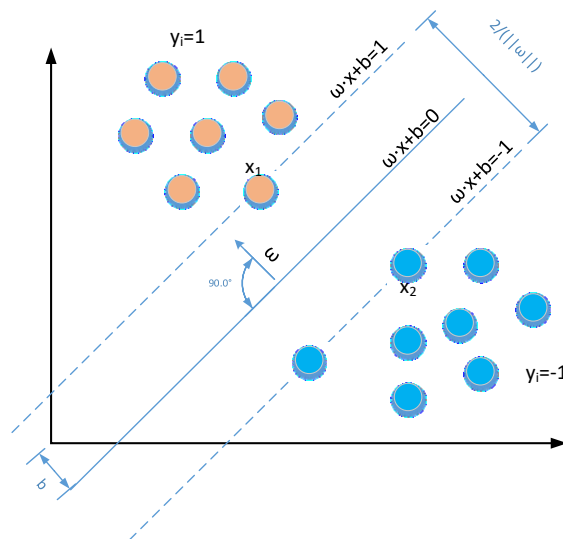
Klasifikator slučajne šume umjesto traženja najvažnije značajke tijekom cijepanja čvora traži najbolju značajku među slučajnim podskupom značajki. To rezultira velikom raznolikošću koja općenito rezultira boljim modelom, stoga se u tom klasifikatoru za razdvajanje čvora uzima u obzir samo slučajni podskup značajki. Još jedna izvrsna kvaliteta klasifikatora slučajne šume je ta što je vrlo lako izmjeriti relativnu važnost svake značajke u predviđanju. Na važnost značajke utječe koliko i koji čvorovi koriste tu značajku kako bi smanjili šum na svim ostalim dijelovima klasifikacije. Algoritam slučajne šume koristi metodu spajanja (engl. *bagging*) kako bi kombinirao jednostavnost stabla odlučivanja s fleksibilnošću, čime povećava točnost i prevladava loše svojstvo stabla odlučivanja koje može biti sklono prekomjernom prilagođavanju podacima za učenje [66].

U nizu radova i tehničkih izvješća [67], [68], [69], [70] pokazalo se da se znatna poboljšanja u klasifikaciji i točnosti regresije mogu postići korištenjem skupova stabala odluke, pri čemu se svako stablo u skupu konstruira u skladu sa slučajnim parametrom. Konačna predviđanja dobivena su agregacijom preko skupa stabala. Kako su stabla bazne sastavnice modela strukturiranih prediktora, a budući da je svako od tih stabala konstruirano korištenjem funkcije slučajnosti, te se procedure nazivaju "slučajne šume". Za tu ideju presudni utjecaj imao je rani rad Amita i Gemana [71] o izboru geometrijskih značajki, zatim metoda slučajnog potprostora u radu Hoa [72] i Dietterichov pristup slučajnog odabira [73]. Kao što je istaknuto raznim empirijskim studijama, algoritam slučajne šume pojavio se kao ozbiljni konkurent najsuvremenijim metodama. Brz je i jednostavan za implementaciju, daje vrlo točna predviđanja i može podnijeti vrlo velik broj ulaznih varijabli, stoga se smatra jednim od najpreciznijih dostupnih tehnika učenja opće namjene. U Breimanovom pristupu [74], svako stablo u zbirci oblikuje se s malom grupom ulaznih podataka, nasumično odabranih u svakom čvoru. Za svaki drugi čvor odabiru se značajke temeljem značajki iz skupa za učenje. Stablo se konstruira CART-ovom (engl. *Classification And Regression Trees*) metodom do maksimalne veličine [75]. Pretpostavlja se uzorak podataka za učenje $D_n = \{(X_1, Y_1), \dots, (X_n, Y_n)\}$ od identičnih i neovisnih slučajnih varijabli u prostoru $[0, 1]^d \times R$ slučajnih varijabli ($d \geq 2$) s istom distribucijom kao nezavisni generički par (X, Y) koji zadovoljava $EY^2 < \infty$. Za fiksni $x \in [0, 1]^d$, cilj je procijeniti regresijsku funkciju $r(x) = E[Y | X = x]$ korištenjem podataka D_n . U tom je smislu procjena regresijske funkcije r_n konzistentna ako $E[r_n(X) - r(X)]^2 \rightarrow 0$ kada $n \rightarrow \infty$. Formalno, slučajna šuma je prediktor koji se sastoji od kolekcije nasumičnih baznih regresijskih stabala $\{r_n(x, \Theta_m, D_n), m \geq 1\}$, gdje su $\Theta_1, \Theta_2, \dots, \Theta_m$ izlazne vrijednosti slučajne

varijable Θ . Ta se slučajna stabla kombiniraju kako bi se stvorila agregirana regresijska procjena $r_n(X, D_n) = E_{\Theta} [r_n(X, \Theta, D_n)]$, gdje E_{Θ} označava očekivanje s obzirom na slučajni parametar. Slučajna varijabla Θ koristi se za određivanje načina na koji se izvode uzastopni rezovi pri konstrukciji pojedinačnih stabala [76].

3.1.6. Klasifikator stroja s potpornim vektorima

Strojevi s potpornim vektorima (engl. *Support Vector Machines*) nadzirani je algoritam strojnog učenja koji se može koristiti u regresijskim i klasifikacijskim zadacima. Temelji se na procesu razdvajanja granica među različitim skupinama podataka radi njihovog grupiranja u određene kategorije ili klase. Uzorci s jedne strane linije bit će jedna klasa, a uzorci s druge strane pripadaju drugoj klasi. Klasifikator teži maksimiziranju udaljenosti među linijom i uzorkom s obje strane radi povećanja pouzdanosti u klasifikaciji uzoraka u određenu klasu. Koordinate se tih podataka zapravo nazivaju „potporni vektori”. U dvoklasnom problemu strojnog učenja, cilj je pronaći najbolju klasifikacijsku funkciju kako bi se razlikovali članovi dviju klasa u podacima koji se koriste za učenje. Koncept takve funkcije "najbolje" klasifikacije može se ostvariti geometrijski kao što je prikazano na slici 3.5., ali većina klasifikacijskih zadataka nisu jednostavni poput tog primjera i često su potrebne složene računske operacije kako bi se napravila optimalna podjela.



Slika 3.5. Prikaz klasifikacije strojem s potpornim vektorima za slučaj dviju klasa [77]

U slučaju linearno odvojivog skupa podataka, uloga je linearne funkcije klasifikacije maksimizirati razmak hiperravnina koje razdvajaju središta dviju klasa. U skupu podataka za učenje $D = \{(X_1, y_1), (X_2, y_2), \dots, (X_n, y_n)\}$ gdje je X_i karakterističan vektor uzoraka podataka za učenje, a y_i je pridružena oznaka (klasa) tom vektoru, može se reći da postoji linearna funkcija koja u potpunosti može odijeliti uzorke u dvije klase. Na slici 3.5. prikazan je linearno odvojiv skup podataka koji prikazuje ravna linija, no postoji bezbroj dodatnih linija kojima se može prikazati odjeljivanje. Optimalna linija klasifikacije može se predočiti jednadžbom [78]:

$$\omega \cdot x + b = 0 \quad (\omega \in R^n, b \in R) \quad (3-1)$$

gdje je ω vektor težina (normala) hiperravnine, a b je pomak hiperravnine. Uzorci iznad linije zadovoljavaju jednadžbu [78]:

$$\omega \cdot x + b > 0, \quad (3-2)$$

a ispod linije zadovoljavaju jednadžbu [78]:

$$\omega \cdot x + b < 0. \quad (3-3)$$

prema [78] granicu za klasu 1 možemo zapisati kao

$$H1: \omega \cdot x + b \geq 1 \text{ za } y_i = 1 \quad (3-4)$$

odnosno

$$H2: \omega \cdot x + b \leq -1 \text{ za } y_i = -1. \quad (3-5)$$

Prema tim jednadžbama najveća se udaljenost postiže najvećom vrijednosti $\frac{2}{\|\omega\|}$, odnosno izračunom najmanje vrijednosti za $\|\omega\|$ [57], [79], [80], [81].

3.1.7. Naivni Bayesov klasifikator

Bayesov klasifikator se u strojnom učenju može upotrijebiti za klasifikaciju dviju ili više klasa. Klasifikator uči analizirajući unaprijed pripremljen skup točno klasificiranih podataka za učenje i može se upotrijebiti za utvrđivanje vjerojatnosti klasa. U tom je modelu pretpostavka da su

značajke uvjetno nezavisne s obzirom na klasu. Temelj cijelog Bayesovog klasifikatora je Bayesov teorem koji uz pomoć jednadžbe [55]:

$$P(A|B) = P(B|A)P(A)/P(B) \quad (3-6)$$

izračunava uvjetnu vjerojatnost temeljenu na prethodnom znanju koji imamo o problemu. Taj klasifikator određuje vjerojatnost kojom određeni podatak pripada nekoj klasi, izračunavajući vjerojatnost da će se dogoditi neki događaj s obzirom na to da se dogodio neki ulazni događaj. Oznaka $P(A|B)$ predstavlja uvjetnu vjerojatnost pojavljivanja događaja, dok $P(A)$ predstavlja vjerojatnost pojavljivanja događaja A, a $P(B)$ predstavlja vjerojatnost pojavljivanja događaja B. Vjerojatnosti su klasa učestalost slučajeva koji pripadaju svakoj klasi podijeljeni s ukupnim brojem primjeraka. U binarnoj klasifikaciji, vjerojatnost uzorka koji pripada klasi 1 računa se prema jednadžbi [82]:

$$P(klasa = 1) = \frac{broj(klasa=1)}{broj(klasa=0)+broj(klasa=1)} \quad (3-7)$$

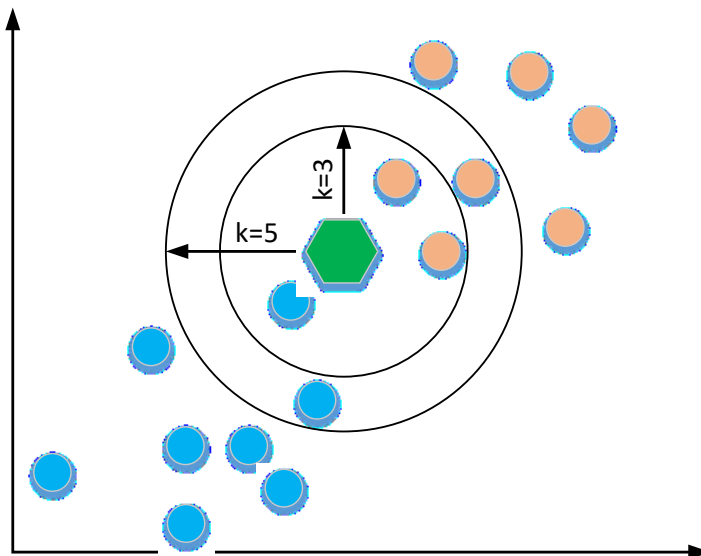
Naivni Bayesov algoritam klasifikacije može se generalizirati na rad s kontinuiranim vrijednostima tako da se najčešće pretpostavi Gaussova raspodjela. Ta pretpostavka omogućuje jednostavno procjenjivanje raspodjele podataka za učenje jer je potrebno samo izračunati srednju vrijednost i standardnu devijaciju podataka za učenje. Iako se za procjenu raspodjele mogu koristiti i druge funkcije, Gaussova (normalna) raspodjela najčešće je odabrana zbog svoje praktičnosti i jednostavnosti u ovom algoritmu. Prema [55] postoje tri vrste Naivnog Bayesova modela:

- Gaussov klasifikator - pretpostavlja da obilježja skupa podataka slijede normalnu raspodjelu. To znači da ako klasifikatori uzimaju kontinuirane vrijednosti umjesto diskretnih, tada model pretpostavlja da su te vrijednosti uzorkovane iz Gaussove raspodjele.
- Multinomialni naivni Bayesov klasifikator koristi se kada su podatci multinomialno distribuirani. Primarno se koristi za probleme s klasifikacijom dokumenata gdje klasifikator koristi učestalost pojavljivanja riječi.
- Bernoullijev klasifikator djeluje slično multinomialnom klasifikatoru, ali prediktorske su varijable neovisne boolean varijable.

Naivni Bayesov model privlači pažnju zbog svoje jednostavnosti i pouzdanosti. Taj je algoritam jedna od najstarijih metoda formalne klasifikacije, ali čak i u svom osnovnom obliku pokazuje iznenađujuću učinkovitost [63][83].

3.1.8. Klasifikator K-najbližeg susjeda

K-najbliži susjed je nadzirani algoritam učenja gdje se rezultat klasificira temeljem većinske klase K-najbližeg susjeda. Taj je klasifikator jednostavan i zasniva se na minimalnoj udaljenosti od promatranog uzorka do uzoraka podatka za učenje kako bi se utvrdili najbliži susjedi. Nakon što se pronađu najbliži susjedi, uzima se klasa većine najbližih susjeda K kako bi se predvidjela klasa promatranog uzorka kao što je prikazano na slici 3.6. Klasa promatranog uzorka označenog zelenom bojom predviđa se temeljem udaljenosti do najbližih susjeda.



Slika 3.6. Klasifikacija uzorka algoritmom K-najbližeg susjeda [84]

Obično se uzima euklidska udaljenost kao metrika za mjerenje udaljenosti od susjeda. Euklidska udaljenost među dvama vektorima A i B , gdje je $A = (x_1, x_2, x_3, \dots, x_n)$ i $B = (y_1, y_2, y_3, \dots, y_n)$ dana jednadžbom [85] :

$$d(A, B) = \sqrt{\sum_{i=1}^m (x_i - y_i)^2} \quad (3-8)$$

Ne postoji način za određivanje najbolje vrijednosti koeficijenta K , stoga treba isprobati različite vrijednosti kako bi se pronašao najbolji rezultat [85], [86]. Uobičajena vrijednost za K je 5, ali važno je obratiti pažnju na nekoliko stvari [57]:

- Smanjenjem vrijednosti K , predviđanja postaju manje sigurna jer se temelje na manjem broju susjeda.
- Povećanjem vrijednosti K , predviđanja postaju sigurnija jer se temelje na usrednjavanju većine susjeda, što često rezultira preciznijim predviđanjima, ali se zahtijeva više računalnih resursa jer model mora uzeti u obzir više susjeda pri donošenju odluka.
- Ako se počne pojavljivati veći broj pogrešaka u predviđanjima, to može ukazivati da je vrijednost K prešla optimalnu granicu.
- U slučajevima klasifikacije, obično se uzima neparan broj za K kako bi se izbjegao neriješen rezultat i nemogućnost određivanja klasifikacije u slučaju podjednakog broja susjeda iz različitih klasa.

3.2. Mjere uspješnosti klasifikacije

Uspješnost algoritma strojnog učenja procjenjuje se metričkim vrijednostima. Problem ocjenjivanja klasifikatora (tj. mjerenja njegove kvalitete) rješava se jedinstvenom ocjenom koja pokušava sažeti posebne uvjete i zanimanja pri ocjenjivanju [87]. Postoji mnogo načina ocjenjivanja klasifikatora, a u problemu klasifikacije najčešće se koriste sljedeće vrste mjernih podataka:

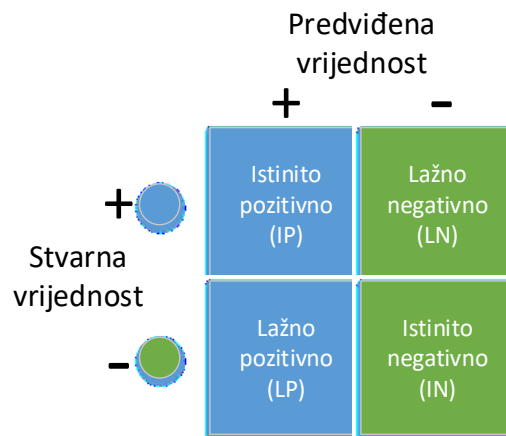
- matrica konfuzije (engl. *Confusion Matrix*)
- mjera točnosti klasifikacije (engl. *Accuracy*)
- mjera uravnotežene točnosti (engl. *Balanced Accuracy*)
- mjera preciznosti (engl. *Precision*)
- mjera opoziva (engl. *Recall*)
- F mjera (engl. *F Score*)

- područje ispod krivulje (engl. *Area under Curve*)
- Matthewsov koeficijent korelacije (engl. *Matthews correlation coefficient*)
- Cohen kappa-koeficijent (engl. *Cohen's kappa coefficient*).

3.2.1. Matrica konfuzije

Matrica konfuzije prikazuje slučajeve istinitih i lažnih predviđanja dobivenih na osnovu poznatih ulaznih podataka. Binarni klasifikacijski model uključuje četiri moguća scenarija rezultata koje dobivamo kroz usporedbu predviđenih i stvarnih vrijednosti u dvjema klasama: pozitivnu i negativnu. Rezultat je izlazna vrijednost u obliku matrice i opisuje performanse modela s četirima mogućim ishodima koja se različito definiraju. Kada se predvidi pozitivan rezultat i dobije pozitivan rezultat, to nazivamo istinito pozitivni rezultat (IP). Kada se predvidi pozitivan rezultat, a dobije negativan rezultat, to nazivamo lažno pozitivnim rezultatom (LP), poznat i kao pogreška tipa 1. Kada se predvidi negativan rezultat, a dobije pozitivan rezultat, to nazivamo lažno negativnim rezultatom (LN), poznat i kao pogreška tipa 2. Četvrti je mogući ishod kada se predvidi negativan rezultat i dobije negativan rezultat, to nazivamo istinito negativan rezultat (IN).

Na slici 3.7., plavi i zeleni krugovi označavaju uzorke za koje se zna da su pozitivni (IP + LN), odnosno negativni (LP + IN), a plava i zelena pozadina/kvadrati prikazuju predviđene pozitivne rezultate (IP + LP) i negativne (LN + IN) rezultate. Matrica konfuzije čini osnovu za druge vrste mjernih podataka koje su opisane u nastavku.



Slika 3.7. Matrica konfuzije za slučaj binarne klasifikacije

3.2.2. Mjera točnosti i pogreške klasifikacije

Tipične su ocjene za mjerenje performansi klasifikatora točnost i klasifikacijska pogreška, koja se za binarni problem može lako izvesti iz matrice konfuzije dimenzije 2 x 2 kao one prikazane na slici 3.7. Mjera je točnosti omjer broja točnih predviđanja i ukupnog broja ulaznih uzoraka i izračunava se prema jednadžbama [87]:

$$\text{točnost} = \frac{\text{broj točnih predviđanja}}{\text{broj ukupnih predviđanja}} = \frac{IP+IN}{IP+LN+IN+LP} \quad (3-9)$$

$$\text{pogreška} = \frac{\text{broj lažnih predviđanja}}{\text{broj ukupnih predviđanja}} = \frac{LP+LN}{IP+LN+IN+LP} \quad (3-10)$$

Te vrijednosti točnosti i pogreške djeluju dobro samo ako postoji jednak broj uzoraka koji pripadaju svakoj klasi. Na žalost, u stvarnim problemima jednake su proporcije uzoraka po klasi prilično rijetke te mjerne vrijednosti mogu dati lažni osjećaj postizanja visoke točnosti. Takva je situacija poznata kao problem nebalansiranosti uzoraka. Empirijski je dokazano da su točnost i stopa pogrešaka pristrani s obzirom na nebalansiranost uzoraka, odnosno ne uzimaju se u obzir slučajevi pogrešnih klasifikacija. Rezultati su snažno pristrani na način da podržavaju klasu koja je u većini i osjetljivi su na tzv. iskrivljenost klasa. Ti mjerni parametri za problem klasifikacije nisu najprikladniji ako se uzimaju bez dodatnog razmatranja [87]. Učestalost većinske klase ima veliki utjecaj na izračun točnosti, dok manje zastupljene klase imaju manji utjecaj, stoga je bitno odabrati odgovarajuću metriku evaluacije koja će pridavati veći značaj manjinskoj klasi. Precizno predviđanje što većeg broja slučajeva manjinske klase ključno je za postizanje boljeg rješenja, posebno u situacijama s vrlo nebalansiranim uzorcima [88].

3.2.3. Uravnotežena točnost

Uravnotežena točnost sprječava pretjerane procjene točnosti na nebalansiranim skupovima podataka. Izračunava se kao prosječna vrijednost mjere osjetljivosti (opoziva) za svaku klasu, stoga je za balansirane skupove podataka rezultat uravnotežene točnosti jednak klasičnoj točnosti. U binarnom problemu, uravnotežena točnost jednaka je aritmetičkoj sredini osjetljivosti (stopi istinito pozitivnih predviđanja) i specifičnosti (stopi istinito negativnih predviđanja) ili površini ispod ROC-krivulje (engl. *Receiver Operating Characteristic*) s binarnim predviđanjima [89].

$$\text{uravnotežena točnost} = \frac{1}{2} \left(\frac{IP}{IP+LN} + \frac{IN}{IN+LP} \right) \quad (3-11)$$

U [90] autori objašnjavaju nedostatke klasične točnosti (nebalansiran skup podataka i nemogućnost izračunavanja smislenog intervala pouzdanosti na temelju broja uzoraka) i prednosti zamjene s mjerom uravnotežene točnosti.

3.2.4. Mjera preciznosti

Broj istinito pozitivnih rezultata podijeljen s brojem pozitivnih rezultata koje je predvidio klasifikator, naziva se mjerom preciznosti i izračunava se prema [91]:

$$\text{preciznost} = \frac{IP}{IP+LP} \quad (3-12)$$

Mjera preciznosti prikladna je kada je fokus strojnog učenja na smanjivanju lažno pozitivnih predviđanja [91].

3.2.5. Mjera opoziva

Broj istinito pozitivnih rezultata podijeljen s brojem svih relevantnih uzoraka (svi uzorci koji bi trebali biti identificirani kao pozitivni), naziva se mjerom opoziva i izračunava se prema [91]:

$$\text{opoziv} = \frac{IP}{IP+LN} \quad (3-13)$$

Mjera opoziva prikladna je kada je fokus strojnog učenja na smanjivanju lažno negativnih predviđanja [91].

3.2.6. F-mjera i G-mjera

Nedostatci mjere točnosti i pogreške rezultirali su novim mjernim vrijednostima kojima se želi postići kompromis u klasifikaciji i na pozitivnim i na negativnim uzorcima podataka. Jedna od takvih alternativnih ocjena su harmonijska i geometrijska sredina izmjerena odvojeno za svaku klasu. Mjere harmonijske i geometrijske sredine daju jednaku važnost svim klasama i korisne su jer uzimaju u obzir oba vida performansi modela, preciznost i osjetljivost, što ih čini prikladnim za evaluaciju modela s nebalansiranim klasama [87].

F-mjera predstavlja harmonijsku sredinu među vrijednostima opoziva i preciznosti u rasponu među vrijednostima 0 i 1 te pokazuje koliko je precizan klasifikator (koliko se uzoraka ispravno klasificira), kao i koliko je robustan (ne griješi u klasifikaciji u značajnom broju uzoraka). Što je veći F-rezultat, to su bolje performanse modela. Matematički se to može izraziti kao [91]:

$$F = 2 * \frac{\text{preciznost} * \text{opoziv}}{\text{preciznost} + \text{opoziv}} \quad (3-14)$$

Geometrijska se sredina koristi za maksimiziranje stope IP i IN, a istovremeno zadržavajući obje stope relativno uravnotežene. Matematički se može izraziti kao [88]:

$$G = \sqrt{IP * IN} \quad (3-15)$$

F-mjere i G-mjere pokazale su se kao bolji parametri nego mjera točnosti u optimizaciji klasifikatora za binarne probleme klasifikacije [88]. F-mjere i G-mjere daju optimalne rezultate kada dosegnu vrijednost 1, dok postižu najslabije rezultate kada su vrijednosti 0, što je slično kao kod mjera preciznosti i osjetljivosti. F-mjera kombinira preciznost i osjetljivost na uravnotežen način. U nekim slučajevima, potrebna je F-mjera koja će više naglasiti preciznost, npr. kada je cilj minimizirati lažno pozitivne predikcije, dok su lažno negativne predikcije i dalje bitne. U drugim slučajevima, potrebna je F-mjeru usredotočena na opoziv, npr. kada je važnije minimizirati lažno negativne predikcije, ali lažno pozitivne predikcije su i dalje važne. F β -mjera (Fbeta) je rješenje za takve probleme, a izvodi se iz F mjere, pri čemu se koeficijent β koristi za kontroliranje ravnoteže među preciznosti i opozivom [91]:

$$F\beta = \frac{(1 + \beta^2) * \text{preciznost} * \text{opoziv}}{(\beta^2 * \text{preciznost}) + \text{opoziv}} \quad (3-16)$$

Tri su uobičajene vrijednosti za β -parametar prema [91]:

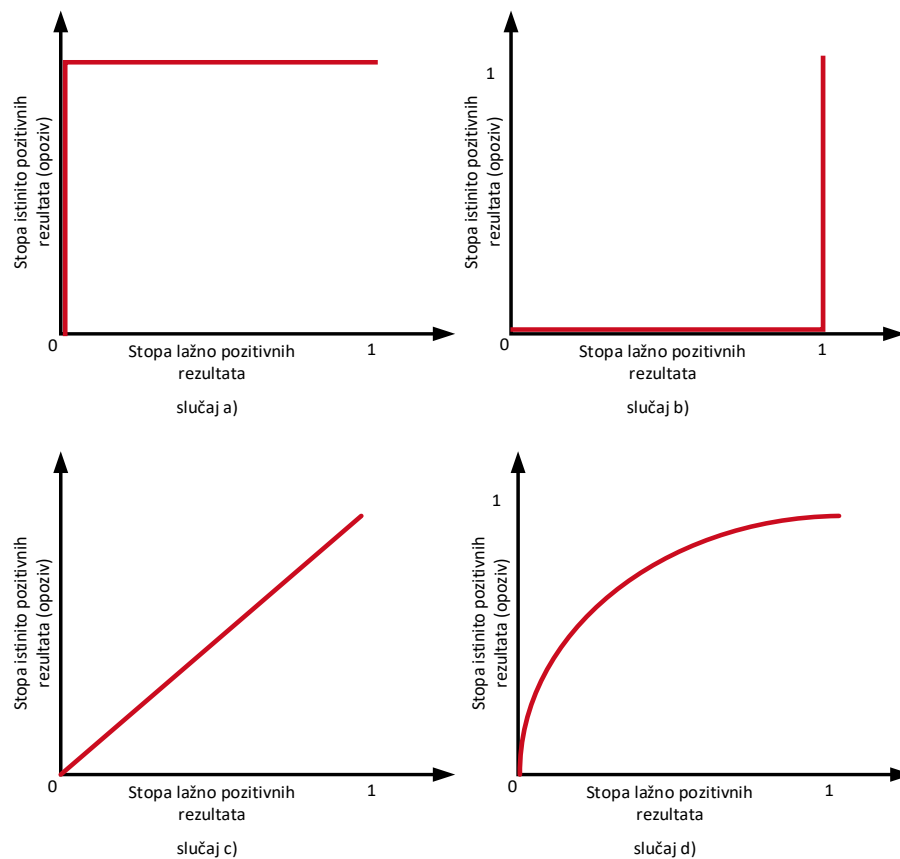
- F0.5 mjera ($\beta = 0,5$): veći naglasak na preciznosti, manji na opozivu
- F1 mjera ($\beta = 1$): jednak naglasak na preciznosti i opozivu
- F2 mjera ($\beta = 2$): veći naglasak na opozivu, manji na preciznost.

3.2.7. Područje ispod krivulje (AUC)

Jedna od najčešće korištene tehnike za ocjenu klasifikatora je ROC-krivulja (engl. *Receiver Operating Characteristic*), koja je grafički prikaz mjere opoziva u odnosu na stopu lažno pozitivnih rezultata [91]:

$$\text{stopa lažno pozitivnih rezultata} = \frac{LP}{LP+IN} \quad (3-17)$$

Informacija o izvedbi klasifikacije u ROC-krivulji, može se sažeti u rezultat poznat kao AUC, odnosno područje ispod ROC-krivulje. Ta je mjera neosjetljivija na neravnomjernost u raspodjeli klasa i predstavlja kompromis među opozivom i stopama lažno pozitivnih rezultata [87]. Ta mjera pokazuje koliko je model sposoban razlikovati klase. Što je veći AUC, to je model bolji u predviđanju stvarno pozitivnih uzoraka kao pozitivnih rezultata i obrnuto, predviđanju stvarno negativnih uzoraka kao negativnih rezultata.



Slika 3.8. Tipične ROC-krivulje [92]

AUC-vrijednosti se kreću među 0 i 1, a tipični su slučajevi prikazani na slici 3.8. Izvrсни model ima AUC blizu vrijednosti 1 (slučaj a.), što znači da može dobro odvojiti klase. Loš model ima AUC blizu vrijednosti 0 (slučaj b.), što znači da jako loše razdvaja klase, odnosno predviđa suprotno od stvarnih vrijednosti. Za model koji uopće nema sposobnost odvajanja klasa, AUC-mjera je jednaka vrijednosti 0,5 (slučaj c.), tipična ROC-krivulja ima oblik prikazan na slici 3.8. za slučaj d.

U ROC-krivulji, veća vrijednost na osi X označava veći broj lažno pozitivnih, nego istinitih negativnih predviđanja, dok veća vrijednost na osi Y ukazuje na veći broj istinito pozitivnih nego lažno negativnih predviđanja. Dakle, izbor praga ovisi o željenom odnosu i ravnoteži među lažno pozitivnim i lažno negativnim predviđanjima [93]. Blizina ROC-krivulje gornjem lijevom kutu na slici 3.8. sugerira da klasifikator ima visoke performanse u razlikovanju među pozitivnim i negativnim uzorcima. AUC predstavlja vjerojatnost da će klasifikator ispravno razlikovati pozitivne i negativne uzorke, a računa se kao površina ispod ROC-krivulje. Drugim riječima, AUC

je mjera koja određuje koliko je vjerojatno da će klasifikator dati bolji rezultat za pozitivne uzorke u odnosu na negativne uzorke [94]. Dobra ROC-krivulja zatvara veću površinu, dok loša ROC-krivulja zatvara manju površinu, što znači da je visoka AUC-vrijednost poželjna, a niska AUC-vrijednost nije poželjna.

Prema [93], prednosti su za korištenje AUC-mjere kod odabira klasifikatora:

- djeluje neovisno s obzirom na raspodjelu klasa
- pruža dobru skalarnu mjeru rangiranja klasifikatora.

Kada se radi s nebalansiranim skupovima podataka, upotrebi se AUC-mjere često daje prednost u istraživanjima. Neravnoteža klasa može otežati otkrivanje manjinske klase i stvoriti pristranost u korist većinske klase, što može utjecati na točnost klasifikacije. Klasična mjera točnosti nije uvijek pouzdana jer može dati varljivo visok rezultat kada je manjinska klasa u velikoj mjeri smanjena, stoga se preporučuje korištenje AUC-mjere s drugim metrikama poput preciznosti i opoziva. Neka istraživanja [95], [96] su pokazala da je korištenje AUC-mjere kao samostalne i robusne metrike kod neravnoteže u klasama podataka vrlo učinkovito.

3.2.8. Matthewsov koeficijent korelacije

Matthewsov je koeficijent korelacije (MCC engl. *Matthews correlation coefficient*) koristan u strojnom učenju za procjenu kvalitete binarnih klasifikacija. On uzima u obzir stvarne pozitivne, lažne pozitivne, stvarne negativne i lažne negativne rezultate predviđanja i obično se koristi kao mjerilo koje se može primijeniti čak i kad je veličina klasa vrlo nejednaka. Vrijednost MMC-a kreće se među -1 i 1, pri čemu koeficijent 1 označava savršeno predviđanje, koeficijent 0 prosječno slučajno predviđanje, a koeficijent -1 inverzno predviđanje. U statistici, MMC je također poznat kao phi-koeficijent. U binarnom (dvoklasnom) slučaju MCC-koeficijent se prema [89] računa jednadžbom :

$$MCC = \frac{IP*IN-LP*LN}{\sqrt{(IP+LP)*(IP+LN)*(IN+LP)*(IN+LN)}} \quad (3-18)$$

3.2.9. Cohen kappa-koeficijent

Cohen kappa-koeficijent se, poput drugih metrika za evaluaciju strojnog učenja, temelji na matrici konfuzije, ali uzima u obzir neravnotežu u raspodjeli klasa. Rezultat Cohen kappa-koeficijenta je među -1 i 1, a vrijednost iznad 0,8 obično se smatra dobrim slaganjem između predviđenih i stvarnih vrijednosti. Ta se metrika može koristiti za usporedbu predviđanja modela strojnog učenja s ručno utvrđenim klasama skupa podataka. Cohenov kappa-koeficijent izračunava se prema [97]:

$$\kappa = \frac{P_0 - P_e}{1 - P_e} \quad (3-19)$$

gdje je P_0 ukupna točnost modela, P_e je mjera slaganja između predviđanja modela i stvarnih vrijednosti klase kao da su slučajno određeni. U binarnom problemu klasifikacije, P_e je suma P_{e1} (vjerojatnost da se slučajna predviđanja slažu sa stvarnim vrijednostima klase 1) i P_{e2} (vjerojatnost da se slučajna predviđanja slažu sa stvarnim vrijednostima klase 2). Cohen kappa-koeficijent se koristi u svrhu isključivanja utjecaja nasumičnog pogađanja na točnost klasifikacije i mjeri broj predviđanja koji se ne mogu pripisati slučajnosti. Osim toga, Cohen kappa-koeficijent koristi se kako bi ispravio pristranost u predviđanju, uzimajući u obzir točne klasifikacije koje se mogu objasniti samo slučajnim pogađanjem [97].

3.3. Pregled primjene strojnog učenja u SDN-mrežama

Istraživanjem literature ustanovljeno je da korištenje strojnog učenja i programski definirane mreže pružaju obećavajuće rješenje za otkrivanje upada. Napredak u razini točnosti, smanjenje potrebnog vremena za učenje i izvođenje, može se još dodatno poboljšati, posebno kako bi se implementiralo u okruženje programski definirane mreže. U nastavku je pregled recentnih provedenih istraživanja koja su povezana s primjenom strojnog učenja u SDN-mrežama.

U radu [41] predstavljeni su dizajn i evaluacija dinamičkog klasifikatora strojnog učenja za mrežu temeljenu na SDN-u. Korištenjem triju ML-tehnika klasificiraju se TCP i UDP tokovi s 3, 5 i 7 parametara uz postizanje visokih točnosti klasifikacije. Također, predstavljen je algoritam sposoban za kombiniranje različitih ML-modela i povećanje točnosti. U svrhu evaluacije

performansi predloženi je sustav implementiran u Ryu-kontroleru, isključivši potrebu ručnog usklađivanja dolaznog prometa i osiguravajući bolji QoS. Rezultati eksperimenata pokazuju značajno smanjenje latencije u analizi prometa.

U radu [42] je predstavljen koncept odabira i obrade tokova radi identifikacije i klasifikacije neovlaštenih aktivnosti u SDN-mrežama. Također je pokazano da odgovarajući odabir i obrada značajki toka omogućuju učinkovitu klasifikaciju mrežnog prometa. Rezultati eksperimenata potvrdili su opravdanost korištenja algoritama za smanjenje broja značajki u predloženom sustavu. Za klasifikaciju SDN-tokova korištena je metoda stroja s potpornim vektorima (SVM). Postupak testiranja bio je usmjeren na postizanje najviše vrijednosti istinito pozitivnih rezultata te s najmanjim brojem značajki i najnižom vrijednosti vremena izvršavanja procesa.

U radu [43] je predložen R-IDPS sustav koji koristi SDN kako bi postigao potpuno autonomno otkrivanje i sprečavanje upada u IoT-sustavima. Dio istraživanja prikazuje evaluaciju performansi centraliziranog IDPS-sustava u usporedbi s distribuiranim IDPS-sustavima u kontekstu IoT-mreža. Integracijom agenata, SDN-a i tehnike strojnog učenja, pokazana je snaga distribuiranog dizajna u smislu učinkovitosti i performansi. Istraživanje pokazuje da SVM-algoritam koristi manje resursa i vrlo je pogodan za uređaje s niskom potrošnjom energije kao što su IoT-uređaji. Za učenje se koristi interni skup podataka sustava. Predloženi dizajn štiti IoT-uređaje od nadolazećih prijetnji bez dužih perioda ispadanja. Performanse otkrivanja sustava protiv DDoS-napada pokazuju izvrsnu preciznost te niske stope lažno pozitivnih i lažno negativnih rezultata.

U radu [44] je predložen sustav za otkrivanje i sprječavanje DDoS-napada u SDN sustavu. U procesu otkrivanja kibernetičkog napada korišten je stroj s potpornim vektorima (SVM) kao klasifikator. Radi bolje preciznosti i smanjenja vremena testiranja, u promatranom modelu primijenjene su KPCA-tehnika (Analiza jezgrenih glavnih komponenata) i GA-tehnika (Generički algoritam), koje služe za izdvajanje glavnih značajki iz skupa podataka s DDoS-napadima i za odabir odgovarajućih parametara za SVM-klasifikator. Eksperimentalni rezultati pokazuju da na promatranom skupu podataka, KPCA ima najbolje performanse, a točnost predloženog modela je bolja od nekih postojećih modela.

U radu [45] su predloženi nadzirani algoritmi strojnog učenja za otkrivanje DDoS-napada u trima tipovima arhitekture mreže: pojedinačna topologija, linearna topologija i više-kontrolerska topologija. Modeli su trenirani na skupovima podataka u generiranom SDN-okruženju emulatorom Mininet i kontrolerom RYU. Rezultati simulacije pokazuju da se pojedini algoritmi poput RF-a (engl. *Random forest*) i KNN-a (engl. *K-Nearest Neighbours*) mogu učinkovito koristiti u modelima predviđanja kibernetičkih napada.

U radu [46] se proučavaju i primjenjuju tehnike nadziranog učenja kako bi pomogle u otkrivanju DDoS-napada u SDN-mreži. Skup podataka za učenje prikupljen je iz eksperimentalnih skupova podataka i generiranog prometa u SDN-u. Različite vrste skupova za učenje testirane su kako bi se vidio utjecaj na rezultate otkrivanja kibernetičkih napada. Uspoređene su performanse svakog korištenog algoritma nadziranog učenja i validirane su metrike u otkrivanju, kao i potrošnja CPU-a i vrijeme odziva. Rezultati simulacije pokazuju da se izvedbe pojedine tehnike nadziranog učenja znatno razlikuju jedna od druge pod istim scenarijem testiranja. Implementacija predloženog rješenja pokazuje da otkrivanje DDoS-napada u stvarnom vremenu korištenjem tehnika nadziranog učenja u SDN-u ima zadovoljavajuću točnost. Ograničenje predloženog sustava jest da je pogodan samo za DDoS-napade temeljene na preplavlivanju, ne može otkriti ne-volumetrijske napade, poput niskorazinskih DDoS-napada i razmatra samo napade usmjerene na kontrolni sloj SDN-mrežne arhitekture.

U radu [47] je predstavljena implementacija modularne i fleksibilne SDN-bazirane arhitekture za otkrivanje transportnih i aplikacijskih DDoS-napada korištenjem više ML-modela i dubokog učenja (engl. *Deep Learning*). Istraživanje različitih ML/DL-modela omogućilo je utvrđivanje koje se metode bolje izvode pod različitim vrstama napada i uvjetima. Testirani su ML/DL-modeli koristeći dva skupa podataka: CICDoS2017 i CICDDoS2019 koji su pokazali visoku točnost pri klasifikaciji testnog prometa. Također je implementirano simulirano okruženje koristeći mrežni emulator Mininet i SDN kontroler ONOS (engl. *Open Network Operating System*).

Rad [48] istražuje poboljšanje mrežne sigurnosti u OpenStack-oblacima integracijom SDN-a, NFV-a (engl. *Network Function Virtualization*) i ML/AI-a (engl. *Machine Learning/Artificial Intelligence*). Predložen je okvir OpenStackDP koji se sastoji od senzora za praćenje i otkrivanje anomalija implementiranih u podatkovni sloj mrežne arhitekture, modula za

kibernetičke prijetnje koji koristi ML/AI-algoritme za otkrivanje prijetnji u stvarnom vremenu te virtualnih mrežnih funkcija (VNFs) za brzo ublažavanje otkrivenih prijetnji. Testiranja su pokazala visoku točnost u otkrivanju anomalija (99,81 %), nisku latenciju i brzo vrijeme odgovora. Rad također nudi detaljnu analizu ranjivosti OpenStack-mreže te minimalističku izvedbu IDS-a i sigurnosni orkestracijski okvir za implementaciju sigurnosnih i kontrolnih funkcija u podatkovnom sloju. Zaključeno je da OpenStackDP značajno poboljšava sigurnost i izvedbe infrastrukture u oblaku, omogućujući brži oporavak od kibernetičkih napada i veću pouzdanost mreže.

U istraživanju [49] je osmišljen novi model za učinkovito otkrivanje upada u SDN-okruženju Internet-stvari (IoT). Predstavljeni model slijedi trostupanjski proces u kojem se tehnika Harmony Search-algoritma koristi za odabir značajki. Zatim se metoda konvolucijskog autoenkodera koristi za otkrivanje i klasifikaciju upada u SDN-okruženju IoT-a te se provodi postupak podešavanja hiperparametara kako bi se poboljšala učinkovitost otkrivanja upada. Uspješnost predloženog modela u otkrivanju upada potvrđena je usporedbom s relevantnim prethodnim istraživanjima.

U radu [50] je predložena hijerarhijska višeklasna klasifikacija za simulirane scenarije DDoS-napada kako bi se provjerila sposobnost otkrivanja kibernetičkih napada. Istraživanje je provedeno kako bi se poboljšala sigurnost SDN-mrežne okoline korištenjem strojnog učenja i mehanizama predviđanja te ublažavanja posljedica kibernetičkih napada. Istraživanje uključuje konstrukciju SDN-topologije mreže, učenje ML-modela i DL-modela kako bi se analizirala njihova sposobnost identifikacije anomalija te poboljšanje izvedbe klasifikacije nebalansiranih skupova podataka, posebno za manjinske klase.

U istraživanju [51] su prikupljeni podaci mrežnih tokova prometa te je izgrađen učinkovit model strojnog učenja RF-algoritom. Model u stvarnom vremenu otkriva DDoS-napad analizom toka mrežnog prometa. Uspješno je provedeno otkrivanje i prevencija kibernetičkog napada zaustavljanjem aplikacije koju koristi tzv. zombi-klijent za izvođenje napada, dok se drugim ispravnim aplikacijama omogućuje djelovanje bez prekida.

Tablica 3.1. Pregled istraživanja upotrebe strojnog učenja u SDN-mrežnim okruženjima

#	god.	skupovi	okolina	ML-algoritmi	SV	problem istraživanja
[41]	2021.	✘	S	SVM, RF, LR	✓	klasifikacija mrežnog prometa u SDN-okruženju
[42]	2018.	✘	RT	LR	✘	praćenje i otkrivanje zlonamjernih aktivnosti u SDN-mrežama
[43]	2021.	✘	S	SVM	✓	obrana od kibernetičkih napada u IoT-mreži
[44]	2020.	1	S	SVM	✘	otkrivanje i sprječavanje DDoS-napada u SDN-mreži
[45]	2022.	✘	S	RF, KNN, NB, LR	✓	otkrivanje i sprječavanje DDoS-napada u SDN-mreži
[46]	2022.	✘	S & RT	KNN, BT, SVM, DT	✓	otkrivanje i sprječavanje DDoS-napada u SDN-mreži
[47]	2021.	2,3	S	KNN, SVM, RF, MLP, CNN, GRU, LSTM	✘	otkrivanje i sprječavanje DDoS-napada u SDN-mreži
[48]	2023.	2,4,5	S	nenadzirani algoritmi strojnog učenja	✘	sigurnost SDN-bazirane infrastrukture u oblaku
[49]	2023.	✘	S	neuronske mreže	✘	otkrivanje upada dubokim učenjem u SDN IoT-okruženju
[50]	2022.	6,7	S	DT, RF, KNN, NB, SVM, AdaBoost, MLP, CNN, RNN, LSTM, GRU	✘	otkrivanje i sprječavanje DDoS-napada u SDN-mreži
[51]	2022.	✘	S	RF	✓	otkrivanje i sprječavanje DDoS-napada u SDN-mreži

Legenda kratica u tablici 3.1.:

1 - NSL-KDD , 2 - CICDoS2017, 3 - CICDDoS2019, 4 - CICIDS2018, 5 - CICIDS2019, 6 - DDoS-SDN, 7 – InSDN, S – simulacijska testna okolina, RT – stvarna testna okolina, SV – korištenje modela u stvarnom vremenu

U tablici 3.1. prikazan je pregled literature i istraživanja u području sigurnosti mreža, posebno usmjereno na integraciju programski definirane mreže (SDN) i strojnog učenja (ML), a pokazuje stalni interes i razvoj unazad nekoliko godina. Svi pregledani radovi naglašavaju važnost SDN-a kao napredne mrežne paradigme, često koristeći javno dostupne skupove podataka za učenje kako bi proučavali različite scenarije kibernetičkih napada. ML-algoritmi, poput *Support Vector Machine* (SVM), *Random Forest* (RF), *Logistic Regression* (LR), *K-Nearest Neighbors* (KNN) i neuronske mreže, primijenjeni su za otkrivanje i sprječavanje napada, pružajući raznolik pristup sigurnosnim izazovima. Većina radova koristi simulacijske testne okoline omogućujući kombinaciju kontroliranih laboratorijskih uvjeta s realnim scenarijima. Simulacije uključuju različite vrste napada, a najčešće DDoS-kibernetičke napade i testiraju različite ML-algoritme u virtualnom okruženju. Unatoč tomu, samo nekoliko istraživanja provodi testiranje u stvarnom vremenu, što sugerira potrebu za daljnjim ispitivanjem predloženih pristupa kako bi se potvrdila njihova praktična primjenjivost u stvarnom vremenu. Glavno središte istraživanja usmjereno je na otkrivanje i sprječavanje različitih vrsta kibernetičkih napada, uključujući i DoS, DDoS-napade u IoT-mrežama. Neki radovi također istražuju sigurnost SDN-bazirane infrastrukture u oblaku te primjenu ML-a za otkrivanje upada u SDN IoT-okruženju. Promatrani rezultati ukazuju na potrebu za daljnjim istraživanjem kako bi se razvile učinkovite i pouzdane metode za očuvanje sigurnosti u modernim mrežama koje koriste SDN-tehnologije.

Mininet-emulator često se koristi za simulaciju virtualnih preklopnika, posebice Open Virtual Switch (OVS), unutar SDN-okoline uz korištenje OpenFlow-protokola. Važno je napomenuti određena ograničenja povezana s prilagodbom preklopnika i dodavanjem aplikacija na računalne čvorove u takvoj simulacijskoj okolini. Ta simulacijska okolina, iako pruža značajne prednosti za ispitivanje mrežnih koncepta, može predstavljati izazove kada je riječ o implementaciji posebnih funkcionalnosti i prilagodbi.

4. PRIJEDLOG RJEŠENJA ZA OTKRIVANJE ANOMALIJA MREŽNOG PROMETA

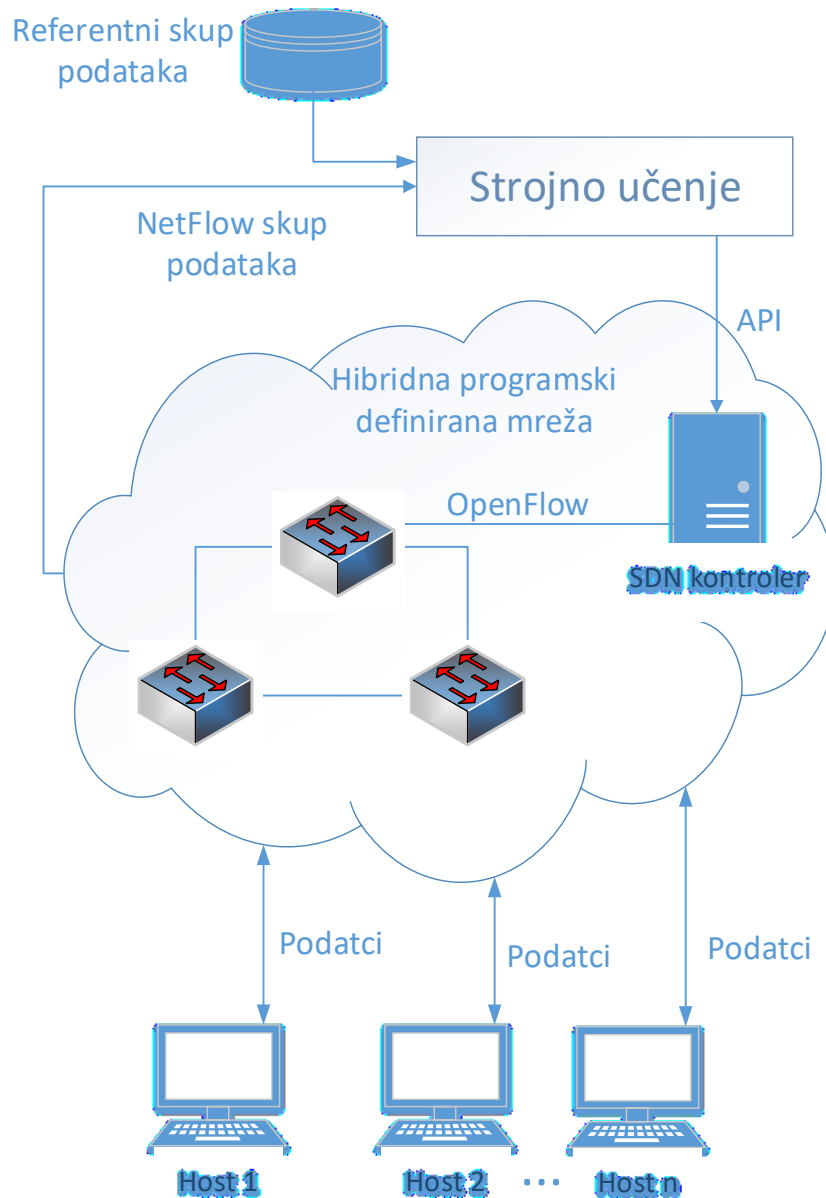
Ručne metode otkrivanja i praćenja mrežnog prometa mogu biti učinkovite u slučaju nekoliko mjernih podataka, ali u suvremenom okruženju s mnogo mjernih podataka i mjernih točaka, potrebno je razmotriti brže i sveobuhvatnije načine otkrivanja različitih vrsta anomalija. Tradicionalni sustavi koji nemaju funkcije u stvarnom vremenu, mogu se koristiti za analizu povijesnih zapisa i događaja kao i za informiranje o budućim odlukama u dugoročnom planiranju.

