Primjena metoda strojnog učenja za otkrivanje upada u sustav

Lunder, Domagoj

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Domagoj Lunder

Primjena metoda strojnog učenja za otkrivanje upada u sustav

ZAVRŠNI RAD

Varaždin, 2023.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Domagoj Lunder

Matični broj: 0016137420

Studij: Poslovni sustavi

Primjena metoda strojnog učenja za otkrivanje upada u sustav

ZAVRŠNI RAD

Mentor/Mentorica:

Izv. Prof. dr. sc. Darko Andročec

Varaždin, rujan 2023.

Domagoj Lunder

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Rad prikazuje implementaciju strojnog učenja u svrhe analize komunikacije računalne mreže kako bi se otkrili potencijalni napadi sustava. Početno je objašnjeno koje se tehnologije koriste za zaštitu računalne mreže, a potom je detaljnije objašnjen sustav za otkrivanje upada što je i cilj ovog rada. Zatim, prikazane su različite vrste sustava za otkrivanje upada i u kojima se od navedenih mogu koristiti metode strojnog učenja kako bi se automatizacijom mogli detektirati potencijalni napadi. Implementacija rada napravljena je unutar programskog jezika „Python“ uz pomoć biblioteke „scikit-learn“ koja omogućava rad sa modelima strojnog učenja. Implementacija posjeduje pet različitih modela strojnog učenja za obradu podataka. Dobiveni podaci modela su prikazani grafovima za pojedine pokazatelje uspješnosti modela strojnog učenja dok je rangiranje modela napravljeno prema jednostavnoj bodovnoj skali.

Ključne riječi: IDS; Strojno učenje, Otkrivanje napada, Računalna mreža, Klasifikacija, Skup podataka, Uspješnost modela, Model strojnog učenja

Sadržaj

1. Uvod

Unazad nekoliko godina ubrzan napredak u polju informacijske i komunikacijske tehnologije rezultirao je nevjerojatnim povećanjem interneta kao mreže, a i količine podataka koja se svakodnevno prenosi njome. Kako bi se osigurao spomenuti mrežni promet, a i svi njegovi akteri, pridodaje se velika važnost istraživanjima na tome prostoru. Kako navode Tarter, Bayerl, Karlović, Akghar, Markarian (2017) kibernetski prostor (eng. *Cyberspace*) danas se osigurava pomoću raznih alata kao što su vatrozid (eng. *Firewall*), antivirusni programi (eng. *Antivirus software*) i u konačnici sustavi za otkrivanje upada (eng. *Intrusion Detection System – IDS*). Cilj spomenutih alata jest otkriti načine na koje neovlaštene stranke žele pristupiti podacima. Takvi pokušaji pristupa u literaturi se vode različitim nazivima, ali u konačnici su to zapravo napadi prema podacima kojima je cilj ostvariti neovlašteni pristup.

Tehnologija zaštite kojim će se baviti ovaj rad jest sustav za otkrivanje upada – IDS. Na početku ćemo usporediti spomenute tehnologije zaštite kako bi se predočila neka šira slika gdje će se smjestiti takav tip sustava unutar računalne mreže. Potom će se detaljno pregledati što je zapravo IDS i s kojim ciljem djeluje takav sustav. Kroz pregled takvih sustava prikazati će se mjesto u kojima je moguće implementirati strojno učenje kako bi se IDS podigao na višu razinu i pružao veću korisnost u svakodnevnom djelovanju. Kako bi mogli razumjeti na koji način će strojno učenje poboljšati IDS napraviti će se prvo mala digresija na tu temu u kojoj će se prikazati što je zapravo strojno učenje i u kojoj mjeri se s njime susrećemo u svakodnevnom korištenju internetom. Navedene informacije će zatim poslužiti za lakše razumijevanje implementacije strojnog učenja u IDS-u te će biti prikazane metode strojnog učenja koje će se koristiti u svrhe realizacije ovog rada. Kako bi bilo jasno kakve podatke koristi IDS prikazati će se standardni mrežni promet kojeg IDS prati za otkrivanje napada na sustav te će se odabrati skup podataka pomoću kojeg će se i realizirati. Praktični dio rada će biti orijentiran prema odabiru javno dostupnih biblioteka koje koriste strojno učenje u ove svrhe, a i prikazati će se detalji programskog jezika koji će sve to omogućiti. Na kraju nam preostaje provjera napravljene implementacije u kojoj će se prikazati opći pokazatelji uspješnosti rada napravljenog IDS-a te će se usporediti različiti pristupi, odnosno metode strojnog učenja u odrađivanju zadanog posla kroz statističke podatke i grafove.

2. Tehnologije zaštite računalne mreže

Obratimo li pozornost na vlastitu okolinu vidimo da je gotovo nemoguće biti ne okružen nekom vrstom tehnologije. Sviđalo se to nama ili ne, činjenica jest da su ljudska bića u urbanoj okolini veoma ovisna o tehnologiji koju koriste za rad, za razvoj pa čak i za život. Uzimajući u obzir navedeno, vidimo kako tehnologija s kojom se svakodnevno služimo posjeduje velike količine podataka koje mnogo govore o nama kao njihovim korisnicima.

Prava informacija je u današnje informativno doba veoma vrijedna, ali i u čestim slučajevima predmet želje drugih. Upravo zbog toga, kako navode Liao, R. Lin, Lin i Tung (2013), pojavila se potreba za sigurnosnim sustavima i alatima koji će omogućiti nadzor i analizu mrežnog prometa i rada uređaja neke računalne mreže.

Sigurnost računalne mreže i računala, kako smo prethodno naveli, osigurana je od strane tri posrednika:

- Vatrozid
- Antivirusni program
- Sustav za otkrivanje upada - IDS

2.1. Vatrozid

Vatrozid je prema članku „*What Is a Firewall?"* (bez dat.) mrežni uređaj koji analizira ulazni i izlazni mrežni promet te odlučuje hoće li ga propustiti ili blokirati. Odluka o tome ovisi o skupini sigurnosnih pravila koja su unesena u sam vatrozid. Ovisno o tipu vatrozida i lokaciji implementacije sigurnosna pravila su definirana ručno od strane korisnika ili su unaprijed definirana od strane neke organizacije koja isporučuje vlastitu implementaciju kao npr. vatrozid sustava Windows tvrtke Microsoft. Shodno tome, vatrozid može biti hardverski ili softverski.

Prema članku „*Types of Firewall"* (Javatpoint, bez dat.) hardverski vatrozid je fizički uređaj koji je smješten unutar računalne mreže. Najjednostavnija implementacija ovakvog tipa vatrozida jest mrežni usmjerivač kojeg posjeduje svako kućanstvo s pristupom internetu. Softverski vatrozid je pak s druge strane programski proizvod koji se najčešće nalazi na krajnjem uređaju mreže kao što je osobno računalo. Poznati primjer jest vatrozid sustava „Windows“ tvrtke Microsoft koji dolazi učitan u sklopu operacijskog sustava. Postoje i klasifikacije vatrozida prema načinu rada.

2.2. Antivirusni program

Najbolji odgovor na što je zapravo antivirusni program odgovara pitanje kako ono zapravo radi. Kako navodi CISA (eng. *Cybersecurity & Infrastructure Security Agency*) (2009) antivirusni program skenira datoteke sustava ili memoriju sustava te traži poznate uzorke zloćudnog softvera kako bi obavijestio o istima i zaštitio od istih.

Rad antivirusnog softvera je poznatiji kao skeniranje sustava koje se može odvijati automatski kroz neki period ili ručno po zahtjevu korisnika. Prilikom pretraživanja uređaja na kojem je učitan antivirusni program u slučaju pronalaska zloćudnog softvera postoje različite unaprijed definirane radnje koje navedeni program poduzima. Ovisno o razini prijetnje, antivirusni softver nudi korisniku opciju uklanjanja datoteka ako je riječ o prijetnji niske razine ili suprotno sam otklanja datoteke koje su prijetnja visoke razine. Kao i u svim dijelovima života, pogreške u radu su moguće pa tako i ovdje. Prijetnje visoke razine se stavljaju u osigurani prostor koji je nazvan „karantena“ te korisnik u konačnici odlučuje poznaje li navedeni predmet ili ne te ga sukladno tome odlučuje ukloniti ili ostaviti na osobnom računalu.

2.3. IDS

Priča o sustavima za otkrivanje upada krenula je još 1980. godine kada je u članku „*Computer Security Threat Monitoring and Surveillance“* James P. Anderson predložio ideju o sustavima za otkrivanje upada. Kako navodi članak autora Z. Ahmad, S. Khan, W. Shiang, Abdullah i F. Ahmad (2021) IDS je sigurnosni alat koji stalno prati uređaj unutar računalne mreže na kojem je učitan kao i mrežni promet unutar računalne mreže kako bi otkrio bilo kakvo ponašanje van normi koje bi moglo kršiti zadana sigurnosna pravila. U slučaju takvih povreda sigurnosnih pravila IDS će obavijestiti ovlaštene osobe o detektiranoj povredi sigurnosti i generirati će izvještaje na temelju toga. Veoma je važno za istaknuti kako je IDS zapravo samo alat kojim se otkriva pokušaj upada neovlaštene osobe u sustav. Ono ne radi nikakvu prevenciju te je zbog toga potrebno ili ovlašteno osoblje za upravljanje napadima ili pak kombinacija s nekim sigurnosnim uređajima koji odrađuju prevenciju napada.

Sada kada su pojašnjeni pojmovi tehnologija koji se danas koriste za zaštitu računalne mreže, može se napraviti usporedba navedenih kako bi se što bolje predočile jednakosti između različitih tehnologija kao i razlike koje su zapravo ključne kako bi se u nekoj kombiniranoj izvedbi računalna mreža osigurala u što većoj mjeri.

2.4. Usporedba tehnologija

Spomenute tehnologije zaštite računalne mreže će sada biti uspoređene kroz neke opće kategorije kako bi se upotpunila slika o različitosti tehnologija.

Kategorije koje sam odabrao za usporedbu su proizvoljne te su temeljene na nekim osnovnim značajkama, a informacije su izvedene iz literature koja je spomenuta do sada.

Kategorije prema kojima ćemo usporediti tehnologije zaštite jesu:

- Vrsta implementacije – hardverski ili softverski proizvod
- Područje praćenja – krajnji uređaj ili mreža
- Reakcija na otkrivenu prijetnju
- Područje primjene

Tablica 1. Usporedba tehnologija za zaštitu računalne mreže

	Vatrozid	Antivirusni program	IDS
Vrsta implementacije	Hardverski uređaj ili softverski proizvod	Softverski proizvod	Softverski proizvod
Područje praćenja	Krajnji uređaj i mreža računala	Krajnji uređaj	Krajnji uređaj i mreža računala
Reakcija	Blokira mrežni promet prema unaprijed definiranim pravilima	Blokira programske proizvode prema unaprijed definiranim pravilima	Obavještava ovlaštene osobe o mogućim napadima
Područje primjene	Računalna mreža organizacije ili kućna računalna mreža	Krajnji uređaj na kojem je softverski proizvod učitan	Računalna mreža

(Izvor: Autorski rad)

Sada kada su nam poznate tehnologije pomoću kojih se može zaštititi računalna mreža, prijeđimo na detaljan opis sustava za otkrivanje upada.

3. Detaljni pregled sustava za otkrivanje upada

Kroz predstojeće poglavlje će se pojasniti što je zapravo sustav za otkrivanje upada, što taj sustav radi te će se prikazati dvije opće klasifikacije, odnosno kategorizacije sustava za otkrivanje upada. Detaljno će se napraviti pregled navedenih klasifikacija te će biti istaknute njihove prednosti i mane.

3.1. Što je sustav za otkrivanje upada?

Nacionalni institut za standarde i tehnologiju (eng. *National Institute of Standards and Technology - NIST*) posjeduje u vlastitim arhivima izvještaj koji je namijenjen kao priručnik koji govori o tome što je sustav za otkrivanje upada, zašto bi se trebao koristiti te ostale korisne informacije koje služe čitatelju da u konačnici implementira vlastiti sustav.

Rebecca Bace & Peter Mell (2001) autori su spomenutog izvještaja koji govore da je sustav za otkrivanje upada (eng. „*Intrusion Detection System – IDS*“) softverski ili hardverski proizvod kojemu je zadaća automatizirati proces nadzora događaja koji se odvijaju unutar nekog krajnjeg uređaja npr. osobnog računala ili neke računalne mreže. Sam proces otkrivanja upada (eng. „*Intrusion Detection – ID*“) jest komponenta sustava za otkrivanje upada, koji je zaslužan za praćenje događaja na krajnjem uređaju ili računalnoj mreži.

Danas postoje različite implementacije ovih sustava, ali kako navode Alamiedy, Anbar, Alqattan i Alzubi (2020) neke od ključnih komponenata IDS-a jesu:

- Nadzor i provjera računala, lokalne mreže računala i ulaznih veza
- Slanje obavijesti ili alarma o upadu ovlaštenim osobama
- Kreiranje zapisa napada (eng. „*Logs*“) za kasnije istraživanje

3.2. Zašto trebamo koristiti ovakve sustave?

Sama pomisao na ovu temu nam govori kako je ovdje riječ o sigurnosti sustava i kako nam taj sustav može pomoći kada je potrebno saznati želi li neovlaštena osoba pristupiti našem sustavu, a ako već je pristupila tada želimo saznati način na koji je to napravljeno. Prethodni odlomak govori o tome kako su neke funkcionalnosti nužne za rad sustava za otkrivanje upada. Bazirajući se na tim funkcionalnostima autori NIST-ovog izvještaja su također nabrojali i dobrobiti korištenja sustava za otkrivanje upada koji dodatno opisuju koje sve zadaće bi ovakav sustav trebao moći ispuniti.

Popis benefita koji je sastavljen prema ideji autora Bace i Mell (2001) je sljedeći:

- Mogućnost otkrivanja i kažnjavanja napadača koji je htio neovlašteno pristupiti našem sustavu
- Ispitivanje sigurnosti sustava
- Otkrivanje napada netom prije što se dogodi
- Dokumentacija provedenih napada na naš sustav
- Sakupljanje detalja o napadu kako bi se sigurnost unaprijedila

Nabrojani benefiti i prethodno spomenute funkcionalnosti sustava za otkrivanje upada se mogu protumačiti kao minimalan skup zahtjeva kojega bi ovakav sustav trebao zadovoljiti kako bi bio koristan u svome području rada.

3.3. Kategorizacija sustava za otkrivanje upada

Prilikom razvoja sustava za otkrivanje upada pojavile su se varijacije implementacija koje su bile nužne kako bi se u zadovoljavajućoj razini osigurao čitav sustav. Kada se spomene čitav sustav misli se na sve krajnje uređaje unutar mreže, a to su računala, mrežni poslužitelji pa čak i mrežna oprema poput usmjerivača, preklopnika i ostalog što čini komunikaciju mogućom. Naravno, ne mora biti riječ striktno o organizaciji velike razine, ali s obzirom da je tamo najveća primjena često je predmet promatranja implementiran u takvoj radnoj okolini.

Prilikom istraživanja ove teme sam naišao na članak u kojemu su autori Z. Ahmad, S. Khan, W. Shiang, Abdullah i F. Ahmad (2021) prikazali metode strojnog učenja i dubokog učenja koje postoje danas za izradu sustava za otkrivanje upada. Uzimajući u obzir temu ovog rada metode dubokog učenja će biti zanemarene te će fokus biti stavljen samo na metode strojnog učenja. Ono što je iz tog članka bitno za ovo poglavlje jest upravo kategorizacija implementacija sustava za otkrivanje upada.

