Učinkovitost treniranja modela predviđanja polu-nadziranog učenja

Krizmanić, Mateo

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij računarstva

Diplomski rad

UČINKOVITOST TRENIRANJA MODELA PREDVIĐANJA POLU-NADZIRANOG UČENJA

Rijeka, rujan 2023.

Mateo Krizmanić
0069073514

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij računarstva

Diplomski rad

UČINKOVITOST TRENIRANJA MODELA PREDVIĐANJA POLU-NADZIRANOG UČENJA

Mentor: Doc. Goran Mauša, dipl. ing.

Rijeka, rujan 2023.

Mateo Krizmanić
0069073514

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 12. ožujka 2021.

Zavod: **Zavod za računarstvo**
Predmet: **Inženjerstvo kompleksnih programskih sustava**
Grana: **2.09.06 programsko inženjerstvo**

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **Mateo Krizmanić (0069073514)**
Studij: Diplomski sveučilišni studij računarstva
Modul: Računalni sustavi

Zadatak: **Učinkovitost treniranja modela predviđanja polu-nadziranog učenja / Training efficiency of semi-supervised learning models**

Opis zadatka:

Proučiti metode polu-nadziranog učenja te opisati njihove prednosti i nedostatke. Izraditi modele polu-nadziranog učenja temeljene na predstavnicima različitih podkategorija strojnog učenja. Vrednovati odabrane modele te izmjeriti utrošak energije, memorije i vremena za scenarije nadziranog i polu-nadziranog učenja. Usporediti dobivene rezultate te istražiti omjer performansi i učinkovitosti u ovisnosti o količini podataka u fazi treniranja.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Krizmanić

Zadatak uručen pristupniku: 15. ožujka 2021.

Mentor:

Doc. Goran Mauša, dipl. ing.

Predsjednik povjerenstva za diplomski ispit:

Izv. prof. dr. sc. Kristijan Lenac

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, rujan 2023.

Krizmanić

Mateo Krizmanić

Zahvala

Zahvaljujem se svom mentoru doc. dr. sc.Goranu Mauši na savjetima, prijedlozima, pomoći i strpljenju prilikom izrade završnog rada.

Također, zahvaljujem se svojoj obitelji, djevojci i prijateljima na podršci tijekom studiranja koji su uvijek vjerovali u mene.

SADRŽAJ

1. Uvod 1
2. Modeli učenja 4
2.1 Nadzirano učenje 4
2.1.1 Kako funkcionira nadzirano učenje? 4
2.1.2 Prednosti nadziranog učenja 5
2.1.3 Mane nadziranog učenja 5
2.1.4 Tehnike nadziranog učenja 6
2.1.5 Primjeri korištenja nadziranog učenja 7
2.2 Polu-nadzirano učenje 7
2.2.1 Metoda samo-učenja 11
2.2.2 Metoda uzajamnog učenja 13
2.2.3 Metoda propagiranja oznaka 16
2.2.4 Algoritam maksimizacije očekivanja 18
2.2.5 Polu-nadzirana metoda potpornih vektora 19
2.2.6 Primjeri korištenja polu-nadziranog učenja 21
2.2.7 Kada koristiti, a kada ne koristiti polunadzirano učenje 22
2.3 Mjerenje utroška energije 22
3. Vrednovanje modela 25
3.1 Unakrsna provjera 25
3.2 Metrike vrednovanja 28
4. Razvojno i testno okruženje 34
4.1 Alati za razvoj modela strojnog učenja 34
4.1.1 Programsko okruženje 35
4.1.2 Python knjižnice za razvoj modela 35
4.1.3 PyRAPL 36
4.2 Testno okruženje 38

5. Rezultati 40
5.1 Opis istraživanja 40
5.1.1 Skupovi podataka 40
5.1.2 Razvoj modela predviđanja 41
5.2 Rezultati istraživanja 46
5.2.1 Skup podataka Tumor 46
5.2.2 Skup podataka Banka 49
5.2.3 Skup podataka 50k 52
5.2.4 Skup podataka Kupon 55
5.2.5 Skup podataka Gljiva 58
6. Zaključak 61
Literatura 64

1. UVOD

Kako tehnologija napreduje, a svjetska populacija raste, raste i količina podataka koje svakodnevno stvaramo. Naš se digitalni otisak produbljuje, kontinuirano se generira i prikuplja sve više podataka, te njihovo sortiranje i analiziranje može biti poprilično težak zadatak. Bilo osobni ili znanstveni, podaci koji samo pasivno leže nisu od nikakve koristi, stoga se svakodnevno pronalaze novi načini kako iskoristiti te podatke i pretvoriti ih u koristan proizvod ili uslugu.

Strojno učenje (eng. machine learning) je grana umjetne inteligencije (eng. artificial intelligence) koja se bavi oblikovanjem algoritama koji poboljšavaju svoju učinkovitost na temelju empirijskih podataka. Sustavima strojnog učenja omogućuje se dobivanje znanja bez eksplicitnog programiranja, odnosno glavna namjera tehnike strojnog učenja je omogućiti računalima učenje bez ljudske pomoći. Jedno je od danas najuzbudljivijih i najaktivnijih područja računarstva, ponajviše zbog brojnih mogućnosti primjene koje se protežu od raspoznavanja uzoraka i dubinske analize podataka pa sve do robotike, umjetne inteligencije, bioinformatike, računalnog vida i računalne lingvistike [1]. Strojno učenje jedno je od najbrže rastućih područja računarstva. Ne radi se samo o tome da se podaci neprestano gomilaju, već i o teoriji koja ih obrađuje i pretvara u znanje.

Strojno učenje koristi teoriju statistike u izgradnji matematičkih modela, jer je temeljni zadatak stvaranje zaključaka iz uzorka (eng. sample). Uloga računarstva je dvostruka: prvo, u obuci su nam potrebni učinkoviti algoritmi za rješavanje problema optimizacije, kao i za pohranu i obradu goleme količine podataka koje općenito imamo. Drugo, nakon što se model nauči, njegova reprezentacija i algoritamsko rješenje za zaključivanje također moraju biti učinkoviti. U određenim primjenama, učinkovitost algoritma učenja ili zaključivanja, naime njegova vremenska i prostorna složenost, može biti jednako važna kao i njegova točnost predviđanja [1].

Danas se strojno učenje koristi u širokom rasponu aplikacija, svi smo u doticaju s njim. Možda je jedan od najpoznatijih primjera strojnog učenja s kojim je većina u doticaju mehanizam za preporuke koji pokreće Facebookova početna stranica koja prikazuje novosti. Facebook koristi strojno učenje kako bi personalizirao početnu stranicu svakog člana. Ako se član često zaustavlja kako bi pročitao postove određene grupe, mehanizam za preporuke počet će prikazivati više aktivnosti te grupe ranije na početnoj stranici. Iza kulisa, mehanizam pokušava ojačati poznate obrasce ponašanja članova dok su na mreži (eng. online). Ako član promijeni obrasce i ne čita

postove iz te grupe u nadolazećim tjednima, vijesti na početnoj stranici će se prilagoditi u skladu s tim. Preostali primjeri popularnih aplikacija koje danas većina koristi su Googleovo korištenje strojnog učenja da poboljša preciznost rezultata traženja, Netflixovo prikazivanje postova ovisno o našim interesima i prošlom ponašanju na društvenoj mreži te digitalni pomoćnici (eng. digital assistants) koji koriste strojno učenje da bi unaprijedili tehnologiju prepoznavanja govora [2].

Područje strojnog učenja često se dijeli u potpodručja na temelju vrsta problema koji se nastoje riješiti dok su glavne vrste strojnog učenja; nadzirano učenje (eng. supervised learning), nenadzirano učenje (eng. unsupervised learning) i ojačano učenje (eng. reinforcement learning) [3]. Kada govorimo o prednostima, strojno učenje može pomoći poduzećima da razumiju svoje klijente na dubljoj razini. Prikupljanjem podataka o klijentima i njihovim povezivanjem s ponašanjem tijekom vremena, algoritmi strojnog učenja mogu naučiti povezivanja i pomoći timovima da prilagode razvoj proizvoda i marketinške inicijative zahtjevima kupaca. Ali strojno učenje dolazi s nedostacima. Prvo i najvažnije, može biti skupo. Projekte strojnog učenja obično pokreću podatkovni inženjeri (eng. data engineer) koji imaju visoke plaće. Ovi projekti također zahtijevaju softversku infrastrukturu koja može biti skupa. Tu je i problem pristranosti strojnog učenja. Algoritmi obučeni na skupovima podataka (eng. datasets) koji isključuju određene populacije ili sadrže pogreške mogu dovesti do netočnih modela koji, u najboljem slučaju, ne uspijevaju, a u najgorem su diskriminirajući [2].

Iako algoritmi strojnog učenja postoje već desetljećima, stekli su novu popularnost kako je umjetna inteligencija postala sve važnija. Modeli dubokog učenja (eng. deep learning) posebice pokreću današnje najnaprednije aplikacije iz područja umjetne inteligencije. Platforme strojnog učenja su među većim tehnološkim poduzećima najkonkurentnija područja, gdje se Amazon, Google, Microsoft, IBM i drugi, utrkuju da prijave korisnike za usluge platforme koje pokrivaju spektar aktivnosti strojnog učenja, uključujući prikupljanje podataka, pripremu podataka, klasifikaciju podataka, izgradnju modela, obuku i implementaciju aplikacija [2].

Ovaj diplomski rad, izrađen u okviru ERASMUS+ projekta *Promoting Sustainability as a Fundamental Driver in Software development Training and Education* s oznakom 2020-1-PT01-KA203-078646, za cilj ima proučiti područje nadziranog i polu-nadziranog (eng. semi-supervised) učenja. Upoznati se s metodama i modelima polu-nadziranog učenja temeljenima na predstavnicima različitih potkategorija strojnog učenja te opisati njih, njihove prednosti i mane. Vrednovati odabrane modele te izmjeriti utrošak energije, memorije i vremena za različite scenarije nadziranog i polu-nadziranog učenja.

Rad je razrađen u četiri poglavlja, od kojih početno poglavlje opisuje modele i metode nadziranog i polu-nadziranog učenja. Sljedeća dva poglavlja opisuju razne evaluacijske metrike te razvojno i testno okruženje gdje su spomenute sve programske knjižnice potrebne za implementaciju cjelokupnog rješenja. Na kraju slijedi poglavlje rezultata koje opisuje skupove podataka, proces razvoja modela predviđanja i rezultate istraživanja.

2. MODELI UČENJA

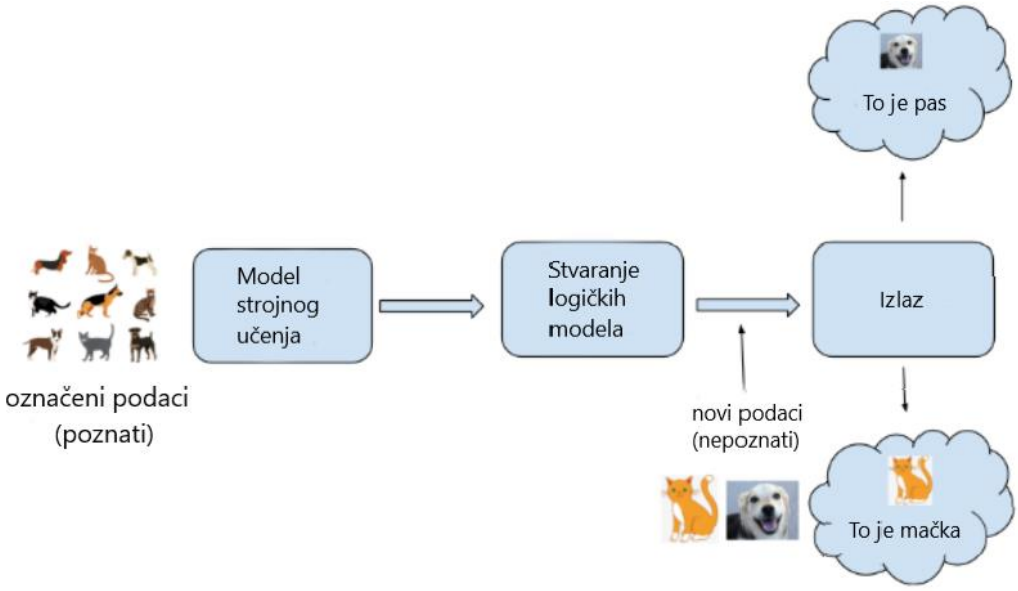
Izbor algoritma je velik, a danas postoji više desetaka algoritama. Ukratko, svaki od algoritama ima drugačiji pristup učenju. Mogli bismo reći da ne postoji najbolja niti univerzalna metoda. Čak i iskusni eksperti ne mogu predvidjeti hoće li algoritam biti djelotvoran pa preostaje metoda pokušaja i pogreške. Odabir algoritma ovisi o broju i vrsti podataka, što se želi postići te gdje i kako će se primijeniti rezultati [1].

2.1 Nadzirano učenje

Značajan broj tehnika nadziranog učenja razvijen je u industriji strojnog učenja u posljednjih deset godina. Velik dio istraživanja strojnog učenja usredotočen je na nadzirano učenje. Brojni pristupi nadziranog učenja našli su primjenu u obradi i analizi različitih vrsta podataka [5]. Definirano je korištenjem označenih skupova podataka za treniranje algoritama koji služe za klasifikaciju podataka. Kako se ulazni podaci unose u model, on prilagođava svoje težine sve dok se model ne prilagodi na odgovarajući način, što se događa kao dio procesa unakrsne provjere (eng. cross-validation). Nadzirano učenje pomaže organizacijama u rješavanju niza problema iz stvarnog svijeta velikih razmjera, kao što je primjerice klasificiranje pristigle korisnikove neželjene pošte (eng. spam) te njeno svrstavanje u zasebnu mapu [5].

2.1.1 Kako funkcionira nadzirano učenje?

Modeli nadziranog učenja treniraju se korištenjem označenih podataka, koji se nazivaju skup podataka za trening, služe za predviđanje rezultata. Ovaj skup podataka za trening uključuje ulazne (eng. input) i ispravne izlazne (eng. output) podatke, koji omogućuju modelu da s vremenom uči. Algoritam mjeri svoju točnost kroz funkciju gubitka, prilagođavajući se dok se pogreška dovoljno ne smanji. Ako uzmemo u obzir da imamo skup podataka koji sadrže podatke o mačkama i psima. Svaki model psa i mačke prvo se mora istrenirati prema kriterijima sličnosti, uzorka, oblika i kontrasta. Glavna funkcija modela je prepoznavanje novih ulaznih podataka kada se procjenjuju korištenjem novog ulaznog skupa podataka koji nije korišten za treniranje modela. Računalo je osposobljeno za prepoznavanje svih vrsta uzoraka, oblika i kontrasta. Dodatno, kategorizira novootkrivene podatke na temelju sličnosti, uzoraka, oblika i kontrasta te predviđa točan rezultat [5]. Prethodno opisan proces nadziranog učenja prikazan je na slici 2.1.



Slika 2.1. *Proces nadziranog učenja* [5]

2.1.2 Prednosti nadziranog učenja

Nadzirano učenje rješava različite računalne probleme koji se javljaju u stvarnom svijetu, uključujući otkrivanje neželjene pošte, identifikaciju objekata, slika i mnoge druge. Koristi prošlo iskustvo za optimizaciju performansa i predviđanje rezultata na temelju prošlih iskustava [5]. Rezultati dobiveni metodom nadziranog učenja točniji su i pouzdaniji u usporedbi s rezultatima dobivenim tehnikama nenadziranog strojnog učenja. To je uglavnom zato što su ulazni podaci u nadziranom algoritmu dobro poznati i označeni što je ujedno i ključna razlika između nadziranog i nenadziranog učenja [6]. Kod nadziranog učenja podaci za treniranje mogu se ponovno upotrijebiti ako nema promjena značajki (eng. feature) [5].

2.1.3 Mane nadziranog učenja

Iako nadzirano učenje može ponuditi dubinski uvid u podatke i poboljšane automatizacije, postoje neka ograničenja i izazovi pri izgradnji održivih modela nadziranog učenja. Nadzirano učenje ograničeno je u raznim slučajevima tako da ne može riješiti neke od složenijih zadataka u strojnom učenju. Ne može klasificirati ili grupirati podatke otkrivajući njihove značajke samostalno, za razliku od nenadziranog učenja. U slučaju klasifikacije, ako se na ulazu nalaze podaci za trening modela koji nisu ni iz jedne klase, tada izlaz može biti pogrešna oznaka klase. Primjerice, recimo da smo istrenirali klasifikator slika s podacima o mačkama i psima. Zatim, ako damo sliku žirafe, izlaz može biti mačka ili pas, što nije točno. Obično je za trening potrebno puno vremena za

računanje kao i za klasifikaciju, osobito ako je skup podataka vrlo velik, što će ujedno i testirati učinkovitost stroja na kojem se vrši nadzirano učenje [7].

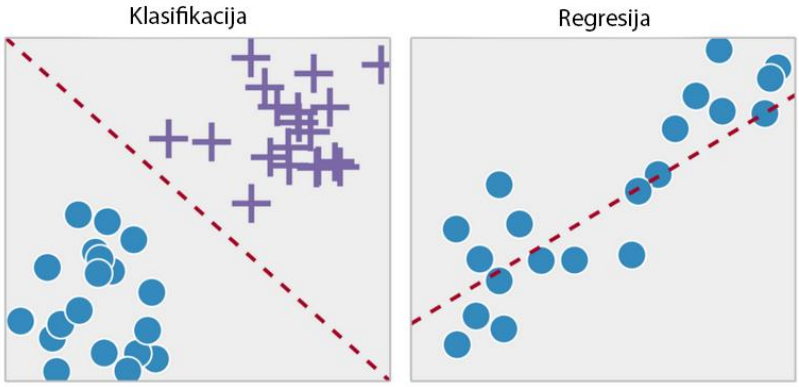
Kao što možemo vidjeti, postoje mnoge prednosti, ali i nedostatci nadziranog učenja općenito, ali većinu vremena prednosti i mane nadziranog učenja ovise o tome koji se algoritam nadziranog učenja koristi.

2.1.4 Tehnike nadziranog učenja

Algoritmi nadziranog učenja mogu se grupirati u dvije tehnike koje se nazivaju klasifikacija (eng. classification) i regresija (eng. regression) čiji se opis nalazi u nastavku.

Klasifikacija je proces kategorizacije zadanog skupa podataka u diskretne klase. Može se provoditi na strukturiranim i nestrukturiranim podacima. Klasifikacija može biti binarna ili problem više klasa. Proces počinje predviđanjem klase danih podatkovnih točaka. Klase se često nazivaju oznaka, cilj ili kategorije. Klasifikacijsko prediktivno modeliranje je zadatak aproksimacije funkcije preslikavanja od ulaznih varijabli do diskretnih izlaznih varijabli. Glavni cilj je identificirati u koju će kategoriju/klasu novi podaci spadati [8].

Regresija je metoda za razumijevanje odnosa između nezavisnih varijabli ili značajki i zavisnih varijabli ili ishoda, gdje je zavisna varijabla kontinuirana vrijednost, odnosno realni broj. Nakon što se procijeni odnos između nezavisnih i zavisnih varijabli tada se mogu predvidjeti ishodi. Regresija strojnog učenja općenito uključuje iscrtavanje linije najboljeg uklapanja kroz podatkovne točke. Udaljenost između svake točke i linije minimalizirana je kako bi se postigla najbolja linija, u kontekstu linearne regresije. Klasifikacija je kategorizacija objekata na temelju naučenih značajki, dok je regresija predviđanje kontinuiranih ishoda [9]. Na slici 2.2 prikazana je razlika klasifikacije i regresije.



Slika 2.2. *Razlika između klasifikacije i regresije* [10]

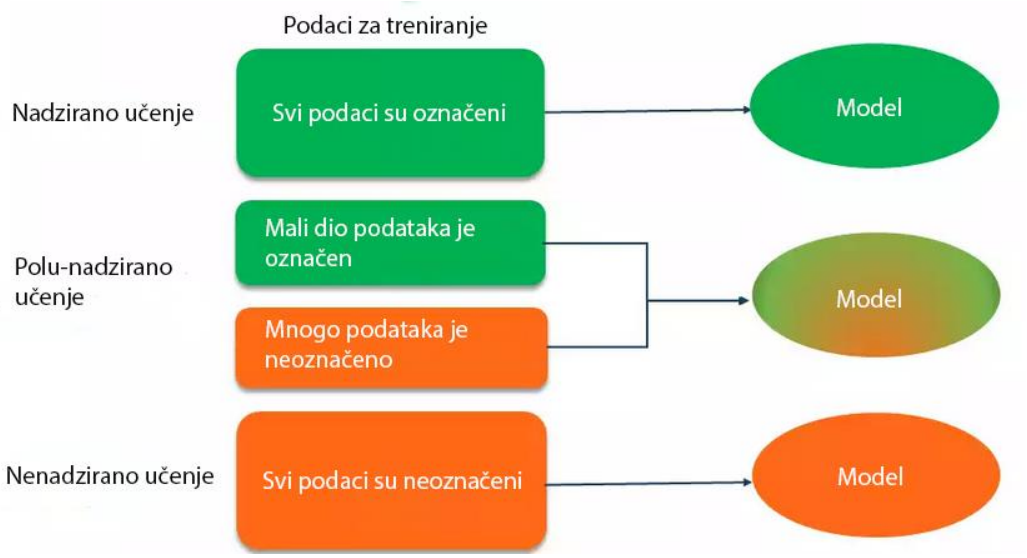
2.1.5 Primjeri korištenja nadziranog učenja

Algoritmi nadziranog učenja imaju široku upotrebu. Neke od mogućih upotreba su:

- Prepoznavanje slika i objekata: Algoritmi nadziranog učenja mogu se koristiti za lociranje, izolaciju i kategorizaciju objekata iz slika ili videozapisa, što ih čini korisnima prilikom primjene na različitim tehnikama računalnog vida i analize slika [4].
- Analitika predviđanja: Široko rasprostranjena upotreba modela nadziranog učenja je u stvaranju analitičkih sustava predviđanja za pružanje dubokog uvida u razne poslovne podatke. To omogućuje poduzećima predviđanje određenih rezultata na temelju dane izlazne varijable, pomažući opravdavanje donesenih odluka za dobrobit organizacije [4].
- Analiza raspoloženja korisnika: korištenjem algoritama nadziranog strojnog učenja, organizacije mogu izvući i klasificirati važne dijelove informacija iz velikih količina podataka (uključujući kontekst, emocije i namjere) uz vrlo male potrebe ljudske intervencije. To može biti veoma korisno pri stjecanju boljeg razumijevanja interakcija s klijentima i može se koristiti za poboljšanje angažmana robne marke [4].
- Otkrivanje neželjene pošte: Otkrivanje neželjene pošte još je jedan primjer modela nadziranog učenja. Korištenjem nadziranih algoritama klasifikacije, organizacije mogu uvježbati baze podataka da prepoznaju uzorke ili anomalije u novim podacima za učinkovito organiziranje neželjene pošte i korespondencije koja nije povezana s neželjenom poštom [4].