U kontekstu istraženih problema, izazova, prednosti i ograničenja primjene programski definirane mreže te potencijala implementacije strojnog učenja za otkrivanje kibernetičkih napada kroz analizu anomalija mrežnog prometa, razvijen je jedinstven i napredan model koji obuhvaća sve navedene poglede. Osnovna ideja predloženog NFMIDS-modela (*NetFlow Machine-learning Intrusion Detection System*) može se jasno ilustrirati slikom 4.1. Model se sastoji od nekoliko ključnih dijelova, među kojima su: hibridna programski definirana mreža, strojno učenje, referentni skup podataka za učenje, skup podataka iz stvarne mrežne okoline te primijenjeni najučinkovitiji algoritam strojnog učenja.

Strojno učenje igra važnu ulogu u modelu, koristeći referentni skup podataka za učenje kako bi se steklo znanje o karakteristikama normalnog mrežnog prometa. Dodatno, skup podataka iz stvarne mrežne okoline pruža stvarne scenarije iz kojih model može učiti i prilagođavati se dinamičnim novonastalim uvjetima. Odabir najučinkovitijeg algoritma strojnog učenja presudan je za postizanje visoke preciznosti i brzog otkrivanja anomalija.

Hibridna programski definirana mreža predstavlja temeljni element arhitekture modela. Uključuje SDN-kontroler koji omogućuje API-komunikaciju prema višim slojevima, poput aplikacija, i nižim slojevima, koji obuhvaćaju SDN-mrežnu opremu. Predloženi model uključuje i postojeće mrežne uređaje poput preklopnika u tradicionalnim mrežnim okolinama.

Sve te komponente tvore integrirani pristup koji kombinira snagu strojnog učenja s fleksibilnošću SDN arhitekture, čime se ostvaruje unaprijeđena sposobnost otkrivanja i obrane od kibernetičkih prijetnji.



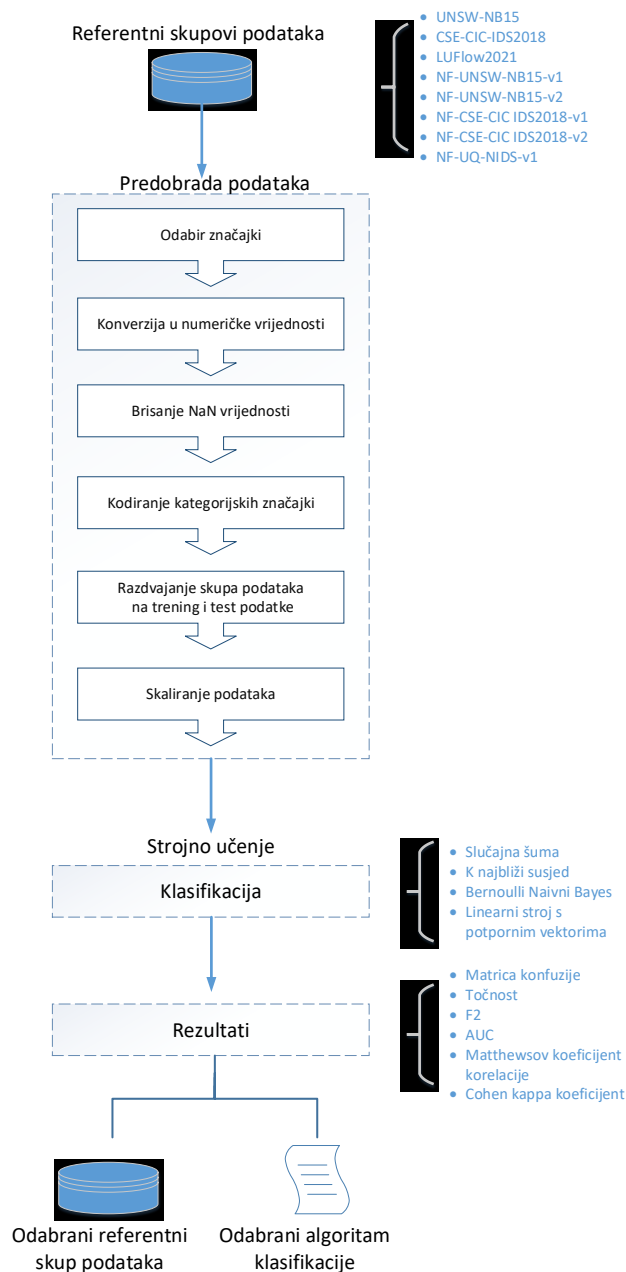
Slika 4.1. Osnovna ideja predloženog rješenja za otkrivanje anomalija mrežnog prometa u hibridnoj SDN-mreži

Narednim koracima u razvoju i implementaciji predloženog modela za otkrivanje anomalija u mrežnom prometu obuhvatilo se niz etapa. Prvo, proveden je pažljiv odabir i usporedba javno dostupnih skupova podataka namijenjenih otkrivanju anomalija u mrežnom prometu, istovremeno birajući referentni skup podataka za učenje unutar domene strojnog učenja. Zatim je slijedio odabir i usporedba uspješnosti različitih algoritama nadziranog strojnog učenja, s posebnom pažnjom na

identifikaciju najprimjenjivijeg algoritma za precizno otkrivanje anomalija u mrežnom prometu. Nakon toga je napravljena detaljna predobrada odabranog skupa podataka, prilagođena karakteristikama NetFlow-zapisa tokova mrežnih podataka u stvarnoj hibridno-programski definiranoj mreži. Taj je korak imao bitnu ulogu u optimizaciji podataka za učinkovitu primjenu strojnog učenja. Sljedeći je korak uključivao fino podešavanje hiperparametara odabranih klasifikatora kako bi se postigle najbolje moguće performanse u klasifikaciji i otkrivanju anomalija. Konačno, implementacija modela obuhvatila je primjenu strojnog učenja na predloženom modelu, što će omogućiti otkrivanje stvarnih kibernetičkih prijetnji. Taj sveobuhvatni pristup osigurao je razvoj visoko učinkovitog sustava za otkrivanje prijetnji u mrežnom prometu, prilagođenog specifičnostima hibridne programski definirane mreže.

4.1. Odabir i prilagodba ulaznih podataka modela za otkrivanje anomalija mrežnog prometa

Mnoga se istraživanja usmjeravaju na otkrivanje anomalija zbog njihove važnosti u otkrivanju novih kibernetičkih napada. Međutim, implementacija sustava otkrivanja anomalija u stvarnom okruženju izuzetno je složena jer zahtijeva opsežno testiranje, evaluaciju i prilagodbu prije nego što se može implementirati. Implementacija takvih sustava pomoću stvarnih i označenih mrežnih tokova prometa s velikim skupom normalnog i neželjenog mrežnog prometa idealna je metodologija za testiranje i evaluaciju njihove učinkovitosti. Proces odabira i prilagodbe ulaznih podataka može se razložiti u nekoliko etapa, koje su prikazane na slici 4.2. i detaljno objašnjene u sljedećim potpoglavljima.



Slika 4.2. Proces odabira i prilagodbe ulaznih podataka

Kako bi se izbjegle nedosljednosti eksperimentalne platforme, sva se ispitivanja implementiraju i izvode u istom okruženju koristeći programski jezik *Python* i hardverske komponente kao što je prikazano u tablici 4.1.

Tablica 4.1. Opis komponenti eksperimentalne platforme za odabir i prilagodbu ulaznih podataka

naziv komponente	karakteristike
računalo	Virtualno računalo na VMWare ESX-platforni
operativni sustav	Linux Ubuntu 20.04 LTS
CPU	32
RAM	64 GB
prostor za pohranu	100 GB
programski jezik	<i>Python 3.8</i>
programski alati	Sci-kit Learn 1.1.1

Za ulazne podatke korišteno je osam referentnih javno dostupnih skupova podataka, koji su zahtijevali različitu predobradu kako bi bili prikladni za analizu. Proces predobrade podataka obuhvatio je sljedeće korake: odabir relevantnih značajki u skladu s NetFlow-strukturom; konverziju i kodiranje kategorijskih značajki u numeričke vrijednosti radi daljnje analize; čišćenje nepotpunih podataka kako bi se osigurala pouzdanost i točnost u rezultatima; razdvajanje skupa podataka na podskupove za učenje i testiranje kako bi se omogućila validacija modela i skaliranje podataka radi bolje usporedivosti među značajkama. Zatim je provedena evaluacija performansi pet različitih klasifikatora u strojnom učenju na predobrađenim podatcima. Rezultati su bili podvrgnuti analizi kroz različite metrike uspješnosti kako bi se odabrao optimalan referentni skup podataka i najučinkovitiji klasifikator za implementaciju u predloženi model. Taj je pristup omogućio identifikaciju najboljih kombinacija podataka i algoritama koji su najprikladniji za ciljani problem u otkrivanju kibernetičkih prijetnji.

4.1.1. Odabir i usporedba javno dostupnih referentnih skupova podataka za otkrivanje anomalija

Izrada modela za klasifikaciju podataka i metodologije počela je prikupljanjem podataka o toku prometa. Podatci se mogu prikupiti snimanjem stvarnog prometa podataka u poznatoj mrežnoj infrastrukturi ili korištenjem javno dostupnih skupova podataka za rješavanje određenog problema. U svrhu odabira optimalnog algoritma i podešavanja parametara klasifikatora strojnog

učenja, provedeno je testiranje na više javno dostupnih skupova podataka. Skupovi podataka opisani u daljnjim dijelovima istraživanja koristili su se za učenje i testiranje modela. Prednost tih skupova podataka je u tomu što se lako koriste (dostupni su sirovi podatci) i korišteni su u mnogim znanstvenim istraživanjima, način na koji su generirani dobro je dokumentiran te su označeni kojoj klasi pripadaju pojedini segmenti podataka. Sustavi za otkrivanje napada u mreži temeljeni na strojnom učenju postali su učinkoviti alati za zaštitu mreža od kibernetičkih napada, a za razvoj i evaluaciju tih sustava, koristi se velik broj javno dostupnih skupova podataka u istraživačkoj zajednici. Međutim, zbog različitih tipova značajki u tim skupovima podataka, vrlo je teško pouzdano usporediti ML-modele koji koriste različite skupove podataka.

4.1.1.1. Cjeloviti skupovi podataka

Tijekom pretraživanja dostupnih javnih podataka za korištenje u modelu strojnog učenja, izdvojena su tri skupa podataka stvorena posljednjih nekoliko godina. Ti odabrani skupovi podataka korišteni su sa svim značajkama, izuzev značajki koje određuju redni broj, IP-adrese ili vremenske informacije nastajanja toka mrežnog prometa.

Tablica 4.2. Osnovne karakteristike odabranih skupova podataka

naziv	broj značajki	broj zapisa (tokova prometa)			veličina (MB)
		normalan	anomalije	ukupno	
UNSW-NB15	49	2218764	321283 (12,65%)	2540047	559
CSE-CIC-IDS2018	80	3522290	1121068 (24,14%)	4643358	1320
LUFlow2021	16	2976850	1273495 (29,96%)	4250345	476

Tablica 4.2. prikazuje osnovne karakteristike odabranih skupova podataka. Svaki skup podataka opisan je brojem značajki, brojem zapisa (tokova prometa) te veličinom u megabajtima. Također, za svaki skup podataka navedeni su podatci o normalnom i neželjenom prometu (anomalije) te ukupan broj zapisa. Otkriva se raznolikost tih skupova podataka u pogledu broja značajki, ukupnog broja zapisa, omjera normalnog i neželjenog mrežnog prometa te njihove veličine. Tablica pruža važne informacije o skupovima podataka koje su od bitne za njihovu daljnju upotrebu.

UNSW-NB15 skup podataka

Skup podataka UNSW-NB15 sastoji se od sirovih mrežnih paketa (PCAP-datoteka, engl. *Packet Capture*) koji su stvoreni alatom IXIA PerfectStorm u laboratoriju Cyber Range australskog Centra za kibernetičku sigurnost. Taj je alat korišten za generiranje primjera stvarnih, suvremenih normalnih aktivnosti i sintetičkih kibernetičkih napada. Skup podataka sadrži devet scenarija napada: *Fuzzers*, *Analysis*, *Backdoor*, *DoS*, *Exploits*, *Generic*, *Reconnaissance*, *Shellcode* i *Worms*. Alatom Argus, Bro-IDS generirano je ukupno 47 značajki s dvije oznake klase mrežnog prometa [98]. U skupu podataka postoje dvije oznake: jedna označava klasu u binarnom obliku koja razlikuje normalni promet od anomalija, dok druga označava vrstu anomalije (kibernetički napad) ako je promet klasificiran kao anomalija. Značajke su kategorizirane u nekoliko tipova, uključujući kategorijske, numeričke (cjelobrojne, decimalne i binarne) i vremenske. Skup podataka sadrži 2 218 764 (87,35%) tokova normalnog mrežnog prometa i 321 283 (12,65 %) anomalija, odnosno ukupno 2 540 047 tokova mrežnog prometa. U ovom dijelu istraživanja koristio se cijeli skup podataka.

CSE-CIC-IDS2018 skup podataka

Taj skup podataka uključuje sedam različitih scenarija napada: *Brute-force*, *Heartbleed*, *Botnet*, *DoS*, *DDoS*, *Web-napadi* i *napad infiltracije u mrežu* koji broje trinaest različitih vrsta kibernetičkih napada. U skupu podataka, zapisi su pohranjeni u obliku od 80 značajki koje su izvučene i obrađene iz zabilježenog prometa. Zapisi su organizirani prema datumu, pri čemu se za svaki od 10 dana prikupljanja pohranjuju sirovi podatci koji uključuju promet mreže (PCAP-datoteke) i zapisnike događaja (Windows i Linux Ubuntu) za svaki pojedini uređaj. U procesu ekstrakcije značajki iz neobrađenih podataka, korišten je alat CICFlowMeter-V3, a izdvojene značajke su pohranjene kao CSV-datoteke (engl. *Comma-Separated Values*) [99]. U tom dijelu istraživanja koristio se skup podataka iz prvih četiriju CSV-datoteka uz dodane tokove mrežnog prometa koji su označeni kao anomalije iz ostalih datoteka. Tako sastavljen skup podataka sastoji se od ukupno 4 643 358 zapisa, 1 121 068 (24,14 %) zapisa su anomalije i ostatak (75,95 %)

normalnog mrežnog prometa. Razlog korištenja tako modificiranog skupa podataka je kako bi se udio normalnog i neželjenog mrežnog prometa približio udjelu u drugim skupovima podataka.

LUFlow2021 skup podataka

LUFlow je skup podataka za otkrivanje napada u mreži temeljen na toku podataka koji sadrži telemetrijske podatke o kibernetičkim napadima na mamce u komunikacijskim mrežama Sveučilišta Lancaster. Anomalije se u toku podataka automatski označavaju uz pomoć korelacije s izvorima *Cyber Threat Intelligence*, što omogućuje kontinuirano hvatanje, označavanje i objavljivanje telemetrijskih podataka u tom skupu podataka. Tokovi koji se nisu mogli odrediti kao zlonamjerni, ali nisu dio normalnog telemetrijskog profila, označeni su kao izvanredni tokovi i uključeni su u skup podataka kako bi se potaknula daljnja analiza. Taj skup podataka sadrži i uobičajeni promet te se kontinuirano ažurira zahvaljujući mehanizmu automatskog označavanja, što omogućuje otkrivanje novih obrazaca napada u stvarnom vremenu [100]. Ograničenje tog skupa podataka je što samo označava promet kao benigni ili zlonamjerni te ne sadrži oznaku vrste napada. To ograničenje proizlazi iz činjenice da se podatci prikupljaju iz stvarnog prometa, stoga je nemoguće odrediti pravu namjeru koja stoji iza svakog prometa. Osim toga, neki se tokovi prometa klasificiraju kao „*outliers*“ ako sadrže sumnjivu aktivnost. Postojanje takve oznake omogućava označavanje nepoznatog prometa te kasnije dodatnom analizom odrediti pravu narav takvog prometa te time poboljšati proces otkrivanja i prevencije zlonamjernih tokova prometa [101]. Spremište tih podataka javno je dostupno i strukturirano je prema godini i mjesecu u kojem su podatci prikupljeni. U ovom dijelu istraživanja disertacije koristile su se zadnje uvrštene datoteke iz veljače 2021. godine koje sadrže ukupno 5 033 685 tokova mrežnog prometa od kojih je 1 273 495 odnosno 29,96 % označeno kao anomalije. Tokovi prometa koji se nisu mogli odrediti kao normalan promet ili anomalije (783 340 zapisa) izbačeni su iz promatranja u ovom istraživanju disertacije, stoga je ukupan broj zapisa umanjen i iznosi 4 250 345 zapisa.

4.1.1.2. NetFlow-skupovi podataka

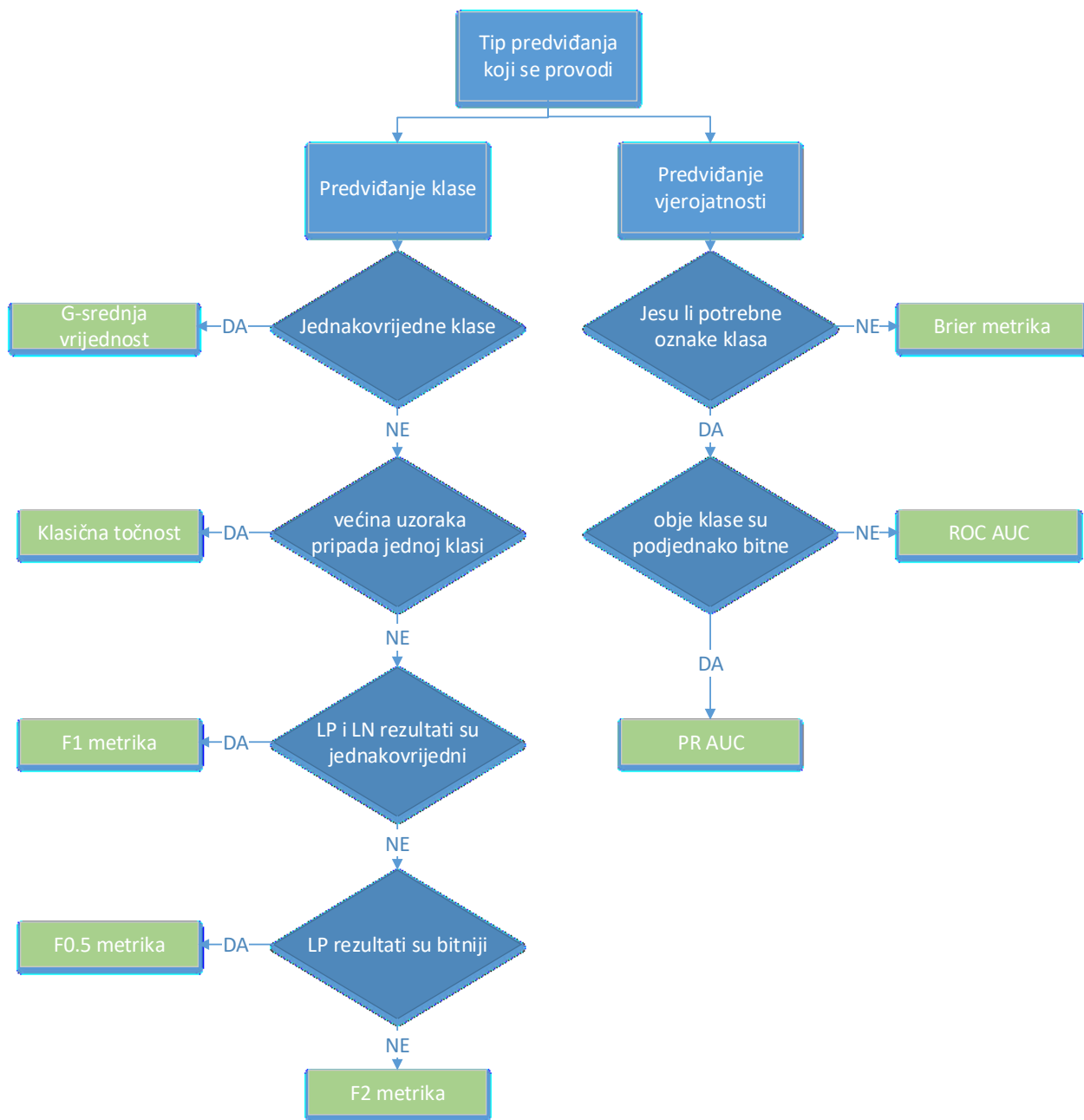
NetFlow-skupovi podataka generirani su iz postojećih referentnih skupova podataka: UNSW-NB15 i CSE-CIC-IDS2018 s osnovnim (zajedničkim) i proširenim skupom značajki, temeljenim na NetFlow-protokolu koji pruža statistiku o paketima koji prolaze kroz usmjerivač i postao je standard za prikupljanje operativnih podataka o IP-mrežama [102]. Prednost korištenja strukture NetFlow-formata uključuje njegovu praktičnu primjenu u stvarnim mrežama. NetFlow-skupovi podataka koji se koriste u ovom istraživanju označeni su kao mrežni promet s binarnim klasama i dodatno su razdijeljeni po broju značajki, gdje se oznakom v1 označavaju skupovi podataka s 12 osnovnih značajki i 2 oznake kategorije prometa (binarna i kategorijska oznaka), dok se oznakom v2 označava skup podataka s proširenim brojem značajki od ukupno 43 značajke i 2 oznake kategorije prometa (binarna i kategorijska oznaka). U tablici 4.3. prikazana je usporedba NetFlow-skupova podataka s obzirom na broj značajki, broj zapisa i veličine datoteke. Prednost tih skupova podataka je što su sve vrijednosti značajki numeričke te tako već prilagođene za proces strojnog učenja. NF-UQ-NIDS je skup podataka koji spaja spomenute i druge (BoT-IoT, ToN-IoT) NetFlow-skupove podataka te predstavlja univerzalni NIDS-skup podataka koji sadrži tokove iz višestrukih mrežnih okruženja i različitih kibernetičkih napada [103]. NF-UQ-NIDS-skup podataka ima jednu značajku više od ostalih NetFlow-skupova podataka, a označava iz kojeg je NF-skupa podataka preuzet zapis, no ta značajka se nije koristila jer nema praktičnu važnost u procesu strojnog učenja. NF-UQ-NIDS-v2-skup podatka nije uzet u obzir tijekom istraživanja, budući da dostupni hardverski resursi (64 GB RAM memorije) nisu dovoljni za rad s tako velikim skupom podataka s obzirom na memorijske resurse korištenog računalnog sustava.

Tablica 4.3. Osnovne karakteristike NetFlow-skupova podataka

naziv	broj značajki	broj zapisa (tokova prometa)			veličina (MB)
		normalan	anomalije	ukupno	
NF-UNSW-NB15-v1	14	1 550 712	72 406 (4%)	1 623 118	114
NF-UNSW-NB15-v2	45	2 295 222	95 053 (4%)	2 390 275	431
NF-CSE-CIC IDS2018-v1	14	7 373 198	1 019 203 (12 %)	8 392 401	605
NF-CSE-CIC IDS2018-v2	45	16635567	2 258 141 (12 %)	18 893 708	3 145
NF-UQ-NIDS-v1	15	9 208 048	2 786 845 (23 %)	11 994 893	1 048

Svi skupovi podataka spadaju u vrlo nebalansirane skupove podataka gdje je udio anomalija višestruko manji od udjela normalnih zapisa.

Općenito se proces strojnog učenja sastoji od dvaju osnovnih dijelova: učenja i evaluacije uspješnosti za nove podatke. Učenje se provodi na referentnim skupovima podataka koji su prethodno označeni. Važnost odabira optimalnog referentnog skupa bitna je jer može značajno poboljšati pripremu modela za otkrivanje anomalija mrežnog prometa, omogućujući mu da što preciznije identificira anomalije. Definiranje je odgovarajućih metrika nužno za procjenu i odabir najpogodnijeg referentnog skupa, omogućavajući usporedbu različitih skupova podataka u kontekstu problema otkrivanja anomalija. Prikaz procesa odlučivanja za nebalansirane skupove podataka na slici 4.3. može pružiti korisnu pomoć pri odabiru odgovarajuće metrike. Kako bi se ocijenile performanse modela koji koristi različite ulazne podatke, potrebno je pravilno odrediti metrike za usporedbu uspješnosti. S obzirom na izrazitu nebalansiranost svih promatranih skupova podataka, ROC AUC-metrika (engl. *Receiver Operating Characteristic Area Under the Curve*) ističe se kao najprikladniji kandidat za daljnje korake odabira i prilagodbe ulaznih podataka u procesu strojnog učenja. ROC AUC se često koristi kao metrika u strojnom učenju, posebno u zadacima binarne i višeklasne klasifikacije, iz više razloga. Prvo, ROC AUC pruža pouzdaniju procjenu performansi u nebalansiranim skupovima podataka, gdje klasična točnost može biti nepouzdana metrika zbog osjetljivosti na neravnotežu klasa. Ta metrika uzima u obzir ravnotežu između osjetljivosti i specifičnosti pri različitim pragovima klasifikacije, što je čini otpornijom na nebalansirane ulazne podatke. Dodatno, ROC AUC je neovisan o pragu, ocjenjujući performanse modela pri svim mogućim pragovima klasifikacije, što je korisno kod utjecaja povezanih s lažno pozitivnim i lažno negativnim rezultatima. Ta metrika nije osjetljiva na distribuciju klasa u skupu podataka, što čini ROC AUC izuzetno korisnim za različite scenarije gdje se udjeli klasa razlikuju. Važno je napomenuti da se ROC AUC može koristiti i za binarne i višeklasne klasifikacije, često izračunavajući prosječnu vrijednost za sve klase u višeklasnom kontekstu [92], [96], [104].



Slika 4.3. Odabir metrike za nebalansirani skup podataka [91]

4.1.2. Predobrada referentnih skupova podataka

Pri prikupljanju podataka za razne namjene, bez obzira na to je li za strojno učenje, rudarenje podataka ili nešto drugo, važno je imati mehanizme za identificiranje i izdvajanje relevantnih podataka koji su potrebni za određenu analitičku svrhu. Svaki sustav strojnog učenja ima

jedinstvene potrebe u pogledu načina na koji se podatci predstavljaju, što znači da je često potrebno izdvojiti i pripremiti relevantne podatke za zadane analitičke svrhe. Izbor specifičnih podataka za analizu može značajno utjecati na modele koji se uče, stoga priprema podataka ključna je etapa u mnogim projektima strojnog učenja ili rudarenja podataka, koja često predstavlja najzahtjevniji dio projekta. U većini slučajeva, priprema podataka sastoji se od niza transformacija koje se moraju ponoviti nekoliko puta. Bez obzira na dostupnost tehnologija za rad s podacima, svaka transformacija može zahtijevati veliku količinu ručnog rada i trošiti značajno vrijeme i trud, zato se često smatra da je priprema podataka najteži i najduži dio procesa analize podataka [105]. Neuredni i neorganizirani podatci najveća su prepreka dobroj i učinkovitoj analizi te se smatra da se 70 % vremena u istraživanju troši na čišćenje podataka [106]. Čišćenje podataka je proces otkrivanja i ispravljanja (ili uklanjanja) oštećenih ili netočnih zapisa iz skupa podataka, tablica ili baze podataka. Prema [105] postupak čišćenja podataka sastoji se od nekoliko etapa. Prvo, potrebno je identificirati i definirati pogreške u podacima, kao što su nepotpunost, netočnost ili irelevantnost. Nakon toga, pogreške se čiste i ispravljaju zamjenom, izmjenom ili brisanjem. U toj važno je dokumentirati vrstu i primjer pogreške. Konačno, potrebno je provjeriti jesu li provedene promjene u skladu sa zahtjevima korisnika ili procesa. Taj proces može se ponavljati više puta kako bi se osiguralo da su podatci čisti i valjani za daljnju upotrebu.

4.1.2.1. Brisanje suvišnih značajki

Ovisno o skupu podataka koji se koristi u procesu strojnog učenja, razlikuje se proces brisanja suvišnih odnosno irelevantnih značajki. U klasifikaciji mrežnog prometa, neke značajke kao što su vrijeme stvaranja zapisa, IP-adrese ili druge specifične značajke koje se izračunavaju temeljem drugih parametara, mogu postati nevažne [107]. U strojnom učenju i predloženom modelu za otkrivanje anomalija, IP-adresa izvora anomalije ili redni broj toka nisu ključni čimbenici u odlučivanju o klasifikaciji mrežnog prometa. Slično tomu, niti informacija o vremenu nastanka ili odredišna IP-adresa ne koristi se u etapi učenja modela jer se vrijeme i način izvođenja kibernetičkog napada ne može predvidjeti iz spomenutih informacija. Odabrane i suvišne značajke obrisane su kako bi se smanjila dimenzionalnost skupa podataka te poboljšale performanse procesa strojnog učenja. Osim toga, neke značajke mogu sadržavati nepotpune podatke i tako smanjiti točnost klasifikacije modela. Kako ističu autori u [108], točnost klasifikacije modela može se

povećati s određenim smanjenjem broja značajki. U radu [109] autori su došli do zaključka da je u SDN-sustavima za otkrivanje kibernetičkih napada, manji broj značajki poželjniji te da povećanje broja značajki nužno ne dovodi do povećanja točnosti klasifikacije. U skupovima podataka često se koriste kategorijske značajke te postoji potencijalna mogućnost povećanja kompleksnosti skupa podataka tijekom kodiranja značajki u numeričke vrijednosti što dovodi do povećanja vremena potrebnog za klasifikaciju. Udio i tip obrisanih značajki prema određenom skupu podataka prikazan je tablicom 4.4.

Tablica 4.4. Udio i tip obrisanih značajki tijekom predobrade skupova podataka

skup podataka	broj značajki	broj obrisanih značajki	udio obrisanih značajki
UNSW-NB15	49	10	20,41 %
CSE-CIC-IDS2018	80	1	1,25 %
LUFlow2021	16	4	25 %
NF-UNSW-NB15-v1	14	3	21,42 %
NF-UNSW-NB15-v2	45	3	6,67 %
NF-CSE-CIC IDS2018-v1	14	3	21,43 %
NF-CSE-CIC IDS2018-v2	45	3	6,67 %
NF-UQ-NIDS-v1	15	4	26,67 %

U toj etapi predobrade podataka uklonjene su značajke koje zbog svoje strukture ili načina prikupljanja nisu značajne ili u daljnjoj obradi (kodiranje) mogu dodatno povećati kompleksnost skupa podataka.

U skupu podataka UNSW-NB15 obrisano je 10 značajki: *redni broj zapisa*, *srcip*, *dstip*, *service*, *Stime*, *Ltime*, *ct_flw_http_mthd*, *is_ftp_login*, *ct_ftp_cmd*, *attack_cat*. Neke od značajki koje se prikupljaju tijekom nadzora mreže, poput vremenske oznake i IP-adrese, nisu nužno bitne za otkrivanje kibernetičkih napada. S druge strane, neke značajke, poput specifičnih vrijednosti koje se računaju drugim alatima, ne mogu se prikupiti izravno s mrežnih uređaja.

Skup podataka CSE-CIC-IDS2018 dobro je strukturiran jer su značajke u izvornom obliku već kodirane i ne sadrži značajku IP-adrese, stoga je obrisana samo jedna značajka, ona koja se odnosi na vrijeme nastanka zapisa, *Timestamp*.

U skupu podataka LUFLOW2021 obrisane su 4 značajke: *dest_ip*, *src_ip*, *time_end*, *time_start*. Iako su *dest_ip*, *src_ip* već u izvornom skupu podataka kodirane, nisu relevantne za klasifikaciju strojnog učenja, dok su *time_end*, *time_start* značajke s vremenima nastanka zapisa također irelevantne.

Referentni NetFlow-skupovi podataka imaju značajke koje su već pripremljene za proces strojnog učenja (kodirane značajke), a razlikuju se po broju značajki, verzija v1 sadrži 14 dok verzija v2 sadrži 45 značajki. U objema verzijama obrisane su tri značajke: *IPV4_SRC_ADDR*, *IPV4_DST_ADDR* i *Attack* koje označavaju IP-adrese u zapisima odnosno vrstu napada (anomalije), dok je u NF-UQ-NIDS-v1-skupu podataka obrisana i značajka *Dataset* koja označava iz kojih su NetFlow-skupova podataka preuzete vrste napada (anomalija). Tu značajku sadrži samo NF-UQ-NIDS-skup podataka. Odabir obrisanih značajki (*Timestamp*, *IP address*) podudara se s odabirom obrisanih značajki u sličnim istraživanjima [110], [111]. Neke od metoda odabira značajki putem strojnog učenja korištenih kao u [112] koristile su se pri korištenju odabranog referentnog skupa podataka, a opisane su u poglavlju 4.1.5.1. Te tehnike odabira značajki pomažu u usmjeravanju na relevantne podatke, čime se poboljšavaju performanse modela u otkrivanju anomalija i prijetnji unutar mrežne infrastrukture.

4.1.2.2. Čišćenje neispravnih vrijednosti značajki

Gotovo svi skupovi podataka s kojima se susrećemo imaju neke vrijednosti koje nedostaju. Izazovno je učinkovito rukovati takvim podacima i stvoriti robustne modele. Postoji niz metoda kojima se u podacima mogu zamijeniti vrijednosti koje nedostaju. Važno je pristupiti svakom skupu podataka na fleksibilan način, koristeći različite metode zamjene takvih vrijednosti, ovisno o problemu i značajkama koje se koriste. Najčešće se koriste neke od metoda zamjene neispravnih vrijednosti značajki kako je opisano prema [113]:

- **Brisanje redaka** - ta se metoda obično koristi za manipulaciju skupa podataka s tzv. nul-vrijednostima. Uobičajeno se briše određeni redak koji sadrži nul-vrijednost za određenu značajku ili cijela značajka ako ima više 70 – 75 % vrijednosti koje nedostaju. Ta se metoda savjetuje samo ako u skupu podataka ima dovoljno uzoraka. Uklanjanje podataka dovodi do gubitka informacija što može dati neočekivane rezultate tijekom procesa predviđanja.

Prednost tog načina čišćenja skupa podataka je stvaranje robustnog i vrlo točnog modela, a brisanje određenog retka ili stupca bez posebnih informacija nema veliki negativan utjecaj na proces strojnog učenja. Nedostatci su te metode gubitak informacija i loš utjecaj na strojno učenje ako je postotak vrijednosti koje nedostaju visok ($> 30\%$) u usporedbi s cijelim skupom podataka.

- **Zamjena sa srednjom vrijednosti** - ta se metoda može primijeniti samo na značajke koje imaju numeričke podatke. Srednja vrijednost ili medijan svih vrijednosti značajke može se izračunati i umetnuti kao zamjena za nedostajuće vrijednosti. Primjenom te metode na skup podataka može doći do povećanja varijabilnosti, ali se istovremeno smanjuje gubitak podataka, što često daje bolje rezultate u usporedbi s metodama poput uklanjanja redaka i stupaca. Prednost je metode sprječavanje gubitka podataka, osobito ako je skup podataka mali. Međutim, nedostatci uključuju povećanje varijabilnosti podataka i moguće povećanje pristranosti.
- **Dodjela jedinstvene oznake** - kategorijskim značajkama može se dodijeliti posebna oznaka za vrijednosti koje nedostaju. Ta metoda dodaje nove informacije u skup podataka, što može utjecati na varijabilnost. S obzirom na to da se radi o kategorijskim značajkama, potrebna je njihova konverzija u numerički zapis kako bi algoritam strojnog učenja mogao funkcionirati. Ovisno o tipu kodiranja, može doći do povećanja dimenzionalnosti skupa podataka, što je jedan od nedostataka metode. Prednost je metode izbjegavanje gubitka informacija.

Tablica 4.5. Udio obrisanih podataka u referentnim skupovima podataka

skup podataka	broj obrisanih redaka	dio obrisanih redaka
UNSW-NB15	308	0,012 %
CSE-CIC-IDS2018	442 505	9,529 %
LUFlow2021	8 621	0,202 %
NF-UNSW-NB15-v1	0	0 %
NF-UNSW-NB15-v2	0	0 %
NF-CSE-CIC IDS2018-v1	0	0 %
NF-CSE-CIC IDS2018-v2	0	0 %
NF-UQ-NIDS-v1	0	0 %

Tijekom procesa predobrade referentnih skupova podataka i čišćenja podataka, korištena je metoda brisanja redaka koji sadrže neispravne vrijednosti (engl. *Not-A-Number*, *NaN*). Ta je metoda odabrana jer je postotak obrisanih redaka bio relativno mali. U tablici 4.5. udio obrisanih redaka pokazuje relativni udio izgubljenih podataka u odnosu na ukupan broj redaka u svakom skupu podataka. NetFlow-skupovi podataka su dobro strukturirani i pročišćeni, stoga nije bilo potrebno brisanje kao u cjelovitim skupovima podataka. Brisanje podataka s neispravnim podatcima koristi se i u sličnim istraživanjima poput [110], [111], [114] gdje se brišu uzorci s nedostajućim vrijednostima ili tzv „NaN“-vrijednostima.

4.1.2.3. Kodiranje kategorijskih značajki


Kodiranje vrijednosti kategorijskih značajki jedan je od važnijih zadataka u pripremi podataka. Većina stvarnih podataka sadrži kategorijske vrijednosti, dok većina modela strojnog učenja radi isključivo s numeričkim vrijednostima. Budući da modeli strojnog učenja izvode matematičke operacije, potrebni su im numerički podatci poput decimalnih ili cijelih brojeva. Stoga se nizovi i druge vrste podataka moraju pretvoriti u odgovarajuće numeričke oblike. Kodiranje kategorijskih podataka je proces pretvaranja tih podataka u numerički format, što omogućava da se podatci s pretvorenim kategorijskim vrijednostima koriste u modelima strojnog učenja za dobivanje rezultata predviđanja. U nastavku je pregled osnovnih metoda pretvaranja kategorijskih u brojčane vrijednosti. Provedeno sustavno empirijsko istraživanje u [115] koje uključuje pet sustava, sedam modela i tri korištena načina kodiranja pokazuje da odabir načina kodiranja nije trivijalan.

Kodiranje labelama

Kodiranje labelama je popularna tehnika kodiranja za prilagodbu kategorijskih značajki gdje se svakoj jedinstvenoj kategorijskoj vrijednosti dodjeljuje jedinstveni cijeli broj na temelju abecednog redoslijeda kategorijskih značajki. Kod tog načina kodiranja nema povećavanja dimenzionalnosti skupa podataka koji može povećati zahtjeve za izračune kod algoritma strojnog učenja, ali postoji vjerojatnost da model neispravno bilježi odnos među neovisnim značajkama, kako je prikazano na primjeru tipa protokola u mrežnom prometu u tablici 4.6. Tim načinom

kodiranja značajka TCP se smješta među UDP i ICMP vrijednostima, što algoritam može pogrešno protumačiti i imati neželjeni učinak na model [66].


Tablica 4.6. Kodiranje labelama značajke tipa protokola

tip protokola		tip protokola nakon kodiranja
tcp		2
tcp		2
udp		3
icmp		1
udp		3

Kodiranje *One-hot*-metodom

Tehnika kodiranja *One-hot*-metodom proširuje određenu kategorijsku značajku u više novih značajki koje se nazivaju binarnim značajkama. Svaka nova binarna značajka ima vrijednost 0 ili 1, ovisno o tome je li pripadajuća kategorijska vrijednost prisutna ili nije prisutna. Taj postupak omogućuje grupiranje kategorijskih podataka bez gubitka informacija. Tablica 4.7. prikazuje primjer *One-hot*-kodiranja za značajku tipa protokola. U ovom primjeru, originalna značajka "tip protokola" ima tri različite vrijednosti: "tcp", "udp" i "icmp". *One-hot*-kodiranje proširuje tu kategorijsku značajku u trima novim binarnim značajkama: "tip protokola_tcp", "tip protokola_udp" i "tip protokola_icmp". Svaki redak u tablici predstavlja jedan primjer iz skupa podataka, a vrijednosti 0 ili 1 u pojedinim stupcima označavaju prisutnost ili odsutnost određenog tipa protokola. Na primjer, u prvom retku tip protokola je "tcp" pa je vrijednost u stupcu "tip protokola_tcp" 1, dok su vrijednosti u stupcima "tip protokola_udp" i "tip protokola_icmp" jednake 0 jer ti protokoli nisu prisutni u primjeru. Svaka vrijednost odabrane značajke tipa protokola u tablici je predstavljena kao 3-dimenzionalni zapis nakon *One-hot*-kodiranja. Općenito, za varijable kardinalnosti d , vektori značajke imat će d -dimenzija. Zanimljivo svojstvo *One-hot* kodiranja je da su značajke neovisne – svaka značajka je jednako udaljena jedna od druge u euklidskom prostoru [116].

Tablica 4.7. *One-hot*-kodiranje značajke tipa protokola

tip protokola		tip protokola_tcp	tip protokola_udp	tip protokola_icmp
tcp		1	0	0
tcp		1	0	0
udp		0	1	0
icmp		0	0	1
udp		0	1	0

Nakon primjene *One-hot* kodiranja, broj značajki može se značajno povećati, što može postati ograničavajući učinak zbog povećane potrebe za računalnim resursima tijekom procesa strojnog učenja. To može rezultirati duljim vremenom izračuna za neke algoritme strojnog učenja.

4.1.2.4. Usporedba kodiranja kategorijskih značajki

Prema dostupnoj dokumentaciji i opisu svih korištenih referentnih skupova podataka, kategorijske značajke kodirane su metodom kodiranja labelama. Takvi su skupovi podataka pripremljeni za strojno učenje odnosno sve značajke imaju numeričke vrijednosti te nije bilo potrebno kodirati niti jednu značajku. Skup podataka UNSW-NB15 jedini je u svom izvornom obliku sadržavao kategorijske značajke te su na tome skupu podataka testirane obje metode kodiranja kategorijskih značajki. Rezultati su pokazali jednaku ili vrlo malu razliku u uspješnosti klasifikacije za četiri različita algoritma klasifikacije. Za svaki algoritam klasifikacije primijenjeni su isti procesi predobrade podataka. Primjena *One-hot*-metode kodiranja pokazala je značajnu razliku u utrošku vremena za klasifikaciju kod algoritama K-NN i LSVC. Temeljem usporedbe točnosti AUC i vremenskog utroška na obama načinima kodiranja kategorijskih značajki, prema tablici 4.8., odabrano je kodiranje labelama za daljnja istraživanja na skupovima podataka stvarnih mrežnih okruženja.

Tablica 4.8. Usporedba trajanja i uspješnosti klasifikacije na skupu podataka UNSWB-NB15

algoritam	AUC		trajanje klasifikacije (sat: minuta: sekunda)	
	Kodiranje labelama	<i>One-hot</i> kodiranje	Kodiranje labelama	<i>One-hot</i> kodiranje
RandomForest	0.9867	0.9866	00:00:21	00:00:21
K-NN	0.978	0.978	00:49:44	01:54:58
BernoulliNB	0.8534	0.8538	00:00:02	00:00:10
LinearSVC	0.9853	0.9892	00:00:29	00:00:25

4.1.2.5. Razdvajanje podataka za učenje i testiranje

Razdvajanje skupa podataka uobičajena je tehnika za ocjenjivanje uspješnosti predviđanja algoritama strojnog učenja. Može se koristiti za probleme klasifikacije ili regresije i može se koristiti za bilo koji nadzirani algoritam strojnog učenja. Postupak uključuje podjelu skupa podataka u dvama podskupima. Prvi se podskup koristi za učenje, dok se drugi podskup koristi kao ulazni skup podataka nad kojim se izrađuju predviđanja i uspoređuju s očekivanim vrijednostima. Drugi podskup podataka naziva se skupom testnih podataka i koristi se za procjenu uspješnosti modela strojnog učenja. Cilj je procijeniti izvedbu modela strojnog učenja na novim podatcima odnosno podatcima koji se ne koriste za učenje modela. Kada je dostupan dovoljno velik skup podataka, preporučljivo je podijeliti ga na dva dijela: skup za učenje i skup za testiranje [117], [118].

U postupku izgradnje modela klasifikacije, koristi se skup podataka za učenje kako bi se model prilagodio različitim postavkama parametara algoritma klasifikacije. Nakon izgradnje modela, provodi se validacija na skupu podataka za testiranje koji sadrži uzorke s poznatim oznakama. Na taj način, predviđanja modela na testnom skupu mogu se koristiti za procjenu točnosti i performansi modela. Uobičajeni omjeri podjele skupa podataka za učenje i testiranje su 80/20 prema [111], [114], [119], [120], [121], zatim 70/30 prema [122], [123], te 50/50 prema [124], [125].