Implementacija sustava za otkrivanje upada se u pravilu svrstava u 2 kategorije:

- Sustavi za otkrivanje upada prema mjestu implementacije
- Sustavi za otkrivanje upada prema načinu detekcije upada

Prema članku kojega su sastavili Ahmad i ostali (2021) kategorizacija sustava za otkrivanje upada prema mjestu implementacije govori da su to sustavi koji se nalaze na uređajima unutar mreže ili suprotno sustavi koji se nalaze na mrežnoj opremi. Da pojasnim, sustav za otkrivanje upada koji se nalazi na krajnjim uređajima (eng. „*Host-based Intrusion Detection System – HIDS*“) je zasebna cjelina koja nadzire događaje samog uređaja, ne uzimajući u obzir ostatak mreže. Suprotno tome postoji sustav za otkrivanje upada koji je

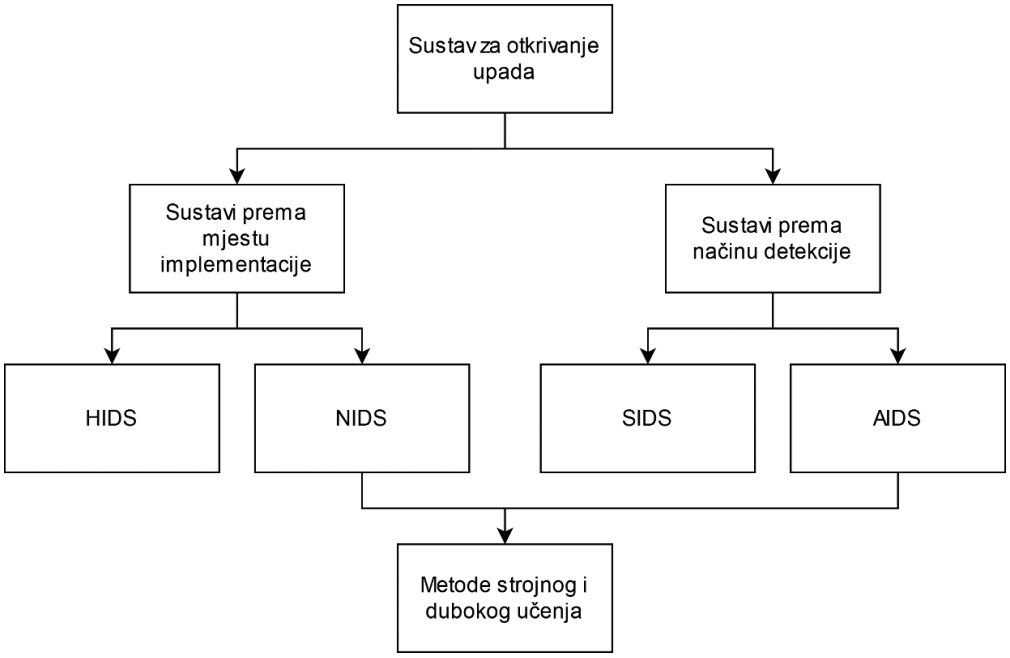
implementiran na mrežnoj opremi (eng. „*Network-based Intrusion Detection System – NIDS*“) te brine samo o komunikaciji između uređaja, točnije provjeravaju mrežni promet, kako navode Ahmad i ostali (2021).

Kategorizacija sustava za otkrivanje upada prema načinu detekcije upada je usmjerena prema načinu rada samog sustava za otkrivanje upada. Naime, kako navode Ahmad i ostali (2021), sustavi za otkrivanje upada prema načinu detekcije se dijele na:

- Sustavi za otkrivanje upada koji rade na principu usporedbe
- Sustavi za otkrivanje upada koji rade na principu analize i predviđanja

Sustavi za otkrivanje upada koji rade na principu usporedbe (eng. „*Signature-based Intrusion Detection System – SIDS*“) posjeduju već poznate mrežne uzorke te usporedbom zaključuju je li riječ o napadu ili ne. Sustavi za otkrivanje upada koji rade na principu analize i predviđanja (eng. „*Anomaly-based Intrusion Detection System – AIDS*“) koriste metode strojnog učenja kako bi analizirali normalan rad računalne mreže te otkrili potencijalne napade i prave pokušaje napada na sustav od strane neovlaštenih osoba.

U prethodno spomenutom članku nalazi se slika koja prikazuje navedenu kategorizaciju sustava za otkrivanje upada. Slika autora iz članka izgleda na sljedeći način.

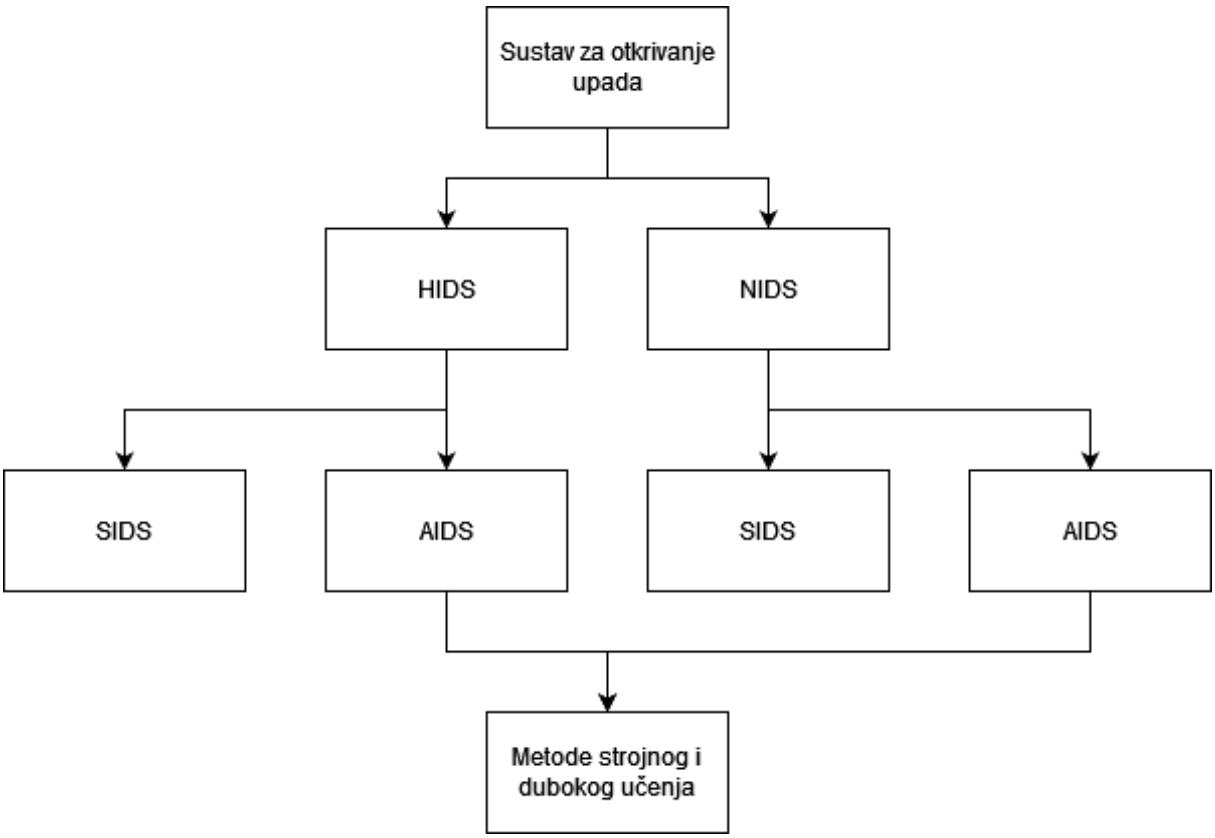


Slika 1. Kategorizacija sustava za otkrivanje upada (Prema: Ahmad i ostali, 2021)

Na slici se vidi prethodno spomenuta podjela sustava koja svrstava sustave za otkrivanje upada u dvije kategorije. Činjenica koju bih ovdje volio dodatno istaknuti, a koja nije

lako uočljiva jest da kategorizacije sustava prema mjestu implementacije i prema načinu detekcije upada nisu međusobno isključive. Drugim riječima sustav za detekciju upada koji je smješten na krajnji uređaj, odnosno HIDS može biti izveden prema dva načina detekcije. Također sustav koji je implementiran na mrežnoj opremi, odnosno NIDS isto tako može biti izveden prema dva različita načina detekcije.

Kako bi navedeno bilo jasnije prepravio sam sliku autora iz članka koja sada na jasniji način prikazuje put do ostvarenja odnosno implementacije sustava za detekciju upada.



Slika 2. Izmijenjena kategorizacija sustava za otkrivanje upada (Autorski rad)

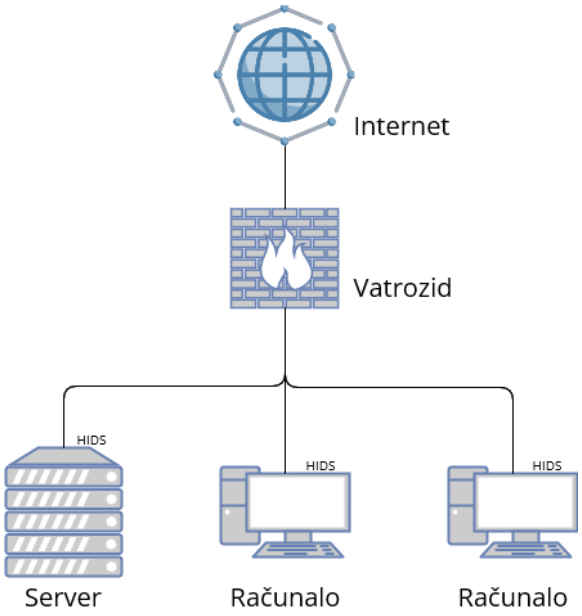
Iz slike iznad se jasno vidi da sustavi prema mjestu implementacije mogu biti izvedeni na jedan od dva spomenuta načina, a svaka implementacija može koristiti jednu od navedenih metoda detekcije upada. Također svaki sustav koji koristi metodu detekciju putem analiza i predviđanja, odnosno AIDS, može u svojoj implementaciji koristiti metode strojnog učenja.

3.4. Sustavi za otkrivanje upada prema mjestu implementacije

Kao što je već prethodno spomenuto, kategorizacija sustava za otkrivanje upada nam govori gdje su smješteni unutar neke računalne mreže. Kroz ovo poglavlje će detaljno biti opisani navedeni sustavi za otkrivanje upada prema mjestu implementacije, biti će prikazani unutar računalne mreže i na kraju će biti navedene prednosti i nedostaci ovog tipa sustava za otkrivanje upada.

3.4.1. HIDS

HIDS (eng. „*Host-based Intrusion Detection System*“) je prema članku „Difference between HIDs and NIDs“ (2022) sustav za otkrivanje upada koji se nalazi na nekom krajnjem uređaju unutar računalne mreže. Takav sustav analizira i provjerava samo podatke uređaja na kojemu se nalazi kao i ulazne i izlazne podatke i datoteke. U suštini sustav radi na način da se aplikacije provjeravaju odnosno uspoređuju s prethodnom slikom aplikacije koja je napravljena kroz neki vremenski period, a ulazni i izlazni podaci se provjeravaju na jedan od prije spomenutih načina detekcije.



Slika 3: Prikaz HIDS-a unutar računalne mreže (Prema: „Difference between HIDs and NIDs“, 2022)

Prethodna slika nam prikazuje smještaj ovakvog tipa sustava unutar računalne mreže. Može se primijetiti kako je bilo i spomenuto da se ovakav tip sustava za otkrivanje upada postavlja na krajnje uređaje unutar računalne mreže.

Prema prethodno navedenome i prikazu ovog tipa sustava sa slike možemo izvesti sljedeće prednosti i nedostatke.

Prednosti korištenja ovog tipa sustava su sljedeće:

- Detekcija upada koji se ne nalaze na mreži
- Nadzor aplikacija koje rade na uređaju

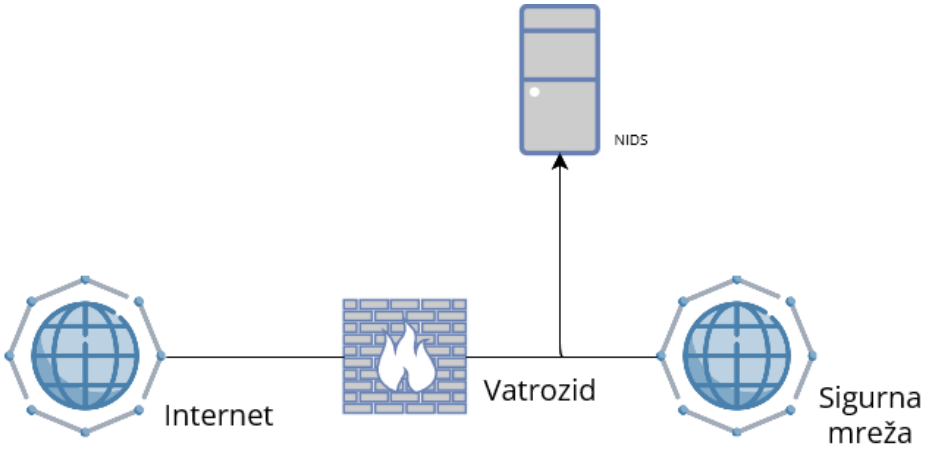
Nedostaci korištenja ovog tipa sustava su sljedeći:

- Odvojen od ostatka računalne mreže
- Potrebna je instalacija na svakom uređaju gdje se ovaj sustav želi primjenjivati
- Dodatno opterećenje resursa krajnjeg uređaja

(„Difference between HIDs and NIDs“, 2022)

3.4.2. NIDS

NIDS (eng. „*Network-based Intrusion Detection System*“) je prema Alamiedyu i ostalima (2020) sustav koji detektira upade na način da provjerava sastav mrežnih paketa neke računalne mreže koji se prenose tijekom komunikacije krajnjih uređaja. NIDS se može smjestiti u bilo kojem dijelu računalne mreže, a uzimajući u obzir njihov tip implementacije gotovo je nemoguće otkriti njihovu lokaciju unutar računalne mreže s vanjske strane iste. S obzirom da se ovakav tip sustava bavi s podacima, odnosno provjerom prometa paketa kroz računalnu mrežu u izradi ovih sustava se mogu primijeniti metode strojnog učenja kako bi se njihova učinkovitost poboljšala te u konačnici kako bi se povećala sigurnost unutar računalne mreže. Jedna od mogućih implementacija ovog sustava izgleda na sljedeći način.



Slika 4: Prikaz NIDS-a unutar računalne mreže (Prema: „Difference between HIDs and NIDs“, 2022)

Prema prethodno navedenom opisu ove kategorije sustava i prema prikazanoj slici, možemo navesti sljedeće prednosti i nedostatke.

Prednosti korištenja ovog tipa sustava su sljedeće:

- Sustav ima mogućnost detekcije upada kroz cijelu računalnu mrežu
- Krajnji uređaji i poslužitelji unutar mreže nisu opterećeni dodatnim softverom
- Prijašnji upadi u sustav se mogu koristiti kao uzorci za detektiranje ponovnih upada koji koriste iste strategije

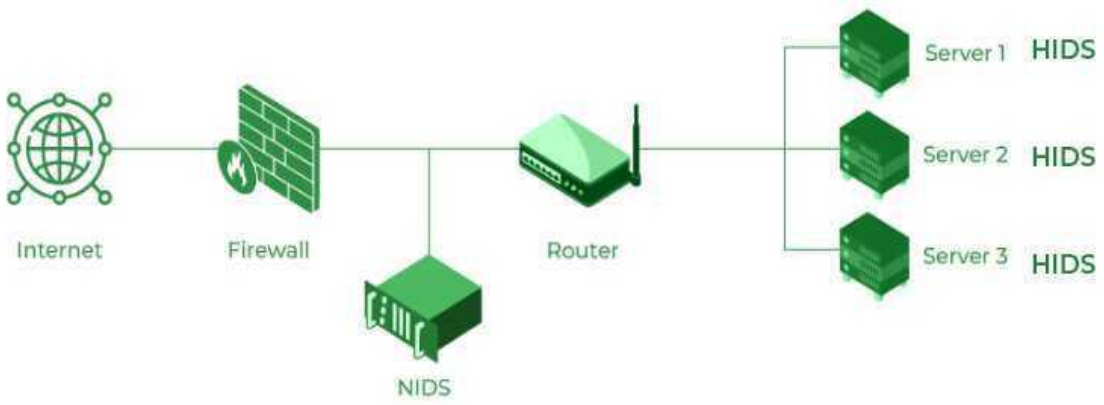
Nedostaci korištenja ovog tipa sustava su sljedeći:

- Brzina prijenosa i provjere je manja u odnosu na današnje brzine mreža
- Računalna mreža je veoma složena zbog dodatnih mrežnih uređaja

(„Difference between HIDs and NIDs“, 2022)

3.4.3. Hibridni sustav

Iz prethodnih pojašnjenja navedenih tipova sustava vidimo kako oba navedena sustava pokrivaju jedan dio mreže. Drugim riječima oba sustava su u svojoj namjeni komplementarni. Kako bi se računalna mreža pokrila u cijelosti logični korak u daljnjoj evoluciji jest postaviti oba sustava da rade na istoj računalnoj mreži. Upravo to možemo vidjeti prikazano na donjoj slici.



Slika 5. Primjena HIDS i NIDS unutar iste računalne mreže
(Izvor: „Difference between HIDs and NIDs“, 2022)

3.5. Sustavi za otkrivanje upada prema načinu detekcije

Podjela sustava prema ovome kriteriju nam govori na koji će način sustav za otkrivanje upada pokušati otkriti upad te na koji će način utvrditi radi li se zaista o upadu. Kao i u prethodnome poglavlju prikazati ćemo detalje pojedine metode, prednosti i nedostatke.

