Izrada modela za predikciju sigurnosnih napada na internetu

Petanović, Alen

Undergraduate thesis / Završni rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Polytechnic of Međimurje in Čakovec / Međimursko veleučilište u Čakovcu**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:110:500624

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-05**

Repository / Repozitorij:

Polytechnic of Međimurje in Čakovec Repository - Polytechnic of Međimurje Undergraduate and Graduate Theses Repository

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
STRUČNI STUDIJ RAČUNARSTVO

ALEN PETANOVIĆ

IZRADA MODELA ZA PREDIKCIJU SIGURNOSNIH NAPADA NA INTERNETU

ZAVRŠNI RAD

Mentor:
mr. sc. Željko Knok, v. pred.

ČAKOVEC, 2023.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI STUDIJ RAČUNARSTVO

ALEN PETANOVIĆ

IZRADA MODELA ZA PREDIKCIJU SIGURNOSNIH NAPADA NA INTERNETU

DEVELOPMENT OF A PREDICTION MODEL FOR SECURITY ATTACKS ON THE INTERNET

ZAVRŠNI RAD

Mentor:

mr. sc. Željko Knok, v. pred.

ČAKOVEC, 2023.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
ODBOR ZA ZAVRŠNI RAD

Čakovec, 7. veljače 2020.

država: **Republika Hrvatska**
Predmet: **Baze podataka II-izborni**
Polje: **2.09 Računarstvo**

ZAVRŠNI ZADATAK br. 2019-RAČ-R-132

Pristupnik: **Alen Petanović (0313020384)**
Studij: redovni preddiplomski stručni studij Računarstvo
Smjer: Programsko inženjerstvo

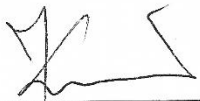
Zadatak: **Izrada modela za predikciju sigurnosnih napada na internetu**

Opis zadatka:

Koristeći metode UI izraditi model koji će biti od pomoći za dijagnozu oboljenja. Pri izradi koristiti programski alat: Python u okruženu Anaconda.

Zadatak uručen pristupniku: 7. veljače 2020.
Rok za predaju rada: 20. rujna 2020.

Mentor:



mr. sc. Željko Knok, v. pred.

Predsjednik povjerenstva za završni ispit:

ZAHVALA

Zahvalio bih mentoru, profesoru Knoku na strpljenju i pomoći s izradom završnoga rada.

Alen Petanović

Sažetak

Sigurnosni napad je neovlašteni pokušaj krađe, oštećenja ili izlaganja podataka iz informacijskog sustava. Izvoditelji tih sigurnosnih napada oslanjaju se na sve sofisticiraniju tehnologiju i svakim danom raste broj novih vrsta cyber napada. Nekad su se sigurnosni napadi mogli izbjeći. U današnje vrijeme, s internetskom vezom dostupnom svugdje i kako sve više softvera zahtijeva internetsku vezu, više nije pitanje hoće li se dogoditi sigurnosna prijetnja, nego je pitanje vremena kada će se dogoditi narušavanje online sigurnosti. Danas postoji softver za otkrivanje prijetnji (eng. Intrusion Detection System, IDS) koji nadgleda mrežni promet i traži poznate prijetnje i sumnjive ili zlonamjerne aktivnosti, a dolazi u nizu različitih vrsta i mogućnosti.

Zadatak ovog završnog rada jest izraditi model koji, uz pomoć učenja na određenom dijelu podataka, može prediktirati je li ili nije riječ o sigurnosnom napadu.. Rješenje je implementirano binarnom i višeklasnom klasifikacijom. Algoritmi korišteni za predikciju su Gaussov Naive Bayes algoritam, K-sljedećih susjeda i stablo odlučivanja. Sam programski kod je pisan u Python programskom jeziku. Kao razvojno okruženje koriste se Jupyter Notebook i Anaconda IDE. Za procese nad podatcima korišteni su NumPy, Pandas, Scikit-learn i Imblearn. Za vizualne prikaze koriste se Matplotlib i Seaborn.

U uvodnom dijelu opisuje se utjecaj strojnog učenja i sigurnosnih napada u današnje vrijeme. U teoretskom dijelu ovog rada ukratko su objašnjeni Python programski jezik, razvojno okruženje Anaconda i Jupyter Notebook te individualne biblioteke potrebne za procese nad podatcima i izradu modela. Također je objašnjena teorija iz strojnog učenja i osnovna podjela te neki od popularnijih algoritama za učenje. Opisani su algoritmi korišteni za predikciju. Nakon toga opisana je procedura izrade prediktivnog modela koja se sastoji od validacije podataka (prikupljanje, unos, čišćenje, analiza, vizualiziranje, transformacija), treniranja podataka te na kraju implementacija na setu za testiranje uz prikaze performansi s klasifikacijskim izvještajem i konfuzijskim matricama.

Ključne riječi: ***Python, Anaconda, Jupyter Notebook, Pandas, Scikit-learn, Seaborn***

Sadržaj

1. Uvod 1
2. Korišteni alati i biblioteke 2
   2.1. Python 2
   2.2. Razvojno okruženje 3
      2.2.1. Anaconda IDE 3
      2.2.2. Jupyter Notebook 5
   2.3. Biblioteke 5
      2.3.1. Pandas 5
      2.3.2. Matplotlib 5
      2.3.3. Numpy 6
      2.3.4. Scikit-learn 6
      2.3.5. Seaborn 6
      2.3.6. Imbalanced-learn 6
3. Strojno učenje 7
   3.1. Nadzirano učenje 7
   3.2. Nenadzirano učenje 8
   3.3. Podržano učenje 9
4. Algoritmi 10
   4.1. Regresija 10
   4.2. Klasifikacija 11
   4.3. Korišteni algoritmi 11
      4.3.1. Stablo odlučivanja 11
      4.3.2. Naive Bayes 12
      4.3.3. K-sljedećih susjeda 13
5. Izrada modela 15
   5.1. Podatci 15
   5.2. Validacija podataka 15
      5.2.1. Unos podataka 15
      5.2.2. Čišćenje podataka 20
      5.2.3. Eksplorativna analiza i vizualiziranje podataka 23
      5.2.4. Transformacija podataka 33

6. Prediktivni modeli 37
6.1. Binarna klasifikacija 37
6.1.1. Gaussov Naive Bayes 40
6.1.2. Stablo odlučivanja 42
6.1.3. K-sljedećih susjeda 42
6.2. Višeklasna klasifikacija 46
6.2.1. Gaussov Naive Bayes 51
6.2.2. Stablo odlučivanja 52
6.2.3. K-sljedećih susjeda 53
7. Zaključak 57
8. Literatura 58
9. Popis slika 59

1. Uvod

Umjetna inteligencija i strojno učenje smatraju se jednim od najvećih inovacija još od izrade mikročipova. Nekad su to bili koncepti ostvarivi samo u znanstveno-fantastičnim filmovima. Danas su prisutni svugdje. Prepoznavanje lica na mobitelima, softver za prevođenje teksta (Google Prevoditelj) i *spam/ham* samo su neki od primjera implementacije strojnog učenja, a postaju i sve popularniji u medicini gdje mogu biti od velike pomoći, spasiti živote i drastično smanjiti mogućnost ljudske greške.

Ljudi danas u prosjeku provode šest sati dnevno na internetu. Prosječni korisnik nije svjestan opasnosti na internetu. Sa sve većom dostupnosti internetske veze i sve većim brojem aplikacija koje ju zahtijevaju vrlo je lako doći u kontakt sa zlućudnim softverom. Hakeri danas postaju sve inteligentniji. Rade kompleksnije softvere i nalaze nove načine za izvođenje krađa. Svatko se na neki način susreo s takvim prijetnjama. Najčešće je riječ o *phishing* mailovima.

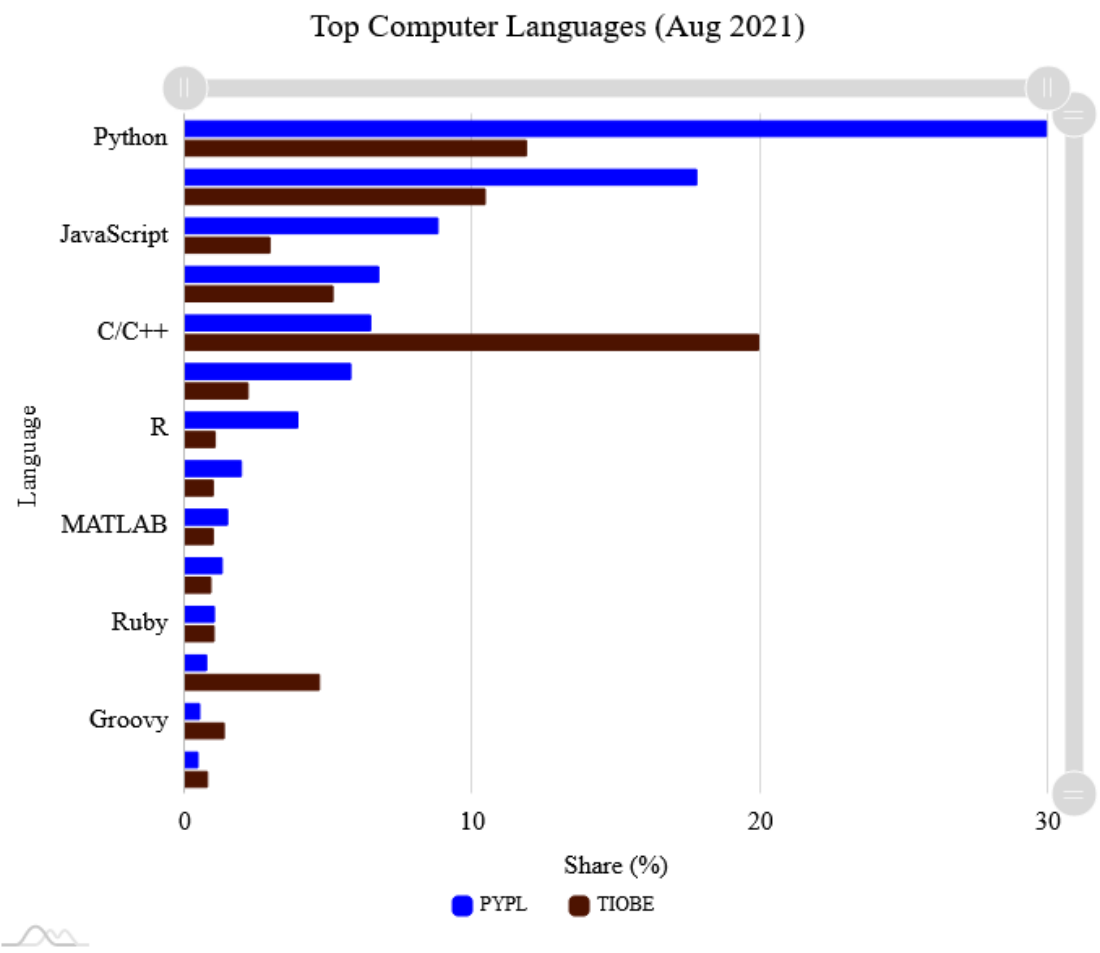
Danas postoje softveri za odbijanje takvih prijetnji. To su takozvani sustavi za otkrivanje upada (eng. *Intrusion Detection System*, IDS). Glavna funkcija im je tijekom očitanja sumnjive aktivnosti poslati upozorenje administratoru, a neki softveri i automatski poduzimaju mjere obrane.

2. Korišteni alati i biblioteke

U ovom poglavlju nalaze se opisi korištenih alata. Naglasak je na Python programskom jeziku u kojem je pisano programsko rješenje. Osim Pythona opisano je razvojno okruženje Anaconda i Jupyter Notebook, biblioteke potrebne za procese nad podatcima te također biblioteke koje na grafički način prikazuju podatke.

2.1. Python

Python je interaktivni, *open-source* programski jezik. Može se koristiti kao objektno-orijentirani i orijentirani jezik. [1] Univerzalan je što se tiče upotrebe – koristi se u analizama podataka, izradi *desktop* i *web*-aplikacija i u video igrama. Ono što ga još čini posebnim jest to što je dobar za automatiziranje procesa. Jedna je od najvećih prednosti Pythona njegova brzina. Postaje sve popularniji među programerima zbog svoje jednostavnosti i lagane čitljivosti. Također popularnosti doprinosi i strojno učenje koje postaje sve popularnije u poslovanju. Budući da dosta programskih jezika koristi točku sa zarezom za odvajanje linija koda, mnogi programeri imaju probleme u tranziciji s drugih jezika na Python jer strukturiranje u Pythonu funkcionira na principu uvlačenja koda (uvlačenje govori Python interpreteru da ta linija pripada određenom bloku koda). Neka od danas najpopularnijih *web*-sjedišta/aplikacija pisana su u Pythonu, npr. Instagram, Spotify, Netflix, Pinterest itd. Trenutna verzija Pythona je 3.9, ali je preporučljivo koristiti 3.7 zbog kompatibilnosti s ostalim alatima.



Slika 1. Najpopularniji jezici u kolovozu 2021.

Izvor: https://statisticstimes.com/tech/top-computer-languages.php

2.2. Razvojno okruženje

2.2.1. Anaconda IDE

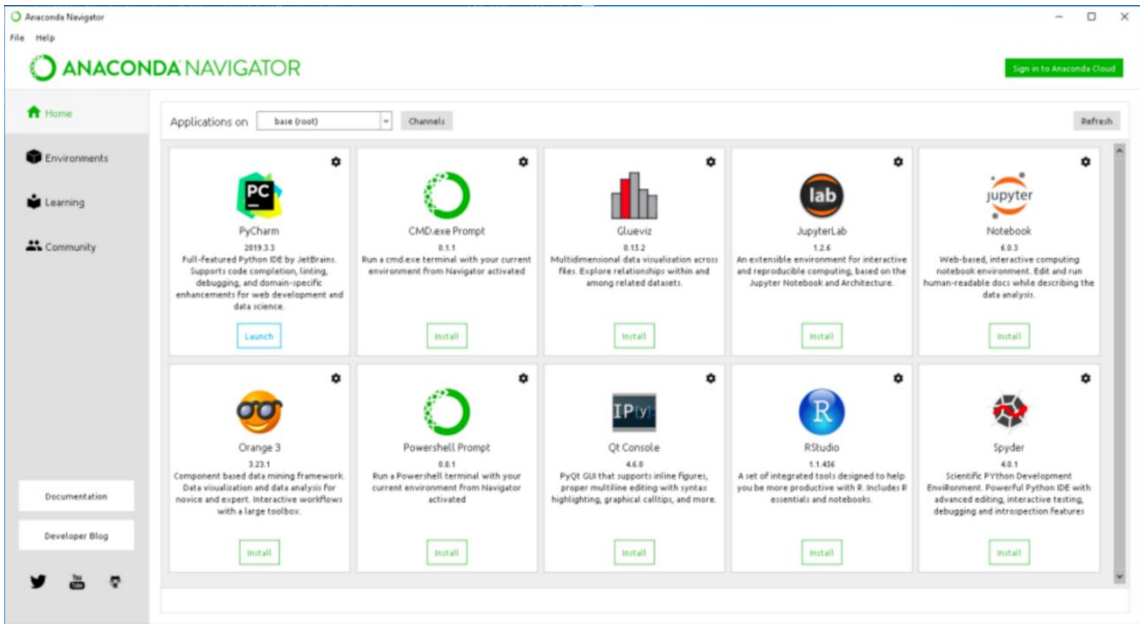
Anaconda je platforma za strojno učenje i analizu podataka koja dolazi u više verzija: *Individual*, *Team*, *Enterprise* i *Cloud*, svaka sa svojim prednostima i zabranama. Dostupna je na Windows, Linux i macOS operacijskim sustavima. Trenutno ima više od dvadeset milijuna korisnika diljem svijeta. U Anacondi je moguće pisati rješenja u Python i R programskim jezicima i sadrži nekoliko tisuća dostupnih *open-source* biblioteka.