Scikit-learn platforma koja se koristila za proces klasifikacije u strojnom učenju omogućuje provedbu postupka podjele izvornog skupa podataka putem ugrađene funkcije `train_test_split()`. Parametar koji se koristi za definiranje omjera podataka za učenje i testiranje je vrijednost „`test_size`“. Ta je vrijednost broj u rasponu 0 do 1, a označava koji se omjer iz ukupnog skupa

podataka uzima za test podataka. Još je jedno važno razmatranje nasumično dodjeljivanje uzoraka podskupovima podataka za učenje i testiranje kako bi se osiguralo da oni prikazuju raznolikost izvornog skupa podataka, što bi trebalo rezultirati reprezentativnim rezultatima. Kada se uspoređuju algoritmi strojnog učenja, poželjno je da se rad i procjena uspješnosti temelje na istim podskupovima izvornog skupa podataka. To se postiže fiksiranjem sjemena za generator pseudo-slučajnih brojeva koji se koristi pri dijeljenju skupa podataka odnosno postavljanjem "*random_state*" parametra na cjelobrojnu vrijednost. Mnogi problemi klasifikacije koji se rješavaju strojnim učenjem uključuju neuravnotežene skupove podataka, što znači da su neke klase zastupljene rjeđe u odnosu na druge. Kada se dijeli skup podataka na podskupove za učenje i testiranje, idealno bi bilo zadržati istu distribuciju uzoraka u svakoj klasi kao što je u izvornom skupu podataka. Odnosno, kako bi se postigla ista distribucija uzoraka u svim klasama u podskupovima kao i u izvornom skupu podataka, može se koristiti parametar "*stratify*" pri podjeli na podskupove za učenje i testiranje. Korištenjem funkcije *train_test_split()* u obliku:

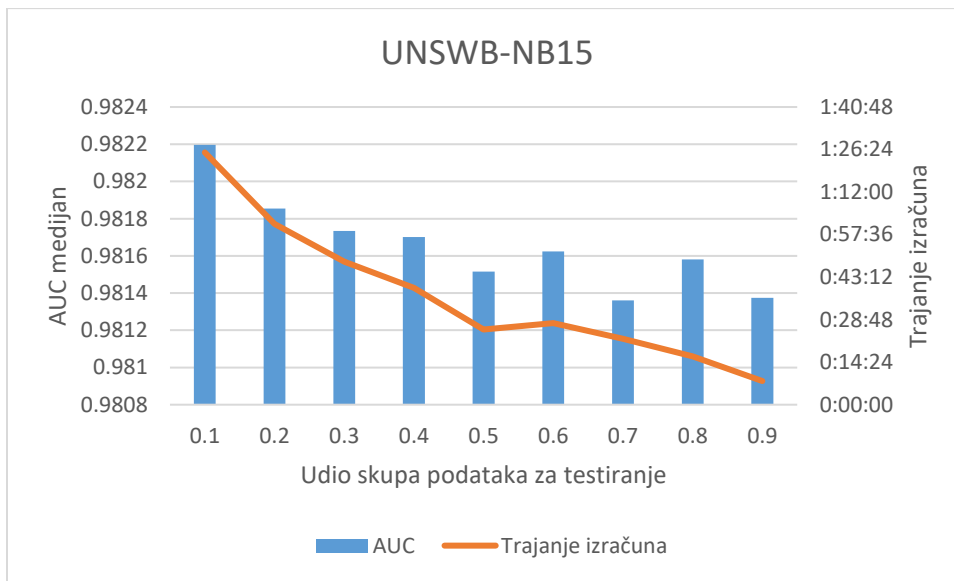
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=<decimal number>, random_state=1, stratify=y)
```

uzima se izvorni skup podataka *X* i *y* kao ulaz i dobiva se skup podataka podijeljen u dvama podskupovima posebno za učenje i testiranje, *X_train*, *y_train* i *X_test* i *y_test*. Podskupovi *X_train*, *y_train* koriste se u etapi učenja, dok se podskup podataka *X_test* i *y_test* koristi za određivanje uspješnosti klasifikacije.

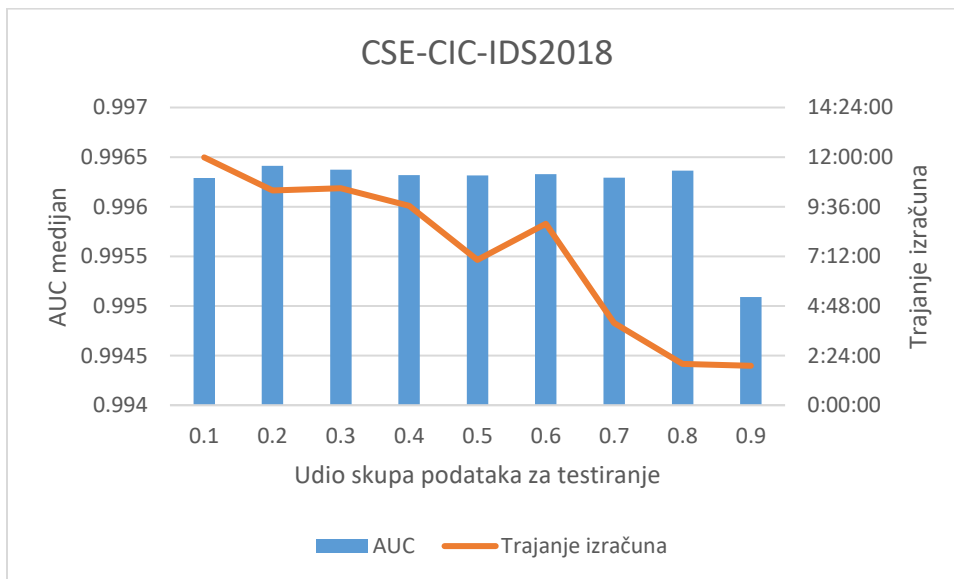
Eksperimentalno testiranje najboljeg omjera podskupova podataka za učenje i testiranje provedeno je četirima klasifikatorima (slučajna šuma, K-najbliži susjed, linearni stroj s potpunim vektorima i Bernoulli Naivni Bayesov klasifikator). Iako su uobičajeni omjeri 80/20, 70/30 ili 50/50 u ovom dijelu istraživanja ispitani su omjeri 10/90 do 90/10 s korakom 10. Tijekom izračunavanja uspješnosti klasifikatora za skup podataka NF-CSE-CIC-IDS2018-v2 s omjerima 20/80 i 10/90, došlo je do pogreške odnosno automatskog prekida izračunavanja AUC-metrike klasifikacije zbog prezahtjevnih računalnih resursa, stoga ti rezultati u grafikonu na slici 4.9. nisu prikazani.

U nastavku su prikazani grafikoni (slika 4.4. – 4.11.) koji prikazuju rezultate za svaki skup podataka te kako korišteni omjeri podataka za učenje i testiranje utječu na točnost klasifikacije. Na lijevoj ordinati grafikona prikazan je medijan AUC-točnosti svih klasifikatora, što bolje opisuje

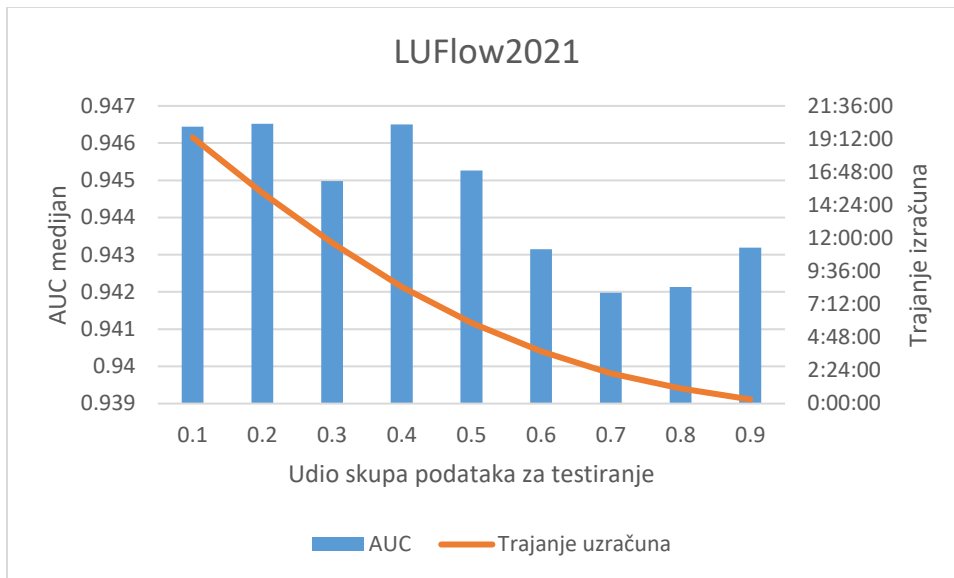
rezultate od srednje vrijednosti u slučaju velikih odstupanja u rezultatima, kao što je primjerice slučaj kod Bernoulli naivnog klasifikatora. Na desnoj ordinati grafikona prikazano je ukupno trajanje izračuna AUC-točnosti, također prikazano kao medijan svih rezultata koje su postigla navedena četiri klasifikatora.



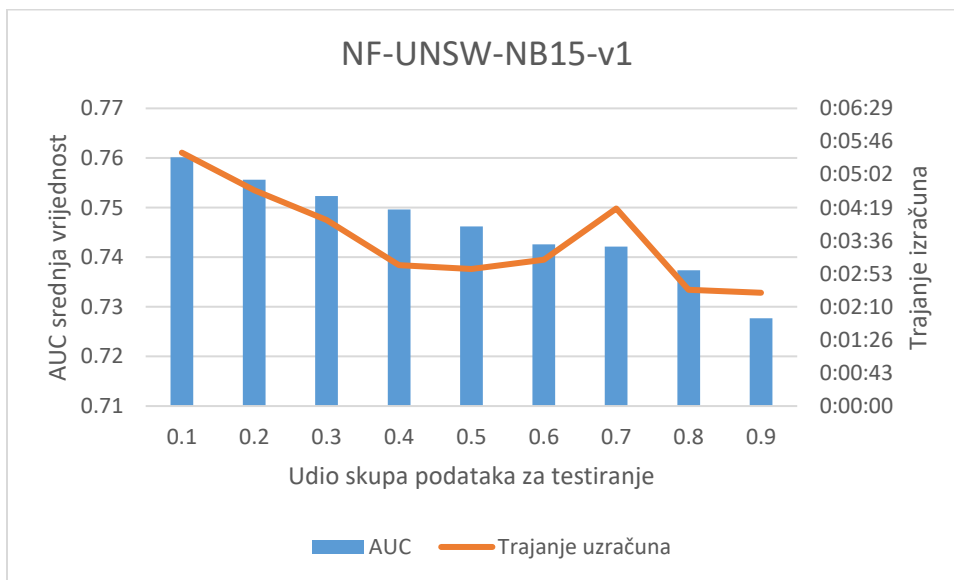
Slika 4.4. Rezultati AUC-točnosti i vremena izračuna u odnosu na omjer podskupova za učenje i testiranje na UNSWB NB15-skupu podataka



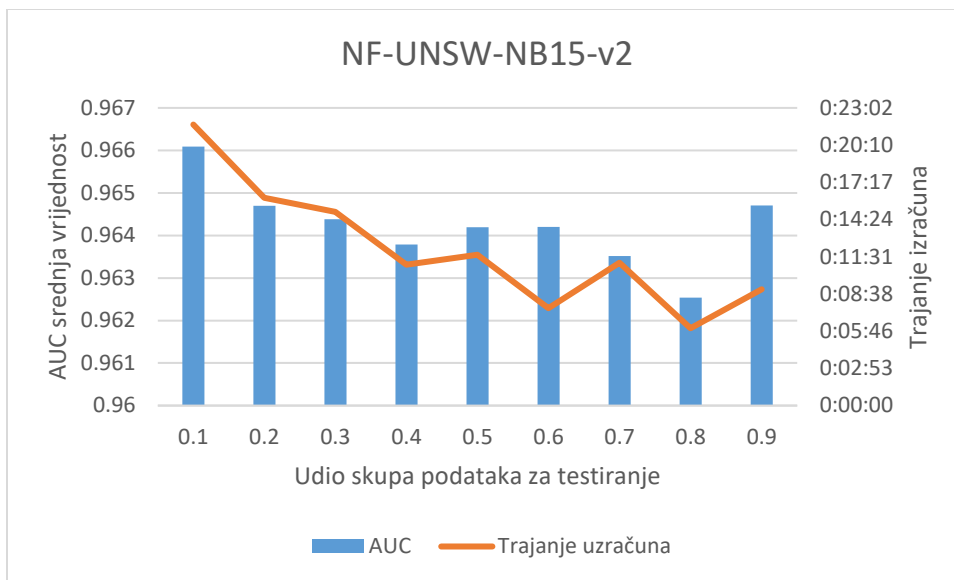
Slika 4.5. Rezultati AUC-točnosti i vremena izračuna u odnosu na omjer podskupova za učenje i testiranje na CSE-CIC-IDS2018-skupu podataka



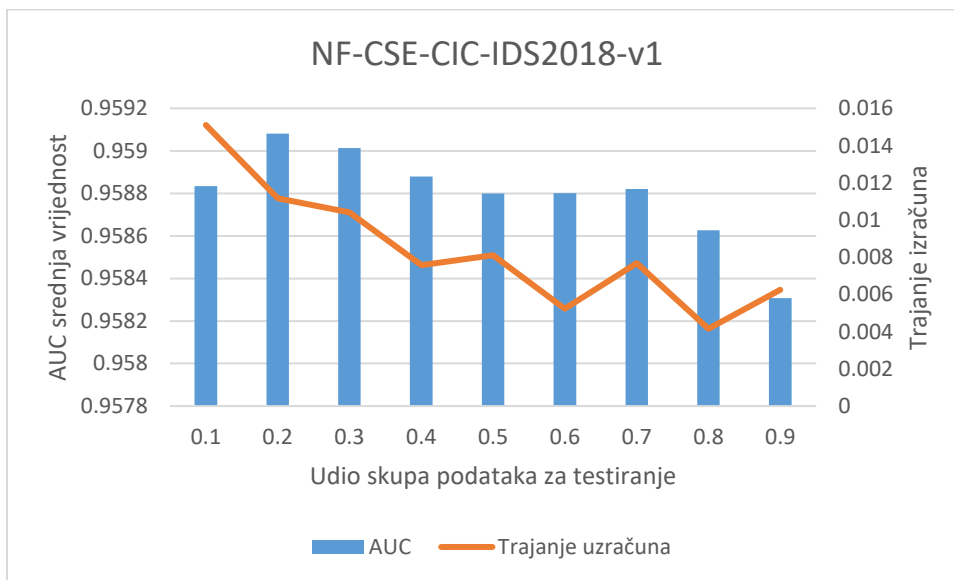
Slika 4.6. Rezultati AUC-točnosti i vremena izračuna u odnosu na omjer podskupova za učenje i testiranje na LUFLOW2021-skupu podataka



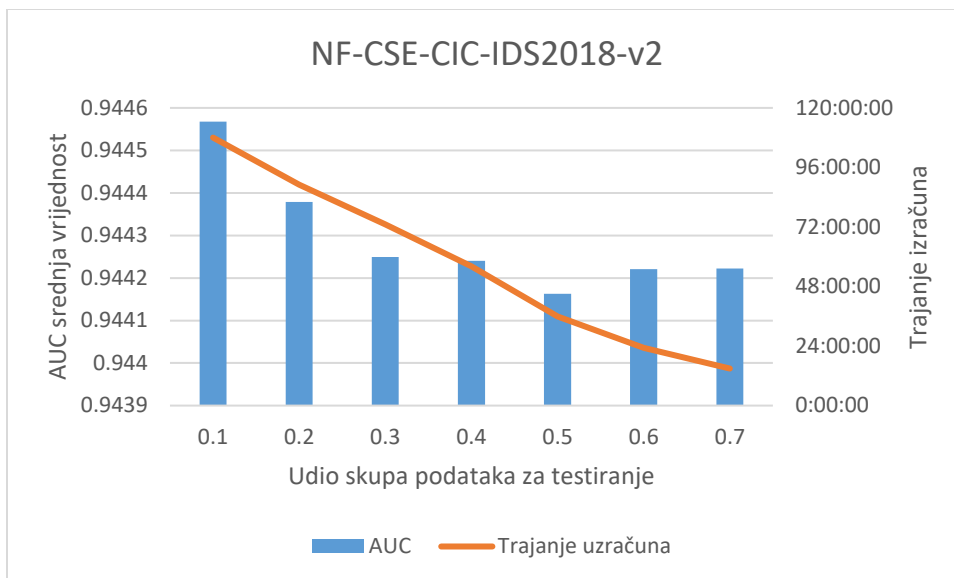
Slika 4.7. Rezultati AUC-točnosti i vremena izračuna u odnosu na omjer podskupova za učenje i testiranje na NF-UNSW-NB15-v1-skupu podataka



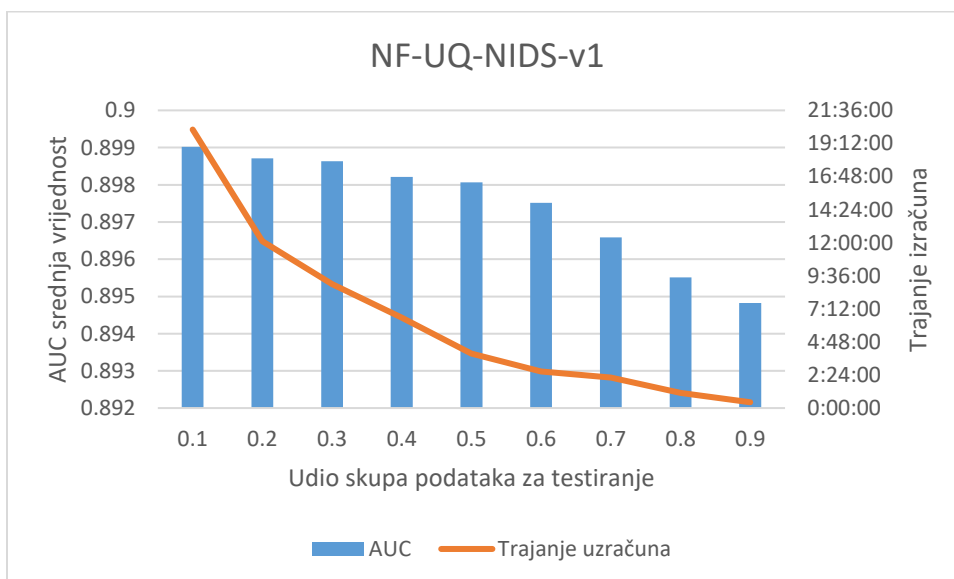
Slika 4.8. Rezultati AUC-točnosti i vremena izračuna u odnosu na omjer podskupova za učenje i testiranje na NF-UNSW-NB15-v2-skupu podataka



Slika 4.9. Rezultati AUC-točnosti i vremena izračuna u odnosu na omjer podskupova za učenje i testiranje na NF-CSE-CIC-IDS2018-v1-skupu podataka



Slika 4.10. Rezultati AUC-točnosti i vremena izračuna u odnosu na omjer podskupova za učenje i testiranje na NF-CSE-CIC-IDS2018-v2-skupu podataka



Slika 4.11. Rezultati AUC-točnosti i vremena izračuna u odnosu na omjer podskupova za učenje i testiranje na NF-UQ-NIDS-v1-skupu podataka

Eksperimentalno određivanje najboljeg omjera razdvajanja skupa podataka za učenje i testiranje pokazuje da su odstupanja u AUC-točnosti vrlo mala (razlika u točnosti je u rasponu 0,1 % - 4,4 %) u ovisnosti o definiranom omjeru. Dodatni je parametar, ali ne i ključni, vrijeme izračunavanja koje se koristi kako bi se usporedile prednosti smanjivanja ili povećanja omjera

podjele skupa podataka. Na osnovu dobivenih rezultata, najbolja AUC-točnost postignuta je omjerom 90/10, odnosno 90 % podataka koristi se za učenje, dok se 10 % koristi na testiranje klasifikacije. Omjer 80/20 također ima visoku AUC-točnost, ali i značajno ukupno smanjenje trajanja izračuna. Pri tom omjeru, točnost se prosječno smanjuje za 0,1 %, a vrijeme potrebno za izračune smanjuje se u prosjeku za 24 %. Tablica 4.9. prikazuje postotak promjene AUC-točnosti i promjene trajanja izračuna među omjerima 90/10 i 80/20. Pri izračunavanju srednje vrijednosti, uzela se apsolutna vrijednost promjene. Srednja vrijednost promjene AUC-točnosti i vremena izračuna prikazana je u zadnjem redu tablice 4.9. Negativan predznak Δ AUC (%) kod određenih skupova podataka označava bolju točnost pri omjeru 80/20, nego 90/10 što je vidljivo iz prethodnih grafikona potpoglavlja.

Tablica 4.9. Usporedba smanjenja AUC-točnosti i vremena izračuna među omjerima 90/10 i 80/20 trening/test podataka

skup podataka	Δ AUC (%)	Δ vrijeme izračuna (%)
UNSWB-NB15	0,03	28,31
NF-CSE-CIC-IDS2018-v2	0,0199	17,53
NF-UNSW-NB15-v2	0,1446	26,05
NF-CSE-CIC-IDS2018	-0,0257	34,54
NF-UQ-NIDS	0,0347	40,11
NF-UNSW-NB15	0,5922	14,8
LUFlow2021	-0,0078	20,80
CSE-CIC-IDS2018	-0,0123	13,4
	0,1084	24,4425

Uzevši u obzir te činjenice i usporedbu omjera korištenih u sličnim istraživanjima u daljnjem razvoju modela, koristio se omjer **80/20** (trening/test). Smanjenje u AUC-točnosti odabirom tog omjera nadoknadio se u dijelu finog podešavanja hiperparametara odabranog klasifikatora.

4.1.2.6. Skaliranje značajki

Skaliranje značajki ulaznog skupa podataka preporučuje se u predobradi podataka tijekom izgradnje modela strojnog učenja, jer može značajno poboljšati performanse modela i učiniti ga učinkovitijim. Pri korištenju algoritama strojnog učenja, ulazne vrijednosti moraju biti numeričke vrijednosti. Ako postoji znatna razlika u rasponima vrijednosti među pojedinim značajkama,

pretpostavka je da značajke s većim rasponom imaju veći utjecaj, što znači da će brojevi s većim rasponom imati važniju ulogu tijekom učenja modela. Ako raspon vrijednosti neobrađenih podataka uvelike varira, u nekim algoritmima strojnog učenja ciljne funkcije ne rade ispravno bez normalizacije odnosno skaliranja. *Python sklearn.preprocessing* dio je biblioteke otvorenog koda za strojno učenje koji pruža različite tehnike za poboljšanje podataka, uključujući skaliranje, kodiranje kategorijskih varijabli, normalizaciju i mnoge druge operacije koje su ključne u pripremi podataka za strojno učenje. Neke od uobičajenih funkcija skaliranja vrijednosti značajki su *Min Max Scaler*, *Standard Scaler* i *Robust Scaler* i koriste se u sličnim istraživanjima [66], [105], [110], [115]. Kako bi se odredila najbolja funkcija skaliranja, uspoređene su funkcije skaliranja na svim referentnim skupovima podataka kako bi se utvrdio njihov utjecaj na povećanje točnosti klasifikatora.

Standardni skaler

Standardno skaliranje (engl. *Standard scaling*) jedna je od najčešćih tehnika skaliranja značajki u strojnom učenju. Postupak standardnog skaliranja uključuje oduzimanje srednje vrijednosti svake značajke iz svakog uzorka i zatim podjelu sa standardnom devijacijom. To osigurava da se sve vrijednosti značajki transformiraju na istu skalu, čime se olakšava uspoređivanje među značajkama. Standardni rezultat uzorka x izračunava se prema [126] na sljedeći način:

$$z = (x - u)/s \quad (4-1)$$

gdje je u srednja vrijednost uzoraka za učenje, a s je standardna devijacija uzoraka za učenje. Primjena standardnog skaliranja osobito je korisna za modele koji zahtijevaju normalnu raspodjelu podataka, poput mnogih linearnih modela, kao i nekih metoda klasteriranja i neuronskih mreža [126].

Robustni skaler

Robustno skaliranje (engl. *Robust scaling*) je tehnika skaliranja značajki u strojnom učenju koja se koristi kada je potrebno skalirati podatke s anomalijama, tj. odstupanjima od uobičajenog obrasca distribucije podataka. Taj skaler uklanja medijan i mjeri podatke prema interkvartilnom rasponu IQR (raspon među 1. kvartilom i 3. kvartilom). Centriranje i skaliranje odvijaju se neovisno o svakoj značajki izračunavanjem relevantne statistike o uzorcima u skupu za učenje.

Robustno skaliranje osobito je korisno za modele koji nisu otporni na anomalije, poput mnogih linearnih modela koji se temelje na srednjoj vrijednosti i standardnoj devijaciji [127].

MinMax-skaler

MinMax-skaliranje (engl. *MinMax scaling*) je tehnika u strojnom učenju koja se koristi za transformiranje podataka. Njena je svrha prilagoditi sve značajke u skupu za učenje unutar određenog opsega. Uobičajeno se primjenjuju dva opsega: od 0 do 1 kada su vrijednosti značajki pozitivne, i od -1 do 1 kada su neke vrijednosti značajki negativne. Mogu se postaviti i neki proizvoljni rasponi poput [0,5]. Taj se skaler dobro ponaša ako je standardna devijacija mala i kada distribucija uzoraka nije Gaussova, a često se koristi za normalizaciju podataka kako bi se osigurala jednaka važnost svake značajke u skupu podataka. Vrlo je osjetljiv na anomalije, a postupak MinMax-skaliranja uključuje oduzimanje minimalne vrijednosti svake značajke iz svakog uzorka i zatim podjelu s rasponom vrijednosti značajke (razlikom među maksimalnom i minimalnom vrijednosti). Transformacija je prema [128] definirana kao:

$$x_n = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (4-2)$$

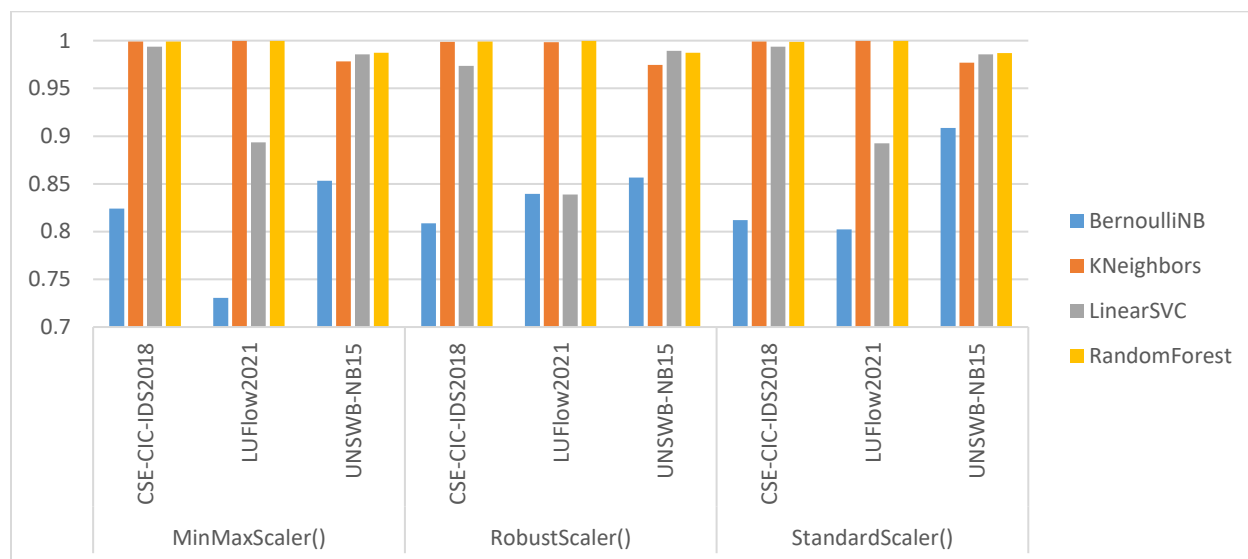
4.1.2.7. Utjecaj algoritma skaliranja na točnost klasifikacije

Kada se usporede AUC-vrijednosti klasifikacije četiri klasifikatora na svim skupovima podataka, uz primjenu svih triju spomenutih tehnika skaliranja, moguće je utvrditi koja je tehnika skaliranja najbolja za primjenu na tim skupovima podataka. Na grafikonima (slika 4.12. i slika 4.13.) su prikazane AUC-vrijednosti pojedinih klasifikatora prema različitim skupovima podataka. Važno je napomenuti da određeni algoritmi skaliranja nisu primjenjivi na svim skupovima podataka, kao što je slučaj s NF-CSE-CIC-IDS2018-v2 gdje su izračuni AUC-vrijednosti automatski prekinuti tijekom izračunavanja zbog manjka računalnih resursa, stoga ti rezultati algoritama skaliranja nisu uvršteni na prikazanim grafikonima. Temeljem rezultata i usporedbe AUC-vrijednosti na svim cjelovitim skupovima podataka, zaključuje se da je MinMax-skaler najbolji izbor za skaliranje značajki (zbog zaokruživanja vrijednosti na četiri decimale RobustScaler i StandardScaler za RF algoritam u tablici 4.10. imaju jednake vrijednosti). Ta

tehnika može se primijeniti na sve skupove podataka, a također pokazuje najveću točnost na većini skupova podataka. U tablici 4.10. prikazane su vrijednosti na cjelovitim skupovima podataka gdje se prema točnosti ističe klasifikator Slučajne šume koristeći MinMax-algoritam skaliranja podataka na skupu podataka LUFlow2021.

Tablica 4.10. Usporedba AUC-vrijednosti i tehnika skaliranja s cjelovitim skupovima podataka

metoda skaliranja	skup podataka	algoritam			
		<i>BNB</i>	<i>K-NN</i>	<i>LSVC</i>	<i>RF</i>
MinMaxScaler	CSE-CIC-IDS2018	0,8239	0,9990	0,9939	0,9990
	LUFlow2021	0,7306	0,9997	0,8933	0,9999
	UNSWB-NB15	0,8533	0,9781	0,9856	0,9874
RobustScaler	CSE-CIC-IDS2018	0,8086	0,9986	0,9737	0,9991
	LUFlow2021	0,8394	0,9983	0,8389	0,9999
	UNSWB-NB15	0,8568	0,9747	0,9892	0,9872
StandardScaler	CSE-CIC-IDS2018	0,8121	0,9990	0,9936	0,9988
	LUFlow2021	0,8021	0,9996	0,8924	0,9999
	UNSWB-NB15	0,9085	0,9770	0,9855	0,9871

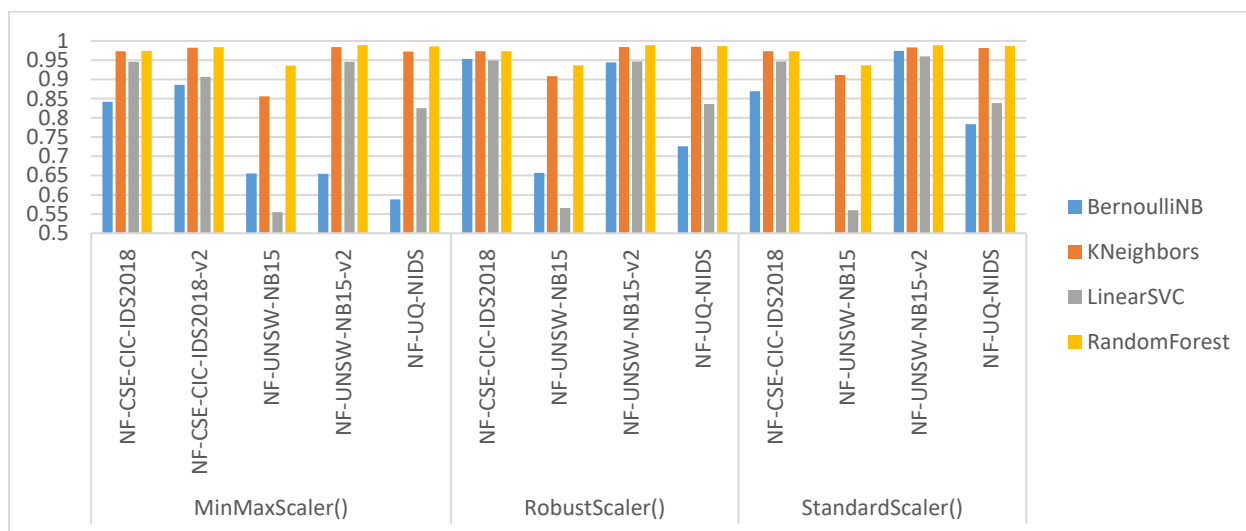


Slika 4.12. Usporedba AUC-vrijednosti i tehnika skaliranja s cjelovitim skupovima podataka

U tablici 4.11. prikazane su vrijednosti na NetFlow-skupovima podataka gdje se prema točnosti također ističe klasifikator slučajne šume koristeći MinMax algoritam skaliranja podataka na skupu podataka NF-UNSW-NB15-v2.

Tablica 4.11. Usporedba AUC-vrijednosti i tehnika skaliranja s NetFlow-skupovima podataka

metoda skaliranja	skup podataka	algoritam			
		<i>BNB</i>	<i>K-NN</i>	<i>LSVC</i>	<i>RF</i>
<i>MinMaxScaler</i>	NF-CSE-CIC-IDS2018	0,8418	0,9728	0,9454	0,9738
	NF-CSE-CIC-IDS2018-v2	0,8859	0,9821	0,9067	0,9838
	NF-UNSW-NB15	0,6550	0,8563	0,5544	0,9361
	NF-UNSW-NB15-v2	0,6548	0,9839	0,9455	0,9890
	NF-UQ-NIDS	0,5881	0,9720	0,8255	0,9855
<i>RobustScaler</i>	NF-CSE-CIC-IDS2018	0,9536	0,9730	0,9491	0,9732
	NF-UNSW-NB15	0,6567	0,9080	0,5654	0,9366
	NF-UNSW-NB15-v2	0,9438	0,9836	0,9465	0,9889
	NF-UQ-NIDS	0,7263	0,9845	0,8363	0,9864
<i>StandardScaler</i>	NF-CSE-CIC-IDS2018	0,8690	0,9733	0,9465	0,9731
	NF-UNSW-NB15	0,5000	0,9119	0,5599	0,9368
	NF-UNSW-NB15-v2	0,9738	0,9829	0,9601	0,9888
	NF-UQ-NIDS	0,7833	0,9813	0,8383	0,9864



Slika 4.13. Usporedba AUC-vrijednosti i tehnika skaliranja s NetFlow-skupovima podataka

Nakon provedenog testiranja primjećuju se vrlo ujednačene i najbolje AUC-vrijednosti za klasifikatore slučajna šume i K-najbližeg susjeda za sve tri primijenjene metode skaliranja. Ukupno gledajući, MinMax-skaliranje najprihvatljivija je metoda skaliranja te su se tom metodom skalirali podatci stvarnog NetFlow-prometa u daljnjem istraživanju i eksperimentalnim postupcima strojnog učenja.

4.1.3. Odabir referentnog skupa podataka i klasifikatora

Kako bi se odabrao najbolji klasifikator i poslije uspješno koristio u modelu za otkrivanje anomalija mrežnog prometa, bitno je uspješnost klasifikatora provjeriti na više dostupnih javnih skupova podataka. U izračunima su se koristila tri cjelovita skupa podataka: UNSW-NB15, CSE-CIC-IDS2018, LUFlow2021 te NetFlow-prilagođeni skupovi podataka: NF-UNSW-NB15-v1, NF-UNSW-NB15-v2, NF-CSE-CIC-IDS2018-v1, NF-CSE-CIC-IDS2018-v2, NF-UQ-NIDS-v1. Skupovi podataka detaljno su opisani u poglavljima 4.1.1.1. – 4.1.1.2. Svi skupovi podataka podvrgnuti su predobradi, no razlike su bile u primjeni ovisno o njihovoj strukturi i potrebama. Skupovi podataka koji su već bili pripremljeni (NetFlow) i kodirani u numeričke vrijednosti, zahtijevali su manje prilagodbe u odnosu na cjelovite skupove podataka koji su bili u svom izvornom obliku. Takvim pristupom i korištenjem cjelovitih i NetFlow-skupova podataka bilo je moguće usporediti ispravnost postupka predobrade osobito u dijelu kodiranja kategorijskih značajki. Svaka etapa predobrade podataka radila se na svim skupovima podataka te se temeljem najboljih AUC-vrijednosti određivao najbolji odabir promatrane predobrade.

Tijekom svih ispitivanja utjecaja na točnost klasifikacije u procesu predobrade podataka, koristila su se četiri najčešća predstavnika klasifikatora prema podjeli algoritama klasifikacije nadziranog strojnog učenja opisanim u poglavlju 3.4. Ispitivanjem utjecaja više čimbenika u predobradi podataka opisanih u poglavlju 4.1.2. utvrđeni su najbolji čimbenici kako slijedi:

- brisanje suvišnih značajki
- čišćenje neispravnih vrijednosti
- kodiranje kategorijskih značajki metodom kodiranja labelama
- 80/20 omjer podataka za učenje i testiranje
- skaliranje podataka MinMax-metodom skaliranja.

Temeljem odabranih čimbenika predobrade skupova podataka, provedena je klasifikacija s četirima različitim klasifikatorima te je temeljem rezultata klasifikacije odabran referentni skup podataka, kao i najuspješniji klasifikator. Proces ispitivanja rađen je prema pseudokodu Algoritam 1., a implemetiran u programskom jeziku *Python*.

Algoritam 1: Odabir referentnog skupa i najboljeg klasifikatora

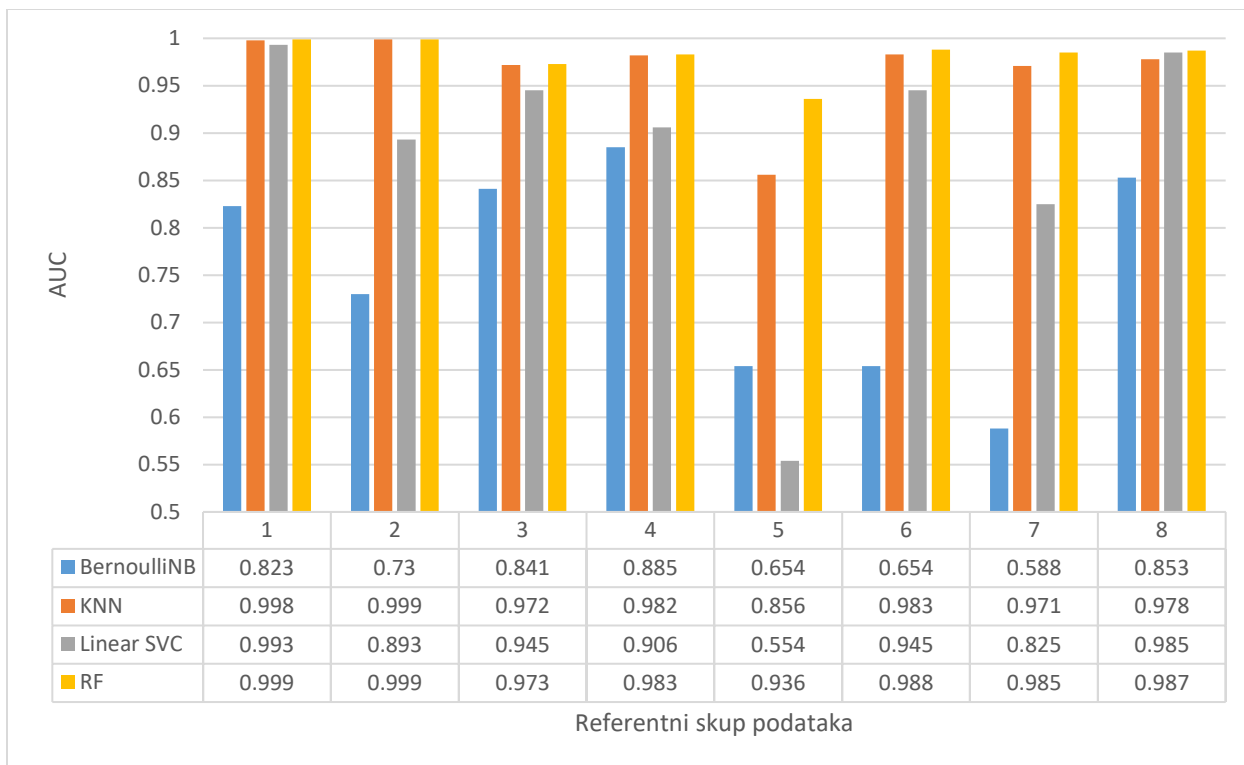
Input1: javno dostupni skupovi podataka CSE-CIC-IDS2018, LUFlow2021, UNSWB-NB15, NF-CSE-CIC-IDS2018, NF-CSE-CIC-IDS2018-v2, NF-UNSW-NB15, NF-UNSW-NB15-v2, NF-UQ-NIDS

Input2: klasifikatori BNB, K-NN, LSVC, RF

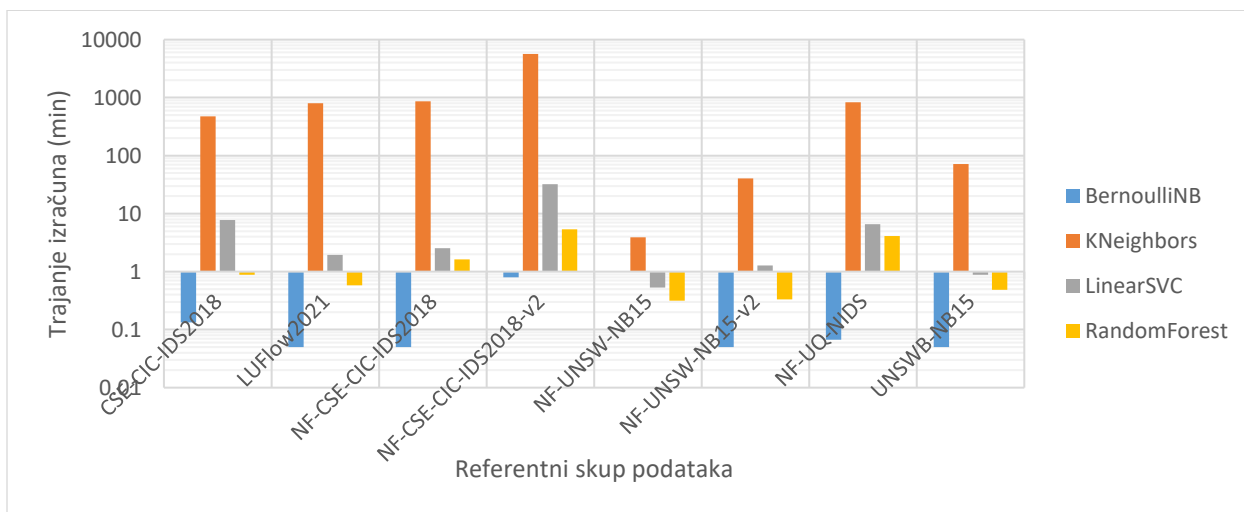
Output: AUC

1. **Za svaki Input1 ponovi:**
 2. **Function:** učitavanje podataka (csv)
 3. **Return:** data
 4. **Function:** Predobrada podataka (data)
 5. **Return:** data_
 6. **Function:** razdvajanje podataka (data, omjer razdvajanja)
 7. $X \leftarrow$ skup podataka svih značajki bez oznake klase
 8. $y \leftarrow$ značajka oznake klase
 9. **Return** X_train, X_test, y_train, y_test
 10. **Function:** Skaliranje podataka
 11. **Return** X_train_scale, X_test_scale, y_train, y_test
 12. **Za svaki Input2 ponovi:**
 13. $AUC \leftarrow$ **klasifikacija** (X_train_scale, X_test_scale, y_train, y_test)
 14. **Ponovi**
 15. **Ponovi**
 16. Kraj
-

Na slici 4.14. prikazani su rezultati izračuna točnosti klasifikacije uz AUC-vrijednosti za sva četiri klasifikatora na promatranim javno dostupnim skupovima podataka. Rezultati pokazuju da je klasifikator slučajne šume najuspješniji. Na skupu podataka LUFlow2021, taj klasifikator postiže izuzetno visoku AUC-točnost od 99,98 %. Najlošije rezultate ostvaruje Bernoulli Naive Bayes klasifikator koji prema tim rezultatima nije pogodan za rješavanje problema klasifikacije s tim tipom podataka. Nešto slabiji rezultat ima klasifikator K-najbližeg susjeda, no vrijeme utrošeno pri klasifikaciji tim klasifikatorom, daleko premašuje vrijeme potrebno za klasifikaciju s klasifikatorom slučajne šume kao što prikazuje slika 4.15.



Slika 4.14. Usporedba uspješnosti klasifikatora na svim skupovima podataka



Slika 4.15. Usporedba vremena izračuna AUC-točnosti klasifikacije na svim skupovima podataka

Na slici 4.15. prikazana je logaritamska raspodjela trajanja izračuna jer su vrijednosti izračuna za klasifikator K-najbližeg susjeda izrazito velike u odnosu na trajanje izračuna ostalih klasifikatora. Neki klasifikatori u *scikit-learn*-platformi imaju mogućnost korištenja i raspodjelu izračuna na sve dostupne procesore te je ta mogućnost znatno poboljšala performanse tijekom izračunavanja. Iako je ta mogućnost raspoloživa i za klasifikator K-najbližeg susjeda, vrijeme izračuna je i dalje mnogostruko veće od ostalih klasifikatora, neovisno o kojem se referentnom skupu podataka radi. Bernoulli Naive Bayes klasifikator izdvaja se po najkraćem vremenu za izračunavanje, ali točnost koju postiže pri klasifikaciji čini ga neprihvatljivim za implementaciju, dok se klasifikator stroja s potpornim vektorom ne izdvaja niti prema točnosti klasifikacije niti prema vremenu potrebnom za izračunavanje.

Tijekom istraživanja za odabir i prilagodbu ulaznih podataka u svim etapama opisanim u prethodnim poglavljima, najbolji su rezultati redovito postignuti korištenjem klasifikatora slučajne šume. Osim najveće AUC-točnosti koja se uzimala kao referentna mjera zbog već ranije opisanih svojstava skupa podataka koji je izrazito nebalansiran, izračunavane su i ostale mjere točnosti opisane u poglavljima 3.5.1. – 3.5.9. Uzevši u obzir i te mjere točnosti, klasifikator slučajne šume postizao je u najvećem dijelu najbolje rezultate. Uz postizanje najbolje točnosti, svaki je puta bio pri vrhu najboljih rezultata vezano za trajanje izračuna. Najbolji primjer pokazuje slika 4.15. gdje su na logaritamskoj skali prikazana vremena potrebna za izračunavanje svih četiriju algoritama na svim skupovima podataka. Kraća vremena izračunavanja imao je jedino Bernoulli Naive Bayes klasifikator, čija je točnost vrlo mala i u nekim slučajevima graniči sa slučajnim pogađanjem klasa. Stoga se može reći da klasifikator slučajne šume i u ovom segmentu postiže najbolje rezultate. U početnim razmatranjima, vrijeme izračuna nije bilo važno pri odabiru klasifikatora, ali kako se naučeni model ne može ažurirati i prilagoditi novim podacima u stvarnom vremenu, potrebno je ponovno provesti postupak učenja s novim skupovima podataka. Stoga, vrijeme izračuna postaje važan čimbenik kako bi se model što prije ažurirao i prilagodio novim podacima. Bitan čimbenik kod brzine izračunavanja je mogućnost korištenja svih raspoloživih računalnih resursa (na infrastrukturi korištenog virtualnog poslužitelja ukupno je bilo raspoloživo 32 procesora). Prema dokumentaciji *scikit-learn*a, jedino klasifikatori Bernoulli NB i Stroj s potpornim vektorima (LinearSVC) nisu mogli koristiti tu mogućnost, ali i bez toga, Bernoulli NB je postizao najbolja vremena pri izračunavanju. Zapažen je vrlo velik utjecaj skaliranja podataka gdje su se uvelike

smanjila vremena izračunavanja posebno kod klasifikatora stroja s potpornim vektorima (LinearSVC).

Uspoređujući s relevantnim istraživanjima otkrivanja anomalija putem strojnog učenja posljednjih godina, prikazanim u tablici 4.12., predloženi klasifikator slučajne šume (RF), korišten na referentnom skupu podataka LUFLOW2021, pokazuje bolje rezultate. Taj pristup ne samo da postiže visoku točnost otkrivanja anomalija korištenjem prikladne AUC-metrike već pruža skalabilnost i učinkovitost u obradi velikih i kompleksnih, prije svega neuravnoteženih skupova podataka, čime se potvrđuje kao pouzdana metoda za otkrivanje anomalija u realnim mrežnim okruženjima.

Legende kratica u tablici 4.12.

ML algoritmi:

RF-Random Forest, LSVC-Linear Support Vector Machines, KNN-K-Nearest Neighbour, BNB-Bernoulli Naive Bayes, ANN-Artificial Neural Network, SVM-Support Vector Machines, NB-Naive Bayes, DT-Decision Tree, GBT-Gradient Boosting Tree, SGD-Stochastic Gradient Descent, LR-Logistic Regression

Skupovi podataka:

1-UNSW-NB15, 2-CIC-IDS2018, 3-LUFLOW2021, 4-NF-UNSW-NB15, 5-NF-CIC-IDS2018, 6-NF-LUFLOW2021, 7-NF-UQ-NIDS, 8-NSL-KDD, 9-CIDD5-001, 10-ISCX2012, 11-CIC-IDS2017, 12-LUFLOW2020, 13-ICS, 14-CTU-13, 15-PantHoney

Evaluacijske metrike:

AUC-Area Under ROC Curve, F2-F2 Score, F1-F1 Score, A-accuracy, BA-Balanced accuracy, R-Recall, P-Precision, CK-Cohen kappa, M-Methews coef., FPR-False Positive Rate, TPR-True Positive Rate, T-time, MSE-Mean Squared Error

Tablica 4.12. Usporedba relevantnih istraživanja otkrivanja anomalija strojnim učenjem

#	godina	ML-algoritmi	skup podataka	cijeli skupovi podataka	evaluacijske metrike	najbolji rezultat
predloženi model		RF , LSVC, KNN, BNB	1,2,3,4,5,6,7	DA	AUC , F2, A, BA, R, P, CK, M, T	99,98 %
[66]	2020.	ANN , SVM, NB, RF, KNN	1,8	NE	A , FPR, T	99,69 %
[101]	2022.	RF , DT, SVM, NB	2,3,11,12	NE	A , P, R, F1	99,91 %
[129]	2018.	RF , SVM, ANN	1	NE	A , CK, FPR, P, R, AUC , T	99,10 %
[130]	2022.	GBT , RF, NB, DT, LSVM	12	NE	A , F1, P , R	99,98 %
[123]	2020.	NB, K-NN, DT, RF	1,11,13	DA	A , P, R, FPR, F1, AUC	99,90 %
[131]	2018.	NB, SVM, RF	1	NE	AUC , TPR, FPR	98,10 %
[121]	2020.	KNN, LR, NB, DT, SGD, RF	1	NE	P , MSE, R, F1, A , TPR, FPR	95,43 %
[132]	2019.	RF, DT-J48	14, 15	NE	A , P, FPR, TPR, F1	97,47 %

Takvi rezultati predloženog modela klasifikacije potvrđuju odabir klasifikatora **slučajne šume (RF)** kao referentnog klasifikatora za model otkrivanja anomalija mrežnog prometa uz predloženi odabir i prilagodbu ulaznih podataka. Klasifikator slučajne šume u većini istraživanja takvog tipa pokazuje najbolje rezultate koristeći različite skupove podataka.

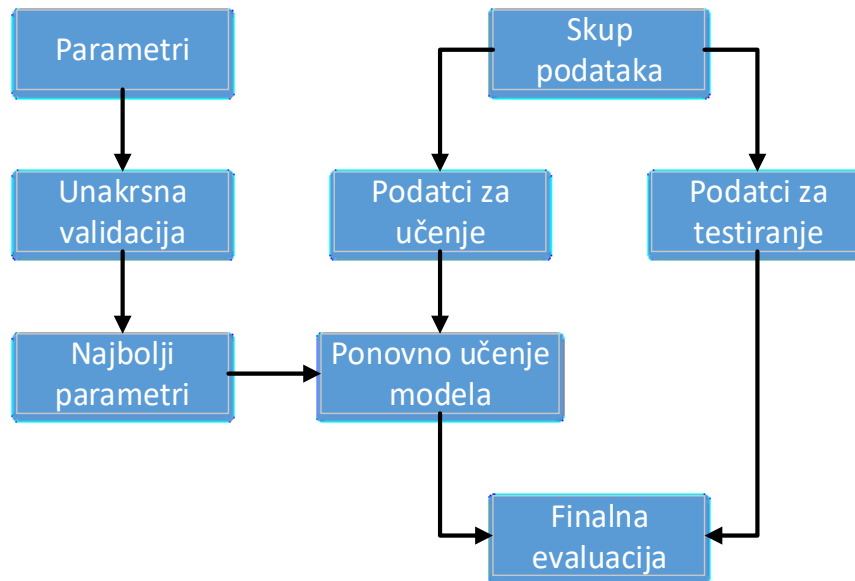
U dostupnoj literaturi primijećeno je da su referentni skupovi podataka starijeg datuma, poput NSL-KDD, dominantni, ali s vremenom se bilježi povećanje uvođenja novijih skupova podataka, što se potvrđuje i analizom nedavno provedenih istraživanja prikazanih u tablici 4.12. Među novijim skupovima podataka ističe se LUFlow, koji je ažuriran novim zapisima od 2020. do 2022. godine, što je obuhvaćeno vremenom nastanka ove disertacije. Kada je riječ o postignutim rezultatima, klasifikator Slučajne šume ostvario je najbolje rezultate kada je primijenjen na cjeloviti skup podataka **LUFlow2021**, što ga čini ključnim odabirom pri izradi modela za otkrivanje anomalija u etapi učenja.

4.1.4. Odabir hiperparametara klasifikatora

Određivanje parametara funkcije predviđanja i testiranja na istim podacima metodološka je pogreška. Osnovno načelo procjene uspješnosti klasifikatora je da se uzorci koji se koriste za učenje ne smiju koristiti za ocjenu uspješnosti. Slično pravilo vrijedi i za metode odabira hiperparametara koje zahtijeva većina metoda strojnog učenja, na primjer, *broj stabala* za algoritam slučajne šume, *sigma* i *C* vrijednosti za algoritme strojeva s potpornim vektorima, te *k* koeficijent za algoritam K-najbližeg susjeda. Vrijednost tih hiperparametara može utjecati na točnost klasifikacije, a samim time je potrebno optimiziranje odabranih vrijednosti. Ugađanje (engl. *tuning*) hiperparametara je obično empirijski proces s različitim vrijednostima i kombinacijama hiperparametara. Pretpostavlja se da su vrijednosti hiperparametara koje generiraju najveću ukupnu točnost ili neku drugu vrstu metrike optimalne za promatrani model odnosno klasifikator [133].

4.1.4.1. Unakrsna validacija

Model koji bi savršeno reproducirao oznake uzoraka na kojima je treniran, imao bi idealan rezultat na tim podacima, ali bi se lošije nosio s novim, neviđenim podacima. Taj problem naziva se prekomjerno prilagođavanje (engl. *overfitting*). Da bi se to izbjeglo, uobičajena je praksa u nadziranom strojnom učenju zadržati dio dostupnih podataka kao testni skup (X_{test} , y_{test}), na kojem se provjeravaju performanse modela na novim podacima i sprječava prekomjerno prilagođavanje. Na slici 4.16. prikazan je dijagram toka tipičnog tijeka unakrsne validacije pri procesu učenja modela. Najbolji parametri mogu se odrediti tehnikama pretraživanja poput *GridSearchCV*.

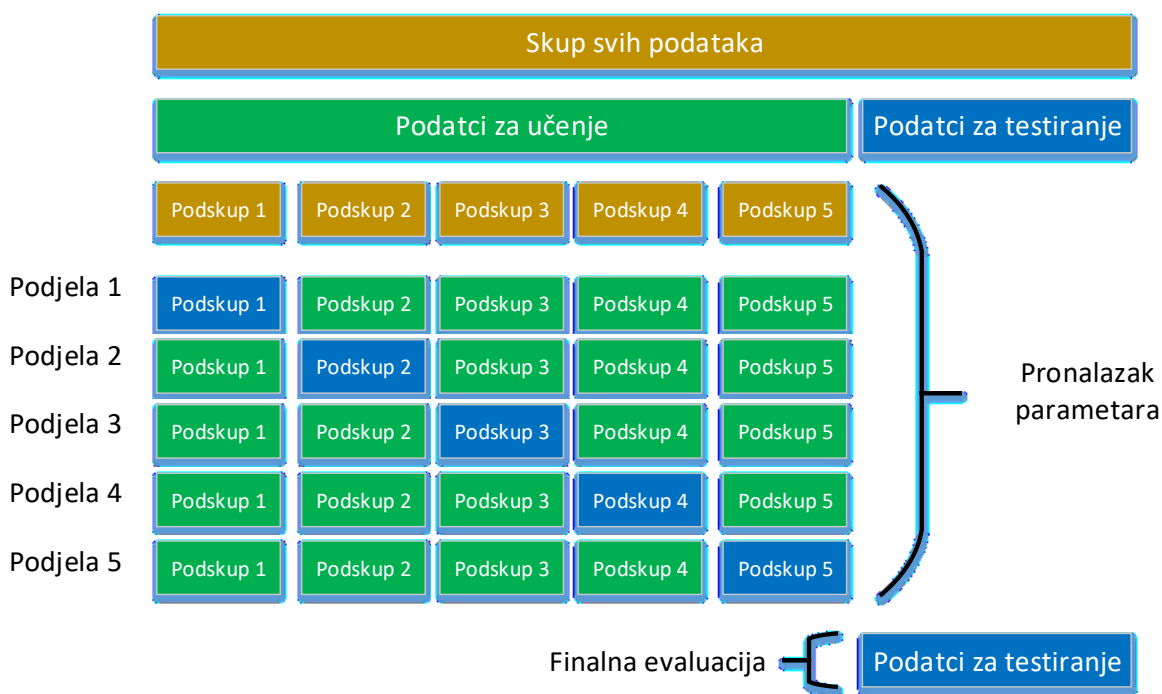


Slika 4.16. Unakrsna validacija i određivanje najboljih parametara modela [134]

Pri evaluaciji različitih postavki (hiperparametara) za klasifikatore, još uvijek postoji rizik od prekomjernog prilagođavanja testnog skupa jer znanje o skupu testnih podataka može "procuriti" u model. Da bi se riješio taj problem, još jedan dio skupa podataka može se predstaviti kao tzv. "skup podataka za provjeru valjanosti". Učenje se izvodi na skupu podataka za učenje, nakon čega se vrši evaluacija na skupu za provjeru valjanosti i kada se postigne zadovoljavajući rezultat, konačna se procjena radi na testnom skupu podataka. Partitioniranjem dostupnih podataka u tri podskupima (učenje, test, validacija), drastično se smanjuje broj uzoraka koji se mogu koristiti za učenje modela, a rezultati mogu ovisiti o određenom slučajnom izboru podskupova. Rješenje je tog problema postupak koji se zove unakrsna validacija (engl. *Cross-Validation*), kako je prikazano slikom 4.17. Testni se skup podataka i dalje zadržava za konačnu evaluaciju, ali skup za provjeru valjanosti više nije potreban u procesu unakrsne validacije. U temeljnom pristupu poznatom kao unakrsna validacija k -podskupovima (engl. *k-fold CV*), skup je podataka za učenje podijeljen na nekoliko manjih podskupova, odnosno k podskupova, nakon čega se postupak učenja primjenjuje na svaki od tih podskupova zasebno:

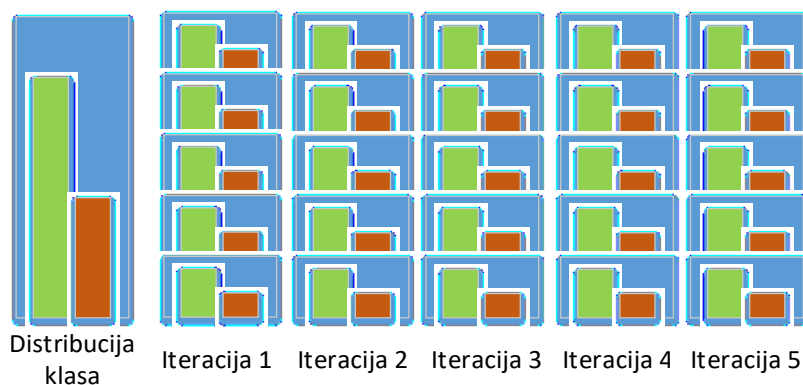
- model se trenira korištenjem podataka iz $k-1$ podskupa
- model se provjerava na preostalom dijelu podataka (tj. koristi se kao testni skup za izračunavanje mjere uspješnosti izvedbe kao što je točnost).

Mjera uspješnosti izvedbe koja se dobije unakrsnom provjerom k-podskupovima prosjek je vrijednosti izračunatih u petlji [134].



Slika 4.17. Određivanje najboljih hiperparametara klasifikatora unakrsnom validacijom [134]

Kako bi bila osigurana pouzdanost i generalizacija predloženog modela, primijenjena je unakrsna provjera k-podskupovima, koja je često korištena u području strojnog učenja [133], [135], [136]. Odabirom *StratifiedKFold* kao što je prikazano na slici 4.18. postiže se jednakomjerna distribucija klasa tijekom iteracija u procesu traženja najboljih hiperparametara za klasifikator. Ta metoda osigurava da model bude pouzdan i dobro generaliziran za različite skupove podataka, a posebno je korisna za nebalansirane skupove podataka, jer osigurava da svaki podskup zadržava približno isti razmjer klasa kao i cijeli skup podataka. Time se osigurava da učenje i evaluacija modela budu pouzdaniji jer svaki preklop sadrži reprezentativnu distribuciju klasa, smanjujući rizik od pristranosti zbog nebalansiranih podataka.



Slika 4.18. Određivanje najboljih hiperparametara klasifikatora unakrsnom validacijom uz jednaku distribuciju klasa (*StratifiedKFold*) [137]

4.1.4.2. Rezultati odabira hiperparametara klasifikatora

U većini projekata strojnog učenja, provodi se učenje različitih modela na odabranom skupu podataka i bira se onaj koji pokazuje najbolje performanse. Međutim, nije uvijek moguće sa sigurnošću utvrditi optimalne hiperparametre modela za rješavanje određenog problema, stoga je cilj unaprijediti model na svaki mogući način. Hiperparametri modela mogu imati važnu ulogu u performansama modela, a postavljanjem optimalnih vrijednosti za hiperparametre može se značajno poboljšati točnost modela. Scikit-learn-paket sadrži funkciju *GridSearchCV* koja omogućuje pronalaženje optimalnih vrijednosti hiperparametara modela radi poboljšanja njegovih performansi. Treba imati na umu da ne postoji način da se unaprijed znaju najbolje vrijednosti za hiperparametre, idealno bi bilo isprobati sve moguće vrijednosti kako bi se pronašle optimalne vrijednosti. Kada ručno podešavanje i ispitivanje hiperparametara zahtijeva značajnu količinu vremena i resursa, moguće je koristiti funkciju *GridSearchCV* za automatizaciju tog procesa koja se koristi za pretraživanje unaprijed definiranih vrijednosti hiperparametara kako bi se pronašle optimalne vrijednosti koje poboljšavaju performanse modela. *GridSearchCV* je funkcija koja omogućuje definiranje rječnika hiperparametara i njihovih vrijednosti koje se mogu koristiti za određeni model. Funkcija provjerava sve kombinacije ponuđenih vrijednosti, procjenjuje model za svaku kombinaciju koristeći metodu unakrsne provjere (engl. *Cross-Validation*) te na kraju daje točnost za svaku kombinaciju hiperparametara. Nakon toga, odabire se kombinacija hiperparametara s najboljim performansama za promatrani model. Algoritam odabira najboljih

hiperparametara je tzv. "brute force"-metoda koja je vremenski vrlo zahtjevna budući da se ispituju sve kombinacije zadanih mogućih hiperparametara [138].

Ukratko su opisani važniji parametri *GridSearchCV* funkcije [139]:

- *estimator*: prosljeđuje instancu modela za koju se provjeravaju hiperparametri
- *params_grid*: rječnik koji sadrži hiperparametre i njihove vrijednosti koji se isprobavaju
- *scoring*: korištena metrika evaluacije
- *cv*: broj unakrsnih provjera za svaki odabrani skup hiperparametara
- *verbose*: detaljan ispis trenutnog rada
- *n_jobs*: broj paralelnih procesa za ovaj zadatak (vrijednost -1 koristi sve dostupne procesore).

Za odabrani algoritam klasifikacije slučajna šuma i referentni skup podataka LUFlow2021, istraženi su različiti hiperparametri kako bi se pronašle optimalne postavke koje osiguravaju najbolje rezultate točnosti. U nastavku su navedeni svi dostupni hiperparametri, pri čemu su crvenom bojom označeni parametri koji su testirani i evaluirani. Za sve ostale hiperparametre koji nisu bili ispitivani, korištene su predefinirane vrijednosti hiperparametara koje su optimizirane za postizanje najboljih rezultata. Hiperparametri koji nisu imali utjecaj na izračune točnosti, poput "*n_jobs*" ili "*verbose*", služili su samo kao pomoćne funkcije, gdje se "*n_jobs*" koristi za korištenje svih dostupnih procesora, a "*verbose*" služi za definiranje razine detalja pri ispisivanju stanja izračuna. Neki od predloženih vrijednosti hiperparametara su određeni nakon više iteracija kako bi se raspon vrijednosti mogao detaljnije odabrati. Pri optimiziranju hiperparametara korištena je funkcija:

```
GridSearchCV(estimator, params_grid, scoring='roc_auc', verbose=10, n_jobs=-1, cv=3)
```

Hiperparametri za algoritam slučajne šume

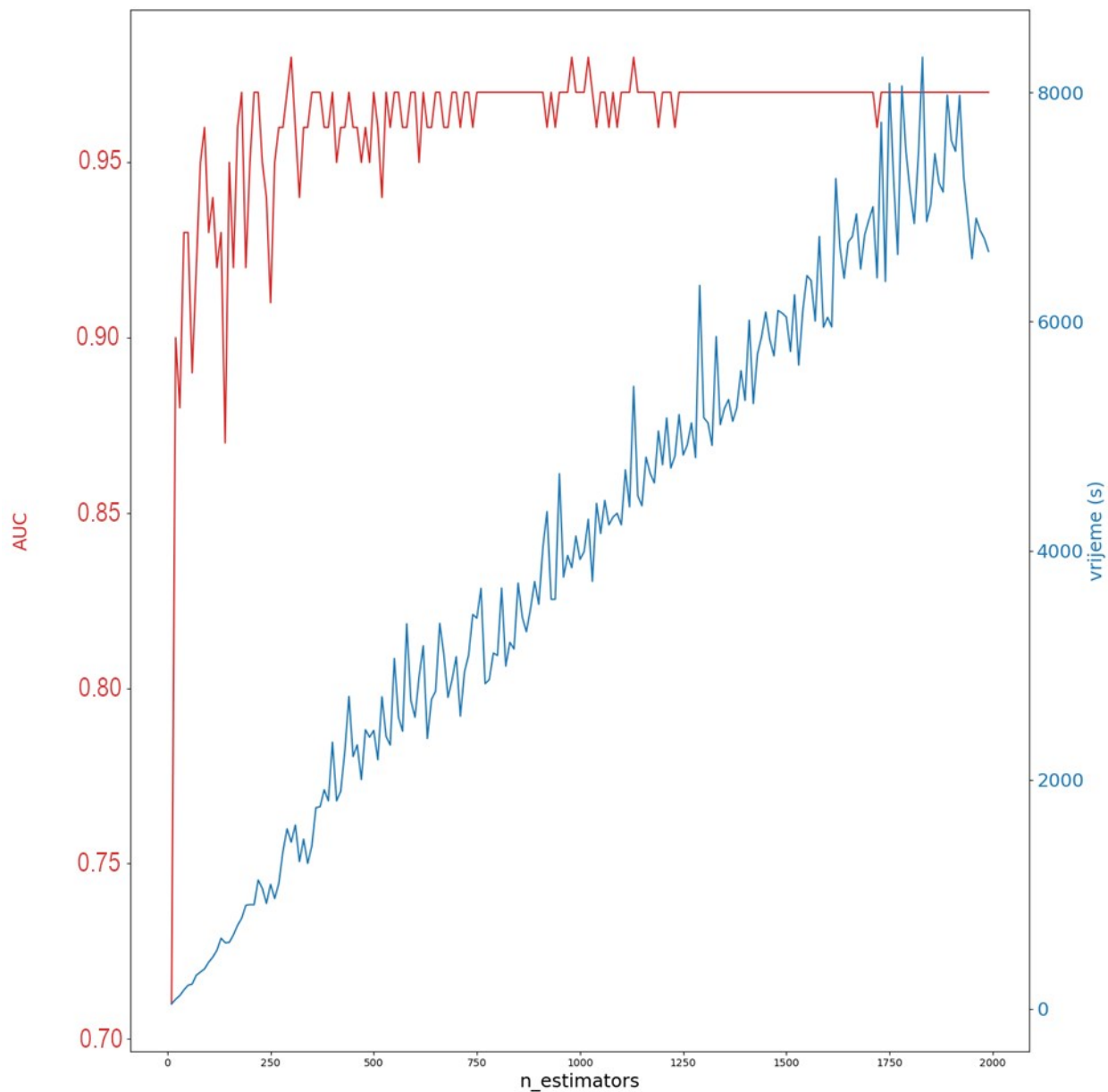
- *n_estimators*: int, default=100
- *criterion*: {"gini", "entropy", "log_loss"}, default="gini"
- *max_depth*: int, default=None
- *min_samples_split*: int or float, default=2
- *min_samples_leaf*: int or float, default=1

- *max_features*: {"sqrt", "log2", None}, int or float, default="sqrt"
- *max_leaf_nodes*: int, default=None
- *min_impurity_decrease*: float, default=0.0
- *oob_score*: bool, default=False
- *n_jobs*: int, default=None
- *random_state*: int, RandomState instance or None, default=None
- *verbose*: int, default=0
- *warm_start*: bool, default=False
- *class_weight*: {"balanced", "balanced_subsample"}, default=None
- *ccp_alpha*: non-negative float, default=0.0
- *max_samples*: int or float, default=None.

Najbolji rezultat AUC točnosti od 0,9999 postiže se pri odabiru hiperparametara:

- *n_estimators*: **1130** (ispitan raspon vrijednosti 10 – 2000, korak 10)
- *criterion*: 'entropy'
- *max_depth*: **None** (ispitan raspon 1,10,100, None vrijednost ide do maksimalne moguće vrijednosti)
- *min_samples_split*: **50** (ispitan raspon vrijednosti 2, 3, 4, 5, 10, 20, 50,100)
- *max_features*: 'sqrt'
- *class_weight*: 'balanced'.

Prema slici 4.19., koja prikazuje rezultate unakrsne validacije, crveni grafikon prikazuje AUC-vrijednosti, a plavi vrijeme potrebno za izračune. Uočena su četiri najbolja AUC-rezultata za hiperparametar *n_estimators*. U tablici 4.13. prikazana su četiri najbolja rezultata za hiperparametar *n_estimators* koja pokazuje da je razlika AUC-vrijednosti vrlo mala, gotovo zanemariva, dok su vrijednosti vremena utrošenog za izračunavanja veća pri korištenju većeg *n_estimators* parametra. Razmatrajući te rezultate, može se uzeti treći najbolji rezultat za AUC-vrijednosti odnosno vrijednost 300 za *n_estimators* hiperparametar kako bi izračuni bili što kraći, a da točnost ostane dovoljno visoko pri klasifikaciji.



Slika 4.19. Rezultati AUC-vrijednosti i vremena izračuna tijekom optimiziranja $n_estimators$ hiperparametara algoritma slučajne šume

Odabirom vrijednosti 300 za $n_estimators$ hiperparametar, vrijeme izračuna skraćuje se 3 puta, dok točnost ostaje gotovo nepromijenjena. Ostali ispitivani hiperparametri nisu imali veliki utjecaj na vrijeme izračunavanja te za njih nije rađena dodatna analiza, već su se uzeli predloženi hiperparametri.

Tablica 4.13. Najbolji rezultati optimizacije hiperparametara $n_estimators$ algoritma slučajne šume

AUC	Vrijeme izračuna (s)	Pozicija	$n_estimators$
0,999997573	5434,62	1	1130
0,999997563	4274,18	2	1020
0,999997555	1455,28	3	300
0,999997522	3851,17	4	980

Svi rezultati unakrsne validacije u ovisnosti o $n_estimators$ hiperparametru prikazani su grafikonom na slici 4.19. Primijećeno smanjenje vremena za ispitivanje vrijednosti hiperparametra " $n_estimators$ " koje se događa kada se približimo gornjoj granici ($n_estimators=2000$), uzrokovano je smanjenim opterećenjem na poslužitelju koji obavlja izračunavanja. Zbog korištenja mogućnosti raspodjele izračunavanja na sve raspoložive procesore (32), kada se približi kraj ispitivanja, dolazi do rasterećenja poslužitelja, što omogućuje daljnju raspodjelu preostalih izračuna na oslobođene procesore. Ta silazna krivulja pri kraju raspona promatranih hiperparametara javljala se svaki put na kraju ekperimentalnog postupka optimiziranja hiperparametara. Ukoliko se zanemari nagli pad krivulje kod vremena izračunavanja, opravdano je uzeti vrijednost 300 za $n_estimators$ hiperparametar.

Tablica 4.14. Usporedba točnosti klasifikacije s predefiniranim i optimiziranim vrijednostima hiperparametara klasifikatora Slučajne šume

ML alg	predefinirane vrijednosti hiperparametara	optimizirane vrijednosti hiperparametara
<i>Acc</i>	0,999897447	0,999893911
<i>Precision</i>	0,999798412	0,999683852
<i>Recall</i>	0,999857694	0,999960471
<i>TN</i>	595317	595288
<i>FP</i>	51	80
<i>FN</i>	36	10
<i>TP</i>	252940	252966
<i>F2</i>	0,999845837	0,999905135
<i>BAcc</i>	0,999886016	0,99991305
<i>AUC</i>	0,999886016	0,99991305
<i>Cohen</i>	0,999754987	0,999746554
<i>Matthews</i>	0,999754988	0,999746574

Ako se provede klasifikacija koristeći klasifikator slučajne šume s optimiziranim hiperparametrima dobivenih postupkom unakrsne validacije i usporedi s predefiniranim hiperparametrima, može se zaključiti da se postiže veća točnost klasifikacije s korištenjem hiperparametara određenih unakrsnom validacijom kako pokazuje tablica 4.14.

Bitan parametar za koji se otkriva napredak je broj lažno negativnih rezultata (FN = 10) koji se smanjio za 72,2 %, a koji predstavlja anomalije odnosno neželjeni tok podataka koji se ne otkriva. Prisutno je i povećanje lažno pozitivnih rezultata (56,9 %), odnosno tokova prometa koji će nepotrebno alarmirati sustav jer je zapis toka prometa normalan. Mjera opoziva (engl. *recall*) definirana i objašnjena u poglavlju 3.5.5 potvrđuje taj ishod smanjenja broja lažno negativnih klasifikacija. Također i ostale mjere, prvenstveno F2 i AUC pokazuju bolje rezultate korištenjem optimiziranih hiperparametara. Cohen kappa-koeficijent ima zanemarivo manju vrijednost koristeći optimizirane hiperparametre isto kao i Matthewsov koeficijent, ali su vrlo visoko i blizu idealne vrijednosti 1 koja opisuje kvalitetu klasifikacije. Uravnotežena točnost i AUC-mjera imaju identične vrijednosti jer se radi o binarnoj klasifikaciji te su načini njihovih izračuna u tom slučaju identični kao što je objašnjeno u poglavlju 3.5.3.

4.1.5. Prilagodba odabranog referentnog skupa podataka NetFlow-strukturi

Metoda prikupljanja NetFlow-mrežnog toka razlikuje se od izravnog snimanja paketa, poput tcpdump-a, po tome što stvara sažetak komunikacije među izvorom i odredištem u mreži. Sažetak mrežnog toka koji se stvara NetFlowom obuhvaća sav promet koji se odnosi na sedam specifičnih ključnih elemenata koji su važni za adresiranje. Ti elementi su izvorna i odredišna IP-adresa, izvorni i odredišni portovi, tip protokola, vrsta usluge i sučelje na mrežnom uređaju. Ti atributi (ponekad se nazivaju 7-morka), zajedno s vremenom početka svakog mrežnog toka, razlikuju mrežne tokove jedne od drugih. Mrežni tok često pokriva više paketa, koji su grupirani zajedno pod zajedničkom oznakom toka. NetFlow-zapis sadrži oznaku i statistiku o paketima koji se nalaze unutar mrežnog toka. Ta statistika uključuje broj paketa, broj bajtova, vrijeme trajanja i vrijeme stvaranja paketa. Budući da je mrežni tok samo sažetak prometa, ne sadrži podatke o sadržaju paketa koji su preneseni unutar mreže [140]. Analizom podataka o protoku, koji se prikuplja, pruža se vidljivost toka i volumena prometa, kao i trag odakle promet dolazi, kamo ide i koliko se

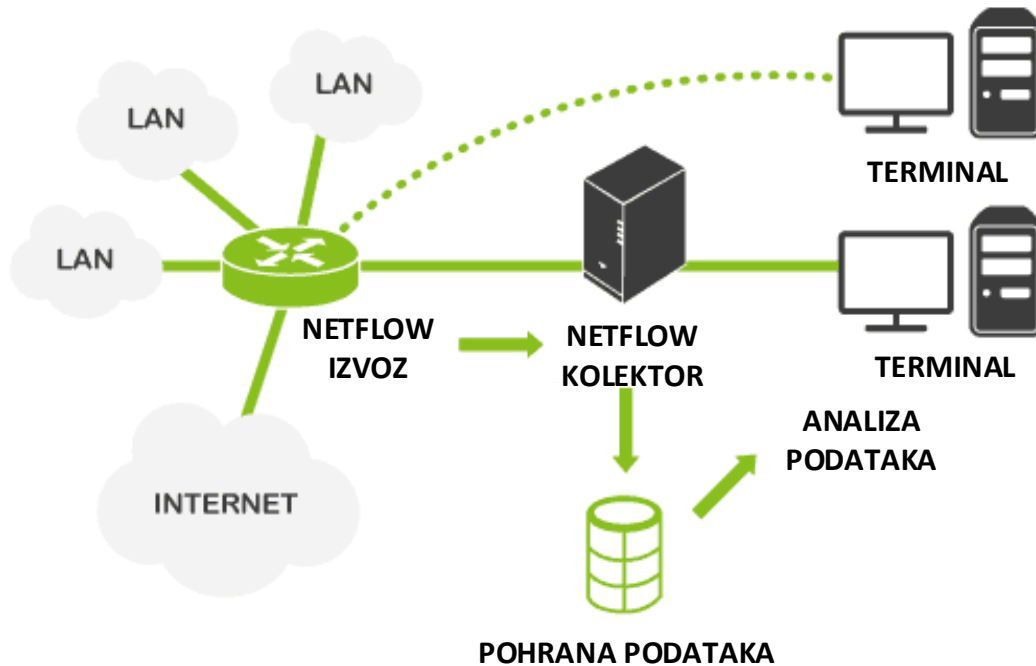
prometa generira u bilo kojem trenutku. Prikupljene informacije mogu se koristiti za praćenje korištenja mreže, otkrivanje anomalija i raznih drugih zadataka upravljanja mrežom. Prvi NetFlow format bio je podržan u svim početnim NetFlow-uređajima. Verzije 2, 3 i 4 bile su dostupne samo kao interna izdanja, dok je NetFlow verzije 5 najpopularnija verzija i još uvijek je podržavaju mnogi proizvođači mrežnih uređaja. NetFlow v5 ima fiksni format paketa, što olakšava praćenje i izvještavanje o mrežnom prometu, budući da je sadržaj svakog paketa brzo prepoznatljiv. Verzija 5 donijela je višestruka poboljšanja kao što su informacije o BGP-u (engl. *Border Gateway Protocol*), AS-u (engl. *Autonomous System*) i brojevima sekvenci toka. Iako su verzije 7 i 8 imale nekoliko dodatnih poboljšanja, više se ne koriste. Aktualna je verzija NetFlow verzije 9. Glavna razlika između NetFlow verzije 9 i prethodnih verzija je što verzija 9 koristi obrasce, što znači da omogućava fleksibilnije prikupljanje podataka. Drugim riječima, omogućava izvoz prilagođenih podataka, izvan osnovnih polja koja su bila uključena u prethodne verzije. NetFlow verzije 9 također podržava IPv6, MPLS i druge napredne mrežne tehnologije, što je čini prikladnijom za modernu mrežu. Dodatno, može izvoziti podatke o tokovima u različitim formatima, uključujući binarni i XML-format. NetFlow verzija 9 pruža snažan alat za mrežne administratore i sigurnosne stručnjake za dobivanje uvida u obrasce mrežnog prometa i identifikaciju potencijalnih sigurnosnih prijetnji ili problema s performansama. Svaki paket koji se prosljeđuje unutar preklopnika ili usmjerivača uključuje sljedeće informacije [141]:

- izvorište IP paketa
- odredište IP paketa
- izvorni port
- odredišni port
- klasa usluge
- tip protokola
- sučelje.

Implementacija prikupljanja NetFlow-zapisa mrežnog prometa kao što je prikazano slikom 4.20 zahtijeva tri glavne komponente [142]:

- izvoz toka prometa: mrežni uređaj poput preklopnika ili usmjerivača
- sakupljanje toka prometa: poslužitelj koji je odgovoran za primanje, pohranjivanje i pripremu podataka za analizu

- aplikacija za analizu: analizira podatke o protoku i omogućuje prikaz toka.



Slika 4.20. Osnovne komponente za prikupljanje i analizu NetFlow-zapisa mrežnog prometa [143]

Mrežna propusnost potrebna za izvoz NetFlow-podataka obično je manja od 0,5 % ukupne potrošnje propusnosti sučelja [144].

Odabrani referentni skup podataka LUFlow2021 sadrži telemetrijske podatke izrađene Cisco Joy alatom koji bilježi višestruka mjerenja povezana s tokovima mrežnog prometa. Iz tih mjerenja konstruirane su značajke skupa podataka, a opisane su u tablici 4.15. prema dokumentaciji [100].

Kako se LUFlow2021-skup podataka sastoji od više značajki od kojih su neke dobivene računski, potrebno je ujednačiti broj i tip značajki sa stvarnim izvozom NetFlow podataka iz mrežnog uređaja. Tijekom predobrade podataka, obrisane su 4 značajke: *dest_ip*, *src_ip*, *time_end*, *time_start* koje su irelevantne za otkrivanje anomalija, za ostatak značajki napravljena je procjena važnosti pomoću algoritama strojnog učenja te su rezultati opisani u sljedećim potpoglavljima.

Tablica 4.15. Opis značajki LUFlow2021-skupa podataka

značajka	opis
<i>src_ip</i>	Izvorišna IP adresa povezana s tokom. Ta je značajka anonimizirana za odgovarajući autonomni sustav.
<i>src_port</i>	Broj izvorišnog porta povezan s tokom.
<i>dest_ip</i>	Određišna IP-adresa povezana s tokom. Značajka je također anonimizirana na isti način kao <i>src_ip</i> .
<i>dest_port</i>	Broj odredišnog porta povezan s tokom.
<i>protocol</i>	Broj protokola povezan s tokom.
<i>bytes_in</i>	Broj bajtova prenesenih od izvora do odredišta.
<i>bytes_out</i>	Broj bajtova prenesenih od odredišta do izvora.
<i>num_pkts_in</i>	Broj poslanih paketa od izvora do odredišta.
<i>num_pkts_out</i>	Broj poslanih paketa od odredišta do izvora.
<i>entropy</i>	Entropija u bitovima po bajtu podatkovnih polja unutar toka. Taj je broj u rasponu od 0 do 8.
<i>total_entropy</i>	Ukupna entropija u bajtovima nad svim bajtovima u podatkovnim poljima toka.
<i>mean ipt</i>	Srednja vrijednost vremena dolaska paketa u toku.
<i>time_start</i>	Vrijeme početka toka.
<i>time_end</i>	Vrijeme završetka toka.
<i>duration</i>	Vrijeme trajanja toka, s preciznošću od mikrosekunde.
<i>label</i>	Oznaka toka.

Struktura NetFlow-skupa podataka iz mrežnog uređaja dobivena je prema preporukama proizvođača mrežne opreme [145]. Mrežni uređaj za izvoz NetFlowa konfiguriran je na način da se kreiraju obrasci za ulazni i izlazni tok „*flow record RECORD_IN*“ i „*flow record RECORD_OUT*“, izvoz tako prikupljenih podataka definira se preko „*flow exporter EXPORTER*“ unosa. Cijeli proces prikupljanja NetFlow-zapisa tada se definira na željenom fizičkom ili logičkom sučelju kao „*flow monitor MONITOR_IN*“ za ulazne podatke i „*flow monitor MONITOR_OUT*“ za izlazne podatke. Primjer cjelokupne konfiguracije potrebne za izvoz NetFlow-podatka iz mrežnog uređaja nalazi se u prilogu A i prilogu B ovog istraživanja.

Usporedbom referentnog skupa podataka LUFlow2021 i NetFlow skupa podataka dobivenog iz mrežnog uređaja uočene su razlike u značajkama koje se trebaju ujednačiti kako bi proces klasifikacije stvarnog mrežnog prometa bio izvediv i usporediv s rezultatima dobivenih korištenjem referentnog skupa podataka. Te razlike u značajkama prikazane su u tablici 4.16.

Tablica 4.16. Razlike u značajkama LUFlow2021-referentnog skupa podataka i stvarnog NetFlow-skupa podataka

LUFlow2021	NetFlow
<i>src_ip</i>	<i>sa</i>
<i>src_port</i>	<i>sp</i>
<i>dest_ip</i>	<i>da</i>
<i>dest_port</i>	<i>dp</i>
<i>protocol</i>	<i>pr</i>
<i>bytes_in</i>	<i>ibyt</i>
<i>bytes_out</i>	<i>obyt</i>
<i>Num pkts in</i>	<i>ipkt</i>
<i>Num pkts out</i>	<i>opkt</i>
<i>entropy</i>	
<i>total_entropy</i>	
<i>mean ipt</i>	
<i>time_start</i>	<i>ts</i>
<i>time_end</i>	<i>te</i>
<i>duration</i>	<i>td</i>
<i>label</i>	

Prema usporedbi značajki iz tablice 4.16., primijećeno je da NetFlow-skup podataka dobiven iz mrežnog uređaja ne sadrži značajke kao što su *entropy*, *total_entropy* i *mean ipt*, stoga je u procesu klasifikacije potrebno izostaviti te značajke iz skupa ulaznih podataka ili skup podataka iz mrežnog uređaja treba proširiti tim značajkama. Utjecaj i izostanak značajki u procesu klasifikacije istražen je i opisan u narednom poglavlju.

4.1.5.1. Smanjivanje broja značajki i prilagodba NetFlow-strukturi

Izbor značajki je bitan aspekt u modelima strojnog učenja jer se želi izbjeći uključivanje nepotrebnih značajki u izračune, a koje ne doprinose boljim rezultatima predviđanja predloženog modela. Pristup smanjivanju broja značajki koristi se za poboljšanje kvalitete skupa značajki u raznim zadacima strojnog učenja kao što su klasifikacija, grupiranje, regresija i drugi oblici predviđanja. Dodatne značajke mogu poboljšati prediktivne sposobnosti modela, ali samo do određene granice. Pojam "prokletstva dimenzionalnosti" često se koristi kako bi se naglasilo da će performanse modela opadati s povećanjem broja značajki, stoga je bitno pažljivo odabrati relevantne značajke kako bi se postigla optimalna prediktivna točnost modela. Zato treba odabrati

samo one značajke koje su učinkovite i omogućuju kvalitetno predviđanje. Odabir značajki sličan je tehnici redukcije dimenzionalnosti, gdje je također cilj smanjiti broj značajki, ali u osnovi se razlikuju u tome što se metodom odabira značajki određuje koje treba zadržati ili ukloniti iz skupa podataka, dok smanjenje dimenzionalnosti stvara novu projekciju podataka što rezultira potpuno novim značajkama. Odabirom značajki može se smanjiti složenost podataka i izračunavanja te se također mogu dobiti i učinkovitiji podskupovi značajki. Prikupljeni neobrađeni podatci obično su veliki pa je poželjno odabrati podskup podataka stvaranjem vektora značajki koji predstavljaju većinu informacija iz podataka [146], [147], [148]. Postoji mnogo metoda odabira značajki, ali korištene su i uspoređene metode odabira značajki prisutnih u scikit-learn-programskom paketu.

Jednovarijantni odabir značajki sa SelectKBest-metodom

SelectKBest-metoda radi na način da ocjenjuje svaku značajku pojedinačno koristeći neki statistički test, poput chi-kvadrata ili ANOVA-testa za kategorijske ili numeričke značajke. Nakon ocjenjivanja, značajke se rangiraju prema njihovoj važnosti ili informacijskoj vrijednosti. Najčešće se specificira željeni broj značajki (K) pa se odabir vrši temeljem tih najboljih K značajki. Odabir se vrši temeljem ocjene koja je izračunata za svaku značajku prema odabranom statističkom testu. Budući da je jednovarijantna metoda odabira značajki namijenjena algoritmima nadziranog strojnog učenja, dijelimo ih na nezavisne i zavisne [146]. Analizom različitih statističkih metoda za odabir značajki, može se zaključiti da smanjeni skup značajki može poboljšati kvalitetu klasifikacije isključivanjem manje relevantnih značajki. Ispitivanjem različitih metoda za odabir značajki, u nekim istraživanjima, pokazalo se da je statistička metoda "SelectKBest" s funkcijom "chi2" najučinkovitija. Upotrebom te metode moguće je dobiti smanjeni skup značajki s točnošću klasifikacije od 90 %, u usporedbi s punim skupom značajki koji daje točnost od 74 % [149].

Rekurzivno uklanjanje značajki

Rekurzivno uklanjanje značajki metoda je odabira koja koristi model strojnog učenja uz isključivanje najmanje važne značajke nakon rekurzivnog razmatranja sve manjeg skupa značajki.

Prvo se važnost svake značajke određuje na početnom skupu, a dobiva se kroz attribute *coef_* ili *feature_importances_* attribute. Zatim se iz trenutnog skupa značajki odvajaju najmanje važne značajke. Taj se postupak rekurzivno ponavlja na reduciranom skupu sve dok se na kraju ne postigne željeni broj značajki. Tom se metodom odabire k broj najboljih značajki. Prema zadanim postavkama u scikit-learn-platformi, broj je značajki odabranih tom metodom medijan ukupnih značajki, a korak (broj značajki koje se isključuju za svaku iteraciju) je jedan. Te postavke podložne su promjeni ovisno o korištenom skupu podataka [146]. U istraživanju [107] je korištenjem tog algoritma za smanjivanje broja značajki skup podataka smanjen za 95 % prema originalnoj veličini skupa podataka. Predloženi model postao je smisleniji te je s manjim skupom podataka postignuta zadovoljavajuća točnost.

Sekvencijski odabir značajki

Sekvencijski odabir značajki je postupak koji iterativno odabire najbolje značajke temeljem unakrsne validacije. Tijekom svake iteracije, odabire se jedna značajka koja najbolje poboljšava performanse modela kada se koristi samo ta jedna značajka za učenje modela. Nakon što je prva značajka odabrana, postupak se ponavlja dodavanjem nove značajke, a zaustavlja se kada se utvrdi da je postignut željeni broj značajki. Sekvencijski odabir značajki može se izvoditi i u obrnutom smjeru, počevši s kompletnim setom značajki i postupno ih isključujući dok se ne dostigne željeni broj značajki. Tijekom svake iteracije, uklanjaju se značajke koje najmanje utječu na performanse modela, sve dok se ne postigne željena razina performansi s manjim brojem značajki. Ta se metoda razlikuje od metode rekurzivnog uklanjanja značajki i jednovarijantnog odabira značajki jer ne zahtijeva *coef_* ili *feature_importances_* attribute. Znatno je sporija jer ocjenjuje rezultat višestrukom provjerom modela [146]. U radu [150], uspoređene su performanse modela na skupu podataka sa sekvencijskim odabirom značajni i bez njega. Rezultati pokazuju da sekvencijski odabir smanjuje broj značajki za 80 % te značajno poboljšava točnost klasifikacije, dok se istovremeno smanjuje vrijeme izračuna značajki iz ekstrahiranog toka podataka. Također, vrijeme potrebno za učenje modela strojnog učenja smanjuje se, što je vrlo važno za proces učenja posebno kada se radi s velikim skupovima podataka. Taj primjer ukazuje na važnost odabira značajki u postizanju optimalnih performansi modela strojnog učenja.

4.1.5.2. Usporedba i primjena algoritama smanjivanja broja značajki

Rezultati usporedbe algoritama za odabir značajki prikazani su u tablici 4.17. za ukupno 11 značajki: *avg_ip*, *bytes_in*, *bytes_out*, *dest_port*, *entropy*, *num_pkts_out*, *num_pkts_in*, *proto*, *src_port*, *total_entropy* i *duration*, koje su korištene u referentnom skupu podataka LUFlow2021. Cilj redukcije značajki je procijeniti važnost i utjecaj osam značajki koje su dio NetFlow-skupa podataka dobivenih iz mrežnih uređaja. Te su značajke: *src_port* (*sp*), *dest_port* (*dp*), *protocol* (*pr*), *bytes_in* (*ibyt*), *bytes_out* (*obyt*), *num_pkts in* (*ipkt*), *num_pkts out* (*opkt*) i *duration* (*td*).

Tablica 4.17. prikazuje rezultate odabira značajki korištenjem različitih metoda: *SelectKBest*, *rekurzivnog* i *sekvencijskog* odabira značajki. U posljednjem je stupcu (Σ) prikazano koliko je puta svaka značajka odabrana kao jedna od osam najrelevantnijih značajki. Temeljem broja pojavljivanja (stupac Σ) za svaku metodu odabira značajki, identificirano je 8 najrelevantnijih značajki. U tom odabiru, uz značajke "*entropy*" i "*avg_ip*", također je isključena značajka "*num_pkts_out*" koja ima manji značaj prema tablici 4.17.

Tablica 4.17. Usporedba metoda odabira značajki

značajka	SelectKBest -funkcija odabira			RecursiveFE-funkcija odabira			Sequential FS-funkcija odabira	Σ
	chi2	f_classif	mi_classif	GB	DT	RF	K-NN	
avg_ip	x					x	x	3
bytes_in	x	x	x			x	x	5
bytes_out	x	x	x	x		x	x	6
dest_port	x	x	x	x	x	x	x	7
entropy			x	x	x			3
num_pkts_out	x	x			x			3
num_pkts_in	x	x		x	x	x	x	6
proto		x	x	x	x			4
src_port	x	x	x	x	x	x	x	7
total_entropy	x	x	x	x	x	x	x	7
duration			x	x	x	x	x	5

Izvršeno je uspoređivanje izvedbe klasifikacije koristeći tri različita skupa značajki u referentnom skupu podataka LUFlow2021: s cijelim skupom značajki (11), s odabranim značajkama (8) dobivenim pomoću algoritama za odabir značajki te s korištenjem značajki koje

su karakteristične NetFlow protokolu (7) dobivenih iz mrežnih uređaja gdje ne postoje značajke “*avg_ip2*”, “*entropy*”, “*total entropy*”.

Rezultati i usporedba provedene klasifikacije sa skupovima podataka i smanjenim brojem značajki prikazana je zabicom 4.18.

Tablica 4.18. Klasifikacija s cijelim i smanjenim brojem značajki

	metoda odabira značajki		
	cijeli set značajki (11)	odabir uz algoritme strojnog učenja (8)	karakteristične NetFlow-značajke (7)
klasična točnost	0,999892	0,999889	0,999895
preciznost	0,999676	0,999664	0,999692
opoziv	0,99996	0,999964	0,999957
TN	595286	595283	595290
FP	82	85	78
FN	10	9	11
TP	252966	252967	252965
F2	0,999904	0,999904	0,999904
uravnotežena točnost	0,999911	0,999911	0,999913
AUC	0,999911	0,999911	0,999913
cohen kappa koeficijent	0,999741	0,999735	0,999749
Matthews koeficijent	0,999741	0,999735	0,999749
trajanje (sat:min:sek)	00:01:04.76	00:00:51.32	00:00:51.29

Usporedbom dobivenih rezultata u tablici 4.18. s nedavnim istraživanjima [66], [129], [150], [151], [152] gdje se analiziraju stope lažno pozitivnih i lažno negativnih rezultata, može se zaključiti da smanjenje broja značajki ne smanjuje učinkovitost klasifikacije. Istovremeno, ukupno vrijeme klasifikacije se smanjuje, što dodatno poboljšava performanse modela. Pri odabiru značajki specifičnih za NetFlow-protokol, dobiveni su rezultati gotovo identični kao i kada se koriste značajke reducirane algoritmima strojnog učenja. Međutim, prednost je u tome što se izbjegava značajka „*total_entropy*“, koja zahtijeva dodatno računanje i informacije o sadržaju IP-paketa koje nisu dostupne u NetFlow-podacima o toku mrežnog prometa. S obzirom na te rezultate i analizu odabira značajki, opravdano je isključiti značajku *total_entropy* iz daljnjeg korištenja unutar ulaznog skupa podataka za klasifikaciju i otkrivanje anomalija strojnim učenjem.

4.2. Prijedlog modela za otkrivanje anomalija mrežnog prometa primjenom strojnog učenja u hibridnoj programski definiranoj mreži

Predloženi model temelji se na klasifikaciji NetFlow-mrežnog prometa, koja se postiže strojnim učenjem i referentnim skupovima podataka kako bi se postigla željena klasifikacija, odnosno predikcija za svaki novi NetFlow-zapis koji se stvara na mrežnim uređajima. Kako bi se osigurala funkcija klasifikacije stvarnog mrežnog prometa u stvarnom vremenu, bitno je smanjiti vrijeme klasifikacije i odziva u slučaju pojave anomalija u mrežnom prometu. Korištenje NetFlow-alata za analizu mrežnog prometa jedna je od metoda za otkrivanje takvih anomalija. Korištenjem NetFlow nfcpad-kolektora (Nfdump NSEL-NEL1.6.18), datoteke se NetFlow-zapisa kreiraju i rotiraju svakih 60 sekundi, što omogućuje otkrivanje anomalija u mrežnom prometu u 60 sekundi. Međutim, novije inačice nfcpad-kolektora omogućuju rotaciju datoteka u dvije sekunde, što značajno smanjuje vrijeme odziva na otkrivanje anomalija u Netflow-prometu. Stoga, nadogradnja predloženog modela i korištenje novije inačice nfcpad-kolektora može osigurati još brže i učinkovitije otkrivanje i reakciju na pojave anomalija mrežnog prometa u stvarnom vremenu. U literaturi [153], [154], [155], [156] se opisuje računalstvo u stvarnom vremenu kao sustavi koji su sposobni odgovoriti u određenom vremenu i u stanju su reagirati na vremenski ograničene zadatke, posebno na one koji se ne javljaju u redovitim intervalima i imaju definirane rokove za odgovor. Oni se razlikuju od sustava koji se bave zadacima koji se izvršavaju skupno ili u vremenski neodređenim intervalima jer sustavi u stvarnom vremenu pružaju usluge i kontrolu za neovisne procese, s mogućnošću prekida i hijerarhijskim sustavom prioriteta. Prema predloženom, model koji je opisan u ovom poglavlju može se smatrati sustavom blizu stvarnom vremenu jer postiže vremenski interval koji je mnogostruko brži od manualne intervencije pri otkrivanju anomalija. Stoga, model pruža usluge i kontrolu neovisnim procesima, ima mogućnost prekida i shemu upravljanja prioritetima, što su ključne značajke sustava u stvarnom vremenu.

Model za otkrivanje anomalija i klasifikaciju prometa trebao bi biti sposoban nadzirati nekoliko različitih funkcija, kao što su one koje su odgovorne za praćenje prometa, klasifikaciju i ublažavanje posljedica kibernetičkog napada uzevši u obzir sljedeće aspekte [151]:

- Sveobuhvatan uvid u mrežu i sposobnost prikupljanja detaljnih i neograničenih informacija o mreži i tokovima prometa, a koristi NetFlow-tehnologiju za dobivanje informacija o tokovima iz različitih izvora.

- Po potrebi omogućiti ljudsku intervenciju za dodatan nadzor i upravljanje sustavom, gdje mrežni administrator prati rad modela, klasifikaciju prometa te ima mogućnost analize zapisa, uz prilagodbu sustava koja je bitna za rekonfiguraciju otkrivanja prema dinamičnim promjenama u mrežnom okruženju.
- Omogućiti automatsku konfiguraciju mrežnih resursa, gdje rekonfiguracija mrežnih uređaja pruža sredstva za rješavanje nepravilnosti, uz mogućnost prilagodbe prema specifičnim zahtjevima situacije.
- Modularnost i podrška prilagodbama koja omogućuje ažuriranje s novim algoritmima strojnog učenja ili strategijama obrane od kibernetičkih napada.

4.2.1. Klasifikacija kombiniranog skupa podataka

S pogleda mrežnog IDS-a, kibernetički se napad očituje kao kompleksan niz vremenski poredanih mrežnih paketa s određenim sadržajem. Najčešće se proces testiranja svodi na generiranje mrežnog prometa koji simulira aktivnosti kibernetičkih napada s pozadinskim mrežnim prometom. Reprodukcijska prethodno zabilježenog mrežnog prometa računalnih napada na posebnom testnom okruženju na kojem je instaliran IDS omogućuje višestruko ponavljanje eksperimentalnih uvjeta tako da će promet računalnih napada biti identično reproduciran svaki put. To je posebno važno kada je potrebno provesti usporedno testiranje sistema za otkrivanje upada. S druge strane, zahtjev za objektivnim rezultatima testiranja pretpostavlja da podatci mrežnog prometa moraju sadržavati statističku varijabilnost u određenim uvjetima funkcioniranja testiranja. Tu varijabilnost moguće je postići primjenom metoda sinteze mrežnog prometa ili metodama unosa anomalija u pripremljeni skup podataka mrežnog prometa [157].

Većina pristupa klasifikaciji mrežnog prometa može se prema [158] podijeliti u jednu od kategorija prema načinu korištenja pozadinskog prometa:

- klasifikacija s primjenom stvarnog pozadinskog prometa
- klasifikacija s primjenom stvarnog i pročišćenog pozadinskog prometa
- klasifikacija s primjenom simuliranog prometa.

Predloženi model u etapi učenja za potrebe klasifikacije koristi stvarni pozadinski promet uz primjenu simuliranog i generiranog mrežnog prometa. Prikupljanje stvarnog mrežnog prometa odvija se uz pomoć NetFlow-kolektora odnosno kolekcije NFDUMP-aplikacija koje podržavaju rad s NetFlow-zapisima podataka verzije v5, v7 i v9. NFDUMP-kolekcija aplikacija sadrži [159]:

nfcapd - prikuplja NetFlow-zapise podataka i pohranjuje podatke u binarne datoteke. Datoteke se stvaraju i rotiraju u određenom razdoblju (u korištenoj inačici od minimalno 60 sekundi). Aplikacija omogućuje jedinstveni nfcapd-proces za više izvora NetFlow-zapisa.

nfdump - omogućuje čitanje NetFlow-zapisa tokova podataka iz datoteka koje je pohranila komponenta nfcapd. Osim podataka o NetFlow-zapisu, prikazuje i razne statističke podatke o tokovima mrežnog prometa.

nfprofile - filtrira podatke prema proizvoljnim skupovima filtera (profilima) i pohranjuje ih u datoteke za kasniju upotrebu.

nfreplay - omogućuje ponavljanje NetFlow-podataka iz datoteka koje su pohranjene uz pomoć nfcapd-komponente.

nfclean.pl - omogućuje čišćenje starih NetFlow-podataka.

U promatranom istraživanju, najznačajnija aplikacija za predloženi NFMIDS-model je ***nfcapd***, s kojom se prikupljaju i spremaju NetFlow-zapisi mrežnog prometa s mrežnih uređaja u datoteke. Podatci se zatim čitaju komponentom ***nfdump*** i pretvaraju u CSV-format datoteke (engl. *comma separated values*) kako bi bili prilagođeni za upotrebu u procesu strojnog učenja. Za pokretanje ***nfcapd***-servisa koristila se sintaksa:

```
nfcapd -p 2055 -l <file path> -I any -t 28800
```

gdje argument *-p 2055* označava port na kojem se osluškiju i prikupljaju podatci iz mrežnih uređaja, *-l <file path>* označava putanju datoteke za pohranu, *-I any* označava identitet sučelja odnosno mrežnog uređaja te *-t 28 800* označava vrijeme rotacije datoteke od 28 800 sekundi odnosno 8 sati.

Nakon prikupljanja NetFlow-mrežnog prometa i spremanja u binarne datoteke, bilo je potrebno konvertirati ih u format čitljiv i drugim aplikacijama i programskim jezicima. U ovom slučaju, za konverziju se koristio CSV-format datoteke, koji omogućava čitanje podataka u

tabličnom obliku. Konverziju je bilo moguće izvršiti u **nfdump**-aplikaciji, gdje se argumentom *-o csv* označio željeni oblik izlazne datoteke u CSV-formatu. Argumentom *-b* sažimaju se tokovi prometa kao bidirekcionalni tokovi.