3.5.1. SIDS

SIDS (eng. „*Signature-based Intrusion Detection System*“) je sustav za otkrivanje upada koji prema autorima Kumar i Sangwan (2012) radi na principu prepoznavanja već poznatih napada. U suštini poznati napadi na sustave koriste već poznate tehnike kojima se želi na neki način napakostiti sustavu. Ovakav tip sustava za otkrivanje upada uspoređuje mrežni promet s uzorcima koje već posjeduje u vlastitoj bazi te ako je uzorak isti kao i onaj koji je u bazi naveden kao zloćudan, sustav obavještava o upadu u sustav.

Prednosti korištenja ovakvih sustava su:

- Mali broj pogrešnih detekcija zbog provjerene baze podataka
- Lakoća korištenja

Nedostaci korištenja ovakvih sustava su:

- Kako bi ovaj sustav funkcionirao potrebno je prethodno znanje o upadima u sustav te su potrebni uzorci istih
- Ovakav sustav će nastaviti obavještavati o napadima na sustav ne uzimajući u obzir da je sustav kojega pokušavamo zaštititi već imun na takve oblike napada
- Poznati uzorci napada su mnogobrojni jer postoje mnoge varijacije uzoraka za pojedine operativne sustave, mrežne poslužitelje i sl.

(Kumar i Sangwan, 2012)

3.5.2. AIDS

AIDS (eng. „*Anomaly-based Intrusion Detection System*“) je prema autorima Khraisat, Gondal, Vamplew i Kamruzzaman (2019) napravljen na način da uspoređuje ponašanje računalne mreže s osnovnim napravljenim modelom kako bi se otkrile razlike koje mogu biti potencijalni pokušaji upada u sustav. Osnovni model ponašanja računalne mreže se izrađuje pomoću metoda strojnog učenja.

Prednosti korištenja ovakvih sustava su:

- Mogućnost detekcije novih napada
- Mogućnost kreiranja novih obrazaca napada

Nedostaci korištenja ovakvih sustava su:

- Potreban početni trening sustava
- Teško je napraviti normalan profil ponašanja mreže u dinamičnom okruženju
- Kriptirani paketi se ne mogu skenirati
- Veliki broj netočnih obavijesti

(Khraisat i ostali, 2019)

Sada kada smo prikazali različite vrste sustava za otkrivanje upada, pogledajmo kako se zapravo primjenjuje strojno učenje u ove svrhe.

4. Strojno učenje

Kroz poglavlje o strojnom učenju će biti prikazano što je zapravo strojno učenje te biti će prikazani neki primjeri iz svakodnevnog života gdje se primjenjuje strojno učenje. Potom, će biti prikazane vrste strojnog učenja, objasniti će se zašto je određena vrsta strojnog učenja odabrana za implementaciju ovog rada te na kraju će se proći metode odabrane vrste koje će biti implementirane u praktičnome dijelu.

4.1. O strojnom učenju

Kako je navedeno u članku „*Machine Learning, Explained" (MIT Sloan*, 2023), strojno učenje je podskup umjetne inteligencije čiji je cilj mogućnost oponašanja ljudskog ponašanja. Drugim riječima, inteligentni sustavi se koriste kako bi obavljali složene zadatke kao i ljudi. Godine 1950. pionir ovog područja, Arthur Samuel, je opisao strojno učenje kao mogućnost računala da uče bez da su eksplicitno programirani što je u mnogim izvorima literature navedeno kao najpreciznija definicija ovog područja.

Precizniji opis strojnog učenja, prema članku „*Machine Learning"* (bez dat.), jest da sustav strojnog učenja uz pomoć skupa podataka za treniranje izrađuje matematičke modele koji omogućuju sustavu donošenje pretpostavki i odluka bez da je za to bio eksplicitno programiran. Što je veći testni skup podataka, to će sustav napraviti bolji matematički model koji će kasnije koristiti.

Skup podataka koji se koristi u ovakve svrhe može biti bilo što od osnovnih statističkih podataka o poslovanju neke organizacije, statistički podaci u laboratorijskim istraživanjima pa sve do snimljenog mrežnog prometa što će biti prikazano ovdje. Područja primjene će ukratko biti prikazana u sljedećem poglavlju dok će se detaljna razrada jedne vrste skupa podataka prikazati u poglavlju „5. Podaci računalne mreže“. Spomenuti matematički modeli će se razraditi u poglavlju „4.4. Klasifikacijske metode strojnog učenja“.

4.2. Područje primjene

Zanimljivo je vidjeti kako danas strojno učenje je veoma rasprostranjeno. Kako navodi članak "*Real-World Examples of Machine Learning"* (Tableau, bez dat.) strojno učenje nam unaprjeđuje svakodnevni život bez da smo toga svjesni. Spomenimo nekoliko primjena koje su uistinu zanimljive i u neku ruku zabrinjavajuće u smislu obrade osobnih podataka za koje

nismo ni svjesni, a sami smo pristali na njih. Svi primjeri koji će biti prikazani spomenuti su u prethodno navedenome članku.

Društvene mreže su veliko područje u kojem je postala norma koristiti ovakve metode zbog ogromnih količina podataka. Sve preporuke koji korisnici primaju, svi oglasi koji su krojeni prema našem pretraživanju i korištenju samih društvenih mreža su upravo rezultat primjena metoda strojnog učenja.

Nadalje, prepoznavanje lica (eng. *„facial recognition“*) jest još jedan od korisnih napredaka koji je omogućen primjenom ove tehnologije. Društvene mreže same označuju korisnike koji su na slikama, mobilni telefoni se mogu otključati našim licem, a i sigurnosni sustavi koji koriste kamere mogu implementirati prepoznavanje lica kako bi se moglo identificirati osobe što u preventivne, a što u kurativne svrhe.

Kada već spominjemo mobilne telefone, možda jedna od zanimljivih značajki potpomognuta ovom tehnologijom jest pretvaranje govora u tekst što je osobama s invaliditetom omogućilo normalno funkcioniranje u svakodnevnom životu.

Naravno postoje još i mnogi drugi primjeri.

4.3. Vrste strojnog učenja

Uzimajući u obzir neke od prethodno spomenutih primjera, vidimo kako je zapravo primjena strojnog učenja široka, a skupovi podataka koji se koriste su veoma različiti. Naravno, računalnim sustavima su to sve samo brojke, ali nama kao krajnjim korisnicima treba različiti način interpretacije tih podataka kako bi ih znali iskoristiti u prave svrhe.

Navedeno je navelo do razvoja različitih vrsta strojnog učenja koje se koriste u različite svrhe i s različitim skupovima podataka. Kako navodi članak „*Types of Machine Learning"* (Javatpoint, bez dat.), temeljeno na načinu učenja postoje sljedeće vrste:

- Nadzirano učenje (eng. *„Supervised learning“*)
- Nenadzirano učenje (eng. *„Unsupervised learning“*)
- Djelomično nadzirano učenje (eng. *„Semi-supervised learning“*)
- Podržano učenje (eng. *„Reinforcement learning“*)

Opis pojedine vrste učenja slijede prethodno spomenuti članak.

4.3.1. Nadzirano učenje

Cilj ovog učenja je napraviti model koji će na temelju unesenog skupa podataka izbaciti predviđanja o unosu. Za treniranje, odnosno učenje ovakvog modela koristi se neki manji skup podataka koji već ima poznate klasifikacije pojedinog unosa. U model za treniranje se unose podaci bez spomenutih klasifikacija kao i same klasifikacije te model mora raspoznati o kakvom se unosu radi. Možemo reći kako je zapravo ovaj način učenja pogodan za raspoznavanje ulaza. Banalni primjer iz svakodnevnog života jest upravo korištenje prepoznavanja kako bi znali koje smo voće prikazali modelu. Da pojasnim, u model se unose slike voća kao npr. jabuke i kruške te klasifikacija koja govori je li na slici stvarno jabuka ili kruška. Model tada treba predvidjeti o kojem se voću stvarno radi.

S obzirom na spomenuto, implementacija ovog rada će upravo koristiti ovakav tip učenja kako bi predvidio moguće napade na temelju unesenog mrežnog prometa. Više o tome slijedi u narednim poglavljima.

4.3.2. Nenadzirano učenje

Ovakav tip strojnog učenja koristi skup podataka koji nije unaprijed klasificiran. U model se unosi skup podataka bez očekivanih klasifikacija. Krajnji cilj ovakvog pristupa jest pronaći nekakve veze između podataka i postaviti ih u nekakve odnose, drugim riječima kategorije. Same kategorije identificira model. Uzmimo ponovno primjer s jabukama i kruškama. Model ih sada neće razvrstati prema jabukama i kruškama jer mu to nije zadano, ali će raspoznati neke druge podjele prema boji, obliku i sličnome.

4.3.3. Djelomično nadzirano učenje

Djelomično nadzirano učenje se može predstaviti kao spoje nadziranog i nenadziranog učenja. Naime, označeni skupovi podataka iziskuju duge periode provjere kao i određene financijske resurse. Kako bi model bio što precizniji, ovakav spoj vrsta se koristi kako bi se u određenoj preciznosti neklasificirani skup podataka pretvorio u klasificirani zbog što boljeg krajnjeg predviđanja. Drugim riječima, neklasificirani skup dobiva kategorije podataka koje su nam uistinu bitne, u odnosu na kategorije koje bi nenadzirani model sam postavio.

4.3.4. Podržano učenje

Podržano učenje je u svojim temeljima zapravo sustav nagrada i kazni za poduzete radnje. Model koji se ovdje naziva agent odrađuje radnje i na temelju poduzetih radnji biva nagrađen ili kažnjen. Krajnji cilj ovakvog pristupa jest prikupiti što više nagrada kako bi agent što bolje obavljao svoj zadatak. Primjer iz svakodnevnog života bi bio sljedeći. Agent jest osoba

koja obavlja neku radnju. U ovome slučaju recimo da igra neku kompetitivnu računalnu igru npr. Tetris. Za svaku dobru radnju agent je nagrađen u smislu bodova za dobra postupanja, a kao kazna jest mogućnost prekida igre ako se previše pogrešnih radnji dogodi slijedno. Cilj našeg agenta je upravo sakupiti što više bodova.

4.4. Klasifikacijske metode nadziranog strojnog učenja

Kako je objašnjeno u prethodnom poglavlju, vrsta strojnog učenja odabrana za implementaciju spada pod nadziranu vrstu strojnog učenja. Kako bi implementacija navedenog bila moguća potrebno je moći u programskom kodu implementirati modele i algoritme strojnog učenja. Istraživanjem mogućih metoda pronašao sam „scikit-learn“ biblioteku programskog jezika „Python“ koja služi upravo tome. Na temelju navedenog odabrao sam pet modela strojnog učenja koji sadrže sljedeće metode za klasifikaciju ulaznih skupova podataka.

Klasifikacijske metode odabranih modela strojnog učenja su:

- Gaussian Naive Bayes
- Decision tree
- Random forest
- Logistic regression
- Gradient descent

Prema članku autora P. Patidar (2023) metoda koja primjenjuje klasifikaciju prema Gaussian Naive Bayes algoritmu pretpostavlja da skup podataka prati normalnu distribuciju podataka te temeljem toga predviđa, odnosno klasificira podatke. Pogledajmo pobliže kako je ova klasifikacija nastala.

Navedena klasifikacija je nastala primjenom Bayesovog teorema, Gaussove distributivne funkcije i Naive Bayes algoritma temeljenog na Bayesovom teoremu.

Bayesov teorem nam govori vjerojatnost nekog događaja uzimajući u obzir neki ulazni dokaz. Matematičkom formulom je prikazan na sljedeći način:

$$P(Y|X) = \frac{P(X|Y) * P(Y)}{P(X)}$$

$P(Y|X)$: Uvjetna vjerojatnost događaja Y za promatrani dokaz X

$P(X|Y)$: Vjerojatnost promatranog dokaza X za događaj Y

$P(Y)$: Početna vjerojatnost događaja Y

$P(X)$: Početna vjerojatnost dokaza X

Gaussova distributivna funkcija, poznata kao i prirodna distribucija nam govori vjerojatnost pojave nekog prirodnog fenomena, odnosno događaja. Zadana je sljedećom matematičkom formulom:

$$P(X) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(X-\mu)^2}{2\sigma^2}}$$

σ^2: Kvadrirana standardna devijacija

μ: Očekivana vrijednost distribucije

Konačno, posljednji dio ove slagalice jest Naive Bayesov klasifikacijski algoritam koji nam uz pretpostavku da su dijelovi skupa podataka međusobno neovisni govori o vjerojatnosti događaja u konačnom skupu svih događaja. Zadan je sljedećom formulom:

$$P(X) = P(X_1) * P(X_2) * \ldots * P(X_n)$$

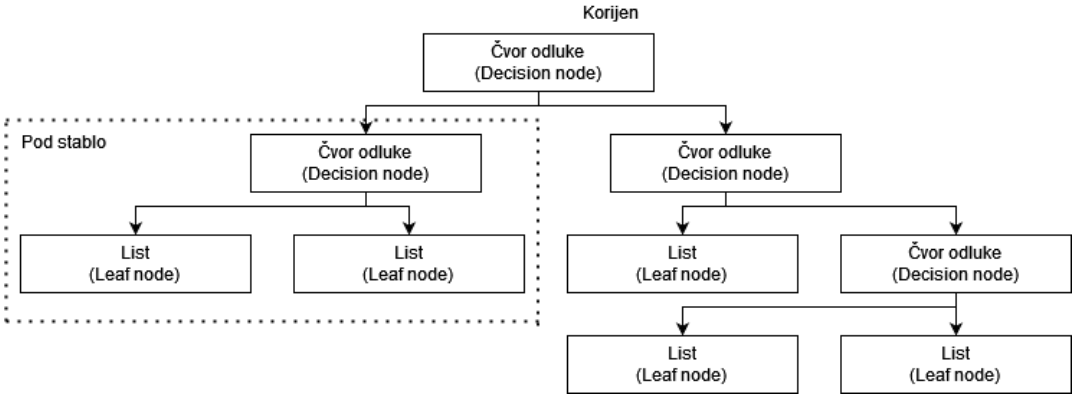
X_1: Događaj 1

X_2: Događaj 2

X_n: Događaj n

Model strojnog učenja koji koristi navedenu klasifikaciju pretpostavlja da početna klasifikacija koju unosimo u model prati prirodnu distribuciju događaja te na temelju toga predviđa koji ulazni podaci spadaju pod koju kategoriju.

Klasifikacijski algoritam stabla odlučivanja (eng. *„decision tree“*) klasificira ulazne podatke prema početnoj klasifikaciji skupa podataka kako navodi članak „*Decision Tree Algorithm in Machine Learning"* (Javatpoint, bez dat.). Stablo odlučivanja se sastoji od dva glavna čvora (eng. *„node“*) koji se nazivaju čvor odluke (eng. *„decision node“*) i (čvor) list (eng. *„leaf node“*). Čvorovi odluke se koriste kako bi se donijele odluke nad podacima, a listovi predstavljaju krajnju klasifikaciju pojedinih zapisa iz skupa podataka. Prilikom korištenja ovog algoritma, ako se radi o složenom skupu podataka s više ulaznih kategorija i više izlaznih klasifikacija, osnovno stablo odlučivanja se može i podijeliti na pod stabla koja se sastoje od istih dijelova, ali jasno prikazuju podjelu ulaznog skupa podataka na dijelove. Pogledajmo primjer sa slike.



Slika 6: Primjer stabla odlučivanja (Prema: „*Decision Tree Algorithm in Machine Learning"* (Javatpoint, bez dat.))

Na prethodnoj slici moguće je zamijetiti spomenuto pod stablo i početni čvor odluke koji je nazvan korijen. Korijen stabla (eng. *„Root node"*) predstavlja početak stabla te su u tom čvoru sadržani ulazni podaci odnosno skup podataka kojeg unosimo u algoritam. Shodno tome, pod stablo predstavlja podskup početnog skupa podataka na kojemu se donose zasebne odluke u odnosu na cijelo stablo. Donošenje odluka u ovom algoritmu je jednostavno, općenito se mogu predstaviti kao da i ne odgovori na pitanja koja postavlja čvor odluke.

Prema članku „*Random Forest Algorithm"* (Javatpoint, bez dat.) ovaj algoritam klasifikacije podataka je proširenje prethodnog algoritma stabla odlučivanja. U suštini, stvara se određeni broj stabala odlučivanja te se uzima prosjek njihovih izlaznih klasifikacija kako bi se povećala preciznost predviđanja nad skupom podataka. Naravno što je veći broj stabala odluke, to će i krajnje predviđanje biti točnije.

Prema navodima iz članka „*Logistic Regression in Machine Learning"* (Javatpoint, bez dat.) navedeni algoritam predviđa izlazne klasifikacije kategorički ovisnih varijabli. Prema tome izlazni podatak algoritma o nekoj varijabli će biti probabilistička vrijednost koja nam govori spada li određena varijabla promatranja u neki zadani klasifikacijski okvir. Što je veća vrijednost pripadnosti varijable nekoj kategoriji, u tu navedenu kategoriju će se varijabla svrstati, odnosno zapis iz skupa podataka će pripasti toj kategoriji.

