Analiza dnevničkih zapisa web poslužitelja s pomoću strojnog učenja

Širić, Marin

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:168:815532

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 3171

ANALIZA DNEVNIČKIH ZAPISA WEB POSLUŽITELJA S POMOĆU STROJNOG UČENJA

Marin Širić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 3171

ANALIZA DNEVNIČKIH ZAPISA WEB POSLUŽITELJA S POMOĆU STROJNOG UČENJA

Marin Širić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 3171

Pristupnik:	**Marin Širić (0036501778)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	izv. prof. dr. sc. Miljenko Mikuc

Zadatak: **Analiza dnevničkih zapisa web poslužitelja s pomoću strojnog učenja**

Opis zadatka:

Dnevnički zapisi web poslužitelja sadrže podatke o pristupima svim web stranicama. Među tim podacima se nalaze i zapisi o pokušajima neovlaštenog pristupa, napadima, potencijalnim prijetnjama i drugim sigurnosnim događajima. Analizom zapisa može se prepoznati sumnjivo ponašanje i poduzeti odgovarajuće mjere zaštite. Pregledavanjem zapisa moguće je i identificirati probleme s performansama poslužitelja te otkriti greške u aplikacijama. Vaš je zadatak istražiti i pokazati mogućnosti primjene strojnog učenja u analizi dnevničkih zapisa web poslužitelja. Na primjerima zapisa s javno dostupnih web poslužitelja, korištenjem vlastitih ili besplatnih alata otvorenog koda demonstrirajte kako algoritmi strojnog učenja mogu poboljšati detekciju sigurnosnih prijetnji, identificirati neobične obrasce ponašanja korisnika te optimizirati performanse web poslužitelja.

Rok za predaju rada: 28. lipnja 2024.

Sadržaj

Uvod

Uz neprekidan rast broja internetskih korisnika i sve složenije web aplikacije, održavanje visoke razine sigurnosti i dobrih performansi web poslužitelja postaje sve izazovnije. Upravo zbog toga, učinkovito nadgledanje i analiza dnevnika web poslužitelja postaju od iznimne važnosti.

Dnevnici web poslužitelja sadrže ključne informacije o svim interakcijama poslužitelja s korisnicima. Ti podaci uključuju detalje o zahtjevima korisnika, odgovorima poslužitelja, korisničkom računalu, statusnim kodovima itd. Dnevnički podaci se analiziraju s ciljem detekcije anomalija, predviđanja potencijalnih sigurnosnih incidenata i optimizacije performansi poslužitelja.

Tradicionalni pristupi analizi dnevničkih zapisa su se često oslanjali na ručnu obradu i jednostavne skripte. Zbog velike količine podataka, ručna obrada je jako neučinkovita, a nije lako automatizirati postupak i pritom zadržati zadovoljavajuću razinu uspješnosti.

U ovom radu bit će istraženo kako strojno učenje može unaprijediti analizu dnevničkih zapisa web poslužitelja. Ovaj pristup bi potencijalno ubrzao i poboljšao učinkovitost procesa nadzora.

Prvi dio ovog rada daje definiciju i strukturu dnevničkih zapisa te postojeće metode i izazove u njihovoj analizi. Zatim su detaljno opisane metodologije strojnog učenje i različiti modeli koji se koriste u ovom kontekstu. U zadnjem dijelu napravljena je konkretna analiza primjene ovih tehnika na stvarnim podacima s javno dostupnih web poslužitelja, te se razmatraju prednosti i ograničenja ovih pristupa.

1. Dnevnički zapisi web poslužitelja

Dnevnički zapisi web poslužitelja su datoteke koje bilježe informacije o svim zahtjevima poslanim prema web poslužitelju te odgovorima koje poslužitelj šalje natrag. Ovi zapisi sadrže različite vrste informacija koje uključuju, ali nisu ograničene na, datum i vrijeme pristupa, IP adrese posjetitelja, vrste zahtjeva (GET, POST itd.), statusne kodove odgovora (kao što su 200 za uspjeh, 404 za nepostojeću stranicu, itd.), veličinu odgovora, URL stranice koja je zahtijevana, informacije o korisnikovom pregledniku (user agent) i slično.[1]

Dnevnički zapisi su korisni za različite svrhe:

- Sigurnost: S pomoću dnevničkih zapisa može se pratiti sumnjivo ponašanje, identificirati pokušaje neovlaštenog pristupa, i analizirati potencijalne sigurnosne prijetnje.
- Analiza prometa: Analiziranjem zapisa moguće je shvatiti obrasce prometa na web stranici, identificirati najpopularnije stranice i vrijeme kada je promet najveći. Analiziranje zapisa također može pomoći pri planiranju kapaciteta računalnih sustava.
- Rješavanje problema: Zapisi pomažu u dijagnosticiranju i rješavanju tehničkih problema na web stranici, kao što su greške u kodu, problemi s poslužiteljem ili spor odziv aplikacija.
- Optimizacija performansi: Analiza zapisa može otkriti uska grla u performansama poslužitelja i pomoći u njihovom optimiziranju za bolje korisničko iskustvo.

Informacije dnevničkih zapisa su relativno korisne pa su često temelj operativnih strategija za upravljanje web poslužiteljima.

[1] '6 Common Log File Formats - CrowdStrike'.

1.1. Struktura dnevničkog zapisa

Postoje različiti standardi za strukturu dnevničkih zapisa.

```
192.168.1.1
-
-
[12/May/2024:08:45:30 +0200]
"GET /index.html HTTP/1.1" 200 1024
"http://example.com/previous_page.html"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36"
```

Kôd 1.1 – Primjer strukture dnevničkog zapisa web poslužitelja

Kôd 1.1 prikazuje primjer strukture dnevničkog zapisa web poslužitelja koji koristi uobičajeni Apache kombinirani format[2] (Combined Log Format). Dnevnički zapis se sastoji od sljedećih elemenata:

1. **IP adresa klijenta**: `192.168.1.1` je IP adresa uređaja koji je poslao zahtjev.
2. **Remote logname**: - ovaj podatak nije dostupan pa je označen crticom
3. **Remote user**: - da je autentikacija bila provedena, ovdje bi se nalazilo korisničko ime, u ovom slučaju nije dostupno
4. **Vrijeme zahtjeva**: `[12/May/2024:08:45:30 +0200]` pokazuje datum i točno vrijeme kada je zahtjev zaprimljen, uključujući vremensku zonu
5. **Zahtjev**: `"GET /index.html HTTP/1.1"` specificira HTTP metodu (GET), ciljani resurs (/index.html) i verziju HTTP protokola (HTTP/1.1)
6. **Statusni kod HTTP odgovora**: `200` označava da je zahtjev uspješno obrađen i da je stranica uspješno vraćena
7. **Veličina odgovora**: 1024 bajta je veličina podataka poslanih klijentu, ne uključujući HTTP zaglavlja

[2] 'Log Files - Apache HTTP Server Version 2.2'.

8. **Referrer**: "`http://example.com/previous_page.html`" pokazuje URL stranice s koje je korisnik došao
9. **User agent**: "`Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36`" identificira preglednik i operativni sustav klijenta

1.2. Postojeće metode

Analiza dnevničkih zapisa web poslužitelja znatno je evoluirala tijekom godina, prvenstveno vođena tehnološkim napretkom, porastom složenosti web okruženja i sve većom potrebom za profinjenijom analizom podataka i sigurnošću.

U početku je analiza log datoteka web poslužitelja bila relativno jednostavna, pri čemu su se logovi poslužitelja uglavnom koristili za rješavanje specifičnih problema. Administratori sustava su ručno pregledavali dnevničke zapise kako bi razumjeli obrasce prometa ili dijagnosticirali greške na poslužitelju.

Krajem 90-ih godina pojavili su se alati poput **AWStatsa**[3] i **Webalizera**[4] koji su nudili osnovnu analitiku, kao što su dnevni posjeti i izvještaje o greškama na poslužitelju.[5]

Početkom 2000. godina, alati su počeli uključivati složenije metrike kao što su trajanje sesije, geografska raspodjela posjetitelja i vršna vremena prometa. To razdoblje obilježila je integracija analize logova s drugim alatima za nadzor, pružajući nešto potpuniji uvid u stanje poslužitelja i ponašanje korisnika.

Alati **Splunk**[6] i **ELK stack**[7] **(Elasticsearch, Logstash, Kibana)** omogućili su praćenje podataka i vizualizaciju u stvarnom vremenu. To je pomoglo rješavati probleme neočekivanog povećanja prometa i sigurnosnih incidenata.

Danas, analiza dnevničkih zapisa često uključuje algoritme strojnog učenja za automatizaciju detekcije anomalija, predviđanje kvarova sustava i optimizaciju performansi

[3] 'AWStats - Open Source Log File Analyzer for Advanced Statistics (GNU GPL)'.
[4] 'Home of The Webalizer'.
[5] 'Log Analysis Explained'.
[6] 'Splunk | The Key to Enterprise Resilience'.
[7] 'What Is Elastic Stack (ELK Stack) - TechTarget.Com'.

poslužitelja. Algoritmi strojnog učenja mogu se koristiti za klasteriranje i klasifikaciju kako bi se grupirale slične vrste zapisa u logovima.

Tehnike korelacije se mogu koristiti u određivanju odnosa između različitih dnevničkih zapisa ili različitih vrsta logova (npr. logovi web poslužitelja, logovi aplikacija i sigurnosni logovi). To je korisno za dijagnosticiranje složenih problema koji mogu obuhvaćati više sustava.

Ako postoje podaci za duži period, mogu se primjenjivati *Time Series* modeli koji mogu biti korisni za uočavanje trendova u pristupu podacima ili grešaka, te za razumijevanje aktivnosti korisnika.

2. Strojno učenje

Strojno učenje je područje umjetne inteligencije koje se bavi izradom modela koji, na osnovu znanja stečenog na viđenim podacima, daju predikciju neke značajke za neviđene podatke. Općenito, model se „uči“ na jednom dijelu podataka tako što podešava brojčane parametre. Zatim koristi jedan dio podataka da evaluira svoje performanse. Ako su rezultati zadovoljavajući, model se potom može koristiti za davanje predikcija za neke potpuno nove podatke.

Ono što strojno učenje čini posebnim u odnosu na klasično programiranje je što koristi indukciju umjesto dedukcije. Dedukcija je zaključivanje specifičnih činjenica iz općih pravila ili aksioma. Indukcija je, pak, donošenje zaključaka o općim pravilima iz specifičnih primjera.

2.1. Vrste strojnog učenja

Strojno učenje možemo podijeliti na nadzirano i nenadzirano učenje, na osnovu toga jesu li podaci unaprijed označeni.

2.1.1. Nadzirano učenje

Nadzirano (en. supervised) učenje podrazumijeva model koji uči iz unaprijed označenih podataka. Algoritam prima ulazne podatke i poznate izlazne podatke, te pokušava naučiti pravilo koje povezuje ulaz i izlaz. Primjeri uključuju klasifikaciju (npr. odlučivanje je li email spam ili ne) i regresiju (npr. predviđanje broja transakcija u trgovini).

2.1.2. Nenadzirano učenje

Kod nenadziranog (en. unsupervised) učenja, podaci nisu označeni i cilj je otkriti skrivene strukture unutar podataka. Primjenjuje se za grupiranje (en. clustering) sličnih primjera unutar podataka, kao u slučaju segmentacije tržišta, ili za smanjenje dimenzionalnosti podataka radi lakše vizualizacije i analize.

2.1.3. Polu-nadzirano učenje

Polu-nadzirano (en. semi-supervised) učenje je hibridna metoda između nadziranog i nenadziranog učenja. Najprije se nenadziranim učenjem grupiraju neoznačeni podaci. Zatim se i ti podaci označavaju. Označeni podaci su često znatno skuplji od neoznačenih, zbog količine vremena koju je potrebno uložiti za označavanje podataka[8].

2.1.4. Učenje s povratnom vezom

Kod učenja s povratnom vezom (en. reinforcement learning), algoritam uči na temelju nagrada koje dobiva nakon svake akcije koju izvrši u dinamičkom okruženju. Cilj je maksimizirati ukupnu nagradu. Ova vrsta učenja često se koristi u robotici, igrama i za navigaciju autonomnih vozila.

2.2. Biblioteke

U nastavku su opisane neke od najvažnijih Python biblioteka korištene u ovom projektu.

2.2.1. NumPy

NumPy je ključna biblioteka za implementaciju matematičkih struktura i operacija u Pythonu. Pruža podršku za velike, višedimenzionalne nizove i matrice, s bogatom kolekcijom matematičkih funkcija za učinkovito manipuliranje tim strukturama. Omogućava brzu obradu numeričkih podataka zahvaljujući implementaciji na nižem nivou programskih jezika kao što su C i Fortran, što rezultira znatnim poboljšanjima u performansama u odnosu na čiste Python strukture poput listi.

2.2.2. Pandas

Pandas je biblioteka za manipulaciju podacima u Pythonu koja se koristi za analizu i obradu podataka. Temelji se na strukturama podataka koje omogućuju brzo i učinkovito manipuliranje nad velikim skupovima podataka. Glavne strukture podataka u Pandas biblioteci su „DataFrame“, koji predstavlja tabularne podatke s mogućnošću imenovanja redova i stupaca i „Series“, jednodimenzionalni niz podataka.