```
nfdump -r <filename>-b -o csv > <output_filename>.csv
```

Tako pripremljeni ulazni podatci spremni su za daljnju obradu za proces klasifikacije uz pomoć strojnog učenja.

Da bi se provjerila učinkovitost klasifikacije u stvarnom okruženju, kombinirao se referentni skup podataka LUFlow2021, koji ima smanjen broj značajki opisan u poglavlju 4.1.5.1, s prikupljenim stvarnim mrežnim prometom iz mrežnih uređaja. Stvarni mrežni promet definiran je kao normalni mrežni promet, budući da se radi o sigurnoj mrežnoj okolini, dok su anomalije mrežnog prometa uvezene iz referentnog skupa podataka LUFlow2021 i NF-UQ-NID. Stvarni mrežni promet prikupljan je tijekom pet tjedana te je prikupljeno ukupno 11,8 GB sirovih podataka podijeljenih u 105 datoteka (rotacija datoteka bila je svakih 8 sati). Kako bi se ostvario jednak omjer neuravnoteženog ulaznog skupa podataka kao u početnim razmatranjima u poglavlju 4.1.1. gdje se udio normalnog prometa u referentnim skupovima podataka kreće 70 – 88 % i time određuje svojstvo vrlo neuravnoteženog skupa podataka, koristio se samo dio prikupljenog mrežnog prometa koji sačinjavaju NetFlow-zapisi u vremenu velike mrežne aktivnosti korisnika (prije podnevni sati). Ostali korišteni zapisi prikupljeni u vremenu manje mrežne aktivnosti korisnika (poslijepodnevni sati) određeni su i uzorkovani slučajnim odabirom. Svi zapisi anomalija iz referentnog skupa podataka LUFlow2021 i NF-UQ-NID (opisanih u poglavlju 4.1.1.1 i 4.1.1.2) pridodani su dijelu prikupljenog NetFlow-prometa s mrežnih uređaja koji je označen kao normalan promet. Dobiveni skup podataka sadrži 70,9 % normalnog prometa i 29,1 % anomalija što je u skladu s početnim razmatranjima neuravnoteženih skupova u poglavlju 4.1.1. Ukupan broj zapisa kombiniranog skupa podataka iznosi 13 885 498, od čega je 9 833 772 zapisa normalnog prometa, a 4 051 726 zapisa uvezene su anomalije referentnih skupova. Tablica 4.19. sadrži detalje o rezultatima klasifikacije, a postignuti rezultati relevantnih metrika poput AUC (područje ispod ROC-krivulje) i F2 (F-beta mjera) od 98,09 % i 99,13 % potvrđuju iznimnu uspješnost predloženog modela. Unatoč relativno visokim apsolutnim vrijednostima lažno pozitivnih i lažno negativnih klasifikacija, gledajući njihove relativne vrijednosti (engl. *FNRate* i *FPRate*), može se reći da je njihov broj u okvirima ostalih postignutih rezultata sličnih istraživanja opisanih u

poglavlju 4.1.5.2. *FNRate* se prema [160] izračunava kao omjer lažno negativnih rezultata u odnosu na ukupan broj pozitivnih uzoraka te pri klasifikaciji stvarnog Netflow-prometa s uvezenim anomalijama iz skupova podataka LUFlow2021 i NF-UQ-NIDS iznosi 0,017.

Tablica 4.19. Klasifikacija stvarnog Netflow-prometa s uvezenim anomalijama iz skupova podataka LUFlow2021 i NF-UQ-NIDS

	rezultati
klasična točnost	0,98115
preciznost	0,953911
opoziv	0,982891
TN	1 928 272
FP	38 483
FN	13 864
TP	796 481
F2	0,976955
uravnotežena točnost	0,981662
AUC	0,981662
Cohen-kappa koeficijent	0,954797
Matthews koeficijent	0,955012
trajanje (sat:min:sek)	00:04:53.11

Lažno pozitivne i lažno negativne rezultate moguće je smanjiti nakon analize i ponovnim učenjem modela. Predloženi model omogućuje detaljnu analizu lažno negativnih klasifikacija, odnosno potencijalnih anomalija koje model nije otkrio. Lažno pozitivni rezultati u klasifikaciji mrežnog prometa nisu toliko kritični za sigurnost IT-sustava jer predstavljaju normalne tokove prometa koje nisu otkrili kao takvi. Međutim, mogu opteretiti sustav nepotrebnim dodatnim analizama prometa. Smanjenje broja lažno pozitivnih i lažno negativnih rezultata postiže se ponovnim učenjem modela s dodatnim podacima. Za izračunavanje metrika uspješnosti klasifikacije opisanog procesa koristio se algoritam pisan u *Python*-skriptnom jeziku prema pseudokodu Algoritam 2.

Algoritam 2: Klasifikacija uvezenih anomalija unutar stvarnog mrežnog prometa

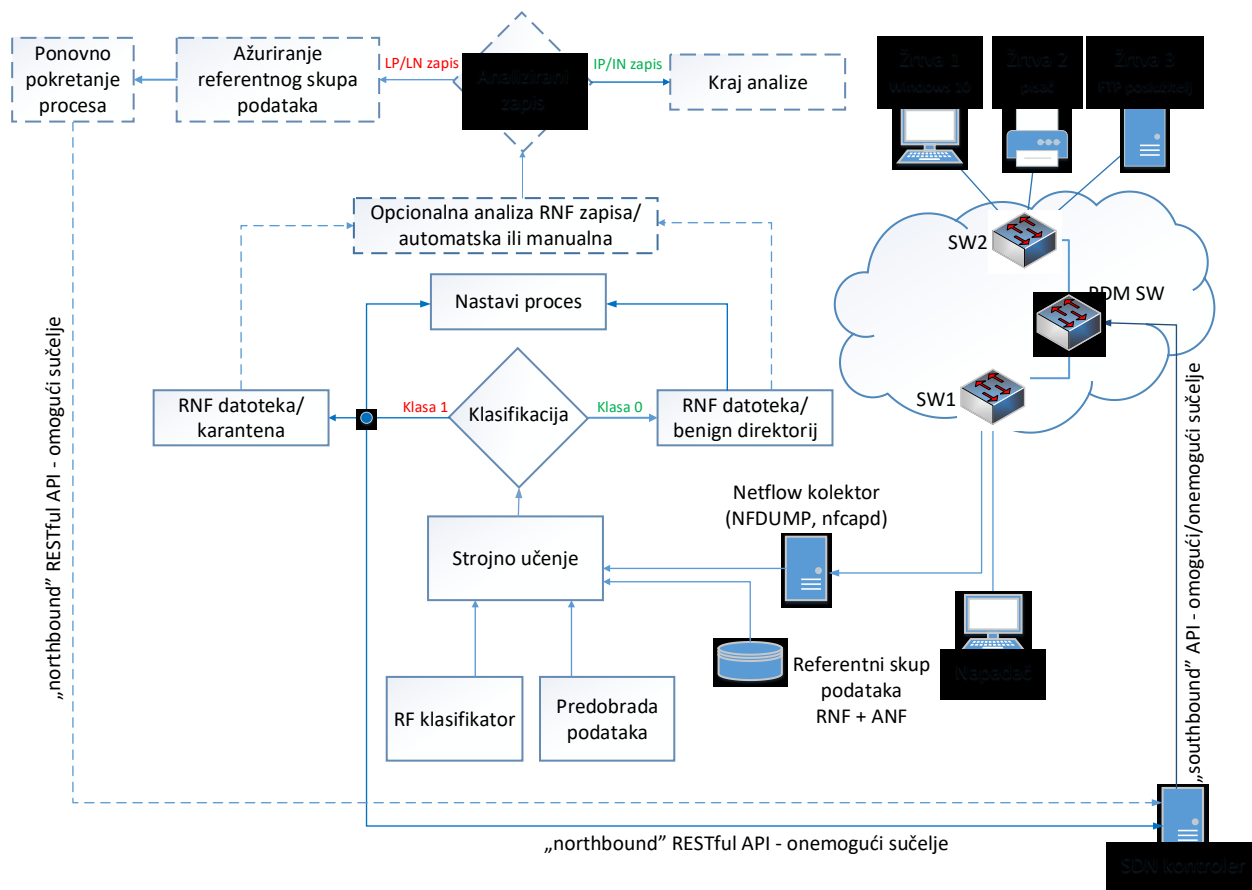
Input: skup podataka za učenje, skup podataka Netflow-mrežnog toka**Output:** AUC, F2

17. **Function:** učitavanje podataka
 18. **data1** ← prikupljeni stvarni Netflow-zapis prometa
 19. **data2** ← anomalije
 20. **Return** data1, data2
 21. **Function:** predobrada podataka (data1, data2)
 22. odabir značajki
 23. čišćenje nepotpunih zapisa
 24. konverzija kategorijskih značajki
 25. spajanje podataka data ← data1, data2
 26. **Function:** razdvajanje podataka (data, omjer razdvajanja)
 27. | X ← skup podataka svih značajki bez oznake klase
 28. | y ← značajka oznake klase
 29. **Return** X_train, X_test, y_train, y_test
 30. skaliranje podataka
 31. **Return** X_train_scale, X_test_scale, y_train, y_test
 32. **Function:** strojno učenje (X_train_scale, X_test_scale, y_train, y_test)
 33. y_pred ← Random Forest
 34. **Return** y_pred
 35. izračunavanje klasifikacijskih metrika AUC, F2 ← $f(y_pred, y_test)$
 36. kraj
-

Predloženi model za otkrivanje anomalija u stvarnom vremenu u hibridnoj programski definiranoj mreži prikazan je na slici 4.21., a sastoji se od hardverskih i softverskih komponenti opisanih prema tablici 4.20.

Tablica 4.20. Hardverske i softverske komponente NFMIDS-modela

hardver	model	softver
SDN-kontroler	Cisco Application Policy Infrastructure Controller	4.2(7v)
SDN-preklopnik	Cisco N9K-C93108TC-FX	Cisco NX-OS(tm) aci, Version 14.2(7v)
poslužitelj za strojno učenje i Netflow-kolektor	ESXi virtualni poslužitelj (64 GB RAM, 32 x CPU Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz, 400 GB HDD)	Ubuntu 20.04.5 LTS Nfdump NSEL-NEL1.6.18 Python 3.8.10 Sci-kit Learn 1.1.1
Host - napadač	HP Probook 6570b	Kali Linux 5.18.0
Host – žrtva1	HP 455 G2	Windows 10 Enterprise 20H2
Host – žrtva2	Lexmark CX510	
Host – žrtva3	HP EliteDesk 800 G4	Windows 10 Pro 22H2 FileZilla FTP Server 1.6.7



Slika 4.21. NFMIDS-model otkrivanja anomalija u stvarnom vremenu

Predloženi model naučen je temeljem klasifikacije kombiniranim mrežnim prometom s jedinom razlikom - nema potrebe za podjelom ulaznih podataka na skup za učenje i skup za testiranje. To je zato što je model već naučen s ulaznim podacima kombiniranog skupa podataka, a prikupljeni NetFlow-podatci postaju podatci za testiranje te se obrađuju i klasificiraju neposredno nakon prikupljanja, što predstavlja osnovnu svrhu rada modela u stvarnom vremenu. Referentni je skup podataka kombinacija stvarnog mrežnog prometa (RNF) i anomalija (ANF) iz skupova podataka LUFLOW2021 i NF-UQ-NID, kako je opisano ranije u ovom poglavlju.

U okviru istraživanja, predloženi NFMIDS-model klasificira NetFlow-podatke iz mrežnih uređaja. Model omogućuje automatsku klasifikaciju podataka u stvarnom vremenu, uzimajući u obzir ulazne podatke koji su prikupljeni nfcapd NetFlow-kolektorom, a dobiveni su unutar 60 sekundi iz stvarnog mrežnog uređaja. Model periodično provjerava postoji li novi NetFlow-skup zapisa, odnosno datoteka koju treba klasificirati i otkriti anomalije mrežnog prometa. Svaka se

datoteka automatski konvertira iz originalnog binarnog .nfcpad formata u .csv format, kako bi se mogla učitati u proces strojnog učenja. U procesu predobrade novih podataka, opisanim u poglavlju 4.1.2, stvaraju se preduvjeti za klasifikaciju novih NetFlow-zapisa. Ovisno o otkrivanju anomalija, datoteka se sprema u poseban direktorij karantene ili se sprema u direktorij benignog prometa. Spremanje datoteka omogućuje dodatnu detaljniju analizu NetFlow-zapisa. Ukoliko se u dodatnoj analizi ustanove lažno pozitivni ili lažno negativni rezultati klasifikacije, vraćaju se postavke zaustavljenog sučelja, ažuriraju se ulazni podatci te se takvi zapisi uvrštavaju u ponovni proces strojnog učenja kako bi se izbjegle daljnje pogrešne klasifikacije i unaprijedio model. Otkrivanje odnosno klasifikacija NetFlow-zapisa kao anomalije pokreće proces automatskog zaustavljanja prometa na sučelju na kojem je uočena anomalija. Ugrađeni REST API zahtjevi s informacijom o sučelju koje generira anomalije šalju se prema SDN-kontroleru koji automatski stavlja sučelje u „disabled“ način rada. Na taj se način automatski zaustavljaju neželjeni tokovi mrežnog prometa u stvarnom vremenu. U normalnom režimu rada predloženog NFMIDS-modela, u slučaju otkrivanja anomalija tijekom klasifikacije mrežnog prometa, aktivira se zahtjev za onemogućavanjem sučelja SDN-preklopnika API-zahtjevom prema SDN-kontroleru. API-zahtjevu za određenu radnju na sučelju SDN-preklopnika prethodi API-zahtjev autentifikacije korisnika čiji je odgovor (*cookie*) sastavni dio API-a za onemogućavanje sučelja. Primjer API-zahtjeva iz *Python*-skripte prema SDN-kontroleru sastoji se od nekoliko dijelova:

```
url = https://<ip_address>/api/node/mo/uni/fabric/outofsvc.json
```

- gdje se koristi *https* protokol za izvršavanje zahtjeva kako bi komunikacija prema SDN-kontroleru na adresi *ip_address* bila sigurna

```
headers = {'Content-Type': 'application/json', 'Cookie': cookie}
```

- definira se *json* tip podataka koji slijedi u zahtjevu i *cookie*-zapis dobiven pri API zahtjevu za autentifikacijom

```
payload={"fabricRsOosPath":{"attributes":{"tDn":"topology/pod-1/paths-1121/pathep-  
[target_interface]","lc":"blacklist"},"children": []}}
```

- označava se sučelje (*target_interface*) i radnja (*blacklist*) koja se izvršava nad sučeljem