Gradient descent klasifikacija, prema članku „*Gradient Descent in Machine Learning"* (Javatpoint, bez dat.), koristi optimizacijski algoritam imena Gradient descent kojega je otkrio znanstvenik Augustin-Louis Cauchy. Cilj navedenog optimizacijskog algoritma jest iteracijom minimizirati funkciju koštanja. Da pojasnimo, funkcija koštanja predstavlja mjeru razlike pogreške između stvarnih i očekivanih vrijednosti. Model na temelju spomenutih dvaju koncepata pokušava pronaći optimalan set parametara funkcije koštanja za koje je će model u najbržem roku obaviti klasifikaciju ulaznog skupa podataka.

5. Podaci računalne mreže

Kroz ovo poglavlje ćemo se upoznati sa odabranim skupom podataka koji će se koristiti u implementaciji. Prikazati će se atributi zapisa od kojih je sastavljen skup podataka te će se pojasniti kategorije napada koje će model na kraju trebati prepoznati.

5.1. Skup podataka

Kroz rad se popriličan broj puta spominju skupovi podataka nad kojima model strojnog učenja radi operacije, uči iz njih i klasificira nove skupove. S obzirom da se ovdje radio o primjeni strojnog učenja za predviđanje upada u sustav potrebni su nam skupovi podataka koji sadrže zapise komunikacije unutar računalne mreže kao i klasifikacija tih zapisa kako bi mogli pomoću nadziranog strojnog učenja provesti klasifikaciju skupa podataka.

Na internetu postoje mnogi skupovi podataka napravljeni posebno u ovakve svrhe no, s obzirom na jasnoću i detaljan opis skupa podataka odabrao sam javno dostupan skup podataka pod nazivom „KDD Cup 1999 Data“. Navedeni skup podataka već posjeduje podatke strukturirane na način za korištenje u ove svrhe. S obzirom na opseg ovog rada neće biti korištene sve datoteke koje skup podataka posjeduje pa će biti opisane samo one koje će se koristiti u implementaciji. Pogledajmo te datoteke. Podaci su preuzeti sa službene stranice pod nazivom „*KDD Cup 1999 Data*“.

Lista datoteka:

- kddcup.names – lista atributa koje skup podataka posjeduje
- kddcup.data.gz – arhiva cijelog skupa podataka zajedno s klasifikacijom
- kddcup.data_10_percent.gz – deseti dio arhive skupa podataka pogodan za treniranje modela
- training_attack_types – lista različitih tipova napada

Unutar arhiva „kddcup.data.gz“ i „kddcup.data_10_percent.gz“ smješteni su zapisi mrežne komunikacije. Prema službenim stranicama, podaci su sastavljeni iz tri dijela. Prvi dio su atributi standardnog TCP povezivanja, drugi dio su atributi vezani uz domenu, a na posljednjem mjestu su atributi mrežnog prometa koji su generirani iz prethodnih atributa. Važno je za napomenuti kako unutar arhiva se nalaze samo čisti podaci bez pridodanih naziva stupaca. Upravo iz toga razloga postoji datoteka „kddcup.names“ koja sadrži cijelu listu atributa te će nam služiti za kasnije označavanje skupa podataka. Na kraju preostaje za spomenuti i datoteku „training_attack_types“ koja sadrži popis svih napada unutar skupa

podataka. Navedeno će nam služiti u implementaciji kako bi postavili izlazne klasifikacije za modele strojnog učenja koje treniramo.

Iako detaljan opis atributa ovdje nije bitan jer računalo je to koje nad njima radi predviđanja, pogledajmo ipak o kakvim se atributima radi unutar skupa podataka kako bi lakše shvatili na temelju kakvih podataka će model strojnog učenja donositi predviđanja.

5.1.1. Atributi skupa podataka

Kao što je već prethodno bilo spomenuto, skup podataka se sastoji od tri cjeline. Prva cjelina jest skup osnovnih atributa standardne TCP veze. Prikažimo ih tablicom i opišimo ih.

Tablica 2: Osnovni atributi standardne TCP veze

Naziv atributa	Opis
duration	vrijeme trajanja veze
protocol_type	tip protokola veze (tcp, udp…)
service	mrežni servis odredišta (http, https…)
src_bytes	broj bajtova prenesenih od izvorišta do odredišta
dst_bytes	broj bajtova prenesenih od odredišta do izvorišta
flag	status veze (normal/error)
land	označava vezu ovisno o poslužitelju
wrong_fragment	broj pogrešnih fragmenata
urgent	broj hitnih paketa

(Izvor: *KDD Cup 1999 Data,* bez dat.)

Sljedeće na listi nam je skup atributa vezanih uz domenu. Pogledajmo tablicu.

Tablica 3: Popis atributa domene

Naziv atributa	Opis
hot	broj „hot“ indikatora
num_failed_logins	broj neuspješnih pokušaja prijave
logged_in	označeno „1“ za uspjele prijave i „0“ za sve ostalo
num_compromised	broj kompromitiranih uvjeta

root_shell	označeno „1“ ako je korisnik pristupio korijenskim privilegijama, inače „0“
su_attempted	označeno „1“ ako je korisnik pokušao dobiti korijenske privilegije, inače „0“
num_root	broj pristupa s korijenskim privilegijama
num_file_creations	broj kreiranih datoteka
num_shells	broj upotrijebljenih naredbi
num_access_files	broj operacija napravljenih na datotekama pristupa
num_outbound_cmds	broj izlaznih naredbi ftp sesije
is_hot_login	označeno „1“ ako pripada hot listi, inače „0“
is_guest_login	označeno „1“ ako se prijavio gost, inače „0“

(Izvor: *KDD Cup 1999 Data,* bez dat.)

Preostalo nam je još za prikazati skupinu atributa koja je kreirana na temelju prethodna dva skupa, a također ima važnu ulogu u konačnom skupu.

Tablica 4: Generirani atributi mrežnog prometa

Naziv atributa	Opis
count	broj veza prema istom poslužitelju kao i trenutna veza
serror_rate	postotak veza sa „SYN“ pogreškom
rerror_rate	postotak veza sa „REJ“ pogreškom
same_srv_rate	postotak veza prema istom servisu
diff_srv_rate	postotak veza prema različitim servisima
srv_count	broj veza prema istom servisu kao i trenutna veza
srv_serror_rate	postotak veza prema servisu sa „SYN“ pogreškom
srv_rerror_rate	postotak veza prema servisu sa „REJ“ pogreškom
srv_diff_host_rate	postotak veza prema različitim poslužiteljima

(Izvor: *KDD Cup 1999 Data,* bez dat.)

Sada kada imamo neku generalnu predodžbu sastava skupa podataka preostalo nam je još za pregledati kakvi sve napadi postoje u navedenom skupu podataka, odnosno kako su podijeljeni.

5.1.2. Kategorije napada

Iz službene dokumentacije saznajemo kako u preuzetom skupu podataka postoje 24 različita tipa napada koji se dijele na četiri glavne kategorije. Nesmisleno je opisivati sva 24 različita napada upravo zato jer su to samo varijacije u odnosu na napade iz iste kategorije, ali ćemo ih kasnije samo pobrojiti. Prema tome, pogledajmo različite kategorije napada i opišimo.

- DoS (eng. *„Denial of Service“*) – Prema „*Understanding Denial-of-Service Attacks"* (CISA, 2021), ovaj napad funkcionira na način da „poplavi“ ciljnu mrežu ili poslužitelj mrežnim prometom ta ga na taj način optereti i onemogući daljnji normalan rad sustava. Cilj ovog napada jest onemogućiti normalno funkcioniranje za ovlaštene korisnike.
- R2L (eng. *„Remote to Local“*) – Kako i samo ime govori, napadač s udaljenosti pokušava pristupiti lokalnoj mreži. Prema radu „An Efficient Classifier for U2R, R2L, DoS Attack" (Gupta, 2020.) najčešće je to neka računalna mreža organizacije kojoj napadač pokušava pristupiti, kako bi mogao neovlašteno pristupiti podacima unutar mreže.
- U2R (eng. *„User to Root“*) – Kako navodi Gupta (2020.) u ovoj kategoriji napada ovlašteni korisnik sustava pokušava pristupiti korijenskim privilegijama kako bi mogao dobiti višu razinu prava unutar računalne mreže. Najčešće je cilj korisnika pristupiti podacima kojima nema pristup kao obični korisnik ili u nekim ekstremnim slučajevima obrisati sve podatke neke organizacije ili čak dati javni pristup svim Internet korisnicima.
- Probing – Za razliku od navedenih kategorija napada, ovaj napad ima nešto drugačiji krajnji cilj. Kako navode Labib i Vemuri (bez dat.) ova kategorija napada služi u informacijske svrhe. Različitim tipovima napada, ova kategorija napada pokušava otkriti ranjivosti računalne mreže ili poslužitelja pomoću kojih će biti moguće provesti „DoS“ kategoriju napada.

Svaka kategorija napada ima svoje tipove, odnosno varijacije u smislu implementacije i provođenja, a opet služe istome cilju. Uzimajući u obzir činjenicu da poznavanje pojedinog napada ne doprinosi iznimno ovoj implementaciji, upoznajmo se samo s njihovim imenima.

Tablica 5: Tipovi napada prema kategorijama

Kategorija napada	Tipovi napada
DoS	back, land, neptune, pod, smurf, teardrop
R2L	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster
U2R	buffer_overflow, loadmodule, pearl, rootkit
Probing	ipsweep, nmap, portsweep, satan,

(Izvor: *KDD Cup 1999 Data,* bez dat.)

Sada kada je prikazan skup podataka koji će se koristiti za implementaciju kao i nekoliko važnih detalja o računalnoj mreži, moguće je započeti s provedbom praktičnog dijela ovog rada.

6. Implementacija sustava za otkrivanje upada

Poglavlje će govoriti o samom programskom jeziku koji je odabran za ovu implementaciju, zatim će biti predstavljeni neki osnovni koncepti samog programskog jezika kao što je to npr. sintaksa programa, petlja i slično. Nakon toga će se prikazati programske biblioteke koje će biti korištene za implementaciju sustava za otkrivanje upada te će se opisati programski kod. Na kraju će se prikazati izlazni podaci koji se prikazuju korisniku tijekom korištenja programa.

6.1. Programski jezik

U svrhe implementacije ovog završnog rada odabran je programski jezik „Python“. Razlog odabira ovog programskog jezika je veoma jednostavan, sam programski jezik posjeduje mnogobrojne biblioteke koje nam služe u statističke svrhe. Ipak, najvažnija stavka zbog koje sam odabrao ovaj programski jezik jest „scikit-learn“ biblioteka koja omogućava veoma jednostavno korištenje strojnog učenja za rad nad ulaznim skupovima podataka. Više o navedenoj biblioteci će biti spomenuto u predstojećem poglavlju, a sad spomenimo nekoliko činjenica o samom programskom jeziku, kada je nastao te zašto je toliko popularan.

Slika 7. Simbol programskog jezika Pyhton (Izvor: https://www.ntuclearninghub.com/documents/51786/4216795/Python-Symbol.png/369e410e-a90f-f887-c2dc-61f7ef761476?t=1679043970578)

Kada pojedinac želi naučiti ponešto o programskom jeziku, često se pronađe na besplatnim Internet stranicama koje služe javnosti za lakše savladavanje tražene tematike. Tako za programiranje postoje različita mjesta koja uvode pojedinca u svijet željenog programskog jezika. Smatram kako članak „*Introduction to Python*" (bez dat.) ističe sve što je nama potrebno kako bi dokazali zašto je Python upravo pravi izbor za ovakav tip implementacije. Prema tome, istaknute karakteristike programskog jezika Python slijediti će navedeni članak.

Programski jezik Python nastao je 1991. godine iz želje za stvaranjem programskog jezika koji će biti lak za čitanje, koji će imati laku sintaksu i koji će koristiti što manje linija programskog koda za ostvarivanje standardnih koncepata programiranja. Upravo to je bila glavna zamisao njegova kreatora Guida van Rossuma. Kasnije kako je programski jezik sazrijevao, zbog svoje jednostavnosti privukao je široke mase koje su cijenile nižu razinu apstrakcije programskog koda pa je tako programski jezik postao pogodan za korištenje u istraživačke svrhe upravo zato jer znanstvenici nisu trebali godine kako bi ga savladali. Danas programski jezik se koristi u sljedeće svrhe:

- Kreiranje serverskih aplikacija
- Spajanje baza podataka i rad s datotekama
- Rukovanje s velikim skupovima podataka kako bi obavljao složene matematičke operacije
- Kreiranje prototipskog i produkcijskog softvera

6.2. Biblioteke programskog jezika

Kako navodi članak „The difference between libraries and frameworks" (2023) programske biblioteke su kolekcija prethodno napisanog programskog koda koje služe za obavljanje specifičnih zadataka u nekoj implementaciji. Za primjer postoje biblioteke za upravljanje datotekama, biblioteke za čitanje datoteka, biblioteke za statističku analizu podataka i mnoge druge. Trenutno postoje dvije vrste biblioteka programskog jezika, jedna su biblioteke samog programskog jezika koje dolaze uz taj jezik, a druge su biblioteke „treće strane“ koje su dostupne kao proširenja za programski jezik. Komponente iz biblioteke programskog jezika ćemo opisati u sklopu programskog koda u predstojećem poglavlju, a sada prođimo kroz vanjske biblioteke koje su korištene u implementaciji ovog rada.

6.2.1. Pandas

Kako navodi članak „About *pandas*" (bez dat.) na službenim stranicama biblioteke, „pandas“ je biblioteka koja nastoji pružati mogućnost analize podataka iz stvarnog svijeta unutar programskog jezika „Python“. Uz mogućnosti analize podržane su i određene metode za oblikovanje spomenutih podataka kako bi se mogle provoditi operacije nad njima.

Pogledajmo neke od koncepata koji su korišteni u implementaciji:

„DataFrame“ – služi nam za kreiranje dvodimenzionalne tablice koja pohranjuje podatke. Tablica posjeduje indeksni redak i indeksni stupac u koje prema želji

smještamo nazive pojedinih redaka, odnosno stupaca te su na njihovom sjecištu smješteni sami podaci.

„ExcelWriter“ - omogućuje nam upis „DataFrame“ tablice u Excel datoteku

„read_csv“ – učitavanje podataka iz neke vanjske datoteke u prethodno kreirani „DataFrame“ kako bi kasnije imali podatke nad kojima možemo obavljati operacije

Primjer korištenja unutar programskog koda:

```
import pandas as pd
StatisticsDF = pd.DataFrame()
StatisticsDF = pd.read_csv(datoteka, naziviStupaca)
writer = pd.ExcelWriter()
```

Navedeni isječak programskog koda nam prikazuje uključenje biblioteke u program, kreiranje dvodimenzionalne tablice „DataFrame“ pod nazivom StatisticsDF za unos podataka, čitanje iz datoteke i unos podataka u dvodimenzionalnu tablicu te korištenje metode koja kreira objekt „writer“ za ispis dvodimenzionalne tablice podataka u MS Excel datoteku.

6.2.2. NumPy

„NumPy“ je javno dostupna biblioteka programskog jezika „Python“ koja se bavi izvođenjem matematičkih operacija i numeričkih operacija nad skupovima podataka, kako navodi članak „*NumPy - About Us"* (bez dat.).

Iz navedene biblioteke korištena je samo jedna metoda:

„diag“ – dohvaćanje podataka iz tablice koji se nalaze na dijagonali s lijeva na desno te s vrha prema dolje

Primjena navedene metode izgleda na sljedeći način:

```
import numpy as np
podaciDijagonale = np.diag(dvodimenzionalnoPolje)
```

U gornjem primjeru uključujemo biblioteku u program te zatim iz nekog dvodimenzionalnog polja pospremamo u varijablu „podaciDijagonale“ vrijednosti koje se nalaze na dijagonali navedenog dvodimenzionalnog polja. Dohvaćanje podataka iz dvodimenzionalnog polja jest prema dijagonali s lijeva na desno te s vrha prema dolje.

6.2.3. Colorama

Biblioteka ne posjeduje službenu stranicu no prema unosu u repozitorij „colorama 0.4.6“ (PyPi, bez dat.) služi nam za oblikovanje, odnosno promjenu boje izlaznog teksta programa.

Metode korištene u biblioteci su sljedeće:

„Fore“ – postavlja oblikovanje izlaznog teksta, odnosno njegove boje u neku zadanu

„Style“ – koristi se oblikovanje izlaznog teksta u cijelosti kao i za ponovno postavljanje oblikovanja

Primjena unutar programskog koda:

```
from colorama import Fore, Style
print(Fore.GREEN + "| Sustav za oktrivanje upada |" + Style.RESET_ALL)
```

Isječak koda iz biblioteke dohvaća objekte „Fore“ i „Style“ kako bi se mogli koristiti u oblikovanju teksta. Zatim tekst kojeg ispisujemo početno postavljamo u zelenu boju sa naredbom „Fore.GREEN“ te ga nakon izlaznog teksta ponovno postavljamo na zadano oblikovanje pomoću naredbe „Style.RESET_ALL“.