2.2 Polu-nadzirano učenje

Polu-nadzirano učenje je paradigma učenja koja se bavi proučavanjem načina na koji računala i prirodni sustavi kao što su ljudi, uče u prisutnosti označenih i neoznačenih podataka. Učenje se tradicionalno proučava u nenadziranoj paradigmi gdje su svi podaci neoznačeni, ili u nadziranoj paradigmi gdje su svi podaci označeni [11]. Kako to ponekad biva, kada jedan pristup ne uspije riješiti problem, pokušamo s drugim. Kada ni taj pristup ne funkcionira, možda bi bilo dobro pokušati kombinirati najbolje dijelove oba pristupa. Barem je to često slučaj s tehnološkim zadacima, a strojno učenje nije iznimka. Stoga, kombiniranjem nenadziranog i nadziranog učenja dobivamo polu-nadzirano učenje. Kako bi lakše razumjeli koncept polu-nadziranog učenja prikazan je na slici 2.3 [12]. Cilj polu-nadziranog učenja je razumjeti kako se kombiniranje označenih i neoznačenih podataka može promijeniti ponašanje učenja i dizajn algoritama koji iskorištavaju prednosti takve kombinacije. Polu-nadzirano učenje je područje od velikog interesa za rudarenje podataka jer može koristiti lako dostupne neoznačene podatke za poboljšanje zadataka nadziranog učenja kada su označeni podaci rijetki ili skupi [11].



Slika 2.3. *Nadzirano naspram nenadziranog i polunadziranog strojnog učenja* [12]

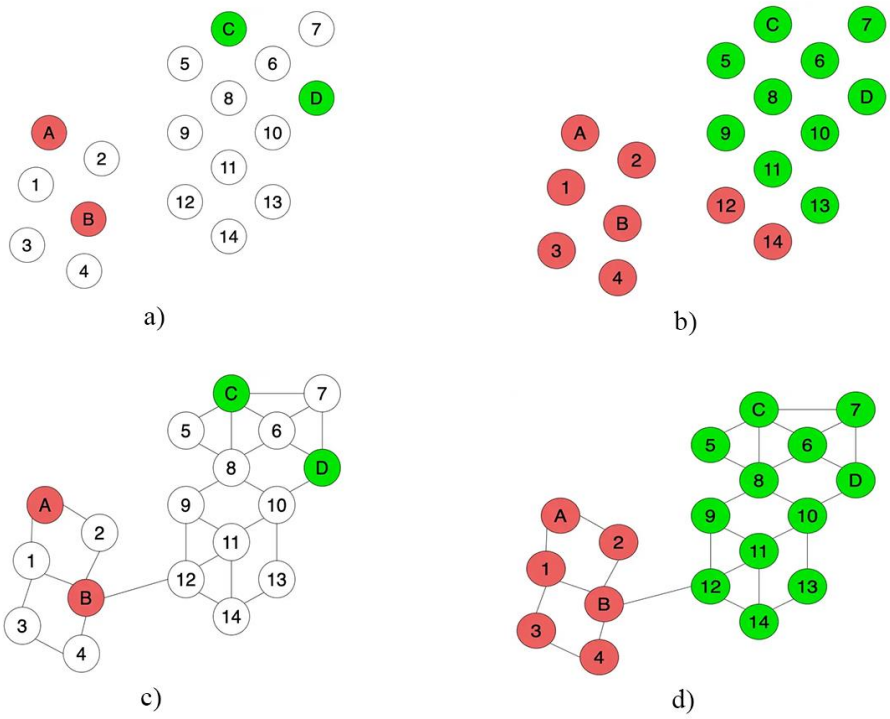
Postoje dvije različite postavke polu-nadziranog učenja, a to su induktivno i transduktivno polu-nadzirano učenje. Podsjetimo se da je u nadziranoj klasifikaciji skup podataka za trening potpuno označen, stoga nas uvijek zanima izvedba na testnim podacima. U polu-nadziranoj klasifikaciji skup podataka za trening sadrži neke neoznačene podatke pa postoje dva različita cilja. Prvi cilj je predvidjeti oznake na budućim testnim podacima i njega nazivamo induktivnim polu-nadziranim učenjem. Drugi cilj je predvidjeti oznake na neoznačenim podacima u skupu podataka namijenjenom za trening i njega nazivamo transduktivnim polu-nadziranim učenjem [11].

Glavna razlika između transduktivnog i induktivnog učenja je u tome što se tijekom transduktivnog učenja već naišlo na skupove podataka za treniranje i testiranje prilikom treniranja modela. Međutim, induktivno učenje susreće se samo s podacima za trening kada se model trenira i primjenjuje istrenirani model na skup podataka koji nikada prije nije vidio. Transdukcija ne gradi model predviđanja, stoga ako se skupu podataka za testiranje doda nova podatkovna točka, morat će se ponovno pokrenuti algoritam od početka, uvježbati model i zatim ga upotrijebiti za predviđanje oznaka. S druge strane, induktivno učenje gradi model predviđanja što znači da kada naiđe na nove podatkovne točke, nema potrebe za ponovnim pokretanjem algoritma od početka. Jednostavnije rečeno, induktivno učenje pokušava izgraditi generički model u kojem bi bilo koja nova podatkovna točka bila predviđena, na temelju promatranog skupa podataka za trening. Naprotiv, transduktivno učenje gradi model koji odgovara podacima za treniranje i testiranje koje je već promatralo [13] [14] [15].

Transduktivno učenje može postati skupo u slučaju kada se nove podatkovne točke uvode putem ulaznog toka. Svaki put kada stigne nova podatkovna točka, morat će se sve ponovno pokrenuti što u konačnici znači i veću potrošnju resursa na računalu. S druge strane, induktivno učenje u početku gradi model predviđanja i nove podatkovne točke mogu se označiti u vrlo kratkom vremenu s manjim proračunima. Primjeri transduktivnih pristupa učenju uključuju transduktivnu metodu potpornih vektora (eng. transductive support vector machines, TSVM) i algoritme propagiranja oznaka temeljene na grafovima (eng. graph-based label propagation algorithms, LPA) [13]. U nastavku slijedi primjer korištenja transduktivnog učenja.

Uzmimo u obzir da imamo skup točaka kao što je prikazano na slici 2.4 pod slovom a). Postoje četiri označene točke A, B, C i D. Naš cilj je označiti/obojiti preostale neoznačene/nebojane točke označene brojevima od 1 do 14. Ako koristimo induktivno učenje za ovaj zadatak, morat ćemo koristiti ove 4 označene točke i izgraditi model nadziranog učenja [13].

Ako pogledamo sliku 2.4 pod slovom b), na prvi pogled možemo vidjeti da postoje dvije odvojene grupe (eng. cluster). Međutim, u induktivnom učenju, budući da imamo vrlo mali broj uzoraka za treniranje, bit će prilično teško izgraditi model predviđanja koji obuhvaća potpunu strukturu podataka. Na primjer, ako se koristi metoda najbližeg susjeda (eng. nearest neighbor method), točke bliže granici kao što su 12 i 14 mogu biti obojene crvenom umjesto zelenom jer su bliže crvenim točkama A i B, a ne zelenim točkama C i D [13].



Slika 2.4. *Primjer transduktivnog učenja* [13]

Ako imamo neke dodatne informacije o podatkovnim točkama kao što su informacije o povezanosti između točaka na temelju značajki poput sličnosti kao što je prikazano na slici 2.4 pod slovom c), možemo koristiti te dodatne informacije dok treniramo model i označavamo neoznačene točke.

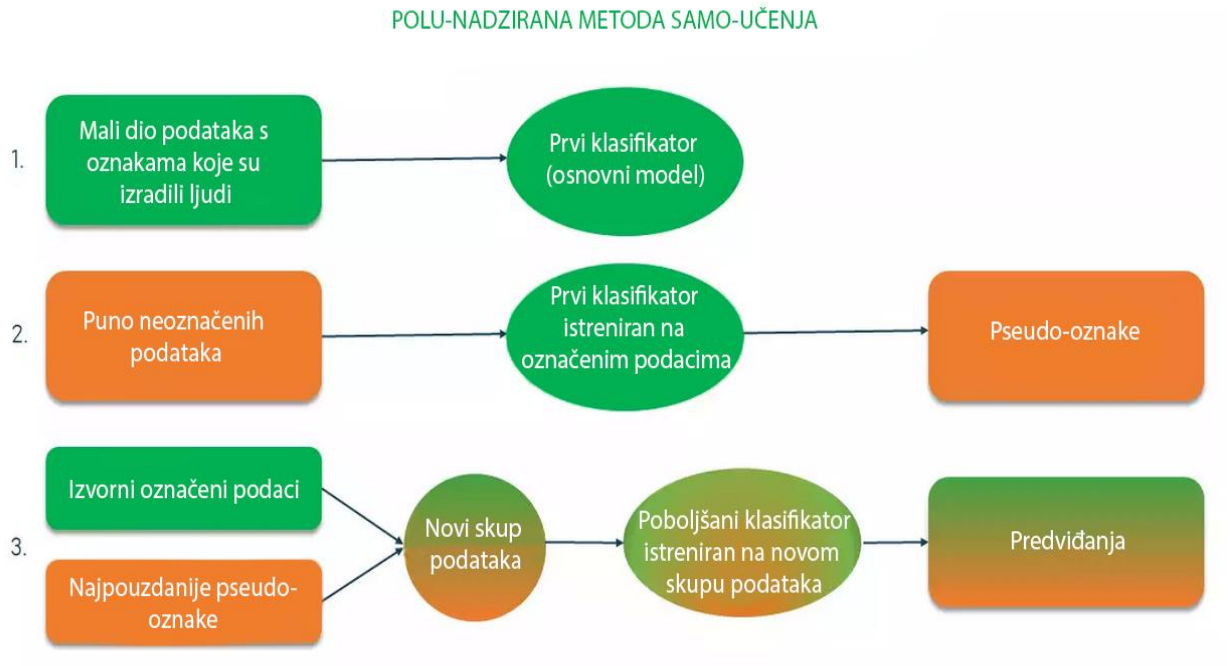
Na primjer, možemo koristiti transduktivni pristup učenju kao što je polu-nadzirani algoritam propagiranja oznaka (eng. lable propagation algorithm, LPA) temeljen na grafu za označavanje neoznačenih točaka kao što je prikazano na slici 2.4 pod slovom d), koristeći strukturne informacije svih označenih i neoznačenih točaka. Točke duž granice kao što su 12 i 14 povezane su s više zelenih točaka, nego crvenih točaka i stoga se označavaju kao zelene, a ne kao crvene [13].

Uzme li se u obzir da smo bili u mogućnosti primijeniti transduktivni pristup učenju kao što je algoritam propagiranja oznaka jer smo na početku naišli na sve podatkovne točke treniranja i testiranja, a podaci testiranja sadrže neke dodatne informacije koje bi mogle biti korisne. Ako u početku nema podatkovnih točaka za testiranje, morat ćemo slijediti induktivni pristup učenju [13].

Cilj polu-nadziranog učenja je korištenje neoznačenih instanci i kombiniranje informacija u neoznačenim podacima s eksplicitnim klasifikacijskim informacijama označenih podataka radi poboljšanja učinkovitosti klasifikacije. Glavno pitanje polu-nadziranog učenja je kako iskoristiti informacije iz neoznačenih podataka. Predstavljen je niz različitih metoda za polu-nadzirano učenje od kojih će neke biti opisane u nastavku. Neke poznate metode su; probabilistički generativni modeli, samo-učenje (eng. self-training), uzajamno-učenje (eng. co-training), grafičko utemeljeni modeli, metoda polu-nadziranih potpornih vektora (eng. semi-supervised suport vector machine, S3VM) i druge [11].

2.2.1 Metoda samo-učenja

Jedna od vjerojatno najranijih ideja i najjednostavnijih primjera polu-nadziranog učenja je metoda samo-treninga koja se još naziva metodom samo-učenja. Samo-učenje je postupak u kojem možemo uzeti bilo koju nadziranu metodu za regresiju ili klasifikaciju i modificirati je da radi na polu-nadzirani način, koristeći prednosti označenih i neoznačenih podataka [12]. Algoritam samo-učenja iterativna je metoda za polu-nadzirano učenje, a njegove prednosti su jednostavnost i činjenica da funkcionira kao metoda omotača (eng. wrapper method). Zbog toga je izbor osnovnog klasifikatora (eng. base learner) potpuno otvoren [16]. Standardni tijek rada je prikazan je na slici 2.5.



Slika 2.5. *Princip rada polu-nadzirane metode samo-učenja* [12]

Princip rada metode samo-učenja opisan je u nastavku:

- Koristit ćemo se istim primjerom kao prilikom objašnjavanja nadziranog učenja. Odaberemo malu količinu označenih podataka, npr. slike koje prikazuju pse i mačke s pripadajućim oznakama, i koristimo taj skup podataka za treniranje osnovnog modela uz pomoć uobičajenih nadziranih metoda [12].
- Zatim primijenimo proces poznat kao pseudo-označavanje (eng. pseudo-labeling), kada uzmemo djelomično uvježban model i koristimo ga za izradu predviđanja za ostatak baze podataka koja još nije označena. Oznake koje se nakon toga generiraju nazivaju se pseudo-oznake jer se proizvode na temelju izvorno označenih podataka koji imaju ograničenja. Recimo, može postojati nejednaka zastupljenost klasa u skupu što rezultira pristranošću (više pasa nego mačaka) [12].
- Nakon pseudo-označavanja uzimamo najpouzdanija predviđanja napravljena pomoću našeg modela (na primjer, želimo pouzdanost od preko 80 posto da određena slika prikazuje mačku, a ne psa). Ako bilo koja od pseudo-oznaka premaši ovu razinu pouzdanosti, dodajemo ih u označeni skup podataka i stvaramo novi, kombinirani unos za treniranje poboljšanog modela [12].
- Proces može proći kroz nekoliko iteracija (10 je često standard) sa sve više i više pseudo-oznaka koje se dodaju svaki put. Pod uvjetom da su podaci prikladni za proces, izvedba modela će se povećavati u svakoj iteraciji [12].

Iako postoje mnogi uspješni primjeri korištenja metode samo-učenja, treba naglasiti da izvedba može dosta varirati od jednog skupa podataka do drugog. Postoji mnogo slučajeva kada korištenje metode samo-učenja može davati lošije rezultate tj. smanjiti izvedbu u usporedbi s korištenjem nadziranog učenja s čime smo se susreli prilikom rada na projektu.

Izvedba algoritma za samo-učenje snažno ovisi o odabranim novooznačenim podacima pri svakoj iteraciji prilikom postupka treniranja. Ova strategija odabira podataka temelji se na povjerenju u predviđanja i stoga je za samo-učenje od velikog značaja da se pouzdanost predviđanja ispravno mjeri. Postoji razlika između algoritama učenja koji daju distribuciju vjerojatnosti, npr. Bayesove metode (eng. Bayesian methods), neuronske mreže (eng. neural networks), logistička regresija, klasifikatori temeljeni na marginama i algoritama koji se obično smatraju samo izlaznim modelom klasifikacije, poput stabala odlučivanja (eng. decision trees). Većina trenutnih pristupa samo-učenja koristi prvu vrstu algoritama učenja kao osnovni klasifikator [16].

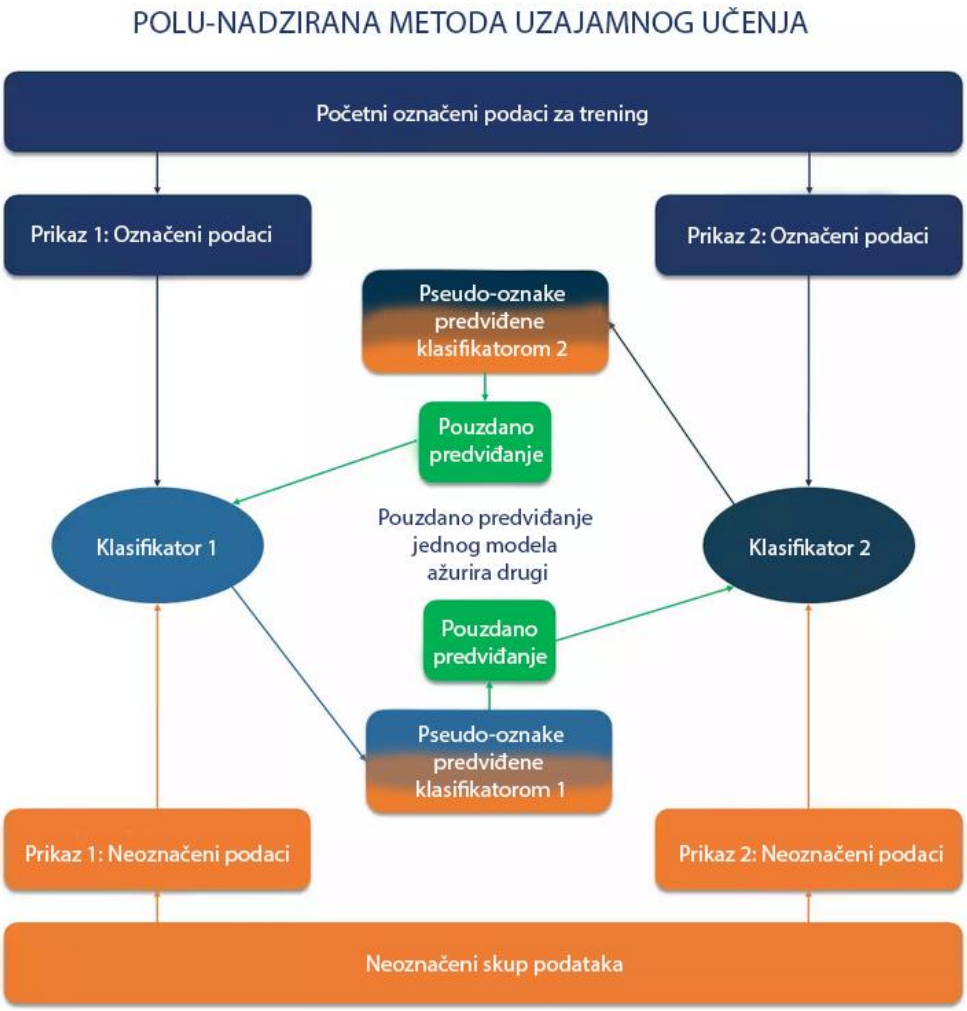
Jedan od mogućih nedostataka je da ako dođe do greške može se desiti da greška u klasifikaciji može ojačati samu sebe. Neki algoritmi pokušavaju to izbjeći tako što "odučavaju" neoznačene točke ako pouzdanost predviđanja padne ispod praga. Samo-učenje je primijenjeno na nekoliko zadataka obrade prirodnog jezika. Yarowsky (1995) koristi metodu samo-učenja za razjašnjavanje smisla riječi, npr. odlučivanje znači li riječ "jezik" sustav znakova za sporazumijevanje ili pokretljiv mišić u usnoj šupljini. Riloff (2003) koristi ju za identifikaciju subjektivnih imenica. Maeireizo (2004) koristi samo-učenje za klasificiranje dijaloga kao "emocionalne" ili "neemocionalne" postupkom koji uključuje dva klasifikatora. Samo-učenje je također primijenjeno za raščlanjivanje i strojno prevođenje. Rosenberg (2005) primjenjuje samo-učenje na sustave za otkrivanje objekata iz slika i pokazuje da se polu-nadzirana tehnika povoljno uspoređuje s najsuvremenijim detektorom [17].

2.2.2 Metoda uzajamnog učenja

Metoda uzajamnog učenja (eng. co-training method) je metoda izvedena iz pristupa metode samo-učenja. Kao njezina poboljšana verzija metoda uzajamnog učenja je još jedna polu-nadzirana tehnika koja se koristi kada je dostupan samo mali dio označenih podataka. Za razliku od tipičnog procesa, uzajamno učenje obučava dva pojedinačna klasifikatora na temelju dva prikaza podataka.

Prikazi (eng. views) su u osnovi različiti skupovi značajki koji pružaju dodatne informacije o svakoj instanci, što znači da su neovisni s obzirom na klasu. Također, svaki prikaz je dovoljan, što znači da se klasa uzorka podataka može točno predvidjeti samo iz svakog skupa značajki.

Metoda uzajamnog učenja je pristup koji se može uspješno koristiti, na primjer za zadatke klasifikacije web sadržaja. Opis svake web stranice može se podijeliti u dva prikaza: jedan s riječima koje se pojavljuju na toj stranici, a drugi sa sidrenim (eng. anchor) riječima u poveznici koja vodi do nje [12]. Na slici 2.6. prikazana je metoda uzajamnog učenja.



Slika 2.6. *Princip rada polu-nadzirane metode uzajamnog učenja* [12]

Princip rada metode uzajamnog učenja opisan je u nastavku:

- Prvo treniramo zaseban klasifikator (model) za svaki prikaz uz pomoć male količine označenih podataka.
- Zatim se dodaje veći skup neoznačenih podataka za primanje pseudo-oznaka.
- Klasifikatori se međusobno obučavaju koristeći pseudo-oznake s najvišom razinom pouzdanosti. Ako prvi klasifikator pouzdano predviđa originalnu oznaku za uzorak podataka, dok drugi čini pogrešku u predviđanju, tada podaci s pouzdanim pseudo-oznakama koje je dodijelio prvi klasifikator ažuriraju drugi klasifikator i obrnuto.
- Posljednji korak uključuje kombiniranje predviđanja iz dvaju ažuriranih klasifikatora kako bi se dobio jedan rezultat klasifikacije.

Kao i kod samo-učenja, uzajamno učenje prolazi kroz mnoge iteracije kako bi se izgradio dodatni skup podataka s oznakom za trening iz ogromne količine neoznačenih podataka [12].

Uzajamno učenje je korišteno za klasificiranje web stranica koristeći tekst na stranici kao jedan prikaz i sidreni tekst hiperveza na drugim stranicama koje upućuju na stranicu kao drugi prikaz. Jednostavno rečeno, tekst u hipervezi na jednoj stranici može dati informacije o stranici na koju vodi. Jedna od prednosti uzajamnog učenja je ta da može raditi na neoznačenom tekstu koji još nije klasificiran ili označen, što je tipično za tekst koji se pojavljuje u e-pošti i na web stranicama. Američki informatičar Tom Mitchell tvrdi da obilježja koja opisuju stranicu su riječi na stranici i poveznice koje upućuju na tu stranicu. Modeli uzajamnog učenja koriste oba klasifikatora za određivanje vjerojatnosti da će stranica sadržavati podatke relevantne za kriterije pretraživanja. Tekst na web-mjestima može procijeniti relevantnost klasifikatora veza, otuda i izraz "uzajamno-učenje". Testiranje temeljeno na novinskim grupama UseNet otkriveno je da je Mitchellov algoritam za uzajamno učenje postigao samo 3,7% pogreške koristeći 6 označenih dokumenata i 1000 neoznačenih – puno bolje od 8,9% pogreške dobivene korištenjem EM-baziranog pristupa [18] [19].