```
response = requests.request("POST", url, headers=headers, json=payload, verify=False)
```

- izvršava se API kao POST metoda koja izvršava željenu akciju nad sučeljem.

Ranije korišten i opisan pseudokod „Algoritam 2“, za ocjenu uspješnosti klasifikacije, modificiran je u „Algoritam 3“ za klasifikaciju NetFlow-zapisa tokova mrežnog prometa u stvarnom vremenu.

Algoritam 3: Otkrivanje anomalija mrežnog prometa u stvarnom vremenu

Input: skup podataka za učenje (referentni NetFlow-skup podataka s anomalijama), Netflow-promet u stvarnom vremenu
Output: klasa 1, klasa 2

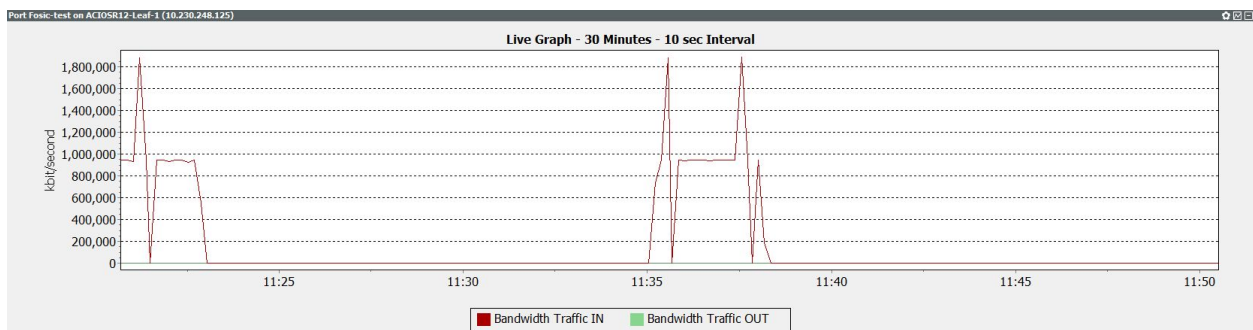
1. **Function:** učitavanje podataka za učenje
2. **data1** ← prikupljeni stvarni NetFlow-zapis prometa
3. **data2** ← anomalije referentnog skupa podataka
4. **Return** data1, data2
5. **Function:** predobrada podataka (data1, data2)
 6. odabir značajki
 7. čišćenje nepotpunih zapisa
 8. konverzija kategorijskih značajki
 9. spajanje podataka data ← data1, data2
 10. skaliranje podataka
11. **Return** X_train_scale, y_train
12. strojno učenje (klasifikator Slučajne šume, X_train_scale, y_train)
13. ponavlja
 14. učitaj podatke data3 ← stvarni Netflow-zapis podataka
 15. predobrada podatka (data3)
 16. strojno učenje (Random Forest, X_scale_data3)
 17. ako je y_pred = klasa 1
 18. sučelje = zapis iz data3
 19. Netflow-datoteka → direktorij za karantenu
 20. izvrši API = disable sučelje
 21. ako je y_pred = klasa 0
 22. Netflow-datoteka → direktorij za normalan promet
 23. sačekaj n-sekundi
24. ponovi
25. kraj

Algoritam 3 proširiv je s funkcijama ponovnog učenja i vraćanja sučelja na prvobitno stanje prosljeđivanja paketa ukoliko se pojavi potreba npr. lažno pozitivan rezultat klasifikacije. Ponovno učenje modela vrši se kombiniranim referentnim skupom podataka za učenje koji se nadopunjuje odabranim zapisima iz pohranjenih RNF-datoteka (stvarnih NetFlow-zapisa) u direktorijima „benign“ ili „karantena“.

4.2.2. Rezultati otkrivanja anomalija u stvarnom vremenu

Za kompleksni kibernetički napad treba uzeti u obzir da se napad sastoji od niza osnovnih napadačkih akcija, odnosno pojedinačnih akcija napadača koje se provode korištenjem različitih softverskih sustava s ciljem postizanja konačnog cilja složenog napada. Kriterij koji povezuje akcije napadača koristeći osnovne napadačke akcije određeni je tip ranjivosti, primjerice, ranjivost prelijevanja međuspremnik u mrežnoj usluzi ili primjena određenog softverskog alata, kao što je softver za pogađanje lozinki. Tijekom napada, napadač donosi odluke o odabiru određenih napadačkih akcija kako bi postigao svoj konačni cilj. Pri odabiru, uzima u obzir očekivane rezultate akcija i informacije o ciljanom sustavu. Svaka napadačka akcija zahtijeva ispunjenje određenih uvjeta, poput prisutnosti otvorenog TCP-porta ili ranjivosti softvera. Neki napadi zahtijevaju i specifična znanja o korisničkim podacima poput imena ili lozinki. Te akcije imaju posljedice koje mogu utjecati na parametre sustava i omogućiti napadaču pristup dodatnim informacijama. Napadi se odvijaju tijekom vremena i mogu se odvijati sinkrono ili asinkrono. Većina je vremenskih odgoda tijekom napada nasumične prirode, ovisno o različitim čimbenicima poput brzine prijenosa mrežnih paketa [157].

Prema shemi na slici 4.21. i Algoritmu 3 testiran je model na trima vrstama kibernetičkih napada u kontroliranom okruženju hibridne SDN-mreže. Inicijalno testiranje otkrivanja anomalija mrežnog prometa uključivalo je NFMIDS-model bez IPS-funkcionalnosti kako bi se učinkovitost modela mogla provjeriti nakon uključivanja IPS-funkcionalnosti. Na slici 4.22. prikazan je grafikon propusnosti sučelja na SDN-uređaju tijekom rada računala napadača u normalnom režimu te u vremenima kibernetičkih napada.



Slika 4.22. Grafikon propusnosti sučelja SDN-uređaja tijekom inicijalnog testiranja bez IPS-funkcionalnosti

Kao primjer kibernetičkih napada koristili su se sljedeći tipovi:

- Tip 1 - DoS – uskraćivanje usluge
- Tip 2 - otkrivanje korisničkih imena i lozinki – dobivanje neovlaštenog pristupa
- Tip 3 - skeniranje portova na udaljenom uređaju.

Razlozi odabira tih vrsta kibernetičkih napada su učestalost i lakoća korištenja. U današnjem okruženju opće povezanosti na internet, *Denial-of-Service* (DoS) ili *Distributed Denial-of-Service* (DDoS) napadi su se jako proširili i napadači čine *online*-sustave nedostupnim legitimnim korisnicima slanjem ogromnog broja paketa ciljanom sustavu. Otkrivanje korisničkih imena i lozinki rječnikom uobičajeni su napadi koji često uspiju jer mnogi korisnici koriste uobičajene varijante nekoliko lozinki te nije potrebno dodatno iskorištavanje sigurnosnih propusta jer su sve sigurnosne kontrole zaobiđene. Uspješan *brute-force*-napad ili napad rječnikom osobito je koristan za napadača kada kompromitirani račun pripada administratoru sustava, koji ima veće privilegije od običnog korisničkog računa. Skeniranje portova jedna je od najpopularnijih tehnika izviđanja koju mnogi napadači koriste za profiliranje pokrenutih usluga na potencijalnoj meti prije pokretanja napada.

Normalan promet uključivao je mrežne aktivnosti:

- *http* i *https* promet (rad u internet pregledniku)
- *ftp* promet (spajanje na udaljene poslužitelje, preuzimanje i pohranjivanje datoteka raznih veličina)
- *icmp* promet (standardni *ping* udaljenih mrežnih uređaja)
- modificirani *icmp* promet (*ping* udaljenih mrežnih uređaja s većim paketima)
- *traceroute* promet (ispitivanje čvorova uključenih u promet paketa od izvorišne do odredišne adrese)
- rad s nekoliko mrežnih aplikacija u korporativnom poslovnom procesu.

Spomenuta tri kibernetička napada samo su dio lepeze poznatih kibernetičkih napada, za koje se može reći da su česti načini pripreme za ozbiljnije kibernetičke napade opisanih u literaturi [161], [162], [163], [164].

Tijek inicijalnog testiranja NFMIDS-modela bez IPS-funkcionalnosti prikazan je tablicom 4.21., dok je propusnost SDN-sučelja napadača prikazana slikom 4.22.

Tablica 4.21. Tijek inicijalnog testiranja NFMIDS-modela bez IPS-funkcionalnosti

vrijeme početka	vrsta prometa	vrijeme otkrivanja	vrijeme završetka
11:35:00	kibernetički napad tip 1	11:35:43	11:38:00
11:38:00	normalan promet	11:39:50	11:41:00
11:41:00	kibernetički napad tip 2	11:42:30	11:44:00
11:44:00	normalan promet	11:45:41	11:47:10
11:47:10	kibernetički napad tip 3	11:47:32	11:47:58
11:47:58	normalan promet	11:49:45	11:50:00

Temeljem analize tablice 4.21., pokazuje se da je predloženi NFMIDS-model uspješno otkrio tri različita kibernetička napada. Prvi je napad pokrenut u 11:35:00, drugi u 11:41:00, a treći u 11:47:10. Svaki je od napada trajao nekoliko minuta prije nego što je namjerno prekinut, nakon čega se sustav vratio u normalno stanje i nastavio bilježiti uobičajeni promet. Vrijeme otkrivanja napada variralo je od nekoliko sekundi do nekoliko minuta, s najdužim kašnjenjem od oko 90 sekundi za drugi tip napada. Ta kašnjenja proizlaze iz vremena pokretanja napada te vremena u kojem je NFMIDS radio klasifikaciju prometa iz kreirane nfcpad-datoteke. Kašnjenje otkrivanja bilo kibernetičkog napada bilo normalnog prometa nakon prestanka kibernetičkog napada, uvelike ovisi o vremenu početka događaja i vremenu rotacije i kreiranja nfcpad-datoteka. Što je početak napada vremenski bliže završetku formiranja nfcpad-datoteke, to model može brže otkriti anomaliju jer će prije započeti obrada NetFlow-zapisa. Razlika u vremenima početka i otkrivanja kibernetičkog napada u tablici 4.21. objašnjava se načinom rada nfcpad-kolektora i prikupljanja NetFlow-zapisa koji omogućava klasifikaciju paketa skupljenih tijekom vremena od 60 sekundi. Također, razlika u vremenima stvarnog zaustavljanja kibernetičkog napada i ponovnog otkrivanja normalnog prometa, rezultat je zaostatka podataka u međuspremniku mrežnog uređaja koji šalje NetFlow-zapise, što dovodi do odgođenog slanja zapisa u kolektor.

Brže rotacije nfcpad-datoteka omogućuje novija inačica NFDUMP-kolekcije aplikacija koja prema dokumentaciji ima minimalno vrijeme rotacije, 2 sekunde, te bi se nadogradnjom predloženog modela problem kašnjenja mogao značajnije umanjiti. Ažuriranje NFDUMP-

kolekcije aplikacija tijekom izrade ove disertacije nije napravljeno iz razloga praktične primjene, stabilnosti i optimalno podešenih postavki parametara strojnog učenja trenutne konfiguracije modela. Budući da je *Python*-programski jezik korišten za strojno učenje na istoj platformi kao i NFDUMP-aplikacije, postoji vjerojatnost nekompatibilnosti različitih verzija korištenih aplikacija na zajedničkoj hardverskoj platformi. Da bi se izbjegla potencijalna nekompatibilnost, NFDUMP nije ažuriran, već je ostao na verziji koja omogućava minimalnu vrijednost od 60 sekundi za rotaciju datoteka.

Predloženi rezultati u tablici 4.21. ukazuju na to da je predloženi sustav učinkovit u otkrivanju kibernetičkih napada i zaštiti mrežne sigurnosti. Analizom podataka u tablici 4.22. s grafikonom na slici 4.22. o propusnosti sučelja na SDN-preklopniku, gdje je spojeno računalo napadača, proizlazi da se samo kibernetički napad tipa 1 (DoS) može povezati s povećanjem prometa na sučelju, dok druga dva napada nisu uzrokovala takvo povećanje. Tablica 4.22. također prikazuje detaljan ispis volumena mrežnog prometa na promatranom sučelju SDN-preklopnika tijekom inicijalnog testiranja gdje su crveno označeni vremenski isječci kibernetičkih napada iz aplikacije za nadzor propusnosti mrežnog sučelja.

Tablica 4.22. Propusnost promatranog SDN-sučelja tijekom inicijalnog testiranja

Vrijeme	Propusnost sučelja (kbit/s)	Tip mrežnog prometa
3/28/2023 11:49 - 11:50	257 300	normalan
3/28/2023 11:48 - 11:49	267 748	normalan
3/28/2023 11:47 - 11:48	341 132	Kibernetički napad tip 3
3/28/2023 11:46 - 11:47	228 131	normalan
3/28/2023 11:45 - 11:46	303 815	normalan
3/28/2023 11:44- 11:45	238 582	Kibernetički napad tip 2
3/28/2023 11:43 - 11:44	339 185	Kibernetički napad tip 2
3/28/2023 11:42- 11:43	296 233	Kibernetički napad tip 2
3/28/2023 11:41 - 11:42	303 227	Kibernetički napad tip 2
3/28/2023 11:40 - 11:41	256 510	normalan
3/28/2023 11:39 - 11:40	273 544	normalan
3/28/2023 11:38 - 11:39	120 459 411	normalan
3/28/2023 11:37 - 11:38	998 839 698	Kibernetički napad tip 1
3/28/2023 11:36 - 11:37	943 143 776	Kibernetički napad tip 1
3/28/2023 11:35 - 11:36	843 298 254	Kibernetički napad tip 1

Prema podacima u tablici 4.22., vidljivo je da je od 11:35 do 11:38 došlo do značajnog povećanja prometa, što se može tumačiti kao indikacija kibernetičkog napada. Zaostatak povećanog prometa od 11:38 do 11:39 kada je kibernetički napad završio, posljedica je petominutnog prosjeka vrijednosti u nadzornom alatu koji se koristio za bilježenje podataka o propusnosti sučelja. Međutim, predloženi model s uključenom IPS-funkcionalnošću može otkriti i zaustaviti kibernetičke napade temeljem prvog otkrivanja anomalije mrežnog prometa. Zaustavljanjem sučelja SDN-mrežnog uređaja na kojem je primijećena anomalija, isključuje se mogućnost pogrešnog otkrivanja povećanog prometa.

Tijekom inicijalnog testiranja i operativnog rada predloženog NFMIDS-modela u normalnom režimu bez mrežne aktivnosti kibernetičkih napada, primijećena je pojava lažno pozitivne klasifikacije svakih deset minuta, što je iznosilo jednu lažno pozitivnu klasifikaciju od ukupno 1 317 prispjelih NetFlow-zapisa mrežnog prometa. Analizom datoteka u kojima je otkrivena anomalija, potvrđena je lažno pozitivna klasifikacija. Uvrštavanjem tog zapisa u proces ponovnog učenja gdje je takav zapis označen kao normalan promet, nije se više ponovila lažno pozitivna klasifikacija u 60 minuta kada je sustav mirovao bez ikakve ili s malom mrežnom aktivnosti. Taj postupak potvrđuje mogućnost smanjenja lažno pozitivnih klasifikacija ponovnim učenjem modela uz dodatne označene ulazne podatke za strojno učenje. Svaka anomalija koja se otkrije u zaustavljanju NetFlow-zapisa (nfcpad-datoteka) koji se obrađuje, pohranjuje se u zaseban direktorij u svrhu pregledavanja zapisa s otkrivenom anomalijom. Takav pristup ima za cilj otkriti lažno pozitivne klasifikacije i omogućiti ponovno učenje modela s novim ulaznim podacima koji sadrže sve lažno pozitivne klasifikacije označene kao normalan promet. To je važno jer će poboljšati točnost budućih klasifikacija predloženog modela. Korištenjem takvog pristupa, moguće je unaprijediti kvalitetu klasifikacije i osigurati veću preciznost u otkrivanju anomalija mrežnog prometa.

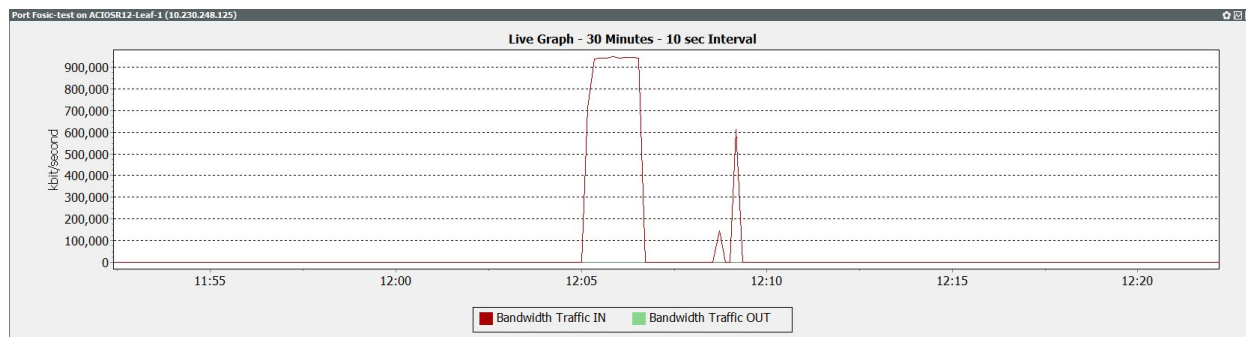
Uključivanjem IPS-funkcionalnosti u NFMIDS-model omogućuje se automatsko zaustavljanje kibernetičkih napada API-zahtjevom, kako je opisano u prethodnom potpoglavlju. Tijekom obrade podataka u klasifikaciji, za svaki obrađeni zapis privremeno se pohranjuje broj sučelja na kojima je otkrivena anomalija. Ta informacija koristi se pri slanju API-zahtjeva kontroleru kako bi se onemogućio mrežni promet na kompromitiranom sučelju SDN-uređaja.

Tablica 4.23. prikazuje rezultate testiranja NFMIDS-modela s uključenom IPS-funkcionalnosti u hibridnoj programski definiranoj mreži u stvarnom vremenu, dok grafikon na slici 4.23. prikazuje zabilježenu propusnost sučelja. U tablici se za svaki testni scenarij prikazuje vrijeme početka, vrsta prometa, vrijeme otkrivanja, vrijeme zaustavljanja sučelja i vrijeme vraćanja sučelja u normalan režim. Rezultati pokazuju sposobnost modela u otkrivanju i automatskom zaustavljanju kibernetičkih napada, kao i vraćanje sučelja u normalan režim rada nakon što je napad prekinut radi pokretanja i otkrivanja novog kibernetičkog napada.

Tablica 4.23. Tijek testiranja NFMIDS-modela s uključenom IPS-funkcijom

vrijeme početka	vrsta prometa	vrijeme otkrivanja	vrijeme zaustavljanja sučelja	vrijeme vraćanja sučelja u normalan režim rada
12:05:00	kibernetički napad tip 1	12:06:22	12:06:29	12:08:00
12:08:00	normalan promet	/	/	/
12:12:00	kibernetički napad tip 2	12:12:15	12:12:22	12:15:00
12:15:00	normalan promet	/	/	/
12:18:00	kibernetički napad tip 3	12:19:03	12:19:10	12:21:00
12:21:00	normalan promet	/	/	/

Tijekom normalnog prometa bez kibernetičkih napada nisu se mjerila vremena, budući da je sučelje prije toga bilo zaustavljeno i nije se pojavljivao nikakav promet na sučelju.



Slika 4.23. Grafikon propusnosti sučelja SDN-uređaja tijekom testiranja s uključenom IPS-funkcijom

Uspoređujući grafikone prikazane na slikama 4.22. i 4.23. te analizirajući tablice 4.22. i 4.23., vidljivo je da je NFMIDS-model uspješno otkrio kibernetičke napade te ih zaustavio unutar vremena rotacije NetFlow-datoteka koje prikuplja NetFlow-kolektor. U tablici 4.24. prikazani su rezultati mjerenja propusnosti SDN-sučelja tijekom testiranja predloženog NFMIDS-modela s aktiviranom IPS-funkcionalnošću.

Tablica 4.24. Propusnost SDN-sučelja tijekom testiranja s IPS-funkcijom

vrijeme	propusnost sučelja (kbit/s)	tip mrežnog prometa
12:22 - 12:23	123 115	normalan
12:21 - 12:22	43 741	normalan
12:20 - 12:21	0	zaustavljeno sučelje – nema prometa
12:19 - 12:20	75 461	kibernetički napad tip 3
12:18 - 12:19	116 510	kibernetički napad tip 3
12:17 - 12:18	124 818	normalan
12:16 - 12:17	266 108	normalan
12:15 - 12:16	45 017	normalan
12:14 - 12:15	0	zaustavljeno sučelje – nema prometa
12:13 - 12:14	0	zaustavljeno sučelje – nema prometa
12:12 - 12:13	123 350	kibernetički napad tip 2
12:11 - 12:12	63 549	normalan
12:10 - 12:11	114 461	normalan
12:09 - 12:10	105 744 867	normalan
12:08 - 12:09	24 753 438	normalan
12:07 - 12:08	0	zaustavljeno sučelje – nema prometa
12:06 - 12:07	596 296 814	kibernetički napad tip 1
12:05 - 12:06	818 809 846	kibernetički napad tip 1
12:04 - 12:05	297 778	normalan

Dobiveni rezultati u tablici 4.24. jasno pokazuju očekivani prekid u mjerenju propusnosti sučelja, koji je posljedica automatskog zaustavljanja sučelja API-zahtjevom prema kontroleru SDN-uređaja nakon što je otkriven kibernetički napad. Također se potvrđuje uspješno otkrivanje i zaustavljanje kibernetičkih napada, što naglašava učinkovitost IPS-funkcionalnosti u okviru

NFMIDS-modela za osiguranje mreže od kibernetičkih prijetnji. Mjerenje propusnosti sučelja u istoj tablici dodatno potvrđuje da se automatsko zaustavljanje sučelja tijekom napada odražava na smanjenje ukupne propusnosti mreže u tom trenutku.

Predloženi model može pružiti značajnu vrijednost u praćenju mrežnog prometa, održavanju sigurnosti i funkcionalnosti u različitim okruženjima, posebno onima koja su od kritične važnosti. Dobiveni rezultati daju uvid u učinkovitost NFMIDS-modela s uključenom IPS-funkcionalnošću i predlažu mogućnost primjene takvog sustava za zaštitu mreža od kibernetičkih napada. Rezultati klasifikacije stvarnog mrežnog prometa jasno predstavljaju važnost kontinuiranog testiranja i razvoja sigurnosnih sustava te potencijalne prednosti upotrebe strojnog učenja u otkrivanju anomalija u mrežnom prometu. Temeljem dobivenih rezultata možemo zaključiti da dodavanje označenih podataka u proces ponovnog učenja značajno unaprjeđuje preciznost otkrivanja, smanjujući broj lažno pozitivnih i lažno negativnih klasifikacija te poboljšavajući cjelokupnu pouzdanost predloženog modela.

5. PRIJEDLOG METODE VERIFIKACIJE MODELA ZA OTKRIVANJE ANOMALIJA U HIBRIDNOJ PROGRAMSKI DEFINIRANOJ MREŽI

Verifikacija i validacija osnovni su procesi za kvantificiranje i izgradnju povjerenja (ili vjerodostojnosti) za modele. Definicije pojmova verifikacije i validacije prema [165] su:

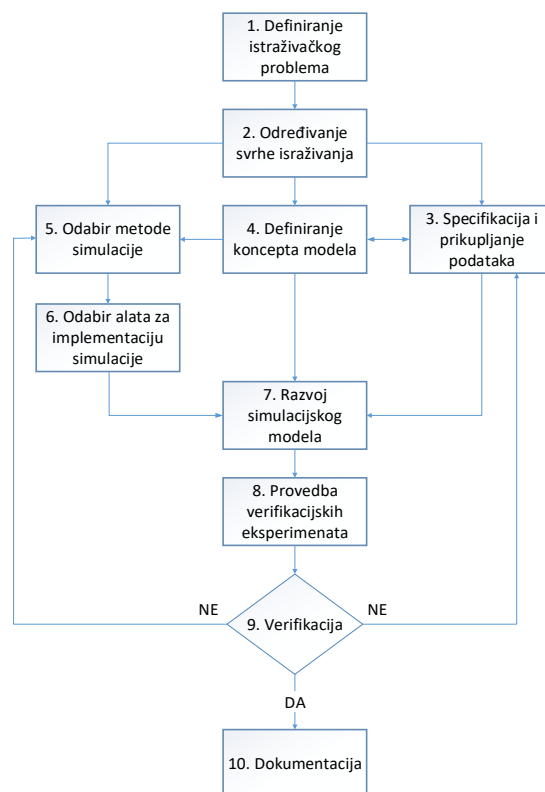
- verifikacija je proces utvrđivanja točnosti implementacije modela u odnosu na konceptualni opis i rješenje prema zamisli autora
- validacija je proces utvrđivanja u kojoj mjeri je model točan prikaz stvarnog okruženja s obzirom na namjene modela.

Procesi verifikacije i validacije ključni su za prikupljanje dokaza o ispravnosti i točnosti modela u određenim scenarijima. Važno je napomenuti da se tim procesima ne može apsolutno dokazati točnost i preciznost modela za sve moguće uvjete i primjene, ali mogu pružiti dokaze da je model upotrebljiv i dovoljno precizan. Proces verifikacije i validacije smatra se završenim kada se postigne zadovoljavajuća razina preciznosti. Verifikacija modela predstavlja glavni vid razvoja simulacijskih modela i provodi se paralelno s procesom dizajna modela. Glavni cilj je osigurati da simulacijski model bude koristan u rješavanju stvarnih problema u posebnim situacijama. Iako verifikacija ne može jamčiti potpunu točnost modela za sve moguće primjene, može pružiti dokaze o njegovoj točnosti u određenim stvarnim problemima. Istraživanja često obrađuju različite poglede i metode verifikacijskog i validacijskog procesa, no primjeri su njihove primjene u stvarnim okruženjima ograničeni [165], [166].

Slikom 5.1. prikazan je postupak verifikacije i potrebnih aktivnosti u rješavanju problema kako je definirano prema [166] :

1. Definiranje istraživačkog problema gdje se razrađuju i identificiraju stvarni problemi i usklađuju očekivanja od istraživačkih rezultata.
2. Određivanje svrhe istraživanja i simulacijskih eksperimenata.
3. Specifikacija podataka i informacija potrebnih za definiciju konceptualnih i simulacijskih modela.

4. Definiiranje konceptualnog modela u skladu s određenom istraživačkom svrhom koristeći prikupljene podatke i informacije kako bi se predstavile relevantne veze s istraživačkim problemom.
5. Odabir metode modeliranja i simulacije za reprezentaciju definiranog istraživačkog problema.
6. Odabir alata u kojem će se implementirati simulacijski model, taj proces odabira uključuje razmatranje dostupnosti i prilagodljivosti alata.
7. Razvija se simulacijski model iz konceptualnog modela odabranom simulacijskom metodom i alatom.
8. Provode se verifikacijski eksperimenti sa simulacijskim modelom i provjerava se pouzdanost i očekivani rezultati za dane ulazne podatke.
9. Pregled rezultata verifikacijskih eksperimenata. Ako je potrebno, koraci 3, 4, 5, 6, 7 i 8 mogu se ponoviti.
10. Dokumentiraju se upute i izrađuju dokumenti koji prate simulacijski model i eksperimente.



Slika 5.1. Procedura verifikacije modela [166]

5.1. Pregled postojećih metoda verifikacije

Najčešći pristup verifikaciji IDS-sustava uključuje upotrebu stvarnog prometa iz stvarnih mreža ili javno dostupnih skupova podataka mrežnog prometa. Izuzetak su privatne korporativne mreže, kojima je pristup ograničen na vrlo malen broj istraživača, zbog čega se javno dostupni skupovi podataka često koriste kao prva opcija za verifikaciju nekog rješenja, međutim ne postoji zajednički test koji bi pokazao da generirani skup podataka sadrži instance stvarnog mrežnog prometa. Također, ne postoji univerzalni verifikacijski test koji bi potvrdio da otkriveni napadi odražavaju stvarne mrežne napade. Važno je temeljito evaluirati IDS u smislu pronalaženja odgovarajućih podataka za učenje i klasifikaciju, kao i za interpretaciju rezultata radi pouzdanog otkrivanja napada. Unatoč brojnim istraživanjima u području IDS-a, uspjeh metoda verifikacije izuzetno ovisi o realnom testnom okruženju [167].

Tablica 5.1. Pregled literature s naglaskom na metode verifikacije IDS-a

#	godina	metode verifikacije	testno okruženje verifikacije	položaj IDS/IPS	SDN
[168]	2017.	usporedba s drugim sustavima i testiranje s uzorcima podataka	stvarno	pasivna	ne
[169]	2017.	testiranje s uzorcima podataka	simulirano	/	ne
[170]	2015.	testiranje s uzorcima podataka	simulirano	aktivna	ne
[171]	2018.	testiranje s uzorcima podataka	/	/	/
[172]	2005.	usporedba s drugim sustavima i testiranje s uzorcima podataka	simulirano	/	ne
[173]	2022.	testiranje s uzorcima podataka	simulirano	/	da
[174]	2023.	testiranje s napadima	stvarno i simulirano	pasivna	ne
[175]	2021.	testiranje s napadima i testiranje s ranjivostima sustava	stvarno	aktivna	ne
[176]	2021.	testiranje s uzorcima podataka i testiranje s napadima	simulirano	/	ne
[177]	2020.	usporedba s drugim sustavima i testiranje s uzorcima podataka	simulirano	/	ne

U istraživanju [168] analizirana su tri popularna *open-source* NIDS-alata: Snort, Suricata i Bro te je provedena usporedna analiza njihovih performansi. Pažnja je bila na čimbenicima koji ograničavaju primjenu tih NIDS-ova u velikim mrežama, uključujući iskorištavanje sistemskih resursa, brzinu obrade paketa i stopu gubitka paketa. Istraživanje je provedeno s podacima stvarne mreže gdje je sveukupni mrežni promet kopiran na uređaje namijenjene provedbi testiranja. Tijekom testiranja primijećene su promjene u performansama NIDS-ova ovisno o različitim konfiguracijama i pri različitim vrstama prometa.

U [169] autori predlažu arhitekturu za testiranje i vrednovanje IDS-sustava unutar vozila kako bi se omogućilo stvaranje realističnog mrežnog prometa i napada uzimajući u obzir specifične izazove u automobilskoj industriji. Koncept pruža mogućnost dijeljenja podataka bez dodatne anonimizacije, čime se poboljšava suradnja i reproducibilnost rezultata testiranja. Iako je osnovni cilj bio dizajnirati alat za procjenu IDS-a u automobilima, isti se može koristiti i za izvođenje funkcionalnih i sigurnosnih testova komunikacijskih mreža u automobilima. Kroz scenarije je moguće generirati benigni i zlonamjerni promet, primijeniti ih na različite konfiguracije vozila i podržati složene scenarije, uključujući napade poput "man-in-the-middle" napada. Za model zlonamjernog opterećenja razmotreno je pet scenarija napada, a postavljeni su i zahtjevi za generiranje stvarnog prometa u Ethernet-baziranim automobilskim mrežama, uključujući i prioritizaciju različitih tokova prometa.

U radu [170] predložen je sustav koji koristi više IDS-ova koji rješava problem centralizirane pohrane podataka. Za rješavanje tog problema, razvijen je glavni čvor koji centralizira upravljanje IDS-ovima i pruža primarnu bazu podataka za pohranu identificiranih prijetnji i nepravilnosti u mreži. Model testiranja uključuje kategorizaciju testova prema karakteristikama IDS-a koje se ispituju te cilja na stvaranje skupa testova koji omogućuju određivanje primjene IDS-a i konfiguracije parametara koji utječu na njegove performanse. Ispitivanje IDS-ova podijeljeno je u kategorije koje se bave performansama IDS-a, točnošću otkrivanja napada te arhitekturom IDS-a. Rezultati testova ukazuju na prednosti arhitekture Suricata IDS-a u brzjoj obradi podataka i boljoj točnosti otkrivanja napada u usporedbi s Snort IDS-om. Također, predložena arhitektura pokazuje potencijal za obradu mrežnog prometa visokog kapaciteta i smanjenje gubitka paketa.

Često se u testiranjima koristi stvarni snimljeni promet komunikacijskih mreža. U praksi, prikupljanje stvarnih mrežnih podataka u značajnim količinama često predstavlja izazov. Taj

nedostatak stvarnih i sveobuhvatnih podataka o mrežnom prometu može predstavljati ograničenje u istraživanjima otkrivanja kibernetičkih prijetnji, posebno u kontekstu analize ponašanja na razini aplikacije. Izostanak relevantnih podataka može rezultirati smanjenom preciznošću i relevantnošću modela. Dodatno, sadrže vrlo malo zlonamjernog prometa što nije dovoljno za potpuno testiranje IDS-a. Taj problem autori u [171] rješavaju predloženim sustavom za generiranje prometa koji kombinira stvarni promet s korisnički definiranim zlonamjernim HTTP-prometom s ciljem evaluacije IDS-a. U istraživanju je predstavljen sustav za generiranje kombiniranog prometa, s naglaskom na evaluaciju IDS-ova temeljenih na potpisima za otkrivanje mrežnih napada. Za kreiranje stvarnog prometa koristio se TRex generator prometa u kombinaciji sa zlonamjernim HTTP-prometom stvorenim putem GENESIDS generatora prometa, koji koristi definicije napada u obliku Snort-pravila. Time se osigurava stvaran promet pomiješan s velikim brojem događaja iz stvarnog svijeta.

U radu [172] je opisan okvir za generiranje prometa koji je namijenjen za *online*-ispitivanje sustava za otkrivanje mrežnih upada. Svrha sustava je generiranje stvarnih i raznolikih tokova normalnog i zlonamjernog prometa. Predložena je metodologija za generiranje normalnog prometa temeljenog na empirijskim skupovima podataka ili standardnih DARPA-podataka. Predstavljani pristup koristan je za testiranje i podešavanje vlastitih sustava otkrivanja upada. Okvir za generiranje prometa implementiran je u alatu koji je nazvan Trident, a njegova korisnost prikazana je kroz niz eksperimenata na open-source NIDS-ovima provedenih u kontroliranom laboratorijskom okruženju. Rezultati tih eksperimenata naglašavaju da sadržaj, kombinacija i obujam podataka imaju ogroman utjecaj na performanse NIDS-ova.

U radu [173] provođeni su testovi IDS-a koristeći InSDN-skup podataka, koji je prikupljen iz okoline s programski definiranom mrežom (SDN) i omogućuje provjeru učinkovitosti modela na SDN-prometu. Za izgradnju strukture modela korišten je *Tensorflow* okvir, a eksperimenti su izvedeni na platformi *Jupyter Notebook* unutar *Anaconda*-okruženja. Podatci korišteni u ovom radu potječu iz virtualnog okruženja sa softverski definiranom mrežom (SDN). *Ubuntu*-sustavi predstavljaju normalne korisnike, dok su *Linux Kali*-sustavi korisnici koji izvode različite vrste napada na SDN-mrežu. Izvedena su dva usporedna eksperimenta: jedan za binarnu klasifikaciju i drugi za višeklasnu klasifikaciju. Korišteni skup podataka sadrži različite vrste kibernetičkih napada, kao što su DDoS-napadi, Probe i drugi. U simuliranom scenariju napadač izvodi različite

vrste kibernetičkih napada na SDN-mrežu, dok ista ta mreža provodi normalan komunikacijski promet. Svrha je predloženog modela testiranja s mješovitim tokovima prometa razlikovanje normalnog prometa od zlonamjernog, dok je druga svrha analiza prometa i dodjela oznaka prometu za svaku vrstu napada.

U istraživanju [174] se predlaže sustav za otkrivanje upada temeljen na kvantitativnoj logaritamskoj transformaciji (QLT) za identifikaciju neželjenih aktivnosti u mrežama velikog opsega u stvarnom vremenu. Predloženi sustav, nazvan QLT-IDS, prikuplja i analizira NetFlow-zapise te analizira pojedinačna ponašanja tokova unutar tih zapisa koristeći statističku metodu temeljenu na QLT. Predloženi QLT-IDS izvodi otkrivanje upada na temelju statističkih značajki mrežnih tokova, ne zahtijeva upotrebu unaprijed definiranih pravila niti prethodni postupak obuke, a vrijeme otkrivanja je manje od jedne minute. Eksperimentalni rezultati pokazuju da QLT-IDS može obraditi masovne NetFlow-zapise i postići preciznost otkrivanja zlonamjernih upada od 95,7 % uključujući skeniranje portova, SSH-upade i RDP-upade.

Autori u [175] koriste NetFlow-skup podataka tokova mrežne komunikacije kako bi testirali IDS-a za SCADA-sustave. Predloženi IDS-sustav temeljen je algoritmu umjetne neuronske mreže dubokog učenja, a NetFlow-skup podataka korišten je za učenje i testiranje algoritama. U testnom SCADA-sustavu izveli su dva različita kibernetička napada. Promatrane su performanse IDS-a u *offline* i *online* načinima rada u testnoj SCADA-mreži. Performanse tijekom DoS-kibernetičkog napada pokazale su slične rezultate tijekom *online*-evaluacije i *offline*-evaluacije. S druge strane, za *reconnaissance*-kibernetički napad, *online*-performanse bile su lošije od *offline*-performansi. Istraživanje je poslužilo kako bi se simulirali stvarni scenariji kibernetičkih napada na SCADA-sustave.

U radu [176] predložen je inovativan okvir za generiranje bidirekcijskih skupova podataka temeljenih na tokovima mrežnog prometa radi evaluacije sustava za otkrivanje upada (IDS). Generirani skup podataka sadrži kombinaciju normalnog pozadinskog prometa i kibernetičkih napada. Pozadinski promet temelji se na standardnim karakteristikama mrežnog prometa, dok se generirani promet kibernetičkih napada temelji na njihovim statističkim karakteristikama. Predloženim okvirom generiran je skup podataka bidirekcijskog prometa temeljenog na tokovima koji se koristi za evaluaciju učinkovitosti sustava za otkrivanje upada.

U istraživanju [177] predstavljena je metodologija koja obuhvaća obvezne korake u procjeni NIDS-a. Primjenjivost predloženog okvira testirana je s klasičnim ML-algoritmima i stvarnim skupom podataka. Osim toga, izvršena je i usporedba s najnovijim NIDS-sustavima temeljenim na strojnom učenju. Rezultati su pokazali da predložena metodologija može pomoći u izgradnji boljih NIDS-rješenja kako bi omogućila njihove usporedbe. Međutim, definiranje konačnog okvira za procjenu nije trivijalno, jer može postojati više od jedne valjane metodologije ovisno o primjeni ili cilju, npr. za borbu protiv određene vrste napada u specifičnim mrežnim okolinama.

U pregledu znanstvenih radova o metodama verifikacije za IDS/IPS-sustave primijećeni su različiti pristupi i tehnike korištene za procjenu učinkovitosti i pouzdanosti tih sustava. Proučeni radovi obuhvaćaju recentno razdoblje, a metode verifikacije uključuju usporedbu s drugim sustavima, testiranje s uzorcima podataka, kibernetičkim napadima i ranjivostima sustava. Jedan od zaključaka iz analize je da postoji naglasak na korištenju simuliranih testnih okruženja, dok je stvarno okruženje manje zastupljeno. Također, većina proučenih sustava nije prilagođena za rad u okruženju programski definirane mreže (SDN). Unatoč raznolikosti metoda verifikacije, u nekim slučajevima primjetan je nedostatak testiranja sa stvarnim kibernetičkim napadima ili ranjivostima. Sukladno proučenoj literaturi, prijedlog za metodu verifikacije treba obuhvaćati sve nedostatke u dosadašnjim metodama. Potrebno je primijeniti metodu verifikacije na stvarno okruženje, umjesto isključive pažnje na simulirane scenarije. Također, potrebno je koristiti stvarne mrežne podatke kako bi se bolje ocijenila stvarna učinkovitost IDS-a u praktičnim situacijama. Predloženo rješenje verifikacije treba biti prilagođeno dinamičnim i promjenjivim karakteristikama SDN-okruženja. Uključivanje stvarnih kibernetičkih napada i ranjivosti predstavlja važan pogled u verifikaciji IDS-a kako bi se simulirali stvarni scenariji sigurnosnih prijetnji i načini ublažavanja neželjenog učinka kibernetičkih napada. Uočeno je da većina postojećih IDS/IPS-sustava preferira pasivni položaj. Česta orijentacija prema pasivnom nadzoru predlaže naglasak na ulozi sustava u analizi i praćenju prometa umjesto izravnom sudjelovanju u obrambenim aktivnostima. IDS/IPS-sustavi mogu se pozicionirati u mrežnom okruženju kroz aktivni ili pasivni način rada [178]. Aktivni način uključuje postavljanje uređaja izravno na putanju mrežnog prometa, omogućujući im analizu i brze akcije poput odbacivanja paketa ako se otkrije prijetnja. Ta metoda pruža bržu zaštitu, ali može uzrokovati kašnjenje u komunikaciji [179]. S

druge strane, pasivni način rada uključuje nadzor kopije prometa, generiranje upozorenja na zlonamjerne aktivnosti, ali nepoduzimanje izravnih mjera zaštite mrežne komunikacije [180]. U oba načina rada, IDS/IPS se implementira kao uređaj na 2. sloju mreže, osiguravajući transparentnost i mogućnost dodavanja sigurnosne komponente bez temeljnih promjena u konfiguraciji ili dizajnu mreže. Ta transparentnost omogućava integraciju sigurnosnih mjera bez narušavanja stabilnosti i funkcionalnosti postojeće mreže, pružajući tako dodatnu sigurnost i obranu od prijetnji [181]. Stoga, pristup pozicioniranja IDS/IPS-a može ovisiti o specifičnim sigurnosnim zahtjevima i ciljevima kibernetičke sigurnosti pojedine organizacije.

Ukupno gledano, ti prijedlozi sugeriraju korake za unapređenje metodologije verifikacije IDS-a, stvarajući poveznicu među teorijskim istraživanjima i praktičnim izazovima u stvarnim mrežnim scenarijima.

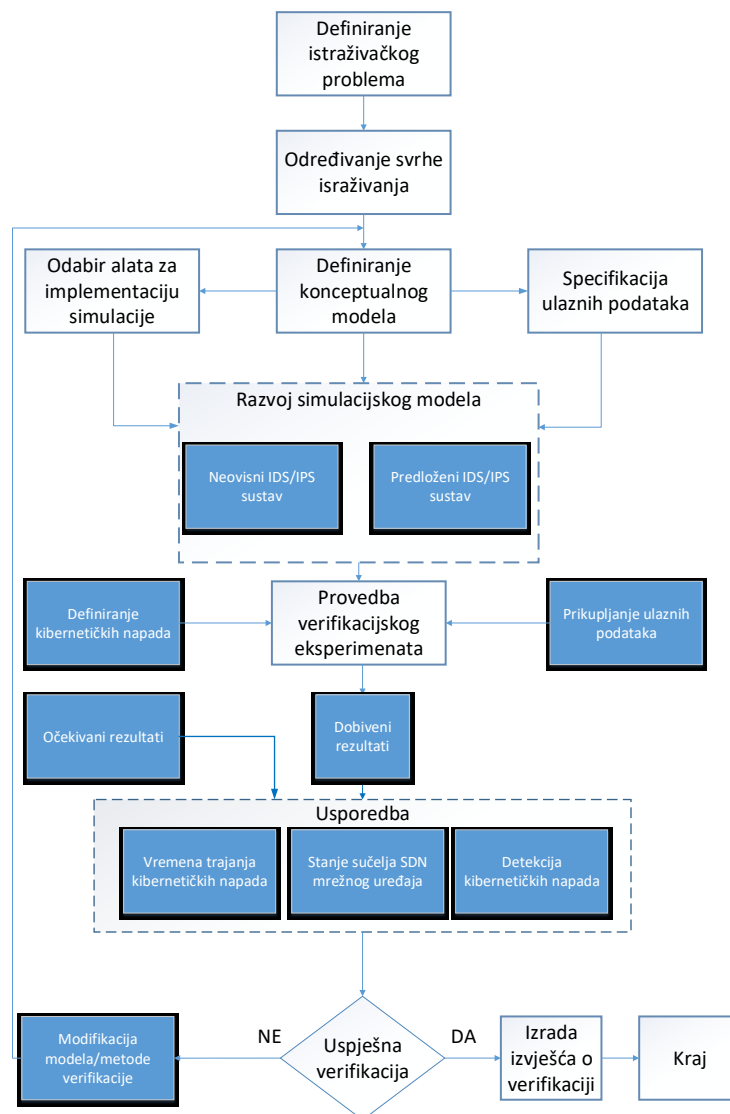
5.2. Prijedlog metode verifikacije modela za otkrivanje anomalija višestrukim NetFlow-zapisima mrežnog prometa

Pregled literature rezultirao je otkrivanjem nedostataka u metodama i procesu verifikacije modela za otkrivanje anomalija u mrežnom prometu, otvarajući mogućnost razvijanju nove metode verifikacije za predloženi NFMIDS-model. Primjena predložene metode verifikacije obuhvatila je nekoliko bitnih koraka, uključujući:

- Usporedbu s drugim IDS/IPS sustavom kako bi se stekle detaljnije informacije o performansama predloženog modela.
- Testiranje kombiniranim uzorcima NetFlow-podataka koji se sastoje od referentnih skupova podataka i stvarnog mrežnog prometa.
- Simulaciju stvarnih kibernetičkih napada radi oponašanja stvarnog scenarija.

Prijedlog nove metode verifikacije usmjeren je na verifikaciju za predloženi model otkrivanja anomalija temeljen na strojnom učenju i klasifikaciji NetFlow-podataka mrežnog prometa.

Prema slici 5.1. i definiranom tijeku procesa verifikacije, predložena metoda verifikacije na slici 5.2. uvodi dodatne korake kako bi se prilagodila različitim mrežnim arhitekturama, uključujući hibridnu SDN-arhitekturu.



Slika 5.2. Predložena metoda verifikacije

Metoda verifikacije prikazana na slici 5.2., uključuje sljedeće korake:

1. Definiranje istraživačkog problema: otkrivanje anomalija u hibridnoj programski definiranoj mreži.
2. Određivanje svrhe istraživanja: osiguranje kritičnih sustava od kibernetičkih napada.
3. Definiranje konceptualnog modela: predloženi NFMIDS-model integrira se u stvarnu mrežnu okolinu. Određuju se mrežni uređaji koji sudjeluju u simulaciji kibernetičkih napada, uz razlikovanje uloge ovisno o tipu mrežnog uređaja (tradicionalni ili SDN).
4. Specifikacija ulaznih podataka: ulazni podatci variraju ovisno o etapi procesa. Tijekom učenja NFMIDS-modela, koristi se kombinirani promet referentnih skupova podataka

- i NetFlow-zapisi iz mrežnih uređaja. U etapi verifikacije, ulazni su podatci isključivo izvezeni NetFlow-zapisi mrežnog prometa s odabranih mrežnih uređaja.
5. Odabir alata za implementaciju simulacije: integracija predloženog NFMIDS-modela s *Python*-skriptama za klasifikaciju mrežnog prometa i funkcionalnostima SDN-kontrolera i mrežnih uređaja. Paralelno se u verifikacijski proces uvodi neovisni IDS/IPS-sustav radi usporedbe i potvrde uspješnosti otkrivanja anomalija.
 6. Razvoj simulacijskog modela: uz predloženi NFMIDS-model koji povezuje *Python*-skripte za klasifikaciju mrežnog prometa s API-funkcionalnošću SDN-kontrolera i SDN-mrežnih uređaja, paralelno se u proces verifikacije ubacuje neovisni IDS/IPS-sustav pomoću kojega se može potvrditi i usporediti uspješnost predloženog modela za otkrivanje anomalija mrežnog prometa.
 7. Provedba verifikacijskog eksperimenta: slijedno se u određenim vremenskim intervalima provode ranije definirani kibernetički napadi, klasifikacijom dolaznih NetFlow-podataka dobivaju se rezultati otkrivanja anomalija. Temeljem unaprijed zadanih vremenskih intervala određenih kibernetičkih napada, uspoređuju se očekivani rezultati s dobivenim rezultatima provedenog verifikacijskog eksperimenta.
 8. Usporedba rezultata: uspoređuju se očekivani i dobiveni rezultati triju parametara verifikacijskog procesa: vrijeme izvršenja i trajanje kibernetičkih napada, stanje sučelja SDN-uređaja tijekom otkrivanja anomalija te uspješnost otkrivanja pojedinih kibernetičkih napada.
 9. Verifikacija predloženog IDS/IPS-modela: temeljem usporedbe rezultata donosi se ocjena verifikacije modela. U slučaju negativne ocjene, proces se vraća na etapu definiranja konceptualnog modela radi primjene drugih ulaznih vrijednosti i načina provedbe verifikacijskog postupka.
 10. Dokumentiranje: za pozitivno ocijenjenu verifikaciju modela preporučuje se detaljno dokumentiranje procesa i rezultata kako bi se omogućio razvoj novih komponenata predloženog modela.

Predložena metoda verifikacije modela za otkrivanje anomalija u hibridnoj programski definiranoj mreži donosi nekoliko značajnih prednosti u odnosu na postojeće metode opisane u recentnoj literaturi.

Implementacija u hibridnom SDN-okruženju predstavlja primarnu prednost, posebno zbog karakteristika hibridnih programski definiranih mreža koje integriraju klasičnu i SDN-arhitekturu. To omogućuje promjenu statusa sučelja u SDN-preklopniku, što je jedan od relevantnih podataka za ocjenu verifikacije. S obzirom na upotrebu strojnog učenja i NetFlow-podataka u predloženom modelu, nužno je osigurati arhitekturu koja će optimalno iskoristiti te komponente i biti primjenjiva u različitim sustavima.

Kroz korištenje višestrukih NetFlow-zapisa podataka, metoda simulira raznolike scenarije koji uključuju normalan mrežni promet i različite kibernetičke napade. Taj pristup omogućava precizno oblikovanje NetFlow-zapisa potrebnih za učinkovito otkrivanje anomalija. Predložena metoda omogućava slijedno testiranje različitih scenarija kibernetičkih napada, uz usporedbu rezultata otkrivanja drugih, neovisnih IDS/IPS-sustava.

Predložena metoda verifikacije transparentna je u pogledu položaja IDS/IPS-sustava, omogućujući oba načina pozicioniranja koji su striktno definirani u postojećim rješenjima. Prednost te transparentnosti očituje se u mogućnosti SDN-uređaja da se programski prilagode novonastalim uvjetima na mreži, poput otkrivanja anomalija i kibernetičkih napada. Implementacija modela za otkrivanje anomalija u stvarno okruženje također povlači implementaciju verifikacije u stvarno okruženje, što pridonosi vjerodostojnosti rezultata te metode.

Kombinirani skup podataka stvarnog mrežnog prometa s izazvanim kibernetičkim napadima pruža prednost u odnosu na testiranje samo s napadima ili fiksnim uzorcima podataka što je uobičajeno u postojećim metodama verifikacije. Osim što omogućuje širu pokrivenost mogućih scenarija, kombinirani skup podataka pruža realističniji prikaz stvarnih mrežnih uvjeta i ponašanja. Također, omogućuje verifikaciju modela pod različitim uvjetima opterećenja i promjenjivim mrežnim okolinama. S druge strane, testiranje samo s napadima ili fiksnim uzorcima podataka može propustiti složenije scenarije ili neobične situacije koje se mogu pojaviti u stvarnim mrežnim uvjetima, stoga je korištenje kombiniranog skupa podataka prednost za provjeru sigurnosti i performansi promatranog modela.

Konačno, dokumentacija procesa verifikacije pruža transparentnost i olakšava daljnje unapređenje modela, čime se postiže integracija ovog modela u stvarne svakodnevne operacije osiguranja kritičnih sustava od kibernetičkih prijetnji.

Predložena metoda verifikacije je potpuno adekvatna za model otkrivanja anomalija u hibridnoj programski definiranoj mreži jer obuhvaća sve relevantne komponente modela i njihovu funkcionalnost. Taj postupak verifikacije jasno pokazuje aktivnu ulogu NFMIDS-modela u otkrivanju anomalija mrežnog prometa putem onemogućavanja sučelja koje napadač koristi za izvršavanje kibernetičkih napada. Dodatno, omogućuje precizno definiranje konfiguracije izvoza NetFlow-podataka; u slučaju neispravnog NetFlow-zapisa, verifikacija ne može biti potvrđena, što usmjerava model prema ispravnom konfiguriranju izvoza NetFlow-zapisa.

Mogućnost paralelne usporedbe očekivanih i stvarnih rezultata s neovisnim IDS/IPS-sustavom doprinosi važnosti procesa verifikacije i dobivenih rezultata. Ipak, u predloženoj metodi verifikacije postoji ograničenje u usporedbi s neovisnim IDS/IPS-sustavom zbog nedostupnosti informacija o stanju sučelja SDN-preklopnika. Budući da neovisni IDS/IPS sustav nema pristup tim informacijama, usporedba rezultata s njim ostaje na informativnoj razini, unatoč tomu pridonosi relevantnosti rezultata verifikacije.

5.3. Verifikacija predloženog modela za otkrivanje anomalija mrežnog prometa predloženom metodom višestrukih Netflow-zapisa

Prema predloženoj metodi verifikacije, koja obuhvaća korake definirane u poglavlju 5.2, prikazanim na slici 5.2., izvršena je verifikacija NFMIDS-modela. Detaljno objašnjenje implementacije pojedinih koraka opisane metode nalazi se u nastavku.

5.3.1. Definiranje istraživačkog problema

Istraživački problem definiran je kao otkrivanje anomalija u mrežnom prometu unutar hibridne programski definirane mreže korištenjem tehnika strojnog učenja.

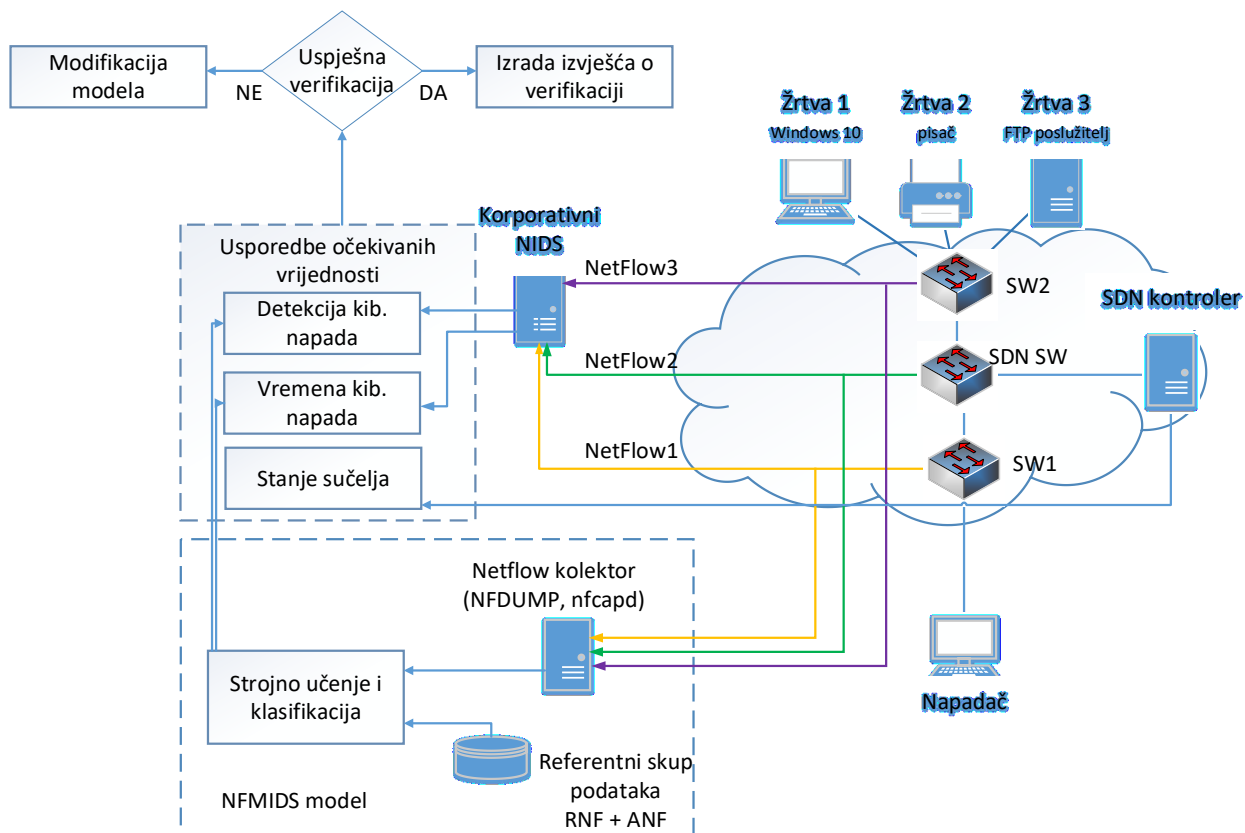
5.3.2. Određivanje svrhe istraživanja

Svrha je istraživanja osigurati kritične sustave poslovanja, reprezentirane kroz tri različita uređaja nazvana Žrtva 1, Žrtva 2 i Žrtva 3, kako je prikazano na slici 5.3.

5.3.3. Definiranje konceptualnog modela

Konceptualni model, opisan prethodno, podvrgnut je verifikaciji implementacijom predloženog NFMIDS-modela u stvarnoj mrežnoj okolini. Uključeni su stvarni mrežni uređaji korporativne IP-mreže koji omogućuju komunikacijsko povezivanje i prikupljanje NetFlow-zapisa radi klasifikacije mrežnog prometa i otkrivanja anomalija. Cisco N9K-C93108TC-FX (SDN SW) preklopnik i SDN-kontroler dio su hibridne programski definirane mreže, dok su Cisco C9300 uređaji (SW1 i SW2) tradicionalni uređaji. Korporativni NIDS integriran je u verifikacijski proces za usporedbu vremena i otkrivanje anomalija.

Verifikacija NFMIDS-modela otkrivanja anomalija mrežnog prometa obuhvaćala je NetFlow-zapise s trima preklopticima koji su na putu napadača prema jednoj od žrtava. Svi rezultati otkrivanja anomalija uspoređeni su s otkrivanjima *Cisco Secure Network Analytics* NIDS-a u korporativnoj mreži. *Cisco Secure Network Analytics*, nekada poznat kao *Stealthwatch* [182], alat je za analizu sigurnosti i praćenje prijetnji unutar komunikacijske mreže koristeći složeno modeliranje ponašanja, strojno učenje i globalne informacije o kibernetičkim prijetnjama. Sve aktivnosti provedene su unutar stvarne hibridne programski definirane mreže, što osigurava relevantnost rezultata u kontekstu specifičnog okruženja.



Slika 5.3. Arhitektura mrežne okoline pri verifikaciji NFMIDS-modela

Na slici 5.3. prikazana je arhitektura mrežnog okruženja i procesa na kojoj je primijenjena verifikacija NFMIDS-modela korištenjem predložene metode višestrukih NetFlow-zapisa. Glavna karakteristika prikazanog mrežnog okruženja je hibridna programski definirana mreža koja obuhvaća tradicionalne preklopnike (SW1 i SW2) te SDN-preklopnik (SDN SW). Tim preklopnicima povezani su uređaji-žrtve kibernetičkih napada i računalo napadača, kako je prikazano na slici 5.3. Konfiguracija izvoza NetFlow-zapisa na svakom preklopniku prilagođena je specifičnim mogućnostima pojedinog uređaja, što je detaljno opisano u prilogu A za preklopnike SW1 i SW2 te u prilogu B za SDN SW preklopnik. Svi NetFlow-zapisi usmjereni su na obradu (klasifikaciju) u NFMIDS te na neovisni korporativni NIDS-sustav prije samog postupka verifikacije. NFMIDS je prethodno naučen za klasifikaciju kombiniranim skupom podataka koji sadrži stvarne NetFlow-zapise i zapise referentnog skupa podataka (anomalije).

Specifikacija i karakteristike uređaja korištenih u procesu verifikacije predloženog modela opisane su u tablici 5.2.

Tablica 5.2. Specifikacija korištenih uređaja tijekom procesa verifikacije

uređaj	model	operativni sustav/aplikacija
napadač	HP Probook 6570b	Kali Linux
žrtva1	HP Probook G455	Windows 10
žrtva2	Pisač Lexmark CX510de	/
žrtva3	HP EliteDesk 800 G4	Windows 10/FileZilla FTP Server 1.6.7
NIDS	Cisco Secure Network Analytics	Version 7.4.1
NFMIDS	Virtualni VMware ESXi poslužitelj	Linux Ubuntu 20.04, <i>Python</i> 3.8
SDN-kontroler	Cisco APIC Server M1	APIC Version 4.2(7v)
NetFlow-kolektor	nfcapd	Version: NSEL-NEL1.6.18;
SW1	Cisco Catalyst 9300 Switch	Cisco IOS Software [Everest], Catalyst L3 Switch Software (CAT9K_IOSXE), Version 16.6.6
SW2	Cisco Catalyst 9300 Switch	Cisco IOS Software [Everest], Catalyst L3 Switch Software (CAT9K_IOSXE), Version 16.6.6
SDN SW	Cisco Nexus 93108TC-FX	Cisco NX-OS(tm) aci, Software (aci-n9000-system), Version 14.2(7v)

Kibernetički su napadi trajali tri minute, s intervalima od također tri minute za uspostavljanje normalnog režima rada i eventualno otkrivanje lažnih klasifikacija. Napad skeniranja portova bio je kraći jer je skeniranje završilo u samo dvije minute. Usporedba i analiza otkrivanja anomalija obuhvatila je jedan od triju tipova kibernetičkih napada izvršenih s računala napadača prema jednoj od žrtava, ovisno o vrsti napada definiranoj u poglavlju 4.2.2:

- tip 1 - DoS – uskraćivanje usluge
- tip 2 - otkrivanje korisničkih imena i lozinki – dobivanje neovlaštenog pristupa
- tip 3 - skeniranje portova na udaljenom uređaju.

Tijekom postupka verifikacije pojedinačno su propušteni Netflow-zapisi sa svakog preklopnika. Ciljano je stanje u stvarnoj infrastrukturi prikupljanje NetFlow-zapisa sa svih uređaja koji podržavaju slanje NetFlow-zapisa u kolektor. Tijekom verifikacije, IPS-funkcionalnost u predloženom NFMIDS modelu nije bila aktivna kako bi se izbjeglo prekidanje verifikacijskog procesa. Umjesto toga, programiran je API-zahtjev prema SDN-kontroleru za zaustavljanje ili ponovno omogućavanje sučelja na SDN-preklopniku.

5.3.4. Specifikacija ulaznih podataka

Ulazni su podatci definirani prema njihovoj upotrebi tijekom procesa verifikacije. U etapi učenja, korišten je kombinirani skup podataka koji obuhvaća stvarni mrežni promet i anomalije iz referentnog skupa podataka. Za klasifikaciju, ulazni podatci sastoje se isključivo od NetFlow-zapisa koje izvoze mrežni uređaji. Tijekom učenja, ti su podatci direktno učitani u *Python*-skripte za strojno učenje, dok su u etapi klasifikacije podatci proslijeđeni preko NetFlow-kolektora u iste skripte.

Proces verifikacije provoden je u stvarnoj mrežnoj okolini, prikazanoj na slici 5.2., koristeći NetFlow-zapise s trima preklopticima, od kojih je jedan u SDN-paradigmi (SDN SW). Različite konfiguracije NetFlow-zapisa, opisane u prilogu A i prilogu B disertacije, variraju prema mogućnostima svakog preklopnika. Na primjer, NetFlow1 i NetFlow3 zapisima promatraju se ulazno i izlazno sučelje paketa, dok se u NetFlow2 zapisu promatraju samo ulazni paketi. Konfiguracije se razlikuju i po mogućnostima "*MATCH*" i "*COLLECT*" naredbi koje određuju koji paketi i koje podatke NetFlow-zapisa prikupljaju.

Uspoređujući rezultate vremena izvođenja, otkrivanja kibernetičkih napada i stanje sučelja SDN-preklopnika tijekom aktiviranja napada uz određene NetFlow-zapise, može se verificirati predloženi IDS/IPS-model. Ta metoda verifikacije karakterizira se robusnošću jer ne ovisi o broju različitih NetFlow-zapisa, što može poboljšati preciznost odluka. Također, omogućuje usporedbu različitih tipova neovisnih IDS-ova.

Prije bilo kakvog kibernetičkog napada, važno je provjeriti ispravnost izvoza NetFlow-zapisa i njihovo uspješno prikupljanje od strane testiranih NFMIDS i korporativnog NIDS-a. To je

presudno za učinkovito otkrivanje i analizu napada jer problemi s izvozom NetFlow-zapisa mogu otežati otkrivanje i smanjiti učinkovitost sigurnosnih mjera. Korištenjem *traceroute* alata (prilog C), pruža se detaljan uvid u rute prometa kroz mrežu i potvrđuju se očekivane putanje mrežnog prometa, što je bitno za verifikaciju, kao što je prikazano na slici 5.2.

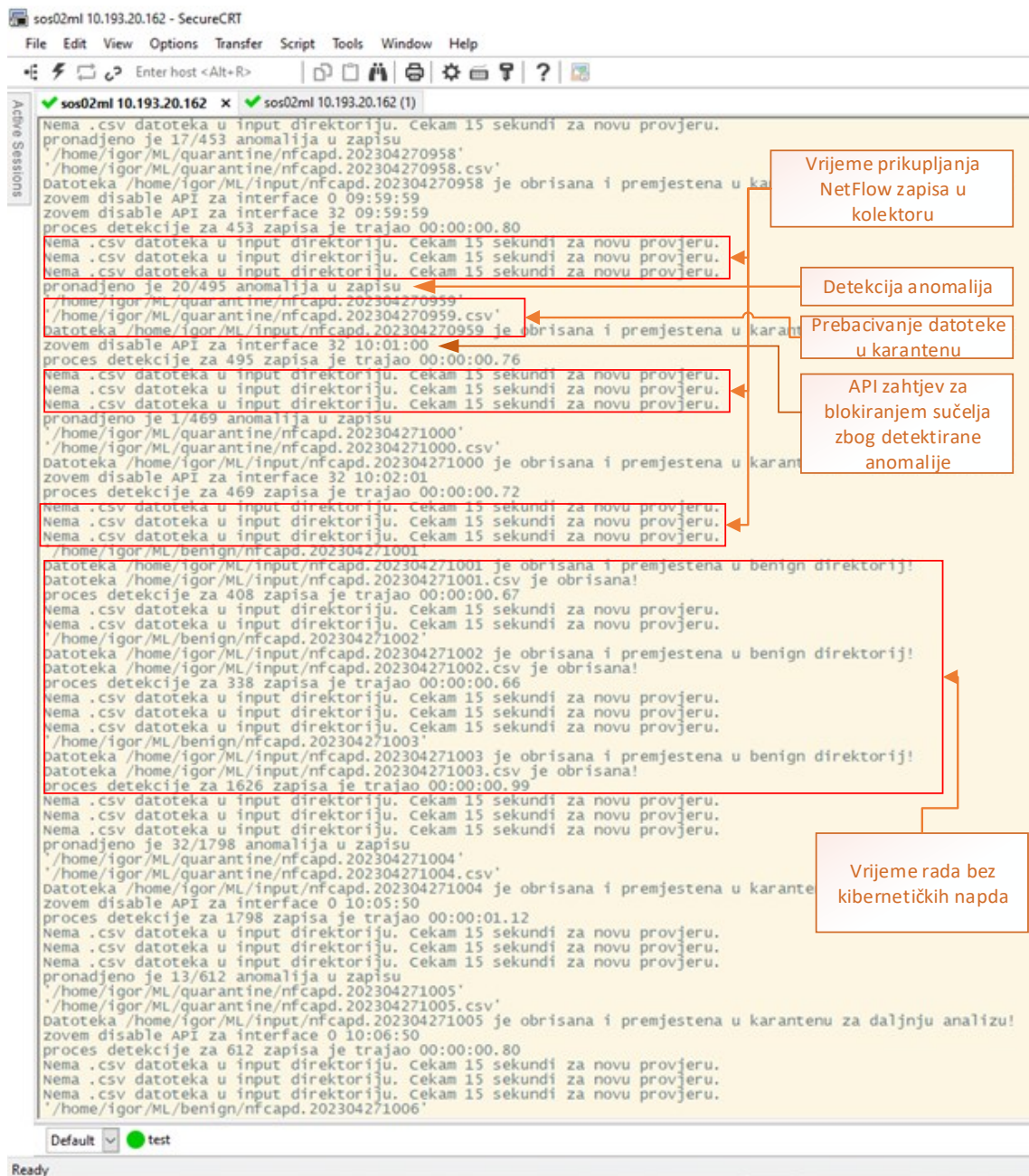
5.3.5. Odabir alata za implementaciju simulacije

Python-programski jezik odabran je za implementaciju klasifikacije strojnim učenjem. Također, *Python* se koristio za izvršavanje API-zahtjeva prema SDN-kontroleru. Za provođenje različitih kibernetičkih napada na kritični sustav, korištena je Kali Linux-distribucija operativnog sustava računala napadača.

5.3.6. Razvoj simulacijskog modela

U sklopu razvoja simulacijskog modela, implementiran je NFMIDS-model koji integrira *Python*-skripte za klasifikaciju mrežnog prometa s API-funkcionalnostima SDN-kontrolera i SDN-mrežnih uređaja. Uz predloženi model, proces verifikacije obogaćen je uključivanjem neovisnog korporativnog NIDS-sustava, što omogućuje potvrdu i usporedbu učinkovitosti predloženog modela u otkrivanju anomalija mrežnog prometa. Slika 5.4. prikazuje isječak rada NFMIDS-a u stvarnom vremenu, uz prisutnost pozadinskog prometa. Na slici su označeni dijelovi koji se odnose na prikupljanje NetFlow-zapisa, otkrivanje anomalija, prebacivanje datoteke u karantenu, API-zahtjeve za zaustavljanje sučelja te vrijeme normalnog rada mreže (bez prisutnosti kibernetičkih napada). Prikaz cjelokupnog testiranja nalazi se u prilogu F ove disertacije, zajedno s komentarima o vremenima početka i završetka generiranih kibernetičkih napada, kao i trajanju normalnog režima rada mreže. Tijekom normalnog režima rada mreže, identificirana su i označena i lažna otkrivanja anomalija. Pozadinski promet uključuje uobičajene aktivnosti korisnika

korporativne mreže, koji nisu bili obaviješteni o testiranju NFMIDS-a, te na taj način, normalan promet predstavlja reprezentativni uzorak stvarne mrežne okoline.



Slika 5.4. Isječak prikaza rada NFMIDS-a u stvarnom vremenu s kombiniranim pozadinskom prometom

5.3.7. Provedba verifikacijskog eksperimenta

Tijekom verifikacijskog eksperimenta, unaprijed definirani kibernetički napadi (DoS – uskraćivanje usluge, otkrivanje korisničkih imena i lozinki – dobivanje neovlaštenog pristupa, skeniranje portova na udaljenom uređaju) su se sukcesivno izvodili u točno određenim intervalima. Klasifikacijom dolaznih NetFlow-podataka dobiveni su rezultati otkrivanja anomalija. Sva tri mrežna uređaja konfigurirana su prema svojim mogućnostima za izvoz NetFlow-podataka u kolektor kako je opisano u prilogima A i B. Ovisno o klasifikaciji dolaznih NetFlow-podataka sa svakog mrežnog uređaja, spremaju se datoteke NetFlow-zapisa u direktorije karantene ili benignog prometa, kako bi se naknadno verifikacijski eksperiment mogao poboljšati, ukoliko se pokaže potreba.

Tijekom procesa testiranja odnosno verifikacije predloženog modela s NetFlow1 konfiguracijom, sustavi NFMIDS i NIDS su otkrili sve tri vrste kibernetičkih napada, s malim postotkom lažno pozitivnih rezultata otkrivanja (1,15 %) za NFMIDS-model. Međutim, za korporativni NIDS nije bilo moguće odrediti postotak lažno pozitivnih otkrivanja anomalija (oznaka * u tablici 5.3. za NIDS-rezultate lažno pozitivnih rezultata), budući da ne postoji uvid u ukupan broj NetFlow-zapisa prispjelih iz preklopnika SW1 kao niti koliko je takvih zapisa rezultiralo kao lažno pozitivne klasifikacije koje su stavljene u grupu otkrivenih kibernetičkih napada. Korporativni NIDS pruža samo grupirane podatke o uočenim sigurnosnim incidentima, što je prikazano na slici 5.5.

Jedan od značajnih nedostataka korporativnog NIDS-sustava leži u činjenici da su sigurnosni incidenti prikazani po specifičnim računalima ili IP-adresama, a u ovom kontekstu nužno je poznavati IP-adresu napadača (10.193.91.100) kako bi se dobila kompletna lista sigurnosnih incidenata, kao što je prikazano na slici 5.5.

Top Security Events for 10.193.91.100

Security Event	Count
Port Scan - 49158	260
Brute Force Login - 21	1
High Traffic	1

Slika 5.5. Prikaz otkrivanja sigurnosnih incidenata u korporativnom NIDS-u

Vremena otkrivanja kibernetičkih napada u predloženom NFMIDS-modelu te u korporativnom NIDS-sustavu u skladu su s vremenom izvršavanja kibernetičkih napada. Stanje sučelja SDN-preklopnika na koji je spojeno računalo napadača odnosno izvršitelja kibernetičkih napada, praćeno je programiranim API-zahtjevom za onemogućavanjem sučelja, vidljivo na slici 5.4. Korporativni IDS nema mogućnost praćenja stanja sučelja SDN-preklopnika, stoga usporedba tog podatka nije dio odluke za uspješnost verifikacije, što je objašnjeno u poglavlju 5.2.

Uz NetFlow2-konfiguraciju i SDN-preklopnik (SDN SW), predloženi NFMIDS nije dao jasne rezultate s obzirom na to da su otkrivanja sigurnosnih incidenata upitne zbog velikog broja lažno pozitivnih otkrivanja, čak 85 %, dok NIDS-sustav nije otkrio niti jednu vrstu kibernetičkog napada. Razlog tako velikom postotku lažno pozitivnih otkrivanja anomalija leži u razlici konfiguracije NetFlow-izvoza iz SDN-mrežnog uređaja. Za korištenje predloženog NFMIDS-a potrebna je konfiguracija po ulaznom i izlazom sučelju te s više parametara NetFlow-zapisa, u SDN-preklopniku s postojećom verzijom pripadajućeg softvera nije moguće podesiti izvoz NetFlow-zapisa po željenoj specifikaciji. Visok postotak lažno pozitivnih otkrivanja izračunat je temeljem detaljne analize rada NFMIDS-a tijekom testiranja, koja pokazuje koliko je zapisa unutar NetFlow-zapisa klasificirano kao anomalije u odnosu na ukupan broj zapisa. Proces klasifikacije strojnog učenja optimiziran je za zadani skup podataka (NetFlow-zapisi) kako bi se postigli najbolji rezultati otkrivanja. Vremena otkrivanja anomalija nisu bila u očekivanim vrijednostima zbog

velikog broja lažno pozitivnih klasifikacija, što je rezultiralo konstantnim API-zahtjevima za onemogućavanjem sučelja SDN-preklopnika, što nije usklađeno s vremenima izvršenja stvarnih kibernetičkih napada. Proces verifikacije uz NetFlow2-konfiguraciju pokazuje da takva konfiguracija nije pogodna za predložen model otkrivanja anomalija mrežnog prometa. Za očekivati je da će se konfiguracije NetFlow-zapisa za SDN-uređaje u budućnosti unaprijediti i omogućiti preciznije otkrivanje anomalija u predloženom modelu.

Konačno, za NetFlow3 konfiguraciju s SW2 preklopnikom, NFMIDS je uspješno otkrio sve tri vrste kibernetičkih napada, dok je korporativni NIDS identificirao samo dvije od tri vrste kao što je prikazano na slici 5.6. NFMIDS se istaknuo niskim postotkom lažno pozitivnih rezultata otkrivanja (0,12 %), što ukazuje na preciznost modela u identifikaciji stvarnih sigurnosnih prijetnji.

Top Security Events for 10.193.91.100

	Security Event	Count
▶	Port Scan - 554	131
▶	High Traffic	1
▶	Reset/tcp - 873	2
▶	Reset/tcp - 306	4
▶	Reset/tcp - 990	4
▶	Reset/tcp - 992	4
▶	Reset/tcp - 903	2
▶	Reset/tcp - 765	4

Slika 5.6. Prikaz sigurnosnih incidenata i lažno pozitivnih otkrivanja u aplikaciji korporativnog NIDS-a

S druge strane, za NIDS-sustav nije moguće odrediti postotak lažno pozitivnih otkrivanja zbog nedostatka informacija o ukupnom broju prikupljenih NetFlow-zapisa tijekom normalnog režima rada bez kibernetičkih napada. U tablici 5.3. označeno je prisustvo lažnih otkrivanja *Reset/tcp (**)* koje su grupirane i prikazane ukupnim brojem tijekom cijelog eksperimenta, što otežava određivanje specifičnog intervala u kojem su se ta lažna otkrivanja anomalija dogodile. *Reset/tcp* u aplikaciji korporativnog NIDS-a opisan je kao: "*Izvorišno računalo je poslalo TCP-paket koji je odbijen od strane ciljnog računala. Taj sigurnosni događaj obično je posljedica neispravnih aplikacija, prepoznavanja operativnog sustava ili skeniranja TCP-porta.*".

5.3.8. Usporedba rezultata

Uspoređeni su očekivani i dobiveni rezultati triju parametara verifikacijskog procesa: vrijeme izvršenja i trajanje kibernetičkih napada, stanje sučelja SDN-preklopnika pri otkrivanju anomalija mrežnog prometa ili kibernetičkih napada te uspješnost otkrivanja pojedinog kibernetičkog napada. Očekivani rezultati definirani su provedbom verifikacijskog procesa, gdje su poznata vremena izvršenja i trajanja kibernetičkih napada s računala napadača. Također, poznato je sučelje na SDN-preklopniku koje je povezano s računalom napadača.

Dodatna usporedba s rezultatima korporativnog NIDS-a pospješuje verifikacijski proces, ali nije moguće usporediti stanje sučelja SDN-preklopnika jer korporativni NIDS ne pruža takve informacije. Usporedba je provedena ručno (zapisivanjem i pregledom promatranih parametara) zbog manjeg broja promatranih parametara, ali u slučaju povećanja broja parametara preporuča se automatizirana usporedba radi ubrzanja procesa.

Vremena otkrivanja u predloženom NFMIDS-u poklapaju se s očekivanim vremenima izvršenja svih triju kibernetičkih napada, kao i vremena API-zahtjeva za onemogućavanjem sučelja SDN-preklopnika. Korporativni NIDS pokazao je očekivana vremena otkrivanja kibernetičkih napada koji su bili otkriveni, osim kibernetičkog napada tipa 3 (otkrivanje korisničkih imena i lozinki – dobivanje neovlaštenog pristupa), koji nije bio otkriven u očekivanom vremenu izvršenja.

U tablici 5.3. su sažeti rezultati testiranja izvršenih i otkrivenih kibernetičkih napada, koji su se odnosili na različite konfiguracije izvoza NetFlow-zapisa svih mrežnih preklopnika. Za izračun apsolutnog i relativnog broja lažno pozitivnih otkrivanja analizirana su otkrivanja tijekom razdoblja normalnog prometa, kada kibernetički napadi nisu bili aktivni.

Tablica 5.3. Rezultati i usporedba otkrivanja kibernetičkih napada tijekom procesa verifikacije

vrsta NetFlow-konfiguracije	IDS-sustav	otkrivanje kibernetičkih napada			lažno pozitivni rezultati otkrivanja
		Tip1	Tip2	Tip3	
NF1 (SW1)	NFMIDS	da	da	da	4/345 (1,15 %)
	NIDS	da	da	da	ne *
NF2 (SDN SW)	NFMIDS	da	da	da	1900/2218 (85 %)
	NIDS	ne	ne	ne	ne
NF3 (SW2)	NFMIDS	da	da	da	17/13906 (0,12 %)
	NIDS	da	da	ne	da**

Važno je napomenuti da je analiza lažnih otkrivanja provedena samo za predloženi NFMIDS, jer je moguć detaljan uvid u NetFlow-zapise. Nažalost, korporativni NIDS nije pružio istu mogućnost za dublji uvid u pristigle NetFlow-zapise. Detaljan pregled NetFlow-zapisa od strane NFMIDS-a omogućio je preciznije izračune apsolutnog i relativnog broja lažno pozitivnih klasifikacija, što je ključno za bolje razumijevanje učinkovitosti sustava otkrivanja.

Iz tablice 5.3. može se zaključiti da su rezultati otkrivanja kibernetičkih napada varirali ovisno o konfiguraciji izvoza NetFlow-zapisa i IDS-sustavu korištenom u procesu verifikacije. Primijećeno je da je u testiranju s pozadinskim prometom (NetFlow3-zapisi) postotak lažno pozitivnih otkrivanja bio povoljniji (manji) u usporedbi s testiranjem bez pozadinskog prometa (NetFlow1-zapisi). To je rezultat većeg broja pristiglih NetFlow-zapisa u nfcapd-kolektor, što je omogućilo bolju klasifikaciju i otkrivanje neželjenog prometa. Visoka apsolutna i relativna stopa lažno pozitivnih rezultata tijekom testiranja predloženog NFMIDS-a s NetFlow2-konfiguracijom izvoza NetFlow-zapisa proizlazi iz neadekvatno prilagođene konfiguracije izvoza NetFlow2-zapisa. Ta se konfiguracija nije pokazala povoljnom za otkrivanje anomalija ni u predloženom NFMIDS-u ni u korporativnom NIDS-u, koji nije uspio identificirati niti jedan kibernetički napad. Iako je NFMIDS-model uspješno identificirao sva tri kibernetička napada, problem leži u nedostatku jasnog razdvajanja stvarnih prijetnji i lažno pozitivnih otkrivanja. Taj visoki postotak

lažno pozitivnih rezultata čini takvo otkrivanje anomalija nepovoljnom za donošenje relevantnih odluka o sprječavanju i prevenciji kibernetičkih napada.

5.3.9. Verifikacija predloženog IDS/IPS-modela

Temeljem usporedbe rezultata, daje se ocjena verifikacije modela prema tablici 5.4. U slučaju negativne ocjene, kao što je bio slučaj s korištenjem NetFlow-zapisa za klasifikaciju dobivenih iz SDN SW-preklopnika, postupak se vraća na etapu definiranja konceptualnog modela. U toj je etapi potrebno razmotriti primjenu drugih ulaznih vrijednosti ili izmijeniti način provedbe verifikacijskog postupka. Kod pozitivne ocjene verifikacije, predloženi je model spreman za implementaciju i upotrebu u stvarnoj produkcijskoj okolini, pružajući zaštitu kritičnog sustava od kibernetičkih napada. U skladu s predloženom metodom verifikacije, tablica 5.4. prikazuje postignute rezultate tijekom verifikacije u odnosu na očekivane vrijednosti promatranih parametara za ocjenu verifikacije predloženog NFMIDS modela za otkrivanje anomalija.

Tablica 5.4. Ocjena verifikacije predloženog NFMIDS-modela

				otkrivanje kib. napada	stanje sučelja SDN preklopnika	očekivano vrijeme i trajanje otkrivanja kib. napada	verifikacija
vrsta NetFlow-konfiguracije	NF1 (SW1)	Tip kib. napada	1	da	zaustavljen	da	da
			2	da	zaustavljen	da	
			3	da	zaustavljen	da	
	NF2 (SDN SW)	Tip kib. napada	1	da	zaustavljen	ne	ne
			2	da	zaustavljen	ne	
			3	da	zaustavljen	ne	
	NF3 (SW3)	Tip kib. napada	1	da	zaustavljen	da	da
			2	da	zaustavljen	da	
			3	da	zaustavljen	da	

Prema predloženoj metodi verifikacije, NFMIDS-model uspješno je verificiran u dvama slučajima kada je izvoz NetFlow-zapisa na preklopniku bio konfiguriran na preporučeni način. U slučaju negativne ocjene verifikacije pri korištenju NF2 konfiguracije NetFlow-zapisa, sučelje je