Neke od osnovnih biblioteka za rad koje dolaze uz osnovnu verziju Anaconda IDE-a su:

- Keras i Tensorflow
- NumPy/Pandas
- Matplotlib/Seaborn

Anaconda se može koristiti na dva načina: preko terminala (Anaconda Prompt) i preko GUI verzije (Anaconda Navigator). Korištenje GUI verzije jednostavnija je varijanta, ali korisnici češće koriste terminal za upravljanje razvojnim okruženjem. Anaconda je također dostupna u *minimal installer* obliku (instalacija samo s nužnim softverom) koji se zove Miniconda.

Anaconda radi na principu jednog ili više okruženja gdje se mogu koristiti aktualne verzije biblioteka ili starije verzije zbog kompatibilnosti. U terminalu se komponente instaliraju pomoću *pip install* ili *conda install* naredbi.



Slika 2. Sučelje Anaconda Navigatora

Izvor: Autor

2.2.2. Jupyter Notebook

Jupyter Notebook je razvojno okruženje za kod, podatke i bilježnice unutar *web*-preglednika. Bilježnica je naziv za dokument stvoren unutar Jupyter Notebook aplikacije i ima svoju posebnu ekstenziju .ipynb. Aplikacije se mogu izvršavati na lokalnom poslužitelju (eng. *Local host*) gdje se može koristiti bez pristupa internetu ili instalirati na udaljenog poslužitelja (eng. *Remote server*) u kojem slučaju zahtijeva pristup internetu.

Svaka bilježnica ima svoj *kernel* koji se automatski pali na pokretanju. *Kernel* za pokretanje Python koda zove se *ipython.* Također postoje *kerneli* za druge programske jezike. Unutar bilježnice moguće je odabrati željeno Anaconda okruženje i verziju Pythona. Jupyter ima i svoj *dashboard* gdje je moguće pregledavati lokalne datoteke, otvarati bilježnice ili gasiti pokrenuti *kernel*. [2]

2.3. Biblioteke

2.3.1. Pandas

Pandas je brz, snažan i jednostavan *open-source cross-platform* alat za analiziranje i manipuliranje podatcima te za upravljanje strukturama podataka. Objavljen je 2008. godine, a 2009. postaje *open-source*. Pisan je u Python i C programskim jezicima. Preko Pandas alata mogu se uvesti podatci iz CSV (eng. *Comma-separated values*), JSON (*JavaScript Object Notation*), SQL i Excel datoteci te takve podatke spojiti, preoblikovati i sl.

2.3.2. Matplotlib

Matplotlib biblioteka je *cross-platform* biblioteka za grafičke prikaze. Zadane podatke, između ostalog, može prikazati u obliku jednostanih grafova, dijagrama i sl. Izrađena je 2003. godine i od tada je nužna knjižica za svaki projekt u grani strojnog učenja.

2.3.3. Numpy

Numpy (*Numerical Python*) je *open-source* biblioteka za izvršavanje operacija s numeričkim vrijednostima. Izradio ju je Travis Oliphant 2005. godine. Dijelom je napisana u Pythonu za radove s poljima, a ima i funkcionalnosti za rad s linearnom algebrom, Fourierovim transformacijama i matricama. *Default* metoda u Pythonu je rad s listama koje služe kao polja, ali se one sporo obrađuju. NumPy biblioteka sadrži *ndarray* objekte koji služe kao ekvivalent Python listama, ali se procesiraju pedeset puta brže.

2.3.4. Scikit-learn

Scikit-learn je biblioteka bazirana na NumPy, SciPy i Matplotlib bibliotekama. Sadrži pregršt alata za analizu podataka poput algoritama za klasifikaciju i regresiju, preprocesiranje podataka, klasteriranje i sl. Stvorena je 2007. s ciljem da je dostupna svim korisnicima.

2.3.5. Seaborn

Seaborn je biblioteka bazirana na Matplotlib biblioteci te također služi za vizualni prikaz podataka, ali uz više izbora.

2.3.6. Imbalanced-learn

Imbalanced-learn je *open-source* biblioteka kojom se nosi s problemima neuravnoteženih podataka kod klasifikacije. Njezin se rad oslanja na scikit-learn biblioteku.

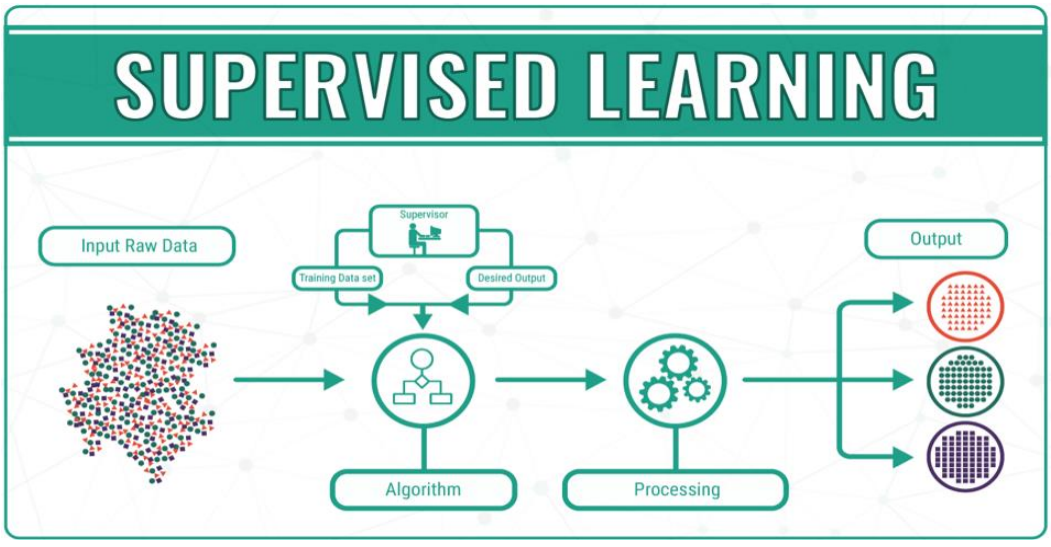
3. Strojno učenje

Strojno učenje je grana umjetne inteligencije bazirana na ideji da računala mogu učiti iz zadanih podataka, prepoznati uzorke i donijeti odluke bez ljudske intervencije. U prošlom desetljeću strojnim učenjem došlo se do raznih postignuća poput samovozećih automobila, prepoznavanja govora, predikcija internetskih pretraživanja na Googleu, sustava za predlaganje sadržaja na društvenim mrežama. Ljudi (neznajući) koriste mnoge stvari do kojih se došlo strojnim učenjem na dnevnoj bazi. Teoretski, strojno učenje može se podijeliti na tri tipa: nadzirano, nenadzirano i podržano učenje. [3]

3.1. Nadzirano učenje

Nadzirano učenje (engl. *Supervised learning*) vrsta je strojnog učenja gdje se izlazni podatak dobiva na temelju ulaznih podataka. Zove se nadzirano učenje jer simulira odnos između učitelja i učenika – učitelj promatra proces i učenik na temelju ulaznih podataka dobiva rezultat i u slučaju pogreške učitelj ispravlja pogrešku. Učenje prestaje onda kad algoritam postigne određene performanse. [4]

Vrste nadziranog učenja mogu se podijeliti na regresiju i klasifikaciju (Više o njima u poglavlju 5).



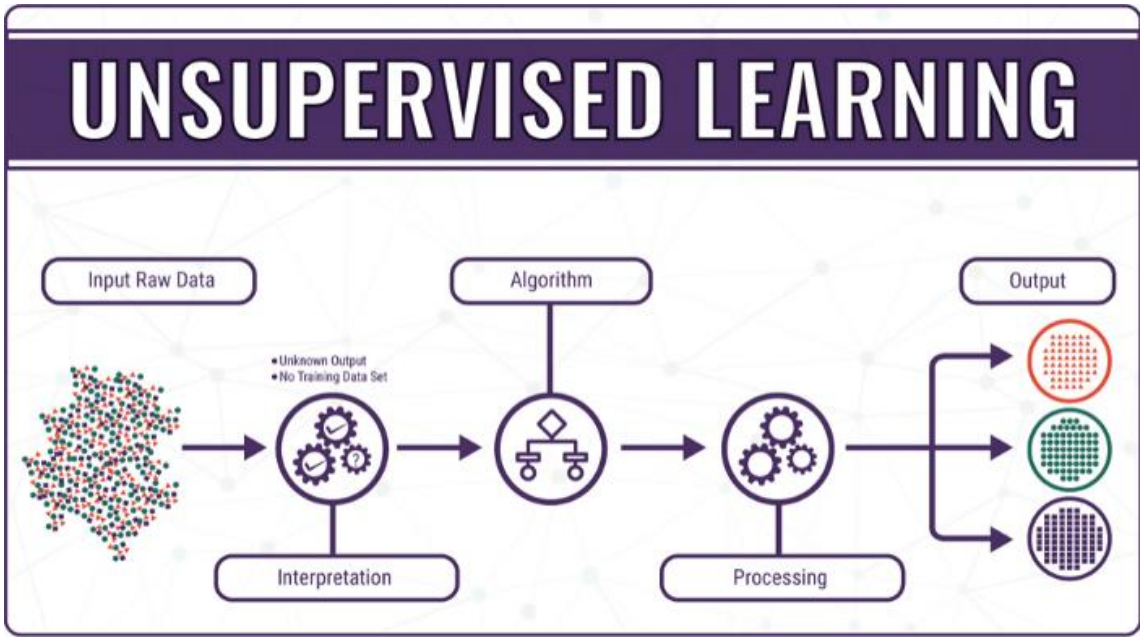
Slika 3. Nadzirano učenje

Izvor: https://miro.medium.com/max/700/0*7lHSm-TrWkbxckI9.png

3.2. Nenadzirano učenje

Nenadzirano učenje (engl. *Unsupervised learning*) vrsta je učenja gdje postoje ulazni podatci, ali nema izlaznog podatka. Rezultat nenadziranog učenja jest da se nauči više o podatcima. Zove se *nenadzirano* jer ne postoji točan ili netočan odgovor i nema učitelja. Stroj sam mora odrediti što podatci znače.

Nenadzirano učenje može se podijeliti na klasteriranje i pravilo asocijacije. Klasteriranjem se pokušava grupirati podatke prema sličnosti (npr. stroj može pretpostaviti u kojoj je regiji najveća vjerojatnost da će se dogoditi potres). Pravilo asocijacije jest pravilo kojim stroj pronalazi sličnosti u podatcima (npr. kupac koji kupuje x također kupuje y). [5]



Slika 4. Nenadzirano učenje

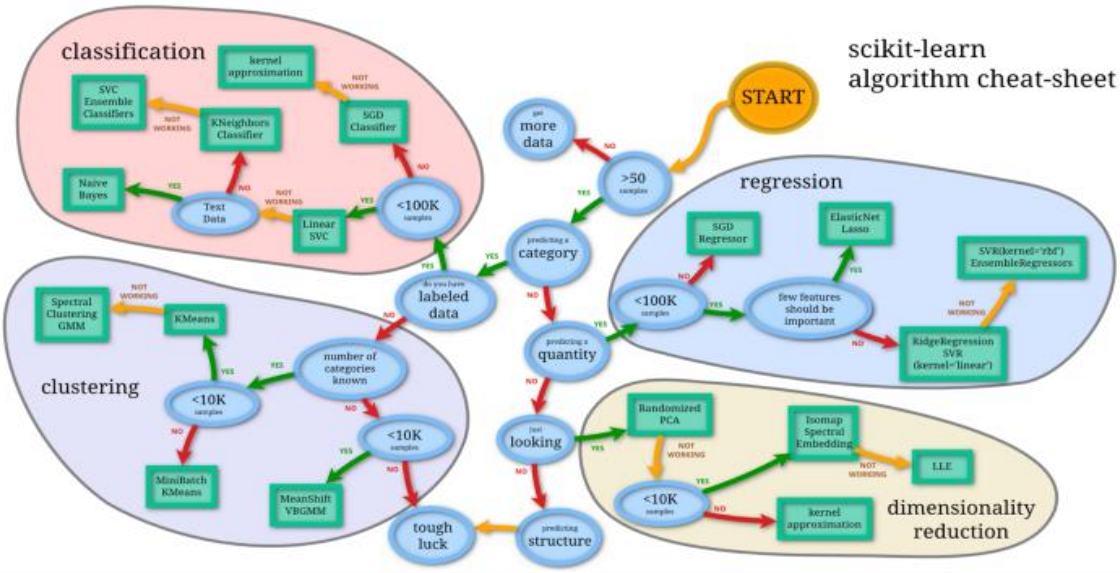
Izvor: https://miro.medium.com/max/700/0*dMW760AWCDbdQW-z.png

3.3. Podržano učenje

Podržano učenje *(engl. Reinforcement learning*) jest učenje s podrškom. Model uči tako da učitelj implementira sustav nagrađivanja i kažnjavanja. [5] Kako bi stroj postigao cilj i maksimizirao broj nagrada, mora proučiti stanja i akcije koji vode do cilja. Ovaj se tip može koristiti u robotici, kemiji, videoigrama, reklamiranju, personaliziranim preporukama, itd. [6]

4. Algoritmi

Postoji pregršt algoritama koji se mogu koristiti za izradu modela. Algoritmi se mogu grupirati ovisno o ciljnoj varijabli, vrsti podataka i broju uzoraka na raspolaganju. Iako se mogu grupirati, neki se algoritmi mogu koristiti za obje kategorije.



Slika 5. Podjela algoritama - cheat sheet

Izvor: https://1.bp.blogspot.com/-ME24ePzpzIM/UQLWTwurfXI/AAAAAAAAANw/W3EETIroA80/s1600/drop_shadows_background.png

4.1. Regresija

Regresija je proces traženja povezanosti između zavisnih i nezavisnih varijabli. Ulazne varijable u regresijskim postupcima mogu biti ili numeričke ili kategoričke, a izlazna je varijabla uvijek numerička, kontinuirana varijabla. Neki slučajevi gdje se koriste regresijski algoritmi jesu npr. prediktiranje cijena nekretnina, trendova ili vremena.

Popularniji algoritmi koji se mogu koristiti za regresiju su:

- Linearna regresija
- *Random Forest*

- K-sljedeći susjed
- Lasso regresija.