Postoje mnoge varijacije i upotrebe uzajamnog učenja, a neke od njih navedene su u nastavku. Nigam i Ghani (2000.) provode opsežne empirijske eksperimente kako bi usporedili metodu uzajamnog učenje s modelima generativne mješavine i algoritma maksimizacija očekivanja (eng. Expectation-Maximization, EM). Njihov rezultat pokazuje da uzajamno učenje ima dobre rezultate ako pretpostavka o uvjetnoj neovisnosti doista vrijedi. Osim toga, bolje je probabilistički označiti cijeli skup neoznačenih podataka umjesto nekoliko najpouzdanijih podatkovnih točaka. Oni ovu paradigmu nazivaju uzajamni-EM (eng. co-EM). Konačno, ako ne postoji prirodna podjela značajki, autori stvaraju umjetnu podjelu tako što nasumično dijele skup značajki na dva podskupa. Oni pokazuju da uzajamno učenje s umjetnom podjelom značajki i dalje pomaže, iako ne toliko kao prije. Collins i Singer (1999.), te Jones (2005.) koristili su uzajamno učenje, uzajamni-EM i druge srodne metode za izvlačenje informacija iz teksta. Balcan i Blum (2006) pokazuju da uzajamno učenje može biti vrlo učinkovito, da je u ekstremnom slučaju za učenje klasifikatoru potrebna samo jedna označena točka. Zhou (2007) predstavlja algoritam uzajamnog učenja koristeći kanoničku korelacijsku analizu (eng. Canonical Correlation Analysis) koja također treba samo jednu označenu točku.

U uzajamnom učenju neoznačeni podaci pomažu smanjenjem veličine prostora verzije (eng. version space). Drugim riječima, dva klasifikatora (ili hipoteze) moraju se složiti oko mnogo većoj količini neoznačenih podataka, kao i oko označenih podataka. Uzajamno učenje daje snažne

pretpostavke o razdvajanju značajki. Netko bi se mogao zapitati mogu li se ti uvjeti ublažiti. Goldman i Zhou (2000.) koriste dva klasifikatora različite vrste, ali oba preuzimaju cijeli skup značajki i u biti koriste podatkovne točke visoke pouzdanosti jednog klasifikatora, identificirane skupom statističkih testova, u neoznačenom skupu podataka kako bi podučili drugog klasifikatora i obrnuto. Chawla i Karakoulas (2005) provode empirijska istraživanja ove verzije uzajamnog učenja i uspoređuju je s nekoliko drugih metoda, posebno za slučaj kada označeni i neoznačeni podaci ne slijede istu distribuciju [17].

2.2.3 Metoda propagiranja oznaka

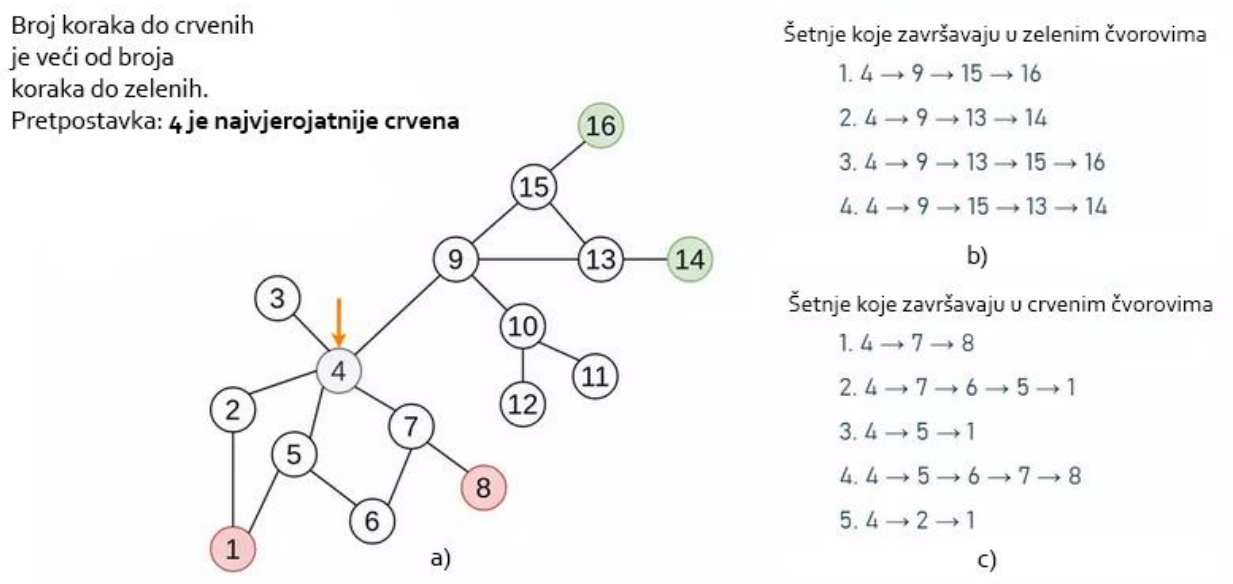
Popularan način korištenja polu-nadziranog učenja je predstavljanje označenih i neoznačenih podataka u obliku grafa nakon čega slijedi primjena algoritma za propagiranje oznaka (eng. label propagation algorithm, LPA) [12]. Algoritam propagiranja oznaka iterativni je algoritam u kojemu neoznačenim točkama dodjeljujemo oznake propagiranjem oznaka kroz skup podataka. Ovaj algoritam prvi su predložili Xiaojin Zhu i Zoubin Ghahramani 2002. godine. LPA spada u transduktivno učenje, budući da želimo predvidjeti oznake neoznačenih podatkovnih točaka koje su nam već dane [20].

Pretpostavimo da imamo mrežu ljudi kao što je prikazano na slici 2.7 u nastavku s dvije klase s oznakama "zainteresirani za nogomet" i "nezainteresirani za nogomet". Dakle, pitanje je možemo li predvidjeti hoće li preostali ljudi biti zainteresirani za nogomet ili ne?

Slika 2.7. *Mreža ljudi zaintresiranih i nezaintresiranih za nogomet*

Da bi metoda propagiranja oznaka funkcionirala u ovom slučaju, moramo napraviti pretpostavku; rub koji spaja dva čvora nosi pojam sličnosti tj. ako su dvije osobe povezane, to znači da je vrlo vjerojatno da te dvije osobe dijele iste interese. Ovu pretpostavku možemo napraviti jer su ljudi skloni povezivanju s drugim ljudima koji imaju slične interese [20].

Na slici 2.8. prikazan je primjer grafa s podatkovnim točkama, od kojih je većina neoznačena s četiri noseće oznake (dvije crvene točke i dvije zelene točke koje predstavljaju različite klase) [12]. Želimo predvidjeti oznaku čvora 4. Možemo nasumično hodati po grafu, počevši od čvora 4 dok ne sretnemo bilo koji označeni čvor. Kada dođemo do označenog čvora, zaustavljamo se. Stoga su ti označeni čvorovi poznati kao apsorbirajuća stanja. Razmotrimo sve moguće šetnje od čvora 4. Od svih mogućih šetnji, šetnje koje će završiti u zelenom čvoru prikazane su na slici 2.8. pod slovom b) dok su šetnje koje će završiti u crvenom čvoru prikazane na slici 2.8. pod slovom c). Na temelju svih mogućih nasumičnih šetnji po grafu počevši od čvora 4, možemo vidjeti da većina hodanja završava u crvenom čvoru. Dakle, možemo obojiti čvor 4 u crveno. Ovo je osnovna intuicija iza LPA [20].



Slika 2.8. *Primjer grafa za algoritam propagiranja oznaka* [20]

LPA koristi oznake već označenih čvorova kao bazu i pokušava predvidjeti oznake neoznačenih čvorova. Međutim, ako je početno označavanje pogrešno, to može utjecati na proces propagiranja oznaka odnosno može doći do propagiranja pogrešnih oznaka. Kako bi se prevladao ovaj problem, uvedeno je širenje oznaka (eng. label spreading), gdje također učimo oznake označenih čvorova

dok učimo oznake neoznačenih čvorova. Ova vrsta također primjenjuje neke ispravke oznaka [20]. LPA pruža jednostavan i distribuiran način za otkrivanje zajednica u velikim mrežama u linearnom vremenu, što ga čini vrlo učinkovitim. Međutim, na njegovu izvedbu utječe slučajnost u algoritmu, što dovodi do nestabilnosti i pojave zajednica čudovišta. To je zbog redoslijeda kojim se čvorovi ažuriraju, budući da probabilistički sustav ažuriranja oznaka s najčešćom oznakom među susjedima rezultira nesigurnim označavanjem [21].

Praktična upotreba metode propagiranja oznaka može se vidjeti u sustavima personalizacije i preporuka. Propagiranjem oznaka možemo predvidjeti interese kupaca na temelju informacija o drugim kupcima. Ovdje možemo primijeniti varijaciju pretpostavke o kontinuitetu – na primjer, ako su dvije osobe povezane na društvenim medijima vrlo je vjerojatno da će dijeliti slične interese [12].

2.2.4 Algoritam maksimizacije očekivanja

Algoritam maksimizacije očekivanja (EM) je klasični statistički algoritam za procjenu modela s obzirom na neke varijable koje postoje u latentnom prostoru varijabli [22]. Osnovna ideja EM algoritma jest da se generativni model proširi skrivenim varijablama. Skrivene varijable su one varijable čije vrijednosti ne opažamo u podatcima, ali ih uvodimo u model jer to može značajno pojednostaviti modeliranje, npr. kada skrivene varijable modeliraju neke zajedničke uzroke više opaženih varijabli. Ako skrivene varijable modeliraju neki apstraktni koncept, tada umjesto naziva skrivene varijable koristimo naziv latentne varijable [23]. Iako se u strojnom učenju EM često povezuje s relativno jednostavnim Gaussovim modelima mješavine (eng. Gaussian mixture models, GMM), može se primijeniti i na kompliciranije modele dubokog učenja kao što su konvolucijske neuronske mreže (eng. Convolutional neural network, CNN), čime se omogućuje obučavanje CNN-ova u prisutnosti latentnih varijabli [22].

U nastavku slijedi detaljnije objašnjenje polu-nadziranog EM algoritma:

1. Inicijalizacija - Slično tradicionalnom EM algoritmu, polu-nadzirani EM počinje početnom procjenom parametara modela.
2. Korak očekivanja (E-korak) - U ovom koraku algoritam izračunava posteriorne vjerojatnosti (koje se nazivaju i odgovornosti) za svaku podatkovnu točku, pokazujući vjerojatnost da svaka točka pripada svakoj klasi ili grupi. Za označene podatkovne točke, algoritam koristi njihove prave oznake za izravno izračunavanje posteriornih vjerojatnosti. Za neoznačene podatkovne točke, algoritam procjenjuje vjerojatnosti njihovih oznaka na temelju trenutnih parametara modela [24].

3. Korak maksimiziranja (M-korak) - U ovom koraku algoritam ažurira parametre modela kako bi maksimizirao očekivanu log-vjerojatnost cijelog skupa podataka. Uzima u obzir i označene i neoznačene podatkovne točke, kao i njihove procijenjene posteriorne vjerojatnosti [24].
4. Ponavljanje - Koraci 2 i 3 ponavljaju se iterativno sve dok parametri modela ne konvergiraju u stabilno rješenje [24].

Ključna prednost polu-nadziranog EM algoritma je da koristi informacije iz neoznačenih podataka kako bi mogao doraditi model, čak i kada imamo ograničene označene podatke. To može dovesti do preciznijih modela, posebno u slučajevima kada je prikupljanje označenih podataka dugotrajno ili skupo [24].

Polu-nadzirani EM naširoko se koristi u raznim aplikacijama strojnog učenja, uključujući klasifikaciju teksta, segmentaciju slika i otkrivanje zajednice u grafikonima, gdje je dobivanje označenih podataka za svaku podatkovnu točku često nepraktično ili skupo [24].

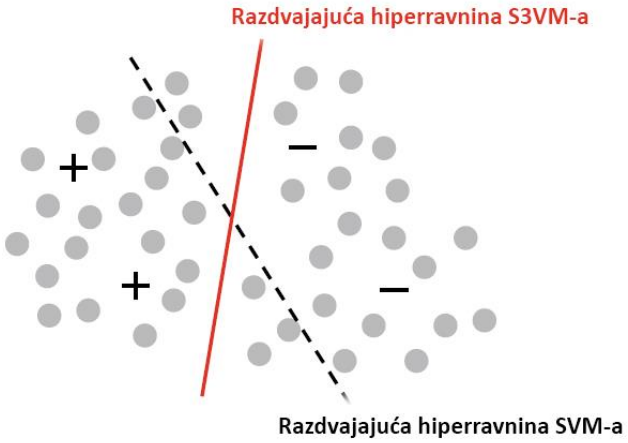
Ukratko, polunadzirani EM algoritam vrijedan je alat za poboljšanje izvedbe modela strojnog učenja, kada imamo pristup označenim i neoznačenim podacima, iterativnim ažuriranjem parametara modela na temelju kombinacije označenih i procijenjenih oznaka za neoznačene podatke točke.

2.2.5 Polu-nadzirana metoda potpornih vektora

Metoda potpornih vektora (eng. support vector machine, SVM) je metoda strojnog učenja temeljena na teoriji statističkog učenja. SVM ima puno prednosti, kao što su čvrsta teorijska osnova, rijetkost rješenja, globalna optimizacija, nelinearnost i generalizacija. Standardni oblik SVM-a odnosi se samo na nadzirano učenje, čiji podaci za trening moraju biti označeni. Međutim, u mnogim praktičnim problemima neoznačeni podaci uvijek prevladavaju. Kada su označeni podaci rijetki, a neoznačenih podataka ima dovoljno, metoda polu-nadziranog učenja može pomoći u poboljšanju izvedbe klasifikacije SVM-a u ovoj situaciji. Stoga će se uvođenjem ideje polu-nadziranog učenja u SVM i njihovim kombiniranjem prevladati nedostaci SVM-a i dobiti bolji rezultati klasifikacije. Razvoj polu-nadzirane metode potpornih vektora uglavnom je usmjerena na to da SVM koristi male količine označenih podataka i velike količine neoznačenih podataka za treniranje i klasifikaciju. U konačnici poboljšava se točnost klasifikacije SVM-a i povećava se njegova izvedba generalizacije [25].

Bennett i Demiriz predložili su polu-nadziranu metodu potpornih vektora koja je temeljena na pretpostavci grupa. Slično ideji SVM-a, S3VM zahtijeva maksimalnu marginu za odvajanje

označenih i neoznačenih podataka. Nova optimalna granica klasifikacije mora zadovoljiti da klasifikacija na izvornim neoznačenim podacima ima najmanju pogrešku generalizacije [25]. U usporedbi sa standardnim SVM-om, čiji je cilj pronaći razdvajajuću hiperravninu s maksimalnom marginom, S3VM ima cilj pronaći razdvajajuću hiperravninu koja ne samo da razdvaja označene uzorke, već također leži u području niske gustoće svih uzoraka. Kao što je prikazano na slici 2.9, ovdje je pretpostavka razdvajanje niske gustoće (eng. low-density separation), što je proširenje pretpostavke grupiranja pod linearnim razdvajajućim hiperravninama. "+" i "−" predstavljaju označene pozitivne i označene negativne uzorke dok sive točke predstavljaju neoznačene uzorke [26].



Slika 2.9. *S3VM i razdvajanje niske gustoće* [26].

Od njegove prve implementacije primijenjen je širok spektar tehnika za rješavanje problema nekonveksne optimizacije povezanog sa S3VM-om. Kako bi riješili ovaj problem, znanstvenici koriste kvazi-Newtonov pristup u S3VM. Godine 2011. Reddy i suradnici primijenili su modificiranu kvazi-Newton metodu za S3VM. Godine 2012. Gieseke i suradnici predložili su rijetku kvazi-Newton optimizaciju za S3VM. Obojica timova učinila su učinkovita poboljšanja [25].

S povećanjem neoznačenih podataka S3VM predstavlja rastući trend u praktičnoj primjeni. Prva primjena polu-nadzirane metode potpornih vektora bila je klasifikacija teksta u kojoj je ostvarila dobre rezultate. Osim klasifikacije teksta S3VM metoda primjenjuje se u drugim područjima kao što je klasifikacija slika, prepoznavanje lica, prepoznavanje slika šarenice, klasifikacija raka, procjena učinka tjelesnog odgoja i brza identifikacija između jestivog ulja i prljavog kuhanog ulja. Što se tiče primjene u aplikacijama, S3VM metoda može se široko koristiti u nekim problemima

klasifikacije. Trenutačno možemo vidjeti da postoji mnogo aplikacija u kojima je primijenjena S3VM metoda, a potencijalne primjene su klasifikacija slika i klasifikacija teksta u kojima je metoda pokazala dobre rezultate [25].

Jedna od poznatijih S3VM metoda je transduktivna metoda potpornih vektora, predložena od strane Vladimir N. Vapnik 1998. godine. TSVM je dizajnirana za probleme binarne klasifikacije. TSVM razmatra sve moguće dodjele oznaka neoznačenih uzoraka, odnosno privremeno tretira svaki neoznačeni uzorak kao negativan ili pozitivan uzorak tijekom optimizacije. Ispitivanjem neoznačenih uzoraka sa svim mogućim dodjelama oznaka, TSVM ima cilj pronaći razdvajajuću hiperravninu koja maksimizira marginu za označene i neoznačene uzorke s dodjelom oznaka. Nakon što se odredi razdjelna hiperravnina, konačna dodjela oznake za neoznačeni uzorak je njegovo predviđanje [25] [26].

2.2.6 Primjeri korištenja polu-nadziranog učenja

Uz količinu podataka koja neprestano raste, ne postoji način da se pravodobno označi. Ako uzmemo za primjer jednu od trenutno najpopularnijih društvenih mreža TikTok, zamislimo aktivnog korisnika TikTok-a koji u prosjeku objavljuje do 15 videa dnevno. TikTok ima 1 milijardu aktivnih korisnika. U takvom scenariju, polu-nadzirano učenje može se pohvaliti širokim spektrom slučajeva upotrebe od prepoznavanja govora i slika do web sadržaja i klasifikacije tekstualnih dokumenata [12].

Što se tiče prepoznavanja govora, označavanje zvuka vrlo je zahtjevan zadatak koji zahtijeva puno vremena i resursa, tako da se polu-nadzirano učenje može koristiti za savladavanje izazova i pružanje boljih performansi. Facebook (sada Meta) uspješno je primijenio polu-nadzirano učenje, točnije metodu samo-učenja na svoje modele prepoznavanja govora i poboljšao ih. Započeli su s osnovnim modelom koji je treniran sa 100 sati audiopodataka s ljudskim bilješkama. Zatim je dodano 500 sati neoznačenih govornih podataka i korištena je metoda samo-učenja za povećanje performansi modela. Što se tiče rezultata, stopa pogrešaka na temelju riječi (eng. word error rate, WER) smanjila se za 33,9 posto, što je značajan napredak [12].

S milijardama web stranica koje predstavljaju sve vrste sadržaja, klasifikacija bi zahtijevala puno ljudskih resursa da organizira informacije na web stranicama dodjeljivanjem odgovarajućih oznaka. Kako bi se poboljšalo korisničko iskustvo koriste se razne varijacije polu-nadziranog učenja za označavanje web-sadržaja i njegovu odgovarajuću klasifikaciju. Pomoću polu-nadziranog učenja Google tražilica pronalazi sadržaj koji je najrelevantniji za određeni korisnički upit [12].

Izrada klasifikatora tekstualnog dokumenta je još jedan primjer kada se polu-nadzirano učenje može uspješno koristiti. Ovdje je metoda učinkovita jer je ljudskim anotatorima zaista teško čitati tekstove kako bi dodijelili osnovnu oznaku kao što je vrsta ili žanr.

Na primjer, klasifikator se može izgraditi na temelju neuronskih mreža dubokog učenja poput modela dugog kratkoročnog pamćenja (eng. long short-term memory, LSTM) koji je sposoban pronaći dugoročne ovisnosti u podacima i prekvalificirati prošle informacije tijekom vremena. Obučavanje neuronske mreže zahtijeva mnogo označenih i neoznačenih podataka. Polu-nadzirani okvir (eng. framework) za učenje funkcionira savršeno jer možemo uvježbati osnovni LSTM model na nekoliko tekstualnih primjera s ručno označenim najrelevantnijim riječima i zatim ga primijeniti na veći broj neoznačenih uzoraka [12].

2.2.7 Kada koristiti, a kada ne koristiti polunadzirano učenje

S obiljem neoznačenih podataka i minimalnom količinom označenih podataka kao što to inače biva, polu-nadzirano učenje pokazuje obećavajuće rezultate u zadacima klasifikacije, dok ostavlja otvorena vrata za druge zadatke strojnog učenja. U osnovi, pristup može koristiti gotovo bilo koji nadzirani algoritam uz neke potrebne izmjene. Povrh toga, polu-nadzirano učenje se dobro uklapa i u svrhe grupiranja i otkrivanja anomalija ako podaci odgovaraju profilu [12]. Iako je relativno novo područje, polu-nadzirano učenje već se pokazalo učinkovitim u mnogim područjima od kojih su neka navedena u prethodnom potpoglavlju.

Polu-nadzirano učenje nije primjenjivo na sve zadatke. Ako dio označenih podataka nije reprezentativan za cijelu distribuciju, pristup može biti neuspješan. Recimo, trebamo klasificirati slike obojenih predmeta koji iz različitih kutova imaju različit izgled. Osim ako nemamo veliku količinu označenih podataka, rezultati će biti loše točni. Ali ako imamo mnogo označenih podataka, onda polu-nadzirano učenje nije pravi izbor. Mnoge aplikacije iz stvarnog života još uvijek zahtijevaju mnogo označenih podataka, tako da nadzirano učenje neće nestati iz upotrebe u bliskoj budućnosti [12].

2.3 Mjerenje utroška energije

Algoritmi strojnog učenja odgovorni su za značajan broj izračuna. Izračuni rastu s napretkom u raznim područjima strojnog učenja. Na primjer, u područjima kao što je duboko učenje, algoritmi moraju raditi tjednima, trošeći velike količine energije. Iako postoji trend optimizacije algoritama strojnog učenja za performanse i potrošnju energije, još uvijek postoji malo znanja o tome kako procijeniti potrošnju energije algoritma. Trenutačno ne postoji jednostavan, višeplatformski pristup procjeni potrošnje energije za različite vrste algoritama. Zbog toga su poznati istraživači u

području računalne arhitekture objavili opsežan rad o pristupima za procjenu potrošnje energije [27].

Algoritmi strojnog učenja značajno su povećali svoje performanse predviđanja tijekom proteklih godina. To je vidljivo u zadacima kao što je prepoznavanje objekata, gdje algoritmi dubokog učenja nadmašuju ljudske klasifikatore. Međutim, to dolazi uz visoke troškove računanja i energije. Za izradu tako preciznih modela, količina izračuna (broj operacija u sekundi) eksponencijalno je porasla te ima izravan utjecaj na potrošnju energije [27].