konstantno bilo onemogućeno (zaustavljeno) zbog izrazito visokog broja lažno pozitivnih rezultata klasifikacije, što je detaljno objašnjeno u tablici 5.3. Iako su svi kibernetički napadi bili otkriveni u tom slučaju, treba napomenuti da je zabilježen izrazito visok broj lažno pozitivnih klasifikacija, što nije u skladu s očekivanim vremenima i trajanjima otkrivanja kibernetičkih napada. Iz tog razloga, verifikacija predloženog modela nije ocijenjena pozitivno pri korištenju NF2-konfiguracije NetFlow-izvoza podataka. S druge strane, kada su korištene NF1 ili NF3 konfiguracije NetFlow-izvoza podataka, model je postigao očekivane rezultate otkrivanja i trajanja svih triju kibernetičkih napada, uz adekvatno stanje sučelja SDN-preklopnika tijekom otkrivanja kibernetičkih napada.

Predložena metoda verifikacije ističe da uspješnost otkrivanja kibernetičkih napada tijekom procesa verifikacije, kao i postotak lažno pozitivnih rezultata otkrivanja, značajno ovise o odabiru konfiguracije izvoza NetFlow-zapisa i IDS-sustava. Verifikacija je obuhvatila kombinaciju metoda s pozadinskim prometom i bez njega, uključujući generički promet predstavljen kibernetičkim napadima. Korišteno je pasivno postavljanje NFMIDS-modela, no uključivanjem IPS-funkcionalnosti omogućeno je aktivno djelovanje koje reagira na otkrivene anomalije.

5.3.10. Dokumentiranje

Pozitivno ocijenjena verifikacija je dokumentirana detaljnim opisom postupka, uključujući shemu topologije mrežne arhitekture, korištene *Python*-skripte te konfiguracije NetFlow-izvoza na svakom tipu mrežnih preklopnika.

Tijekom verifikacije identificirani su nedostaci i prednosti ispitivanog NFMIDS-a, što može usmjeriti daljnji razvoj i poboljšanje modela. Uspoređujući ga s postojećim korporativnim IDS-sustavom, NFMIDS-model je pokazao veću fleksibilnost u korištenju NetFlow-zapisa, što omogućuje smanjenje lažno pozitivnih otkrivanja dodatnom inspekcijom NetFlow-zapisa koji uzrokuju takve lažno pozitivne alarme. Uključivanje takvih zapisa u proces ponovnog strojnog učenja značajno doprinosi smanjenju broja lažno pozitivnih otkrivanja.

Uz pravilno otkrivanje sigurnosnih incidenata, korporativni NIDS pruža i opis otkrivene anomalije. U slučaju provedenog procesa verifikacije kibernetički napadi opisani su kao:

- *High Traffic*: Prosječna stopa prometa na računalu tijekom pet minuta premašila je ograničenje prihvatljivih vrijednosti prometa - za kibernetički napad tipa 1.
- *Port Scan*: Izvor se IP-adrese pokušao povezati s prekomjernim brojem vrata na ciljnoj IP-adresi – za kibernetički napad tipa 2.
- *Brute Force Login* - 21: Računalo otkriva niz kratkih TCP-veza koje se podudaraju s pokušajem grubog probijanja lozinke kroz ponavljane prijave – za kibernetički napad tipa 3.

Ti opisi mogu pružiti detaljnije smjernice za moguće strategije obrane od budućih sigurnosnih incidenata te su korisni za nadogradnju NFMIDS-modela. U budućim etapama učenja, model će trebati koristiti označene skupove podataka koji detaljno opisuju svaki kibernetički napad u ulaznim zapisima kako bi se osigurala preciznost procesa strojnog učenja. Nadalje, preporučuje se proširenje predloženih algoritama pseudokoda, tj. Algoritma 2 i Algoritma 3.

Alati i metodologije potrebni za standardizirano testiranje sustava za otkrivanje upada još uvijek nisu dovoljno razvijeni u javno dostupnom prostoru kako bi omogućili potrebne usporedne testove i evaluacije novih algoritama. Postoji značajna potreba za otvorenim okruženjem koje bi omogućilo generiranje prometa i umetanje napada na način koji se može uniformno koristiti za evaluaciju otkrivanja upada. Većina prijedloga za verifikaciju IDS-sustava navedenih u literaturi ostaje na razini koncepta i često nisu implementirani u stvarnim okruženjima. Nasuprot tomu, NFMIDS predstavljen u ovoj disertaciji, implementiran je u stvarnoj korporativnoj mreži i dokazao se kao učinkovito rješenje za klasifikaciju prometa te sposoban za otkrivanje anomalija mrežnog prometa u stvarnom vremenu.

6. ZAKLJUČAK

U okviru doktorske disertacije razmatran je model za otkrivanje napada u mreži temeljen na strojnom učenju te se istražila njegova učinkovitost u zaštiti mreža od kibernetičkih prijetnji. Proces izrade modela klasifikacije mrežnog prometa za otkrivanje anomalija uključivao je prikupljanje podataka o toku prometa, ispitivanje algoritama strojnog učenja na javno dostupnim skupovima podataka te predobradu podataka s ciljem optimizacije performansi modela. Korištena su tri cjelovita skupa podataka (UNSW-NB15, CSE-CIC-IDS2018, LUFlow2021) i pet prilagođenih NetFlow-skupova podataka. Analiza utjecaja različitih postupaka predobrade identificirala je postupke za optimizaciju ulaznih podataka. Postupci predobrade podataka uključivali su isključivanje nepotrebnih značajki, čišćenje i uklanjanje neispravnih vrijednosti, kodiranje kategorijskih značajki, odabir omjera podataka za učenje i testiranje te skaliranje podataka kako bi poboljšali performanse modela strojnog učenja.

Ispitivanjem četiriju često korištenih klasifikatora (slučajna šuma, stroj s potpornim vektorima, naivni Bayes i K-najbliži susjed) na javno dostupnim skupovima podataka, ukazalo je na najveću učinkovitost klasifikatora slučajne šume, postižući AUC-točnost od 99,98 % na skupu podataka LUFlow2021. Optimizacija hiperparametara za klasifikator slučajne šume provedena je kroz unakrsnu validaciju funkcijom GridSearchCV, postižući najbolje rezultate AUC-točnosti od 99,99 % s odabranim parametrima. Redukcijom i odabirom najrelevantnijih značajki pokazano je da osam odabranih značajki rezultira jednakom točnosti klasifikacije kao kada se koristi cijeli skup od jedanaest značajki na referentnom skupu podataka LUFlow2021. Predloženi NFMIDS-model, temeljen na optimiziranom klasifikatoru slučajne šume i referentnom skupu podataka LUFlow2021 sa smanjenim brojem značajki, pokazuje visoku točnost klasifikacije. Integracija tog modela u tradicionalno mrežno okruženje nadograđeno programski definiranim mrežnim uređajima omogućava brzu reakciju na otkrivene prijetnje. Predloženi je model testiran s prikupljenim stvarnim mrežnim prometom, a rezultati potvrđuju njegovu uspješnost s visokim vrijednostima AUC-točnosti i F2-točnosti od 98,09 % i 99,13 %.

Kibernetički napadi, uključujući DoS, otkrivanje korisničkih podataka i skeniranje portova, bili su dio procesa verifikacije. Svi su napadi uspješno identificirani, a vremena otkrivanja su se kretala od nekoliko sekundi do minute. Razlike u vremenima otkrivanja mogu se objasniti

karakteristikama mrežnog uređaja te dizajnom predloženog modela za otkrivanje anomalija. Implementacijom predloženog NFMIDS-modela u hibridnoj programski definiranoj mreži postignuta je visoka učinkovitost u otkrivanju i zaustavljanju kibernetičkih napada. Integracijom IPS-funkcionalnosti omogućilo se uspješno zaustavljanje napada, čime su istaknute prednosti implementacije modela u hibridnoj programski definiranoj mreži..

Implementacija predloženog modela za otkrivanje anomalija u stvarnom mrežnom prometu kompleksan je proces koji uključuje temeljito testiranje, verifikaciju i prilagodbu. Završna etapa ovog istraživanja usmjerena je na verifikaciju i sposobnost otkrivanja anomalija analizom NetFlow-zapisa s trima preklopnicima između napadača i žrtve, dodatno uspoređujući rezultate s postojećim korporativnim NIDS-om. Predložena je verifikacija višestrukim NetFlow-zapisima koja je obuhvatila tri vrste kibernetičkih napada: DoS, pokušaje otkrivanja korisničkih imena i lozinki te skeniranje portova na udaljenim uređajima. NetFlow-zapisi korišteni u istraživanju bili su definirani za ulazni i izlazni promet (NetFlow1 i NetFlow3) te samo za ulazni promet (NetFlow2). Bitna razlika u konfiguraciji NetFlow-zapisa proizlazi iz različitih mogućnosti tradicionalnih i SDN-preklopnika. U procesu verifikacije, IPS funkcionalnost NFMIDS-modela nije bila aktivna kako bi se izbjeglo prekidanje procesa. Metoda predložene verifikacije uključivala je testiranje modela s dvama scenarijima: jedan bez prisustva pozadinskog prometa (samo preklopnik SW1) i drugi s kombiniranim stvarnim pozadinskim prometom (NetFlow-zapisi iz SDN-preklopnika ili preklopnika SW2). NFMIDS-model bio je postavljen u pasivnom načinu rada kako bi minimalno utjecao na normalnu propusnost korporativne mreže tijekom verifikacije.

Ostvareni znanstveni doprinosi ove disertacije su sljedeći:

1. Postupak odabira i prilagodbe skupa ulaznih podataka za algoritam klasifikacije u svrhu otkrivanja anomalija u hibridnoj programski definiranoj mreži.
2. Model za otkrivanje anomalija mrežnog prometa u hibridnoj programski definiranoj mreži primjenom algoritma za klasifikaciju strojnim učenjem.
3. Metoda verifikacije predloženog modela u stvarnom okruženju.

Znanstveni doprinos disertacije obuhvaća postupak odabira i prilagodbe skupa ulaznih podataka s ciljem optimizacije algoritma klasifikacije za otkrivanje anomalija u hibridnoj programski definiranoj mreži. Tim se putem omogućava sustavno i učinkovito upravljanje ulaznim podatcima kako bi se poboljšale performanse klasifikacije, posebno u kontekstu hibridnog

programski definiranog okruženja. Kroz integraciju algoritama klasifikacije s posebnim naglaskom na strojnom učenju, istraživački doprinos temelji se na razvoju modela za otkrivanje anomalija u hibridnoj programski definiranoj mreži. Predloženi model postavlja temelj za daljnja istraživanja u području otkrivanja anomalija u hibridnim programski definiranim mrežama. Doprinos ovog istraživanja ogleda se i u razvoju i primjeni metode verifikacije koja se temelji na analizi očekivanih i postignutih rezultata otkrivanja anomalija, vremena trajanja kibernetičkih napada te stanja sučelja SDN-preklopnika pomoću višestrukih NetFlow-zapisa u stvarnom mrežnom okruženju.

NFMIDS-model otkrivanja anomalija, predstavljen u ovoj disertaciji, nije samo teorijski koncept, već je implementiran i testiran u stvarnoj korporativnoj mreži. Rezultati su potvrdili izvrsnu klasifikaciju mrežnog prometa i sposobnost otkrivanja anomalija u stvarnom vremenu. Ta praktična implementacija čini predloženi model relevantnim i učinkovitim za stvarne mrežne okoline, što je ključno za učinkovito otkrivanje i obranu od kibernetičkih prijetnji.

Buduća istraživanja usmjeravat će se na proučavanje nenadziranih algoritama strojnog učenja, usporedbu njihovih performansi s nadziranim algoritmima korištenim u ovoj disertaciji te analizu trajanja potrebnog za učenje i klasifikaciju. Cilj je bolje razumjeti prednosti i nedostatke nenadziranih metoda u odnosu na nadzirane, istražujući ponašanje različitih algoritama u različitim scenarijima mrežne sigurnosti. To će pridonijeti boljem odabiru algoritama za buduća istraživanja o otkrivanju i zaštiti od kibernetičkih prijetnji. Dodatne mogućnosti poboljšanja otkrivanja anomalija predloženog modela uključuju prilagodbu postupka onemogućavanja sučelja mrežnog uređaja uz uzimanje u obzir vremenskih okvira i zadržki kako bi se smanjile posljedice lažno pozitivnih klasifikacija. Umjesto trenutnog isključivanja sučelja pri otkrivanju anomalija, moguća je nadogradnja sustava koja uključuje dodatne provjere prije donošenja konačne odluke, uključujući praćenje kontinuiranosti anomalija kroz intervale. Takav bi pristup omogućio sustavu bolje razlikovanje stvarnih prijetnji od privremenih fluktuacija ili nepravilnosti u mrežnom prometu. Buduća istraživanja će također razmotriti podršku za IPFIX-protokol radi kompatibilnosti s otvorenim rješenjima za programski definirane mreže, poput OpenDaylight-kontrolera i OpenFlow-protokola za integraciju algoritama strojnog učenja izravno u SDN-kontrolere. Nadalje, buduće istraživanje će se usmjeriti na razvoj novih metoda i tehnika za

optimizaciju algoritama strojnog učenja u SDN-okruženju kako bi se osigurala učinkovita obrada velike količine mrežnog prometa u stvarnom vremenu.

Kontinuirano poboljšanje i testiranje NFMIDS-modela predstavljaju ključnu komponentu njegove održivosti i učinkovitosti, s obzirom na širok spektar kibernetičkih prijetnji koje se konstantno pojavljuju. Buduća unaprjeđenja, uključujući i proširenje postojeće IPS-funkcionalnosti, donose dodatne prednosti u stvaranju snažnijeg i otpornijeg sustava za zaštitu mreža od potencijalnih prijetnji. Kroz ovaj proces unaprjeđenja, mogu se identificirati potencijalne slabosti i izazovi te razviti nove strategije za njihovo rješavanje.

LITERATURA

- [1] N. al Khater and R. E. Overill, "Network traffic classification techniques and challenges," in *2015 Tenth International Conference on Digital Information Management (ICDIM)*, IEEE, Oct. 2015, pp. 43–48. doi: 10.1109/ICDIM.2015.7381869.
- [2] A. Malik, R. de Frein, M. Al-Zeyadi, and J. Andreu-Perez, "Intelligent SDN Traffic Classification Using Deep Learning: Deep-SDN," in *2020 2nd International Conference on Computer Communication and the Internet (ICCCI)*, IEEE, Jun. 2020, pp. 184–189. doi: 10.1109/ICCCI49374.2020.9145971.
- [3] N. Ahmed *et al.*, "Network Threat Detection Using Machine/Deep Learning in SDN-Based Platforms: A Comprehensive Analysis of State-of-the-Art Solutions, Discussion, Challenges, and Future Research Direction," *Sensors*, vol. 22, no. 20, p. 7896, Oct. 2022, doi: 10.3390/s22207896.
- [4] U. Ghosh, P. Chatterjee, S. S. Shetty, C. Kamhoua, and L. Njilla, "Towards Secure Software-Defined Networking Integrated Cyber-Physical Systems: Attacks and Countermeasures," *Cybersecurity and Privacy in Cyber-Physical Systems*, no. February, pp. 103–132, 2019, doi: 10.1201/9780429263897-6.
- [5] F. N. Khan, Y. Saleem, and M. K. Bashir, "Migration of Multiplatform Legacy Network to Single Software-Defined-Network (SDN)," *Proceedings - 2018 International Conference on Computing, Electronics and Communications Engineering, iCCECE 2018*, pp. 333–338, 2019, doi: 10.1109/iCCECOME.2018.8658652.
- [6] C. Lorenz *et al.*, "An SDN/NFV-Enabled Enterprise Network Architecture Offering Fine-Grained Security Policy Enforcement," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 217–223, 2017, doi: 10.1109/MCOM.2017.1600414CM.
- [7] J. H. Cox *et al.*, "Advancing software-defined networks: A survey," *IEEE Access*, vol. 5, pp. 25487–25526, 2017, doi: 10.1109/ACCESS.2017.2762291.
- [8] V. H. Dixit, S. Kyung, Z. Zhao, A. Doupé, Y. Shoshitaishvili, and G.-J. Ahn, "Challenges and Preparedness of SDN-based Firewalls," in *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, New York, NY, USA: ACM, Mar. 2018, pp. 33–38. doi: 10.1145/3180465.3180468.
- [9] R. Amin, M. Reisslein, and N. Shah, "Hybrid SDN networks: A survey of existing approaches," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 4, pp. 3259–3306, 2018. doi: 10.1109/COMST.2018.2837161.
- [10] M. Canini, A. Feldmann, D. Levin, F. Schaffert, and S. Schmid, "Software-defined networks: Incremental deployment with panopticon," *Computer (Long Beach Calif)*, vol. 47, no. 11, pp. 56–60, 2014, doi: 10.1109/MC.2014.330.

- [11] K. Humayun, "Software Defined Networking (SDN): A Revolution in Computer Network," *IOSR J Comput Eng*, vol. 15, no. 5, pp. 103–106, 2013, doi: 10.9790/0661-155103106.
- [12] I. Alsmadi and D. Xu, "Security of Software Defined Networks: A survey," *Comput Secur*, vol. 53, pp. 79–108, 2015, doi: 10.1016/j.cose.2015.05.006.
- [13] "Software-Defined Networking (SDN) Definition." [Online]. Available: <https://www.opennetworking.org/sdn-definition/>
- [14] "APIC Enterprise Module API Overview." Accessed: Jan. 26, 2023. [Online]. Available: <https://developer.cisco.com/docs/apic-em/#!/overview/cisco-apic-em-northbound-interface>
- [15] O. N. F. Solution, B. September, and ONF, "SDN in the Campus Environment," *ONF workshop*, 2013.
- [16] X. Huang, S. Cheng, K. Cao, P. Cong, T. Wei, and S. Hu, "A Survey of Deployment Solutions and Optimization Strategies for Hybrid SDN Networks," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1–25, 2018.
- [17] M. Nkosi, A. Lysko, L. Ravhuanzwo, T. Nandeni, and A. Engelberench, "Classification of SDN distributed controller approaches: A brief overview," *Proceedings - 2016 3rd International Conference on Advances in Computing, Communication and Engineering, ICACCE 2016*, pp. 342–344, 2017, doi: 10.1109/ICACCE.2016.8073772.
- [18] P. Vijay Tijare and D. Vasudevan, "THE NORTHBOUND APIs OF SOFTWARE DEFINED NETWORKS," © *International Journal of Engineering Sciences & Research Technology*, vol. 501, no. January 2019, 2016, doi: 10.5281/zenodo.160891.
- [19] S. Bailey *et al.*, "SDN Architecture Overview," *EPC and 4G Packet Networks*, pp. 17–64, 2013, doi: 10.1016/b978-0-12-394595-2.00002-5.
- [20] J. N. Binlun, T. S. Chin, L. C. Kwang, Z. Yusoff, and R. Kaspin, "Challenges and Direction of Hybrid SDN Migration in ISP networks," *2018 IEEE International Conference on Electronics and Communication Engineering, ICECE 2018*, pp. 60–64, 2019, doi: 10.1109/ICECOME.2018.8644812.
- [21] S. Vissicchio, L. Vanbever, and O. Bonaventure, "Opportunities and research challenges of hybrid software defined networks," *Computer Communication Review*, vol. 44, no. 2, pp. 70–75, 2014, doi: 10.1145/2602204.2602216.
- [22] S. Vissicchio, L. Vanbever, L. Cittadini, G. G. Xie, and O. Bonaventure, "Safe update of hybrid SDN networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1649–1662, 2017, doi: 10.1109/TNET.2016.2642586.
- [23] J. Galán-Jiménez, "Exploiting the control power of SDN during the transition from IP to SDN networks," *International Journal of Communication Systems*, vol. 31, no. 5, pp. 1–19, 2018, doi: 10.1002/dac.3504.

- [24] W. Wang, W. He, and J. Su, "Boosting the Benefits of Hybrid SDN," *Proc Int Conf Distrib Comput Syst*, pp. 2165–2170, 2017, doi: 10.1109/ICDCS.2017.302.
- [25] Sandhya, Y. Sinha, and K. Haribabu, "A survey: Hybrid SDN," *Journal of Network and Computer Applications*, vol. 100, no. March, pp. 35–55, 2017, doi: 10.1016/j.jnca.2017.10.003.
- [26] L. F. Carvalho, T. Abrão, L. de S. Mendes, and M. L. Proença, "An ecosystem for anomaly detection and mitigation in software-defined networking," *Expert Syst Appl*, vol. 104, pp. 121–133, Aug. 2018, doi: 10.1016/j.eswa.2018.03.027.
- [27] P. Radoglou-Grammatikis *et al.*, "Modeling, Detecting, and Mitigating Threats against Industrial Healthcare Systems: A Combined Software Defined Networking and Reinforcement Learning Approach," *IEEE Trans Industr Inform*, vol. 18, no. 3, 2022, doi: 10.1109/TII.2021.3093905.
- [28] H. A. Alamri, V. Thayananthan, and J. Yazdani, "Machine Learning for Securing SDN based 5G Network," *Int J Comput Appl*, vol. 174, no. 14, 2021, doi: 10.5120/ijca2021921027.
- [29] L. Zhang, G. Shou, Y. Hu, and Z. Guo, "Deployment of Intrusion Prevention System based on Software Defined Networking," *International Conference on Communication Technology Proceedings, ICCT*, pp. 26–31, 2013, doi: 10.1109/ICCT.2013.6820345.
- [30] H. Wang and B. Wu, "SDN-based hybrid honeypot for attack capture," *Proceedings of 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2019*, no. Itnec, pp. 1602–1606, 2019, doi: 10.1109/ITNEC.2019.8729425.
- [31] W. Fan and D. Fernandez, "A novel SDN based stealthy TCP connection handover mechanism for hybrid honeypot systems," *2017 IEEE Conference on Network Softwarization: Softwarization Sustaining a Hyper-Connected World: en Route to 5G, NetSoft 2017*, 2017, doi: 10.1109/NETSOFT.2017.8004194.
- [32] J. Chukwu, O. Osamudiamen, and A. Matrawy, "IDSaaS in SDN: Intrusion Detection System as a service in software defined networks," *2016 IEEE Conference on Communications and Network Security, CNS 2016*, pp. 356–357, 2017, doi: 10.1109/CNS.2016.7860509.
- [33] M. Monshizadeh, V. Khatri, and R. Kantola, "Detection as a service: An SDN application," *International Conference on Advanced Communication Technology, ICACT*, pp. 285–290, 2017, doi: 10.23919/ICACT.2017.7890099.
- [34] J. E. Varghese and B. Muniyal, "An Efficient IDS Framework for DDoS Attacks in SDN Environment," *IEEE Access*, vol. 9, pp. 69680–69699, 2021, doi: 10.1109/ACCESS.2021.3078065.

- [35] M. Al Razib, D. Javeed, M. T. Khan, R. Alkanhel, and M. S. A. Muthanna, "Cyber Threats Detection in Smart Environments Using SDN-Enabled DNN-LSTM Hybrid Framework," *IEEE Access*, vol. 10, pp. 53015–53026, 2022, doi: 10.1109/ACCESS.2022.3172304.
- [36] M. Ammar, M. Rizk, A. Abdel-Hamid, and A. K. Aboul-Seoud, "A framework for security enhancement in SDN-based datacenters," *2016 8th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2016*, pp. 3–6, 2016, doi: 10.1109/NTMS.2016.7792427.
- [37] C. Birkinshaw, E. Rouka, and V. G. Vassilakis, "Implementing an intrusion detection and prevention system using software-defined networking: Defending against port-scanning and denial-of-service attacks," *Journal of Network and Computer Applications*, vol. 136, no. February, pp. 71–85, 2019, doi: 10.1016/j.jnca.2019.03.005.
- [38] C. V. Neu, C. G. Tatsch, R. C. Lunardi, R. A. Michelin, A. M. S. Orozco, and A. F. Zorzo, "Lightweight IPS for port scan in OpenFlow SDN networks," *IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, NOMS 2018*, pp. 1–6, 2018, doi: 10.1109/NOMS.2018.8406313.
- [39] R. F. Pratama, N. A. Suwastika, and M. A. Nugroho, "Design and implementation adaptive Intrusion Prevention System (IPS) for attack prevention in software-defined network (SDN) architecture," *2018 6th International Conference on Information and Communication Technology, ICoICT 2018*, vol. 0, no. c, pp. 299–304, 2018, doi: 10.1109/ICoICT.2018.8528735.
- [40] J. Xie *et al.*, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 1. Institute of Electrical and Electronics Engineers Inc., pp. 393–430, Jan. 01, 2019. doi: 10.1109/COMST.2018.2866942.
- [41] P. K. Mondal, L. P. Aguirre Sanchez, E. Benedetto, Y. Shen, and M. Guo, "A dynamic network traffic classifier using supervised ML for a Docker-based SDN network," *Conn Sci*, vol. 33, no. 3, pp. 693–718, Jul. 2021, doi: 10.1080/09540091.2020.1870437.
- [42] D. Jankowski and M. Amanowicz, "A study on flow features selection for malicious activities detection in software defined networks," *2018 International Conference on Military Communications and Information Systems, ICMCIS 2018*, pp. 1–9, 2018, doi: 10.1109/ICMCIS.2018.8398697.
- [43] N. Mazhar, R. Salleh, M. Zeeshan, M. M. Hameed, and N. Khan, "R-IDPS: Real time SDN based IDPS system for IoT security," in *HONET 2021 - IEEE 18th International Conference on Smart Communities: Improving Quality of Life using ICT, IoT and AI*, 2021. doi: 10.1109/HONET53078.2021.9615449.
- [44] K. S. Sahoo *et al.*, "An Evolutionary SVM Model for DDOS Attack Detection in Software Defined Networks," *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.3009733.

- [45] M. A. Mohsin and A. H. Hamad, "Performance Evaluation of SDN DDoS Attack Detection and Mitigation Based Random Forest and K-Nearest Neighbors Machine Learning Algorithms," *Revue d'Intelligence Artificielle*, vol. 36, no. 2, pp. 233–240, Apr. 2022, doi: 10.18280/ria.360207.
- [46] S. Wang *et al.*, "Detecting flooding DDoS attacks in software defined networks using supervised learning techniques," *Engineering Science and Technology, an International Journal*, vol. 35, p. 101176, Nov. 2022, doi: 10.1016/j.jestch.2022.101176.
- [47] N. M. Yungaicela-Naula, C. Vargas-Rosales, and J. A. Perez-Diaz, "SDN-Based Architecture for Transport and Application Layer DDoS Attack Detection by Using Machine and Deep Learning," *IEEE Access*, vol. 9, pp. 108495–108512, 2021, doi: 10.1109/ACCESS.2021.3101650.
- [48] P. Krishnan, K. Jain, A. Aldweesh, P. Prabu, and R. Buyya, "OpenStackDP: a scalable network security framework for SDN-based OpenStack cloud infrastructure," *Journal of Cloud Computing*, vol. 12, no. 1, p. 26, Feb. 2023, doi: 10.1186/s13677-023-00406-w.
- [49] M. Maray *et al.*, "Optimal Deep Learning Driven Intrusion Detection in SDN-Enabled IoT Environment," *Computers, Materials & Continua*, vol. 74, no. 3, pp. 6587–6604, 2023, doi: 10.32604/cmc.2023.034176.
- [50] H.-M. Chuang, F. Liu, and C.-H. Tsai, "Early Detection of Abnormal Attacks in Software-Defined Networking Using Machine Learning Approaches," *Symmetry (Basel)*, vol. 14, no. 6, p. 1178, Jun. 2022, doi: 10.3390/sym14061178.
- [51] S. R. A. Kanavalli, A. Gupta, A. Pattanaik, and S. Agarwal, "Real-time DDoS Detection and Mitigation in Software Defined Networks using Machine Learning Techniques," *International Journal of Computing*, pp. 353–359, Sep. 2022, doi: 10.47839/ijc.21.3.2691.
- [52] Javaid Nabi, "Machine Learning —Fundamentals." Accessed: Mar. 07, 2022. [Online]. Available: <https://towardsdatascience.com/machine-learning-basics-part-1-a36d38c7916>
- [53] H. Singh, *Practical Machine Learning with AWS*. 2021. doi: 10.1007/978-1-4842-6222-1.
- [54] S. Chibani and F.-X. Coudert, "Machine learning approaches for the prediction of materials properties," *APL Mater*, vol. 8, no. 8, Aug. 2020, doi: 10.1063/5.0018384.
- [55] "Machine Learning Classification Algorithm." [Online]. Available: <https://www.javatpoint.com/machine-learning>
- [56] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach Learn*, vol. 109, no. 2, pp. 373–440, 2020, doi: 10.1007/s10994-019-05855-6.
- [57] P. C. Sen, M. Hajra, and M. Ghosh, *Supervised Classification Algorithms in Machine Learning: A Survey and Review*, vol. 937. Springer Singapore, 2020. doi: 10.1007/978-981-13-7403-6_11.

- [58] G. Louppe, “Understanding Random Forests: From Theory to Practice,” Jul. 2014, [Online]. Available: <http://arxiv.org/abs/1407.7502>
- [59] T. Zhang, *Mathematical Analysis of Machine Learning Algorithms*. Cambridge University Press, 2023. doi: 10.1017/9781009093057.
- [60] O. F.Y, A. J.E.T, A. O, H. J. O, O. O, and A. J, “Supervised Machine Learning Algorithms: Classification and Comparison,” *International Journal of Computer Trends and Technology*, vol. 48, no. 3, pp. 128–138, Jun. 2017, doi: 10.14445/22312803/IJCTT-V48P126.
- [61] “Decision Trees.” [Online]. Available: <https://scikit-learn.org/stable/modules/tree.html>
- [62] Farrukh Nizam Arain, “Decision Tree Classification Algorithm.” Accessed: Mar. 09, 2022. [Online]. Available: <https://www.devops.ae/decision-tree-classification-algorithm/>
- [63] J. Pustejovsky and A. Stubbs, *Natural Language Annotation for Machine Learning -- A guide to Corpus-building for applications*. 2013.
- [64] Md. A. M. Hasan, M. Nasser, B. Pal, and S. Ahmad, “Support Vector Machine and Random Forest Modeling for Intrusion Detection System (IDS),” *Journal of Intelligent Learning Systems and Applications*, vol. 06, no. 01, pp. 45–52, 2014, doi: 10.4236/jilsa.2014.61005.
- [65] “Random Forest Algorithm.” Accessed: Aug. 01, 2023. [Online]. Available: <https://www.javatpoint.com/machine-learning-random-forest-algorithm>
- [66] M. A. Umar and C. Zhanfang, “Effects of Feature Selection and Normalization on Network Intrusion Detection,” pp. 1–25, 2020, doi: 10.36227/techrxiv.12480425.
- [67] L. Breiman, “Consistency for a simple model of random forests,” 2004.
- [68] L. Breiman, “Random Forests,” *Mach Learn*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
- [69] L. Breiman, “Some infinity theory for predictor ensembles,” 2000.
- [70] L. Breiman, “Bagging predictors,” *Mach Learn*, vol. 24, no. 2, pp. 123–140, Aug. 1996, doi: 10.1007/BF00058655.
- [71] Y. Amit and D. Geman, “Shape Quantization and Recognition with Randomized Trees,” *Neural Comput*, vol. 9, no. 7, pp. 1545–1588, Oct. 1997, doi: 10.1162/neco.1997.9.7.1545.
- [72] Tin Kam Ho, “The random subspace method for constructing decision forests,” *IEEE Trans Pattern Anal Mach Intell*, vol. 20, no. 8, pp. 832–844, 1998, doi: 10.1109/34.709601.

- [73] T. G. Dietterich, “An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization,” *Mach Learn*, vol. 40, no. 2, pp. 139–157, 2000, doi: 10.1023/A:1007607513941.
- [74] L. Breiman, “Random Forests,” *Mach Learn*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
- [75] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification And Regression Trees*. Routledge, 2017. doi: 10.1201/9781315139470.
- [76] G. Biau, “Analysis of a Random Forests Model,” May 2010, [Online]. Available: <http://arxiv.org/abs/1005.0208>
- [77] A. Cano, “A survey on graphic processing unit computing for large-scale data mining,” *Wiley Interdiscip Rev Data Min Knowl Discov*, vol. 8, no. 1, p. e1232, Jan. 2018, doi: 10.1002/widm.1232.
- [78] A. Mammone, M. Turchi, and N. Cristianini, “Support vector machines,” *Comput Stat*, pp. 283–289, 2009, Accessed: Dec. 06, 2023. [Online]. Available: https://d1wqtxts1xzle7.cloudfront.net/45694827/wics.4920160517-26685-10le7kw-libre.pdf?1463470793=&response-content-disposition=inline%3B+filename%3DSupport_vector_machines.pdf&Expires=1701863797&Signature=dcFT8ynf9VneoH7yk9pF~h~lZppHvm1v0TXvVYfdhEJs6G~mLmhkHiVN0c-ffsOLq8-wXwO5KHaAuFxr8APy-ueUqJ2ZAJH~jfdXNtw-aLUMmbf-WysoWNliap6JMb8UzqJ1NzNDMrBvFftmkfVTUEI5I~TJ0yjdIgVsdLf4BsQpO13XN~oIjNAmQehiEyBQFfih2chVNVb1ra~fGZWAI8Fm2crLyMSNz3508bmlfqhap5KjpIWpncbScSiJ5PB4ATKo9IPo4pSzuy82DE66GBo~qY6H5tzfEaJ2CibmI7i24wITm3xp2NJbXkJTg-1OfrvxoOfmWXX8X-Zef-8Q__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA
- [79] D. Nelson, “Overview of Classification Methods in Python with Scikit-Learn.” [Online]. Available: <https://stackabuse.com/overview-of-classification-methods-in-python-with-scikit-learn/>
- [80] X. Wu *et al.*, *Top 10 algorithms in data mining*, vol. 14, no. 1. 2008. doi: 10.1007/s10115-007-0114-2.
- [81] “Support Vector Machines.” [Online]. Available: <https://scikit-learn.org/stable/modules/svm.html>
- [82] B. Vivek, “Naive Bayes algorithm.” Accessed: Dec. 06, 2023. [Online]. Available: <https://www.topcoder.com/thrive/articles/naive-bayes-algorithm>
- [83] V. Metsis, I. Androustopoulos, and G. Paliouras, “Spam filtering with Naive Bayes - Which Naive Bayes?,” *3rd Conference on Email and Anti-Spam - Proceedings, CEAS 2006*, 2006.
- [84] K. Taunk, S. De, S. Verma, and A. Swetapadma, “A Brief Review of Nearest Neighbor Algorithm for Learning and Classification,” in *2019 International Conference on*

- Intelligent Computing and Control Systems (ICCS)*, IEEE, May 2019, pp. 1255–1260. doi: 10.1109/ICCS45141.2019.9065747.
- [85] L. Y. Hu, M. W. Huang, S. W. Ke, and C. F. Tsai, “The distance function effect on k-nearest neighbor classification for medical datasets,” *Springerplus*, vol. 5, no. 1, 2016, doi: 10.1186/s40064-016-2941-7.
- [86] P. Cunningham and S. J. Delany, “k-Nearest neighbour classifiers 2nd edition (with python examples),” *ArXiv*, no. 1, pp. 1–22, 2020.
- [87] K. Stapor, “Evaluating and Comparing Classifiers: Review, Some Recommendations and Limitations,” in *Advances in Intelligent Systems and Computing*, vol. 578, no. February, 2018, pp. 12–21. doi: 10.1007/978-3-319-59162-9_2.
- [88] M. Hossin and M. N. Sulaiman, “A Review on Evaluation Metrics for Data Classification Evaluations,” *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, pp. 01–11, Mar. 2015, doi: 10.5121/ijdkp.2015.5201.
- [89] “Metrics and scoring: quantifying the quality of predictions.” [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#
- [90] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, “The balanced accuracy and its posterior distribution,” *Proceedings - International Conference on Pattern Recognition*, pp. 3121–3124, 2010, doi: 10.1109/ICPR.2010.764.
- [91] J. Brownlee, *Imbalanced Classification with Python Better Metrics, Balance Skewed Classes, Cost-Sensitive Learning*, V1.3. Machine Learning Mastery, 2021. [Online]. Available: <https://books.google.hr/books?id=jaXJDwAAQBAJ&printsec=copyright#v=onepage&q&f=false>
- [92] S. Narkhede, “Understanding AUC - ROC Curve,” Towards Data Science. [Online]. Available: <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>
- [93] Y. Kim, K. A. Toh, A. B. J. Teoh, H. L. Eng, and W. Y. Yau, “An online AUC formulation for binary classification,” *Pattern Recognit*, vol. 45, no. 6, pp. 2266–2279, 2012, doi: 10.1016/j.patcog.2011.11.020.
- [94] J. L. Leevy and T. M. Khoshgoftaar, “A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data,” *J Big Data*, vol. 7, no. 1, Dec. 2020, doi: 10.1186/s40537-020-00382-x.
- [95] M. Buda, A. Maki, and M. A. Mazurowski, “A systematic study of the class imbalance problem in convolutional neural networks,” *Neural Networks*, vol. 106, pp. 249–259, Oct. 2018, doi: 10.1016/j.neunet.2018.07.011.
- [96] J. S. Lee, “AUC4.5: AUC-Based C4.5 Decision Tree Algorithm for Imbalanced Data Classification,” *IEEE Access*, vol. 7, pp. 106034–106042, 2019, doi: 10.1109/ACCESS.2019.2931865.

- [97] M. Widmann, “Cohen’s Kappa: What It Is, When to Use It, and How to Avoid Its Pitfalls.” [Online]. Available: <https://thenewstack.io/cohens-kappa-what-it-is-when-to-use-it-and-how-to-avoid-its-pitfalls/>
- [98] T. Janarthanan, “Feature Selection in UNSW-NB15 and KDDCUP ’99 datasets,” 2017.
- [99] “CSE-CIC-IDS2018 on AWS.” Accessed: Apr. 12, 2022. [Online]. Available: <https://www.unb.ca/cic/datasets/ids-2018.html>
- [100] “LUFLOW Network Intrusion Detection Data Set.” Accessed: Apr. 20, 2022. [Online]. Available: <https://www.kaggle.com/datasets/mryanm/luflow-network-intrusion-detection-data-set>
- [101] T.-H. Chua and I. Salam, “Evaluation of Machine Learning Algorithms in Network-Based Intrusion Detection System,” Mar. 2022, [Online]. Available: <http://arxiv.org/abs/2203.05232>
- [102] “NetFlow overview.” Accessed: Feb. 15, 2023. [Online]. Available: https://documentation.meraki.com/MX/Monitoring_and_Reporting/NetFlow_Overview
- [103] S. and M. N. and P. M. Sarhan Mohanad and Layeghy, “NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems,” in *Big Data Technologies and Applications*, H. and H. R. and R. S. and C. N. Deze Zeng and Huang, Ed., Cham: Springer International Publishing, 2021, pp. 117–135.
- [104] “Classification: ROC Curve and AUC.” Accessed: Oct. 05, 2022. [Online]. Available: <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc>
- [105] R. Sehrawat, “Data Preparation.”
- [106] “Data science report.” Accessed: Apr. 13, 2022. [Online]. Available: <https://visit.figure-eight.com/2015-data-scientist-report.html>
- [107] S. Ustebay, Z. Turgut, and M. A. Aydin, “Intrusion Detection System with Recursive Feature Elimination by Using Random Forest and Deep Learning Classifier,” in *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, IEEE, Dec. 2018, pp. 71–76. doi: 10.1109/IBIGDELFT.2018.8625318.
- [108] S. Ü. M. A. A. T. A. D. Aksu, “Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm,” in *International symposium on computer and information sciences*, Springer, 2018, pp. 141–149.
- [109] C. Song, Y. Park, K. Golani, Y. Kim, K. Bhatt, and K. Goswami, “Machine-learning based threat-aware system in software defined networks,” *2017 26th International Conference on Computer Communications and Networks, ICCCN 2017*, 2017, doi: 10.1109/ICCCN.2017.8038436.

- [110] S. Seth, G. Singh, and K. Kaur Chahal, "A novel time efficient learning-based approach for smart intrusion detection system," *J Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00498-8.
- [111] R. Atefinia and M. Ahmadi, "Network intrusion detection using multi-architectural modular deep neural network," *Journal of Supercomputing*, vol. 77, no. 4, pp. 3571–3593, Apr. 2021, doi: 10.1007/s11227-020-03410-y.
- [112] Cornellius Yudha Wijaya, "5 Feature Selection Method from Scikit-Learn you should know." Accessed: Jun. 03, 2022. [Online]. Available: <https://towardsdatascience.com/5-feature-selection-method-from-scikit-learn-you-should-know-ed4d116e4172>
- [113] Kishan Maladkar, "5 Ways To Handle Missing Values In Machine Learning Datasets." Accessed: Apr. 13, 2022. [Online]. Available: <https://analyticsindiamag.com/5-ways-handle-missing-values-machine-learning-datasets/>
- [114] R. B. Basnet, R. Shash, C. Johnson, L. Walgren, and T. Doleck, "Towards Detecting and Classifying Network Intrusion Traffic Using Deep Learning Frameworks." [Online]. Available: <https://rambasnet.github.io>
- [115] J. Gong and T. Chen, "Does Configuration Encoding Matter in Learning Software Performance? An Empirical Study on Encoding Schemes," Mar. 2022, doi: 10.1145/3524842.3528431.
- [116] C. Seger, "An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing," *Degree Project Technology*, p. 41, 2018, [Online]. Available: <http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-237426%0Ahttp://www.diva-portal.org/smash/get/diva2:1259073/FULLTEXT01.pdf>
- [117] K. K. Dobbin and R. M. Simon, "Optimally splitting cases for training and testing high dimensional classifiers," *BMC Med Genomics*, vol. 4, no. 1, p. 31, 2011, doi: 10.1186/1755-8794-4-31.
- [118] Y. Xu and R. Goodacre, "On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning," *J Anal Test*, vol. 2, no. 3, pp. 249–262, 2018, doi: 10.1007/s41664-018-0068-2.
- [119] W. Xu, Y. Fan, and C. Li, "I2DS: Interpretable Intrusion Detection System Using Autoencoder and Additive Tree," *Security and Communication Networks*, vol. 2021, no. 3, 2021, doi: 10.1155/2021/5564354.
- [120] M. Ahsan, R. Gomes, Md. M. Chowdhury, and K. E. Nygard, "Enhancing Machine Learning Prediction in Cybersecurity Using Dynamic Feature Selector," *Journal of Cybersecurity and Privacy*, vol. 1, no. 1, pp. 199–218, 2021, doi: 10.3390/jcp1010011.

- [121] G. Kocher and G. Kumar, "Performance Analysis of Machine Learning Classifiers for Intrusion Detection using UNSW-NB15 Dataset," pp. 31–40, 2020, doi: 10.5121/csit.2020.102004.
- [122] T. Ahmad, D. Truscan, J. Vain, and I. Porres, "Early Detection of Network Attacks Using Deep Learning," Jan. 2022, [Online]. Available: <http://arxiv.org/abs/2201.11628>
- [123] N. Elmrabit, F. Zhou, F. Li, and H. Zhou, "Evaluation of Machine Learning Algorithms for Anomaly Detection," in *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, IEEE, Jun. 2020, pp. 1–8. doi: 10.1109/CyberSecurity49315.2020.9138871.
- [124] J. Brownlee, "Train-Test Split for Evaluating Machine Learning Algorithms." [Online]. Available: <https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/>
- [125] M. Nawir, A. Amir, O. B. Lynn, N. Yaakob, and R. Badlishah Ahmad, "Performances of Machine Learning Algorithms for Binary Classification of Network Anomaly Detection System," *J Phys Conf Ser*, vol. 1018, no. 1, 2018, doi: 10.1088/1742-6596/1018/1/012015.
- [126] "sklearn.preprocessing.StandardScaler." Accessed: Jun. 09, 2022. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler>
- [127] "sklearn.preprocessing.RobustScaler." Accessed: Jun. 09, 2022. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html#sklearn.preprocessing.RobustScaler>
- [128] "sklearn.preprocessing.MinMaxScaler." Accessed: Jun. 09, 2022. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn.preprocessing.MinMaxScaler>
- [129] P. Dahiya and D. K. Srivastava, "Network Intrusion Detection in Big Dataset Using Spark," *Procedia Comput Sci*, vol. 132, pp. 253–262, 2018, doi: 10.1016/j.procs.2018.05.169.
- [130] R. Mills, A. K. Marnerides, M. Broadbent, and N. Race, "Citrus Practical Intrusion Detection of Emerging Threats," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 582–600, Mar. 2022, doi: 10.1109/TNSM.2021.3091517.
- [131] S. Zwane, P. Tarwireyi, and M. Adigun, "Performance Analysis of Machine Learning Classifiers for Intrusion Detection," in *2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)*, IEEE, Dec. 2018, pp. 1–5. doi: 10.1109/ICONIC.2018.8601203.

- [132] R. Kumar and R. Singh, "Netflow based cyber threat classification using J48 and random forest machine learning algorithms," *Int J Eng Adv Technol*, vol. 9, no. 1, pp. 2973–2979, Oct. 2019, doi: 10.35940/ijeat.A1326.109119.
- [133] C. A. Ramezan, T. A. Warner, and A. E. Maxwell, "Evaluation of sampling and cross-validation tuning strategies for regional-scale machine learning classification," *Remote Sens (Basel)*, vol. 11, no. 2, Jan. 2019, doi: 10.3390/rs11020185.
- [134] "Cross-validation: evaluating estimator performance." Accessed: Mar. 16, 2022. [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html
- [135] D. J. V. Lopes, G. W. Burgreen, and E. D. Entsminger, "North American hardwoods identification using machine-learning," *Forests*, vol. 11, no. 3, Mar. 2020, doi: 10.3390/f11030298.
- [136] N. Darapureddy, N. Karatapu, and T. K. Battula, "Research of machine learning algorithms using k-fold cross validation," *Int J Eng Adv Technol*, vol. 8, no. 6 Special issue, 2019, doi: 10.35940/ijeat.F1043.0886S19.
- [137] "Cross Validation." Accessed: Feb. 13, 2024. [Online]. Available: <https://medium.com/@ompramod9921/cross-validation-623620ff84c2>
- [138] P. Liashchynskyi and P. Liashchynskyi, "Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS," Dec. 2019, [Online]. Available: <http://arxiv.org/abs/1912.06059>
- [139] "sklearn.model_selection.GridSearchCV." Accessed: Mar. 07, 2022. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
- [140] "What is network flow data?" Accessed: Jun. 29, 2022. [Online]. Available: <https://tools.netsa.cert.org/silk/faq.html#what-is-flow>
- [141] "What is NetFlow?" Accessed: Jun. 28, 2022. [Online]. Available: <https://www.manageengine.com/products/netflow/what-is-netflow.html>
- [142] "The Evolution of Network Flow Monitoring, from NetFlow to IPFIX." Accessed: Jun. 28, 2022. [Online]. Available: <https://www.noction.com/blog/network-flow-monitoring>
- [143] "NetFlow probe using Raspberry," Pandora FMS team. [Online]. Available: <https://pandorafms.com/blog/netflow-probe-using-raspberry/>
- [144] Steve Petryschuk, "NetFlow Basics: An Introduction to Monitoring Network Traffic." Accessed: Feb. 17, 2023. [Online]. Available: <https://www.auvik.com/franklyit/blog/netflow-basics/>
- [145] "Network Management Configuration Guide, Cisco IOS XE Everest 16.6.x (Catalyst 9300 Switches)." Accessed: Jun. 28, 2022. [Online]. Available: <https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst9300/software/release/16->

6/configuration_guide/nmngmt/b_166_nmngmt_9300_cg/b_166_nmngmt_9300_cg_chapter_0111.html

- [146] Cornelli Yudha Wijaya, “5 Feature Selection Method from Scikit-Learn you should know.” Accessed: Jun. 29, 2022. [Online]. Available: <https://towardsdatascience.com/5-feature-selection-method-from-scikit-learn-you-should-know-ed4d116e4172>
- [147] M. Bahrololum, E. Salahi, and M. Khaleghi, “Machine Learning Techniques for Feature Reduction in Intrusion Detection Systems: A Comparison,” in *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*, IEEE, 2009, pp. 1091–1095. doi: 10.1109/ICCIT.2009.89.
- [148] B. Xue, M. Zhang, W. N. Browne, and X. Yao, “A Survey on Evolutionary Computation Approaches to Feature Selection,” *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, Aug. 2016, doi: 10.1109/TEVC.2015.2504420.
- [149] O. I. Sheluhin and V. P. Ivannikova, “COMPARATIVE ANALYSIS OF INFORMATIVE FEATURES QUANTITY AND COMPOSITION SELECTION METHODS FOR THE COMPUTER ATTACKS CLASSIFICATION USING THE UNSW-NB15 DATASET,” *T-Comm*, vol. 14, no. 10, pp. 53–60, 2020, doi: 10.36724/2072-8735-2020-14-10-53-60.
- [150] A. Pasyuk, E. Semenov, and D. Tyuhtyaev, “Feature Selection in the Classification of Network Traffic Flows,” in *2019 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*, IEEE, Oct. 2019, pp. 1–5. doi: 10.1109/FarEastCon.2019.8934169.
- [151] A. Santos da Silva, J. A. Wickboldt, L. Z. Granville, and A. Schaeffer-Filho, “ATLANTIC: A framework for anomaly traffic detection, classification, and mitigation in SDN,” in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, IEEE, Apr. 2016, pp. 27–35. doi: 10.1109/NOMS.2016.7502793.
- [152] M. M. Isa and L. Mhamdi, “Native SDN Intrusion Detection using Machine Learning,” in *2020 IEEE Eighth International Conference on Communications and Networking (ComNet)*, IEEE, Oct. 2020, pp. 1–7. doi: 10.1109/ComNet47917.2020.9306093.
- [153] “Real Time.” Accessed: Jan. 26, 2023. [Online]. Available: <https://www.gartner.com/en/information-technology/glossary/real-time>
- [154] “Definition Real-Time.” Accessed: Jan. 26, 2023. [Online]. Available: <https://www.suse.com/suse-defines/definition/real-time/>
- [155] C. Leng, Y. Qiao, X. S. Hu, and H. Wang, “Co-scheduling aperiodic real-time tasks with end-to-end firm and soft deadlines in two-stage systems,” *Real-Time Systems*, vol. 56, no. 4, pp. 391–451, Oct. 2020, doi: 10.1007/s11241-020-09352-1.

- [156] W. A. Halang, R. Gumzej, M. Colnarić, and M. Druzovec, "Measuring the Performance of Real-Time Systems," *Real-Time Systems*, vol. 18, no. 1, pp. 59–68, 2000, doi: 10.1023/A:1008102611034.
- [157] D. A. Khorkov, "Methods for testing network-intrusion detection systems," *Scientific and Technical Information Processing*, vol. 39, no. 2, pp. 120–126, Apr. 2012, doi: 10.3103/S0147688212020128.
- [158] P. Mell, R. Lippmann, J. Haines, and M. Zissman, "An Overview of Issues in Testing Intrusion Detection Systems." Accessed: Apr. 11, 2023. [Online]. Available: <https://www.govinfo.gov/content/pkg/GOVPUB-C13-12311f1b97702aad61bee4378916ab9e/pdf/GOVPUB-C13-12311f1b97702aad61bee4378916ab9e.pdf>
- [159] "NFDUMP tools overview." Accessed: Nov. 14, 2022. [Online]. Available: <https://nfdump.sourceforge.net/>
- [160] "False negative rate difference in Watson OpenScale fairness metrics." Accessed: Aug. 31, 2023. [Online]. Available: <https://www.ibm.com/docs/en/cloud-paks/cp-data/4.7.x?topic=metrics-false-negative-rate-difference>
- [161] A. Basuki and A. Adriansyah, "Response time optimization for vulnerability management system by combining the benchmarking and scenario planning models," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 1, pp. 561–570, Feb. 2023, doi: 10.11591/ijece.v13i1.pp561-570.
- [162] L. E. J. Guterres and A. Ashari, "THE ANALYSIS OF WEB SERVER SECURITY FOR MULTIPLE ATTACKS IN THE TIC TIMOR IP NETWORK," *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, vol. 14, no. 1, p. 103, Jan. 2020, doi: 10.22146/ijccs.53265.
- [163] I. Cisco Systems, "What Are the Most Common Cyber Attacks?," Products and Services. [Online]. Available: https://www.cisco.com/c/en_au/products/security/common-cyberattacks.html
- [164] J. Melnick, "Top 10 Most Common Types of Cyber Attacks," *Netwrix*. pp. 1–2, 2022. [Online]. Available: [https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/%0Ahttps://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/#Eavesdropping attack](https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/%0Ahttps://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/#Eavesdropping%20attack)
- [165] B. H. Thacker, S. W. Doebling, F. M. Hemez, M. C. Anderson, J. E. Pepin, and E. A. Rodriguez, "Concepts of Model Verification and Validation." 2004. Accessed: Jan. 25, 2024. [Online]. Available: https://inis.iaea.org/search/search.aspx?orig_q=RN:36030870
- [166] C. Yin and A. McKay, "Model verification and validation strategies and methods: An application case study," in *ISCIIA and ITCA 2018 - 8th International Symposium on Computational Intelligence and Industrial Applications and 12th China-Japan International Workshop on Information Technology and Control Applications*, 2018.

- [167] A. Thakkar and R. Lohiya, “A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions,” *Artif Intell Rev*, vol. 55, no. 1, pp. 453–563, Jan. 2022, doi: 10.1007/s10462-021-10037-9.
- [168] Q. Hu, M. R. Asghar, and N. Brownlee, “Evaluating network intrusion detection systems for high-speed networks,” in *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, IEEE, Nov. 2017, pp. 1–6. doi: 10.1109/ATNAC.2017.8215374.
- [169] C. Corbett, T. Basic, T. Lukaseder, and F. Kargl, “A Testing Framework Architecture Concept for Automotive Intrusion Detection Systems,” 2017.
- [170] M. Ennert, E. Chovancová, and Z. DudlÁková, “Testing of IDS model using several intrusion detection tools,” *Journal of Applied Mathematics and Computational Mechanics*, vol. 14, no. 1, pp. 55–62, Mar. 2015, doi: 10.17512/jamcm.2015.1.05.
- [171] F. Erlacher and F. Dressler, “Testing IDS using GENESIDS,” in *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*, New York, NY, USA: ACM, Aug. 2018, pp. 153–155. doi: 10.1145/3234200.3234204.
- [172] J. Sommers, V. Yegneswaran, and P. Barford, “Toward comprehensive traffic generation for online ids evaluation,” 2005.
- [173] L. Kou, S. Ding, T. Wu, W. Dong, and Y. Yin, “An Intrusion Detection Model for Drone Communication Network in SDN Environment,” *Drones*, vol. 6, no. 11, p. 342, Nov. 2022, doi: 10.3390/drones6110342.
- [174] B. Lan, T.-C. Lo, R. Wei, H.-Y. Tang, and C.-K. Shieh, “A Quantitative Logarithmic Transformation-Based Intrusion Detection System,” *IEEE Access*, vol. 11, pp. 20351–20364, 2023, doi: 10.1109/ACCESS.2023.3248261.
- [175] M. A. Teixeira, M. Zolanvari, K. M. Khan, R. Jain, and N. Meskin, “Flow-based intrusion detection algorithm for supervisory control and data acquisition systems: A real-time approach,” *IET Cyber-Physical Systems: Theory and Applications*, vol. 6, no. 3, 2021, doi: 10.1049/cps2.12016.
- [176] K. Wilailux and S. Ngamsuriyaroj, “Novel Bi-directional Flow-based Traffic Generation Framework for IDS Evaluation and Exploratory Data Analysis,” *Journal of Information Processing*, vol. 29, no. 0, pp. 256–265, 2021, doi: 10.2197/ipsjjip.29.256.
- [177] R. Magán-Carrión, D. Urda, I. Díaz-Cano, and B. Dorronsoro, “Towards a Reliable Comparison and Evaluation of Network Intrusion Detection Systems Based on Machine Learning Approaches,” *Applied Sciences*, vol. 10, no. 5, p. 1775, Mar. 2020, doi: 10.3390/app10051775.

- [178] “Deploying IPS engines in IDS or IPS mode.” Accessed: Mar. 30, 2023. [Online]. Available: <https://help.stonesoft.com/onlinehelp/StoneGate/SMC/6.5.0/GUID-3DDAC105-041F-44FB-854F-E52B8F57B6CB.html>
- [179] P. Vanin *et al.*, “A Study of Network Intrusion Detection Systems Using Artificial Intelligence/Machine Learning,” *Applied Sciences*, vol. 12, no. 22, p. 11752, Nov. 2022, doi: 10.3390/app122211752.
- [180] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, “Network intrusion detection system: A systematic study of machine learning and deep learning approaches,” *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, Jan. 2021, doi: 10.1002/ett.4150.
- [181] “TEST METHODOLOGY Next Generation Intrusion Prevention System (NGIPS),” Jan. 2019.

POPIS SLIKA

Slika 2.1. Arhitektura programski definirane mreže.....	10
Slika 2.2. Usporedba tradicionalnih i SDN-mrežnih uređaja	12
Slika 3.1. Osnovni princip rada strojnog učenja [54]	22
Slika 3.2. Princip rada nadziranog strojnog učenja [55].....	25
Slika 3.3. Princip klasifikacije stablom odluke [62].....	28
Slika 3.4. Prikaz rada klasifikatora slučajne šume [65].....	28
Slika 3.5. Prikaz klasifikacije strojem s potpornim vektorima za slučaj dviju klasa [77]	30
Slika 3.6. Klasifikacija uzorka algoritmom K-najbližeg susjeda [84].....	33
Slika 3.7. Matrica konfuzije za slučaj binarne klasifikacije	35
Slika 3.8. Tipične ROC-krivulje [92]	40
Slika 4.1. Osnovna ideja predloženog rješenja za otkrivanje anomalija mrežnog prometa u hibridnoj SDN-mreži	49
Slika 4.2. Proces odabira i prilagodbe ulaznih podataka	51
Slika 4.3. Odabir metrike za nebalansirani skup podataka [91]	58
Slika 4.4. Rezultati AUC-točnosti i vremena izračuna u odnosu na omjer podskupova za učenje i testiranje na UNSWBNB15-skupu podataka.....	68
Slika 4.5. Rezultati AUC-točnosti i vremena izračuna u odnosu na omjer podskupova za učenje i testiranje na CSE-CIC-IDS2018-skupu podataka.....	68
Slika 4.6. Rezultati AUC-točnosti i vremena izračuna u odnosu na omjer podskupova za učenje i testiranje na LUFlow2021-skupu podataka	69
Slika 4.7. Rezultati AUC-točnosti i vremena izračuna u odnosu na omjer podskupova za učenje i testiranje na NF-UNSW-NB15-v1-skupu podataka	69
Slika 4.8. Rezultati AUC-točnosti i vremena izračuna u odnosu na omjer podskupova za učenje i testiranje na NF-UNSW-NB15-v2-skupu podataka	70
Slika 4.9. Rezultati AUC-točnosti i vremena izračuna u odnosu na omjer podskupova za učenje i testiranje na NF-CSE-CIC-IDS2018-v1-skupu podataka.....	70
Slika 4.10. Rezultati AUC-točnosti i vremena izračuna u odnosu na omjer podskupova za učenje i testiranje na NF-CSE-CIC-IDS2018-v2-skupu podataka.....	71

Slika 4.11. Rezultati AUC-točnosti i vremena izračuna u odnosu na omjer podskupova za učenje i testiranje na NF-UQ-NIDS-v1-skupu podataka	71
Slika 4.12. Usporedba AUC-vrijednosti i tehnika skaliranja s cjelovitim skupovima podataka..	75
Slika 4.13. Usporedba AUC-vrijednosti i tehnika skaliranja s NetFlow-skupovima podataka....	76
Slika 4.14. Usporedba uspješnosti klasifikatora na svim skupovima podataka.....	79
Slika 4.15. Usporedba vremena izračuna AUC-točnosti klasifikacije na svim skupovima podataka	79
Slika 4.16. Unakrsna validacija i određivanje najboljih parametara modela [134]	84
Slika 4.17. Određivanje najboljih hiperparametara klasifikatora unakrsnom validacijom [134].	85
Slika 4.18. Određivanje najboljih hiperparametara klasifikatora unakrsnom validacijom uz jednaku distribuciju klasa (<i>StratifiedKfold</i>) [137].....	86
Slika 4.19. Rezultati AUC-vrijednosti i vremena izračuna tijekom optimiziranja $n_estimators$ hiperparametara algoritma slučajne šume.....	89
Slika 4.20. Osnovne komponente za prikupljanje i analizu NetFlow-zapisa mrežnog prometa [143].....	93
Slika 4.21. NFMIDS-model otkrivanja anomalija u stvarnom vremenu	106
Slika 4.22. Grafikon propusnosti sučelja SDN-uređaja tijekom inicijalnog testiranja bez IPS-funkcionalnosti.....	109
Slika 4.23. Grafikon propusnosti sučelja SDN-uređaja tijekom testiranja s uključenom IPS-funkcijom	114
Slika 5.1. Procedura verifikacije modela [166]	118
Slika 5.2. Predložena metoda verifikacije.....	125
Slika 5.3. Arhitektura mrežne okoline pri verifikaciji NFMIDS-modela.....	130
Slika 5.4. Isječak prikaza rada NFMIDS-a u stvarnom vremenu s kombiniranim pozadinskom prometom	134
Slika 5.5. Prikaz otkrivanja sigurnosnih incidenata u korporativnom NIDS-u	136
Slika 5.6. Prikaz sigurnosnih incidenata i lažno pozitivnih otkrivanja u aplikaciji korporativnog NIDS-a	137

POPIS TABLICA

Tablica 2.1. Pregled literature s aspekta kibernetičkih napada i SDN-mrežne arhitekture	18
Tablica 3.1. Pregled istraživanja upotrebe strojnog učenja u SDN-mrežnim okruženjima.....	46
Tablica 4.1. Opis komponenti eksperimentalne platforme za odabir i prilagodbu ulaznih podataka	52
Tablica 4.2. Osnovne karakteristike odabranih skupova podataka.....	53
Tablica 4.3. Osnovne karakteristike NetFlow-skupova podataka	56
Tablica 4.4. Udio i tip obrisanih značajki tijekom predobrade skupova podataka	60
Tablica 4.5. Udio obrisanih podataka u referentnim skupovima podataka.....	62
Tablica 4.6. Kodiranje labelama značajke tipa protokola.....	64
Tablica 4.7. <i>One-hot</i> -kodiranje značajke tipa protokola.....	65
Tablica 4.8. Usporedba trajanja i uspješnosti klasifikacije na skupu podataka UNSWB-NB15..	66
Tablica 4.9. Usporedba smanjenja AUC-točnosti i vremena izračuna među omjerima 90/10 i 80/20 trening/test podataka	72
Tablica 4.10. Usporedba AUC-vrijednosti i tehnika skaliranja s cjelovitim skupovima podataka	75
Tablica 4.11. Usporedba AUC-vrijednosti i tehnika skaliranja s NetFlow-skupovima podataka	76
Tablica 4.12. Usporedba relevantnih istraživanja otkrivanja anomalija strojnim učenjem.....	82
Tablica 4.13. Najbolji rezultati optimizacije hiperparametara $n_estimators$ algoritma slučajne šume	90
Tablica 4.14. Usporedba točnosti klasifikacije s predefiniranim i optimiziranim vrijednostima hiperparametara klasifikatora Slučajne šume	90
Tablica 4.15. Opis značajki LUFLOW2021-skupa podataka	94
Tablica 4.16. Razlike u značajkama LUFLOW2021-referentnog skupa podataka i stvarnog NetFlow-skupa podataka	95
Tablica 4.17. Usporedba metoda odabira značajki	98
Tablica 4.18. Klasifikacija s cijelim i smanjenim brojem značajki	99
Tablica 4.19. Klasifikacija stvarnog Netflow-prometa s uvezenim anomalijama iz skupova podataka LUFLOW2021 i NF-UQ-NIDS.....	104
Tablica 4.20. Hardverske i softverske komponente NFMIDS-modela.....	105

Tablica 4.21. Tijek inicijalnog testiranja NFMIDS-modela bez IPS-funkcionalnosti.....	111
Tablica 4.22. Propusnost promatranog SDN-sučelja tijekom inicijalnog testiranja.....	112
Tablica 4.23. Tijek testiranja NFMIDS-modela s uključenom IPS-funkcijom	114
Tablica 4.24. Propusnost SDN-sučelja tijekom testiranja s IPS-funkcijom	115
Tablica 5.1. Pregled literature s naglaskom na metode verifikacije IDS-a.....	119
Tablica 5.2. Specifikacija korištenih uređaja tijekom procesa verifikacije	131
Tablica 5.3. Rezultati i usporedba otkrivanja kibernetičkih napada tijekom procesa verifikacije	139
Tablica 5.4. Ocjena verifikacije predloženog NFMIDS-modela	140

SAŽETAK

Disertacija predstavlja inovativan pristup kibernetičkoj sigurnosti u kontekstu hibridnih programski definiranih mreža (SDN), s naglaskom na integraciju tehnika strojnog učenja za otkrivanje anomalija u mrežnom prometu. Istraživanje je usmjereno na rješavanje sigurnosnih izazova u SDN-arhitekturama kroz razvoj sustava za otkrivanje napada (NIDS) temeljenog na strojnom učenju, koji omogućava učinkovito otkrivanje i odgovor na prijetnje u stvarnom vremenu. Predloženi model kombinira prednosti tradicionalnih mrežnih arhitektura s prilagodljivošću SDN-a, pružajući skalabilno i učinkovito otkrivanje anomalija. Postupak odabira i prilagodbe ulaznih podataka, zajedno s optimizacijom algoritma klasifikacije, omogućava visoku točnost modela, što je vrlo važno za zaštitu kritičnih segmenata IT-sustava u korporativnoj mreži. Disertacija također uključuje detaljan pregled postojeće literature, istraživanje metoda integracije strojnog učenja sa SDN-om te rezultate verifikacije modela u stvarnom mrežnom okruženju. Dodatni istraživački doprinos uključuje razvoj metode verifikacije temeljene na analizi višestrukih NetFlow-zapisa, čime se dodatno potvrđuje učinkovitost predloženog modela.

Znanstveni doprinosi ove disertacije uključuju:

1. Postupak odabira i prilagodbe skupa ulaznih podataka za algoritam klasifikacije radi otkrivanja anomalija u hibridnoj programski definiranoj mreži.
2. Predložen je model za otkrivanje anomalija u mrežnom prometu hibridne SDN-mreže, koji koristi algoritme strojnog učenja.
3. Osmišljena je metoda za verifikaciju predloženog modela u stvarnom mrežnom okruženju.

Disertacija donosi značajan doprinos području mrežne sigurnosti, s posebnim naglaskom na napredna i inteligentna sigurnosna rješenja unutar hibridnih SDN-arhitektura.

Ključne riječi: kibernetička sigurnost, hibridna programski definirana mreža, otkrivanje anomalija, strojno učenje, algoritam klasifikacije, metoda verifikacije modela, NetFlow.

ABSTRACT

Anomaly Detection in Network Traffic Using Machine Learning Classification in a Hybrid Software-Defined Network

This dissertation presents an innovative approach to cybersecurity in the context of hybrid Software-Defined Networks (SDN), with a focus on integrating machine learning techniques for anomaly detection in network traffic. The research is aimed at addressing security challenges in SDN architectures through the development of a Network Intrusion Detection System (NIDS) based on machine learning, enabling effective real-time threat recognition and response. The proposed model combines the strengths of traditional network architectures with the flexibility and adaptability of SDN, providing scalable and efficient anomaly detection. The process of selecting and adapting input data, along with the optimization of the classification algorithm, enables high model accuracy, which is crucial for protecting critical segments of IT systems within corporate networks. The dissertation also includes a detailed review of existing literature, an exploration of methods for integrating machine learning with SDN, and results from the model's verification in a real network environment. An additional research contribution includes the development of a verification method based on the analysis of multiple NetFlow records, further confirming the effectiveness of the proposed model.

The scientific contributions of this dissertation include:

1. The development of a process for selecting and adapting the input data set for the classification algorithm to detect anomalies in a hybrid Software-Defined Network.
2. A proposed model for anomaly detection in the network traffic of a hybrid SDN, utilizing machine learning algorithms.
3. The design of a method for verifying the proposed model in a real network environment.

The dissertation makes a significant contribution to the field of network security, with a particular emphasis on advanced and intelligent security solutions within hybrid SDN architectures.

Keywords: cybersecurity, hybrid Software-Defined Network, anomaly detection, machine learning, classification algorithm, model verification method, NetFlow.

ŽIVOTOPIS

Igor Fosić rođen je 26. svibnja 1976. godine u Osijeku. Oženjen je i otac je troje djece. Nakon završenog srednjoškolskog obrazovanja u III. gimnaziji u Osijeku upisuje studij Elektrotehnike, smjer elektronika i automatizacija, na Elektrotehničkom fakultetu Sveučilišta J. J. Strossmayera u Osijeku, gdje uspješno diplomira 2001. godine. 2002. godine zapošljava se u Hrvatskoj Elektroprivredi d.d., u Sektoru za informatiku i telekomunikacije, Područna služba Osijek, gdje radi na poslovima održavanja i dizajniranja mrežne i serverske infrastrukture. Paralelno s radom, 2007. godine upisuje, a 2011. završava Poslijediplomski specijalistički studij Napredne komunikacijske tehnologije, također na Elektrotehničkom fakultetu u Osijeku. Krajem 2017. godine upisuje Poslijediplomski doktorski studij, modul Komunikacije i informatika, na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. U svom znanstveno-istraživačkom radu bavi se kibernetičkom sigurnošću, s posebnim naglaskom na otkrivanje anomalija mrežnog prometa primjenom strojnog učenja. Objavio je više znanstvenih radova u međunarodnim i domaćim časopisima te na konferencijama. Trenutno je zaposlen kao rukovoditelj Područne službe Osijek, Sektora za održavanje i izgradnju mreže u HEP Telekomunikacije d.o.o.

PRILOZI

Prilog A

Konfiguracija Cisco 9300 (SW1 i SW2) preklopnika za izvoz i slanje NetFlow-zapisa u nfcpad kolektor