4.2. Klasifikacija

Klasifikacija je proces gdje računalo uzima u obzir ulazne varijable i prema njima odlučuje kojoj od kategorija pripada. Za razliku od regresije, izlazna varijabla u klasifikaciji diskretna je vrijednost. Klasificiranje može, naprimjer, biti sortiranje elektroničke pošte (ili je *spam* ili nije), kategoriziranje voća ili geometrijskih oblika, itd. Neki od klasifikacijskih algoritama su:

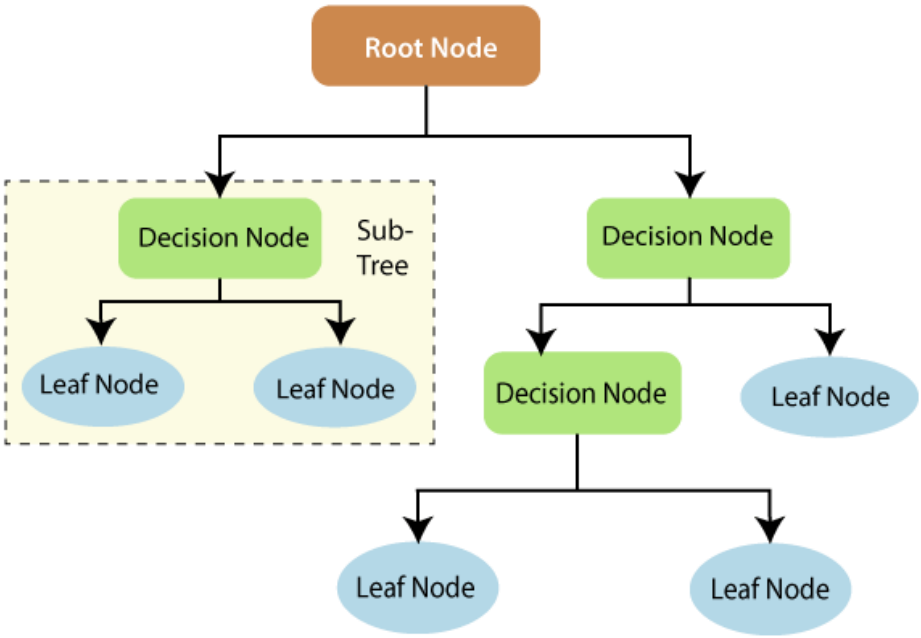
- Logistička regresija
- *Naïve Bayes*
- K-sljedeći susjed
- Stablo odlučivanja
- *Random Forest.*

4.3. Korišteni algoritmi

U ovom dijelu ukratko su opisani korišteni algoritmi Stablo odlučivanja (eng. *Decision Tree*), *Naïve Bayes* i k-sljedeći susjed.

4.3.1. Stablo odlučivanja

Stablo odlučivanja je algoritam koji se može koristiti i za regresijske i klasifikacijske modele. Ime potječe od strukture koja podsjeća na binarno stablo. Sastoji se od korijena, čvorova, listova i grana koje ih povezuju. Svaki čvor unutar stabla predstavlja jedan atribut (stupac u podatcima), a svaki list u stablu predstavlja vrijednost ciljne (izlazne) varijable ako su dane vrijednosti ulaznih varijabli predstavljene putom od korijena stabla do tog lista. Stablo se dobiva učenjem na podatcima, na način da se vrši grananje izvornog skupa podataka u podskupove na temelju testiranja vrijednosti varijabli. Taj se proces rekurzivno ponavlja, a završava kad podskup specifičnog čvora ima jednake vrijednosti kao i izlazna varijabla ili ako grananje dodatno ne poboljšava rezultat. [7]



-

Slika 6. Stablo odlučivanja

Izvor: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/decision-tree-nodes.png

4.3.2. Naïve Bayes

U statistici, *Naïve Bayes* klasifikatori predstavljaju skup jednostavnih, ali vrlo snažnih probabilističkih klasifikatora koji se temelje na primjeni Bayesovog teorema [8]. Bayesov teorem matematička je formula za određivanje uvjetne vjerojatnosti događaja [9]:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \quad (4.1.)$$

gdje su:

- P(Y) – vjerojatnost događaja Y
- P(X) – vjerojatnost događaja X
- P(Y|X) – vjerojatnost pojavljivanja događaja Y u slučaju gdje se događaj X dogodio
- P(X|Y) - vjerojatnost pojavljivanja događaja X u slučaju gdje se događaj Y dogodio.

Treba napomenuti da su događaji A i B nezavisni, tj. vjerojatnost ishoda događaja A ne ovisi o vjerojatnosti ishoda događaja B.

Kod implementacije *Naïve Bayes* klasifikatora velika je šansa da neće biti samo jedna nego više X varijabli, u kojem slučaju bi izgled Bayesovog teorema bio:

$$P(Y(X_1, \dots, X_n) = \frac{P(Y)P(X_1, \dots, X_n|Y)}{P(X_1, \dots, X_n)}) \quad (4.2.)$$

.

Postoji nekoliko *Naïve Bayes* algoritama [10]:

- Gaussov *Naïve Bayes* – klasifikator koji pretpostavlja da su vjerojatnosti značajki normalno distribuirane
- Bernoullijev *Naïve Bayes* – klasifikator za diskretne značajke, namijenjen za binarne/bool značajke
- Kategorički *Naïve Bayes* – klasifikator pogodan za klasificiranje diskretnih značajki koje su kategorički raspoređene
- Multinomijalni *Naïve Bayes* - klasifikator za diskretne značajke; distribucija obično zahtijeva frekvencije značajki
- Komplementarni *Naïve Bayes* – klasifikator za neuravnotežene skupove podataka.

4.3.3. K-sljedećih susjeda

K-sljedećih susjeda (eng. *K-nearest neighbours*, kNN) jest algoritam koji pripada kategoriji nadziranog učenja. Većinom se koristi za klasifikaciju, ali se može koristiti i kod problema regresije. Algoritam funkcionira tako da pokušava prediktirati točnu klasu za testne podatke računanjem udaljenosti između test-podataka i svih točaka treninga. Predikcije se izvode tako da traži prvih k najbližih točaka i klasificira ovisno o tome kome pripada najviše točaka. [11]

Postoji nekoliko opcija odabira k parametra. U većini se slučajeva k vrijednost računa korjenovanjem broja elemenata u trening-setu.

Alternativni način je korištenje *cross-validation* pristupom. Uzima se jedan dio trening-seta kojeg će se definirati kao set za validaciju. Nakon toga izvršava se predikcija na setu za validaciju gdje se kao k parametar zadaje raspon od 1 do krajnje željene vrijednosti. Najbolja k vrijednost jest ona koja daje najbolje performanse na korištenom validacijskom setu.

Poželjno je da k vrijednost bude neparan broj. U slučaju da tijekom klasificiranja jednak broj točaka pripada različitim klasama, doći će do izjednačenog rezultata i nasumično će birati kojoj od tih klasa pripada. Ako je k vrijednost velika, bit će manje osjetljiva na buku i time će se povećati performanse algoritma. Ako je k vrijednost premala, klasifikator će biti osjetljiv na buku što će dovesti do nestabilnih granica odlučivanja. [12]

5. Izrada modela

5.1. Podatci

Podatci su skinuti s Kaggle *web*-sjedišta, platforme za entuzijaste strojnog učenja i znanosti o podatcima na kojoj je moguće tražiti i dijeliti skupove podataka, raditi i dijeliti projekte te održavati natjecanja. Skup podataka NSL_KDD je derivat popularnog KDD'99 skupa koji je dobiven simuliranjem raznih upada unutar vojnog mrežnog okruženja. Dolazi s već spremnim skupovima za treniranje i testiranje u nekoliko formata. NSL-KDD ima nekoliko prednosti nad KDD'99 skupom:

- izbačeni su suvišni podatci iz trening-seta kako klasifikatori ne bi bili skloni zapisima koji se češće pojavljuju
- nema duplih zapisa u test-setu
- broj odabranih zapisa iz svake skupine razine težine obrnuto je proporcionalan postotku zapisa u izvornom KDD skupu podataka, što rezultira točnijom procjenom tehnika učenja
- sadrži razuman iznos zapisa za razliku od KDD'99 skupa koji ima oko 750 milijuna zapisa.

5.2. Validacija podataka

5.2.1. Unos podataka

Nakon prikupljanja željenih podataka, potrebno ih je na neki način unijeti u projekt. To je upravo jedna od funkcija Pandas biblioteke. Kao što je navedeno u poglavlju 3.3.1., Pandas može unositi podatke iz datoteka različitih formata. Prikupljeni su podatci spremljeni u .arff i .txt datotekama.

Prije izvršavanja unosa podataka potrebno je učitati potrebne biblioteke. Tijekom učitavanja moguće je navesti i *alias* koje je samo još jedno dodatno ime za unesenu biblioteku. Inače se koristi kako bi se brže pozivalo objekte iz biblioteke.

Pandasovom metodom *read_csv()* i unosom putanje učitavaju se podatci iz datoteke u *dataframe,* Pandasov dvodimenzionalni objekt za spremanje podataka. Budući da su

podatci odvojeni u trening i test-setove, mora ih se odvojeno unositi i kasnije spajati. Trening-set se unosi u *train_df*, a test-set u *test_df*. Također, odmah će se *print()* metodom ispisati.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math

train_df = pd.read_csv('nsl-kdd/KDDTrain+.txt')
test_df = pd.read_csv('nsl-kdd/KDDTest+.txt')

print(train_df, test_df)
```

Kôd 1. Učitavanje biblioteka, unos skupova podataka u dva *dataframea*

Izvor: Autor

```
        0  tcp  ftp_data   SF   491   0.1  0.2  0.3  0.4  0.5  ...  0.17  \
0       0  udp     other   SF   146     0    0    0    0    0  ...  0.00
1       0  tcp   private   S0     0     0    0    0    0    0  ...  0.10
2       0  tcp      http   SF   232  8153    0    0    0    0  ...  1.00
3       0  tcp      http   SF   199   420    0    0    0    0  ...  1.00
4       0  tcp   private  REJ     0     0    0    0    0    0  ...  0.07
...    ..  ...       ...  ...   ...   ...  ...  ...  ...  ...  ...   ...
125967  0  tcp   private   S0     0     0    0    0    0    0  ...  0.10
125968  8  udp   private   SF   105   145    0    0    0    0  ...  0.96
125969  0  tcp      smtp   SF  2231   384    0    0    0    0  ...  0.12
125970  0  tcp    klogin   S0     0     0    0    0    0    0  ...  0.03
125971  0  tcp  ftp_data   SF   151     0    0    0    0    0  ...  0.30

        0.03  0.17.1  0.00.6  0.00.7  0.00.8  0.05  0.00.9   normal  20
0       0.60    0.88    0.00    0.00    0.00  0.00    0.00   normal  15
1       0.05    0.00    0.00    1.00    1.00  0.00    0.00  neptune  19
2       0.00    0.03    0.04    0.03    0.01  0.00    0.01   normal  21
3       0.00    0.00    0.00    0.00    0.00  0.00    0.00   normal  21
4       0.07    0.00    0.00    0.00    0.00  1.00    1.00  neptune  21
...      ...     ...     ...     ...     ...   ...     ...      ...  ..
125967  0.06    0.00    0.00    1.00    1.00  0.00    0.00  neptune  20
125968  0.01    0.01    0.00    0.00    0.00  0.00    0.00   normal  21
125969  0.06    0.00    0.00    0.72    0.00  0.01    0.00   normal  18
125970  0.05    0.00    0.00    1.00    1.00  0.00    0.00  neptune  20
125971  0.03    0.30    0.00    0.00    0.00  0.00    0.00   normal  21

[125972 rows x 43 columns]       0   tcp   private   REJ    0.1    0.2  0.3  0.4  0.5  0.6  ...  0.04.1  \
0       0   tcp   private   REJ      0      0    0    0    0    0  ...    0.00
1       2   tcp  ftp_data    SF  12983      0    0    0    0    0  ...    0.61
2       0  icmp     eco_i    SF     20      0    0    0    0    0  ...    1.00
3       1   tcp    telnet  RSTO      0     15    0    0    0    0  ...    0.31
4       0   tcp      http    SF    267  14515    0    0    0    0  ...    1.00
...    ..   ...       ...   ...    ...    ...  ...  ...  ...  ...  ...     ...
22538   0   tcp      smtp    SF    794    333    0    0    0    0  ...    0.72
22539   0   tcp      http    SF    317    938    0    0    0    0  ...    1.00
22540   0   tcp      http    SF  54540   8314    0    0    0    2  ...    1.00
22541   0   udp  domain_u    SF     42     42    0    0    0    0  ...    0.99
22542   0   tcp    sunrpc   REJ      0      0    0    0    0    0  ...    0.08

       0.06.1  0.00.3  0.00.4  0.00.5  0.00.6  1.00.2  1.00.3  neptune  21
0        0.06    0.00    0.00    0.00     0.0    1.00    1.00  neptune  21
1        0.04    0.61    0.02    0.00     0.0    0.00    0.00   normal  21
2        0.00    1.00    0.28    0.00     0.0    0.00    0.00    saint  15
3        0.17    0.03    0.02    0.00     0.0    0.83    0.71    mscan  11
4        0.00    0.01    0.03    0.01     0.0    0.00    0.00   normal  21
...       ...     ...     ...     ...     ...     ...     ...      ...  ..
22538    0.06    0.01    0.01    0.01     0.0    0.00    0.00   normal  21
22539    0.00    0.01    0.01    0.01     0.0    0.00    0.00   normal  21
22540    0.00    0.00    0.00    0.00     0.0    0.07    0.07     back  15
22541    0.01    0.00    0.00    0.00     0.0    0.00    0.00   normal  21
22542    0.03    0.00    0.00    0.00     0.0    0.44    1.00    mscan  14

[22543 rows x 43 columns]
```

Slika 7. Rezultat metode za ispis

Izvor: Autor

Pogledom na gornju sliku vidljivo je da su podatci učitani. Međutim, podatci nemaju svoje identifikatore, tj. oznake (eng. *Label*). Potrebno ih je ručno unijeti. Unose se u polje columns.

```
columns = ([
'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', 'level'
])
train_df.columns = columns
test_df.columns = columns
```

Kôd 2. Unos oznaka elemenata u polje i primjena na skupove podataka

Izvor: Autor

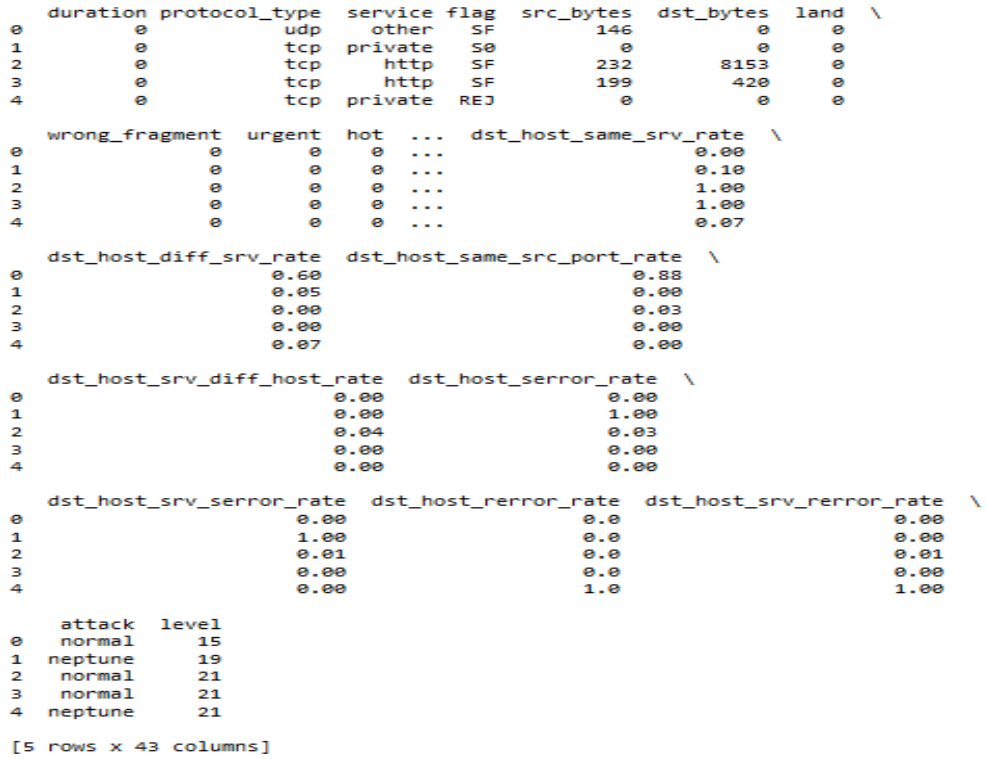
Te je oznake moguće dodati u *dataframe* na više načina. Mogu se prvo spojiti skupovi podataka i nakon toga dodati oznake ili se mogu prvo dodijeliti oznake i onda spojiti u jedan *dataframe*. Ovdje se išlo s drugom procedurom. Rezultat je isti, tako da je metoda nebitna.

Razlog zbog kojeg će se spojiti dva skupa u jedan jest taj što će na taj način doći do ponovne podjele na trening i test-setove pa će se moći nasumično birati podatci i uvijek će prediktirati drukčije. To će se implementirati s *concat* metodom u idućem kodu i uz to će se ispisati prvih pet elemenata pomoću metode head() da se vidi promjena na *dataframeu*. Također je zakomentirana alternativna metoda ispisa, koja u ovom slučaju nije korisna iz prezentacijskih razloga.