Postoje izazovi prilikom mjerenja potrošnje energije računalnog programa, budući da su uključene mnoge varijable kao npr. DRAM pristupi, pogoci predmemorije (eng. cache), promašaji predmemorije itd. Postoji nekoliko metoda za mjerenje potrošnje energije računala. Tradicionalne empirijske metode za mjerenje potrošnje energije uključuju korištenje mjerača snage na različitim mjestima, od zidne utičnice do izravnih mjerenja na matičnoj ploči računala. Ovaj pristup daje stvarnu potrošnju energije u određenom trenutku, ali ne daje informacije o tome gdje se energija troši. Rafiniraniji pristupi mjerenju potrošnje energije uključuju korištenje simulatora ili korištenje brojača za praćenje performansi. Računalo troši energiju ili snagu koristeći svoje hardverske komponente. Točna količina potrošene energije ovisi o tome kako se komponente koriste i koliko energije svaka komponenta troši [27].

Budući da je ovo kompliciran i izazovan pristup, predlažu se modeli snage za procjenu ukupne potrošnje energije. Prednost korištenja simulatora je u tome što istraživaču pružaju detaljan pregled načina na koji određeni program pristupa svakoj hardverskoj komponenti. Također omogućuju instrumentaciju koja može pokazati potrošnju energije raznih funkcija programa. Glavni nedostatak korištenja simulatora su dodatni troškovi koji onemogućuju izvođenje energetskih mjerenja u stvarnom vremenu. Brojači za praćenje performansi (eng. performance monitoring counters, PMC) dostupni su u gotovo svim modernim procesorima. Oni pružaju mogućnost brojanja mikro arhitektonskih događaja procesora tijekom rada. Budući da su dostupni za svaku jezgru i mogu ispisati mnogo različitih statistika, kao što su upute po ciklusu (eng. instructions per cycle, IPC) i pristupi predmemoriji, mnogi su ih istraživači koristili za izradu energetskih modela. Osim toga, Intel je razvio energetski model nazvan RAPL koji procjenjuje potrošnju energije na temelju PMC vrijednosti [27]. Koristili smo ovo sučelje za procjenu potrošnje energije tijekom rada na ovom diplomskom radu. Iako smo se i mi fokusirali na mjerenje utroška energije, memorije i vremena u ovom diplomskom radu, nama nije bio cilj samo izmjeriti navedene utroške, već istražiti omjer između dobivenih performansi i učinkovitosti s obzirom na

potrošnju resursa prilikom korištenja nadziranog i polu-nadziranog učenja u ovisnosti o količini podataka u fazi treniranja.

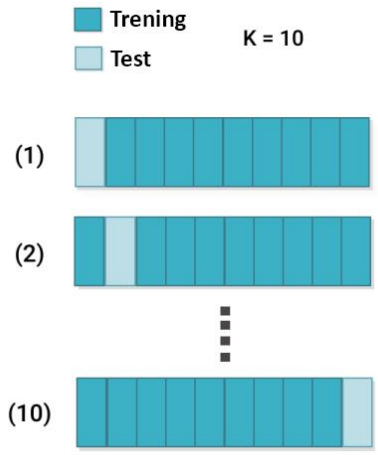
3. VREDNOVANJE MODELA

Za vrednovanje performansa modela nadziranog i polu-nadziranog učenja u ovom diplomskom radu koristili smo unakrsnu provjeru, točnost predviđanja, F1-mjeru (eng. F1-measure), geometrijsku srednju vrijednost (eng. geometric mean, g-mean) i ROC-AUC. U ovom poglavlju opisat ćemo metrike koje su korištene za vrednovanje modela nadziranog i polu-nadziranog učenja.

3.1 Unakrsna provjera

Unakrsna provjera je ključna tehnika koja se koristi u strojnom učenju za procjenu izvedbe modela i njegovu sposobnost generalizacije na nove, neviđene podatke. Uključuje sustavno dijeljenje dostupnih podataka u podskupove za treniranje i testiranje valjanosti modela više puta, pružajući pouzdaniju, stabilniju i temeljitiju procjenu od korištenja podjele na skupove za treniranje i testiranje. Unakrsna provjera pomaže u rješavanju problema kao što su pretreniranje (eng. overfitting) (kada model savršeno radi na primjerima za testiranje, ali na novim primjerima griješi) i nedovoljno treniranje (eng. underfitting) (kada model ne radi točno ni na primjerima za treniranje). Pomaže u usporedbi i odabiru odgovarajućeg modela za određeni problem prediktivnog modeliranja [28]. Unakrsnu provjeru je lako razumjeti, lako implementirati i obično ima manju pristranost od drugih metoda koje se koriste za brojanje rezultata učinkovitosti modela. Sve to čini unakrsnu provjeru moćnim alatom za odabir najboljeg modela za određeni zadatak [29].

Koncept unakrsne provjere je sljedeći. U unakrsnoj provjeri, dostupni podaci podijeljeni su u "k" podskupova ili dijelova (eng. folds) približno jednake veličine. Na koliko dijelova će se podaci podijeliti određuje sam kreator modela strojnog učenja te on najčešće iznosi 5 ili 10. Model se trenira na "k-1" dijelova, a preostali dio služi kao skup podataka za testiranje. Ovaj se proces ponavlja "k" puta, pri čemu svaki dio jednom djeluje kao skup za provjeru valjanosti koji je poznat i kao skup validacije. Mjerni podaci o izvedbi dobiveni iz svake iteracije zatim se izračunavaju u prosjeku kako bi se dobila konačna procjena izvedbe modela. Prethodno opisana metoda je ujedno i najčešće korištena metoda unakrsne provjere te se naziva k-struka unakrsna provjera. K-struku unakrsnu provjeru koristili smo i prilikom razvoja naših modela nadziranog i polu-nadziranog učenja, gdje smo ukupni skup podataka podijelili na 4 dijela (k=4). Na slici 3.1. prikazana je "k"-struka unakrsna provjera gdje su podaci podijeljeni na 10 dijelova.

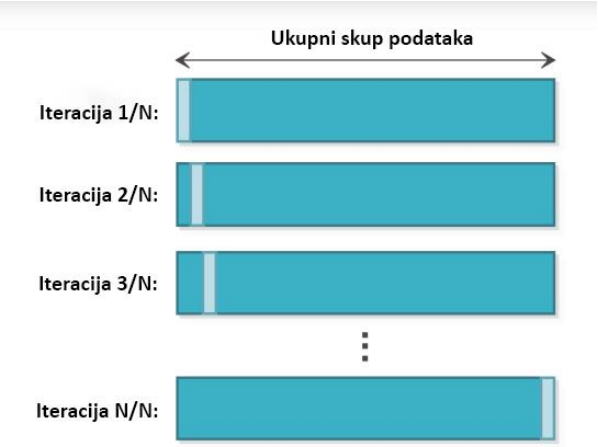


Slika 3.1. *Skica k-struke unakrsne provjere* [29]

Postoji više metoda unakrsne provjere, a neke koje se najčešće koriste su:

- k-struka unakrsna provjera
- unakrsna provjera "izdvoji jednog" (eng. leave-one-out cross-validation, LOOCV)
- ugniježđena k-struka validacija (eng. nested k-folds validation)
- stratificirana k-struka validacija (eng. stratified k-folds validation)

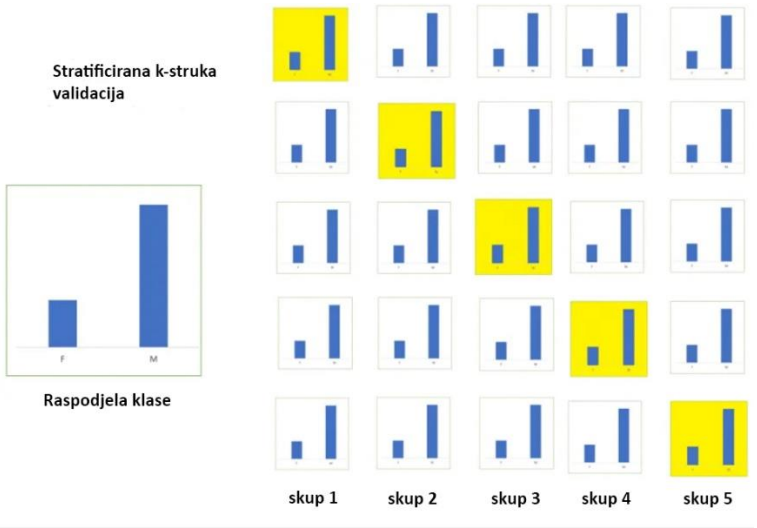
Unakrsna provjera "izdvoji jednog" je poseban slučaj "k"-struke unakrsne provjere gdje je "k" jednak broju podatkovnih točaka. U svakoj se iteraciji jedna podatkovna točka drži kao validacijski skup, a model se uvježbava na preostalih "n-1" podatkovnih točaka. Ovaj postupak se ponavlja "n" puta gdje "n" predstavlja ukupan broj podatkovnih točaka. Prednost "izdvoji jednog" unakrsne provjere je da pruža robusniju evaluaciju dok je nedostatak da može biti računski skup za velike skupove podataka.



Slika 3.2. *Unakrsna provjera "izdvoji jednog"* [29]

Za razliku od drugih tehnika unakrsne provjere ugniježđena unakrsna provjera valjanosti koristi se za procjenu izvedbe modela i istovremeno podešavanje hiperparametara. Sastoji se od vanjske petlje za procjenu modela pomoću k-struke unakrsne validacije i unutarnje petlje za podešavanje hiperparametara pomoću druge k-struke unakrsne validacije te je zbog toga i dobila naziv ugniježđena. Time se sprječava curenje informacija tijekom podešavanja hiperparametara i pruža robusniju procjenu performansa modela.

Stratificirano uzorkovanje je tehnika uzorkovanja gdje se uzorci biraju u istom omjeru (dijeleći populaciju u skupine koje se nazivaju "stratumi" na temelju karakteristika) u kakvom se pojavljuju u populaciji. Na primjer, ako populacija ima 20% muških i 80% ženskih subjekata, tada dijelimo populaciju u dvije ("muške" i "ženske") skupine i odabiremo 20% uzorka iz "muške" skupine i 80% uzorka iz "ženske" skupine [30]. Stratificirana k-struka validacija je varijanta tradicionalne tehnike k-struka unakrsne provjere. Glavni cilj stratificirane unakrsne provjere je osigurati da je distribucija klasa u svakom struku slična ukupnoj distribuciji klasa u skupu podataka, što pomaže u poboljšanju pouzdanosti procjene izvedbe za modele strojnog učenja. Jednoliku raspodjelu u svakom struku moguće je vidjeti na slici 3.3. gdje je prikazana stratificirana k-struka validacija. Posebno je korisna kada se radi o neuravnoteženim skupovima podataka gdje određene klase imaju znatno manje uzoraka te sprječava situacije u kojima bi određena klasa s malo uzoraka mogla biti u potpunosti izostavljena, što dovodi do pristranih procjena izvedbe.



Slika 3.3. *Stratificirana k-struka validacija* [28]

Unakrsna provjera valjanosti moćan je alat. Svaki podatkovni znanstvenik trebao bi biti upoznat s njome. Unakrsna provjera valjanosti bitna je tehnika u tijeku rada strojnog učenja, koja služi kao informativan i pouzdan alat za procjenu, odabir i podešavanje modela. Rješavanjem uobičajenih izazova kao što su neuravnoteženi podaci, pretreniranje, nedovoljno treniranje, unakrsna validacija omogućuje podatkovnim znanstvenicima da izgrade robusnije i preciznije modele za širok raspon aplikacija u stvarnom svijetu [29].

3.2 Metrike vrednovanja

Točnost (eng. accuracy, Acc) Točnost je vjerojatno najpoznatija metoda provjere valjanosti modela strojnog učenja koja se koristi za vrednovanje problema klasifikacije. Jedan od razloga njezine široke popularnosti je njezina jednostavnost. Lako ju je razumjeti i implementirati. Točnost je dobra metrika za procjenu izvedbe modela u jednostavnijim slučajevima. Međutim, u scenarijima stvarnog života problemi modeliranja rijetko su jednostavni. Često se pojavljuju izazovi kao što su neuravnoteženi skupovi podataka ili problemi klasifikacije s više klasa ili više oznaka. Ponekad nam visoka točnost možda i nije cilj. Kako rješavamo složenije probleme strojnog učenja, izračunavanje i korištenje točnosti postaje manje očito i zahtijeva dodatna razmatranja. Iz tog je razloga važno razumjeti što je to točnost, kako se izračunava i koje su joj slabosti u različitim kontekstima strojnog učenja [31].

Točnost se koristi u problemima klasifikacije kako bi se odredio postotak točnih predviđanja modela. Ocjena metrike točnosti u strojnom učenju je procjena koja mjeri broj točnih predviđanja napravljenih od strane modela u odnosu na ukupni broj napravljenih predviđanja. Izračunava se

formulom koja slijedi u nastavku tako da broj točnih predviđanja podijelimo s ukupnim brojem predviđanja [31].

$$Točnost = \frac{točne\ predikcije}{ukupan\ broj\ predikcija} \tag{3.1}$$

Ova formula pruža lako razumljivu definiciju točnosti koja pretpostavlja problem binarne klasifikacije. Jedan od bitnijih nedostataka točnosti je da nije dobra metrika za vrednovanje modela kada imamo neuravnotežene podatke te zbog toga koristimo metrike za vrednovanje koje su opisane u nastavku. Jedan od načina rješavanja problema neravnoteže u klasama je rad na našem uzorku. S određenim metodama uzorkovanja možemo ponovno uzorkovati svoj skup podataka na način da podaci više nisu neuravnoteženi. Zatim možemo ponovno koristiti točnost kao metriku. Drugi način za rješavanje problema neravnoteže klasa je korištenje boljih metrika za vrednovanje modela kao što je F1-mjera, koja uzima u obzir ne samo broj pogrešaka u predviđanju koje čini naš model, već također promatra vrstu pogrešaka koje naš model radi [32].

F1-mjera koja se još naziva i F1-rezultat (eng. F1-score) je metrika strojnog učenja koja se može koristiti u modelima klasifikacije. F1-mjera predstavlja poboljšanje dviju jednostavnijih metrika valuacije, a to su preciznost (eng. precision) i odziv (eng. recall) koje su i temelj F1-mjere. Odziv se još može nazvati osjetljivost (eng. sensitivity). Preciznost i odziv su najčešće metrike koje uzimaju u obzir klasnu neravnotežu. Preciznost predstavlja prvi dio F1-mjere koja se također može koristiti kao individualna metrika strojnog učenja. Preciznost se izračunava formulom u nastavku [32].

$$Preciznost = \frac{TP}{TP + FP} \tag{3.2}$$

Prije tumačenja formule bitno je razumjeti pojmove kao što su:

- Istinski pozitivni (eng. True positive - TP) – Broj opservacija koje su ispravno klasificirane kao pozitivne.
- Lažno pozitivni (eng. False positive - FP) – Broj opservacija koje su klasificirane kao pozitivne, međutim prava vrijednost im je negativna.
- Lažno negativni (eng. False negative - FN) – Broj opservacija koje su klasificirane kao negativne, međutim prava vrijednost im je pozitivna.
- Istinski negativni (eng. True negative - TN) – Broj opservacija koje su ispravno klasificirane kao negativne.

Ovu formulu možemo protumačiti na sljedeći način. Unutar svega što je predviđeno kao pozitivno, metrika preciznost broji postotak koji je točan pa stoga imamo:

- Neprecizan model može pronaći puno pozitivno predviđenih podataka, ali modelova metoda odabira nije najbolja: također pogrešno otkriva mnoge pozitivne podatke koje zapravo nisu pozitivni.
- Precizan model je vrlo "čist ": možda ne pronalazi sve pozitivne, ali one koje model smatra pozitivnima vrlo je vjerojatno da su točne.

Odziv predstavlja drugi dio metrike F1-mjera. Odziv je metrika koja se također može koristiti kao individualna metrika strojnog učenja. Formula za računanje metrike odziva dana je u nastavku [32].

$$Odziv = \frac{TP}{TP + FN} \tag{3.3}$$

Formulu odziva možemo protumačiti na sljedeći način. Od svega što je zapravo pozitivno, koliko je model uspio pronaći:

- Model s visokim brojem odziva uspješno pronalazi sve pozitivne slučajeve u podacima, čak iako oni također mogu pogrešno identificirati neke negativne slučajeve kao pozitivne slučajeve.
- Model s niskim odzivom nije u mogućnosti pronaći sve (ili veliki dio) pozitivne slučajeve u podacima.

Metrike preciznost i odziv predstavljaju dva građevna bloka F1-mjere. Cilj F1 metrike je kombiniranje metrike preciznosti i odziva u jednu metriku. Međutim, F1-mjera dizajnirana je da dobro funkcionira na neuravnoteženim podacima. F1-mjera definirana je kao harmonijska sredina preciznosti i odziva, gdje harmonijska sredina predstavlja alternativnu metriku za uobičajenu aritmetičku sredinu. Harmonijska sredina često se koristi kada se računa prosječna stopa. U F1-mjeri izračunavamo prosjek preciznosti i odziva. Obje su stope, što čini korištenje harmonijske sredine logičnim izborom. Formula F1-mjere prikazana je u nastavku [32].

$$F1\ mjera = 2 \times \frac{Preciznost \times Odziv}{Preciznost + Odziv} \tag{3.4}$$

Budući da je F1-mjera prosjek metrika preciznosti i odziva, to znači da F1-mjera daje jednaku težinu preciznosti i prisjećanju:

- F1-mjera modela će biti visoka ako su preciznost i odziv visoki
- F1-mjera modela će biti niska ako su preciznost i odziv niski
- F1-mjera modela će biti srednja ako je jedna od metrika preciznost i odziva niska, a druga visoka.

Geometrijska sredina (g-sredina) je statistička mjera koja se koristi u strojnom učenju za procjenu izvedbe modela klasifikacije, posebno kada se radi o neuravnoteženim skupovima podataka ili scenarijima u kojima je distribucija klasa iskrivljena. Za razliku od aritmetičke sredine, koja zbraja vrijednosti i dijeli s brojem vrijednosti, g-sredina izračunava "n-ti korijen" umnoška "n" vrijednosti. U kontekstu klasifikacije, ove vrijednosti obično predstavljaju metrike poput specifičnosti i odziva. Specifičnosti izračunavamo sljedećom formulom.

$$Specifičnost = \frac{TN}{TN + FP} \tag{3.5}$$

G-sredina je osobito korisna kada želimo uravnotežiti učinke lažno pozitivnih i lažno negativnih rezultata. Pruža nam metriku koja uzima u obzir i sposobnost modela da ispravno identificira pozitivne slučajeve (osjetljivost ili odziv) i njegovu sposobnost da izbjegne pogrešnu klasifikaciju negativnih slučajeva (specifičnost). To čini metriku g-sredine prikladnim izborom kada točnost možda ne odražava točno performanse modela zbog neravnoteže klase. G-sredinu izračunavamo sljedećom formulom [33].

$$G - sredina = \sqrt{Specifičnost * Odziv} \tag{3.6}$$

Uzimanjem kvadratnog korijena produkta s pecifičnosti i odziva, g-sredina smanjuje utjecaj ekstremnih vrijednosti i pruža uravnoteženiju mjeru performansa modela. Rješava slučajeve u kojima su ili specifičnost ili odziv niski tako da promiče uravnoteženu razmjenu između to dvoje [33].

Ukratko, geometrijska sredina vrijedan je alat u strojnom učenju za procjenu modela klasifikacije kada imamo scenarije u kojima je ključno postizanje ravnoteže između specifičnosti i odziva. G-sredina pomaže uzeti u obzir nijanse neuravnoteženih skupova podataka i pruža sveobuhvatniju perspektivu o učinkovitosti modela.

Krivulja odnosa specifičnosti i odziva klasifikatora (eng. Receiver Operator Characteristic, ROC) je grafički prikaz dijagnostičkog kapaciteta binarnog klasifikatora. ROC krivulja daje vizualno i opsežno izvješće o točnosti predviđanja. ROC krivulja je naširoko primjenjiva, neovisno o izvoru

predviđanja. Njezino podrijetlo je u teoriji detekcije signala, ali primjenjiva je u raznim područjima kao što su primjerice prirodne katastrofe, radiografija, medicina i strojno učenje. Široka primjena ROC krivulje u ovim područjima dovoljno govori o njezinoj važnosti [34] [35].

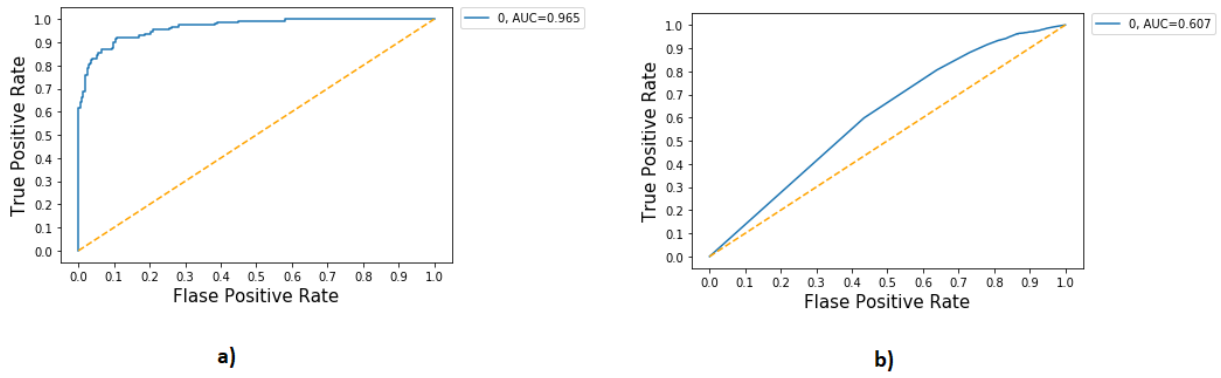
Stvarna pozitivna stopa (eng. true positive rate, TPR), koja je u stvari odziv, iscrtava se u odnosu na lažno pozitivnu stopu (eng. false positive rate, FPR), koju još nazivamo lažni alarm (eng. false alarm, fall-out), kako bi se stvorila ROC krivulja u strojnom učenju. Stvarna pozitivna stopa je postotak svih pozitivnih opažanja za koje se ispravno očekivalo da budu pozitivni. Slično tome, lažno pozitivna stopa je udio negativnih opažanja koja su projicirana pogrešno kao pozitivna. Formula za lažnu pozitivnu stopu ili lažni alarm slijedi u nastavku [34].

$$Lažni\ alarm = \frac{FP}{FP + TN} \tag{3.7}$$

Stvarnu pozitivnu stopu možemo prikazati na primjeru gdje u medicinskom testiranju predstavlja postotak pacijenata za koje se točno prepozna da su pozitivni na dotičnu bolest.