---

[8] ‘Types of Machine Learning - Javatpoint’.

2.2.3. Tensorflow

TensorFlow je open-source biblioteka za umjetnu inteligenciju i strojno učenje. Razvio ju je Google Brain[dodaj ref] tim za internu upotrebu. Koristi se za kreiranje i treniranje različitih vrsta neuronskih mreža za automatsko prepoznavanje uzoraka i odlučivanje koje se temelji na podacima. Ova biblioteka je posebno popularna u područjima koja uključuju duboko učenje, gdje je potrebno obraditi ogromne količine podataka.

2.2.4. Scikit-learn

Scikit-learn je jedna od najkorisnijih Python biblioteka za strojno učenje. Sadrži metode za:

- provođenje klasterizacije, smanjenja dimenzionalnosti, regresije i klasifikacije s pomoću različitih algoritama nadziranog i nenadziranog učenja, metode za ekstrakciju značajki iz slika i teksta,
- pretprocesiranje podataka koje uključuje skaliranje, normalizaciju, centriranje, binarizaciju itd.
- odabir modela i evaluaciju (unakrsna provjera, različiti metrici za mjerenje performansi modela)
- konstrukcija pipeline-a koji objedinjuju niz međusobno povezanih koraka obrade podataka i modeliranja, dizajniranih kako bi automatizirali, standardizirali i pojednostavili proces izgradnje, treniranja, evaluacije i implementacije modela
- dohvaćanje velikog broja testnih setova podataka koji se mogu slobodno koristiti za učenje i istraživanje (Digits, Iris, Diabetes itd.)

2.3. Metode evaluacije rezultata

U ovom poglavlju opisano je razlikovanje primjera prema točnosti klasifikacije. Zatim su objašnjene mjere koje su korištene za evaluaciju modela i matrica zabune, tablica koja omogućava jednostavnu vizualnu procjenu učinkovitosti modela.

2.3.1. Vrste primjera prema točnosti klasifikacije

Prema točnosti klasifikacije, razlikujemo sljedeće slučajeve:

1. **Pravi pozitivni** (True positives, TP) – točno predviđeni pozitivi

2. **Lažni pozitivni** (False positives, FP) – netočno predviđeni pozitivi
3. **Lažni negativni** (False negatives, FN) – netočno predviđeni negativi
4. **Pravi negativni** (True negatives, TN) – točno predviđeni negativi

2.3.2. Mjere evaluacije klasifikatora

Odziv (*recall, sensitivity*) je mjera udjela pozitivnih slučajeva koje klasifikator ispravno predviđa kao pozitivne (1). Na primjer, u medicinskom kontekstu, odziv označava koliko je dobro model sposoban prepoznati sve pacijente koji zaista imaju određenu bolest. U tom slučaju je ta vrijednost najbitnija, jer je cilj da model prepozna što više pozitiva, pod cijenu povećavanja lažnih negativa[9].

$$R = \frac{TP}{TP + FN} \tag{1}$$

Preciznost (*precision*) je mjera udjela ispravno identificiranih pozitivnih slučajeva među svim slučajevima koje je klasifikator označio kao pozitivne (2). Preciznost pokazuje koliko su predikcije modela točne kada model izjavi da je slučaj pozitivan. Ova mjera je posebno važna u situacijama gdje su lažno pozitivni rezultati skupi ili problematični, kao što su pravosudni ili financijski sektor[10].

$$P = \frac{TP}{TP + FP} \tag{2}$$

F1-mjera (*F1-score, F1-factor, F1-measure*) je harmonijska sredina preciznosti i odziva (3). Tako se postiže uravnoteženost ove dvije mjere, osobito u situacijama kad jedna od njih ima izrazito visoke ili niske vrijednosti, što bi moglo dovesti do pogrešne ako se koristi aritmetička ili geometrijska sredina. Da bi F1-mjera bila visoka, i preciznost i osjetljivost moraju biti visoke[11].

$$F_1 = \frac{2PR}{P + R} \tag{3}$$

[9] Powers, 'Evaluation'.
[10] Powers.
[11] 'Sklearn - Precision, Recall, F-Score, Support'.

Support je veličina koja ukazuje na broj primjera neke klase unutar skupa podataka. Support može pomoći razumjeti koliko je pouzdan rezultat evaluacije modela za svaku klasu. Ako ne postoji dovoljan broj primjera za neku klasu, mjere točnosti mogu biti manje pouzdane. U tom slučaju, moguće je da model nije naučio kako pravilno klasificirati primjere te klase[12].

Točnost (*accuracy*) je omjer broja točno klasificiranih primjera i ukupnog broja primjera (4). Ova mjera može biti zavaravajuća ako postoji velika neuravnoteženost između klasa. Na primjer, ako u skupu podataka postoji 95% primjera jedne klase i 5% primjera druge klase, točnost modela koji uvijek predviđa dominantnu klasu će biti 95%.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Makro-F1 je mjera prosječnih performansi klasifikacijskog modela (5) koja se često koristi u višeklasnoj klasifikaciji, pogotovo kada su klase neravnomjerno zastupljene. Izračunava se aritmetička sredina F1-mjere modela za svaku klasu, bez uzimanja u obzir „support" vrijednosti. Tako svaka klasa jednako doprinosi ukupnom prosječnom rezultatu, što može dati jasniju sliku performansi modela na manje zastupljenim klasama.

$$F_1^{macro} = \frac{1}{K}\sum_{i=1}^{K} F_i \tag{5}$$

U formuli (5), K je broj klasa, dok F_i predstavlja F-mjeru klase i.

Može se uzeti i težinski prosjek (*weighted average*). Time se osigurava da klase s većim brojem primjera imaju veći utjecaj na ukupnu metriku nego klase s manjim brojem primjera.

2.3.3. Matrica zabune

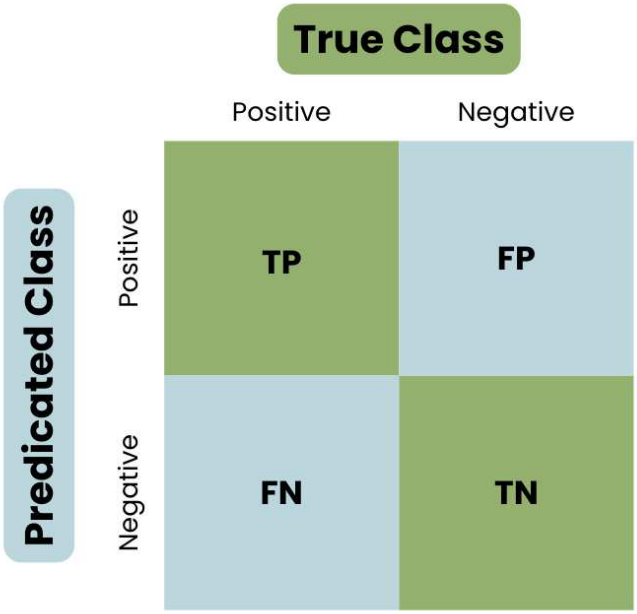
Matrica zabune je specifična vrsta tablice kontingencije koja se koristi za procjenu učinkovitosti modela klasifikacije. U problemu s dvije klase, ova matrica je rešetka dimenzija 2x2 u kojoj svaki stupac predstavlja stvarne klase, a svaki red predstavlja klase koje je predvidio model[13].

[12] 'Sklearn - Precision, Recall, F-Score, Support'.
[13] Provost and Fawcett, *Data Science for Business*.

U matrici zabune su, prema točnosti klasifikacije, primjeri prikazani na sljedeći način (Slika 2.1):

- Pravi pozitivni – gore, lijevo
- Lažni pozitivni – gore, desno
- Lažni negativni – dolje, lijevo
- Pravi negativni – dolje, desno

Dakle, glavna dijagonala sadrži točno klasificirane primjere, dok sporedna dijagonala sadrži netočne. Zbog još bolje vizualizacije, intenzitet boje određenog kvadranta odgovara broju primjera u toj kategoriji.



Slika 2.1 – Vrste primjera prema točnosti klasifikacije u matrici zabune[14]

[14] 'What Is A Confusion Matrix in Machine Learning?'

3. Modeli

U ovom dijelu rada bit će opisani modeli koji su korišteni za klasifikaciju primjera.

3.1. Logistička regresija

Logistička regresija je statistički model koji se koristi za binarnu klasifikaciju, što znači da predviđa hoće li neki primjer pripadati jednoj od dvije moguće kategorije. Iako naziv sugerira da se radi o regresiji, logistička regresija je zapravo klasifikacijski model. Spada u poopćene linearne modele (*generalized linear models*). To su modeli gdje je neka nelinearna funkcija primijenjena na skalarni umnožak vektora težina i vektora značajki.

1. **Model**

   Logistička regresija koristi sigmoidnu (logističku) funkciju kako bi transformirala linearno kombinirane ulazne varijable u vrijednost između 0 i 1. Sigmoidna funkcija je definirana kao:

$$\sigma(x) = \frac{1}{1 + e^{x}} \tag{6}$$

   Primjenjivanjem sigmoide na vektor težina i vektor značajki dobit ćemo izraz za izlaz modela logističke regresije:

$$h(x, w) = \sigma(w^{T}x) = \frac{1}{1 + e^{w^{T}x}} \tag{7}$$

2. **Funkcija gubitka**

   Logistička regresija za funkciju gubitka koristi pogrešku unakrsne entropije (*cross-entropy loss*):

$$E(w) = -\frac{1}{N}\sum_{n=i}^{N} y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i) \tag{8}$$

U jednadžbi (3) y_i predstavlja stvarnu oznaku klase dok je $\hat{y}_i$ predviđena oznaka za primjer i.

3. **Optimizacijski postupak**

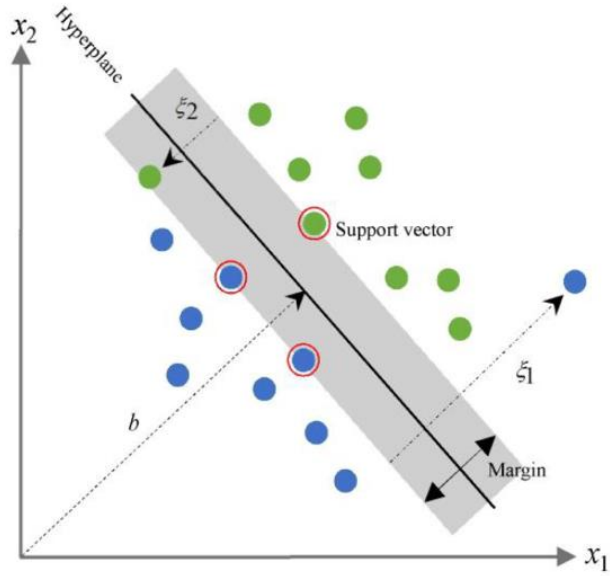
   Funkcija gubitka se minimizira s pomoću gradijentnog spusta. To je iterativna metoda koja gradijent funkcije gubitka u odnosu na parametre modela i ažurira te parametre u smjeru suprotnom od gradijenta kako bi se postigao minimum.

3.2. Random forest

Random Forest je ansambl metoda strojnog učenja koja kombinira rezultate više stabala odluke (*decision tree*) kako bi poboljšala predikcijsku točnost i kontrolirala problem prenaučenosti (*overfitting*).

3.3. Support Vector Machine (SVM)

Algoritam SVM se temelji na pronalaženju optimalne hiperravnine koja razdvaja klase u prostoru značajki s maksimalnom marginom.



Slika 3.1 – Hiperravnina i margine u modelu stroja potpornih vektora[15]

[15] 'Fig. 1. Illustration of Support Vector Machine (SVM) to Generalize The...'

Slika 3.1 pokazuje kako hiperravnina odvaja primjere različitih klasa (točke obilježene različitim bojama i oblicima). Radi jednostavnosti, slika je prikazana u 2D prostoru.

U n-dimenzionalnom prostoru, hiperravnina je potprostor dimenzije $n-1$. U 2D prostoru to je jednodimenzionalni potprostor, odnosno pravac.

Margina je udaljenost između hiperravnine i najbližih točaka iz svake klase koje se nazivaju potporni vektori. SVM teži maksimiziranju margine kako bi osigurao što bolju separaciju između klasa.

Model SVM-a je obični linearni model:

$$h(x, w) = w^T x \tag{9}$$

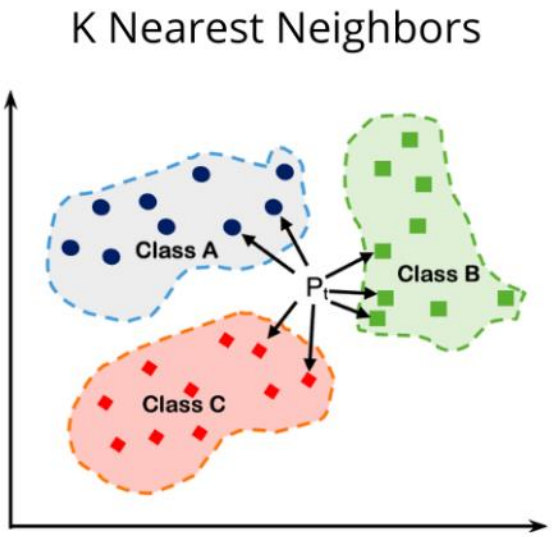
Optimizacijski problem svodi se na minimizaciju funkcije (10) uz ograničenja (11).

$$\underset{w}{\operatorname{argmin}} \frac{1}{2} \|w\|^2 \tag{10}$$

$$y_i w^T x^{(i)} \geq 1, i = 1, \ldots, N \tag{11}$$

3.4. K-Nearest Neighbors

Algoritam k najbližih susjeda (*k-nearest neighbors, k-NN*) se temelji na ideji da slični podaci pripadaju istoj klasi ili imaju slične vrijednosti. Spada u neparametarske metode, što znači da se ne gradi eksplicitni model za treniranje, već se cijeli skup podataka koristi za predikciju.