```
df = pd.concat([train_df, test_df], ignore_index=True)


print(df.head())
#df
```

Kôd 3. Spajanje skupova i ispis *dataframea*

Izvor: Autor



```
   duration protocol_type   service flag  src_bytes  dst_bytes  land  \
0         0           udp     other   SF        146          0     0
1         0           tcp   private   S0          0          0     0
2         0           tcp      http   SF        232       8153     0
3         0           tcp      http   SF        199        420     0
4         0           tcp   private  REJ          0          0     0

   wrong_fragment  urgent  hot  ...  dst_host_same_srv_rate  \
0               0       0    0  ...                    0.00
1               0       0    0  ...                    0.10
2               0       0    0  ...                    1.00
3               0       0    0  ...                    1.00
4               0       0    0  ...                    0.07

   dst_host_diff_srv_rate  dst_host_same_src_port_rate  \
0                    0.60                         0.88
1                    0.05                         0.00
2                    0.00                         0.03
3                    0.00                         0.00
4                    0.07                         0.00

   dst_host_srv_diff_host_rate  dst_host_serror_rate  \
0                         0.00                  0.00
1                         0.00                  1.00
2                         0.04                  0.03
3                         0.00                  0.00
4                         0.00                  0.00

   dst_host_srv_serror_rate  dst_host_rerror_rate  dst_host_srv_rerror_rate  \
0                      0.00                   0.0                      0.00
1                      1.00                   0.0                      0.00
2                      0.01                   0.0                      0.01
3                      0.00                   0.0                      0.00
4                      0.00                   1.0                      1.00

    attack  level
0   normal     15
1  neptune     19
2   normal     21
3   normal     21
4  neptune     21

[5 rows x 43 columns]
```

Slika 8. Prikaz prvih pet elemenata *dataframea* nakon spajanja skupova podataka

Izvor:Autor

Ako se izostavi parametar *ignore_index* od metode *concat()*, elementi u *dataframe* objektu bili bi indeksirani od 0 do n-1 (gdje je n broj elemenata u trening-setu) i nakon toga bi išli ponovno od 0 do n-1 (gdje je n broj elemenata u test-skupu). Postavljanjem *ignore_index=True* ignorirat će prošle indekse i elementi unutar novog *dataframe* objekta normalno će indeksirati.

5.2.2. Čišćenje podataka

Čišćenje podataka je postupak popravljanja ili uklanjanja netočnih, oštećenih, neispravno formatiranih, duplih ili nepotpunih podataka unutar skupa podataka. Autor *dataseta* prethodno je odradio proces uklanjanja duplih podataka. Pomoću Pandas biblioteke to se može izvesti na više načina. Sljedećom linijom koda ispisat će se broj *null* vrijednosti u svakom stupcu.

```
df.isnull().sum()
```

Kôd 4. Metoda za ispis broja *null* vrijednosti u stupcu

```
duration                        0
protocol_type                   0
service                         0
flag                            0
src_bytes                       0
dst_bytes                       0
land                            0
wrong_fragment                  0
urgent                          0
hot                             0
num_failed_logins               0
logged_in                       0
num_compromised                 0
root_shell                      0
su_attempted                    0
num_root                        0
num_file_creations              0
num_shells                      0
num_access_files                0
num_outbound_cmds               0
is_host_login                   0
is_guest_login                  0
count                           0
srv_count                       0
serror_rate                     0
srv_serror_rate                 0
rerror_rate                     0
srv_rerror_rate                 0
same_srv_rate                   0
diff_srv_rate                   0
srv_diff_host_rate              0
dst_host_count                  0
dst_host_srv_count              0
dst_host_same_srv_rate          0
dst_host_diff_srv_rate          0
dst_host_same_src_port_rate     0
dst_host_srv_diff_host_rate     0
dst_host_serror_rate            0
dst_host_srv_serror_rate        0
dst_host_rerror_rate            0
dst_host_srv_rerror_rate        0
attack                          0
level                           0
dtype: int64
```

Slika 9. Rezultat metode

Izvor: Autor

Pogledom na Sliku 9. vidljivo je da nema *null* vrijednosti.

Info() metoda je alternativni način traženja *null* vrijednosti. Osim toga ispisuje osnovne informacije o podatcima: indeks, ime stupca, broj *non-null* vrijednosti i koja je vrsta podatka.

```
df.info()
```

Kôd 5. Metoda za prikaz informacija o *dataframeu*

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148515 entries, 0 to 148514
Data columns (total 43 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   duration                     148515 non-null  int64
 1   protocol_type                148515 non-null  object
 2   service                      148515 non-null  object
 3   flag                         148515 non-null  object
 4   src_bytes                    148515 non-null  int64
 5   dst_bytes                    148515 non-null  int64
 6   land                         148515 non-null  int64
 7   wrong_fragment               148515 non-null  int64
 8   urgent                       148515 non-null  int64
 9   hot                          148515 non-null  int64
 10  num_failed_logins            148515 non-null  int64
 11  logged_in                    148515 non-null  int64
 12  num_compromised              148515 non-null  int64
 13  root_shell                   148515 non-null  int64
 14  su_attempted                 148515 non-null  int64
 15  num_root                     148515 non-null  int64
 16  num_file_creations           148515 non-null  int64
 17  num_shells                   148515 non-null  int64
 18  num_access_files             148515 non-null  int64
 19  num_outbound_cmds            148515 non-null  int64
 20  is_host_login                148515 non-null  int64
 21  is_guest_login               148515 non-null  int64
 22  count                        148515 non-null  int64
 23  srv_count                    148515 non-null  int64
 24  serror_rate                  148515 non-null  float64
 25  srv_serror_rate              148515 non-null  float64
 26  rerror_rate                  148515 non-null  float64
 27  srv_rerror_rate              148515 non-null  float64
 28  same_srv_rate                148515 non-null  float64
 29  diff_srv_rate                148515 non-null  float64
 30  srv_diff_host_rate           148515 non-null  float64
 31  dst_host_count               148515 non-null  int64
 32  dst_host_srv_count           148515 non-null  int64
 33  dst_host_same_srv_rate       148515 non-null  float64
 34  dst_host_diff_srv_rate       148515 non-null  float64
 35  dst_host_same_src_port_rate  148515 non-null  float64
 36  dst_host_srv_diff_host_rate  148515 non-null  float64
 37  dst_host_serror_rate         148515 non-null  float64
 38  dst_host_srv_serror_rate     148515 non-null  float64
 39  dst_host_rerror_rate         148515 non-null  float64
 40  dst_host_srv_rerror_rate     148515 non-null  float64
 41  attack                       148515 non-null  object
 42  level                        148515 non-null  int64
dtypes: float64(15), int64(24), object(4)
memory usage: 48.7+ MB
```

Slika 10. Ispis *info()* metode

Izvor: Autor

5.2.3. Eksplorativna analiza i vizualiziranje podataka

U ovom dijelu nalaze se grafički prikazi uglavnom vezani za varijablu *'attack'* koja je ciljna varijabla i prati se njen odnos s kategoričkim varijablama. Izvršit će se transformacija istih kategoričkih varijabli u oblik u kojem se može koristiti u prediktivnom modelu. Te se kategorije moraju pretvoriti u brojčani oblik jer većina algoritama radi samo s brojčanim vrijednostima. Isto tako će se i ciljna varijabla transformirati i koristiti ovisno o odabiru između binarne i višeklasne klasifikacije.

Za početak potrebno je vidjeti kakva je distribucija napada. Pandas ima metodu *value_counts()* koja vraća niz koji sadrži broj jedinstvenih elemenata (redova) u *dataframeu*.

```
df['attack'].value_counts()
```

Kôd 6. Metoda za prikaz jedinstvenih elemenata u *dataframeu*

Izvor: Autor

```
normal              77053
neptune             45870
satan                4368
ipsweep              3740
smurf                3311
portsweep            3088
nmap                 1566
back                 1315
guess_passwd         1284
mscan                 996
warezmaster           964
teardrop              904
warezclient           890
apache2               737
processtable          685
snmpguess             331
saint                 319
mailbomb              293
pod                   242
snmpgetattack         178
httptunnel            133
buffer_overflow        50
land                   25
multihop               25
rootkit                23
named                  17
ps                     15
sendmail               14
xterm                  13
imap                   12
loadmodule             11
ftp_write              11
xlock                   9
phf                     6
perl                    5
xsnoop                  4
spy                     2
worm                    2
sqlattack               2
udpstorm                2
```

Slika 11. Ispis rezultata metode *value_counts()* na ciljnoj varijabli

Izvor: Autor

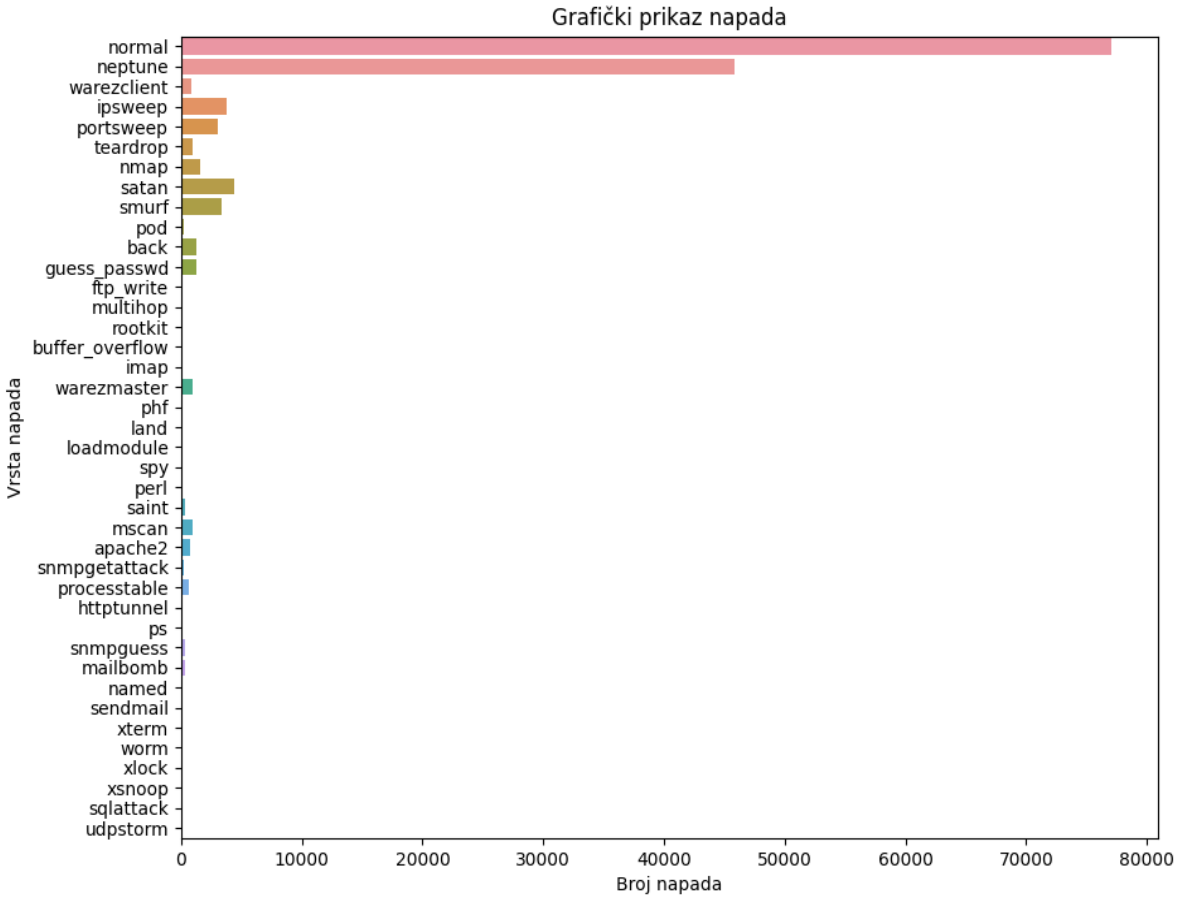
Na slici 11. vidljivo je da oko 50 % zabilježenih događaja nisu napadi. Od stvarnih napada najviše ima iz *neptune* klase. Također, pregledom distribucije zaključuje se da je *dataset* neuravnotežen. Neuravnoteženi *dataset* je *dataset* gdje su klase nejednako distribuirane. Ako model koristi ovakvu distribuciju, bit će sklon prediktiranju većinskih klasa, tj. *normal* i *neptune* klase. Više o ovome kod modeliranja.

Za prikaz distribucije koristi se *countplot* graf iz seaborn biblioteke. Taj graf bi se moglo definirati kao histogram za kategoričke varijable. Budući da su seaborn grafovi samo gotovi grafovi izrađeni s matplotlib bibliotekom, moguće ih je zajedno kombinirati. Iz matplotlib biblioteke prvo se poziva figure() metoda koja izrađuje plohu za graf. Ako se ne pozove, kod iscrtavanja grafa automatski će se napraviti s *default* vrijednostima. Uz to se zadaje i parametar *figsize=()* gdje se određuju željene dimenzije grafa. Nakon toga se iz seaborn biblioteke poziva *countplot()* metoda za iscrtavanje grafa i u parametrima se dodaju data za odabir podataka i y vrijednost kojom će se na y-osi prikazati distribucija napada. Također, metodom *title()* dodaje se naslov grafa, *xlabel()* za dodavanje oznake x-osi i isto tako za y-os metodom *ylabel()*.

```
plt.figure(figsize=(10, 8))
attack_dist = sns.countplot(data=df, y='attack')
plt.title('Grafički prikaz napada')
plt.xlabel('Broj napada')
plt.ylabel('Vrsta napada')
```

Kôd 7. Vizualiziranje distribucije napada

Izvor: Autor



Slika 12. Grafički prikaz distribucije napadačkih klasa

Izvor: Autor

Gore iscrtani graf može poslužiti kao bolji prikaz neuravnoteženosti klasa.