Ravnoteža između TPR i FPR prikazana je ROC krivuljom. Klasifikatori s krivuljama koje su bliže gornjem lijevom kutu imaju bolje rezultate. Primjer takvog klasifikatora možemo vidjeti na slici 3.3. a). Od nasumičnog klasifikatora se očekuje da će dati dijagonalne točke (FPR = TPR) kao osnovnu liniju. Test postaje manje točan kada se krivulja približi dijagonali ROC prostora od 45 stupnjeva, što možemo vidjeti na slici 3.3. b). Bitno je napomenuti da je ROC krivulja neovisna o raspodjeli klasa. To je čini idealnom za testiranje klasifikatora koji predviđaju rijetke događaje poput prirodnih katastrofa ili bolesti [34] [35].

Kada se uspoređuju različiti klasifikatori može biti korisno sažeti učinak svakog klasifikatora u jednu mjeru. Jedna od tipičnih metoda je izračunavanje površine ispod ROC krivulje (eng. area under curve, AUC). AUC koji se približava 0 ukazuje na loš model, što znači da ima najnižu mjeru odvojivosti. Takav model predviđa da će 1 biti 0, a 0 da će biti 1. Kada je površina ispod krivulje jednaka 0,5, tada model nema sposobnost razlikovanja između klasa. Izvanredan model ima AUC blizu 1, što ukazuje da ima visoku razinu odvajanja [34].



Slika 3.5. *ROC krivulja boljeg i lošijeg modela sa AUC vrijednostima*

Zaključak bi bio da kada imamo priliku za to, svakako bi prilikom razvoja modela trebali pogledati više različitih metrika za svaki model koji isprobavamo. Svaka metrika ima svoje prednosti i nedostatke i svaka od njih će nam pružiti specifične informacije o prednostima i nedostacima našeg modela kojeg razvijamo.

4. RAZVOJNO I TESTNO OKRUŽENJE

U ovom poglavlju opisano je razvojno i testno okruženje. Programski jezik koji je korišten za razvoj modela nadziranog i polu-nadziranog učenja sa korištenim knjižnicama. Osim knjižnica korištenih za razvoj modela opisat ćemo i knjižnicu korištenu za mjerenje utroška energije, memorije i vremena prilikom izvođenja programa. Kako rezultati mjerenja RAM-a, CPU-a i vremena trajanja treniranja modela ovisi o računalu na kojem se izvodi mjerenje u nastavku slijedi opis komponenti korištenog računala.

4.1 Alati za razvoj modela strojnog učenja

Odabir programskog jezika za strojno učenje ovisi o različitim čimbenicima, uključujući naše poznavanje programskog jezika, specifične zadatke strojnog učenja kojima se želimo baviti, dostupne knjižnice i okvire (eng. framework) te cjelokupni ekosustav. Najčešće korišteni programski jezici sa dobrom podrškom za strojno učenje su:

- Python
- R
- Julia
- Java
- C++

Za izradu modela strojnog učenja za potrebe ovog diplomskog rada korišten je Python programski jezik. Razlog odabira Python-a je prijašnje pozitivno iskustvo korištenja u području strojnog učenja. Python je interpreterski programski jezik jednostavne sintakse koji pruža napredan analitički proces. Python je daleko najpopularniji programski jezik korišten za strojno učenje. Ima bogat ekosustav knjižnica i okvira koji olakšavaju razvoj, implementaciju i eksperimentiranje modela strojnog učenja. Programeri mogu koristiti Python za razvoj raznih aplikacija jer se dobro integrira s drugim softverom, dok ga njegova jednostavna sintaksa čini dobrim izborom za kodiranje algoritama i suradnju među timovima. Python također može raditi na bilo kojem operativnom sustavu, od Windowsa do Linuxa, Unixa, macOS-a i drugih. Ono što je možda najvažnije, Python je jednostavan za čitanje, omiljen kod ogromne zajednice programera koji također doprinose razvojem novih paketa koji olakšavaju strojno učenje [36].

4.1.1 Programsko okruženje

Model strojnog učenja za potrebe ovog diplomskog rada razvijan je u Spyder programskom okruženju. Spyder je moćno znanstveno integrirano razvojno okruženje (eng. integrated development environment, IDE) napisano u Pythonu, za Python, a dizajnirano od strane i za inženjere, znanstvenike i analitičare podataka. Sadrži kombinaciju napredne analize, uređivanja, otklanjanja pogrešaka i funkcionalnosti profiliranja sveobuhvatnog razvojnog alata s istraživanjem podataka, interaktivnim izvođenjem, dubokim pregledom i prekrasnim mogućnostima vizualizacije znanstvenog paketa. Spyder nudi ugrađenu integraciju s mnogim popularnim znanstvenim paketima kao što su Pandas, NumPy, SciPy, IPython, Matplotlib, QtConsole, SymPy i drugi. Osim brojnih ugrađenih značajki, Spyder se može još više proširiti putem dodataka trećih strana. Spyder se također može koristiti kao knjižnica proširenja PyQt5, što nam omogućuje da nadogradimo njegovu funkcionalnost i ugradimo njegove komponente, poput naprednog uređivača ili interaktivne konzole, u vlastiti softver [37].

4.1.2 Python knjižnice za razvoj modela

NumPy je temeljna Python biblioteka za numeričko računanje. Pruža podršku za stvaranje nizova (jednodimenzionalnih i višedimenzionalnih), manipulaciju nizova te razne matematičke operacije na tim nizovima. Jedna od osnovnih značajki koje NumPy pruža je ndarray (n-dimenzionalni niz), koji može predstavljati vektore, matrice i višedimenzionalne nizove. Ovi nizovi su učinkoviti za numeričke proračune i memorijski učinkovitu pohranu. Pruža izvođenje matematičkih operacija s elementima u nizovima bez potrebe za eksplicitnim petljama. Numpy uključuje širok raspon funkcija, uključujući aritmetičke, trigonometrijske funkcije, eksponencijalne funkcije i operacije linearne algebre [38].

Scikit-learn (skraćeno sklearn) jedna je od najcjenjenijih, najsnažnijih i najčešće korištenih knjižnica za strojno učenje u Pythonu. Pruža izbor učinkovitih alata za strojno učenje, rudarenje, analizu podataka i statističko modeliranje uključujući klasifikaciju, regresiju, grupiranje i smanjenje dimenzionalnosti putem konzistentnog sučelja u Pythonu. Ova biblioteka, koja je uglavnom napisana u Pythonu, izgrađena je na NumPy, SciPy i Matplotlib knjižnicama [39].

Neke od prednosti sklearn knjižnice su:

- Jednostavnost korištenja - Scikit-learn poznat je po svom korisničkom API-ju i dosljednom sučelju za razne algoritme strojnog učenja. To olakšava prebacivanje između različitih algoritama bez potrebe za značajnim prepisivanjem koda.

- Širok raspon algoritama - Knjižnica sadrži širok spektar algoritama strojnog učenja, uključujući regresiju, klasifikaciju, grupiranje, smanjenje dimenzionalnosti i još mnogo toga [40].
- Kvalitetna dokumentacija - Scikit-learn pruža opsežnu dokumentaciju, upute i primjere koji olakšavaju razumijevanje i implementaciju algoritama strojnog učenja.
- Integracija s drugim knjižnicama – Pruža neprimjetnu integraciju s drugim Python bibliotekama kao što su Matplotlib, pandas i NumPy, omogućujući nam učinkovitu prethodnu obradu podataka, manipuliranje podacima i vizualizaciju rezultata.
- Vrednovanje modela - Scikit-learn pruža funkcije za procjenu izvedbe modela, uključujući metrike kao što su točnost, preciznost, odziv, F1-mjera i više. Također podržava razne tehnike unakrsne provjere [41].

Pandas je moćna i široko korištena Python knjižnica za podatkovnu manipulaciju i analizu izgrađena na temelju NumPy knjižnice. Pruža dvije primarne podatkovne strukture, koje se nazivaju serije (eng. series) i podatkovni okviri (eng. DataFrames), i funkcije koje olakšavaju rad sa strukturiranim podacima, kao što su tablični podaci u obliku tablica ili proračunskih tablica (slična Excel propračunskoj tablici). Jedna od bitnijih prednosti Pandas knjižnice je ta da omogućuje različite tipove podataka po stupcima što ga čini pristupačnim za obradu i analizu podataka. Još jedna od značajki je sposobnost čitanja različitih datotečnih formata, uključujući CSV, Excel, SQL baze podataka i druge. To olakšava uvoz i izvoz podataka za analizu [42].

Matplotlib je široko korištena, multiplatformska knjižnica za stvaranje statičkih, interaktivnih i animiranih 2D vizualizacija podataka izgrađenih na NumPy nizovima. Jedna od najvećih prednosti vizualizacije je ta što nam omogućuje vizualni pristup ogromnim količinama podataka u lako probavljivim vizualizacijama. Matplotlib se sastoji od nekoliko dijagrama kao što su linija, stupac, raspršenost, histogram itd. Matplotlib također nudi brojne mogućnosti prilagodbe, kao što su mijenjanje boja, stilova linija i stilova markera. Jedna od glavnih prednosti Matplotliba je njegova sposobnost stvaranja kvalitetnih slika koje se mogu izvesti u različite formate, kao što su SVG, PDF i PNG. Međutim, stvaranje složenih vizualizacija s Matplotlibom može biti dugotrajno i zahtijeva puno koda [43].

4.1.3 PyRAPL

Kao što je rečeno u potpoglavlju 2.3, za mjerenje utroška energije, memorije i vremena u ovom diplomskom radu koristili smo se Intelovim energetskim modelom pyRAPL koji predstavlja inačicu koja se koristi za Python programski jezik. pyRAPL je softverski alat koji služi za mjerenje energetskog otiska računala tijekom izvođenja dijela Python koda. pyRAPL koristi tehnologiju

Intel "Running Average Power Limit" (RAPL) koja procjenjuje potrošnju energije CPU-a. Ova je tehnologija dostupna na Intel CPU-u od Sandy Bridge generacije. Jezgra Sandy Bridge predstavljena je 9. siječnja 2011., kada je Intel lansirao prve desktop i mobilne procesore core i5 i core i7, temeljene na Sandy Bridge mikroarhitekturi. Osim mjerenja vremena pyRAPL može mjeriti potrošnju energije sljedećih CPU domena, a to su paket CPU utičnica (eng. socket), dinamički ram (eng. dynamic random access memory, DRAM) (za poslužiteljske arhitekture) i grafički procesor (eng. graphic processing unit, GPU) (za klijentsku arhitekture). Trenutno se pyRAPL oslanja na API jezgre (eng. kernel) Linuxa za dobivanje očitanja energije iz RAPL tehnologije. Kao rezultat toga, praćenje energije za CPU, RAM i integrirani GPU nije dostupno na Windows ili MacOS platformama [44]. Prilikom mjerenja navedenih domena treba imati na umu da navedena potrošnja energije nije samo potrošnja energije koda koji izvodimo. Uključuje ukupnu potrošnju energije svih procesa koji se izvode na računalu tijekom tog razdoblja, uključujući operativni sustav i druge aplikacije. Stoga se preporučuje uklanjanje svih dodatnih programa koji bi mogli promijeniti potrošnju energije računala na kojem se izvode eksperimenti i zadržavanje samo koda koji se mjeri (tj. bez dodatnih aplikacija, poput grafičkog sučelja, zadataka koji se izvode u pozadina, itd.). Ovo je najbolji način za određivanje stvarne potrošnje energije izmjerenog koda [45].

Informacije o potrošnji CPU-a i RAM-a osim korištenja pyRAPL-a možemo dobiti korištenjem "psutil" knjižnice. Jedna od popularnih Python knjižnica "psutil" dizajnirana je da ponudi korisničko sučelje za dobivanje informacija o sustavu i upravljanje procesima. Akronim "psutil" predstavlja "uslužne programe procesa i sustava", a posebno je dizajnirana da bude kompatibilna s različitim operativnim sustavima kao što su Linux, Windows, macOS i BSD. Psutil omogućuje pristup informacijama o pokrenutim procesima na sustavu. Može dohvatiti pojedinosti kao što su ID-evi procesa, imena, potrošnja energije CPU-a i upotreba RAM memorije, vrijeme stvaranja procesa, status i još mnogo toga. Kada bi naš fokus bio na dobivanju podataka o korištenju procesora i RAM-a za opće praćenje sustava i upravljanje procesima, izbor bi pao na psutil knjižnicu. Iako možda ne pruža izravna mjerenja potrošnje energije, može ponuditi vrijedan uvid u performanse sustava i korištenje resursa. Kako je pyRAPL posebno dizajniran za mjerenje snage i nadzor na Intel CPU-ima te pruža mogućnost mjerenja samo željenog dijela programskog koda izbor je pao negovo korištenje [46].

Prije samog mjerenja pomoću parametara funkcije pyRAPL.setup možemo jednostavno konfigurirati što ćemo mjeriti, koji uređaj i koju utičnicu nadzirati. Sljedeći primjer predstavlja dio programskog koda korištenog u ovom projektu gdje vršimo nadziranje potrošnje CPU energije i

potrošnju RAM memorije na svim CPU utičnicama. Nakon definiranja što mjerimo stvaramo varijablu pyRAPL klase (u našem slučaju nazvali smo je measure) kojoj dodjeljujemo oznaku. Stvorena pyRAPL varijabla sadržavat će podatke mjerenja kojima možemo pristupiti sa measure.result gdje su vrijeme trajanja mjerenja izraženi u mikro sekundama (μs), a potrošnja CPU-a i RAM memorije izraženi u mikro džulima (μJ). Kako bi mjerili energiju izvršavanja samo određenog programskog koda mjerenje započinjemo sa measure.begin() kojeg postavljamo prije programskog koda kojeg želimo mjeriti. U našem programskom kodu mjerimo potrošnju energije treniranja modela nadziranog učenja gdje mjerenje završavamo sa measure.end() metodom.

```
# pyRAPL measurement
pyRAPL.setup()
rapl_rez_SL = pd.DataFrame()
measure = pyRAPL.Measurement('supervised')
measure.begin()

########## Step 2 - Model Fitting ##########
model = RandomForestClassifier(n_estimators = 100, max_depth = 5,
max_features=round(math.sqrt(25)), random_state=0)

# Fit the model
clf = model.fit(X_baseline, y_baseline)

measure.end()
print(measure.result)
```

4.2 Testno okruženje

Komponente osobnog računala imaju značajnu ulogu pri obučavanju modela strojnog učenja. Performanse i učinkovitost korištenog računala mogu imati izravan utjecaj na to koliko brzo i učinkovito možemo trenirati modele strojnog učenja, posebno za složene algoritme i velike skupove podataka. Neke od ključnih komponenti koje treba uzeti u obzir prilikom treniranja modela su CPU, GPU, RAM i uređaj za pohranu podataka.

Snažan CPU može značajno ubrzati vrijeme treniranja modela, posebno kada se koriste algoritmi koji zahtjevaju više vremena na CPU. Više jezgri i veće brzine takta također su korisne za paralelnu obradu i multitasking. CPU s više jezgri može biti prednost s obzirom da neki okviri za strojno učenje mogu iskoristiti više jezgri za ubrzavanje određenih zadataka [47]. Računalo koje je korišteno za razvoj i testiranje ovog diplomskog rada sadži Intel Core i5-9300H CPU. Intel Core i5-9300H je četverojezgreni procesor za prijenosna računala koji pripada obitelji 9. generacije Intel

Core "Coffee Lake" sa osnovnom frekvencijom od 2.4 GHz dok mu maksimalna turbo frekvencija iznosi 4.10 GHz [48].

GPU je procesor koji koristi ubrzane izračune za prikaz intenzivnih slika i grafika visoke razlučivosti. Osim njihove glavne zadaće (renderiranje 2D i 3D slika, videa i animacija na računalu) današnji GPU-ovi se koriste u aplikacijama daleko izvan grafičke obrade, uključujući veliku analitiku i strojno učenje. GPU-ovi su prikladni za obuku modela dubokog učenja i nekih drugih algoritama strojnog učenja zbog svoje sposobnosti izvođenja paralelnih izračuna. To rezultira bržom obradom specijaliziranih računalnih zadataka. Okviri dubokog učenja kao što su TensorFlow i PyTorch mogu iskoristiti GPU-ove za ubrzavanje vremena treniranja modela [49]. Računalo koje je korišteno za razvoj i testiranje ovog diplomskog rada sadži NVIDIA GeForce GTX 1650 4 GB.

Kada je potrebno rukovati sa velikim skupovima podataka bez ograničenja memorije dovoljna količina RAM-a je neophodna. Za zadatke koji zahtijevaju veliku memoriju, više RAM-a može spriječiti padove i usporavanje uzorkovane nedostatkom RAM memorije. Računalo koje je korišteno za razvoj i testiranje ovog diplomskog rada sadži 24 GB DDR4-3200 RAM-a.

Brza pohrana, kao što su SSD (eng. Solid-State Drives), može značajno smanjiti vrijeme učitavanja potrebnih podataka i poboljšati ukupni odziv sustava. Zbog potrebe čestog čitanja i pisanja pri radu s velikim skupovima podataka tijekom treniranja modela SSD-ovi su osobito korisni. Za pohranu podataka potrebnih za treniranje i testiranje modela korištena je SAMSUNG MZVLB512HAJQ-000H1 SSD memorija, 512 GB kapaciteta.

5. REZULTATI

Koristeći neke od prethodno navedenih metoda, provedeno je istraživanje stvaranja modela nadziranog i polu-nadziranog učenja na nekoliko različitih skupova podataka. Prilikom stvaranja modela predviđanja provedeno je mjerenje utroška energije. U ovom poglavlju slijedi opis skupova podataka koji su korišteni u ovom diplomskom radu za razvoj nadziranog i polu-nadziranog modela predviđanja. Osim opisa skupova podataka slijedi i opis razvoja modela predviđanja od podijele skupa podataka za treniranje i testiranje, polu-nadziranoj metodi korištenoj za pseudo-označavanje pa sve do unakrsne provjere korištene za potrebe diplomskog rada. Završetkom opisa istraživanja i razvoja modela slijedi opis rezultata za pojedine skupove podataka.

5.1 Opis istraživanja

5.1.1 Skupovi podataka

Za razvoj modela nadziranog i polu-nadziranog učenja odabrano je 5 skupova podataka na kojima se vršilo istraživanje. Svi skupovi podataka preuzeti su s UCI online repozitorija skupova podataka za strojno učenje [50]. Podaci su prikazani u sljedećoj tablici s kratkim opisom, brojem instanci i značajki te brojem 0 i 1. Odabrani skupovi podataka variraju u broju instanci i značajki kako bi usporedili ponašanje, potrošnju resursa i preciznost modela nadziranog i polu-nadziranog učenja na veće i manje skupove podataka. Binarna klasifikacija je zadatak strojnog učenja gdje je cilj kategorizirati ulazne podatke u jednu od dvije moguće klase.

Tablica 5.1. *Skupovi podataka korišteni za razvoj modela predviđanja*

Skup podataka	Kratki opis	Broj instanci	Broj značajki	Broj 0/1
Tumor	Je li uzorak maligan ili benigan	569	31	0 = 357 1 = 212
Banka	Hoće li se klijent pretplatiti na oročeni depozit	45211	17	0 = 39922 1 = 5289
50k	Zarađuje li korisnik > ili < od 50k$	32561	15	0 = 24720 1 = 7841
Kupon	Hoće li klijent iskoristiti kupon	12684	26	0 = 5474 1 = 7210
Gljiva	Je li uzorak gljive otrovna ili jestiva	8124	23	0 = 4208 1 = 3916

5.1.2 Razvoj modela predviđanja

Kako bi razvili modele predviđanja nadziranog i polu-nadziranog učenja, prvo su učitani podaci potrebni za razvoj iz csv datoteke. Učitavanje podataka i obrada podataka odrađeni su korištenjem funkcija iz pandas knjižnice. Podaci potrebni za razvoj modela pohranjeni su u podatkovnim okvirima koji predstavljaju primarnu pandas podatkovnu strukturu.

Moramo napraviti još par stvari prije početka treniranja modela. Potrebno je napraviti podjelu podataka, koja predstavlja jednu od ključnih stvari prilikom treniranja modela. Podjela podataka za stvaranje modela uključuje dijeljenje označenih i neoznačenih podataka u različite podskupove kako bi se olakšao razvoj modela, vrednovanje i treniranje. Glavna zadaća dijeljenja podataka je osigurati da se naš model trenira na jednom skupu podataka, a testira na drugom kako bi se spriječilo pretreniranje. Podjelu podataka započinje se dijeljenjem označenih podataka u dva glavna podskupa, a to su skup za treniranje i skup za testiranje modela. Skup za treniranje koristi se za treniranje parametara modela, dok se skup za testiranje koristi za podešavanje hiperparametara i procjenu izvedbe modela tijekom razvoja. Skup podataka u ovom diplomskom radu podijeljen je u omjeru 75/25 gdje je 75% podataka korišteno za treniranje modela dok je preostalih 25% korišteno za testiranje modela.

Prvo je razvijan model nadziranog učenja. Kako se radi o nadziranom učenju gdje se za treniranje modela koriste samo označeni podaci, a označeni podaci su skupi, za treniranje modela korišteno je samo 5% označenih podataka. Kasnije se u svrhu istraživanja, gdje su rezultati prikazani u sljedećem poglavlju, povećavao postotak označenih podataka. Sada kada su podaci spremni, treniran je klasifikacijski model nadziranog učenja na označenim podacima kako bi se uspostavila referentna vrijednost modela. To nam omogućava da prosudimo je li polu-nadzirani pristup koji je razvijan u sljedećem koraku bolji ili lošiji od standardnog nadziranog modela. Prilikom treniranja modela mjerimo vrijeme trajanja treniranja modela, potrošnju RAM-a i CPU-a korištenjem pyRAPL knjižnice opisane u poglavlju 4.1.3 programskim kodom prikazanim u istom poglavlju. Za razvoj klasifikacijskog modela možemo odabrati bilo koji klasifikacijski algoritam kao što su npr. metoda potpornih vektora, logistička regresija, metoda najbližeg susjeda, stabla odluke (eng. decision tree), slučajne šume (eng. random forest), itd. Završetkom razvoja modela slijedi testiranje modela čije rezultate pohranjujemo u pandas podatkovne okvire.

Nakon razvoja i testiranja modela nadziranog učenja slijedi razvoj polu-nadziranog modela, gdje je odluka pala na metodu samo-učenja. Samo-učenje je polu-nadzirana tehnika učenja gdje se

treniranje modela odvija na označenim podacima, a zatim se model koristi za označavanje dodatnih neoznačenih podatkovnih točaka. Proces je iterativan, a novooznačene podatkovne točke uključene su u skup podataka za trening kako bi se poboljšala izvedba modela. Metoda samo-učenja detaljnije je opisana u potpoglavlju 2.2.1. Uključivanjem neoznačenih podataka tj. kombiniranjem označenih podataka za treniranje s neoznačenim podacima stvaramo veći kombinirani skup podataka. Ovaj je pristup posebno koristan kada su označeni podaci ograničeni, ali je dostupan velik skup neoznačenih podataka. Budući da je naš cilj procijeniti izvedbu nadziranog modela u usporedbi s polu-nadziranim modelom, model pojedinog učenja treniran je na istim 5% označenim podacima kako bi rezultati bili korektniji.