6.2.4. scikit-learn

Biblioteka „scikit-learn“ prema službenoj dokumentaciji autora Pedregosa i ostalih (2011) služi kako bi se unutar programskog jezika „Python“ mogli služiti algoritmima i modelima strojnog učenja.

Koncepti korišteni iz ove biblioteke:

„naive_bayes“ – uvođenje Gaussian Naive Bayes modela i algoritma u program

„tree“ – uvođenje Decision Tree modela i algoritma u program

„ensemble“ – uvođenje Random Forest i Gradient Boosting modela i algoritma u program

„linear_model“ – uvođenje Logistic Regression modela i algoritma u program

S obzirom da je način korištenja jednak za sve modele strojnog učenja pogledajmo primjer za model strojnog učenja koji koriti Gaussian Naive Bayes klasifikaciju:

```
from sklearn.naive_bayes import GaussianNB
modelGNB = GaussianNB()
model.fit(skupPodatakaZaTreniranjeModela, klasifikacijaZapisaSkupaPodataka)
prediction = model.predict(testniSkupPodataka)
```

Gornji primjer preuzima objekt koji koristi „Gaussian Naive Bayes“ algoritam iz biblioteke, inicijalizira navedeni objekt te odrađuje treniranje modela pomoću „fit“ naredbe i provodi testiranje, odnosno predviđanja na neklasificiranom skupu podatka pomoću naredbe „predict“.

6.2.5. Matplotlib

Biblioteka služi za kreiranje statičkih, animiranih i interaktivnih vizualizacija unutar programa „Python“.

Metode biblioteke korištene u implementaciji:

„plot“ – iscrtavanje grafa

„xticks“ – služi za oblikovanje labela grafa

„title“ – služi za dodavanje naziva grafu

„legend“ – služi za upravljanje legendom iscrtanog grafa

„savefig“ – sprema iscrtani graf van okvira programskog koda

Primjena unutar programskog koda:

```
trainingTimeValuesDF.plot(kind = 'bar')
plt.xticks(rotation=0)
plt.title('Usporedba vremena treniranja pojedinog modela')
plt.legend(bbox_to_anchor=(1.02, 0.1), loc='upper left', borderaxespad=0)
plt.savefig('treniranje.png',bbox_inches='tight')
```

Na gornjem primjeru je prikazano iscrtavanje grafa za različita vremena treniranja modela strojnog učenja. Kako bi graf iscrtali dodajemo mu skup podataka naredbom „plot“, a zatim mu oblikujemo izgled naredbama „xticks“, „title“ i „legend“. Na posljetku se graf sprema van okvira programskog koda narebdom „savefig“.

6.3. Programski kod

Implementaciju sustava za otkrivanje upada unutar programskog koda sam podijelio na sljedeće cjeline:

1. Inicijalizacija programa – uvođenje potrebnih programskih biblioteka u program i inicijalizacija globalnih varijabli
2. Učitavanje skupova podataka – čitanje podataka iz datoteka skupa podataka

3. Oblikovanje skupova podataka – pripremanje skupova podataka za unos u model strojnog učenja
4. Inicijalizacija ulaznih varijabli modela – uzimamo oblikovane skupove podatka iz učitanih datoteka te ih postavljamo u varijable za unos u model
5. Normalizacija ulaznih varijabli modela – normalizacija podataka prema „MinMax“ skali
6. Inicijalizacija modela strojnog učenja – kreiranje modela strojnog učenja s različitim ugrađenim algoritmima
7. Rad sa modelima strojnog učenja – treniranje i testiranje pojedinog modela strojnog učenja, spremanje podataka u MS Excel datoteku i spremanje podataka za iscrtavanje grafova statističkih pokazatelja
8. Iscrtavanje i spremanje grafova – kreiranje grafova pojedinog statističkog pokazatelja i spremanje van okvira programa

Prije prolaska kroz programski kod volio bih istaknuti kako su pojedine cjeline koda opisane komentarima te program kod obavljanja svog zadatak ispisuje poruke korisnike kako bi znao na čemu trenutno program radi.

Prođimo sada kroz pojedine dijelove programskog koda. Prvo će biti prikazan isječak programskog koda, a zatim će biti objašnjeno čemu pojedini dio služi.

6.3.1. Inicijalizacija programa

```
# Uvoz potrebnih biblioteka
from colorama import Fore, Style
from pathlib import Path
import pandas as pd
import time
import numpy as np
from sklearn.metrics import confusion_matrix, precision_score,
accuracy_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
import matplotlib.pyplot as plt

# Globalne varijable
```

```
# Varijable vezane uz statističke izračune
TrainingTimeValues = {}
TestingTimeValues = {}
PrecisionValues = {}
RecallValues = {}
FPRValues = {}
TNRValues = {}
AccuracyValues = {}
FMeassureValues={}
# Varijable vezane uz kreiranje MS Excel datoteke
fileName = 'IDS.xlsx'
writer = pd.ExcelWriter(fileName, engine = 'xlsxwriter',
engine_kwargs={'options':{'strings_to_formulas': False}})
# Prema dokumentaciji skupa podataka dodajemo nazive stupaca
# te pridodajemo dodatan naziv stupca koji označava naziv napada.
columnHeaderNames =['duration', 'protocol_type', 'service', 'flag',
'src_bytes', 'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell',
'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
'num_access_files', 'num_outbound_cmds', 'is_host_login', 'is_guest_login',
'count', 'srv_count', 'serror_rate', 'srv_serror_rate', 'rerror_rate',
'srv_rerror_rate', 'same_srv_rate','diff_srv_rate', 'srv_diff_host_rate',
'dst_host_count', 'dst_host_srv_count', 'dst_host_same_srv_rate',
'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
'dst_host_srv_rerror_rate', 'attack_name']
# Prema dokumentaciji skupa podataka dodajemo nazive napada
# zajedno s njihovim kategorijama. Postoje 4 glavne kategorije.
attacksNamesAndCategory = {'normal': 'normal','back':
'dos','buffer_overflow': 'u2r','ftp_write': 'r2l','guess_passwd':
'r2l','imap': 'r2l','ipsweep': 'probe','land': 'dos','loadmodule':
'u2r','multihop': 'r2l','neptune': 'dos','nmap': 'probe','perl':
'u2r','phf': 'r2l','pod': 'dos','portsweep': 'probe','rootkit':
'u2r','satan': 'probe','smurf': 'dos','spy': 'r2l','teardrop':
'dos','warezclient': 'r2l','warezmaster': 'r2l'}


# Ispis početne poruke
print("=" * 30)
print(Fore.GREEN + "| Sustav za oktrivanje upada |" + Style.RESET_ALL)
print("=" * 30)
```

Prvi korak jest učitavanje potrebnih biblioteka u program. zatim, postavljamo globalne varijable koje se koriste kroz programski kod u cijelosti. Potrebno je bilo postaviti ih iz razloga njihove dostupnosti kako bi se mogle koristiti unutar funkcija u programu. Nadalje prikazujemo

početnu poruku programa i učitavamo putanje datoteka u kojima se nalaze skupovi podataka s kojim radimo u programu. Završni korak jest unos naziva atributa iz skupa podataka i unos različitih tipova napada kao i njihovih kategorija.

6.3.2. Učitavanje skupova podataka

```
# Dohvaćanje putanje mape u kojoj se nalazi program.
currentPath = Path(__file__).parent.absolute()
# Postavljanje putanje za skup podataka kojim će se trenirati model za detekciju.
trainingDataPath = Path.joinpath(currentPath, "KDD Cup 1999 Data/kddcup.data_10_percent.gz")
# Postavljanje putanje za skup podataka kojim će se evaluirati uspješnost napravljenog modela za detekciju.
testingDataPath = Path.joinpath(currentPath, "KDD Cup 1999 Data/kddcup.data.gz")
# Učitavanje početnog skupa podataka za treniranje modela.
print(Fore.YELLOW + "Učitavam skup podataka za treniranje." + Style.RESET_ALL)
trainDF = pd.read_csv(trainingDataPath, names = columnHeaderNames)
# Učitavanje početnog skupa podataka za testiranje modela.
print(Fore.YELLOW + "Učitavam skup podataka za testiranje." + Style.RESET_ALL)
testDF = pd.read_csv(testingDataPath, names = columnHeaderNames)
```

Navedeni isječak koda pomoću biblioteke „pandas“ učitava zapise iz skupa podataka za treniranje i testnog skupa podataka u „DataFrame“ biblioteke pandas te sada možemo započeti s obradom unesenih podataka.

6.3.3. Oblikovanje skupova podataka

```
# Dodavanje stupca koji označava atribut napada u skupu podataka.
print(Fore.YELLOW + "Dodajem kategorije napada u skupove podataka." + Style.RESET_ALL)
trainDF['attack_category'] = trainDF.attack_name.apply(lambda r:attacksNamesAndCategory[r[:-1]])
testDF['attack_category'] = testDF.attack_name.apply(lambda r:attacksNamesAndCategory[r[:-1]])
# Dohvaćanje oznaka kategorije napada iz testnog skupa podataka. Oznake su dohvaćene silazno prema broju
# pojavljivanja unutar testnog skupa podataka. Lista kategorija se kasnije koristi za oblikovanje statističkih skupova podataka.
AttackCategoryListDF = testDF['attack_category'].value_counts()
AttackCategoryList = AttackCategoryListDF.index.tolist()
```

```
# Pretvaranje simboličkih pokazatelja u numeričke.
print(Fore.YELLOW + "Pretvaram simboličke pokazatelje u numeričke." +
Style.RESET_ALL)
pmap = {'icmp':0,'tcp':1,'udp':2}
trainDF['protocol_type'] = trainDF['protocol_type'].map(pmap)
testDF['protocol_type'] = testDF['protocol_type'].map(pmap)
fmap = {'SF':0,'S0':1,'REJ':2,'RSTR':3,'RSTO':4,'SH':5 ,'S1':6
,'S2':7,'RSTOS0':8,'S3':9 ,'OTH':10}
trainDF['flag'] = trainDF['flag'].map(fmap)
testDF['flag'] = testDF['flag'].map(fmap)
# Uklanjanje nevažnog pokazatelja iz skupova podataka.
print(Fore.YELLOW + "Oblikujem skupove podataka za unos u model." +
Style.RESET_ALL)
print(Fore.YELLOW + "Uklanjam \"Service\" simbolički pokazatelj." +
Style.RESET_ALL)
trainDF.drop('service',axis = 1,inplace= True)
testDF.drop('service',axis = 1,inplace= True)
# Uklanjane atributa koji označava vrstu zapisa iz skupova podataka.
print(Fore.YELLOW + "Uklanjam \"Attack name\" simbolički pokazatelj." +
Style.RESET_ALL)
trainDF = trainDF.drop(['attack_name',], axis=1)
testDF = testDF.drop(['attack_name',], axis=1)
```

Redoslijedom izvršenja programskog koda radimo sljedeće. Početno dodajemo u skup podataka za treniranje i testiranje novi atribut „attack_name“ koji nam govori o kakvom se zapisu radi. Konkretno je li riječ o nekom napada iz navedenih kategorija ili o normalnom zapisu. Razlog tome jest kako bi u sljedećem koraku mogli prebrojati različite vrste zapisa što će nam kasnije služiti u statističkoj analizi i usporedbi predviđanja modela i stvarnog stanja. Nadalje, s obzirom kako svi atributi u skupu podataka nisu numerički moramo promijeniti navedeno u numeričke podatke ili ih u potpunosti ukloniti zbog toga jer ne utječu na krajnji rezultat. Pa tako atribute „protocol_type“ i „flag“ pretvaramo u numeričke, a pokazatelje „service“ i prethodno definirani „attack_type“ uklanjamo iz skupa podataka. Atribut „attack_type“ je smisleno ukloniti iz razloga jer model treba sam raspoznati zapise po tim kategorijama.

6.3.4. Inicijalizacija ulaznih varijabli modela

```
# Postavljanje varijabla za treniranje i testiranje koje će koristiti
model.
# Varijable sa sufiksom X predstavljaju neoznačeni skup podataka.
# varijable sa sufiksom Y predstavljaju izlazne klasifikacije prema kojima
model obrađuje skup podataka X.
```

```
print(Fore.YELLOW + "Postavljam X i Y varijable za unos skupa podataka u
model." + Style.RESET_ALL)
trainDF_Y = trainDF[['attack_category']]
trainDF_X = trainDF.drop(['attack_category',], axis=1)
testDF_Y = testDF[['attack_category']]
testDF_X = testDF.drop(['attack_category',], axis=1)
```

Modeli strojnog učenja zahtijevaju određene ulazne podatke kako bi mogli odraditi svoj posao. Tako vidimo ovdje da postoje po dva skupa za treniranje i testiranje modela podataka. Skup podataka označen sa sufiksom Y nam predstavlja klasifikacije zapisa skupa podataka prema kojima model mora rasporediti zapise iz skupa s sufiksom X. Dakle, Y sadržava samo nazive kategorija napada ili oznaku da je zapisa normalan dok X predstavlja neoznačeni skup podataka na temelju kojeg se model trenira, a potom i testira.

6.3.5. Normalizacija skupova podataka

```
# Normalizacija skupova podataka po MinMax skali kako bi model bolje
funkcionirao.
print(Fore.YELLOW + "Normaliziram skupove podataka u X i Y varijablama." +
Style.RESET_ALL)
scaler = MinMaxScaler()
trainDF_X = scaler.fit_transform(trainDF_X)
testDF_X = scaler.fit_transform(testDF_X)
```

Skupovi podataka se normaliziraju kako bi se postigla ujedinjenost podataka. Da pojasnim, konkretno atribut skupa podataka „protocol_type“ ima raspon vrijednosti od 0 do 2 dok atribut „flag“ ima raspon vrijednosti od 0 do 10. Kako bi se izbjeglo dodavanje veće težine većim iznosima, od strane modela, podaci se postavljaju na isti raspon vrijednosti dok im se njihovi omjeri zadržavaju. Prema tome vidimo kako ulazne skupove podataka normaliziramo po „MinMax“ skali da postignemo što bolje performanse modela.

6.3.6. Inicijalizacija modela strojnog učenja

```
# Inicijalizacija modela za različite metode strojnog učenja.
print(Fore.YELLOW + "Inicijaliziram modele strojnog učenja." +
Style.RESET_ALL)
modelGNB = GaussianNB()
modelDT = DecisionTreeClassifier(criterion = "entropy", max_depth = 10)
modelRF = RandomForestClassifier(n_estimators = 30)
modelLR = LogisticRegression(max_iter = 1200000)
modelGB = GradientBoostingClassifier(random_state = 0)
```

Navedeni isječak koda preuzima predefinirane modele strojnog učenja iz biblioteke „scikit-learn“ te ih inicijalizira kako bi se mogli koristiti u implementaciji.