Za daljnju analizu najbolji kandidati su značajke koje nisu u brojčanom obliku. To su *protocol_type, flag i service*. Za to će se koristiti Pandasov objekt koji se zove unakrsna tablica (eng. *Crosstab*). Njome se mogu izračunati frekvencije dviju ili više značajki iz *dataframea.*

Za početak implementirat će se unakrsna tablica s *'attack'* i *'protocol_type'* značajkama. Poziva se *crosstab()* metoda s parametrima za odabir značajki i sprema se u varijablu. Također se ispisuje.

```
attacks_per_protocol = pd.crosstab(df.attack, df.protocol_type)
attacks_per_protocol
```

Kôd 8. Kôd za izradu unakrsne tablice
Izvor: Autor

protocol_type attack	icmp	tcp	udp
apache2	0	737	0
back	0	1315	0
buffer_overflow	0	50	0
ftp_write	0	11	0
guess_passwd	0	1284	0
httptunnel	0	133	0
imap	0	12	0
ipsweep	3258	482	0
land	0	25	0
loadmodule	0	11	0
mailbomb	0	293	0
mscan	0	996	0
multihop	1	16	8
named	0	17	0
neptune	0	45870	0
nmap	981	338	247
normal	1402	61441	14210
perl	0	5	0
phf	0	6	0
pod	242	0	0
portsweep	5	3083	0
processtable	0	685	0
ps	0	15	0
rootkit	0	20	3
saint	98	203	18
satan	33	2619	1716
sendmail	0	14	0
smurf	3311	0	0
snmpgetattack	0	0	178
snmpguess	3	0	328
spy	0	2	0
sqlattack	0	2	0
teardrop	0	0	904
udpstorm	0	0	2
warezclient	0	890	0
warezmaster	0	964	0
worm	0	2	0
xlock	0	9	0
xsnoop	0	4	0
xterm	0	13	0

Slika 13. Izgled unakrsne tablice

Izvor: Autor

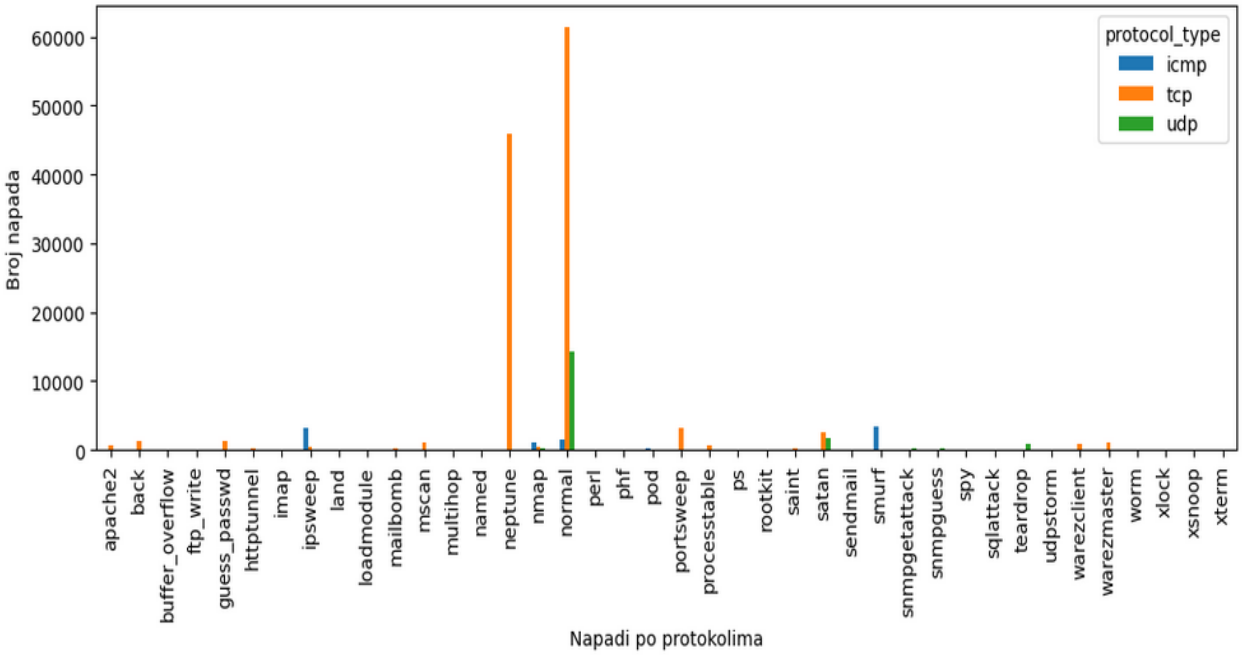
Vidljivo je da je većina zabilježenih događaja registrirana na TCP protokolu.
Unakrsna tablica se također može prikazati kao graf:

```
app = attacks_per_protocol.plot(kind='bar', figsize=(12, 4),
xlabel="Napadi po protokolima", ylabel="Broj napada")
```

Kôd 9. Izrada grafa za iscrtavanje gore definirane unakrsne tablice

Izvor: Autor

Jednostavno se može iscrtati metodom *plot()*. Za analizu podataka iz unakrsne tablice najbolje je koristiti stupčasti graf. Dodavanjem zadanih parametara dobiva se sljedeći graf:



Slika 14. Grafički prikaz unakrsne tablice

Izvor: Autor

Zbog neuravnoteženosti podataka teško je doći do bilo kakvih novih zaključaka.
Istim principom može se prikazati i stanje s *flag* i *service* značajkama, ovisno o distribuciji.

```
df.service.value_counts()
```

Kôd 10. Linija koda za ispis frekvencija servisa

Izvor: Autor

```
http         48191
private      26626
domain_u      9937
smtp          8247
ftp_data      7710
              ...
tftp_u           4
http_8001        2
aol              2
harvest          2
http_2784        1
Name: service, Length: 70, dtype: int64
```

Slika 15. Frekvencije servisa

Izvor: Autor

Ovdje je uočljivo da daljnja analiza nije isplativa iz dva razloga: ponovno postoji velika razlika u distribuciji i također postoji sedamdeset različitih jedinstvenih kategorija.

```
df.flag.value_counts()
```

Kôd 11. Linija koda za ispis frekvencija značajke *'flag'*

Izvor: Autor

```
SF        89819
S0        36864
REJ       15082
RSTR       3090
RSTO       2335
S1          386
SH          344
S3          298
S2          142
RSTOS0      105
OTH          50
```

Slika 16. Prikaz frekvencija značajke *flag*

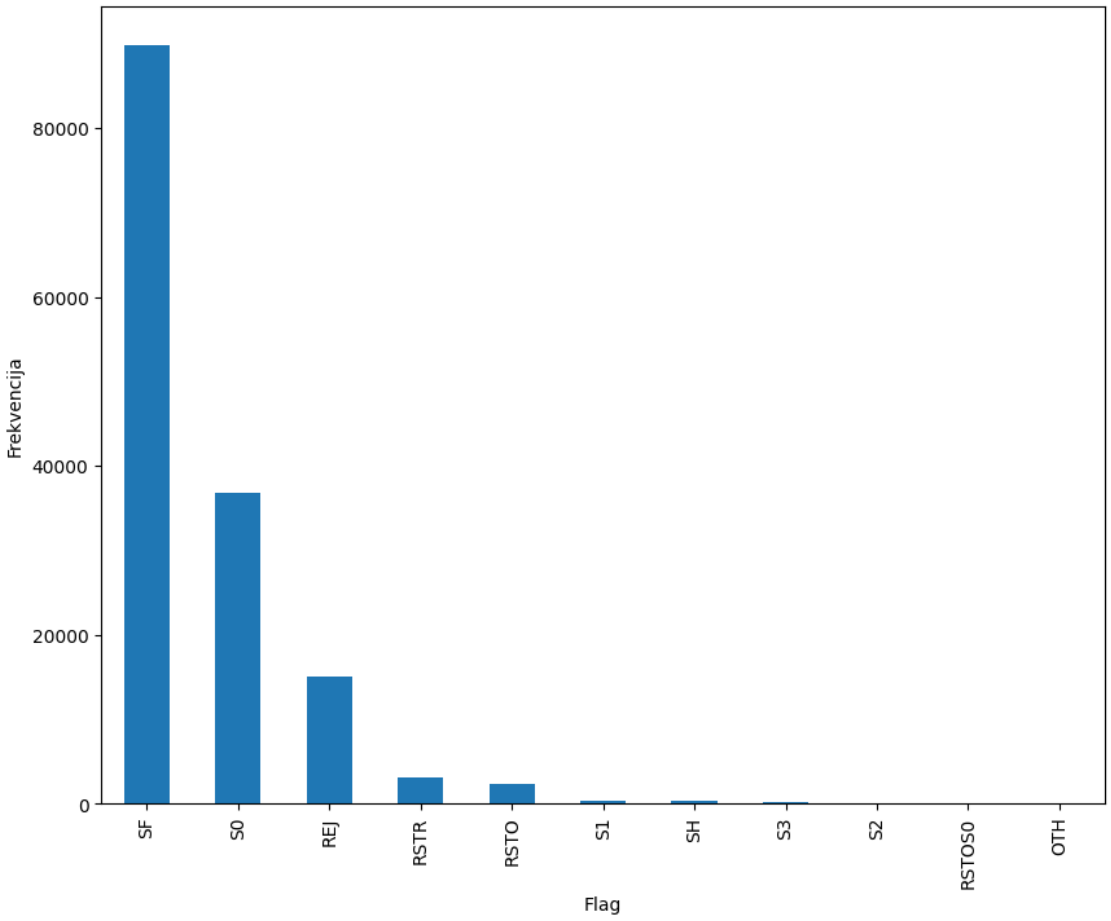
Izvor: Autor

Ovdje ima jedanaest jedinstvenih kategorija. Isplativo je napraviti graf s frekvencijama. Procedura s unakrsnom tablicom ovdje ne bi bila korisna zato jer se ne bi moglo ništa očitati s grafa.

```
df.flag.value_counts().plot(figsize=(10, 8), kind='bar',
ylabel='Frekvencija', xlabel='Flag')
```

Kôd 12. Kod za grafički prikaz *'flag'* značajke

Izvor: Autor



Slika 17. Grafički prikaz frekvencija *flag* značajke

Izvor: Autor

Nadalje, moguće je klasificirati napade prema njihovoj vrsti:

- *DoS (Denial of Service): apache2, back, land, Neptune, mailbomb, pod, processtable, smurf, teardrop, udpstorm, worm*

- *Probe: ipsweep, mscan, nmap, portsweep, saint, satan*
- *U2R: buffer_overflow, loadmodule, perl, ps, rootkit, sqlattack, xterm*
- *R2L: ftp_write, guess_passwd, http_tunnel, imap, multihop, named, phf, sendmail, snmpgetattack, snmpguess, spy, warezclient, warezmaster, xclock, xsnoop.*

```
df['attack_classes'] = df.loc[:, 'attack'].values
df
```

Kôd 13. Kopiranje 'attack' značajke i ispis

Izvor: Autor

Kako bi se također sačuvali i neklasificirani napadi, kopirat će se u novi stupac *'attack_classes'*. Gore navedeni kôd je procedura za kopiranje i ispis novog stupca, tj. značajke. Idući korak je na neki način transformirati te kopirane podatke.

```
def attack_label(dataframe):

    dataframe.attack_classes.replace(['apache2','back','land',
    'neptune','mailbomb','pod','processtable','smurf','teardro
    p','udpstorm','worm'],'DoS', inplace=True)
    dataframe.attack_classes.replace(['ftp_write','guess_passw
    d','httptunnel','imap','multihop',
    'named','phf','sendmail',
    'snmpgetattack','snmpguess','spy','warezclient','warezmast
    er','xlock','xsnoop'],'R2L', inplace=True)
    dataframe.attack_classes.replace(['ipsweep','mscan',
    'nmap','portsweep','saint', 'satan'],'probe',
    inplace=True)

    dataframe.attack_classes.replace(['buffer_overflow','loadm
    odule','perl','ps','rootkit','sqlattack','xterm'],'U2R',
    inplace=True)

attack_label(df)
```

Kôd 14. Transformacija novog stupca

Izvor: Autor

Ovdje se opet može iskoristiti prikaz pomoću unakrsne tablice.

```
attacks_per_protocol2 = pd.crosstab(df.attack_classes,
df.protocol_type)

attacks_per_protocol2
```

Kôd 15. Izrada nove unakrsne tablice

Izvor: Autor

protocol_type	icmp	tcp	udp
attack_classes			
DoS	3553	48927	906
R2L	4	3362	514
U2R	0	116	3
normal	1402	61441	14210
probe	4375	7721	1981

Slika 18. Izgled unakrsne tablice

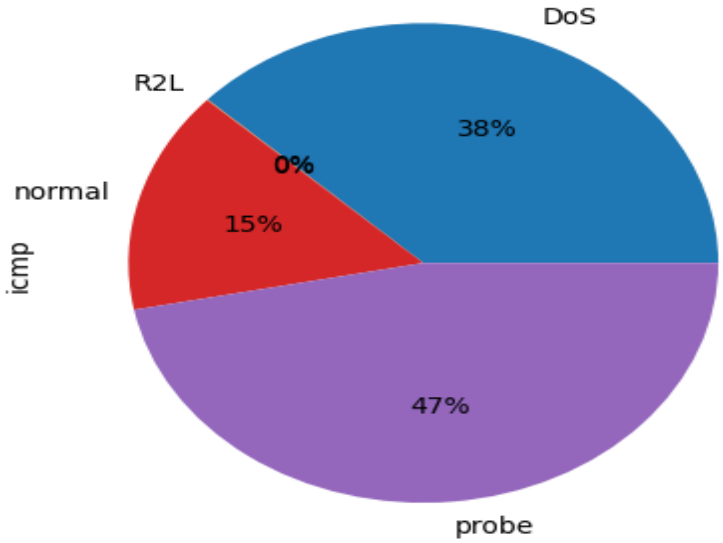
Izvor: Autor

Pomoću nove tablice moguće je prikazati distribuciju na još jedan način.

```
icmp_att = attacks_per_protocol2.icmp
tcp_att = attacks_per_protocol2.tcp
udp_att = attacks_per_protocol2.udp

pies = [icmp_att, tcp_att, udp_att]
for p in pies:
    p.plot.pie(autopct='%.0f%%')
    plt.show()
```

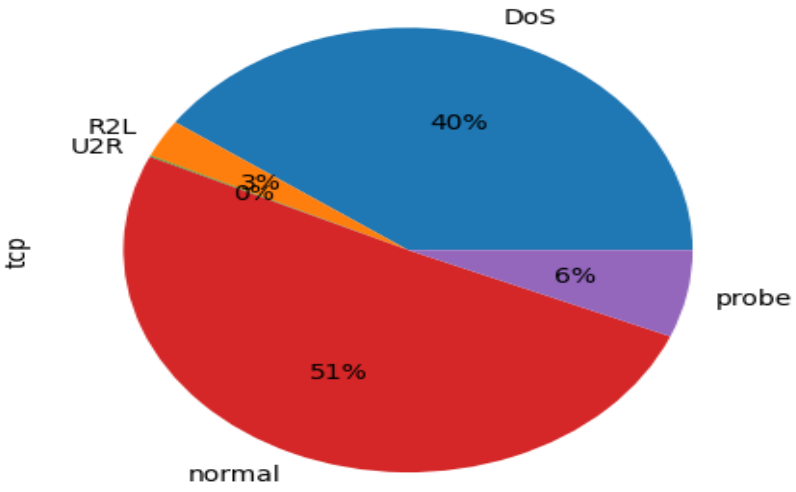
Kôd 16. Kôd za prikaz distribucije napadačkih tipova i protokola

Izvor: Autor



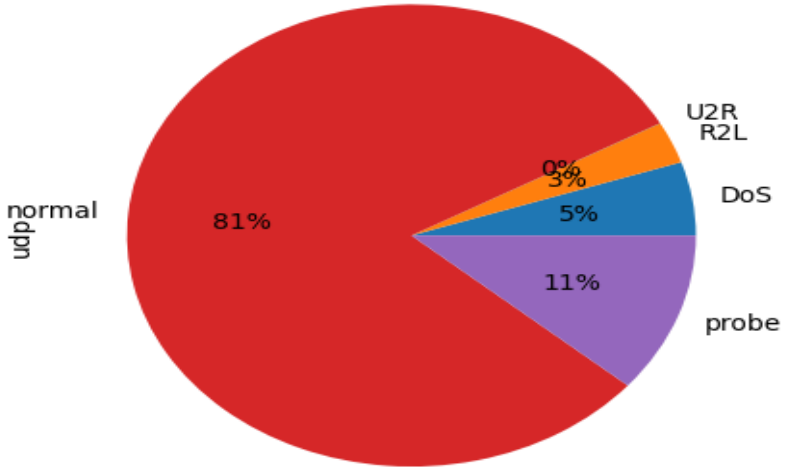
Slika 19. Distribucija napadačkih tipova u ICMP protokolu

Izvor: Autor



Slika 20. Distribucija napadačkih tipova u TCP protokolu

Izvor: Autor



Slika 21. Distribucija napadačkih tipova u UDP protokolu

Izvor: Autor

5.2.4. **Transformacija podataka**

Jedan od zadataka bio je istražiti koji su danas još uvijek aktualni napadi. Od 43 napada, tri će se izbaciti:

- *Security Administrator Tool for Analyzing Networks* (SATAN) – zamijenili su ga *nmap* i *Saint* alati
- *Mscan* – nema podataka, pretpostavlja se da više nije u upotrebi
- *Named* – nema podataka.