Slika 3.2 – Grupiranje primjera u klase algoritmom k-NN[16]

Za svaki primjer se gleda klasifikacija ili vrijednost njegovih k najbližih susjeda. Primjer poprima poprima većinsku klasu njegovih susjeda, ili prosječnu vrijednost ako se radi o regresiji.

Algoritam k-NN se odvija u sljedećim koracima:

1. **Odabir hiperparametra *k***

   Vrijednost *k* predstavlja broj najbližih susjeda koji će se koristiti za evaluaciju vrijednosti nekog primjera. Taj parametar se može odabrati unaprijed ili optimizirati s pomoću metode unakrsne provjere (*cross-validation*)

2. **Izračunavanje udaljenosti**

   Za svaki novi primjer koji treba evaluirati, izračunava se udaljenost između tog primjera i svih primjera u skupu podataka. Udaljenost se najčešće mjeri s pomoću Euklidske udaljenosti (12), ali se mogu koristiti i druge mjere poput Manhattan udaljenosti, Minkowski udaljenosti i drugih.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{12}$$

3. **Evaluacija**

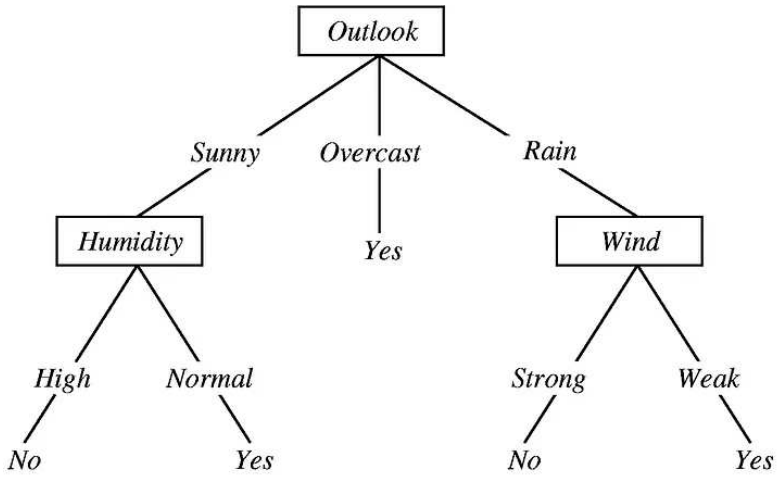
[16] Sachinsoni, 'K Nearest Neighbours — Introduction to Machine Learning Algorithms'.

Nakon izračuna udaljenosti do ostalih susjeda, potrebno je identificirati *k* najbližih susjeda zadanom primjeru. Na osnovu tih udaljenosti se računa vrijednost za zadani primjer. Ako se radi klasifikacija, klasa novog primjera postaje većinska klasa najbližih susjeda. Kod regresije, za vrijednost novog primjera se uzima prosječna vrijednost najbližih susjeda.

Kako bi se postigao učinak da bliži susjedi imaju veći utjecaj na primjer od daljih, doprinos svakog susjeda može se povećati odnosno smanjiti računanjem težinske sume, gdje je težina s kojom se množi pojedina vrijednost jednaka recipročnoj udaljenosti tog susjeda od primjera $\frac{1}{d(p,q)^2}$.

3.5. Decision Tree

Decision Tree (stabla odluke) je algoritam strojnog učenja koji predviđa izlazne vrijednosti na temelju ulaznih značajki koristeći strukturu stabla, gdje svaki unutarnji čvor predstavlja odluku temeljenu na vrijednosti jedne od značajki, a svaki list predstavlja izlaznu vrijednost.[17]



Slika 3.3 – Struktura stabla odluke[18]

Stabla odluke se sastoje od čvorova i grana. Postoje tri vrste čvorova:

[17] Singh, 'Criterion Used in Constructing Decision Tree'.
[18] Yadav, 'Decision Tree in Machine Learning'.

1. **Korijenski čvor** – Početni čvor stabla koji sadrži cijeli skup podataka i prvi uvjet za podjelu
2. **Unutarnji čvorovi** – Predstavljaju uvjete na značajkama podataka koji dijele podatke na dvije ili više grana
3. **Listovi** – Krajnji čvorovi stabla koji predstavljaju izlazne vrijednosti (klase u slučaju klasifikacije ili kontinuirane vrijednosti u slučaju regresije)

Grane povezuju čvorove i predstavljaju rezultat odluke na čvoru, vodeći do sljedećeg čvora ili lista.

Za odabir najbolje podjele čvora koriste se kriteriji poput Gini nečistoće, entropije i informacijske dobiti[19]. Podaci se dijele na temelju odabrane značajke i praga. Proces se rekurzivno ponavlja za svaki nastali podskup podataka sve dok se ne ispuni jedan od kriterija za zaustavljanje (svi podaci u čvoru pripadaju istoj klasi, dostignuta je maksimalna dubina stabla ili nema dovoljno podataka za daljnju podjelu).

3.6. Naive Bayes

Naive Bayes je skup jednostavnih probabilističkih klasifikacijskih algoritama temeljenih na Bayesovom teoremu (13) s pretpostavkom nezavisnosti među značajkama. Iako je pretpostavka nezavisnosti često nerealna u stvarnim podacima, Naive Bayes često daje dobre rezultate, osobito u klasifikaciji teksta i filtriranju neželjene pošte.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{13}$$

- $P(A|B)$ – posteriorna vrijednost hipoteze A s obzirom na podatke B
- $P(B|A)$ – vjerojatnost podataka B s obzirom na hipotezu A
- $P(A)$ – apriorna vjerojatnost hipoteze A
- $P(B)$ – vjerojatnost podataka B

Naive Bayes klasifikator koristi Bayesov teorem za klasifikaciju, čime se zajednička vjerojatnost razloži u umnožak pojedinačnih vjerojatnosti.

[19] 'Growing Decision Trees - MATLAB & Simulink'.

3.7. Gradient Boosting

Gradient Boosting je ansambl metoda koja kreira model tako što iterativno kombinira niz slabijih modela, obično stabla odluke. Osnovna ideja je da svaki novi model popravlja greške prethodnog modela.

Slabi modeli (*base learners*) su jednostavni modeli koji sami po sebi imaju nešto bolju točnost od nasumičnog pogađanja. U praksi se obično koriste plitka stabla odluke kao slabi modeli.

Gradient Boosting gradi modele iterativno, zatim koristi gradijentni spust kako bi minimizirao funkciju gubitka.

Algoritam Gradient Boosting sadrži sljedeće korake:

1. **Inicijalizacija**
   Počinje s jednostavnim modelom, najčešće konstantom koja minimizira funkciju gubitka (npr. srednja vrijednost ciljne varijable u slučaju regresije).
2. **Izračunavanje reziduala**
   Računa se razlika između stvarnih vrijednosti ciljne varijable i predviđenih vrijednosti trenutnog modela. Ta razlika se naziva rezidual.
3. **Treniranje novog modela**
   Novi model se trenira na rezidualima izračunatim u prethodnom koraku. Cilj je minimizirati greške prethodnog modela.
4. **Ažuriranje modela**
   Novi model se dodaje postojećem modelu, s odgovarajućom stopom učenja α koja određuje koliko će novi model pridonijeti konačnom modelu.

$$F_i(x) = F_{i-1}(x) + \alpha h_m(x) \tag{14}$$

5. **Ponavljanje**
   Koraci od 2. do 4. se ponavljaju određen broj puta ili dok se ne postigne zadovoljavajuća točnost modela.

3.8. XGBoost

XGBoost (Extreme Gradient Boosting) je napredni implementacijski okvir Gradient Boosting algoritma na istim osnovnim principima kao i tradicionalni Gradient Boosting, ali uključuje niz optimizacija i poboljšanja koja ga čine učinkovitijim i bržim, posebno za velike skupove podataka.

Od običnog Gradient Boosting algoritma se razlikuje po sljedećim faktorima[20]:

- Koristi regularizaciju (L1 i L2 regularizacija) kako bi smanjio prenaučenost i poboljšao generalizaciju modela
- Uključuje različite optimizacije za povećanje brzine i učinkovitosti poput paralelne obrade i rezanja stabala
- Podržava paralelizaciju, što omogućuje brže treniranje, osobito na velikim skupovima podataka
- Ima ugrađen mehanizam za rješavanje problema nepotpunih podataka (vrijednosti koje nedostaju)

3.9. AdaBoost

AdaBoost (*Adaptive boosting*) je također jedan u nizu ansambl algoritama. Slično kao kod prethodne dvije metode, model se iterativno poboljšava.

Osnovni klasifikator je često panj odluke (*decision stump*), tj. stablo odluke dubine 1.

AdaBoost može biti osjetljiv na šum u podacima i stršeće primjere, jer im može dodijeliti velike težine. Neuravnoteženost klasa može uzrokovati slabije performanse.

AdaBoost radi na sljedeći način:

1. **Inicijalizacija težina**

   Svakom podatku u skupu za treniranje dodjeljuje se početna težina jednaka $\frac{1}{N}$, gdje je N ukupan broj podataka
2. **Iterativno treniranje**

[20] Karale, 'Understanding The Difference Between GBM vs XGBoost'.

U svakoj iteraciji, klasifikator se trenira na cijelom skupu podataka koristeći težine pridijeljene svakom primjeru

3. **Izračunavanje pogreške**

   Nakon svakog koraka treniranja, izračunava se pogreška E_j otežana s w_j.

4. **Pouzdanost klasifikatora**

$$\alpha_j = \frac{1}{2}\ln\frac{E_j}{1-E_j} \tag{15}$$

5. **Ažuriranje težina**

$$\boldsymbol{w_{j+1}^i} \leftarrow \boldsymbol{w_{j+1}^i exp}\left(\alpha_j 1\{h_j(x_i) \neq y_i\}\right) \tag{16}$$

6. **Konačna predikcija**

   Konačna predikcija kombinira predikcije svih slabih klasifikatora težinskim glasanjem.

$$\boldsymbol{h(x) = sign}\left(\alpha_j \boldsymbol{h_j(x)}\right) \tag{17}$$

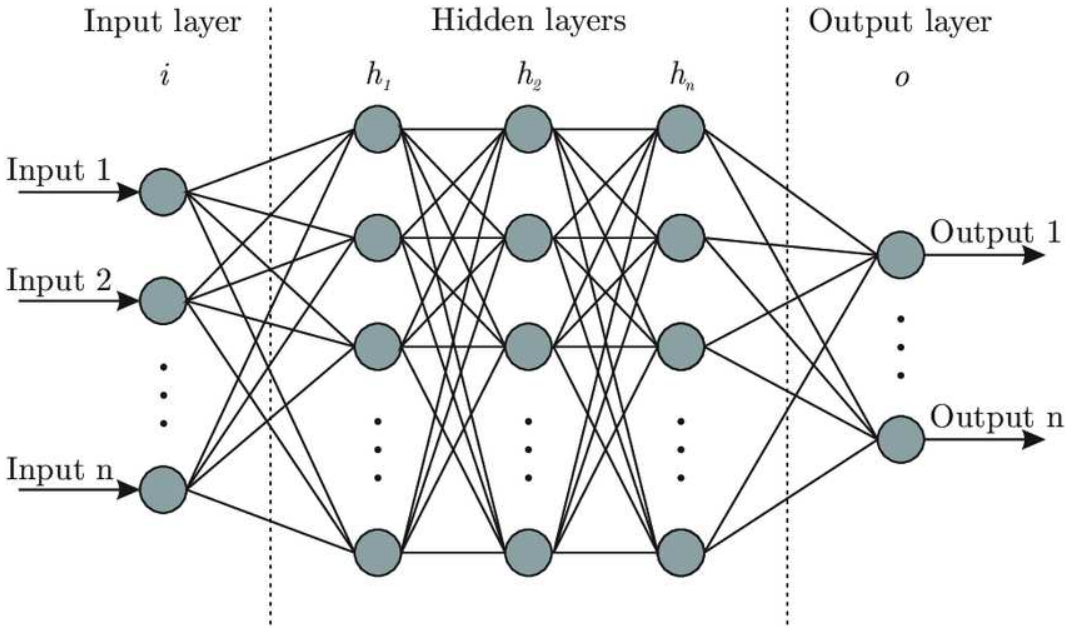
3.10. Artificial Neural Network (ANN)

ANN je model inspiriran strukturom i funkcijom neuronskih mreža ljudskog mozga.

Neuronske mreže izgrađene su od tri sloja (Slika 3.4):

1. **Ulazni sloj**: prihvaća podatke i prosljeđuje ih prema naprijed
2. **Skriveni slojevi**: proizvoljan broj slojeva koji primaju ulaze iz prethodnog sloja, izvode linearne i nelinearne transformacije i prosljeđuju rezultate dalje kroz mrežu
3. **Izlazni sloj**: daje konačne predikcije

Svaki sloj sastoji se od neurona (čvorova) koji su međusobno povezani. Svaka veza ima težinu koja se prilagođava tijekom treniranja.