6.3.7. Rad sa modelima strojnog učenja

```
# Unos parametara u pojednini model pomoću metode za obradu modela
useModel(trainDF_Y, trainDF_X, testDF_Y, testDF_X, modelGNB, "Gaussian Naive Bayes")
useModel(trainDF_Y, trainDF_X, testDF_Y, testDF_X, modelDT, "Decission Tree")
useModel(trainDF_Y, trainDF_X, testDF_Y, testDF_X, modelRF, "Random Forest")
useModel(trainDF_Y, trainDF_X, testDF_Y, testDF_X, modelLR, "Logistic Regression")
useModel(trainDF_Y, trainDF_X, testDF_Y, testDF_X, modelGB, "Gradient Boosting")
# Funkcija koja izvršava treniranje i testiranje modelima strojnog učenja.
    # Zatim, statistički podaci modela se pospremaju na dva različita "mjesta".
    # Jedan dio služi za spremanje podataka u MS Excel datoteku, a drugi za iscrtavanje grafova.
def useModel(trainDF_Y, trainDF_X, testDF_Y, testDF_X, model, modelName):

    # Treniranje modela.
    print(Fore.YELLOW + "Započinjem obradu " + modelName + " modela." + Style.RESET_ALL)
    print(Fore.YELLOW + "Započinjem treniranje modela." + Style.RESET_ALL)
    startTime = time.time()
    model.fit(trainDF_X, trainDF_Y.values.ravel())
    endTime = time.time()
    trainingTime = endTime-startTime
    print(Fore.YELLOW + "Treniranje modela završeno." + Style.RESET_ALL)

    # Testiranje modela.
    print(Fore.YELLOW + "Započinjem testiranje modela." + Style.RESET_ALL)
    startTime = time.time()
    prediction = model.predict(testDF_X)
    endTime = time.time()
    testingTime = endTime-startTime
    print(Fore.YELLOW + "Testiranje modela završeno." + Style.RESET_ALL)

    # Izračun statističkih pokazatelja na temelju predviđanja modela.
```

```
    print(Fore.YELLOW + "Izračunavam statističke podatke modela." +
Style.RESET_ALL)
    confusionMatrix = confusion_matrix(testDF_Y, prediction)
    truePositive = np.diag(confusionMatrix)
    falsePositive = confusionMatrix.sum(axis=0) - np.diag(confusionMatrix)
    falseNegative = confusionMatrix.sum(axis=1) - np.diag(confusionMatrix)
    trueNegative = confusionMatrix.sum() - (falsePositive + falseNegative +
truePositive)
    precision = precision_score(testDF_Y, prediction, average = 'weighted')
* 100
    recall = truePositive / (truePositive + falseNegative) * 100
    falsePositiveRate = falsePositive / (falsePositive + trueNegative) *
100
    trueNegativeRate = trueNegative / (trueNegative + falsePositive) * 100
    accuracy = accuracy_score(testDF_Y, prediction) * 100
    fMeassure = 2 * ((precision * recall) / (precision + recall))


    # 1. Spremanje podataka u MS Excel datoteku


    # Priprema statističkih podataka modela za spremanje u MS Excel
datoteku.
    print(Fore.YELLOW + "Prirpemam statističke podatke za ispis u MS Excel
datoteku." + Style.RESET_ALL)
    indexStatisticalData = ['Vrijeme treniranja', 'Vrijeme testiranja',
'Preciznost', 'Tocnost']
    StatisticalDataDF = pd.DataFrame(index = indexStatisticalData)
    StatisticalDataDF[modelName] = [trainingTime, testingTime, precision,
accuracy]
    StatisticalDataPerCategoryDF = pd.DataFrame(index = AttackCategoryList)
    StatisticalDataPerCategoryDF['TP'] = truePositive
    StatisticalDataPerCategoryDF['FP'] = falsePositive
    StatisticalDataPerCategoryDF['FN'] = falseNegative
    StatisticalDataPerCategoryDF['TN'] = trueNegative
    StatisticalDataPerCategoryDF['FPR'] = falsePositiveRate
    StatisticalDataPerCategoryDF['TNR'] = trueNegativeRate
    StatisticalDataPerCategoryDF['Opoziv'] = recall
    StatisticalDataPerCategoryDF['F-mjera'] = fMeassure
    # Spremanje statističkih podataka modela u MS Excel datoteku.
    print(Fore.YELLOW + "Spremanje u MS Excel datoteku." + Style.RESET_ALL)
    workbook = writer.book
    worksheet = workbook.add_worksheet(modelName)
    worksheet.set_column(0, 0, 20)
```

```
    worksheet.set_column(1, 1, 18)
    worksheet.set_column(3, 7, 10)
    worksheet.set_column(8, 11, 12)
    writer.sheets[modelName] = worksheet
    StatisticalDataDF.to_excel(writer, sheet_name = modelName, startrow =
0, startcol = 0)
    StatisticalDataPerCategoryDF.to_excel(writer, sheet_name = modelName,
startrow = 0, startcol = 3)

    # 2. Spremanje podataka u globalne varijable programa za kasnije
iscrtavanje grafova.

    TrainingTimeValues[modelName] = trainingTime
    TestingTimeValues[modelName] = testingTime
    PrecisionValues[modelName] = precision
    RecallValues[modelName] = recall
    FPRValues[modelName] = falsePositiveRate
    TNRValues[modelName] = trueNegativeRate
    AccuracyValues[modelName] = accuracy
    FMeassureValues[modelName] = fMeassure

    print(Fore.YELLOW + "Obrada modela završena." + Style.RESET_ALL)

# Finaliziranje i spremanje konačne MS Excel datoteke.
writer.close()
print(Fore.YELLOW + "Kreirana je MS Excel datoteka pod nazivom \"" +
str(fileName) + "\" u korijenskoj datoteci ovog programa." +
Style.RESET_ALL)
```

Pošto se u ovom programskom rješenju radi sa više različitih modela strojnog učenja, kako bi smanjio repetitivnost programskog koda konstruirao sam funkciju pod nazivom „useModel“ koja provodi operacije nad pojedinim modelom te izračunava njegove statističke podatke i na kraju ih dodaje u izlaznu MS Excel datoteku koja nam služi za spremanje statističkih podataka van okvira ovog programa.

Funkcija „useModel“ prima ulazne vrijednosti „trainDF_Y“, „trainDF_X“, „testDF_Y“, „testDF_X“, samu instancu modela s kojim će raditi i naziv modela. Slijedom izvođenja programa funkcija odrađuje sljedeće. Početno trenira model i zatim ga testira. Tijekom treniranja i testiranja modela koristi se programska biblioteka za mjerenje vremena kako bi kasnije mogli usporediti vremena potrebna za dvije navedene radnje kroz različite modele strojnog učenja. Potom program izračunava statističke podatke za pojedini model te ih sprema

u MS Excel datoteku i određene globalne varijable kako bi se kasnije na temelju istih prikazali grafovi prema statističkim pokazateljima modela. Statistički pokazatelji modela su opisani u predstojećem poglavlju „7. Analiza implementiranog sustava za otkrivanje upada“. Kada program obradi sve modele strojnog učenja MS Excel datoteka se finalizira, odnosno zatvara.

6.3.8. Iscrtavanje i spremanje grafova

```
print(Fore.YELLOW + "Kreiram grafove pojedinih pokazatelja." +
Style.RESET_ALL)
# Vrijeme treniranja modela
trainingTimeValuesDF = pd.DataFrame(TrainingTimeValues, index =
['Vremena'])
trainingTimeValuesDF.plot(kind = 'bar')
plt.xticks(rotation=0)
plt.title('Usporedba vremena treniranja pojedinog modela')
plt.legend(bbox_to_anchor=(1.02, 0.1), loc='upper left', borderaxespad=0)
plt.savefig('treniranje.png',bbox_inches='tight')
# Vrijeme testiranja modela
testingTimeValuesDF = pd.DataFrame(TestingTimeValues, index = ['Vremena'])
testingTimeValuesDF.plot(kind = 'bar')
plt.xticks(rotation=0)
plt.title('Usporedba vremena testiranja pojedinog modela')
plt.legend(bbox_to_anchor=(1.02, 0.1), loc='upper left', borderaxespad=0)
plt.savefig('testiranje.png',bbox_inches='tight')
# Preciznost
precisonValuesDF = pd.DataFrame(PrecisionValues, index = ['Vrijendosti
pokazatelja'])
precisonValuesDF.plot(kind = 'bar')
plt.xticks(rotation=0)
plt.title('Preciznost pojedinog modela')
plt.legend(bbox_to_anchor=(1.02, 0.1), loc='upper left', borderaxespad=0)
plt.savefig('precision.png',bbox_inches='tight')
# Opoziv
recallValuesDF = pd.DataFrame(RecallValues, index = AttackCategoryList)
recallValuesDF.plot(kind = 'bar')
plt.xticks(rotation=0)
plt.title('Opoziv po kategorijama')
plt.legend(bbox_to_anchor=(1.02, 0.1), loc='upper left', borderaxespad=0)
plt.savefig('recall.png',bbox_inches='tight')
# FPR
falsePositiveValuesDF = pd.DataFrame(FPRValues, index = AttackCategoryList)
```

```
falsePositiveValuesDF.plot(kind = 'bar')
plt.xticks(rotation=0)
plt.title('Stopa pogrešnih klasifikacija napada po kategorijama')
plt.legend(bbox_to_anchor=(1.02, 0.1), loc='upper left', borderaxespad=0)
plt.savefig('fpr.png',bbox_inches='tight')
# TNR
trueNegativeValuesDF = pd.DataFrame(TNRValues, index = AttackCategoryList)
trueNegativeValuesDF.plot(kind = 'bar')
plt.xticks(rotation=0)
plt.title('Stopa točnih klasifikacija normalnih zapisa po kategorijama')
plt.legend(bbox_to_anchor=(1.02, 0.1), loc='upper left', borderaxespad=0)
plt.savefig('tnr.png',bbox_inches='tight')
# Točnost
accuracyValuesDF = pd.DataFrame(AccuracyValues, index = ['Vrijendosti
pokazatelja'])
accuracyValuesDF.plot(kind = 'bar')
plt.xticks(rotation=0)
plt.title('Točnost pojedinog modela')
plt.legend(bbox_to_anchor=(1.02, 0.1), loc='upper left', borderaxespad=0)
plt.savefig('accuracy.png',bbox_inches='tight')
# F-mjera
fMeassureValuesDF = pd.DataFrame(FMeassureValues, index =
AttackCategoryList)
fMeassureValuesDF.plot(kind = 'bar')
plt.xticks(rotation=0)
plt.title('F-mjera modela po kategorijama')
plt.legend(bbox_to_anchor=(1.02, 0.1), loc='upper left', borderaxespad=0)
plt.savefig('fMeassure.png',bbox_inches='tight')


print(Fore.YELLOW + "Slike grafova statističkih pokazatelja su spremljene u
korijenski direktorij ovog programa." + Style.RESET_ALL)


print(Fore.YELLOW + "Kraj programa." + Style.RESET_ALL)
```

Navedeni isječak koda iscrtava grafove na temelju izračunatih podataka iz prethodnog poglavlja. Grafovi služe kako bi modele strojnog učenja usporedili po pokazateljima uspješnosti modela. Svaki graf je kreiran prema istome principu koji je opisan u sklopu opisa biblioteke „Matplotlib“. Nakon što je program iscrtao grafove za sve pokazatelje uspješnosti modela i spremio ih, izvođenje programa završava.

Prikazani programski kod je javno dostupan kao „GitHub“ repozitorij te se može pogledati na poveznici https://github.com/TheLjunder/Sustav-za-oktrivanje-upada--Intrusion-detection-system.

6.3.9. Izlaz programa

```
==============================
| Sustav za oktrivanje upada |
==============================
Učitavam skup podataka za treniranje.
Učitavam skup podataka za testiranje.
Dodajem kategorije napada u skupove podataka.
Pretvaram simboličke pokazatelje u numeričke.
Oblikujem skupove podataka za unos u model.
Uklanjam "Service" simbolički pokazatelj.
Uklanjam "Attack name" simbolički pokazatelj.
Postavljam X i Y varijable za unos skupa podataka u model.
Normaliziram skupove podataka u X i Y varijablama.
Inicijaliziram modele strojnog učenja.
Započinjem obradu Gaussian Naive Bayes modela.
Započinjem treniranje modela.
Treniranje modela završeno.
Započinjem testiranje modela.
Testiranje modela završeno.
Izračunavam statističke podatke modela.
Prirpemam statističke podatke za ispis u MS Excel datoteku.
Spremanje u MS Excel datoteku.
Obrada modela završena.
Započinjem obradu Decission Tree modela.
Započinjem treniranje modela.
Treniranje modela završeno.
Započinjem testiranje modela.
Testiranje modela završeno.
Izračunavam statističke podatke modela.
Prirpemam statističke podatke za ispis u MS Excel datoteku.
Spremanje u MS Excel datoteku.
Obrada modela završena.
Započinjem obradu Random Forest modela.
Započinjem treniranje modela.
Treniranje modela završeno.
```

```
Započinjem testiranje modela.
Testiranje modela završeno.
Izračunavam statističke podatke modela.
Prirpemam statističke podatke za ispis u MS Excel datoteku.
Spremanje u MS Excel datoteku.
Obrada modela završena.
Započinjem obradu Logistic Regression modela.
Započinjem treniranje modela.
Treniranje modela završeno.
Započinjem testiranje modela.
Testiranje modela završeno.
Izračunavam statističke podatke modela.
Prirpemam statističke podatke za ispis u MS Excel datoteku.
Spremanje u MS Excel datoteku.
Obrada modela završena.
Započinjem obradu Gradient Boosting modela.
Započinjem treniranje modela.
Treniranje modela završeno.
Započinjem testiranje modela.
Testiranje modela završeno.
Izračunavam statističke podatke modela.
Prirpemam statističke podatke za ispis u MS Excel datoteku.
Spremanje u MS Excel datoteku.
Obrada modela završena.
Kreirana je MS Excel datoteka pod nazivom "IDS.xlsx" u korijenski
direktorij ovog programa.
Kreiram grafove pojedinih pokazatelja.
Slike grafova statističkih pokazatelja su spremljene u korijenski
direktorij ovog programa.
Kraj programa.
```

Ukratko, na ovaj način izgledaju izlazne poruke ovog programa. Kao što se može primijetiti program za svaki korak obrade ispisuje korisniku poruku kako bi znao što program radi. Uz navedeni izlazni tekst u datoteci gdje se nalazi sam program kreira se spomenuta MS Excel datoteka koja služi za spremanje statističkih podataka modela kao i iscrtani grafovi pojedinog pokazatelja uspješnosti modela strojnog učenja.

7. Analiza implementiranog sustava za otkrivanje upada

Kroz ovo poglavlje biti će prikazani statistički pokazatelji modela strojnog učenja, biti će objašnjeno kako se izračunavaju i što interpretiraju. Zatim će se pogledati dobiveni podaci različitih implementiranih modela strojnog učenja kroz grafove te će se usporediti dobivene vrijednosti. Na kraju će modeli strojnog učenja biti rangirani prema jednostavnoj bodovnoj skali.

7.1. Statistički pokazatelji modela

Kako bi mogli evaluirati pojedine algoritme modela korištenih u implementaciji koristimo se određenim pokazateljima koji nam govore o predviđanjima koje je napravio implementirani model strojnog učenja. U nastavku ćemo nabrojati i ukratko opisati pojedine pokazatelje. Pokazatelji su preuzeti iz rada autora Ahmad i ostalih (2021).

Pokazatelji uspješnosti modela strojnog učenja su:

- Preciznost modela (eng. *„precision“*)
- Opoziv (eng. *„recall“*)
- Stopa pogrešnih klasifikacija zapisa kao napad (eng. *„false alarm rate“*)
- Stopa točnih klasifikacija normalnih zapisa (eng. *„true negative rate“*)
- Točnost pojedinog modela (eng. *„accuracy“*)
- F-mjera (eng. *„F-measure“*)

Za prikaz navedenih pokazatelja potrebni su nam određeni podaci kako bi ih mogli izračunati. Spomenute pokazatelje dobivamo iz konfuzijske matrice. Kako objašnjava članak „Confusion Matrix in Machine Learning“ (2017) konfuzijska matrica je matrica podataka koja sumira performanse modela strojnog učenja na temelju unesenih testnih podataka. Drugim riječima je to dvodimenzionalna tablica koja na jednoj osi sadrži stvarne podatke, dok na drugoj osi se nalaze klasifikacije koje je predvidio model.

Konfuzijska matrica predviđanja nam služi kako bi analizom mogli utvrditi istinitost izlazne klasifikacije prema četiri bitna pokazatelja. To su:

- True positive (TP) – zapisi iz skupa podataka koji su točno identificirani kao napada od strane modela
- False negative (FN) – zapisi iz skupa podataka koji su pogrešno identificirani kao normalni zapisi od strane modela

- False positive (FP) – zapisi iz skupa podataka koji su pogrešno identificirani kao napadi od strane modela
- True negative (TN) – zapisi iz skupa podataka koji su točno identificirani kao normalni zapisi od strane modela

Pogledajmo kako se na temelju podataka iz konfuzijske matrice izračunavaju pokazatelji uspješnosti modela strojnog učenja.