Svakom elementu pregledat će se '*attack'* vrijednost i ako nisu '*satan'*, '*mscan'* ili '*named'* ostat će, a ako jesu, izbacit će se.

```
df                                                                    =
df[df.attack.str.contains("satan|mscan|named")==False].copy()
```

Kôd 17. Kôd za izbacivanje neaktualnih tipova napada

Izvor: Autor

Kad se izbacuju elementi, indeksi se automatski ne mijenjaju pa je među njima nastao kaos. Jednom ih se linijom koda može resetirati:

```
df.reset_index(drop=True, inplace=True)
```

Kôd 18. Resetiranje indeksa

Izvor: Autor

Idući korak je *string* značajke pretvoriti u brojčani oblik jer većina algoritama može koristiti samo brojčane vrijednosti. To se može odraditi na dva načina:

- *Label Encoder* objekt pretvara *string* vrijednosti u niz brojeva od 1 do n
- *OneHotEncoding* objekt za svaku jedinstvenu vrijednost radi stupac s vrijednosti 0 ili 1

```
from sklearn.preprocessing import LabelEncoder

to_encode = ['protocol_type', 'service', 'flag']

le = LabelEncoder()

df[to_encode] =
df[to_encode].apply(LabelEncoder().fit_transform)
```

Kôd 19. Primjena Label Encoder objekta na *string* značajke

Izvor: Autor

protocol_type	service	flag
udp	other	SF
tcp	private	S0
tcp	http	SF
tcp	http	SF
tcp	private	REJ
...	...	...
icmp	ecr_i	SF
tcp	smtp	SF
tcp	http	SF
tcp	http	SF
udp	domain_u	SF

Slika 22. Značajke prije primjene *Label Encodera*

Izvor: Autor

protocol_type	service	flag
2	40	9
1	44	5
1	22	9
1	22	9
1	44	1
...	...	...
0	14	9
1	49	9
1	22	9
1	22	9
2	11	9

Slika 23. Značajke nakon primjene *Label Encodera*

Izvor: Autor

Sad kad su sve značajke osim ciljne varijable u brojčanom obliku, potrebno je izvršiti standardiziranje. Standardiziranje je metoda pretprocesiranja koja pretvara kontinuirane varijable kako bi izgledale kao da imaju normalnu distribuciju vrijednosti. Ovaj korak je inače potrebno izvršiti jer većina modela pretpostavlja da je distribucija podataka u trening-setu normalna. [13]

```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()

df[columns[:40]] = ss.fit_transform(df[columns[:40]])

df
```

Kôd 20. Primjena standardizacije

Izvor: Autor

Standardiziranje se izvršava na prvih četrdeset značajki, a '*level'* će se značajka, koja se u *dataframeu* nalazi između dviju napadačkih stupaca, ionako izbaciti prije treniranja.

duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot
-0.11464	2.287301	0.790178	0.722250	-0.007548	-0.004699	-0.014954	-0.087092	-0.010597	-0.095790
-0.11464	-0.110515	1.062579	-0.770091	-0.007575	-0.004699	-0.014954	-0.087092	-0.010597	-0.095790
-0.11464	-0.110515	-0.435627	0.722250	-0.007532	-0.002538	-0.014954	-0.087092	-0.010597	-0.095790
-0.11464	-0.110515	-0.435627	0.722250	-0.007538	-0.004588	-0.014954	-0.087092	-0.010597	-0.095790
-0.11464	-0.110515	1.062579	-2.262431	-0.007575	-0.004699	-0.014954	-0.087092	-0.010597	-0.095790
...	...	...	...	...	...	...	...	...	...
-0.11464	-2.508330	-0.980429	0.722250	-0.007387	-0.004699	-0.014954	-0.087092	-0.010597	-0.095790
-0.11464	-0.110515	1.403080	0.722250	-0.007430	-0.004611	-0.014954	-0.087092	-0.010597	-0.095790
-0.11464	-0.110515	-0.435627	0.722250	-0.007517	-0.004450	-0.014954	-0.087092	-0.010597	-0.095790
-0.11464	-0.110515	-0.435627	0.722250	0.002323	-0.002495	-0.014954	-0.087092	-0.010597	0.879684
-0.11464	2.287301	-1.184729	0.722250	-0.007567	-0.004688	-0.014954	-0.087092	-0.010597	-0.095790

Slika 24. Izgled prvih deset značajki nakon standardiziranja

Izvor: Autor

Prvi prediktivni model će klasificirati binarno, tako da klasificirani napadački tipovi u stupcu *attack_classes* više ne trebaju. Njih se izbacuje, a *attack* stupac s izvornim vrijednostima će se urediti tako da *normal* napadi postaju vrijednost 0, a svi ostali 1.

```
attack_flag = df.attack.map(lambda a: 0 if a == 'normal' else 1)

df['attack_flag'] = attack_flag

df.drop(['attack_classes', 'attack'], axis=1, inplace=True)
```

Kôd 21. Izbacivanje napadačkih klasa i izmjena izvornih napada u binarni oblik

Izvor: Autor

6. Prediktivni modeli

U ovom dijelu će se najprije birati podatci za modele i nakon toga će se izvršiti podjela na trening i test-setove. Koriste se Gaussov *Naïve Bayes* algoritam, stablo odlučivanja i K-sljedeći susjed.

6.1. Binarna klasifikacija

U prvom modelu koriste se podatci koji imaju korelaciju 0.5 i više s ciljnom varijablom. Korelacija se može prikazati u matričnom obliku pomoću *corr()* metode ili se *for* petljom prolazi kroz svaki stupac, mjeri se i ispisuje korelacija. Koristit će se modificirana verzija druge opcije – *for* petljom će proći kroz korelacije i ako je korelacija manja od 0.5, onda se dodaje u polje *to_drop*. Važno je napomenuti da se gleda korelacija od 0.5 i više i -0.5 i manje. Kako bi se mjerila negativna korelacija, potrebno je uzeti njezinu apsolutnu vrijednost.

```
to_drop = []

dfc = df.copy()

for x in dfc:

    if abs(dfc[x].corr(dfc['attack_flag'])) < 0.5:

        to_drop.append(x)

to_drop

df.drop(to_drop, axis=1, inplace=True)
```

Kôd 22. Kôd za filtriranje nezadovoljavajućih značajki

Izvor: Autor

```
['duration',
 'protocol_type',
 'service',
 'src_bytes',
 'dst_bytes',
 'land',
 'wrong_fragment',
 'urgent',
 'hot',
 'num_failed_logins',
 'num_compromised',
 'root_shell',
 'su_attempted',
 'num_root',
 'num_file_creations',
 'num_shells',
 'num_access_files',
 'is_host_login',
 'is_guest_login',
 'srv_count',
 'rerror_rate',
 'srv_rerror_rate',
 'diff_srv_rate',
 'srv_diff_host_rate',
 'dst_host_count',
 'dst_host_diff_srv_rate',
 'dst_host_same_src_port_rate',
 'dst_host_srv_diff_host_rate',
 'dst_host_rerror_rate',
 'dst_host_srv_rerror_rate',
 'level']
```

Slika 25. Ispis značajki s korelacijom manjom od 0.5

Izvor: Autor

```
for c in df:

    print(c, ": ", df[c].corr(df['attack_flag']))
```

Kôd 23. Ispis neizbačenih značajki i njihove korelacije

Izvor: Autor

```
flag :  -0.6374281056339409
logged_in :  -0.6568210008702907
num_outbound_cmds :  nan
count :  0.5387860125956017
serror_rate :  0.621900116394487
srv_serror_rate :  0.6224806610074821
same_srv_rate :  -0.7071794362248465
dst_host_srv_count :  -0.6834328342882657
dst_host_same_srv_rate :  -0.6557232947979804
dst_host_serror_rate :  0.6226960676117843
dst_host_srv_serror_rate :  0.6297733875799368
attack_flag :  1.0
```

Slika 26. Ispis zadovoljavajućih značajki

Izvor: Autor

Na ispisu je vidljivo da za stupac *num_outbound_cmds* piše da je korelacija *NaN* (*Not-a-Number*) pa se jednostavno izbacuje.

```
df.drop('num_outbound_cmds', axis=1, inplace=True)
```

Kôd 24. Izbacivanje stupca kojemu ispisuje korelaciju *NaN*

Izvor: Autor

Sad su podatci spremni za treniranje. Prije svega treba učitati *train_test_split* iz sklearn biblioteke. U X varijablu idu sve vrijednosti osim ciljne, a ciljna varijabla ide u y varijablu. *Train_test_split* funkcionira tako da će nasumično uzeti određeni dio podataka za treniranje, a ostatak će se koristiti za predikciju. U ovom slučaju, budući da model klasificira binarno i s podatcima visoke korelacije, može se trening-setu dati manje podataka za učenje.

```
from sklearn.model_selection import train_test_split

X = df.iloc[:,:-1]

y = df.iloc[:,-1]

X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=0.3)
```

Kôd 25. Treniranje podataka za prvi model

Izvor: Autor

Sad je sve spremno za implementiranje algoritama za predikciju. Prvo će se implementirati Gaussov *Naïve Bayes* algoritam te će se rezultati ispisati u obliku klasifikacijskog izvještaja i matrice konfuzije:

6.1.1. Gaussov Naïve Bayes

```
from sklearn.metrics import classification_report

from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()

gnb.fit(X_train, y_train)

gnb_predict=gnb.predict(X_test)

print(classification_report(y_test, gnb_predict))
```

Kôd 26. Implementacija Gaussovog *Naïve Bayes* algoritma

Izvor: Autor

Točnost predikcija provjerava se korištenjem klasifikacijskog izvještaja:

	precision	recall	f1-score	support
0	0.83	0.95	0.88	23082
1	0.93	0.77	0.84	19859
accuracy			0.87	42941
macro avg	0.88	0.86	0.86	42941
weighted avg	0.87	0.87	0.86	42941

Slika 27. Klasifikacijski izvještaj za binarni Gaussov Naive Bayes

Izvor: Autor

Precision je omjer broja pozitivnih i zbroja pozitivnih i lažno pozitivnih vrijednosti.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6.1.}$$

Recall je omjer broja pozitivnih vrijednosti i zbroja pozitivnih i lažno negativnih vrijednosti.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6.2.}$$

F1-score je harmonijska sredina *precision* i *recall* vrijednosti.

$$\text{F1} - \text{score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.3.}$$

Da bi se prikazala konfuzijska matrica, trebaju dva argumenta: varijabla s prediktiranim vrijednostima i za raznolikost je dodana mogućnost promjene boje. Konfuzijska matrica funkcionira tako da uspoređuje *y_test* vrijednosti s prediktiranim vrijednostima iz algoritma.

6.1.2. Stablo odlučivanja

Postupak za implementaciju stabla odlučivanja više-manje je isti:

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(random_state=0, max_depth=50)
dtc.fit(X_train, y_train)
dtc_predict=dtc.predict(X_test)
print(classification_report(y_test, dtc_predict))
```

Kôd 27. Klasifikacija sa stablom odlučivanja

Izvor: Autor

	precision	recall	f1-score	support
0	0.95	0.96	0.95	23082
1	0.95	0.94	0.94	19859
accuracy			0.95	42941
macro avg	0.95	0.95	0.95	42941
weighted avg	0.95	0.95	0.95	42941

Slika 28. Klasifikacijski izvještaj binarnog stabla odlučivanja

Izvor: Autor

6.1.3. K-sljedećih susjeda

Kod ovog algoritma performanse mogu ovisiti o odabiru vrijednosti *n_neighbors*. Vrijednost *n_neighbors* može se dobiti na više načina:

- izračunavanjem drugog korijena broja elemenata u trening-setu
- korištenjem *GridSearchCV (Cross-Validation)*
- uzme se dio trening-seta i ti podatci se koriste kao skup za validaciju prediktiranjem nad validacijskim setom s k vrijednostima u određenom rasponu i uzme se k vrijednost koja ima najbolje performance.

Vrijednost će s u ovom slučaju računati prvom metodom.

```
Import math
from sklearn.neighbors import KNeighborsClassifier

n=int(math.sqrt(len(X_train)))

knn = KNeighborsClassifier(n_neighbors=n)
knn.fit(X_train, y_train)
knn_predict=knn.predict(X_test)
print(classification_report(y_test, knn_predict))
```

Kôd 28. Klasifikacija implementacijom KNN-a

Izvor: Autor

	precision	recall	f1-score	support
0	0.92	0.96	0.94	23045
1	0.95	0.91	0.93	19896
accuracy			0.94	42941
macro avg	0.94	0.93	0.94	42941
weighted avg	0.94	0.94	0.94	42941

Slika 29. Klasifikacijski izvještaj kNN-a

Izvor: Autor

Performanse algoritama će se provjeriti i korištenjem konfuzijske matrice.