Slika 3.4 - Slojevi neuronske mreže[21]

Neuroni su osnovne jedinice obrade u mreži. Svaki neuron prima ulazne signale, primjenjuje težine na te signale, zbraja ih i zatim koristi aktivacijsku funkciju kako bi proizveo izlaz. Matematički, izlaz neurona prikazan je u formuli (18).

$$z_j = \sum_i w_{ji} x_i + b_j \tag{18}$$

Težina $\boldsymbol{w_{ji}}$ povezuje neuron **i** u prethodnom sloju s neuronom $\boldsymbol{j}$. Varijabla $\boldsymbol{x_i}$ je ulaz iz neurona $\boldsymbol{i}$, a $\boldsymbol{b_j}$ je pristranost (bias) neurona $\boldsymbol{j}$.

Na izlaz neurona $\boldsymbol{j}$ se još primjenjuje aktivacijska funkcija (19).

$$y_j = f(z_j) \tag{19}$$

Aktivacijske funkcije rade nelinearne transformacije na ulazu, što omogućuje rješavanje kompleksnijih problema. Neke od često korištenih aktivacijskih funkcija su sigmoida (20), tangens hiperbolični (21) i ReLU (*Rectified Linear Unit*).

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{20}$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{21}$$

[21] 'Artificial Neural Network Architecture (ANN)'.

$$f(x) = max(0, x) \tag{22}$$

ANN može naučiti i modelirati nelinearne i složene odnose među podacima. Koristi se za rješavanje raznih problema, od regresije i klasifikacije pa do obrade prirodnog jezika (NLP), računalnog vida itd.

Mane ANN-a su što treniranje može biti računalno intenzivno, osobito za mreže s velikim brojem čvorova. Također, rezultate je teško interpretirati tj. teško je zaključiti na osnovu kojih pravila mreža dolazi do određenog rezultata.

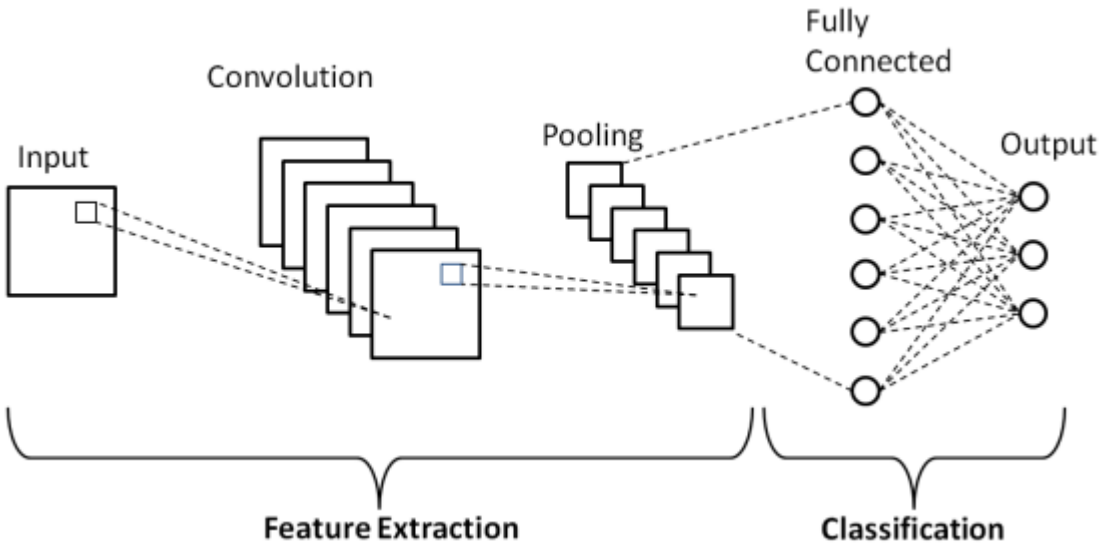
3.11. Convolutional Neural Network (CNN)

CNN (konvolucijska neuronska mreža) je model neuronske mreže koji se pretežito koristi za obradu slika, videozapisa i drugih vrsta podataka koji su organizirani u rešetkastoj strukturi.

Struktura CNN-a se sastoji od tri osnovna sloja (Slika 3.5):

1. **Konvolucijski sloj (*convolutional layer*)**
   Ovaj sloj služi za detekciju lokalnih obrazaca u ulaznim podacima s pomoću operacija koje se nazivaju konvolucije. Primjenjuje se niz filtera (kernela) na ulazne podatke kako bi se izveli neki uzorci ili značajke. Svaki filter stvara mapu značajki (*feature map*) koja sadrži informacije o prepoznatim uzorcima.
2. **Slojevi podotipkavanja (*pooling layers*)**
   Ovi slojevi smanjuju broj dimenzija mapa značajki. Najčešće korištena metoda je maksimalno poduzorkovanje (*max pooling*), koje uzima maksimalnu vrijednost iz manjeg područja mape značajki. Time se smanjuje broj parametara uz zadržavanje najvažnijih informacija.
3. **Potpuno povezani sloj (*fully connecnted layer*)**
   Ovaj sloj povezuje prethodne izdvojene značajke kako bi napravio konačnu predikciju.

Ovaj model često se primjenjuje u svrhe detekcije objekata, računalnog vida, prepoznavanje teksta, analizi medicinskih snimki itd.



Slika 3.5 - Struktura konvolucijske mreže[22]

[22] 'Figure 1. Schematic Diagram of a Basic Convolutional Neural Network...'

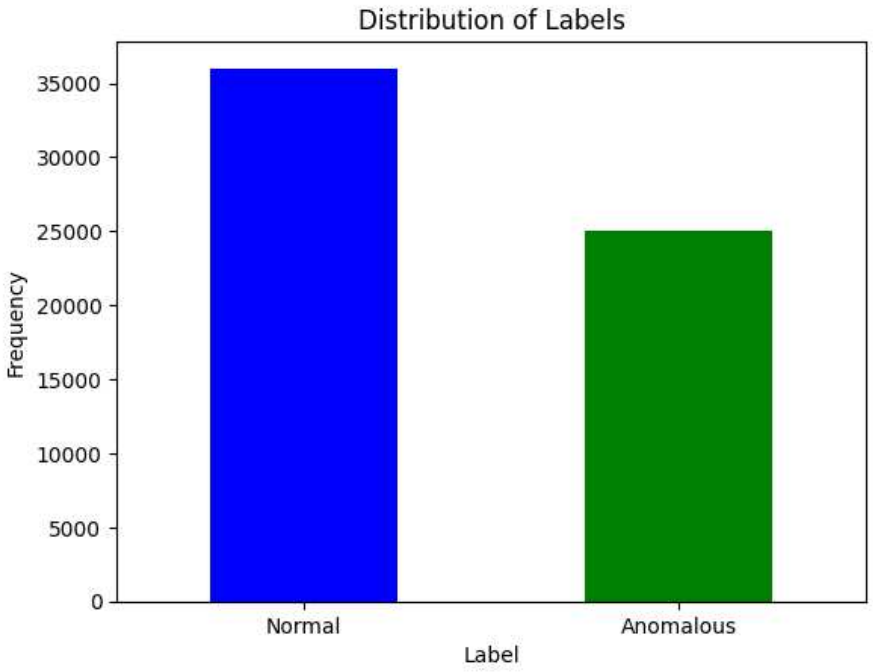
4. Analiza skupa podataka

U ovom dijelu rada koristit ćemo skup podataka CSIC 2010 Web Application Attacks[23], preuzet s platforme Kaggle, koja sadrži veliki broj skupova podataka koji se mogu koristiti u istraživačke svrhe.

Prvo ćemo se osvrnuti na postupak pripreme podataka i odabira značajki (*feature engineering*). Zatim ćemo obraditi postupak treniranja modela.

4.1. Distribucija klasa

Skup podataka koji ćemo koristiti je označen (labeliran). Svaki redak sadrži informaciju o klasifikaciji tog retka u klasu *Normal* ili *Anomalous*. Za dobar rad klasifikacijskih modela bitno je da ni jedna klasa nije prezastupljena ili podzastupljena u odnosu na ostale, odnosno da je broj primjera u svakoj klasi barem približno jednak.



Slika 4.1 – Distribucija klasa u skupu podataka

[23] 'CSIC 2010 Web Application Attacks'.

Slika 4.1 pokazuje distribuciju klasa u skupu podataka. Učestalost klasa nije previše neizjednačena pa nije potrebno raditi poduzorkovanje (*undersampling*) i preuzorkovanje (*oversampling*).

4.2. Odabir značajki

Skup podataka sadrži sljedeće značajke:

- **Label** – kategorička, izlazna varijabla (varijabla koju pokušavamo predvidjeti), može biti *Normal* ili *Anomalous*
- **Method** – HTTP metoda koja se koristi za zahtjev (najčešće su to „GET“ za dohvat resursa, „POST“ za slanje podataka na server, „PUT“ za ažuriranje resursa i „DELETE“ za brisanje resursa
- **User-Agent** – identificira klijentski softver koji pristupa web stranici (web preglednik i operacijski sustav)
- **Pragma** – zastarjelo HTTP zaglavlje koje se koristilo za upravljanje korištenja priručne memorije (*cache*)
- **Cache-Control** – novija implementacija upravljanja priručnom memorijom
- **Accept** – zaglavlje koje pokazuje koje vrste sadržaja klijent može pravilno obraditi (npr. text/html, application/xhtml+xml)
- **Accept-encoding** – pokazuje koji encoding sadržaja klijent može obraditi (npr. gzip, deflate)
- **Accept-charset** – pokazuje koje kodne tablice klijent prihvaća
- **language** – jezici koje klijent prihvaća
- **host** – zaglavlje koje specificira domenu i port na koji je upućen zahtjev
- **cookie** – podaci koje poslužitelj šalje klijentu, a klijent vraća poslužitelju svaki put kad ponovno pristupa stranici (koriste se za upravljanje stanjem i spremanje korisničkih postavki)
- **content-type** – izvorna vrsta podatka koji se šalje (prije encodinga)
- **connection** – određuje hoće li veza ostati otvorena nakon što transakcija završi

- **Content-Length** – veličina tijela zahtjeva u bajtovima
- **content** – tijelo zahtjeva
- **classification** – binarna, izlazna varijabla, ista kao i *Label*, ali s vrijednostima 0 i 1 umjesto naziva klasa
- **URL** – jedinstveni lokator resursa koji je zatražen na poslužitelju, uključujući putanju i parametre upita[24]

Izlazna varijabla koju će klasifikacijski modeli predviđati je *classification.* To je binarna reprezentacija varijable *Label* pri čemu 0 odgovara vrijednosti *Normal*, dok 1 odgovara vrijednosti *Anomalous*. Ostale varijable potrebno je razmotriti kao potencijalne značajke.

Značajke ćemo birati tako da najprije izbacimo sve značajke koje ne mogu biti od pomoći. To su značajke koje imaju jednaku vrijednost za sve primjere i značajke koje imaju jedinstvenu vrijednost za svaki primjer (npr. „cookie“). One ne donose nikakvu informaciju modelu, jer na osnovu njih ne može naučiti nikakve odnose među primjerima.

4.3. Kodiranje značajki

Zbog načina na koji modeli strojnog učenja obrađuju podatke, važno je kodirati značajke prilikom klasifikacije. Većina modela, osobito algoritmi poput regresije, neuronskih mreža i SVM-a očekuju numeričke vrijednosti kao ulazne podatke.

Značajke se mogu kodirati s pomoću metode **LabelEncoder** koja se nalazi u Sklearn biblioteci.

4.4. Optimizacija hiperparametara

Optimizacija hiperparametara u strojnom učenju je proces pronalaženja najboljih kombinacija hiperparametara za zadani model kako bi se postigli najbolji rezultati na skupu podataka. Hiperparametri su varijable koje nisu naučene tijekom treniranja modela, već se

[24] ‘HTTP Headers - HTTP | MDN’.

moraju unaprijed definirati. Različiti modeli imaju različite hiperparametre. Oni kontroliraju ponašanje algoritma učenja i izravno utječu na performanse modela.

Loše odabrani hiperparametri mogu imati za posljedicu podnaučenost ili prenaučenost modela. Loše odabrana stopa učenja može dovesti i do toga proces treniranja traje predugo ili da prilikom optimizacijskog postupka funkcija ne konvergira. Primjeri hiperparametara su stopa učenja, broj skrivenih slojeva u neuronskoj mreži, koeficijent regularizacije, broj stabala, dubina stabla itd.

U ovom radu korištene su sljedeće metode optimizacije hiperparametara[25]:

1. **Pretraživanje po rešetci (grid search)**
   Ova metoda pretražuje unaprijed definirani skup hiperparametara te tako isprobava sve moguće kombinacije. Jednostavna je za implementaciju i, ako su rasponi parametara dobro definirani, može pronaći najbolju kombinaciju. Mana ovog pristupa je što je računalno vrlo intenzivan i može dugo trajati ako postoji velik broj hiperparametara ili širok raspon vrijednosti.
2. **Nasumično pretraživanje (random search)**
   Umjesto isprobavanja svih mogućih kombinacija, random search nasumično odabire vrijednosti hiperparametara unutar definiranih raspona. Brži je od grid searcha jer ne isprobava sve kombinacije, ali vrlo često uspijeva postići istovjetne rezultate.
3. **Ručno podešavanje**

   Ručno podešavanje podrazumijeva iterativni proces testiranja različitih kombinacija hiperparametara na temelju iskustva i intuicije.