Za razvoj koristimo Sklearnovu metodu za samo-učenje, koja se naziva "SelfTrainignClassifier", koristeći isti algoritam za klasifikator kao za razvoj nadziranog modela kako bi usporedba bila korektna. Bitna stavka je da možemo odabrati bilo koji nadzirani algoritam klasifikacije za korištenje unutar "SelfTrainignClassifier" metode. Metoda se temelji na Yarkowskyjevom algoritmu koji omogućuje određenom nadziranom klasifikatoru da funkcionira kao polu-nadzirani klasifikator, dopuštajući mu da uči iz neoznačenih podataka. To postiže iterativnim predviđanjem pseudo-oznaka za neoznačene podatke i njihovim dodavanjem u skup za treniranje [51]. "SelfTrainignClassifier" korištena za razvoj polu-nadziranog modela za potrebe ovog diplomskog rada prikazana je u programskom kodu koji slijedi.

```
self_training_model = SelfTrainingClassifier(base_estimator=model,
                                              threshold=0.75,
                                              criterion='threshold',
                                              max_iter=10,
                                              verbose=False)
```

Značenje atributa "SelfTrainignClassifier" metode:

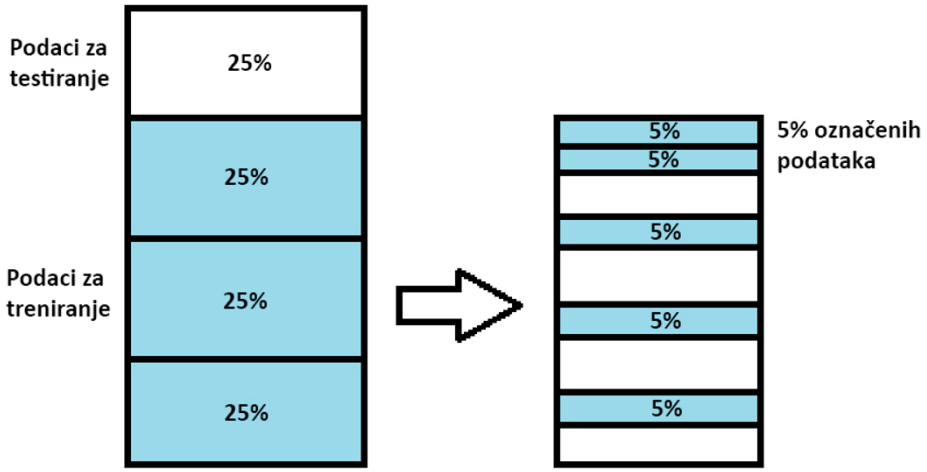
- osnovni_procijenitelj (eng. base_estimator) - Objekt procjenitelja koji implementira fit i predict_proba metode. Pozivanje metode fit će prilagoditi klon proslijeđenog procjenitelja, koji će biti pohranjen u atributu base_estimator.
- prag (eng. treshold) - Prag odluke za korištenje s criterion='threshold'. Vrijednost mu treba biti između 0 i 1. Kada se koristi kriterij praga potrebno je koristiti dobro kalibriran klasifikator.
- kriterij (eng. criterion), zadano='prag' – Odlučuje koji će biti kriterij odabira koji se koristi za odabir oznaka koje želimo dodati skupu za trening. Ako je kriterij='prag', skupu

podataka dodaju se pseudo-oznake s vjerojatnostima predviđanja iznad vrijednosti praga. Ako je kriterij='k_best', k_best pseudo-oznaka s najvećim vjerojatnostima predviđanja dodaju se skupu podataka u svakoj iteraciji.

- max_iter, zadano=10 - Maksimalni dopušteni broj ponavljanja. Trebao bi biti veći od ili jednak 0. Ako je Ništa (eng. None), klasifikator će nastaviti predviđati oznake dok se ne dodaju nove pseudo-oznake ili dok se svi neoznačeni uzorci ne označi.
- verbose, zadano=False - Ispisuje podatke koji bi mogli biti zanimljivi prilikom svake iteracije kao što je primjerice koliko je oznaka predviđeno i dodano u svakoj iteraciji.

Kao što je prethodno spomenuto, metoda odabira novih pseudo-oznaka za trening može se prilagoditi prema našim preferencama. Odabir može biti najboljih k_best predviđanja ili postavljanjem definiranog praga vjerojatnosti. Za potrebe ovog diplomskog rada koristili smo prag vjerojatnosti od 0.75. To znači da će svaka podatkovna točka koja prikazuje vjerojatnost klase jednaku ili veću od 0,75 biti uključena u zbirku pseudo-označenih podataka. Te dodane podatkovne točke će zatim pridonijeti prilikom treniranja modela u sljedećoj iteraciji. Nakon što smo instancirali klasifikator, korištenjem fit() metode dodjeljujemo mu potrebne skupove podataka za treniranje, te se obavlja stvarno treniranje modela. Prilikom treniranja modela mjerimo vrijeme trajanja treniranja modela, potrošnju RAM-a i CPU-a koje pri završetku mjerenja zapisujemo u csv datoteku. Završetkom razvoja polu-nadziranog modela slijedi testiranje, gdje izračunavamo različite metrike vrednovanja kao što su geometrijska sredina, točnost, F1-mjeru i ROC-AUC te pohranjujemo ih u pandas podatkovne okvire za daljnju obradu.

Jedna od najbitnijih tehnika vrednovanja koja je korištena i implementirana je unakrsna provjera. Unakrsna provjera je ključna tehnika koja se koristi u strojnom učenju za procjenu izvedbe modela i njegovu sposobnost generalizacije na nove, neviđene podatke. Uključuje sustavno dijeljenje dostupnih podataka u podskupove za treniranje i testiranje valjanosti modela više puta, pružajući pouzdaniju, stabilniju i temeljitiju procjenu od korištenja podjele na skupove za treniranje i testiranje. Unakrsna provjera detaljnije je opisana u poglavlju 3.1. Za potrebe diplomskog rada korištena je k-struka unakrsna provjera gdje je iznos k=4, odnosno skup podataka podijeljen je na 4 jednaka dijela od kojih su 3 korištena za treniranje modela dok je testiranje odvijano na preostalom dijelu. Na slici 5.1. prikazana je unakrsna provjera korištena u ovom diplomskom radu.

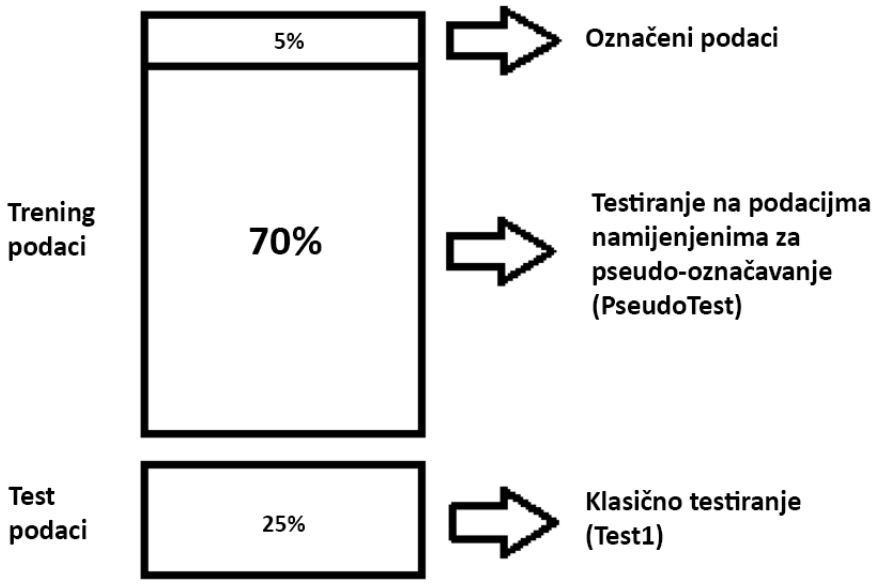


Slika 5.1. *Unakrsna provjera korištena u diplomskom radu*

Za unakrsnu provjeru nadziranog modela postoji metoda iz sklearn knjižnice koja se naziva cross_val_score(), dok za polu-nadzirano učenje ta ista metoda ne funkcionira. Kako scikit-learnt, koja je najpopularnija knjižnica za strojno učenje u Pythonu, ne pruža ugrađenu podršku za polu-nadzirano učenje ili polu-nadziranu unakrsnu provjeru odlučili smo se implementirati našu unakrsnu provjeru za model polu-nadziranog učenja. Potrebno je imati na umu da primjena unakrsne provjere na polu-nadzirano učenje može biti složenije od standardnog nadziranog učenja jer moramo osigurati da pseudo-oznake generirane tijekom treninga ne propuštaju informacije iz validacije ili testnih preklopa u proces treniranja. Bitna je pažljiva implementacija. Kako koristimo iste inicijalno označene podatke za treniranje nadziranog i polu-nadziranog modela odlučili smo ne koristiti cross_val_score() metodu za nadzirano učenje, već i za nadzirano učenje implementirati unakrsnu provjeru. Prilikom implementacije unakrsne provjere za polu-nadzirano učenje bilo je potrebno izmijeniti pristup. Potrebno je odabrati strategiju pseudo-označavanja gdje je izbor pao na metodu samo-učenja. Bilo je potrebno prilagoditi tradicionalni postupak unakrsne provjere kako bi uključili naš polu-nadzirani pristup učenju. Taj prilagođeni pristup uključuje ponavljanje (iteriranje) preko svakog preklopa, uvježbavanje našeg modela na označenim podacima unutar svakog preklopa, primjenu strategije pseudo-označavanja na neoznačene podatke u preklopu i procjenu izvedbe modela. Za svaki preklop ponavljan je proces pseudo-označavanja i treniranja modela te su rezultati unakrsne provjere pohranjeni u pandas podatkovne okvire kako bi procijenili izvedbu našeg polu-nadziranog pristupa koristeći odgovarajuće metrike. Programski kod unakrsne provjere za potrebe ovog diplomskog rada prikazan je u nastavku.

```
i = 0
for i in range(4):
    df_test = df_cv_split[i]
    df_train_default = pd.DataFrame()
    j=0
    for j in range(4):
        if j != i:
            df_train_default = df_train_default.append(df_cv_split[j],
ignore_index=True)

    df_train = pd.DataFrame()
    k=0

    for k in range(5):

        '''
        Ovdje se odvija sva raspodjela podataka (trening, test),
        određuje % označenih podataka, treniranje nadziranog modela,
        pseudo-označavanje metodom samo-učenja,
        treniranje polu-nadziranog modela
        '''

        df_train = pd.DataFrame(None)
```

Testiranje modela predviđanja odvijano je na dva skupa podataka. Osim klasičnog testiranja u unakrsnoj provjeri gdje se testiranje odvija na preostalom preklopu u svrhu istraživanja modele nadziranog i polu-nadziranog učenja testirali smo i na podacima korištenima za pseudo-označavanje. Klasično testiranje nazvano je "Test1" dok je testiranje na podacima korištenima za pseudo-označavanje nazvano "PresudoTest". Skica testiranja prikazana je u nastavku na slici 5.2.



Slika 5.2. *Skica testiranja modela predviđanja.*

5.2 Rezultati istraživanja

Cilj projekta je izraditi model polu-nadziranog učenja, usporediti ga s modelom standardnog nadziranog učenja kada je dostupna mala količina označenih podataka. Osim izrade modela potrebno je vrednovati odabrane modele te izmjeriti utrošak energije, memorije i vremena za scenarije nadziranog i polu-nadziranog učenja. Prilikom mjerenja performansi i potrošnje resursa modela, izvedena su 3 mjerenja za odabrane klasifikatore gdje je povećavan postotak označenih podataka (5%, 10% i 15%). Prilikom mjerenja performansi i učinkovitosti modela nadziranog i polu-nadziranog učenja korišteni su isti klasifikatori. Završetkom mjerenja izračunati su prosječni rezultati. Izračunate su metrike točnost, geometrijska sredina i F1-mjera za oba dva načina testiranja. Svrha vrednovanja korištenja različitih klasifikatora je njihova usporedba na temelju njihove upotrebe CPU-a i RAM-a te točnosti na zajedničkom skupu podataka gdje je cilj ove usporedbe pronaći ravnotežu između računalne učinkovitosti i moći predviđanja.

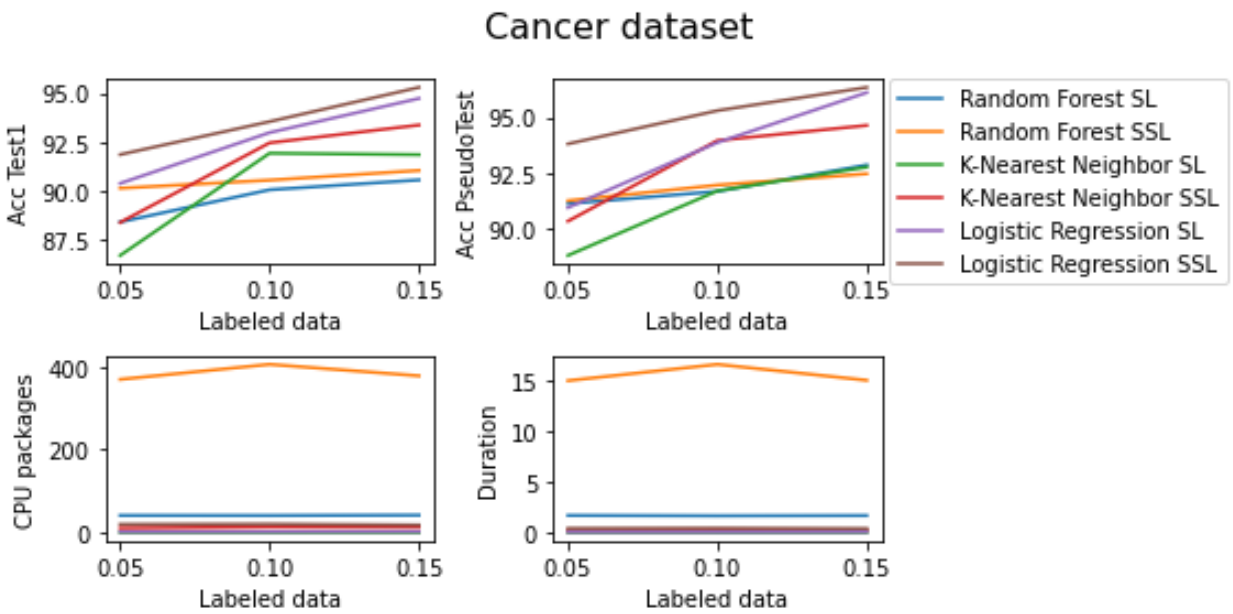
U ovom poglavlju slijedi opis rezultata na korištenim skupovima podataka, gdje možemo vidjeti koji model učenja je bolji, učinkovitost korištenja pojedinog učenja i koji klasifikator je bio bolji na pojedinom skupu podataka. Rezultati su opisani u poglavljima gdje svaki skup podataka predstavlja novo poglavlje. Skupovi podataka prikazani su u tablici 5.1.

5.2.1 Skup podataka Tumor

Skup podataka "Tumor" prikazuje je li uzorak maligan ili benigan. Značajke su izračunate iz digitalizirane slike aspirata fine igle mase dojke. One opisuju karakteristike staničnih jezgri

prisutnih na slici. Skup podataka "Tumor" je skup s najmanje instanci koji je korišten u ovom diplomskom radu. Na slici 5.3 prikazani su rezultati različitih klasifikatora strojnog učenja na temelju upotrebe CPU-a, vremenu trajanja treniranja modela i točnosti na istom skupu podataka. Iz prikazanih rezultata možemo vidjeti računalne učinkovitosti i izvedbe predviđanja klasifikatora. Algoritmi koji su korišteni kao klasifikatori za stvaranje modela nadziranog i polu-nadziranog učenja su:

- Slučajne šume
- Metoda najbližeg susjeda
- Logistička regresija

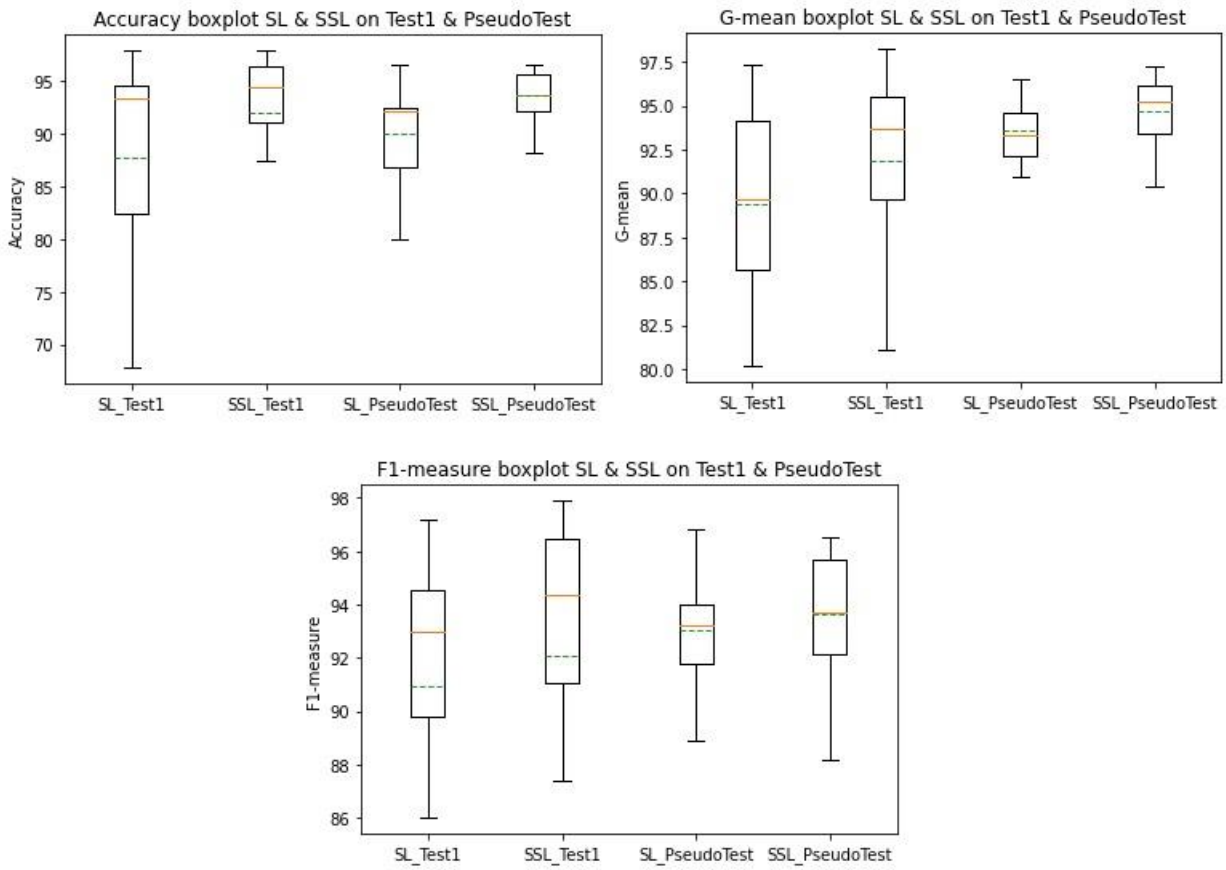


Slika 5.3. *Prikaz računalne učinkovitosti i izvedbe predviđanja klasifikatora na skupu podataka "Tumor"*

Iz rezultata prikazanih na slici 5.3. možemo vidjeti da se kao najbolji klasifikator pokazao klasifikator logističke regresije polu-nadziranog modela na oba dva testa (Test1 i PseudoTest). Klasifikator logističke regresije pruža najbolji omjer performansa i učinkovitosti. Ako usporedimo grafove točnosti oba dva testiranja, logistička regresija prikazuje nešto bolje rezultate na podacima korištenima za pseudo-označavanje tj. PseudoTestu, iako se klasifikator logističke regresije nadziranog učenja približava povećanjem postotka označenih podataka. Točnost predviđanja klasifikatora logističke regresije povećava se povećanjem postotka označenih podataka, možemo reći skoro pa linearno. Algoritam slučajnih šuma se na ovom skupu podataka pokazao kao najlošiji

odabir dok se metoda najbližeg susjeda pokazala kao drugi najbolji izbor. Kod metode najbližeg susjeda možemo vidjeti značajan skok u performansama kada se povećao postotak označenih podataka s 5% na 10%. Što se tiče učinkovitosti po potrošnji CPU-a i RAM-a te vremenu trajanja treniranja modela, najgore rezultate ima klasifikator slučajnih šuma polu-nadziranog učenja. Još jedna od bitnih stvari koje možemo primijetiti je da su kod svih algoritama gledajući metriku točnosti polu-nadzirane verzije algoritma bolje u odnosnu na standardno nadzirano učenje. U prosjeku su polu-nadzirani modeli 1,95% bolji u odnosu na nadzirane modele.

Upotreba RAM-a nije prikazana na grafu (iako je mjerena) jer prati potrošnju CPU-a, odnosno klasifikator koji prikazuje najveću potrošnju CPU-a ima ujedno i najveću potrošnju RAM-a i obrnuto, u tom slučaju je to algoritam slučajnih šuma.



Slika 5.4. *Prikaz točnosti, g-sredine, F1-mjere skupa podataka "Tumor" na kutijastom dijagramu*

Na slici 5.4. prikazane su metrike vrednovanja točnost, g-sredine, F1-mjere pomoću kutijastog dijagrama na kojem zelena iscrtkana linija predstavlja srednju vrijednost dok narančasta linija predstavlja medijan.