```
from sklearn.metrics import confusion_matrix

def conf_matrix(pred, color):
    conf_matrix=confusion_matrix(y_test, pred)
    sns.heatmap(conf_matrix, annot=True, cmap=color,
fmt="d")

predictions = [ gnb_predict, dtc_predict, knn_predict ]
```

Kôd 29. Izrada funkcije za prikaz konfuzijske matrice i unos predikcija triju algoritama

Izvor: Autor

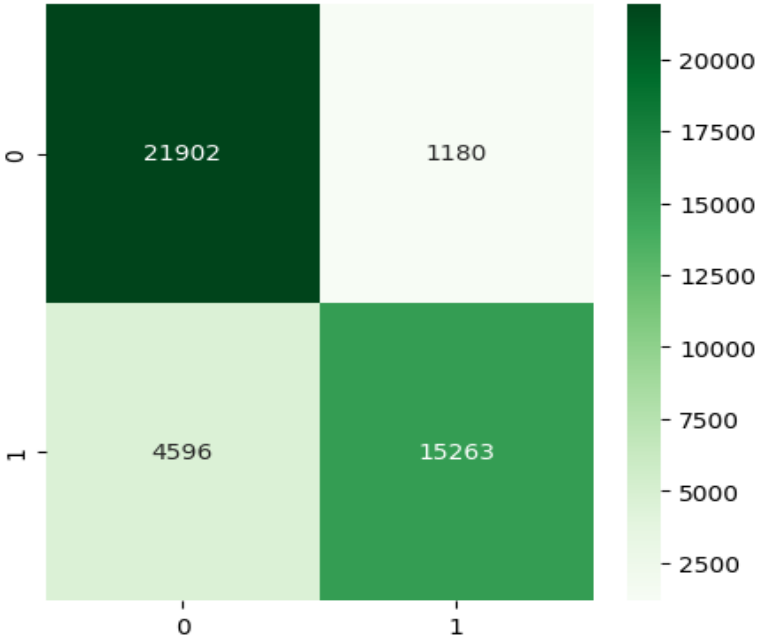
Da bi se prikazala konfuzijska matrica, trebaju dva argumenta: varijabla s prediktiranim vrijednostima i za raznolikost je dodana mogućnost promjene boje. Konfuzijska matrica funkcionira tako da uspoređuje *y_test* vrijednosti s prediktiranim vrijednostima iz algoritma. Konfuzijske matrice će se ispisivati sve zajedno *for* petljom:

```
for p in predictions:
    plt.figure(figsize=(5, 5))
```

```
conf_matrix(p, 'Greens')
```

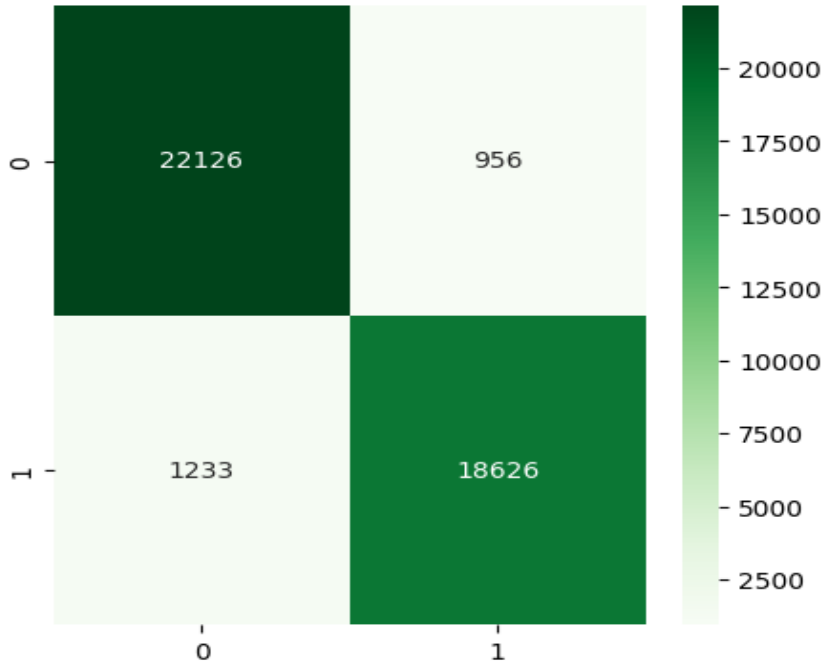
Kôd 30. Ispis konfuzijskih matrica

Izvor: Autor



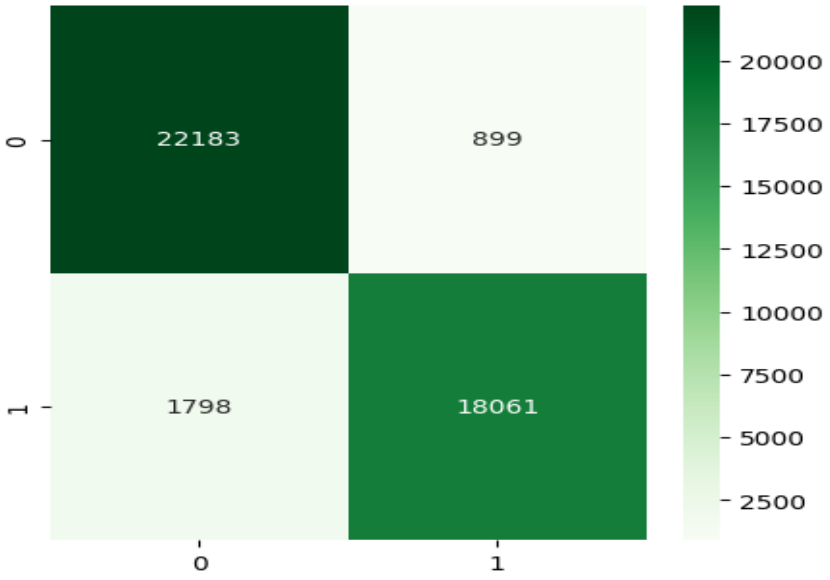
Slika 30. Prikaz konfuzijske matrice za Gaussov Naive Bayes

Izvor: Autor



Slika 31. Prikaz konfuzijske matrice za stablo odlučivanja

Izvor: Autor



Slika 32. Prikaz konfuzijske matrice za kNN

Izvor: Autor

6.2. Višeklasna klasifikacija

U ovom potpoglavlju vrši se višeklasna klasifikacija. Za razliku od prvog modela gdje se koristila klasifikacija kao metoda za odabir podataka, ovdje se bira prema uzročnosti.

U znanstvenom članku iz 2015. godine koristio se KDD'99 *dataset* s ciljem analiziranja i izrade sustava za detekciju probe napada. [14] Ovaj će se model bazirati na istim atributima koji su se koristili za izradu toga modela, a uključuje sljedeće:

Index Number	*Attribute Name*
2,3,4	*Protocol_type, Service, Flag*
23,27,28	*Count, Rerror_rate, Srv_rerror_rate*
29,30,31	*Same_srv_rate, Diff_srv_rate, Srv_diff_host_rate*
32,33,34	*Dst_hosy_count, Dst_host_srv_count, dst_host_same_srv_rate*
35,36,37	*dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate*
38,40,41	*Dst_host_serror_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate*

Slika 33. Atributi korišteni za predikciju probe napada u modelu iz znanstvenog članka

Izvor: Autor

Razlika između ta dva modela jest u tome što će se u modelu izrađenom za ovaj rad prediktirati sve kategorije napada, umjesto samo probe napada.

Prvi korak jest pronaći gdje se *dataframe* nalazio u najboljem stanju te se kopira za dalju transformaciju podataka u svrhu optimiziranja za ovaj model. U priloženom kôdu 21. na stranici 35 ovoga rada izvodilo se izbacivanje napadačkih klasa i ostali su klasificirani napadi i nenapadi. U ovoj situaciji potrebno je obrnuto. Stvara se novi *dataframe* unutar kojeg se metodom *copy()* kopira korišteni *dataframe*. U tom slučaju priloženi kôd 21. sad izgleda ovako:

```
df2 = df.copy()

attack_flag = df.attack.map(lambda a: 0 if a == 'normal' else 1)
df['attack_flag'] = attack_flag
df.drop(['attack_classes', 'attack'], axis=1, inplace=True)
df
```

Slika 34. Modificirani kôd 21.

Izvor: Autor

Nadalje, optimalno je u ovom slučaju ispisati značajke u svrhu pregleda njihovih indeksa. Za razliku od prošlog modela, gdje se moglo *for* petljom izvući korelacijske vrijednosti, ovdje se podatci moraju ručno izbacivati. Najpregledniji je način pozivom info() metode za ispis informacija o *dataframeu*:

```
df2.info()
```

Kôd 31. Ispis informacija o *dataframeu*

Izvor: Autor

```
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   duration                     143134 non-null  float64
 1   protocol_type                143134 non-null  float64
 2   service                      143134 non-null  float64
 3   flag                         143134 non-null  float64
 4   src_bytes                    143134 non-null  float64
 5   dst_bytes                    143134 non-null  float64
 6   land                         143134 non-null  float64
 7   wrong_fragment               143134 non-null  float64
 8   urgent                       143134 non-null  float64
 9   hot                          143134 non-null  float64
 10  num_failed_logins            143134 non-null  float64
 11  logged_in                    143134 non-null  float64
 12  num_compromised              143134 non-null  float64
 13  root_shell                   143134 non-null  float64
 14  su_attempted                 143134 non-null  float64
 15  num_root                     143134 non-null  float64
 16  num_file_creations           143134 non-null  float64
 17  num_shells                   143134 non-null  float64
 18  num_access_files             143134 non-null  float64
 19  num_outbound_cmds            143134 non-null  float64
 20  is_host_login                143134 non-null  float64
 21  is_guest_login               143134 non-null  float64
 22  count                        143134 non-null  float64
 23  srv_count                    143134 non-null  float64
 24  serror_rate                  143134 non-null  float64
 25  srv_serror_rate              143134 non-null  float64
 26  rerror_rate                  143134 non-null  float64
 27  srv_rerror_rate              143134 non-null  float64
 28  same_srv_rate                143134 non-null  float64
 29  diff_srv_rate                143134 non-null  float64
 30  srv_diff_host_rate           143134 non-null  float64
 31  dst_host_count               143134 non-null  float64
 32  dst_host_srv_count           143134 non-null  float64
 33  dst_host_same_srv_rate       143134 non-null  float64
 34  dst_host_diff_srv_rate       143134 non-null  float64
 35  dst_host_same_src_port_rate  143134 non-null  float64
 36  dst_host_srv_diff_host_rate  143134 non-null  float64
 37  dst_host_serror_rate         143134 non-null  float64
 38  dst_host_srv_serror_rate     143134 non-null  float64
 39  dst_host_rerror_rate         143134 non-null  float64
 40  dst_host_srv_rerror_rate     143134 non-null  float64
 41  level                        143134 non-null  int64
 42  attack_classes               143134 non-null  object
```

Slika 35. Informacije o *dataframeu*

Izvor: Autor

Odabrane značajke na kojima će se implementirati model nalaze se na indeksima 1-3 i 23-40. Ciljna varijabla je na indeksu 42. Sljedećim kodom izvrši se izbacivanje svih ostalih značajki kombinacijom ručnog izbacivanja i pomoću *array slice* metode:

```
df2 = df2.drop(df2.columns[4:22], axis = 1)

df2 = df2.drop(['duration', 'level', 'attack'], axis=1)
```

Kôd 32. Izbacivanje neželjenih značajki

Izvor: Autor

Nakon toga podatci su spremni za drugu trening-test podjelu:

```
X = df2.iloc[:,:-1]

y = df2.iloc[:,-1]


X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)
```

Kôd 33. Trening-test podjela za drugi model

Izvor: Autor

U prvom modelu birali su se podatci prema vrijednosti korelacije i znalo se kakva im je *prediktivna moć*. Ovdje se ne zna, tako da je sigurnije testirati na 20 % podataka kako bi model bolje naučio iz trening-seta.

Trebalo bi, prije svega, ponovno pogledati i distribuciju napadačkih klasa. Budući da je *dataset* neuravnotežen s velikim razlikama u distribucijama napada, morat će se po potrebi izvršiti uzorkovanje (eng. *Sampling*). Na taj način će se izjednačiti i ublažiti neuravnoteženost. Može se izvršiti na dva načina:

1. Prekomjerno uzorkovanje (eng. *Oversampling*) duplicira podatke manjinskih klasa u trening-setu. Budući da se dupliciraju podatci, može doći do veće sklonosti (eng. *Bias*).
2. Poduzorkovanje (eng. *Undersampling*) izbacuje podatke iz većinskih klasa da uravnoteži distribuciju. Izbacivanjem podataka može doći do gubitka informacija važnih klasifikatoru za bolju predikciju.

```
df2['attack_classes'].value_counts()
```

Kôd 34. Ispis napadačkih klasa

Izvor: Autor

Rezultat čega je:

```
normal    77053
DoS       53386
probe      8713
R2L        3863
U2R         119
```

Slika 36. Distribucija napadačkih klasa

Izvor: Autor

Prekomjerno uzorkovanje može se izvršiti na dva načina:

1. Nasumičnim prekomjernim uzorkovanjem (eng. Random *Oversampling*) gdje će nasumično duplicirati događaje iz manjinskih klasa.
2. Implementacijom SMOTE-a za izradu sintetičkih podataka

Oba načina uključuju biblioteku *imbalanced-learning*. U ovom slučaju koristit će se SMOTE prema odabiru autora. SMOTE prekomjerno uzorkovanje izradit će sintetičke podatke za svaku klasu dok ne bude svaka klasa imala broj napada kao većinska klasa, u ovom slučaju *normal* klasa. SMOTE se implementira na sljedeći način:

```
from imblearn.over_sampling import SMOTE


X_train_resampled, y_train_resampled =
SMOTE().fit_resample(X_train, y_train)
```

Kôd 35. Implementacija SMOTE-a

Izvor: autor

Iz sintakse je vidljivo da je implementacija SMOTE-a identična podjeli na trening i test-setove. Ključna razlika jest u tome da se izvodi samo na trening-setu.

Ispisat će se *y_train* i gore definirani *y_train_resampled* za uvid u promjenu distribucije:

```
print(y_train.value_counts(), "\n\n",
y_train_resampled.value_counts())
```

Kôd 36. Ispis *y_train* i *y_train_resampled*

Izvor: Autor

rezultat čega jest:

```
normal    61538
DoS       42824
probe      6976
R2L        3076
U2R          93
Name: attack_classes, dtype: int64

 normal    61538
DoS       61538
probe     61538
R2L       61538
U2R       61538
Name: attack_classes, dtype: int64
```

Slika 37. Prikaz nove distribucije napadačkih klasa u trening setu

Izvor: Autor

Ovaj proces je osigurao to da će model moći bolje učiti na manjinskim klasama. Međutim, sve ovisi o tome kolika je frekvencija događaja manjinskih klasa u podatcima za testiranje.

6.2.1. Gaussov Naive Bayes

Proces prediktiranja je identičan prvome modelu. Razlika je u varijablama fit() metode, budući da se koriste uzorkovani podatci.

```
gnb.fit(X_train_resampled, y_train_resampled)
gnb_predict=gnb.predict(X_test)
print(classification_report(y_test, gnb_predict))
```

Kôd 36. Implementacija Gaussovog Naive Bayes algoritma

Izvor: Autor

	precision	recall	f1-score	support
DoS	0.97	0.89	0.93	10562
R2L	0.02	0.11	0.03	787
U2R	0.01	0.92	0.01	26
normal	0.91	0.42	0.57	15515
probe	0.53	0.86	0.66	1737
accuracy			0.61	28627
macro avg	0.49	0.64	0.44	28627
weighted avg	0.89	0.61	0.69	28627

Slika 38. Prikaz klasifikacijskog izvještaja za višeklasni Gaussov *Naive Bayes* algoritam

Izvor: Autor

Na izvještaju je vidljivo da klasifikacija *DoS* napada ima najbolje performanse. Klasificiranje *normal* napada ima preciznost 91 % na 42 % pokrivenih podataka. *Probe* napadima preciznost je prosječna – 53 % na 86 % pokrivenih podataka. R2L napadi imaju najslabije performanse s vrlo slabom preciznošću na vrlo malo podataka. U2R ima slabu preciznost, ali je 92 % U2R podataka pokriveno predikcijom.

6.2.2. Stablo odlučivanja

Stablo odlučivanja također se slično implementira, jedina razlika je u zadanim parametrima *fit()* metode.