Još neke metode koje se često koriste su Bayesovska optimizacija i Hyperband.

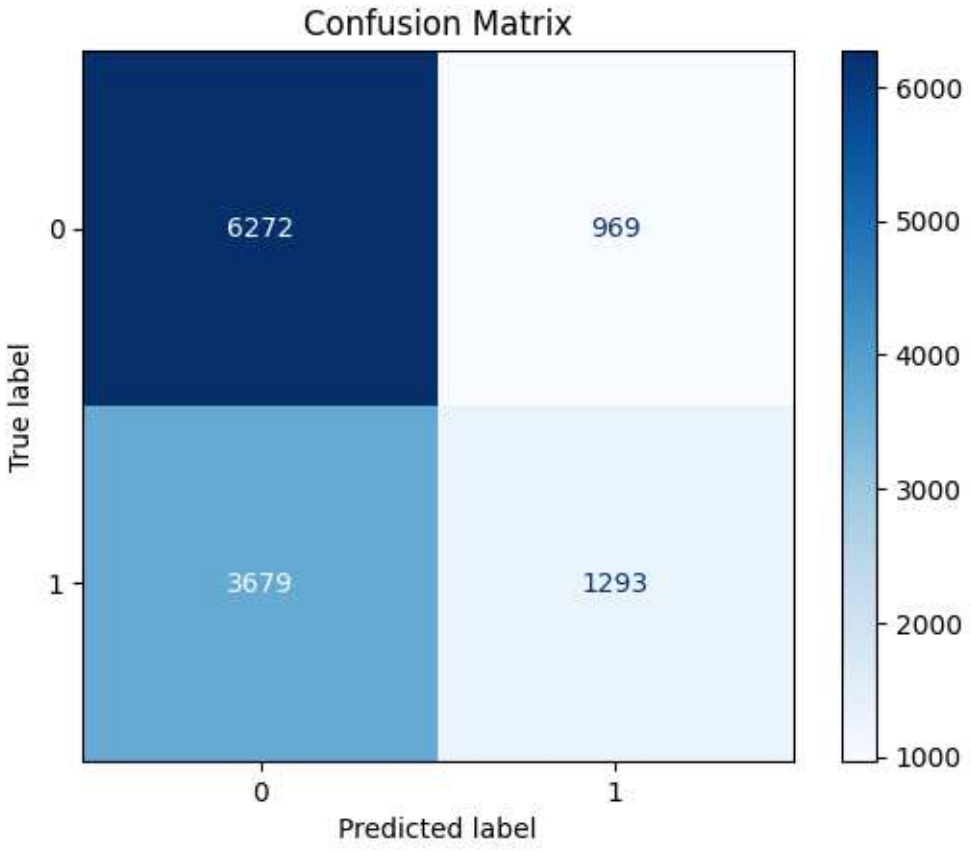
[25] 'Hyperparameter Tuning'.

5. Rezultati treniranja

5.1. Logistička regresija

Tablica 5.1 – Rezultati logističke regresije

	Precision	Recall	F1-score	Support
Class 0 (Normal)	0.63	0.87	0.73	7241
Class 1 (Anomalous)	0.57	0.26	0.36	4972
Accuracy			0.62	12213
Macro average	0.60	0.56	0.54	12213
Weighted average	0.61	0.62	0.58	12213

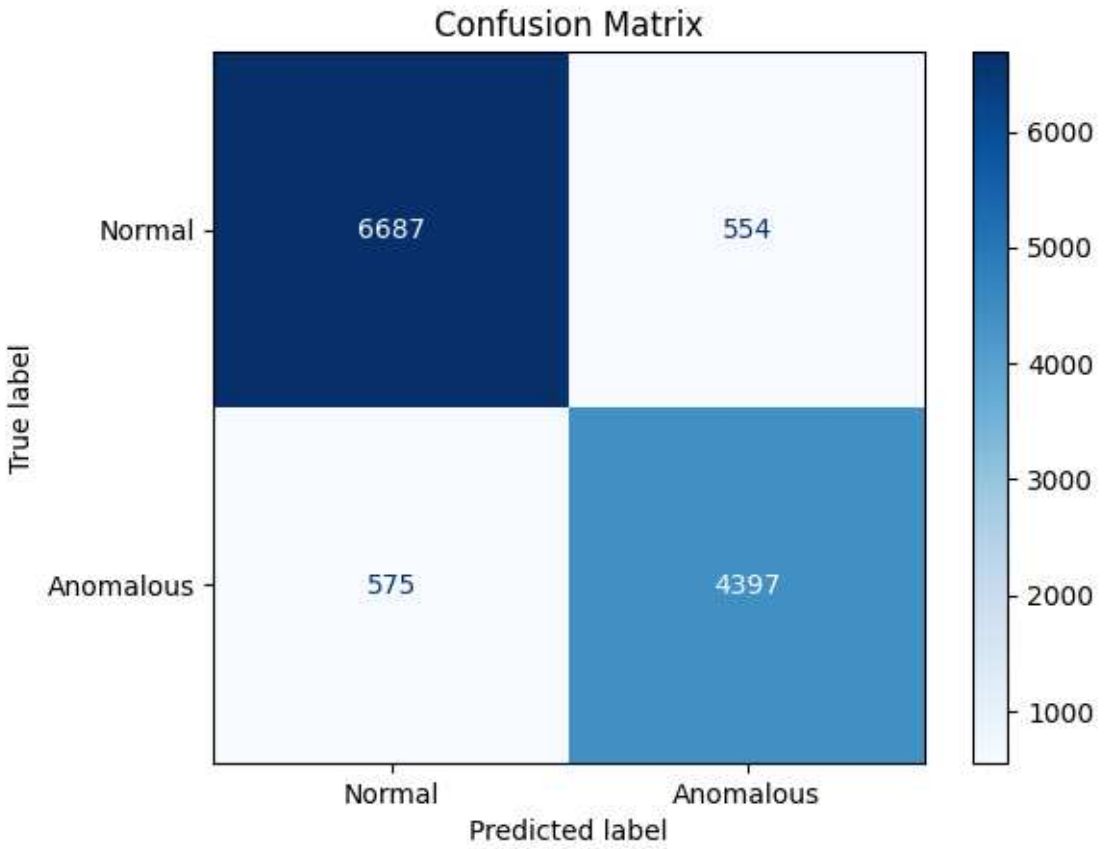


Slika 5.1 – Konfuzijska matrica logističke regresije

5.2. Random forest

Tablica 5.2 - Rezultati random foresta

	Precision	Recall	F1-score	Support
Class 0 (Normal)	0.92	0.92	0.92	7241
Class 1 (Anomalous)	0.89	0.88	0.89	4972
Accuracy			0.91	12213
Macro average	0.90	0.90	0.90	12213
Weighted average	0.91	0.91	0.91	12213

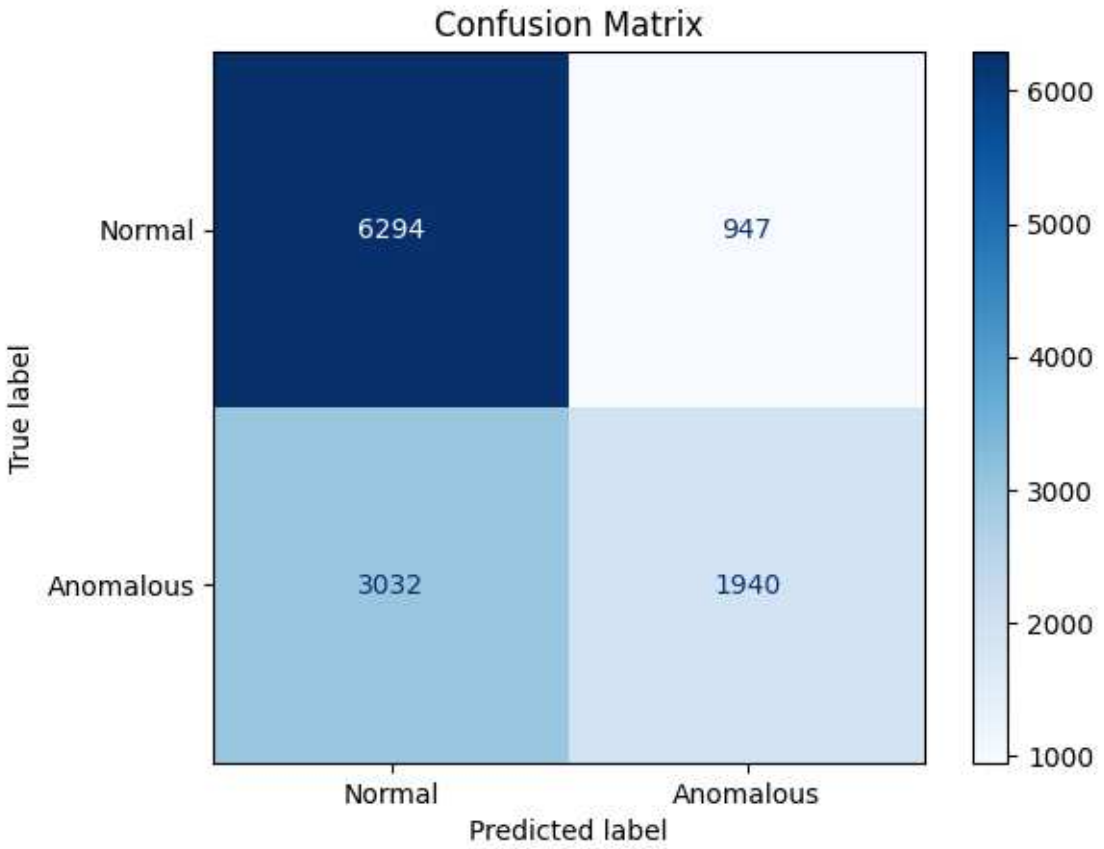


Slika 5.2 – Konfuzijska matrica random foresta

5.3. Support Vector Machine (SVM)

Tablica 5.3 - Rezultati SVM-a

	Precision	Recall	F1-score	Support
Class 0 (Normal)	0.67	0.87	0.76	7241
Class 1 (Anomalous)	0.67	0.39	0.49	4972
Accuracy			0.67	12213
Macro average	0.67	0.63	0.63	12213
Weighted average	0.67	0.67	0.65	12213

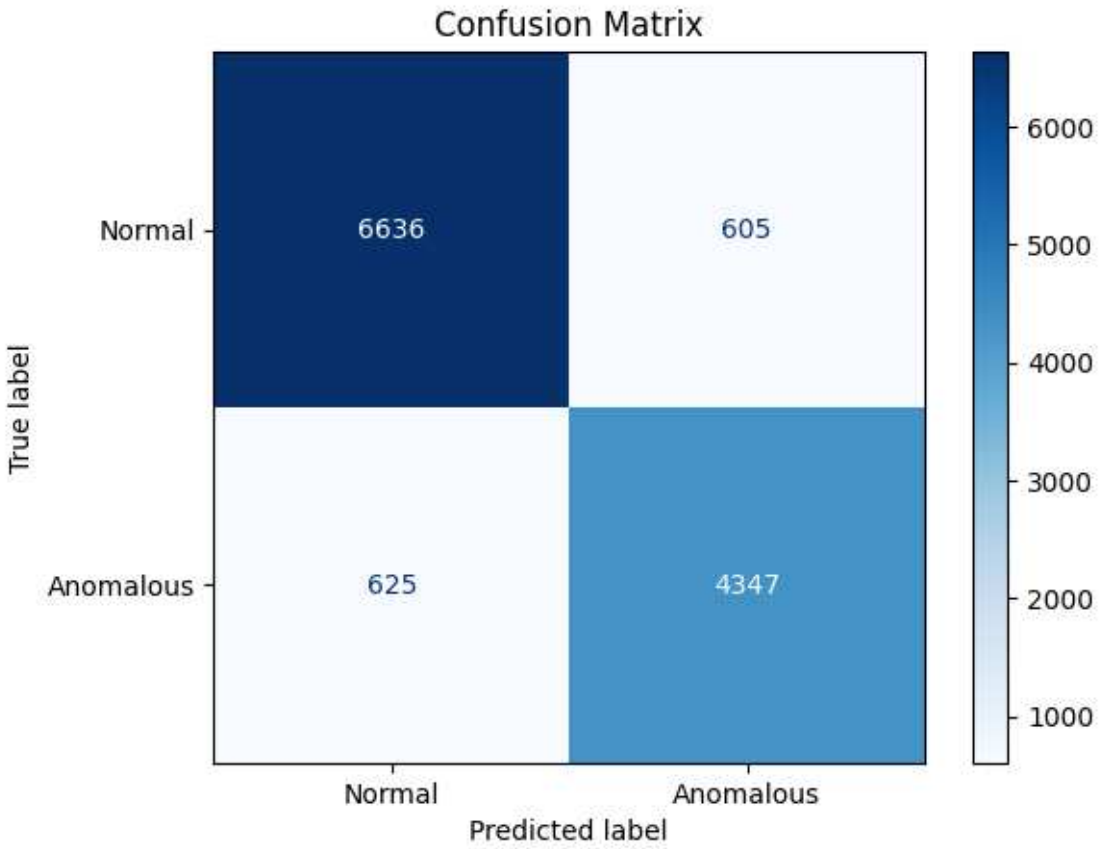


Slika 5.3 – Konfuzijska matrica modela SVM

5.4. K-Nearest Neighbors

Tablica 5.4 - Rezultati KNN-a

	Precision	Recall	F1-score	Support
Class 0 (Normal)	0.91	0.92	0.92	7241
Class 1 (Anomalous)	0.88	0.87	0.88	4972
Accuracy			0.90	12213
Macro average	0.90	0.90	0.90	12213
Weighted average	0.90	0.90	0.90	12213