Preciznost modela (eng. *„precision“*) nam govori omjer točno predviđenih zapisa napada u odnosu na sve zapise koji su identificirani kao napad. Navedeno se dobiva sljedećom formulom.

$$Precision = \frac{TP}{TP + FP}$$

Opoziv (eng. *„recall“*) nam govori o postotku zapisa točno identificiranih u odnosu na se zapise koji su stvarno ispravni. U nekim terminologijama se još naziva i stopa detekcije (eng. *„detection rate“*). Navedeni pokazatelj je dan sljedećom formulom.

$$Recall = \frac{TP}{TP + FN}$$

Stopa pogrešnih klasifikacija zapisa kao napad (eng. *„false alarm rate“*) govori o postotku, odnosno omjeru pogrešno identificiranih napada u odnosu na sve zapise koji su normalni. Izračunava se prema sljedećoj formuli.

$$False\ alarm\ rate = \frac{FP}{FP + TN}$$

Stopa točnih klasifikacija normalnih zapisa (eng. *„true negative rate“*) ukazuje na omjer točno identificiranih normalnih zapisa u odnosu na ukupan broj identificiranih normalnih zapisa. Izračunava se prema sljedećoj formuli.

$$True\ negative\ rate = \frac{TN}{TN + FP}$$

Točnost pojedinog modela (eng. *„accuracy“*) prikazuje omjer točnih identifikacija zapisa modela u odnosu na ukupan broj identifikacija zapisa. Još se naziva i točnost detekcije (eng. *„detection accuracy“*). Dobiva se prema sljedećoj formuli.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

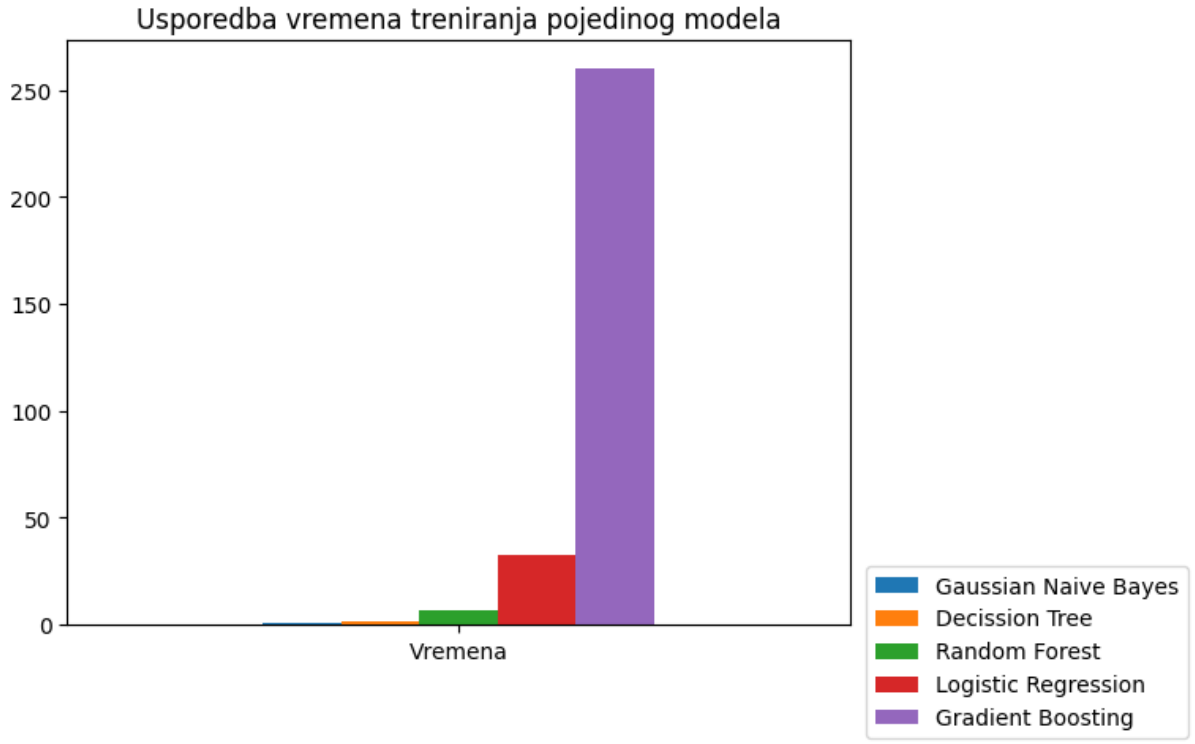
F-mjera (eng. *„F-measure“*) prikazuje nam harmonijsku sredinu prethodna dva pokazatelja, konkretno harmonijsku sredinu preciznosti i opoziva. Drugim riječima

izračunavamo preciznost modela na temelju tih dvaju pokazatelja. Dano je sljedećom formulom.

$$F - measure = 2(\frac{Precision * Recall}{Precision + Recall})$$

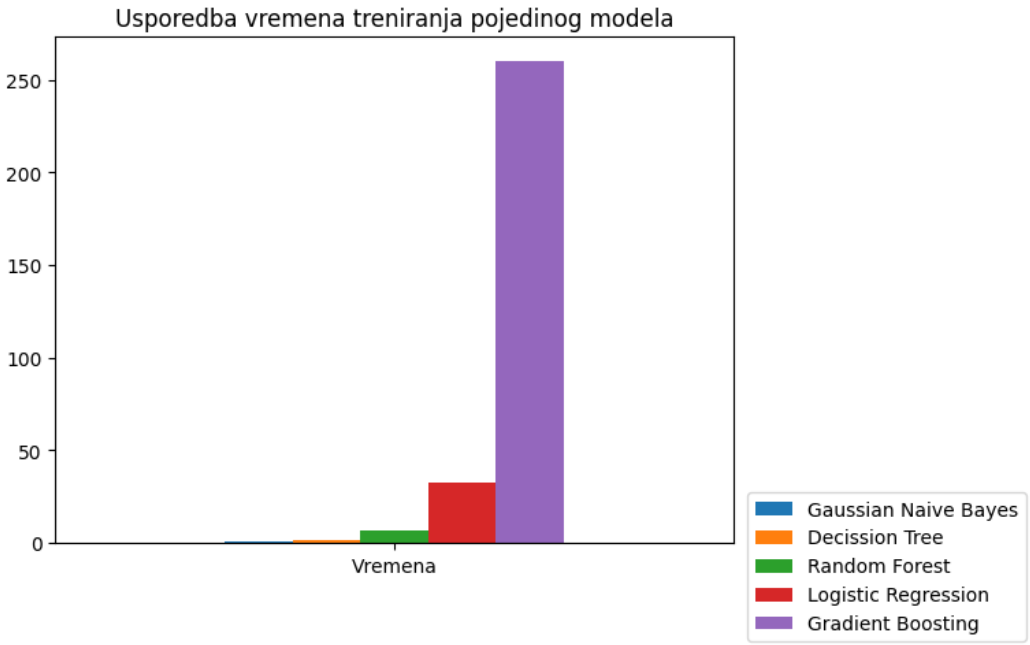
7.2. Statistička analiza implementacije

Sada kada smo opisali pojedine pokazatelje pogledajmo kakve smo podatke dobili iz implementacije metoda strojnog učenja. Uz pokazatelje uspješnosti modela iz prethodnog poglavlja biti će prikazana i vremena koja su bila potrebna za treniranje i testiranje pojedinog modela.



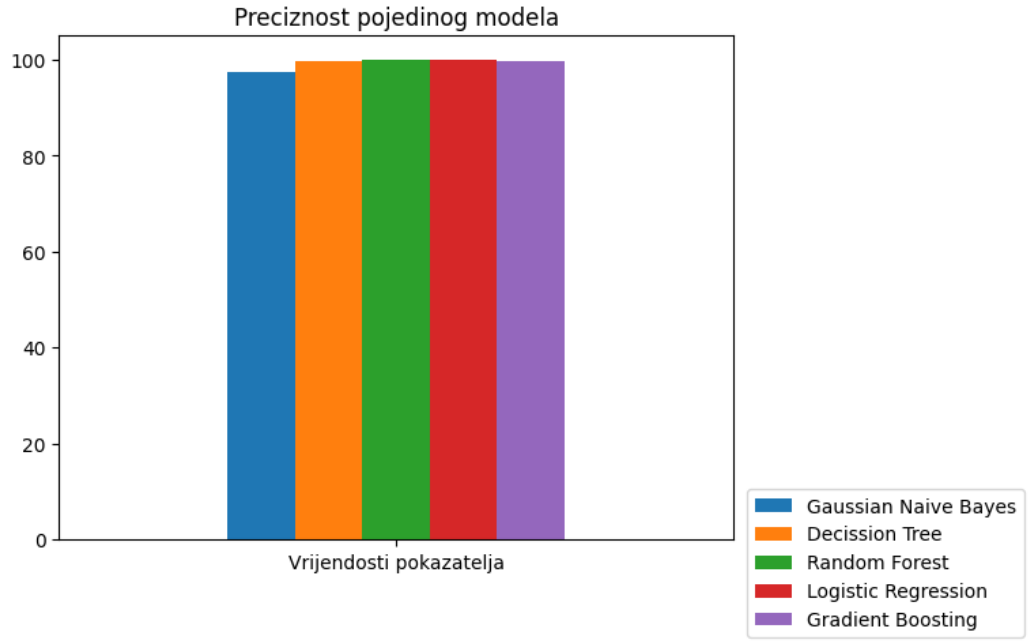
Slika 8. Usporedba vremena treniranja pojedinog modela (Autorski rad)

Usporedbom vremena treniranja moguće je zaključiti kako je model strojnog učenja koji koristi Gaussian Naive Bayes metodu klasifikacije najbrži za treniranje dok je model s metodom Gradient Boosting najsporiji. Uvedimo sada jednostavni bodovni sustav kako bi kasnije mogli procijeniti koji je model strojnog učenja naj zadovoljavajući za ovakvu vrstu primjene. S obzirom kako je ovdje što manje vrijeme bolje reći ćemo da će redom pojavljivanja modeli strojnog učenja dobiti bodove od 5 do 1.



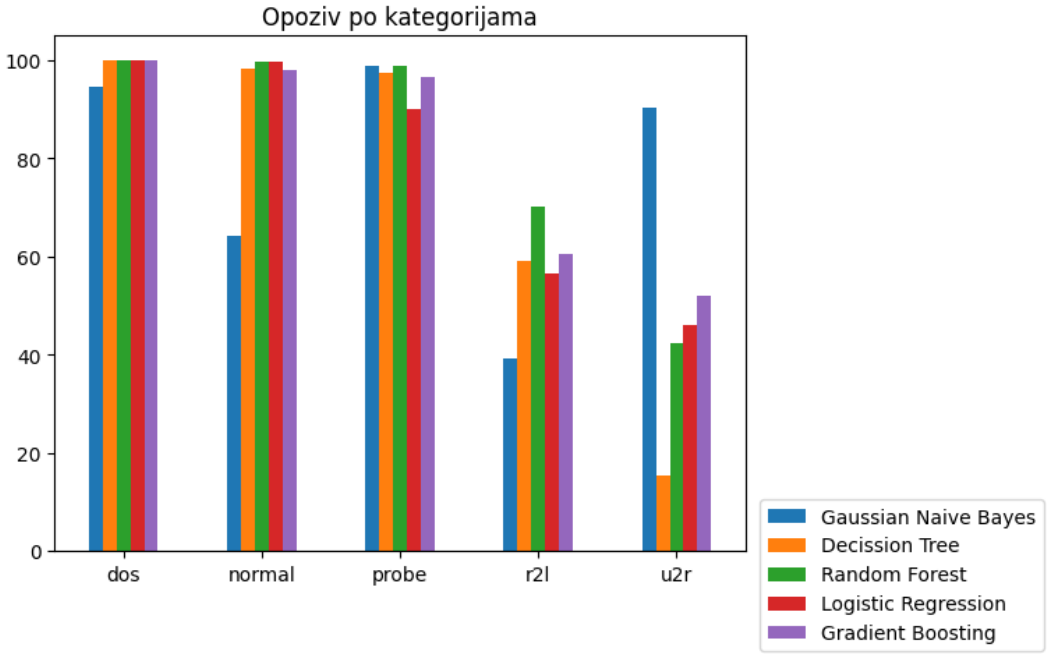
Slika 9. Usporedba vremena treniranja pojedinog modela (Autorski rad)

Slično kao i s vremenom treniranja modela ovdje nas prati isti trend s vremenom testiranja pa prema tome modeli strojnog učenja ponovno dobivaju ocjene od 5 do 1.



Slika 10. Preciznost pojedinog modela (Autorski rad)

Preciznost modela nam ukazuje na postotak korektno identificiranih zapisa skupa podataka u odnosu na ukupan zbroj zapisa. Redom pojavljivanja u grafu modeli strojnog učenja su postigli sljedeće postotke: 97.22%, 99.74%, 99.88%, 99.79% i 99.66%. Kako je cilj postići sto veću preciznost modela, redoslijedom pojavljivanja modeli učenja dobivaju sljedeće bodove: 1, 3, 5, 4, 2.



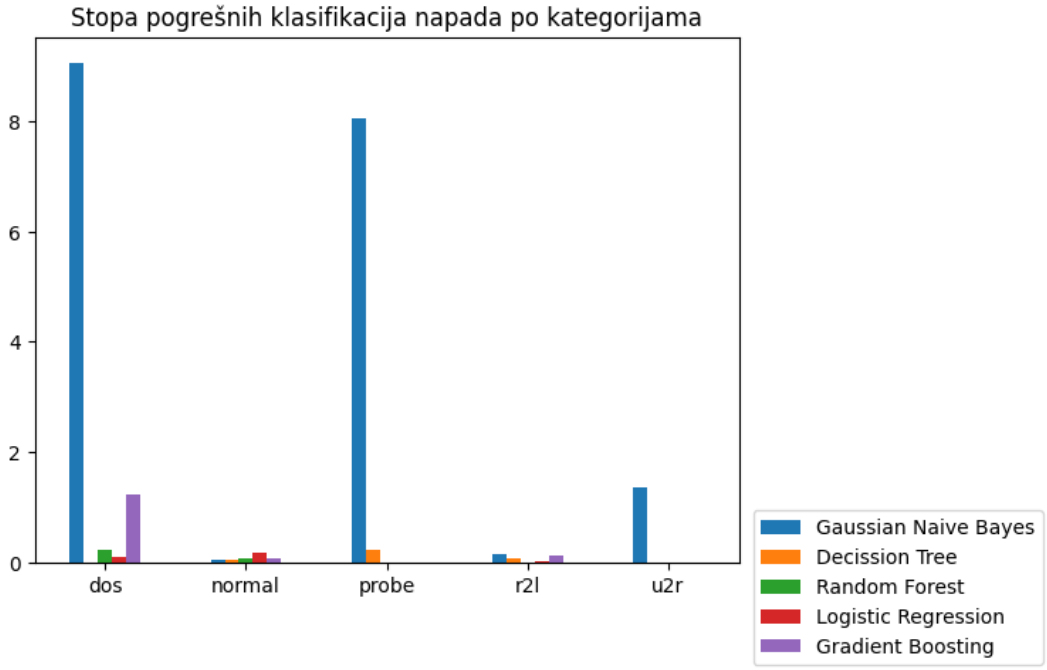
Slika 11. Vrijednosti opoziva pojedine kategorije po modelima (Autorski rad)

Opoziv modela strojnog učenja je pokazatelj, kao što je prethodno spomenuto, koji nam govori o postotku ispravno predviđenih zapisao u odnosu na ukupan zbroj korektnih zapisa. Kako skup podataka posjeduje četiri različite kategorije napada i oznaku za normalan zapis izračunavamo ovaj pokazatelj za svaku kategoriju svakog modela. Cilj je postići što veći postotak koji u prijevodu znači da naš model ispravno klasificira različite zapise. Kako bi mogli pojedinom modelu na temelju ovog pokazatelja pridodati bodove za konačnu procjenu, poslužimo se jednostavnom aritmetičkom sredinom kao mjerom usporedbe različitih modela.

Aritmetička sredina je dana formulom:

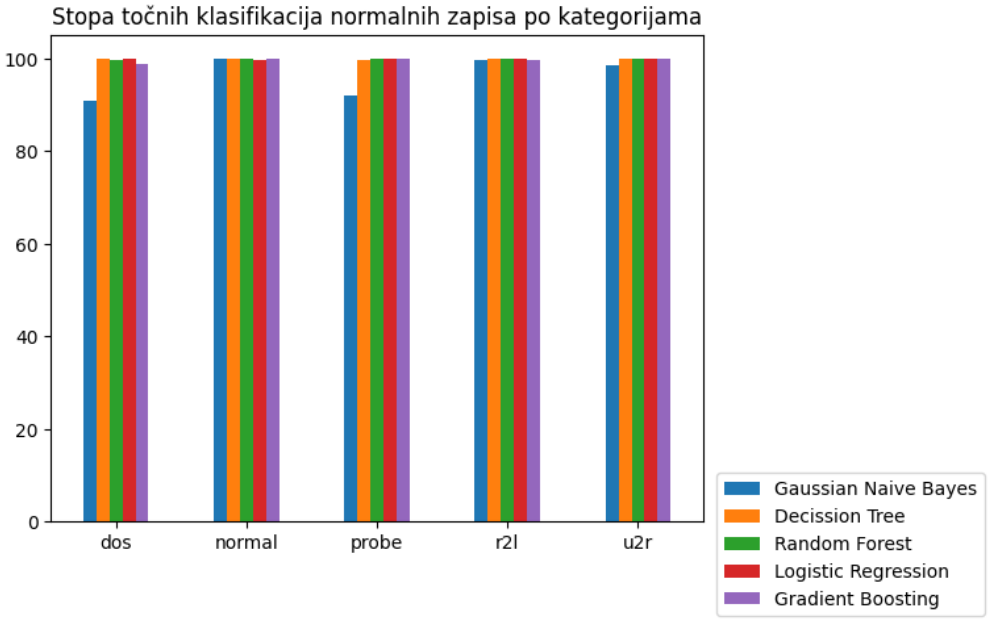
$$\bar{X} = \frac{X_1 + \cdots + X_n}{n}$$

Koristeći navedenu formulu redoslijedom pojavljivanja metoda dobivamo sljedeće rezultate: 77.52%, 74.11%, 82.19%, 78.48% i 81.47%. Prema navedenome modeli dobivaju sljedeće bodove: 2, 1, 5, 3, 4.