```
flow record RECORD_IN
match flow direction
match interface input
match ipv4 destination address
match ipv4 protocol
match ipv4 source address
match ipv4 tos
match transport destination-port
match transport source-port
collect counter bytes long
collect counter packets long
collect interface output
collect transport tcp flags
collect timestamp absolute first
collect timestamp absolute last
exit
```

```
flow record RECORD_OUT
match flow direction
match interface output
match ipv4 destination address
match ipv4 protocol
match ipv4 source address
match ipv4 tos
match transport destination-port
match transport source-port
collect counter bytes long
collect counter packets long
collect interface input
collect transport tcp flags
collect timestamp absolute first
collect timestamp absolute last
exit
```

```
flow exporter EXPORTER
destination 10.193.20.162
source Vlan91
transport udp 2055
export-protocol netflow-v9
template data timeout 30
exit
flow monitor MONITOR_IN
exporter EXPORTER
cache timeout inactive 10
cache timeout active 60
record RECORD_IN
exit
flow monitor MONITOR_OUT
exporter EXPORTER
cache timeout inactive 10
cache timeout active 60
record RECORD_OUT
exit
vlan configuration 91
ip flow monitor MONITOR_IN input
ip flow monitor MONITOR_OUT output
```

Prilog B

Konfiguracija Cisco N9K-C93108TC-FX (SDN SW) preklopnika za izvoz i slanje NetFlow-zapisa u nfcpad-kolektor

NetFlow Records

```
<?xml version="1.0" encoding="UTF-8"?><imdata totalCount="1">
<netflowRecordPol
annotation=""
childAction=""
collect="count-bytes,count-pkts,pkt-disp,sampler-id,src-intf,tcp-flags,ts-first,ts-recent"
descr=""
dn="uni/tn-HEP_TK_ZP/recordpol-NF_REC_test"
extMngdBy=""
lcOwn="local"
match="dst-ipv4,dst-port,proto,src-ipv4,src-port"
modTs="2023-02-10T10:36:20.743+02:00"
name="NF_REC_test"
nameAlias=""
ownerKey=""
ownerTag=""
status=""
uid="16002"/></imdata>
```

NetFlow Exporters

```
<?xml version="1.0" encoding="UTF-8"?>
<imdata totalCount="1">
<netflowExporterPol
annotation=""
childAction=""
descr=""
dn="uni/tn-HEP_TK_ZP/exporterpol-NF_EXP_test"
dscp="VA"
dstAddr="10.193.20.162"
dstPort="2055"
extMngdBy=""
lcOwn="local"
modTs="2023-04-24T13:54:37.028+02:00"
monPolDn="uni/tn-common/monepg-default"
name="NF_EXP_test"
nameAlias=""
ownerKey=""
ownerTag=""
sourceIpType="inband-mgmt-ip"
srcAddr="0.0.0.0"
status=""
uid="16002"
ver="v9"/></imdata>
```

NetFlow Monitors

```
<?xml version="1.0" encoding="UTF-8"?>
<imdata totalCount="1">
<netflowMonitorPol
annotation=""
childAction=""
descr=""
dn="uni/tn-HEP_TK_ZP/monitorpol-NF_MON_test"
extMngdBy=""
lcOwn="local"
modTs="2023-02-10T10:37:55.460+02:00"
monPolDn="uni/tn-common/monepg-default"
name="NF_MON_test"
nameAlias=""
ownerKey=""
ownerTag=""
status=""
uid="16002"/></imdata>
```

Prilog C

Provjera puta paketa od računala napadača do računala žrtvi

└─\$ traceroute 10.193.53.21

traceroute to 10.193.53.21 (10.193.53.21), 30 hops max, 60 byte packets

```
1 10.193.91.1 (10.193.91.1) 0.523 ms 0.700 ms 0.912 ms
2 10.230.250.61 (10.230.250.61) 0.602 ms 10.230.250.57 (10.230.250.57) 0.503 ms 0.718 ms
3 10.230.250.62 (10.230.250.62) 0.297 ms 10.230.250.58 (10.230.250.58) 0.300 ms 10.230.250.62 (10.230.250.62) 0.195 ms
4 tkos-teren.data.centar (10.193.53.21) 0.522 ms * *
```

└─\$ traceroute 10.193.52.12

traceroute to 10.193.52.12 (10.193.52.12), 30 hops max, 60 byte packets

```
1 10.193.91.1 (10.193.91.1) 0.514 ms 0.696 ms 0.853 ms
2 10.230.250.61 (10.230.250.61) 0.442 ms 10.230.250.57 (10.230.250.57) 0.455 ms 10.230.250.61 (10.230.250.61) 0.624 ms
3 10.230.250.58 (10.230.250.58) 0.292 ms 0.244 ms 10.230.250.62 (10.230.250.62) 0.299 ms
4 10.193.52.12 (10.193.52.12) 0.337 ms 0.422 ms 0.501 ms
```

└─\$ traceroute 10.193.52.200

traceroute to 10.193.52.200 (10.193.52.200), 30 hops max, 60 byte packets

```
1 10.193.91.1 (10.193.91.1) 0.651 ms 0.861 ms 1.032 ms
2 10.230.250.57 (10.230.250.57) 0.600 ms 10.230.250.61 (10.230.250.61) 0.497 ms 10.230.250.57 (10.230.250.57) 0.808 ms
3 10.230.250.58 (10.230.250.58) 0.272 ms 10.230.250.62 (10.230.250.62) 0.223 ms 10.230.250.58 (10.230.250.58) 0.209 ms
4 TKOS-ifosic.data.centar (10.193.52.200) 0.685 ms * *
```

Prilog D

Python glavni kod primijenjen za pseudokod Algoritam 2 i Algoritam 3

```
import os
os.chdir('/home/igor/ML/scripts/')
#Ucitavanje podataka
from ML_script_real_function1 import fn_dataset
#data=fn_dataset(baza, data_num, baza_name) #(baza, broj_datoteka, ime_baze)
#data1=fn_dataset(1, 6, 'LUFLOW2021') # LUFLOW2021
#data2=fn_dataset(2, 0, 'RealNFExp') # Real Netflow snimka iz srpnja,kolovoza 2022
#data3=fn_dataset(3, 1, 'NF-UQ-NIDS-v1') # Netflow anomalies from CSE-CIC-IDS2018, BoT-IoT, ToN-IoT, UNSW-NB15
#data4=fn_dataset(4, 0, 'RealNF-FalsePositive') # Real Netflow za FP u produkciji
data5=fn_dataset(5, 0, 'RealNF-attack') # Real Netflow laptop attack
data6=fn_dataset(6, 0, 'RealNF-normal') # Real Netflow laptop normal

#Feature selection
from ML_script_real_function1 import fn_feature_select
#data1=fn_feature_select(data1, 1)
#data2=fn_feature_select(data2, 2)
#data3=fn_feature_select(data3, 3)

#Feature convert
#brise sve redove koji imaju NaN vrijednosti
from ML_script_real_function1 import fn_feature_convert
#data1=fn_feature_convert(data1, 'LUFLOW2021')
#data2=fn_feature_convert(data2, 'RealNF')
#data3=fn_feature_convert(data3, 'NF-UQ-NIDS-v1')
data5=fn_feature_convert(data5, 'NF-Real-attack')
data6=fn_feature_convert(data6, 'NF-Real-normal')

#Feature encoding LUFLOW
from ML_script_real_function1 import fn_feature_encoding_LUFLOW
#data1=fn_feature_encoding_LUFLOW(data1)

#Feature encoding RealNF
from ML_script_real_function1 import fn_feature_encoding_realNF
#data2=fn_feature_encoding_realNF(data2)
data5=fn_feature_encoding_realNF(data5)
data6=fn_feature_encoding_realNF(data6)

#Anomaly merge, selection and combining
#from ML_script_real_function1 import fn_anomaly_merge
import pandas as pd
#data=fn_anomaly_merge(data1, data2, data3, data4)
#extract anomalies
#data_anomaly_LUFLOW2021=data1[data1['label']==1]
#data_anomaly_NF_UQ_NIDS_v1=data3[data3['label']==1]
#data = pd.concat([data_anomaly_LUFLOW2021, data_anomaly_NF_UQ_NIDS_v1, data5, data6]).reset_index(drop=True)
#data = pd.concat([data_anomaly_LUFLOW2021, data5, data6]).reset_index(drop=True)
data = pd.concat([data5, data6]).reset_index(drop=True)

#Podatci za ucenje
X_train = data.iloc[:, :-1] #sve osim target stupca
y_train = data.iloc[:, -1:] #target stupac
import numpy as np
y_train = np.ravel(y_train)

#Skaliranje podataka za ucenje
from sklearn.preprocessing import MinMaxScaler
var_sc = MinMaxScaler()
X_train_scale = var_sc.fit_transform(X_train)

#ML initial
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(criterion='entropy', class_weight='balanced', max_features='sqrt', n_estimators=300, n_jobs=-1)
clf.fit(X_train_scale, y_train)

#Real-time detection
import numpy as np
```

```

import time
from ML_script_real_function1 import fn_timer

#brisanje svih pocetnih datoteka
from ML_script_real_function1 import fn_delete_files #pozivam funkciju fn_iz file-a ML_
fn_delete_files('/home/igor/ML/input/')
fn_delete_files('/home/igor/ML/benign/')
fn_delete_files('/home/igor/ML/quarantine/')
fn_delete_files('/home/igor/ML/check/')
fn_delete_files('/home/igor/ML/info/')

os.chdir('/home/igor/ML/input/')
from ML_script_real_function1 import fn_convert_files
from ML_script_real_function1 import fn_load_dataset
from ML_script_real_function1 import fn_load_dataset_info
from ML_script_real_function1 import fn_API_interface

while True:
    time_start = time.time() #pocetak procesa detekcije
    raw_files = [f for f in os.listdir('/home/igor/ML/input/') if f.startswith('nfcapd') and 'current' not in f]
    if raw_files:
        src = str(raw_files[0]) #u input direktoriju, treba kod prebacivanja i brisanja datoteka
        src_csv = str(raw_files[0]+''.csv') #u input direktoriju, treba kod prebacivanja i brisanja datoteka
        csv_file=fn_convert_files(raw_files[0])
        X_test = fn_load_dataset(csv_file)
        X_info = fn_load_dataset_info(csv_file)
        if X_test.empty():
            fn_delete_files('/home/igor/ML/input/')
        else:
            #Encoding features
            X_test=fn_feature_encoding_realNF(X_test)
            #Scaling
            X_test_scale = var_sc.transform(X_test.values)
            #print("Izvršeno je skaliranje")

            num_rec=len(X_test_scale)
            #Machine learning algoritam
            y_pred = clf.predict(X_test_scale)
            #y_pred_log_proba=clf.predict_log_proba(X_test_scale)
            y_pred_proba=clf.predict_proba(X_test_scale)

            count = np.count_nonzero(y_pred == 1) #racuna broj jedinica u predikciji
            pos = np.where(y_pred == 1) #array pozicija gdje se nalaze anomalije
            if (np.size(pos) > 0):
                print('pronadjeno je '+ str(np.size(pos))+''+str(y_pred.shape[0])+' anomalija u zapisu ')

                #spremam anomalije ako su FP rezultati u csv datoteku
                import csv
                import pandas as pd
                anomaly_record=X_test.loc[pos[0],:]
                anomaly_record_info=X_info.loc[pos[0],:]
                anomaly_record_merge = pd.merge(anomaly_record, anomaly_record_info, on=anomaly_record.index)
                anomaly_record_merge.set_index('key_0', inplace=True)
                anomaly_record_merge=anomaly_record_merge.reset_index(drop=True)
                var_filename = '/home/igor/ML/check/check_anomalies'+ '_' +src + '.csv'
                var_filename_info = '/home/igor/ML/info/check_anomalies_info'+ '_' +src + '.csv'
                anomaly_record.to_csv(var_filename, index=False)
                anomaly_record_merge.to_csv(var_filename_info, index=False)

                #prebacivanje datoteke u karantenu
                import shutil
                dst = '/home/igor/ML/quarantine/'+ src
                dst_csv = '/home/igor/ML/quarantine/'+ src_csv
                shutil.move(src, dst)
                shutil.move(src_csv, dst_csv)
                #os.remove(src_csv)
                print('Datoteka '+os.path.abspath(os.getcwd())+''+src+' je obrisana i premjestena u karantenu za daljnju analizu!')

            #zovem API za zaustavljanje interface-a

```

```

import time
#timestr = time.strftime("%Y%m%d-%H%M%S")
anomaly_interface_unique_values = anomaly_record_info['if_out'].unique()
for if_num in range(anomaly_interface_unique_values.shape[0]):
    print ('zovem disable API za interface ' + str(anomaly_interface_unique_values[if_num]) + ' '+str(time.strftime("%H:%M:%S")))
    controllerIP='10.230.249.11'
    interface=str(anomaly_interface_unique_values[if_num])
    #fn_API_interface('disable',controllerIP,interface)
else:
    import shutil
    os.chdir('/home/igor/ML/input/')
    dst = '/home/igor/ML/benign/'+ str(raw_files[0])
    shutil.move(src, dst)
    os.remove(src_csv)
    print('Datoteka '+os.path.abspath(os.getcwd())+'/'+'src+' je obrisana i premjestena u benign direktorij!)
    print('Datoteka '+os.path.abspath(os.getcwd())+'/'+'src_csv+' je obrisana!')
    time_end = time.time() #kraj procesa detekcije
    str_trajanje=fn_timer(time_start, time_end)
    print('proces detekcije za '+str(num_rec)+' zapisa je trajao '+str_trajanje )
else:
    print("Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.")

#print('Cekam 15 sekundi za novu provjeru')
time.sleep(15)

```

Prilog E

Python-funkcije glavnog koda primijenjenog za pseudokod Algoritam 2 i Algoritam 3

```
def fn_timer(time_start, time_end):
    import time
    hours, rem = divmod(time_end-time_start, 3600)
    minutes, seconds = divmod(rem, 60)
    str_trajanje="{:0>2}:{:0>2}:{:05.2f}".format(int(hours),int(minutes),seconds)
    return str_trajanje

def fn_time():
    import time
    return time.time()

def fn_dataset(baza,data_num, baza_name):
    import pandas as pd
    data = pd.DataFrame() #blank dataframe
    # raspored stupaca: bytes_in bytes_out dest_port num_pkts_in proto src_port duration label

    if (baza == 1):
        print ('Ucitavanje seta podataka LUFlow2021...')
        path="/home/igor/ML/dataset/LUFlow2021/"
        for x in range(1,data_num+1):
            #for x in [i for i in range(1,data_num+1) if i != 4]: #izostavlja se datoteka 4 jer je prevelika
            #datapart = pd.read_csv(path+"LUFlow2021 ("+str(x)+").csv", header=None, low_memory=False, skiprows=1) #skiprows - ne učitava se
            #1 red jer je nepotreban, a cijeli dataset bude dtypes=object pa treba raditi konverziju u float
            datapart = pd.read_csv(path+"LUFlow2021 ("+str(x)+").csv", header=None, low_memory=False,
            usecols=[1,2,4,7,8,10,14,15],skiprows=1) #skiprows - ne učitava se 1 red jer je nepotreban, a cijeli dataset bude dtypes=object pa treba raditi
            #konverziju u float
            data = pd.concat([data, datapart]).reset_index(drop=True)
            print ('Ucitano ',str(x),'',str(data_num),' datoteka')
            print('Koristi se set podataka prvih', data_num,'datoteka LUFlow2021, broj zapisa:',len(data.index))

        #reorder columns
        column_names = [1,2,4,7,8,10,15,14] #pozicije kolona
        data = data.reindex(columns=column_names)
        #rename columns
        data.rename({1: 'bytes_in', 2: 'bytes_out', 4:'dest_port', 7:'num_pkts_in', 8:'proto', 10:'src_port', 15:'duration', 14:'label'}, axis=1, inplace=True)
        return data

    elif (baza == 2):
        print ('Ucitavanje NF seta podataka', str(baza_name),'...')
        path="/home/igor/ML/netflow/"

        import glob, os
        os.chdir(path)
        fileCounter = len(glob.glob1(path,"*.csv"))
        fileNum = 1
        for file in glob.glob("*.csv"):
            #print(file)
            datapart = pd.read_csv(file, header=None, usecols=[2,5,6,7,11,12,14],skiprows=1, skipfooter=3, engine='python') #7 features, bez prvog i
            #zadnja 3 reda
            #datapart = pd.read_csv(file, header=None, usecols=[2,5,7,11,12,14], skipfooter=3, engine='python')
            #datapart = pd.read_csv(file, header=None, low_memory=False) #full features
            data = pd.concat([data, datapart]).reset_index(drop=True)
            print ('Ucitana je datoteka ', file, fileNum,' od ukupno ',fileCounter)
            fileNum+=1
        #reorder columns
        column_names = [12,14,6,11,7,5,2] #pozicije kolona prema LUFlow2021
        data = data.reindex(columns=column_names)
        data['label']=0 #dodaje novu kolonu i oznacava kao benigni promet
        #rename columns
        data.rename({12: 'bytes_in', 14: 'bytes_out', 6:'dest_port', 11:'num_pkts_in', 7:'proto', 5:'src_port', 2:'duration'}, axis=1, inplace=True)
        print('Koristi se set podataka datoteke RealNetflow, broj zapisa:',len(data.index))
        return data

    elif (baza==3):
        print ('Ucitavanje NF seta podataka', str(baza_name),'...')
```

```

path="/home/igor/ML/dataset/NFDatasets/"
data = pd.read_csv(path+str(baza_name)+".csv", header=None, usecols=[1,3,4,6,7,8,11,12],skiprows=1, engine='python')
#reorder columns
column_names = [6,8,3,7,4,1,11,12] #pozicije kolona
data = data.reindex(columns=column_names)
#rename columns
data.rename({6: 'bytes_in', 8: 'bytes_out', 3:'dest_port', 7:'num_pkts_in', 4:'proto', 1:'src_port', 11:'duration', 12:'label'}, axis=1, inplace=True)
print('Koristi se set podataka datoteke NF-UQ-NIDS-v1, broj zapisa:',len(data.index))
return data

elif (baza == 4):
print ('Ucitavanje NF seta podataka za RealNetFlowFalsePositive', str(baza_name),'...')
path="/home/igor/ML/check/"

import glob, os
os.chdir(path)
fileCounter = len(glob.glob1(path,"*.csv"))
fileNum = 1
for file in glob.glob("*.csv"):
    #print(file)
    datapart = pd.read_csv(file, header=None,skiprows=1, engine='python')
    data = pd.concat([data, datapart]).reset_index(drop=True)
    print ('Ucitana je datoteka ', file, fileNum,' od ukupno ',fileCounter)
    fileNum+=1
data['label']=0 #dodajem novu kolonu i oznacavam kao benigni promet
data.rename({0: 'bytes_in', 1: 'bytes_out', 2:'dest_port', 3:'num_pkts_in', 4:'proto', 5:'src_port', 6:'duration'}, axis=1, inplace=True)
print('Koristi se set podataka datoteke RealNetflowFalsePositive, broj zapisa:',len(data.index))
return data

elif (baza == 5):
print ('Ucitavanje NF seta podataka', str(baza_name),'...')
path="/home/igor/ML/netflow_attack/"

import glob, os
os.chdir(path)
fileCounter = len(glob.glob1(path,"*.csv"))
fileNum = 1
for file in glob.glob("*.csv"):
    #print(file)
    datapart = pd.read_csv(file, header=None, usecols=[2,5,6,7,11,12,14],skiprows=1, skipfooter=3, engine='python') #7 features, bez prvog i
zadnja 3 reda
    #datapart = pd.read_csv(file, header=None, usecols=[2,5,7,11,12,14], skipfooter=3, engine='python')
    #datapart = pd.read_csv(file, header=None, low_memory=False) #full features
    data = pd.concat([data, datapart]).reset_index(drop=True)
    print ('Ucitana je datoteka ', file, fileNum,' od ukupno ',fileCounter)
    fileNum+=1
#reorder columns
column_names = [12,14,6,11,7,5,2] #pozicije kolona prema LUFLOW2021
data = data.reindex(columns=column_names)
data['label']=1 #dodajem novu kolonu i oznacavam kao benigni promet
#rename columns
data.rename({12: 'bytes_in', 14: 'bytes_out', 6:'dest_port', 11:'num_pkts_in', 7:'proto', 5:'src_port', 2:'duration'}, axis=1, inplace=True)
print('Koristi se set podataka datoteke RealNetflowattack, broj zapisa:',len(data.index))
return data

elif (baza == 6):
print ('Ucitavanje NF seta podataka', str(baza_name),'...')
path="/home/igor/ML/netflow_normal/"

import glob, os
os.chdir(path)
fileCounter = len(glob.glob1(path,"*.csv"))
fileNum = 1
for file in glob.glob("*.csv"):
    #print(file)
    datapart = pd.read_csv(file, header=None, usecols=[2,5,6,7,11,12,14],skiprows=1, skipfooter=3, engine='python') #7 features, bez prvog i
zadnja 3 reda
    #datapart = pd.read_csv(file, header=None, usecols=[2,5,7,11,12,14], skipfooter=3, engine='python')
    #datapart = pd.read_csv(file, header=None, low_memory=False) #full features
    data = pd.concat([data, datapart]).reset_index(drop=True)

```



```

    print('Učitana je datoteka ', file, fileNum, ' od ukupno ', fileCounter)
    fileNum+=1
#reorder columns
column_names = [12,14,6,11,7,5,2] #pozicije kolona prema LUFlow2021
data = data.reindex(columns=column_names)
data['label']=0 #dodaje novu kolonu i oznacava kao benigni promet
#rename columns
data.rename({'12': 'bytes_in', 14: 'bytes_out', 6:'dest_port', 11:'num_pkts_in', 7:'proto', 5:'src_port', 2:'duration'}, axis=1, inplace=True)
print('Koristi se set podataka datoteke RealNetflowNormal, broj zapisa:',len(data.index))
return data

def fn_feature_select(data,baza):
    print('Brisanje nepotrebnih znacajki...')

    if (baza == 1): # LUFlow2021
        data=data[data['label'].str.contains("label|outlier")==False] #brise sve redove koji sadrze string "label" i "outlier" u stupcu label
        print("LUFlow2021 dataset feature selection")
    elif (baza == 2): # Real NetFlow
        #ostaju samo stupci sp,dp,pr,ibyt,obyt,ipkt,td -> 5,6,7,12,14,11,2

#data.drop(columns=[0,1,3,4,8,9,10,13,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47],
inplace=True)
    print("Real NetFlow dataset feature selection - none")
    elif (baza == 3): # NF-UQ-NIDS-v1
        print("NF-UQ-NIDS-v1 dataset feature selection - none")
    return data

def fn_feature_convert(data, baza_name):
    import pandas as pd
    TotalDataBefore=len(data)
    print('Konverzija i ciscenje NaN vrijednosti baze ', baza_name)
    data = data.dropna() #brise sve redove koji imaju NaN vrijednosti
    data = data.reset_index(drop=True) #reindex redova
    TotalDataAfter=len(data)
    print('Obrisano ukupno ',str(TotalDataBefore-TotalDataAfter),' zapisa, odnosno ',str((TotalDataBefore-
TotalDataAfter)/TotalDataBefore*100),'%')
    return data

def fn_feature_encoding_realNF(data):
    import pandas as pd
    #print('Kodiranje kategorijskih znacajki, proto znacajka')
    le_name_mapping = {'L2TP': 115, 'ETHIP': 97, 'IGMP': 2, 'GRE': 47, 'ICMP': 1, 'IPIP': 94, 'OSPF': 89, 'TCP': 6, 'UDP': 17, 'VRRP': 112}
    #data.iloc[:, -1:].value_counts() #unique vrijednosti
    data["proto"].replace(le_name_mapping, inplace=True)
    return data

def fn_feature_encoding_LUFlow(data):
    # za bazu LUFlow2021 treba target prebaciti u binarni oblik
    # data.iloc[:, -1:].value_counts() #imena unique vrijednosti
    import pandas as pd
    num=len(data.columns)-1 #kolona label je pod rednim brojem 'num' tj. 7
    data.iloc[:,num].replace(to_replace='^benign', value=0, regex=True, inplace=True) # sve Bening pretvara u 0
    data.iloc[:,num].replace(to_replace='^malicious', value=1, regex=True, inplace=True) # sve ostalo u pretvara u 1
    data.iloc[:,num] = data.iloc[:,num].apply (pd.to_numeric, errors='coerce') #prebacuje target u dtypes.integer
    return data

def fn_anomaly_merge(data1, data2, data3, data4):
    import pandas as pd
    #extract anomalies
    #data_anomaly_LUFlow2021=data1[data1['label']==1]
    #data_anomaly_NF_UQ_NIDS_v1=data3[data3['label']==1]
    #data = pd.concat([data_anomaly_LUFlow2021, data_anomaly_NF_UQ_NIDS_v1, data2, data4]).reset_index(drop=True)
    data = pd.concat([data1, data3, data2, data4]).reset_index(drop=True)
    #data = pd.concat([data1, data_anomaly_NF_UQ_NIDS_v1, data2]).reset_index(drop=True)
    return data

def fn_data_splitting(data):
    print('Splitting...')
    import numpy as np
    y = data.iloc[:, -1:] #target kolona

```

```

y = np.ravel(y)
X = data.iloc[:, :-1] #sve osim target kolone
var_test_percentage = 0.2 #train size 80%, test size 20%
from sklearn.model_selection import train_test_split
#random_state = 1 da svaki puta bude isti raspored, stratify=y da se ravnomjerno raspodjele klase po train i test datasetu
print ("Radi se splitting s omjerom test_size:", var_test_percentage)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=var_test_percentage, random_state=1, stratify=y)
return X_train, X_test, y_train, y_test, var_test_percentage

def fn_email(var_start):
    print ('Saljem e-mail...')
    import smtplib
    import datetime
    from email.message import EmailMessage
    body_string = 'ML skripta pocela u:' + var_start + ' ML skripta zavrsla u:' + datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    msg = EmailMessage()
    msg.set_content(body_string)
    msg['Subject'] = 'ML algoritam CVSearch'
    msg['From'] = 'MachineLearning@sos02ml'
    msg['To'] = 'igor.fosic@hep.hr'

    s = smtplib.SMTP('10.10.20.10', 25)
    s.send_message(msg)
    s.quit()
    return

def fn_filename(baza_name, var_ekstenzija):
    import time
    timestr = time.strftime("%Y%m%d-%H%M%S")
    var_filename = '/home/igor/ML/scripts/'+baza_name+'_'+timestr + var_ekstenzija #koristi datum+vrijeme+imeskripte za png file, putanja ista
gdje su skripte
    return var_filename

def fn_delete_files(path):
    import os
    raw_files_input_dir = [f for f in os.listdir(path) if 'current' not in f]
    if raw_files_input_dir:
        os.chdir(path)
        for i in range(len(raw_files_input_dir)):
            os.remove(raw_files_input_dir[i])
            print('Obrisana datoteka '+ str(path)+str(raw_files_input_dir))

def fn_convert_files(raw_file):
    import subprocess
    convert_action = "nfdump -r " + raw_file + " -b -o csv > "+raw_file+".csv"
    output = subprocess.check_output(convert_action, shell=True)
    csv_file=raw_file+".csv"
    return csv_file

def fn_load_dataset(csv_file):
    import pandas as pd
    data4 = pd.read_csv(
        csv_file,
        header=None,
        usecols=[2,5,6,7,11,12,14,15],
        skiprows=1,
        skipfooter=3,
        engine='python'
    )#7 features, bez prvog i zadnja 3 reda, 15 je input interface
    column_names = [12,14,6,11,7,5,2] #pozicije kolona prema LUFLOW2021
    data4 = data4.reindex(columns=column_names)
    data4.rename({'12': 'bytes_in', 14: 'bytes_out', 6: 'dest_port', 11: 'num_pkts_in', 7: 'proto', 5: 'src_port', 2: 'duration'}, axis=1, inplace=True)
    return data4

def fn_load_dataset_info(csv_file):
    import pandas as pd
    data = pd.read_csv(
        csv_file,
        header=None,
        usecols=[3,4,15,16,44],

```

```

        skiprows=1,
        skipfooter=3,
        engine='python'
    )#7 features, bez prvog i zadnja 3 reda, 15 je input interface
data.rename({'3': 'src', 4: 'dst', 15:'if_in', 16:'if_out', 44:'device'}, axis=1, inplace=True)
return data

def fn_API_interface(action,controllerIP,interface):
    import urllib3
    import json
    import requests
    import time
    urllib3.disable_warnings()
    #url = "https://10.230.249.11/api/aaaLogin.json"

    #authentication
    url = "https://"+str(controllerIP)+"/api/aaaLogin.json"
    payload={"aaaUser":{"attributes":{"name":"fosictest","pwd":"testfosic"}}}
    headers = {'Content-Type': 'application/json'}
    response = requests.request("POST", url, headers=headers, json=payload, verify=False)
    #response.status_code
    r_json = response.json()
    token = r_json["imdata"][0]["aaaLogin"]["attributes"]["token"]
    cookie = 'APIC-cookie='+token
    interface='eth1/1' # interface se dobije iz NetFlow-a

    #disable
    if (action=='disable'):
        payload2={"fabricRsOosPath":{"attributes":{"tDn":"topology/pod-1/paths-1121/pathep-["+interface+"]","lc":"blacklist"},"children": []}}

    #enable
    elif (action=='enable'):
        payload2={"fabricRsOosPath":{"attributes":{"dn":"uni/fabric/outofsvc/rsoosPath-[topology/pod-1/paths-1121/pathep-["+interface+"]","status":"deleted"},"children": []}}

    url="https://"+str(controllerIP)+"/api/node/mo/uni/fabric/outofsvc.json"
    headers = {'Content-Type': 'application/json', 'Cookie': cookie}
    response2 = requests.request("POST", url, headers=headers, json=payload2, verify=False)
    #print(action+' status: '+str(response2.status_code))

```

Prilog F

Rad NFMIDS-modela bez IPS-funkcionalnosti u stvarnom vremenu s kombiniranim pozadinskim prometom

Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
'/home/igor/ML/benign/nfcapd.202304270950'
Datoteka /home/igor/ML/input/nfcapd.202304270950 je obrisana i premjestena u benign direktorij!
Datoteka /home/igor/ML/input/nfcapd.202304270950.csv je obrisana!
proces detekcije za 358 zapisa je trajao 00:00:00.64
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
***** komentar *** Početak kibernetičkog napada uskraćivanjem usluge – tip 1.**
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
pronadjeno je 21261/57773 anomalija u zapisu
'/home/igor/ML/quarantine/nfcapd.202304270951'
'/home/igor/ML/quarantine/nfcapd.202304270951.csv'
Datoteka /home/igor/ML/input/nfcapd.202304270951 je obrisana i premjestena u karantenu za daljnju analizu!
zovem disable API za interface 0 09:53:00
zovem disable API za interface 32 09:53:00
zovem disable API za interface 41 09:53:00
proces detekcije za 57773 zapisa je trajao 00:00:07.36
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
pronadjeno je 54709/65850 anomalija u zapisu
'/home/igor/ML/quarantine/nfcapd.202304270952'
'/home/igor/ML/quarantine/nfcapd.202304270952.csv'
Datoteka /home/igor/ML/input/nfcapd.202304270952 je obrisana i premjestena u karantenu za daljnju analizu!
zovem disable API za interface 0 09:53:54
zovem disable API za interface 41 09:53:54
zovem disable API za interface 32 09:53:54
proces detekcije za 65850 zapisa je trajao 00:00:08.43
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
pronadjeno je 54922/65891 anomalija u zapisu
'/home/igor/ML/quarantine/nfcapd.202304270953'
'/home/igor/ML/quarantine/nfcapd.202304270953.csv'
Datoteka /home/igor/ML/input/nfcapd.202304270953 je obrisana i premjestena u karantenu za daljnju analizu!
zovem disable API za interface 0 09:55:02
zovem disable API za interface 32 09:55:02
zovem disable API za interface 41 09:55:02
proces detekcije za 65891 zapisa je trajao 00:00:08.56
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
***** komentar *** Kraj kibernetičkog napada uskraćivanjem usluge – tip 1.**
***** komentar *** Početak režima normalnog prometa**
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
pronadjeno je 65520/65916 anomalija u zapisu
'/home/igor/ML/quarantine/nfcapd.202304270954'
'/home/igor/ML/quarantine/nfcapd.202304270954.csv'
***** komentar *** bufferirani promet s mrežnih uređaja**
Datoteka /home/igor/ML/input/nfcapd.202304270954 je obrisana i premjestena u karantenu za daljnju analizu!
zovem disable API za interface 0 09:55:56
proces detekcije za 65916 zapisa je trajao 00:00:08.85
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
'/home/igor/ML/benign/nfcapd.202304270955'
Datoteka /home/igor/ML/input/nfcapd.202304270955 je obrisana i premjestena u benign direktorij!
Datoteka /home/igor/ML/input/nfcapd.202304270955.csv je obrisana!
proces detekcije za 371 zapisa je trajao 00:00:00.74
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
'/home/igor/ML/benign/nfcapd.202304270956'
Datoteka /home/igor/ML/input/nfcapd.202304270956 je obrisana i premjestena u benign direktorij!

Datoteka /home/igor/ML/input/nfcapd.202304270956.csv je obrisana!
proces detekcije za 314 zapisa je trajao 00:00:00.69
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
'/home/igor/ML/benign/nfcapd.202304270957'
Datoteka /home/igor/ML/input/nfcapd.202304270957 je obrisana i premjestena u benign direktorij!
Datoteka /home/igor/ML/input/nfcapd.202304270957.csv je obrisana!
proces detekcije za 438 zapisa je trajao 00:00:00.70
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
***** komentar *** Početak kibernetičkog napada otkrivanja korisničkog imena i lozinke – tip 2.**
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
pronadjeno je 17/453 anomalija u zapisu
'/home/igor/ML/quarantine/nfcapd.202304270958'
'/home/igor/ML/quarantine/nfcapd.202304270958.csv'
Datoteka /home/igor/ML/input/nfcapd.202304270958 je obrisana i premjestena u karantenu za daljnju analizu!
zovem disable API za interface 0 09:59:59
zovem disable API za interface 32 09:59:59
proces detekcije za 453 zapisa je trajao 00:00:00.80
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
pronadjeno je 20/495 anomalija u zapisu
'/home/igor/ML/quarantine/nfcapd.202304270959'
'/home/igor/ML/quarantine/nfcapd.202304270959.csv'
Datoteka /home/igor/ML/input/nfcapd.202304270959 je obrisana i premjestena u karantenu za daljnju analizu!
zovem disable API za interface 32 10:01:00
proces detekcije za 495 zapisa je trajao 00:00:00.76
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
pronadjeno je 1/469 anomalija u zapisu
'/home/igor/ML/quarantine/nfcapd.202304271000'
'/home/igor/ML/quarantine/nfcapd.202304271000.csv'
Datoteka /home/igor/ML/input/nfcapd.202304271000 je obrisana i premjestena u karantenu za daljnju analizu!
zovem disable API za interface 32 10:02:01
proces detekcije za 469 zapisa je trajao 00:00:00.72
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
***** komentar *** Početak režima normalnog prometa**
'/home/igor/ML/benign/nfcapd.202304271001'
Datoteka /home/igor/ML/input/nfcapd.202304271001 je obrisana i premjestena u benign direktorij!
Datoteka /home/igor/ML/input/nfcapd.202304271001.csv je obrisana!
proces detekcije za 408 zapisa je trajao 00:00:00.67
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
'/home/igor/ML/benign/nfcapd.202304271002'
Datoteka /home/igor/ML/input/nfcapd.202304271002 je obrisana i premjestena u benign direktorij!
Datoteka /home/igor/ML/input/nfcapd.202304271002.csv je obrisana!
proces detekcije za 338 zapisa je trajao 00:00:00.66
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
'/home/igor/ML/benign/nfcapd.202304271003'
Datoteka /home/igor/ML/input/nfcapd.202304271003 je obrisana i premjestena u benign direktorij!
Datoteka /home/igor/ML/input/nfcapd.202304271003.csv je obrisana!
proces detekcije za 1626 zapisa je trajao 00:00:00.99
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
***** komentar *** Početak kibernetičkog napada skeniranja portova – tip 3.**
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
pronadjeno je 32/1798 anomalija u zapisu
'/home/igor/ML/quarantine/nfcapd.202304271004'
'/home/igor/ML/quarantine/nfcapd.202304271004.csv'
Datoteka /home/igor/ML/input/nfcapd.202304271004 je obrisana i premjestena u karantenu za daljnju analizu!
zovem disable API za interface 0 10:05:50
proces detekcije za 1798 zapisa je trajao 00:00:01.12
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.

Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
pronadjeno je 13/612 anomalija u zapisu
'/home/igor/ML/quarantine/nfcapd.202304271005'
'/home/igor/ML/quarantine/nfcapd.202304271005.csv'
Datoteka /home/igor/ML/input/nfcapd.202304271005 je obrisana i premjestena u karantenu za daljnju analizu!
zovem disable API za interface 0 10:06:50
proces detekcije za 612 zapisa je trajao 00:00:00.80
***** komentar *** Početak režima normalnog prometa**
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
'/home/igor/ML/benign/nfcapd.202304271006'
Datoteka /home/igor/ML/input/nfcapd.202304271006 je obrisana i premjestena u benign direktorij!
Datoteka /home/igor/ML/input/nfcapd.202304271006.csv je obrisana!
proces detekcije za 371 zapisa je trajao 00:00:00.66
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.
Nema .csv datoteka u input direktoriju. Cekam 15 sekundi za novu provjeru.