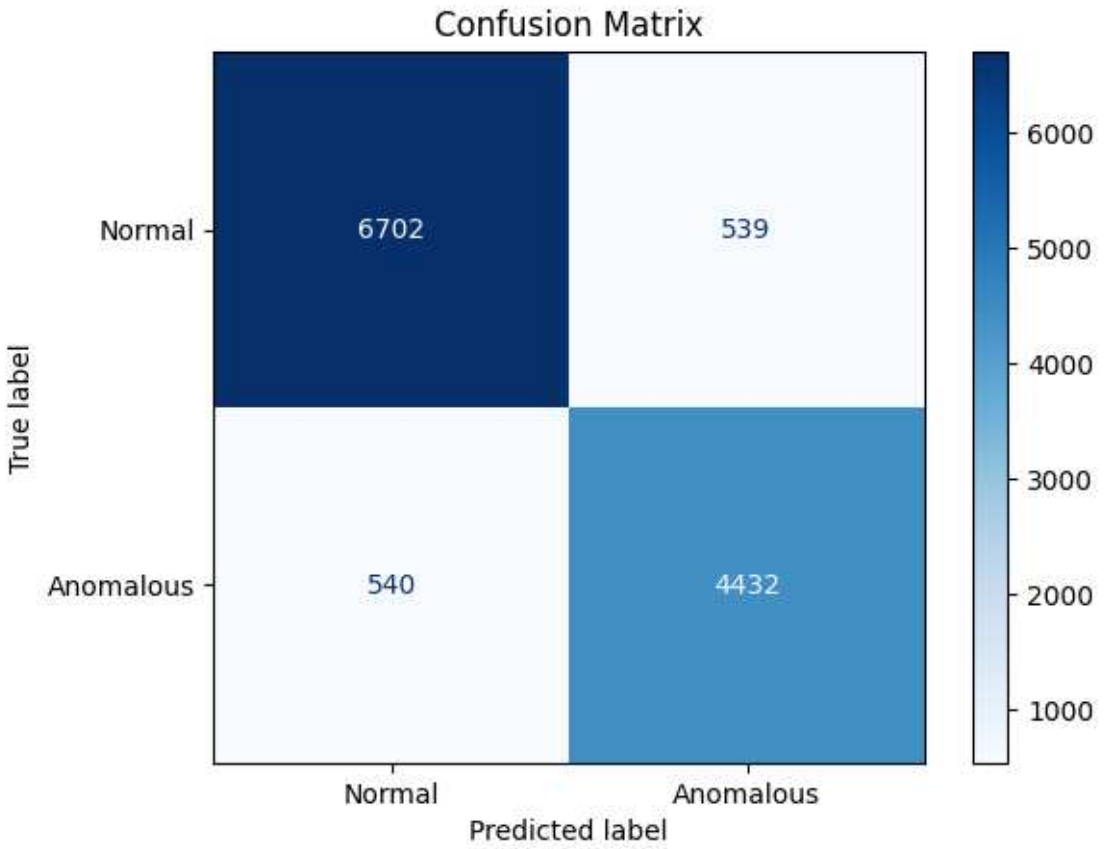


Slika 5.4 – Konfuzijska matrica modela KNN

5.5. Decision Tree

Tablica 5.5 - Rezultati modela Decision Tree

	Precision	Recall	F1-score	Support
Class 0 (Normal)	0.93	0.93	0.93	7241
Class 1 (Anomalous)	0.89	0.89	0.89	4972
Accuracy			0.91	12213
Macro average	0.91	0.91	0.91	12213
Weighted average	0.91	0.91	0.91	12213

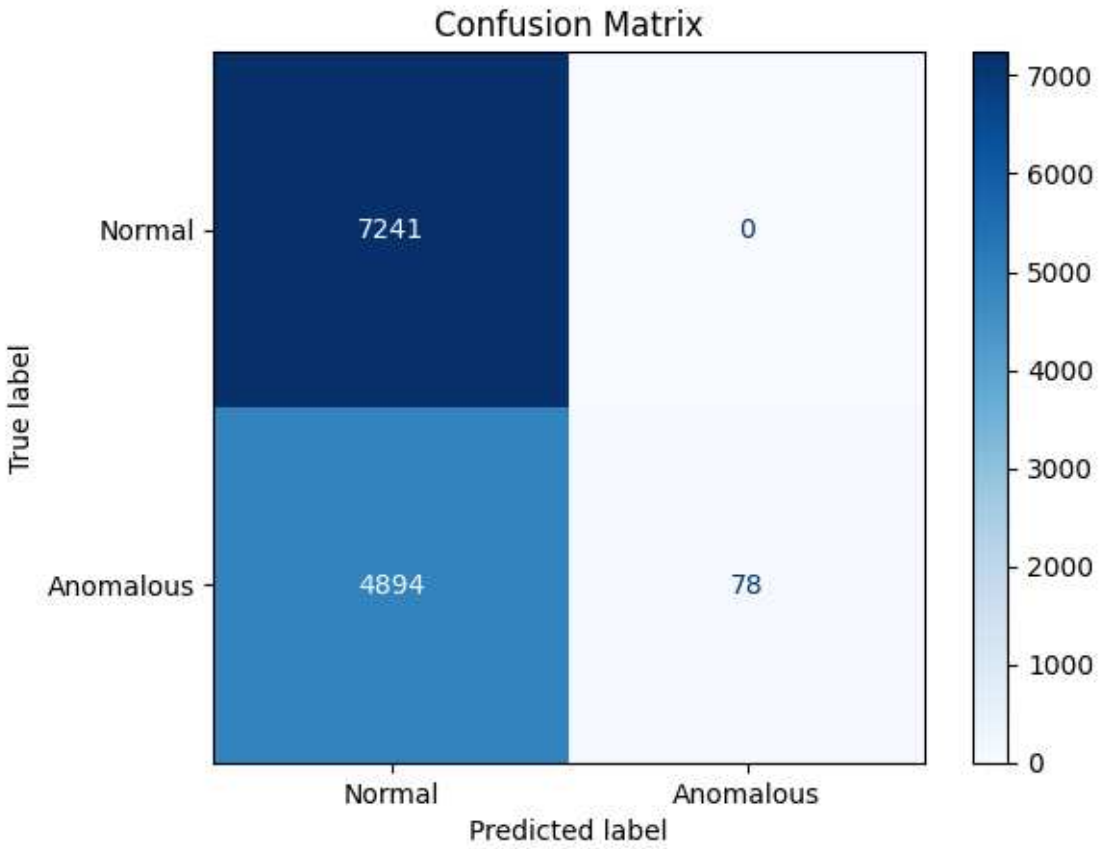


Slika 5.5 - Konfuzijska matrica modela Decision Tree

5.6. Naive Bayes

Tablica 5.6 – Rezultati modela Naive Bayes

	Precision	Recall	F1-score	Support
Class 0 (Normal)	0.60	1.00	0.75	7241
Class 1 (Anomalous)	1.00	0.02	0.03	4972
Accuracy			0.60	12213
Macro average	0.80	0.51	0.39	12213
Weighted average	0.76	0.60	0.46	12213

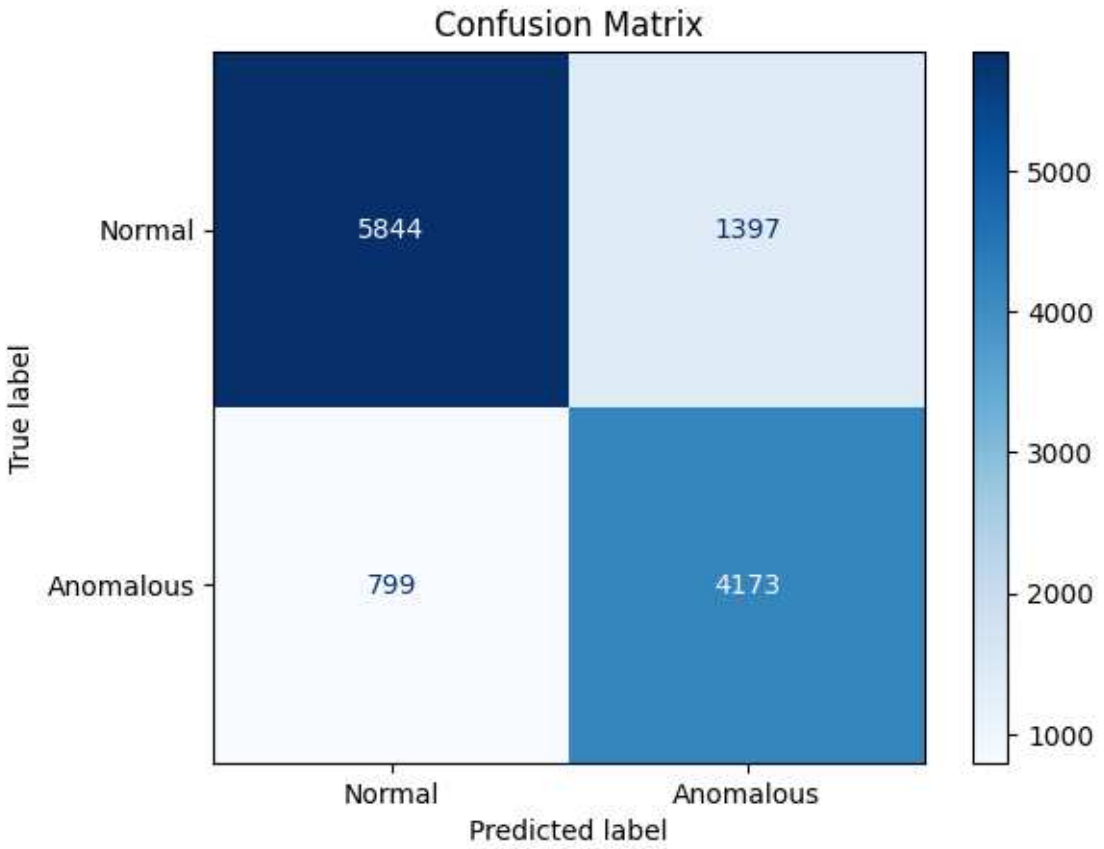


Slika 5.6 – Konfuzijska matrica modela Naive Bayes

5.7. Gradient Boosting

Tablica 5.7 – Rezultati modela Gradient Boosting

	Precision	Recall	F1-score	Support
Class 0 (Normal)	0.88	0.81	0.84	7241
Class 1 (Anomalous)	0.75	0.84	0.79	4972
Accuracy			0.82	12213
Macro average	0.81	0.82	0.82	12213
Weighted average	0.83	0.82	0.82	12213

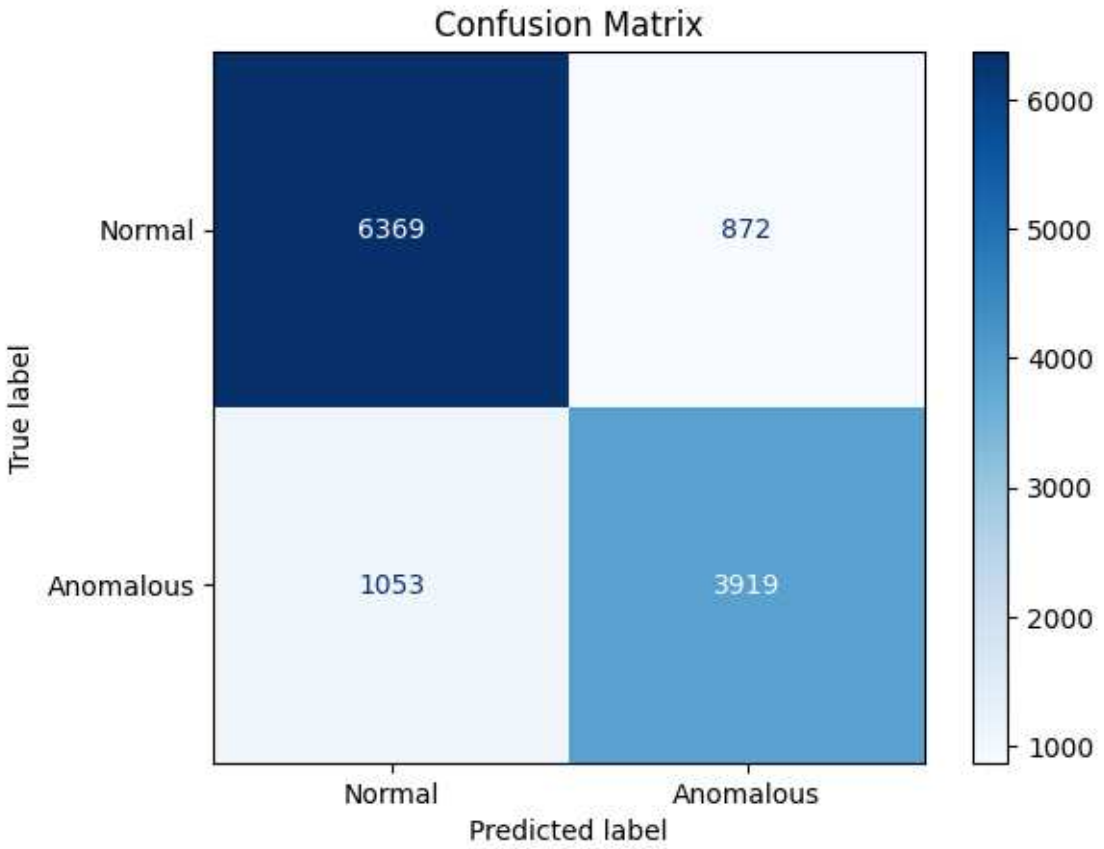


Slika 5.7 – Konfuzijska matrica random foresta

5.8. XGBoost

Tablica 5.8 – Rezultati modela XGBoost

	Precision	Recall	F1-score	Support
Class 0 (Normal)	0.86	0.88	0.87	7241
Class 1 (Anomalous)	0.82	0.79	0.80	4972
Accuracy			0.84	12213
Macro average	0.84	0.83	0.84	12213
Weighted average	0.84	0.84	0.84	12213