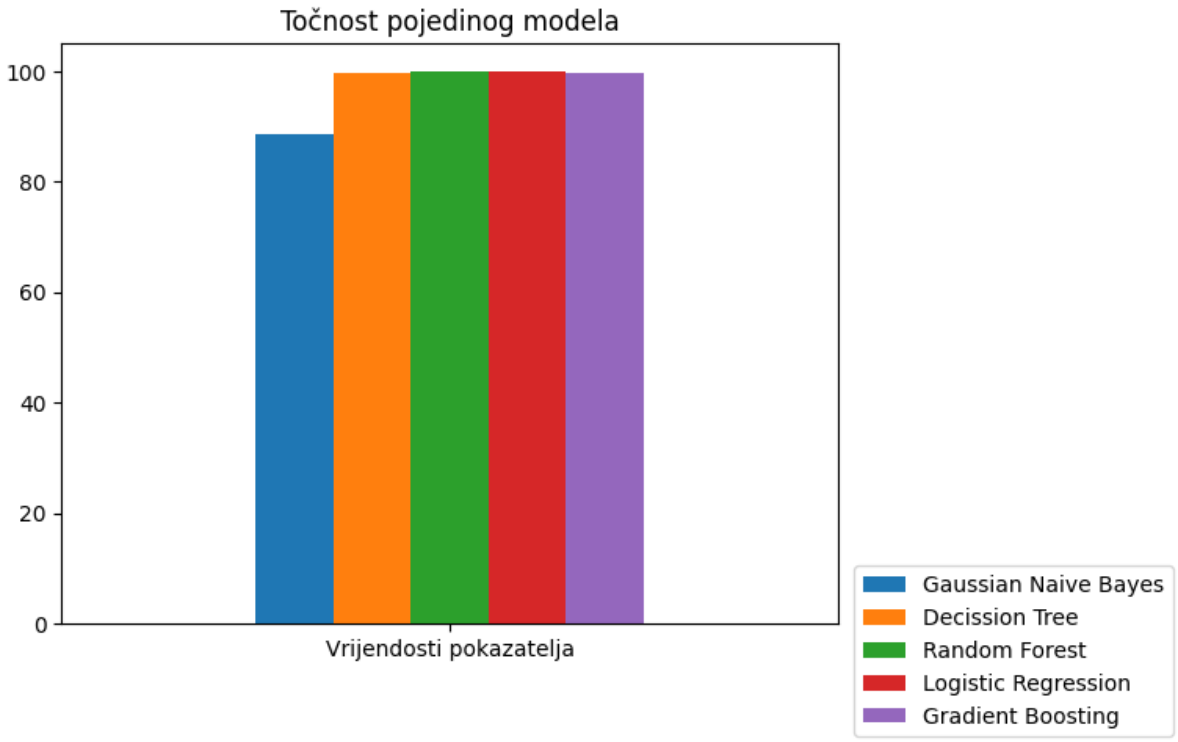
Slika 12. Stopa pogrešnih klasifikacija napada po kategorijama napada za pojedini model (Autorski rad)

Ovaj pokazatelj govori o postotku pogrešno identificiranih zapisa kao napad u odnosu na zapise koji nisu klasificirani kao napad. Drugim riječima, pokazatelj prikazuje postotak pogrešne identifikacije zapisa kao napad za pojedini model strojnog učenja. Koristeći aritmetičku sredinu za raspored rezultata dobivamo sljedeće postotke: 3.73%, 0.07%, 0.064%, 0.066% i 0.28%. Kako je ovdje cilj što manji postotak modelima redoslijedom pojavljivanja dajemo sljedeće bodove: 1, 2, 4, 3, 5.



Slika 13. Stopa točnih klasifikacija normalnih zapisa po kategorijama napada za pojedini model (Autorski rad)

Trenutni pokazatelj nam govori postotak korektno identificiranih normalnih zapisa u odnosu na ukupan zbroj normalnih zapisa. Drugim riječima, u kojem će postotku pojedini model strojnog učenja klasificirati pojedini zapis kao normalan. Aritmetičkom sredinom dobivamo sljedeće vrijednosti: 96.26%, 99.92%, 99.935%, 99.933% i 99.71%. Cilj je što veći postotak pa prema tome modeli dobivaju sljedeće bodove: 1, 3, 5, 4, 2.

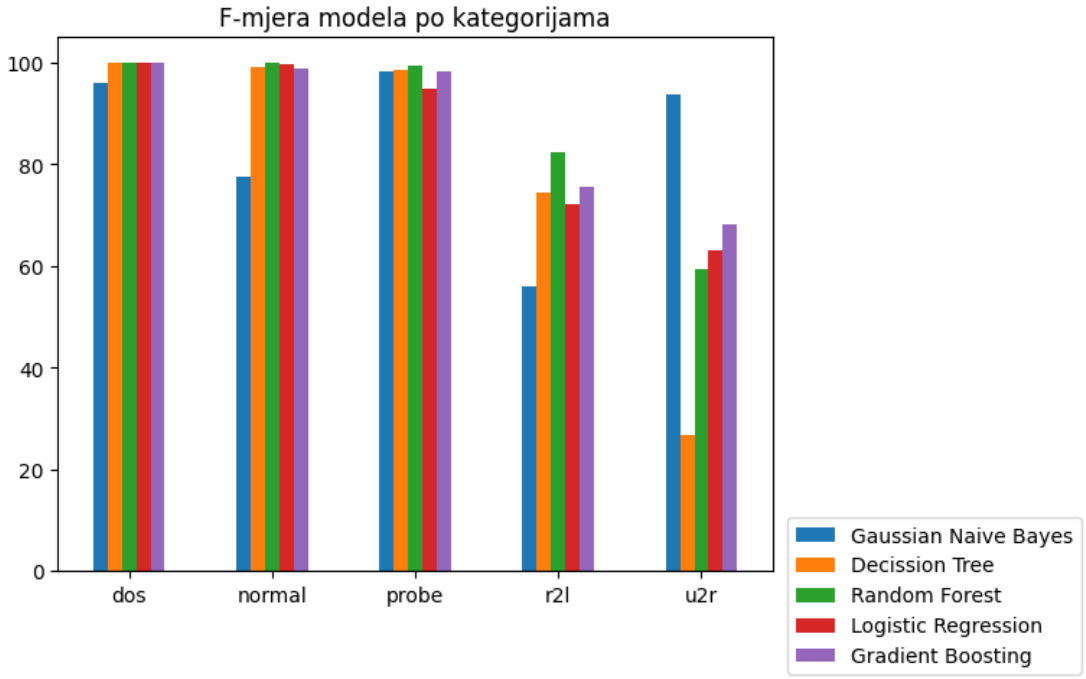


Slika 14. Točnost pojedinog modela (Autorski rad)

Pokazatelj točnosti modela strojnog učenja ukazuje na postotak ispravno identificiranih zapisa skupa podataka u odnosu na ukupan zbroj svih identifikacija modela strojnog učenja. Cilj modela jest postići što veći postotak što u prijevodi znači da model ispravno klasificira zapise iz skupa podataka. Postignute vrijednosti pojedinog modela, redoslijedom pojavljivanja, su sljedeće: 88.57%, 99.64%, 99.88%, 99.79% i 99.55%. Dodajmo sada bodove pojedinom modelu: 1, 3, 5, 4, 2.

Preostao nam je još samo jedan pokazatelj za analizirati. F-mjera modela strojnog učenja je još jedan način prikazivanja točnosti predviđanja modela. U odnosu na prethodni izračun točnosti ovaj način izračuna uzima u obzir preciznost i opoziv modela kako bi se izračunala točnost. Cilj je postići što veći postotak.

Pogledajmo izgled grafa i dobivene vrijednosti pokazatelja.



Slika 15. Izračun F-mjere modela po kategorijama napada (Autorski rad)

Dobivene vrijednosti pokazatelja su sljedeće: 84.21%, 79.71%, 88.18%, 85.92% i 88.10%. Kao i u prethodnim analizama dodajemo sljedeće bodove: 2, 1, 5, 3, 4.

Nakon što smo analizirali sve pokazatelje prikažimo bodovno stanje pojedinog modela strojnog učenja tablicom kao i konačne sume bodova.

Tablica 6. Bodovanje korištenih modela strojnog učenja

Naziv pokazatelja	Gaussian Naive Bayes	Decision tree	Random forest	Logistic regression	Gradient boosting
Vrijeme treniranja	5	4	3	2	1
Vrijeme testiranja	5	4	3	2	1
Preciznost	1	3	5	4	2
Opoziv	2	1	5	3	4
FPR	1	2	4	3	5
TNR	1	3	5	4	2
Točnost	1	3	5	4	2
F-mjera	2	1	5	3	4
Suma bodova	18	21	35	25	21

(Izvor: Autorski rad)

8. Zaključak

Kroz ovaj rad bilo je potrebno prikazati primjenu metoda strojnog učenja za otkrivanje upada u sustav na temelju zapisa komunikacije računalne mreže. Kroz poglavlja rada pojašnjeno je pozicioniranje ovakvog sustava unutar računalne mreže, način na koji radi kao i poticaj za korištenje ovakvog sustava. Također, napravljena je i kratka digresija na ostale načine zaštite računalne mreže.

Kako bi implementacija bila jasnija objašnjen je pojam strojnog učenja, navedeno je nekoliko primjera primjene iz stvarnog svijeta te su prikazane vrste strojnog učenja koje postoje danas. Za implementaciju je odabrana nadzirana vrsta strojnog učenja s obzirom kako je ovdje zadatak strojnog modela bio klasificirati ulazne podatke, odnosno zapise komunikacije računalne mreže. Za nadziranu vrstu strojnog učenja razvijen je broj različitih algoritama na temelju kojih model strojnog učenja obrađuje podatke. Za implementaciju sustava za otkrivanje upada korišteni su algoritmi Gaussian Naive Bayes, Decision tree, Random forest, Logistic regression i Gradient boosting. Kako bi navedene algoritme upotrijebili u programskom kodu korištena je biblioteka „scikit-learn“ koja služi za rad sa strojnim učenjem unutar programskog jezika „Python“.

Kako bi mogli procijeniti uspješnost pojedinog modela, odnosno algoritma klasifikacije strojnog učenja potrebno je bilo izračunati opće pokazatelje uspješnosti modela. Uzimajući u obzir činjenicu da su podaci pokazatelja uspješnosti postoci uveli smo jednostavnu bodovnu skalu kako bi mogli rangirati pojedini implementirani model.

Analizirajući grafove pojedinih pokazatelja uspješnosti modela kao i postavljenu bodovnu skalu dolazimo do zaključka kako je model strojnog učenja koji koristi Random forest algoritam klasifikacije bio najuspješniji u obradi podataka dok je najgori model bio onaj koji koristi Gaussain Naive Bayes algoritam klasifikacije podataka. Uz postavljenu bodovnu skalu valja istaknuti još jedan pokazatelj uspješnosti modela prema kojem bi se također moglo rangirati modele strojnog učenja. Pokazatelj točnosti modela je pokazatelj koji govori o postotku točnih klasifikacija ulaznih podataka. Prema točnosti modela Random forest algoritam klasifikacije je ponovno ispao najbolji, odnosno najtočniji u predviđanju zapisa ulaznih podataka dok je Gaussian Naive Bayes ponovno neprecizan u predviđanju zapisa ulaznog skupa podataka.

Popis literature

Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, *32*(1), e4150. https://doi.org/10.1002/ett.4150

Alamiedy, T. A., Anbar, M., Alqattan, Z. N. M., & Alzubi, Q. M. (2020). Anomaly-based intrusion detection system using multi-objective grey wolf optimisation algorithm. *Journal of Ambient Intelligence and Humanized Computing*, *11*(9), 3735–3756. https://doi.org/10.1007/s12652-019-01569-8

Confusion Matrix in Machine Learning. (2017, listopad 15). *GeeksforGeeks*. https://www.geeksforgeeks.org/confusion-matrix-machine-learning/

Decision Tree Algorithm in Machine Learning—Javatpoint. (bez dat.). Www.Javatpoint.Com. Preuzeto 25. kolovoz 2023., od https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

Difference between HIDs and NIDs. (2022, lipanj 30). *GeeksforGeeks*. https://www.geeksforgeeks.org/difference-between-hids-and-nids/

Gradient Descent in Machine Learning—Javatpoint. (bez dat.). Www.Javatpoint.Com. Preuzeto 28. kolovoz 2023., od https://www.javatpoint.com/gradient-descent-in-machine-learning

Hartley, J. (bez dat.). *colorama: Cross-platform colored terminal text.* (0.4.6) [Python; OS Independent]. Preuzeto 28. kolovoz 2023., od https://github.com/tartley/colorama

Introduction to Python. (bez dat.). Preuzeto 27. kolovoz 2023., od https://www.w3schools.com/python/python_intro.asp

James P Anderson. (1980). *Computer Security Threat Monitoring and Survaillance*.

KDD Cup 1999 Data. (bez dat.). Preuzeto 27. kolovoz 2023., od http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity*, *2*(1), 20. https://doi.org/10.1186/s42400-019-0038-7

Kumar, V., & Sangwan, D. O. P. (2012). Signature Based Intrusion Detection System Using SNORT. *International Journal of Computer Applications*.

Labib, K., & Vemuri, V. R. (bez dat.). *Detecting Denial-of-Service And Network Probe Attacks Using Principal Component Analysis*.

Liao, H.-J., Richard Lin, C.-H., Lin, Y.-C., & Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, *36*(1), 16–24. https://doi.org/10.1016/j.jnca.2012.09.004

Logistic Regression in Machine Learning—Javatpoint. (bez dat.). Preuzeto 28. kolovoz 2023., od https://www.javatpoint.com/logistic-regression-in-machine-learning

Machine learning, explained | MIT Sloan. (2023, kolovoz 10). https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained

Machine Learning Random Forest Algorithm—Javatpoint. (bez dat.). Www.Javatpoint.Com. Preuzeto 27. kolovoz 2023., od https://www.javatpoint.com/machine-learning-random-forest-algorithm

Machine Learning: What It is, Tutorial, Definition, Types - Javatpoint. (bez dat.). Www.Javatpoint.Com. Preuzeto 14. kolovoz 2023., od https://www.javatpoint.com/machine-learning

NumPy—About Us. (bez dat.). Preuzeto 28. kolovoz 2023., od https://numpy.org/about/

pandas—Python Data Analysis Library. (bez dat.). Preuzeto 28. kolovoz 2023., od https://pandas.pydata.org/about/index.html

Patidar, P. (2023, ožujak 1). *Classification using Gaussian Naive Bayes from scratch*. Medium. https://levelup.gitconnected.com/classification-using-gaussian-naive-bayes-from-scratch-6b8ebe830266

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*(85), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

Real-World Examples of Machine Learning (ML) | Tableau. (bez dat.). Preuzeto 25. kolovoz 2023., od https://www.tableau.com/learn/articles/machine-learning-examples

Rebecca Bace & Peter Mell. (2001). *NIST Special Publication on Intrusion Detection Systems* (str. 52). https://apps.dtic.mil/sti/citations/ADA393326

Research scholor,Department of CSEUniversity institute of technologyAffiliated toRgpv, & Gupta, P. (2020). An Efficient Classifier for U2R, R2L, DoS Attack. *International Journal of Recent Technology and Engineering (IJRTE)*, *9*(1), 644–647. https://doi.org/10.35940/ijrte.A1942.059120

Tarter, A. (2017). Importance of Cyber Security. U P. S. Bayerl, R. Karlović, B. Akhgar, & G. Markarian (Ur.), *Community Policing—A European Perspective: Strategies, Best Practices and Guidelines* (str. 213–230). Springer International Publishing. https://doi.org/10.1007/978-3-319-53396-4_15

Types of Firewall—Javatpoint. (bez dat.). Www.Javatpoint.Com. Preuzeto 25. kolovoz 2023., od https://www.javatpoint.com/types-of-firewall

Types of Machine Learning—Javatpoint. (bez dat.). Www.Javatpoint.Com. Preuzeto 25. kolovoz 2023., od https://www.javatpoint.com/types-of-machine-learning

Understanding Anti-Virus Software | CISA. (2009, lipanj 30). https://www.cisa.gov/news-events/news/understanding-anti-virus-software

Understanding Denial-of-Service Attacks | CISA. (2021, veljača 1). https://www.cisa.gov/news-events/news/understanding-denial-service-attacks

What Is a Firewall? (bez dat.). Cisco. Preuzeto 14. kolovoz 2023., od https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html

Woke, G. (2023, ožujak 24). The difference between libraries and frameworks. *Simple Talk*. https://www.red-gate.com/simple-talk/development/other-development/the-difference-between-libraries-and-frameworks/

Popis slika

Popis tablica