```
dtc.fit(X_train_resampled, y_train_resampled)
dtc_predict=dtc.predict(X_test)
print(classification_report(y_test, dtc_predict))
```

Kôd 37. Implementacija stabla odlučivanja

Izvor: Autor

	precision	recall	f1-score	support
DoS	0.99	1.00	1.00	10562
R2L	0.85	0.93	0.88	787
U2R	0.19	0.35	0.24	26
normal	0.99	0.99	0.99	15515
probe	0.99	0.99	0.99	1737
accuracy			0.99	28627
macro avg	0.80	0.85	0.82	28627
weighted avg	0.99	0.99	0.99	28627

Slika 39. Prikaz klasifikacijskog izvještaja za višeklasno stablo odlučivanja

Izvor: Autor

Prema izvještaju stablo odlučivanja ima puno bolje performanse od *Naive Bayes* algoritma. *DoS*, *normal* i *probe* klasifikacija su više-manje savršene. R2L napadi imaju nešto niže performanse, ali još se uvijek smatraju dobrima. U2R ima nešto bolje performanse naspram prethodne situacije, ali jednostavno ima premalo uzoraka u test-setu.

6.2.3. K-sljedećih susjeda

Implementacija je također vrlo slična, samo što uz promjenu u *fit()* metodi treba uzeti u obzir u promjenu u k vrijednosti, budući da trening-set ima puno više vrijednosti zbog uzorkovanja.

```
n=int(math.sqrt(len(X_train_resampled)))

knn.fit(X_train_resampled, y_train_resampled)

knn_predict=knn.predict(X_test)

print(classification_report(y_test, knn_predict))
```

Kôd 38. Implementacija kNN algoritma

Izvor: Autor

	precision	recall	f1-score	support
DoS	0.98	0.97	0.98	10562
R2L	0.43	0.93	0.59	787
U2R	0.03	0.92	0.06	26
normal	0.99	0.88	0.93	15515
probe	0.90	0.98	0.94	1737
accuracy			0.92	28627
macro avg	0.67	0.94	0.70	28627
weighted avg	0.97	0.92	0.94	28627

Slika 40. Prikaz klasifikacijskog izvještaja za višeklasni kNN

Izvor: Autor

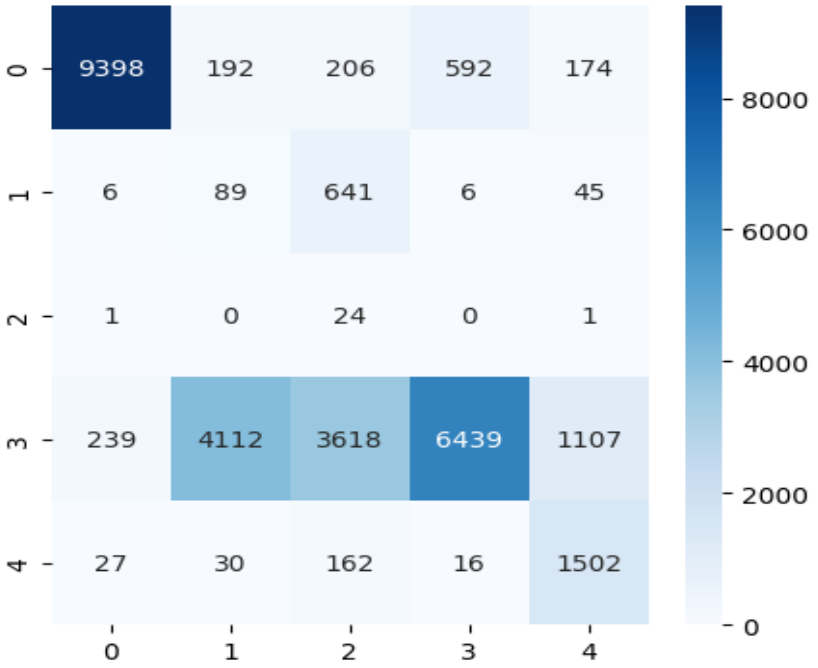
DoS, *normal* i *probe* napadi ponovno imaju skoro savršene performanse. R2L napadi imaju malo slabije performanse naspram stabla odlučivanja. U2R ima ponovno vrlo slabe performanse zbog male količine podataka u test-setu.

Predikcije se stavljaju u polje i ispisuju se preko *for* petlje izrađenom funkcijom za ispis konfuzijske matrice:

```
predictions = [ gnb_predict, dtc_predict, knn_predict ]

for p in predictions:

    plt.figure(figsize=(5, 5))

    conf_matrix(p, 'Blues')
```

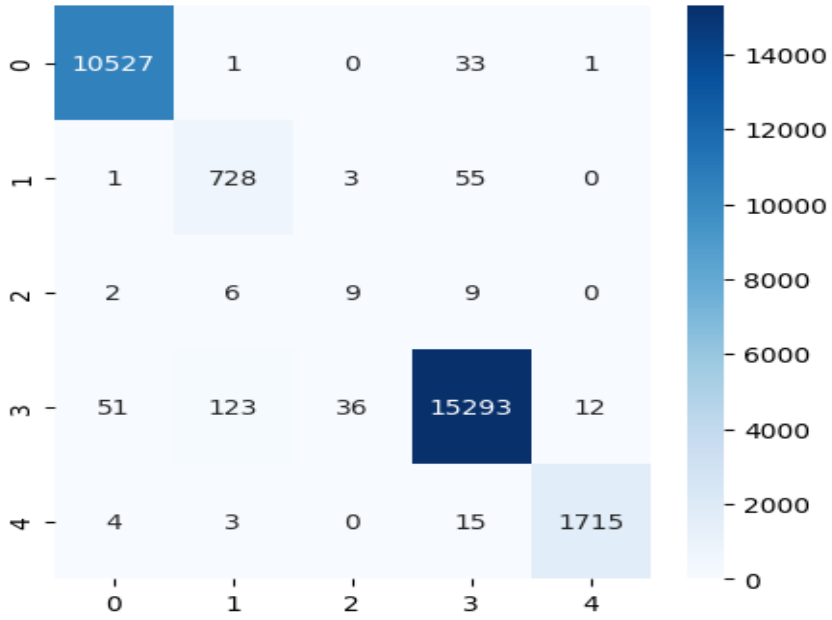
Kôd 39. Kôd za ispis novih predikcija

Izvor: Autor



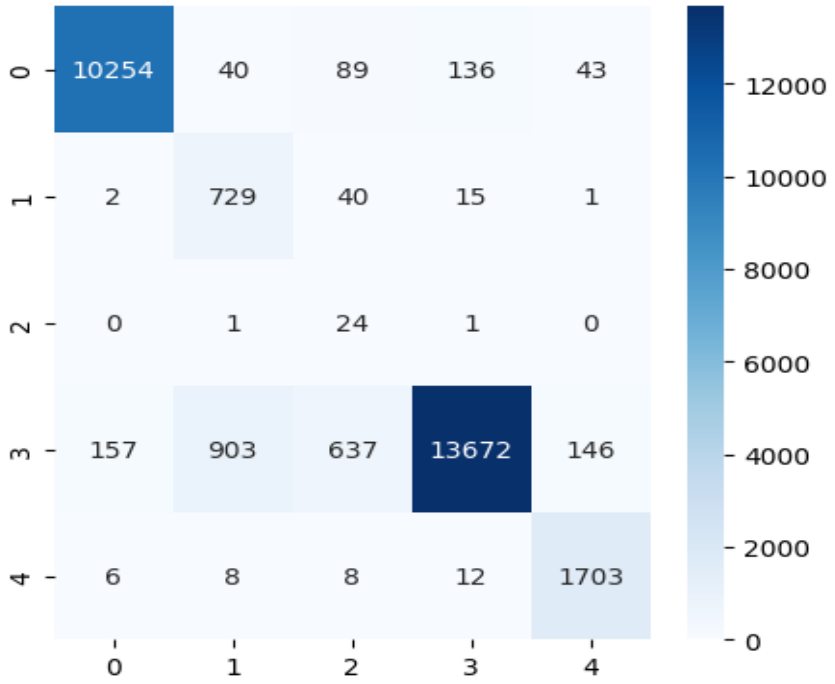
Slika 41. Konfuzijska matrica za višeklasni Gaussov *Naive Bayes*

Izvor: Autor



Slika 42. Konfuzijska matrica za višeklasno stablo odlučivanja

Izvor: Autor



Slika 43. Konfuzijska matrica za višeklasni kNN

Izvor: Autor

7. Zaključak

Dok danas dosta tvrtki na neki način implementira strojno učenje, ono još nije doseglo svoj vrhunac. Strojno učenje pojavilo se kao revolucionarna tehnologija koja je pomogla tvrtkama da povećaju učinkovitost, točnost i rasipanje resursima. Ono postaje sve veći trend u današnjem svijetu i sve više radnika s informatičkom podlogom traži karijeru kao inženjer strojnog učenja.

Cilj ovog rada je bio napraviti model koji na postojećim podatcima može prediktirati sigurnosne napade. Izrađena su dva modela: jedan koji klasificira binarno na podatcima odabranim prema vrijednosti korelacije s napadačkom klasom, i drugi koji višeklasno klasificira prema raznim postotcima serverskih grešaka. Dok je izbor algoritama bitan, veći je izazov naći odgovarajuće podatke i na optimalan način ih iskoristiti.

Jedan je od izazova današnjice pratiti aktualnost starih i biti u toku s novim, kompleksnijim zlođudnim softverom. Danas postoji pregršt sustava za otkrivanje takvih sigurnosnih napada koji pružaju dodatni sloj zaštite, što ga čini kritičnim elementom u *cybersecurity* strategiji.

8. LITERATURA

[1] Brewster C. 16 Examples of Global Companies Using Python in 2023 [Online]. Dostupno na: https://trio.dev/blog/companies-using-python. (30.1.2023.)

[2] Jupyter Notebook. Dostupno na: https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html. (30.1.2023.)

[3] Machine learning, Dostupno na: https://www.sas.com/en_us/insights/analytics/machine-learning.html. (30.1.2023.)

[4] Brownlee, J., (2016). Supervised and unsupervised learning [Online]. Dostupno na: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

[5] Reinforcement learning. Dostupno na: https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/. (30.1.2023.)

[6] Chan, G., (2016). Applications of Reinforcement Learning in Real World [Online]. Dostupno na: https://towardsdatascience.com/applications-of-reinforcement-learning-in-real-world-1a94955bcd12. (30.1.2023.)

[7] Zekić-Sušac, M. (2017). Stabla odlučivanja [Online]. Dostupno na: http://www.efos.unios.hr/sustavi-poslovne-inteligencije/wp-content/uploads/sites/192/2017/10/P4_Stabla-odlucivanja-2017.pdf. (31.1.2023.)

[8] *Naive Bayes classifier* [Online]. Dostupno na: https://en.wikipedia.org/wiki/Naive_Bayes_classifier. (31.1.2023.)

[9] Bayes' Theorem [Online]. Dostupno na: https://corporatefinanceinstitute.com/resources/data-science/bayes-theorem. (31.1.2023.)

[10] Naive Bayes API reference [Online]. Dostupno na: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.naive_bayes. (31.1.2023.)

[11] Christopher, A., (2021). K-Nearest Neighbor [Online]. Dostupno na: https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4. (31.1.2023.)

[12] Algoritam najbližih susjeda [Online]. Dostupno na: https://hr.wiki-code.net/20514147-nearest-neighbors-algorithm. (31.1.2023.)

[13] Radečić, D., (2020). Data scaling for machine learning [Online]. Dostupno na: https://betterdatascience.com/data-scaling-for-machine-learning/. (31.1.2023.)

[14] Ch. Ambedkar, V. Kishore Babu (2015). Detection of Probe Attacks Using Machine Learning Techniques. International Journal of Research Studies in Computer Science and Engineering (IJRSCSE) [Online]. Dostupno na: http://45.113.122.54/pdfs/ijrscse/v2-i3/7.pdf. (12.2.2023.)

Popis slika

Slika 1. Najpopularniji jezici u kolovozu 2021. ... 3
Slika 2. Sučelje Anaconda Navigatora ... 4
Slika 3. Nadzirano učenje ... 7
Slika 4. Nenadzirano učenje ... 8
Slika 5. Podjela algoritama - *cheat sheet* ... 10
Slika 6. Stablo odlučivanja ... 12
Slika 7. Rezultat metode za ispis ... 17
Slika 8. Prikaz prvih pet elemenata *dataframea* nakon spajanja skupova podataka ... 20
Slika 9. Rezultat metode ... 21
Slika 10. Ispis *info()* metode ... 22
Slika 11. Ispis rezultata metode *value_counts()* na ciljnoj varijabli ... 23
Slika 12. Grafički prikaz distribucije napadačkih klasa ... 25
Slika 13. Izgled unakrsne tablice ... 26
Slika 14. Grafički prikaz unakrsne tablice ... 27
Slika 15. Frekvencije servisa ... 28
Slika 16. Prikaz frekvencija značajke *flag* ... 28
Slika 17. Grafički prikaz frekvencija *flag* značajke ... 29
Slika 18. Izgled unakrsne tablice ... 31
Slika 19. Distribucija napadačkih tipova u ICMP protokolu ... 32
Slika 20. Distribucija napadačkih tipova u TCP protokolu ... 32
Slika 21. Distribucija napadačkih tipova u UDP protokolu ... 33
Slika 22. Značajke prije primjene *Label Encodera* ... 35
Slika 23. Značajke nakon primjene *Label Encodera* ... 35
Slika 24. Izgled prvih deset značajki nakon standardiziranja ... 36
Slika 25. Ispis značajki s korelacijom manjom od 0.5 ... 38
Slika 26. Ispis zadovoljavajućih značajki ... 39
Slika 27. Klasifikacijski izvještaj za binarni Gaussov *Naive Bayes* ... 40
Slika 28. Klasifikacijski izvještaj binarnog stabla odlučivanja ... 42
Slika 29. Klasifikacijski izvještaj kNN-a ... 43
Slika 30. Prikaz konfuzijske matrice za Gaussov *Naive Bayes* ... 44
Slika 31. Prikaz konfuzijske matrice za stablo odlučivanja ... 45
Slika 32. Prikaz konfuzijske matrice za kNN ... 45
Slika 33. Atributi korišteni za predikciju *probe* napada u modelu iz znanstvenog članka ... 46
Slika 34. Modificirani kôd 21. ... 46

Slika 35. Informacije o *dataframeu* 48
Slika 36. Distribucija napadačkih klasa 50
Slika 37. Prikaz nove distribucije napadačkih klasa u trening-setu 51
Slika 38. Prikaz klasifikacijskog izvještaja za višeklasni Gaussov *Naive Bayes* algoritam 52
Slika 39. Prikaz klasifikacijskog izvještaja za višeklasno stablo odlučivanja 53
Slika 40. Prikaz klasifikacijskog izvještaja za višeklasni kNN 54
Slika 41. Konfuzijska matrica za višeklasni Gaussov *Naive Bayes* 55
Slika 42. Konfuzijska matrica za višeklasno stablo odlučivanja 55
Slika 43. Konfuzijska matrica za višeklasni kNN 56