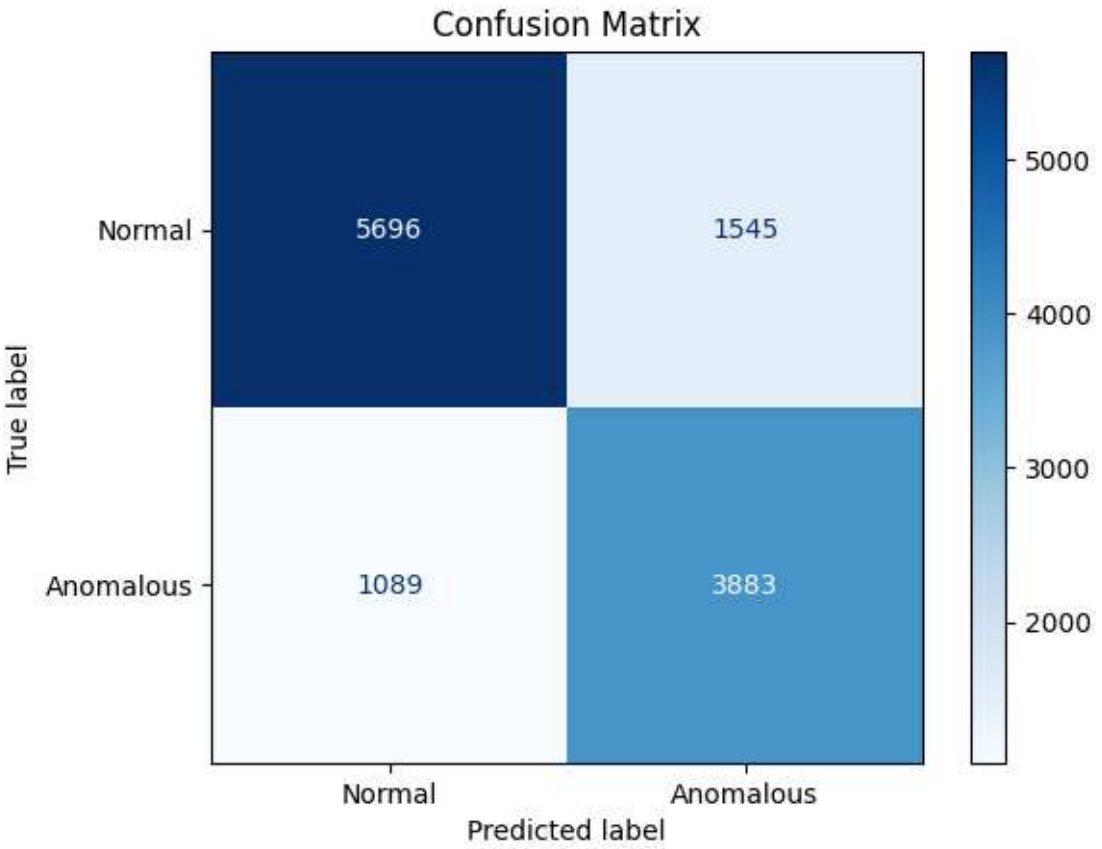


Slika 5.8 – Konfuzijska matrica modela XGBoost

5.9. AdaBoost

Tablica 5.9 – Rezultati modela AdaBoost

	Precision	Recall	F1-score	Support
Class 0 (Normal)	0.84	0.79	0.81	7241
Class 1 (Anomalous)	0.72	0.78	0.75	4972
Accuracy			0.78	12213
Macro average	0.78	0.78	0.78	12213
Weighted average	0.79	0.78	0.79	12213

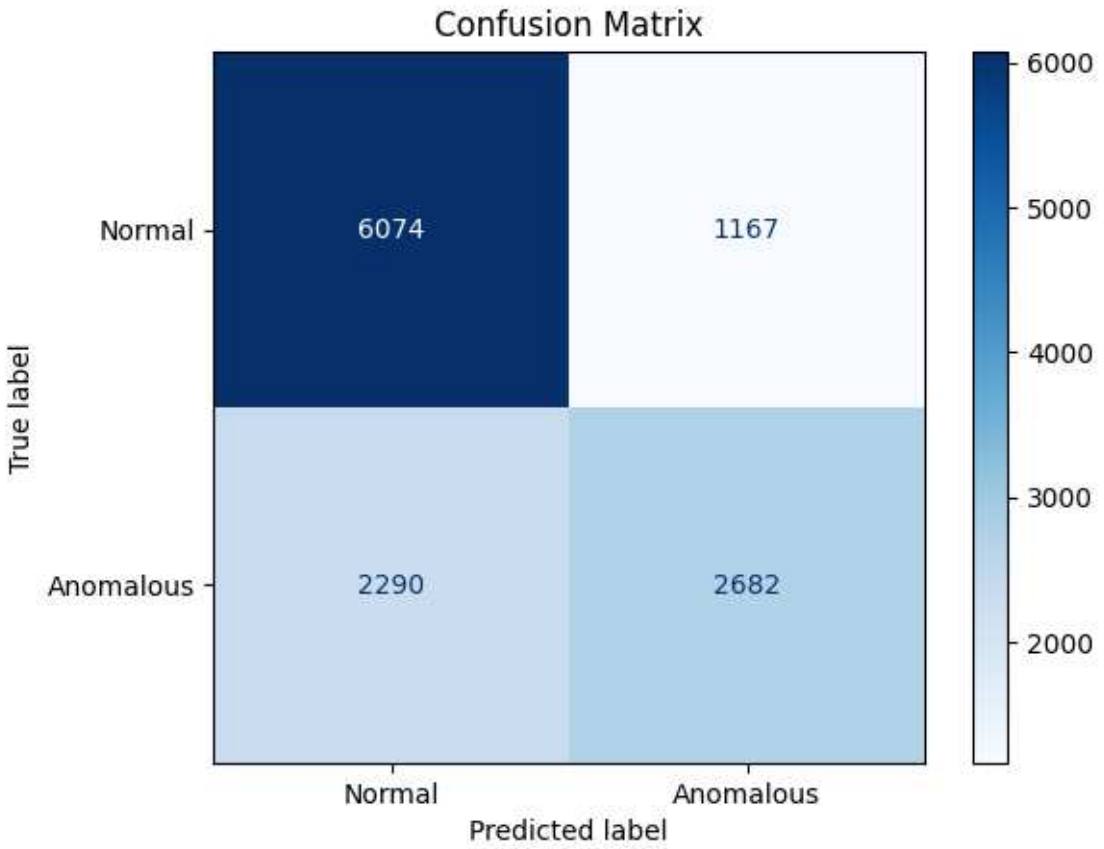


Slika 5.9 – Konfuzijska matrica modela AdaBoost

5.10. Artificial Neural Network (ANN)

Tablica 5.10 – Rezultati modela ANN

	Precision	Recall	F1-score	Support
Class 0 (Normal)	0.73	0.84	0.78	7241
Class 1 (Anomalous)	0.70	0.54	0.61	4972
Accuracy			0.72	12213
Macro average	0.71	0.69	0.69	12213
Weighted average	0.71	0.72	0.71	12213

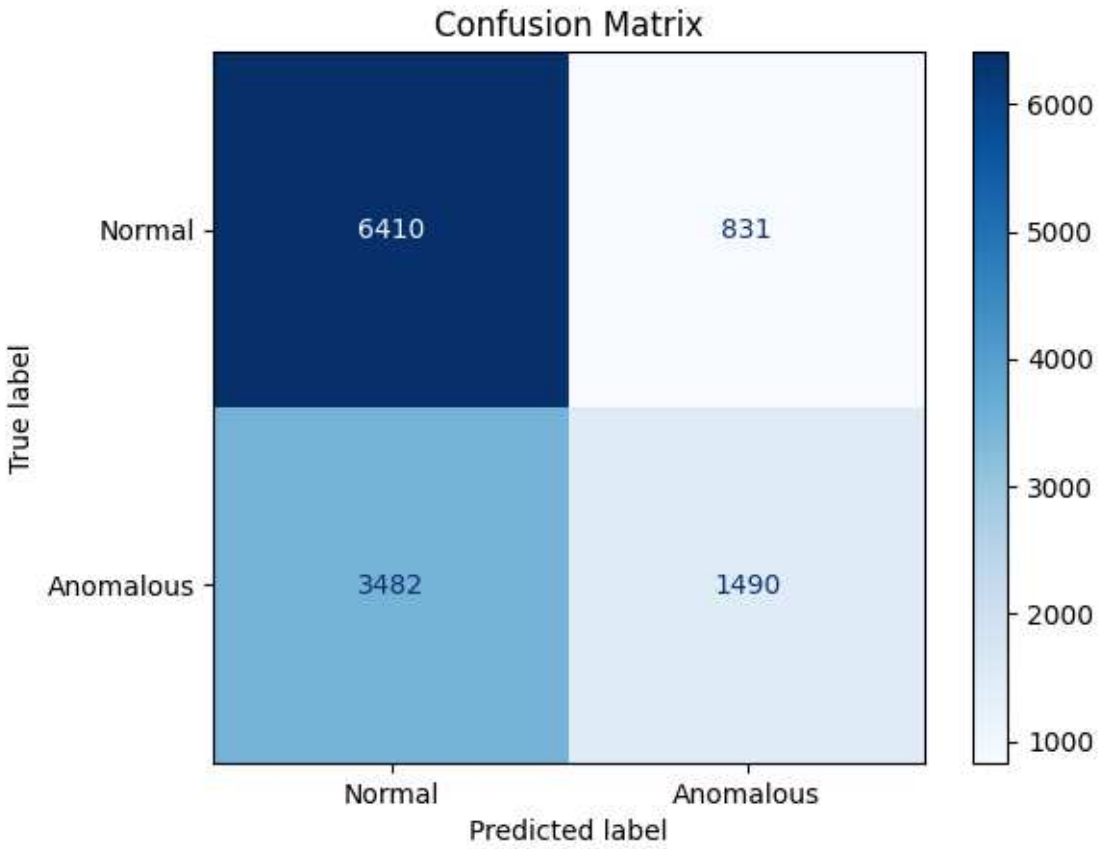


Slika 5.10 – Konfuzijska matrica ANN-a

5.11.Convolutional Neural Network (CNN)

Tablica 5.11 - Rezultati modela CNN

	Precision	Recall	F1-score	Support
Class 0 (Normal)	0.65	0.89	0.75	7241
Class 1 (Anomalous)	0.64	0.30	0.41	4972
Accuracy			0.65	12213
Macro average	0.64	0.59	0.58	12213
Weighted average	0.65	0.65	0.61	12213