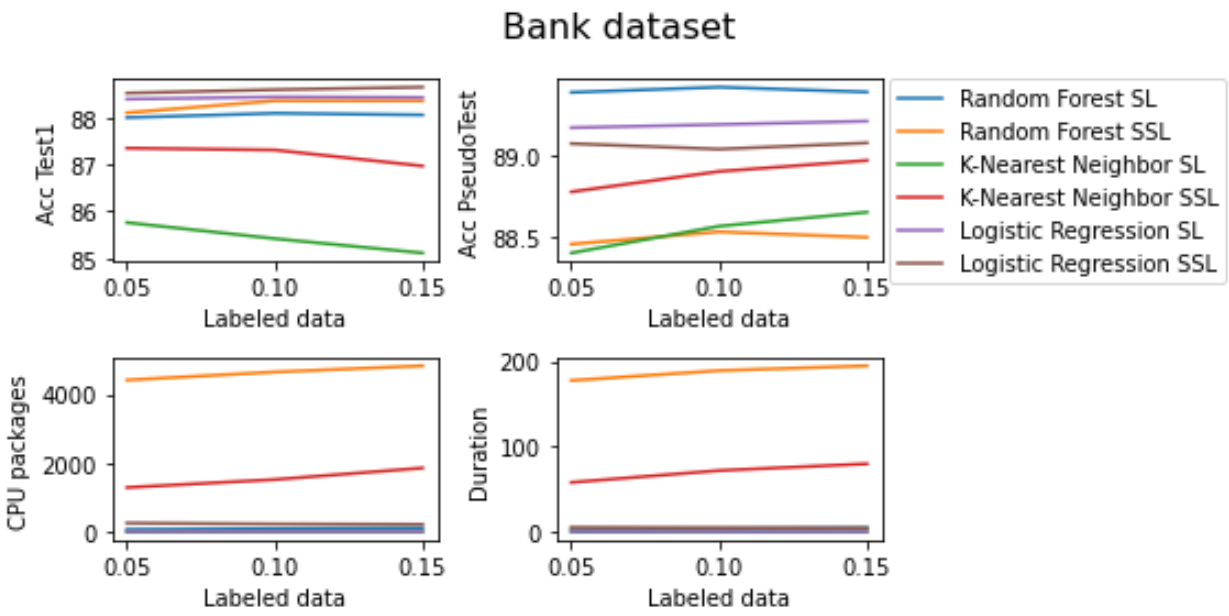
Geometrijska sredina često se koristi u kontekstu neuravnoteženih skupova podataka kako bi se pružio uravnoteženiji prikaz izvedbe klasifikatora u različitim klasama. Skup podataka "Tumor" sadrži 62.7% negativnih i 37.3% pozitivnih instanci pa možemo smatrati da je skup podataka djelomično neuravnotežen. Ako pogledamo visoke vrijednosti g-sredine na kutijastom dijagramu koje su modeli nadziranog i polu-nadziranog učenja ostvarili možemo zaključiti da klasifikator dobro predviđa 0 i 1. Rezultati testiranja nadziranog i polu-nadziranog modela na PseutoTestu prikazuje stisnutije kutije u usporedbi sa Testom1. To nam govori da klasifikator manje griješi. Iako je točnost polu-nadziranog modela testiranog na Test1 veća u odnosu na nadzirani model na PseudoTestu ako promatramo g-sredinu nadzirani model je pouzdaniji.

Kutije na kutijastom dijagramu koji prikazuje F1-mjeru nalaze u rasponu od 90% do 96%. F1-mjera u rasponu od 90% do 96% za model klasifikacije općenito je indikacija modela s dobrom izvedbom. F1-mjera je metrika koja uravnotežuje kompromis između preciznosti i odziva, pružajući sveobuhvatnu procjenu sposobnosti modela da točno klasificira instance, posebno u slučajevima kada bi klase mogle biti neuravnotežene. Takve vrijednosti F1-mjere pokazuju da model klasifikacije postiže visoku razinu točnosti i učinkovitosti u izradi predviđanja. To ukazuje na to da model vrši točne klasifikacije dok održava dobru ravnotežu između preciznosti i odziva.

Iz sva 3 kutijasta dijagrama možemo vidjeti da je polu-nadzirano učenje ostvarilo bolje rezultate.

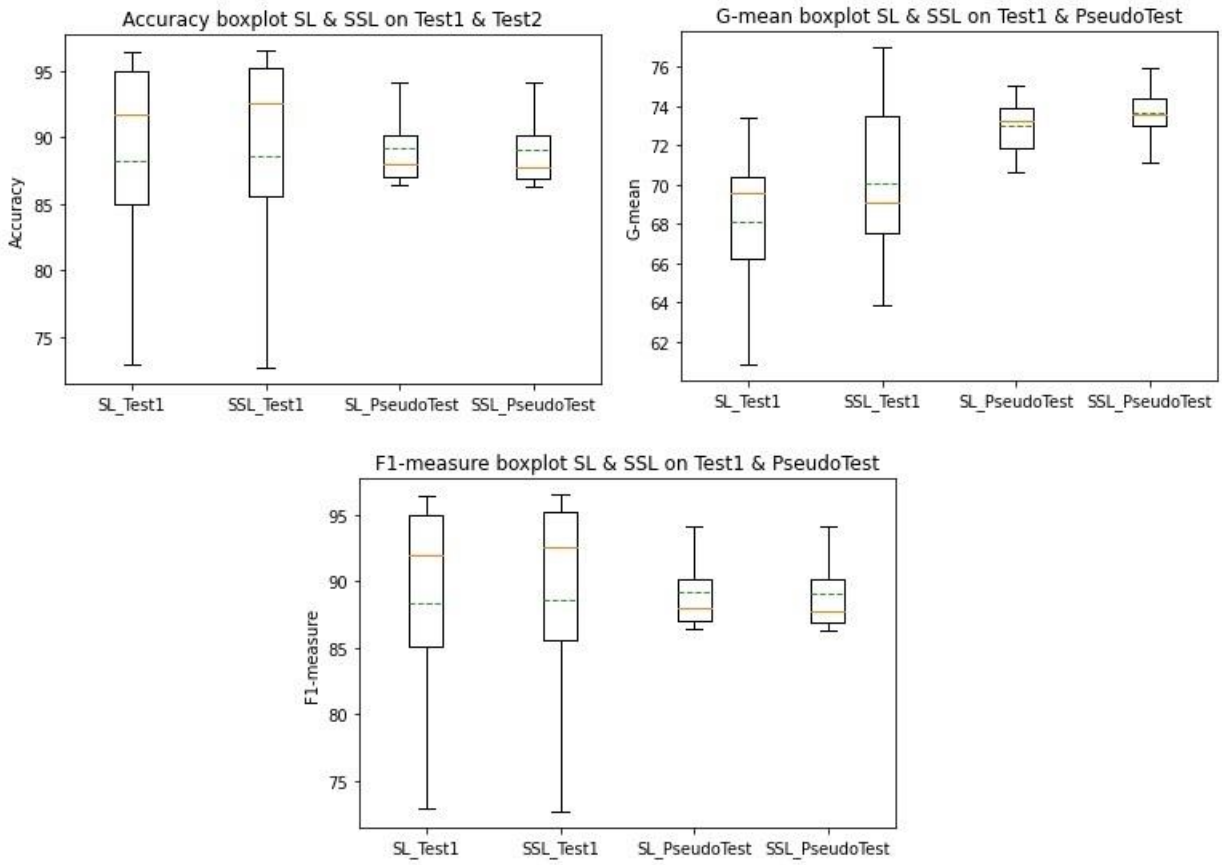
5.2.2 Skup podataka Banka

Skup podataka "Banka" je najveći skup podataka korišten u ovom diplomskom radu. Sastoji se od 45211 instanci i 17 značajki. Podaci se odnose na izravne marketinške kampanje jedne portugalske bankarske institucije. Marketinške kampanje temeljenje su na telefonskim pozivima gdje je često bilo potrebno više od jednog poziva s istim klijentom kako bi odlučili hoće li proizvod (bankovno oročenje) biti ili ne pretplaćen. Na slici 5.5 prikazani su rezultati klasifikatora na temelju upotrebe CPU-a, vremenu trajanja treniranja modela i točnosti na istom skupu podataka. Iz prikazanih rezultata možemo vidjeti učinkovitost i klasifikatorske sposobnosti predviđanja.



Slika 5.5. *Prikaz računalne učinkovitosti i izvedbe predviđanja klasifikatora na skupu podataka "Banka"*

Iz rezultata istraživanja uzimajući u obzir metrike točnosti, g-sredinu, F1-mjeru i potrošnju resursa kao najbolji i najučinkovitiji klasifikator na ovom skupu podataka se pokazao algoritam logističke regresije. Prilikom testiranja na Testu1 gledajući točnost, najbolje rezultate ostvario je polu-nadzirani model koristeći klasifikator logističke regresije, dok je na PseudoTestu najbolje rezultate ostvario nadzirani model s klasifikatorom slučajne šume. Kako je "Banka" najveći skup podataka korišten za potrebe diplomskog rada s preko 45 000 instanci treniranje modela polu-nadziranog učenja trajalo je najduže u usporedbi s preostalim skupovima podataka. Pošto je logistička regresija poznata kao brzi klasifikator, obje verzije modela istrenirana su prilično brzo. Stvaranje polu-nadziranih modela koristeći klasifikatore slučajnih šuma i najbližih susjeda trajalo je znatno duže uz veću potrošnju CPU-a, RAM-a pa ih zbog toga ne smatramo učinkovitima na ovom skupu podataka. Klasifikator slučajnih šuma potrošio je 4410 J CPU energije, dok je klasifikator najbližih susjeda potrošio 1285 J što je puno u usporedbi s potrošnjom koju je ostvarila logistička regresija, a iznosi 251 J. Povećanjem postotka označenih podataka performanse predviđanja modela ne povećavaju se signifikantno, ali povećava se potrošnja resursa prilikom treniranja modela.



Slika 5.6. *Prikaz točnosti, g-sredine, F1-mjere skupa podataka "Banka" na kutijastom dijagramu*

Na slici 5.6. prikazane su metrike vrednovanja točnost, g-sredina i F1-mjera pomoću kutijastog dijagrama. Kutijasti dijagram je grafički prikaz koji se često koristi u statistici i vizualizaciji podataka za prikaz distribucije skupa podataka. U kontekstu strojnog učenja, kutijasti dijagrami se mogu koristiti za dobivanje uvida u distribuciju značajki, identificiranje stršećih vrijednosti (eng. outliers) i usporedbu distribucija između različitih kategorija ili grupa.

U skupu podataka "Banka" korištenom za vrednovanje modela, postoji značajna neravnoteža u klasama gdje većinskoj klasi (Klasa 0) pripada 88,3% uzoraka, dok manjinskoj klasi (Klasa 1) pripada samo 11,7%. Ova neravnoteža može utjecati na vrednovanje modela i treba je uzeti u obzir pri tumačenju rezultata. Zbog tog razloga prilikom vrednovanja modela ne gledamo samo metriku točnosti koja nije prikladna u tom slučaju budući da model koji predviđa većinsku klasu može postići visoku točnost. Mjerni podaci kao što su geometrijska sredina i F1-mjera informativniji su u procjeni izvedbe modela.

Prvi kutijasti dijagram ilustrira distribuciju vrijednosti točnosti postignutih našim modelima kroz više procjena. Većina vrijednosti kreću se od 85% do 95%, što ukazuje na relativno dosljednu i

povoljnu izvedbu iako ima nekoliko vrijednosti koje odskaču. Interkvartilni raspon (IQR) je relativno uzak, što sugerira da je točnost modela stabilna i da nije sklona značajnim oscilacijama. Vrijednosti točnosti na Testu1 obje metode učenja kreću se od 72% do 97% što ukazuje na široki spektar rezultata izvedbe. Prilikom testiranja na PseudoTestu odstupanja su rijetka, što ukazuje na to da model općenito radi dosljedno dobro i da ne pokazuje ekstremna odstupanja u točnosti. Srednja vrijednost točnosti, koja se nalazi unutar ovog IQR-a, služi kao pouzdana mjera središnje tendencije. Srednja vrijednost obje metode učenja je skoro pa jednaka. Na Testu1 postiže vrijednost 88% dok na PseudoTestu 89%. Općenito, kutijasti dijagram odražava robusnu izvedbu u smislu točnosti, pri čemu većina procjena rezultira točnostima iznad 85%.

Drugi kutijasti dijagram prikazuje distribuciju geometrijskih srednjih vrijednosti izmjerene prilikom unakrsne provjere. Većina vrijednosti se kreće od 66% do 74%, što ukazuje na nešto uži raspon performansi u usporedbi s točnošću i F1-mjerom što nam sugerira relativno stabilnu izvedbu. Čini se da je IQR i ovdje relativno uzak, što implicira dosljedno ponašanje modela u smislu g-sredine. Iako je prisutno nekoliko ekstremnih vrijednosti, one nisu pretjerano udaljene od većine podatkovnih točaka, što ukazuje da izvedba modela ostaje relativno stabilna i pouzdana. Srednja vrijednost g-sredine nalazi se unutar središnjeg raspona podataka, što ukazuje da predstavlja tipičnu izvedbu modela.

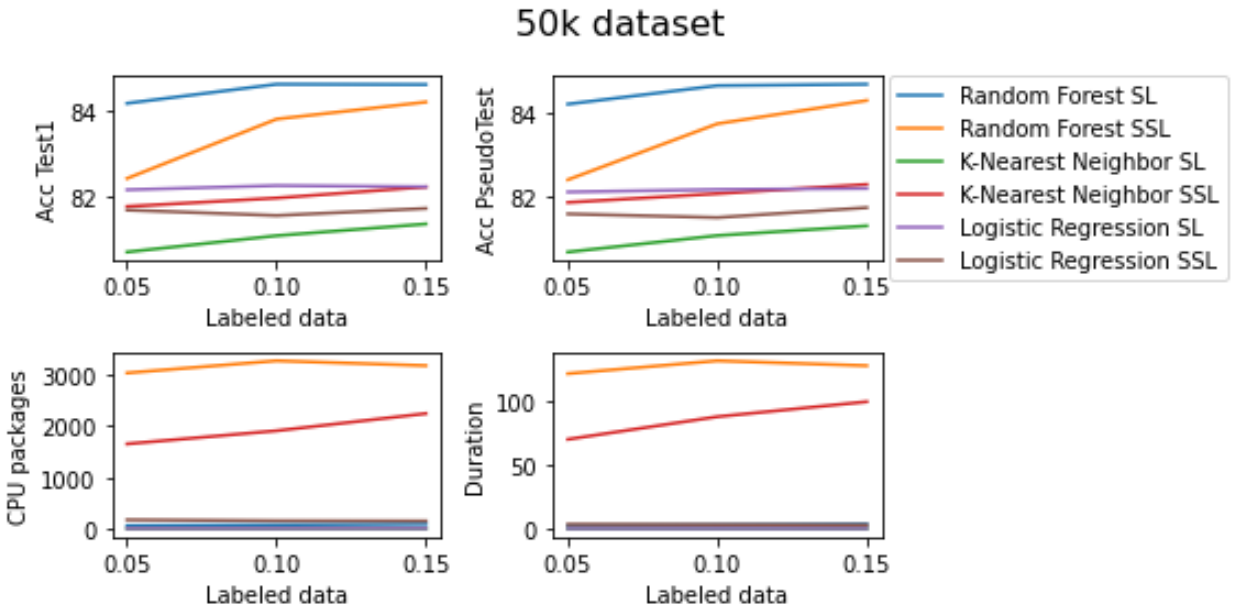
Rezultati metrike F1-mjere su gotovo identični vrijednostima točnosti. Zanimljivo je da je IQR koncentriran unutar jednakog raspona kao kod točnosti. Ovo usklađivanje s točnosti IQR označava dosljednu korespondenciju između dvije metrike. Srednja vrijednost F1-mjere, koja se podudara s točnošću, naglašava uravnoteženu preciznost i odziv modela.

Nakon usporedbe kutijastih dijagrama između dva testa, vidljivo je da modeli ostvaruju bolje rezultate prilikom testiranja na PseudoTestu u odnosu na Test1. To je vidljivo jer su kutije PseudoTesta primjetno kraće u usporedbi s kutijama Testa1, odnosno IQR je znatno uži. Uži IQR govori nam da je točnost modela na PseudoTestu stabilna i da nije sklona značajnim oscilacijama. Sveukupno, ovi okvirni dijagrami zajedno ukazuju na model s dobrom izvedbom s dosljednim i pouzdanim rezultatima procjene.

5.2.3 Skup podataka 50k

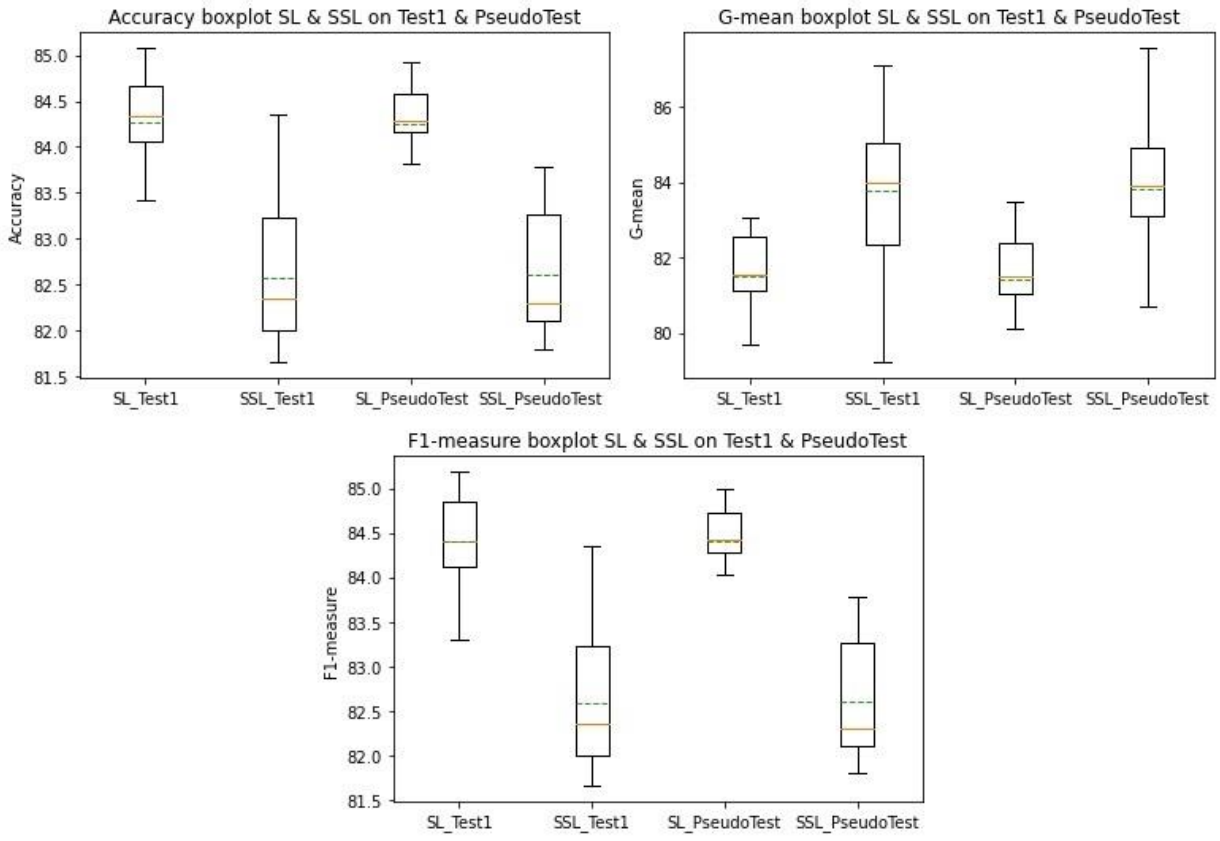
Skup podataka "50k" je skup podataka u kojem je cilj predvidjeti hoće li prihod osobe premašiti 50 tisuća dolara godišnje na temelju podataka iz popisa stanovništva. "50k" je nešto manji skup podataka u odnosu na "Banku". Sastoji se od 32561 instanci, gdje 75.92% uzoraka pripada klasi 0, a 24.08% klasi 1. Na slici 5.7 prikazani su rezultati različitih klasifikatora na temelju upotrebe

CPU-a, vremenu trajanja treniranja modela i točnosti. Iz prikazanih rezultata možemo vidjeti učinkovitost i sposobnosti predviđanja klasifikatora.



Slika 5.7. *Prikaz računalne učinkovitosti i izvedbe predviđanja klasifikatora na skupu podataka "50k"*

Iz rezultata prikazanih na slici 5.7. možemo vidjeti da se kao najbolji klasifikator za razvijanje modela nadziranog i polu-nadziranog učenja pokazao algoritam slučajnih šuma. Klasifikator slučajnih šuma nadmašio je klasifikatore najbližih susjeda i logističku regresiju, pokazujući svoje performanse gledajući točnost, učinkovitost CPU-a, RAM-a i vremenu trajanja treniranja modela. Te performanse proizašle su iz njegove sposobnosti da učinkovito uhvati zamršene odnose podataka i granice odlučivanja. Algoritam slučajnih šuma kombinirao je višestruka stabla odlučivanja, učinkovito rješavajući pitanja pretreniranosti i poboljšavajući svoju sposobnost generalizacije na nove, neviđene podatke, što je rezultiralo povećanom preciznošću predviđanja modela. Nadzirano učenje slučajnih šuma ostvarilo je bolje rezultate točnosti i učinkovitosti u usporedbi s polu-nadziranim modelom. Obje verzije modela slučajnih šuma odskaču od preostalih algoritama gledajući točnost. Polu-nadzirani modeli metode najbližih susjeda i slučajnih šuma ostvarili su najlošije rezultate potrošnje resursa prilikom treniranja modela. Rezultati testiranja modela na oba dva testa (Test1 i PseudoTest) su skoro pa jednaka tj. razlika je neznatna.



Slika 5.8. *Prikaz točnosti, g-sredine, F1-mjere skupa podataka "50k" na kutijastom dijagramu*

Na slici 5.8. predstavljena su tri kutijasta dijagrama koji ilustriraju metriku točnosti, g-sredine i F1-mjeru za nadzirani i polu-nadzirani model. Svaki kutijasti dijagram pruža uvid u distribuciju rezultata postignutih primjenom algoritma slučajne šume nakon unakrsne provjere gdje skup podataka za treniranje modela sadrži 5% označenih podataka. Pošto su rezultati testiranja modela na Testu1 i PseudoTestu skoro jednaki, prilikom komentiranja rezultata fokusirat ćemo se samo na rezultate Testa1.

Promatrajući i spoređujući dvije kutije (nadziranog i polu-nadziranog učenja) na kutijastom dijagramu točnosti, primjećujemo da model nadziranog učenja općenito ima više vrijednosti točnosti i uži IQR od polu-nadziranog modela. To implicira da izvedba nadziranog modela nije samo konzistentnija, nego također postiže više razine točnosti u prosjeku. S druge strane, polu-nadzirani model pokazuje širu distribuciju vrijednosti točnosti, potencijalno ukazujući na veću varijabilnost ili osjetljivost na uvjete koji su mu predstavljeni.