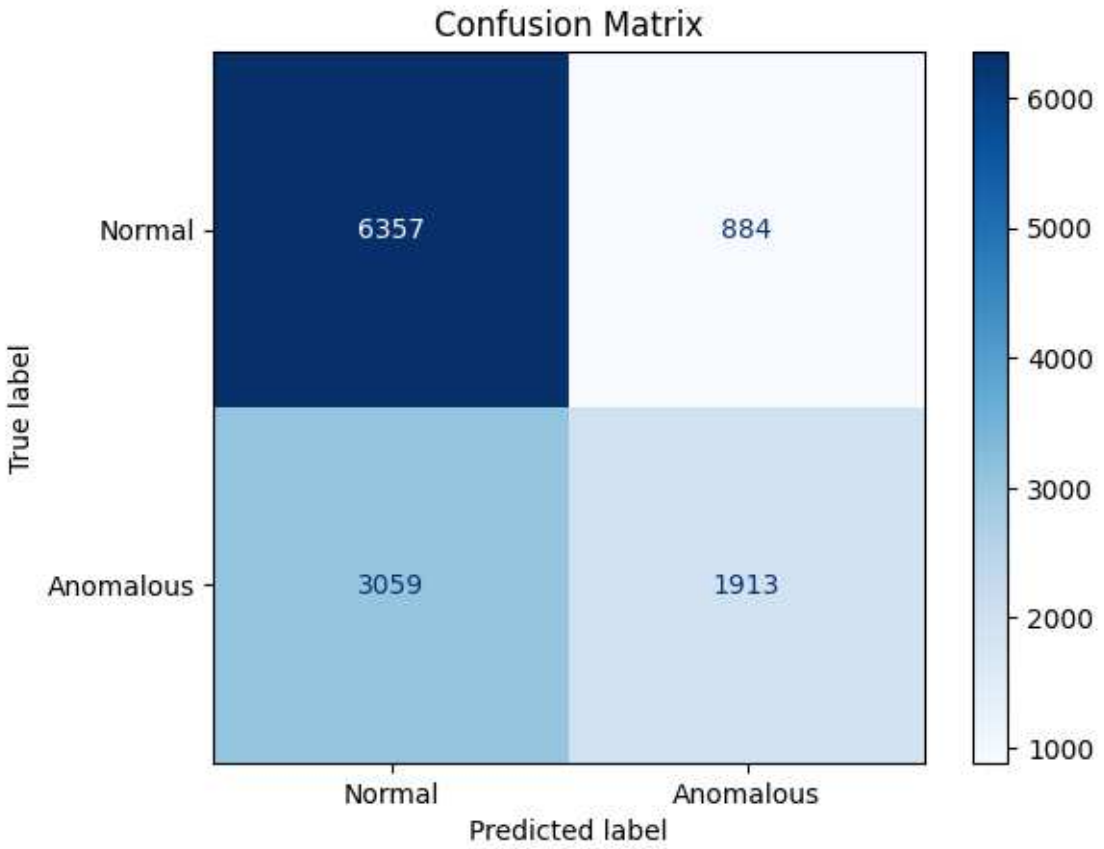


Slika 5.11 – Konfuzijska matrica CNN-a

5.12. Long Short-Term Memory (LSTM)

Tablica 5.12 - Rezultati modela LSTM

	Precision	Recall	F1-score	Support
Class 0 (Normal)	0.68	0.88	0.76	7241
Class 1 (Anomalous)	0.68	0.38	0.49	4972
Accuracy			0.68	12213
Macro average	0.68	0.63	0.63	12213
Weighted average	0.68	0.68	0.65	12213



Slika 5.12 - Konfuzijska matrica LSTM-a

5.13. Primjeri anomalija

5.13.1. Primjeri točne klasifikacije anomalija

Tablica 5.13 – Primjer točno klasificiranog napada „Cross Site Scripting (XSS)“

Method	POST
Host	localhost:8080
Content length	143
Content	'errorMsg=Credenciales+incorrectasparos%2522%2Bstyle%253D%2522background%253Aurl%2528javascript%253Aalert%2528%2527Paros%2527%2529%2529%26id%3D2'
URL	http://localhost:8080/tienda1/publico/entrar.jsp
True label	Anomalous
Predicted label	Anomalous

Ovaj primjer je klasificiran kao anomalija jer predstavlja pokušaj **Cross Site Scripting (XSS)** napada. Vrijednost errorMsg sadrži tekst koji se može interpretirati kao pokušaj ubacivanja JavaScript koda putem CSS-a.

Tablica 5.14 – Primjer točno klasificiranog napada „Server-Side Includes (SSI)“

Method	POST
Host	localhost:8080
Content length	315

Content	'modo=registro&login=%3C%21--%23include+file%3D%22archivo_secreto%22+--%3E&password=pirr7n108&nombre=Siro&apellidos=Casaramon+Tetatzin&email=malcolm_motaung%40enlanzarote.com.yu&dni=30635705G&direccion=Cristo+Rey%2C+59%2C+2B&ciudad=Arenales+de+San+Gregorio&cp=24745&provincia=Huelva&ntc=2796424278292099&B1=Registrar'
URL	http://localhost:8080/tienda1/miembros/editar.jsp
True label	Anomalous
Predicted label	Anomalous

Ovaj POST zahtjev uključuje element koji ukazuje na potencijalnu sigurnosnu prijetnju poznatu kao Server-Side Includes (SSI) eksploatacija. Dekodiranjem **login** parametra, njegova vrijednost postaje **<!--#include file="archivo_secreto" -->**. Ako server ne sanira ulazne podatke prije njihove obrade, ova naredba može omogućiti napadaču učitavanje sadržaja serverskih datoteka što predstavlja ozbiljnu sigurnosnu ranjivost.

5.13.2. Primjer netočne klasifikacije anomalija

Tablica 5.15 – Primjer netočno klasificiranog napada „Cross Site Scripting (XSS)“

Method	POST
Host	localhost:8080
Content length	284
Content	'modo=registro&login=olivia&password=Pa-lTa&nombre=Clidanor&apellidos=Weffer+Vaamonde%22%3E%3C%21--%23EXEC+cmd%3D%22dir+%22--

	%3E%3C&email=dozier%40interanuncios.li&dni=58040403H&direccion=Vista+Alegre+90%2C+&ciudad=Copons&cp=43143&provincia=Valladolid&ntc=4437841608475479&B1=Registrar'
URL	http://localhost:8080/tienda1/publico/registro.jsp
True label	Anomalous
Predicted label	Normal

Analizom content polja može se utvrditi da je riječ o pokušaju XSS-a. Dekodiranjem URL-a vidi se pokušaj ubacivanja HTML komentara <!-- -->, a potom i izvršavanje naredbe **#EXEC cmd="dir".**

Zaključak

Ovaj rad je istražio primjenu različitih klasifikacijskih modela strojnog učenja za detekciju anomalija u dnevničkim zapisima web poslužitelja. Cilj rada bio je identificirati model koji s najvećom točnošću prepoznaje anomalije u velikom broju dnevničkih podataka.

Kroz rad, prvo je opisana struktura dnevničkih zapisa i njihov značaj u sigurnosti i performansima web poslužitelja. Spomenute su i metode koje su do sada bile korištene za analizu dnevničkih podatake te kako su se razvijale vremenom.

Zatim je navedena teoretska podloga za razumijevanje strojnog učenja te najvažnijih biblioteka koje se danas koriste. U posebnom poglavlju opisana je matematička podloga iza svih modela koji su korišteni u istraživanju.

U posljednjem dijelu su opisane metode prikupljanja i pripreme skupa podataka za klasifikaciju, način odabira relevantnih značajki i optimizacija hiperparametara modela. Za svaki model su navedeni relevantni metrici točnosti i dodatno je prikazana konvolucijska matrica koja grafički prikazuje broj primjera koji su točno ili netočno klasificirani. Izdvojeno je nekoliko primjera anomalija koje su točno ili netočno klasificirane.

Rezultati pokazuju da najbolju točnost imaju modeli stabla odlučivanja i slučajnih šuma (91%). Na trećem mjestu je algoritam k-NN (*k* najbližih susjeda) s točnošću od 90%.

Literatura

‘AWStats - Open Source Log File Analyzer for Advanced Statistics (GNU GPL)’, n.d. https://www.awstats.org/.

crowdstrike.com. ‘6 Common Log File Formats - CrowdStrike’, n.d. https://www.crowdstrike.com/cybersecurity-101/observability/log-file-formats/.

‘CSIC 2010 Web Application Attacks’, n.d. https://www.kaggle.com/datasets/ispangler/csic-2010-web-application-attacks.

‘Growing Decision Trees - MATLAB & Simulink’. Accessed 17 September 2024. https://www.mathworks.com/help/stats/growing-decision-trees.html.

‘Home of The Webalizer’, 1 January 2024. https://web.archive.org/web/20240101230033/https://webalizer.net/.

‘HTTP Headers - HTTP | MDN’, 9 September 2024. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers.

‘Hyperparameter Tuning: Examples and Top 5 Techniques’, n.d. https://www.run.ai/guides/hyperparameter-tuning.

isecjobs.com. ‘Log Analysis Explained’, n.d. https://isecjobs.com/insights/log-analysis-explained/.

IT Operations. ‘What Is Elastic Stack (ELK Stack) - TechTarget.Com’, n.d. https://www.techtarget.com/searchitoperations/definition/Elastic-Stack.

Karale, Jayadeep. ‘Understanding The Difference Between GBM vs XGBoost’. *The Talent500 Blog* (blog), 22 April 2024. https://talent500.co/blog/understanding-the-difference-between-gbm-vs-xgboost/.

‘Log Files - Apache HTTP Server Version 2.2’, n.d. https://httpd.apache.org/docs/2.2/logs.html.

Powers, David. ‘Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation’. *Mach. Learn. Technol.* 2 (1 January 2008).

Provost, Foster, and Tom Fawcett. *Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking*. 1. ed., 2. release. Beijing Köln: O’Reilly, 2013.

ResearchGate. ‘Artificial Neural Network Architecture (ANN)’, n.d. https://www.researchgate.net/figure/Artificial-neural-network-architecture-ANN-i-h-1-h-2-h-n-o_fig1_321259051.

ResearchGate. ‘Fig. 1. Illustration of Support Vector Machine (SVM) to Generalize The...’, n.d. https://www.researchgate.net/figure/Illustration-of-support-vector-machine-SVM-to-generalize-the-optimal-separating_fig1_331308937.

ResearchGate. ‘Figure 1. Schematic Diagram of a Basic Convolutional Neural Network...’, n.d. https://www.researchgate.net/figure/Schematic-diagram-of-a-basic-convolutional-neural-network-CNN-architecture-26_fig1_336805909.

Sachinsoni. ‘K Nearest Neighbours — Introduction to Machine Learning Algorithms’. *Medium* (blog), 11 June 2023. https://medium.com/@sachinsoni600517/k-nearest-neighbours-introduction-to-machine-learning-algorithms-9dbc9d9fb3b2.

scikit-learn. ‘Sklearn - Precision, Recall, F-Score, Support’, n.d. https://scikit-learn/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html.

Singh, Deeksha. ‘Criterion Used in Constructing Decision Tree’. *Geek Culture* (blog), 9 March 2022. https://medium.com/geekculture/criterion-used-in-constructing-decision-tree-c89b7339600f.

Splunk. ‘Splunk | The Key to Enterprise Resilience’, n.d. https://www.splunk.com.

‘Types of Machine Learning - Javatpoint’, n.d. https://www.javatpoint.com/types-of-machine-learning.

‘What Is A Confusion Matrix in Machine Learning? The Model Evaluation Tool Explained’, n.d. https://www.datacamp.com/tutorial/what-is-a-confusion-matrix-in-machine-learning.

Yadav, Prince. ‘Decision Tree in Machine Learning’. Medium, 23 September 2019. https://towardsdatascience.com/decision-tree-in-machine-learning-e380942a4c96.

Sažetak

Naslov: Analiza dnevničkih zapisa web poslužitelja s pomoću strojnog učenja

Sažetak: Ovaj rad istražuje primjenu modela strojnog učenja na detekciju anomalija u dnevničkim zapisima web poslužitelja. Nakon analize strukture i sadržaja dnevničkih zapisa, primijenjeni su različiti modeli, uključujući logističku regresiju, slučajne šume (random forest), stabla odlučivanja (decision tree) i neuronske mreže te je evaluirana njihova efikasnost u identifikaciji anomalija. Rezultati su pokazali da modeli stabla odlučivanja i slučajnih šuma ostvaruju najveću točnost, približno 0.91, zahvaljujući njihovoj sposobnosti da efikasno modeliraju složene obrasce u podacima. Zaključak je da algoritmi strojnog učenja potencijalno mogu biti vrlo korisni u analizi anomalija dnevničkih zapisa web poslužitelja.

Ključne riječi: Strojno učenje; Analiza dnevničkih zapisa; Web poslužitelji; Detekcija anomalija; Sigurnosni dnevnički podaci; Nadzirano učenje; Duboko učenje; Feature engineering; Detekcija upada; Optimizacija performansi; Sigurnost web aplikacija; Analiza prometa

Summary

Title: Web server log analysis using machine learning

Summary: This paper explores the application of machine learning models on anomaly detection in web server logs. After analyzing the structure and content of the log files, various models were tested, including logistic regression, random forests, decision trees, and neural networks, and their effectiveness in identifying anomalies was evaluated. The results showed that decision tree and random forest models achieve the highest accuracy, approximately 0.91, thanks to their ability to efficiently model complex patterns in the data. The conclusion is that machine learning algorithms can potentially be very useful in analyzing anomalies in web server logs.

Keywords: Machine Learning; Log Analysis; Web Servers; Anomaly Detection; Security Log Data; Supervised Learning; Deep Learning; Feature Engineering; Intrusion Detection; Performance Optimization; Web Application Security; Traffic Analysis

Privitak

Uz rad je priložena Jupyter bilježnica u kojoj se nalazi programski kod za treniranje i prikaz rezultata svih modela koji su opisani u rezultatima.

Za pokretanje bilježnice potrebno je imati Python 3.11.

Prije instalacije potrebnih modula preporučuje se kreiranje virtualnog okruženja u direktoriju programskog koda:

```
> python -m venv venv
```

Zatim je potrebno aktivirati okruženje i instalirati potrebne module koji se nalaze u *requirements.txt*

```
> ./venv/Scripts/activate

(venv) > pip install -r requirements.txt
```

Pri pokretanju Jupyter bilježnice potrebno je za kernel odabrati python iz gore navedenog virtualnog okruženja.