Prelazeći na izvedbu geometrijske sredine, okvirni dijagram otkriva intrigantne trendove. Za nadzirani model, geometrijske srednje vrijednosti kreću se od 79,8% do 83%, a IQR se kreće od

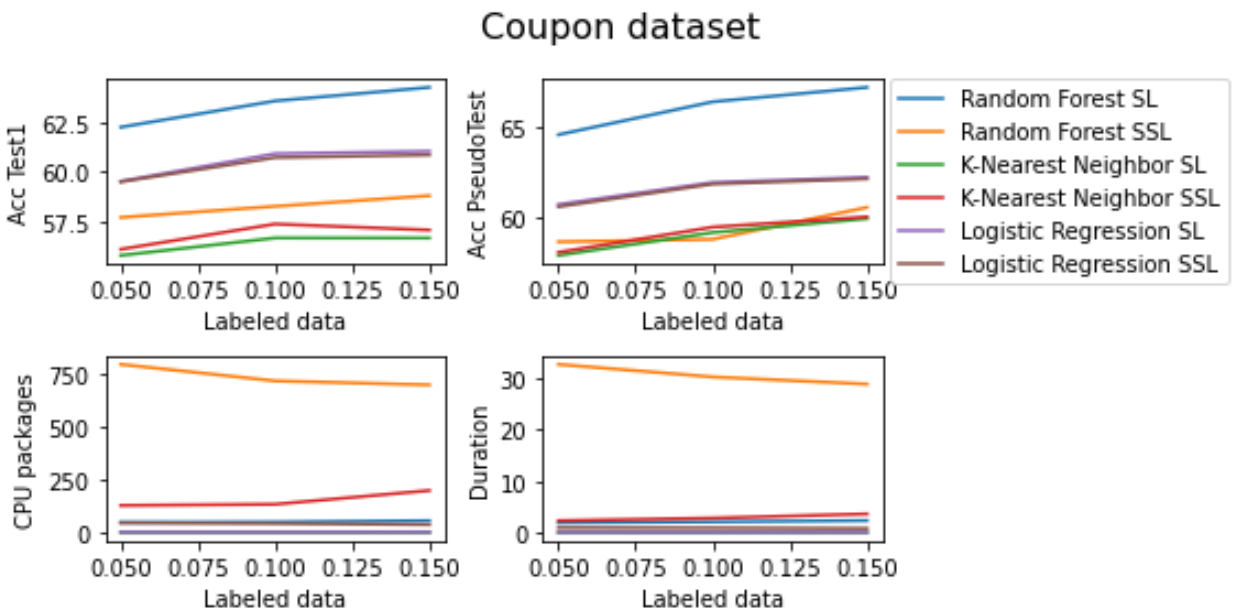
81,2% do 82,5%. Ovo ukazuje na dosljednu izvedbu unutar relativno uskog raspona. Suprotno tome, polu-nadzirani model prikazuje širu distribuciju, s geometrijskim srednjim vrijednostima u rasponu od 79,2% do 87%. IQR za polu-nadzirani model je širi, u rasponu od 82,5% do 85,2%. Ovi rezultati naglašavaju potencijalne kompromise između modela, pri čemu polu-nadzirani pristup pokazuje i više visoke i potencijalno niže niske rezultate geometrijske sredine u usporedbi s nadziranim modelom.

Na kraju, kutijasti dijagram F1-mjere pruža daljnje uvide. Rezultati F1-mjere ukazuje na dosljednu i usku distribuciju vrijednosti F1-mjere. Kao što je uočeno u drugim metrikama, F1-mjera polu-nadziranog modela također pokazuje širu distribuciju u usporedbi s nadziranim modelom.

Modeli predviđanja oboje učenja daju prilično jednake rezultate na oba dva testa. Jedina razlika koju možemo primijetiti je da su kutije modela testiranih na PseudoTestu uže, što nam ipak ukazuje da su modeli stabilniji i nisu skloni značajnim oscilacijama prilikom testiranja na podacima korištenim za pseudo-označavanje.

5.2.4 Skup podataka Kupon

Skup podataka "Kupon" prikupljen je putem ankete za Amazon. Anketa opisuje različite scenarije vožnje uključujući odredište, vremensku prognozu, trenutno vrijeme, putnika u automobilu itd., a zatim pita osobu hoće li prihvatiti kupon ako je vozač. Sastoji se od 12684 instanci od kojih 43.16% uzoraka pripada klasi 0, a 56.84% klasi 1. Na slici 5.9 prikazani su rezultati korištenih klasifikatora za izradu modela na temelju upotrebe CPU-a, vremenu trajanja treniranja modela i točnosti. Iz prikazanih rezultata možemo vidjeti učinkovitost i klasifikatorske izvedbe predviđanja.

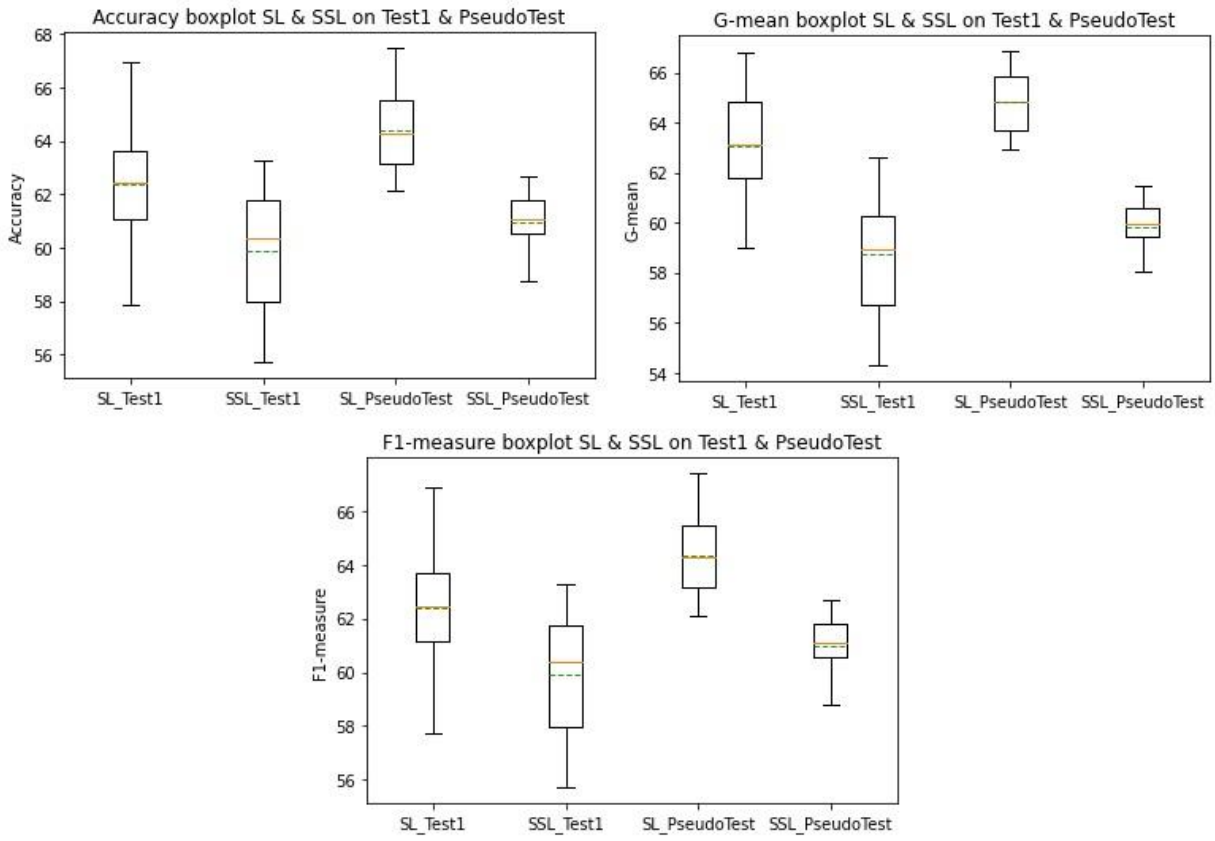


Slika 5.9. *Prikaz računalne učinkovitosti i izvedbe predviđanja klasifikatora na skupu podataka "Kupon"*

Analizirajući rezultate iz četiri grafikona, možemo izvući nekoliko zanimljivih zaključaka. Počevši od vrijednosti točnosti prikazanih na prvom i drugom grafikonu, vidljivo je da nadzirani model koji koristi klasifikator slučajnih šuma postiže najveću točnost. Ovo pokazuje robusnost algoritma slučajnih šuma u hvatanju složenih odnosa unutar podataka. Međutim, intrigantno je primijetiti da i nadzirani i polu-nadzirani modeli izgrađeni na klasifikatoru logističke regresije također imaju izvanredne rezultate, pokazujući učinkovitost uključivanja neoznačenih podataka u proces učenja.

Promatrajući treći grafikon koji ilustrira korištenje CPU-a tijekom treninga, uočavamo jasan trend. Polu-nadzirani model slučajnih šuma koristi najviše resursa CPU-a kao i na prošlim skupovima podataka, što ukazuje na njegov računalni intenzitet zbog njegove prirode ansambla. Zanimljivo, druga najveća potrošnja CPU-a pripada polu-nadziranom modelu razvijanom koristeći metodu najbližeg susjeda. Ovo sugerira da dok se metoda najbližih susjeda općenito smatra laganim algoritmom, uključivanje neoznačenih podataka može uvesti određenu složenost tijekom obuke.

Grafikon koji prikazuje vrijeme potrebno za treniranje modela vrlo je sličan grafikonu korištenja CPU-a, potvrđujući naša ranija opažanja. Takvo usklađivanje između upotrebe CPU-a i vremena implicira da računalni zahtjevi modela izravno utječu na vrijeme potrebno za obuku. Važno je napomenuti da polu-nadzirani model slučajnih šuma, koji je imao najveću upotrebu CPU-a, također treba najdulje za obuku, naglašavajući kompromis između točnosti i učinkovitosti obuke.



Slika 5.10. *Prikaz točnosti, g-sredine, F1-mjere skupa podataka "Kupon" na kutijastom dijagramu*

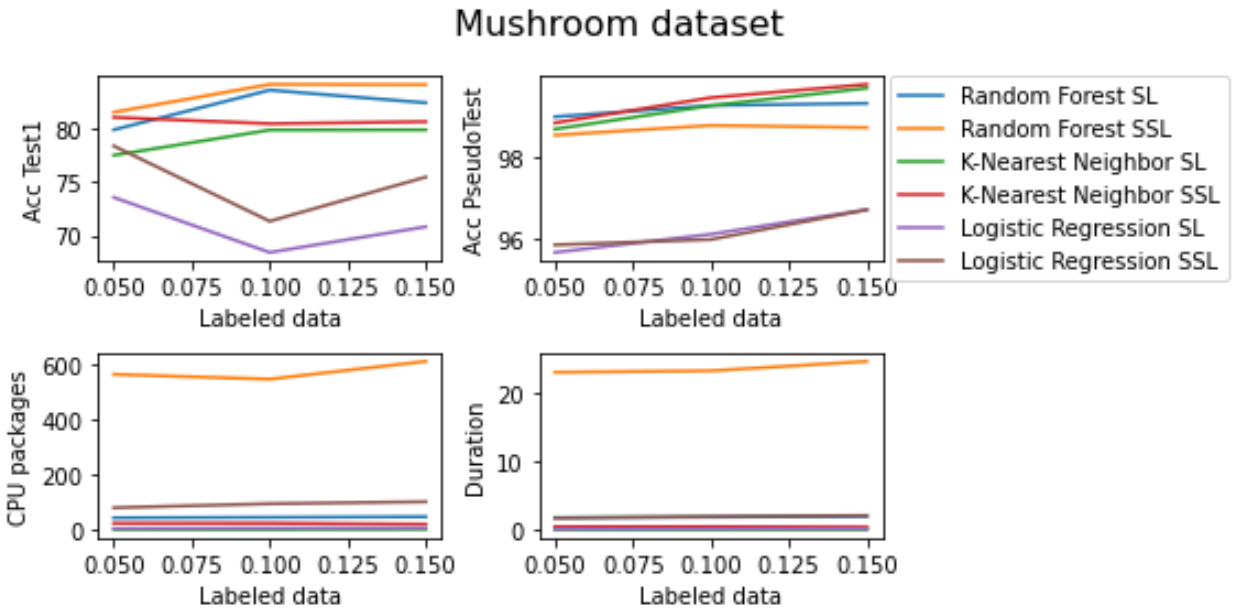
Promatrajući sve metrike prikazane na slici 5.10. možemo zaključiti da model nadziranog učenja ima bolje rezultate od polu-nadziranog modela. Raspon vrijednosti točnosti na PseudoTestu za oba dva modela je relativno uzak, što nam ukazuje da modeli pružaju stabilne performance te da nisu sklone značajnim oscilacijama.

Kod geometrijske sredine imamo sličnu situaciju kao i na kutijastom dijagramu koji prikazuje točnost. Nadzirani model ima tendenciju nadmašiti polu-nadzirani model u smislu geometrijske sredine. Iako nadzirani model ostvaruje veće vrijednosti od polu-nadziranog modela, kutija polu-nadziranog modela na PseudoTestu je uža u odnosu na nadzirani model.

Ako promatramo kutijasti dijagram F1-mjere, nadzirani model pokazuje bolje rezultate u usporedbi s polu-nadziranim modelom. Vrijednosti F1-mjere u skladu su s trendovima točnosti i geometrijske sredine. Iz navedenih podataka možemo vidjeti da nadzirani model ima bolje rezultate u sve tri metrike. Kako su razlike izvedbe između dva modela statistički značajne, to nam može biti jasan pokazatelj da je nadzirani pristup prikladniji za ovaj skup podataka.

5.2.5 Skup podataka Gljiva

Skup podataka uključuje opise hipotetskih uzoraka koji odgovaraju 23 vrste škrgastih gljiva iz obitelji Adaricus i Lepiota. Svaka vrsta je označena kao otrovna ili neotrovna. Sastoji se od 8124 uzoraka od kojih je svaki uzorak opisan sa 22 atributa. 51.8% podataka pripada klasi 0 dok 48.2% pripada klasi 1. Ovakav skup podataka predstavlja dobar primjer uravnoteženog skupa podataka. Na slici 5.11. prikazani su rezultati vrijednosti točnosti i potrošnje CPU-a i vremena trajanja prilikom razvoja modela nadziranog i polu-nadziranog modela.



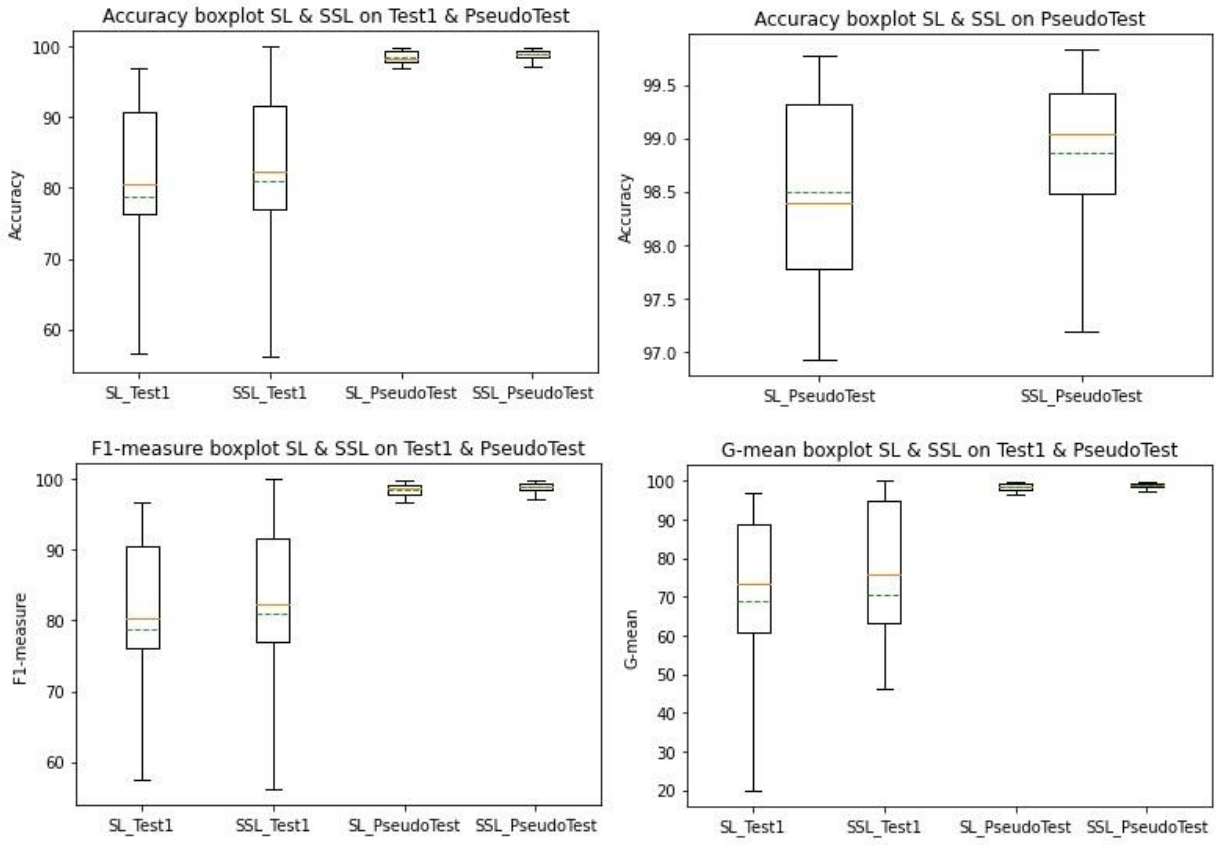
Slika 5.11. *Prikaz računalne učinkovitosti i izvedbe predviđanja klasifikatora na skupu podataka "Gljiva"*

Kada bi gledajući grafove točnosti morali zaključiti koji je klasifikator za stvaranje modela najbolji možda bi odabir pao na slučajnu šumu. Kako ne gledamo samo sposobnost predviđanja modela, već i učinkovitost tj. potrošnju resursa i vrijeme treniranja modela izbor pada na algoritam najbližeg susjeda gdje polu-nadzirana verzija modela ostvaruje najbolje rezultate na ovom skupu podataka.

Polu-nadzirani model kreiran klasifikatorom metode najbližeg susjeda postigao je najveću točnost među svim vrednovanim modelima. Iako se model kreiran algoritmom slučajne šume nalazi dosta blizu po rezultatima točnosti, ako pogledamo potrošnju procesora i vrijeme treniranja modela vidljivo je da modeli koji koriste metodu najbližeg susjeda kao klasifikator ostvaruju bolje rezultate. Polu-nadzirani model temeljen na algoritmu slučajne šume pokazao je najveću potrošnju

resursa. Rezultati potrošnje RAM-a nisu prikazani na grafu jer prate potrošnju CPU-a, odnosno klasifikator koji prikazuje najveću potrošnju CPU-a ima ujedno i najveću potrošnju RAM-a i obrnuto.

Vrijeme potrebno za treniranje modele, kao što je prikazano na grafikonu, prilično je slično grafu upotrebe CPU-a. Polu-nadzirani model temeljen na klasifikatoru slučajne šume trebao je najdulje za trening. S druge strane, polu-nadzirani model temeljen na metodi najbližeg susjeda imao je najkraće vrijeme treniranja, te je ujedno postigao najbolju točnost uz relativno nižu upotrebu CPU-a tijekom obuke. Stoga, njega smatramo najučinkovitijim modelom na ovom skupu podataka kada uzmemo u obzir performanse koje pruža za potrošene resurse.



Slika 5.12. *Prikaz točnosti, g-sredine, F1-mjere skupa podataka "Gljiva" na kutijastom dijagramu*

Kutijasti dijagram točnosti uspoređuje izvedbu nadziranog i polu-nadziranog modela, oba koristeći metodu najbližeg susjeda kao klasifikator. Ako uzmemo u obzir IQR kutija nadziranog i polu-nadziranog učenja na PseudoTestu, možemo primjetiti da je IQR polu-nadziranog modela nešto uži što nam sugerira da ostvaruje bolje rezultate točnosti te da nije sklon značajnim oscilacijama.

Osim IQR možemo uzeti u obzir i zelenu iscrtkana liniju koja prestavlja srednju vrijednost. Kako je srednja vrijednost polu-nadziranog modela viša od srednje vrijednosti nadziranog modela, to znači da polu-nadzirani model nadmašuje nadzirani model u točnosti.

Kod kutijastog dijagrama g-sredine također možemo vidjeti uži IQR kod polu-nadziranog modela, što nam sugerira da polu-nadzirani model postiže bolje rezultate geometrijske sredine u usporedbi sa nadziranim modelom.

U sve tri prikazane metrike polu-nadzirani model izgrađen na klasifikatoru najbližeg susjeda dosljedno pokazuje nešto bolju izvedbu u usporedbi s nadziranim modelom. Osim toga, polu-nadzirani model općenito pokazuje uži IQR što dokazuje na dosljedne rezultate u njegovoj izvedbi. Modeli ostvaruju bolje rezultate prilikom testiranja na PseudoTestu, gdje nema nekakvih ekstreminh odstupanja. Kako osim dobrih performansi model pruža i malu potrošnju CPU-a i RAM-a možemo reći da je ovaj model učinkovit za testirani skup podataka.

6. ZAKLJUČAK

Ideja diplomskog rada bila je izraditi model polu-nadziranog učenja, usporediti ga s modelom standardnog nadziranog učenja kada je dostupna mala količina označenih podataka. Osim izrade modela potrebno je vrednovati odabrane modele te izmjeriti utrošak energije, memorije i vremena potrebnog za treniranje modela za scenarije nadziranog i polu-nadziranog učenja. Za mjerenje utroška CPU-a, RAM-a i vremena treniranja modela korištena je pyRAPL knjižnica funkcija koja pruža mogućnost mjerenja samo određenog dijela programskog koda. U našem slučaju je to bio programski kod treniranja nadziranog i polu-nadziranog modela. Prilikom mjerenja performansi i potrošnje resursa modela, izvedena su 3 mjerenja za odabrane klasifikatore gdje je povećavan postotak označenih podataka (5%, 10% i 15%). Klasifikatori koji su korišteni za razvoj modela su; algoritam slučajne šume, metoda najbližeg susjeda i logistička regresija. Prilikom mjerenja performansi i učinkovitosti modela nadziranog i polu-nadziranog učenja korišteni su isti klasifikatori. Završetkom mjerenja izračunati su prosječni rezultati. Kako scikit-learnt, koja je najpopularnija knjižnica za strojno učenje u Pythonu, ne pruža ugrađenu podršku za unakrsnu provjeru na polu-nadziranom učenju odlučili smo se implementirati našu unakrsnu provjeru za model polu-nadziranog učenja. Potrebno je imati na umu da primjena unakrsne provjere na polu-nadzirano učenje može biti složenije od standardnog nadziranog učenja jer moramo osigurati da pseudo-oznake generirane tijekom treninga ne propuštaju informacije iz testnih preklopa u proces treniranja. Bilo je potrebno prilagoditi tradicionalni postupak unakrsne provjere kako bi uključili naš polu-nadzirani pristup učenju. Polu-nadzirani pristup korišten za stvaranje modela je bio metoda samo-učenja. Osim običnog testiranja gdje se testiranje odvija samo na skupu za testiranje u svrhu istraživanja, u ovom diplomskom radu odrađeno je i testiranje na podacima namijenjenima za pseudo-označavanje. Za oba dva načina testiranja izračunate su metrike točnost, geometrijska sredina i F1-mjera. Svrha vrednovanja korištenja različitih klasifikatora za razvijanje modela je njihova usporedba na temelju njihove upotrebe CPU-a i RAM-a te točnosti na zajedničkom skupu podataka gdje je cilj ove usporedbe pronaći ravnotežu između računalne učinkovitosti i moći predviđanja.

Promatrajući rezultate možemo vidjeti da nema neke prevelike razlike u performansama između modela nadziranog učenja i modela polu-nadziranog učenja. Na skupovima podataka korištenima za potrebe ovog diplomskog rada možemo vidjeti skupove podataka na kojima su bolje performanse ostvarivali modeli polu-nadziranog učenja, a to su "Tumor" i "Gljiva". Dok imamo i skupove podataka na kojima su modeli standardnog učenja nadjačali polu-nadzirane modele, a

to su skupovi podataka "Banka", "50k" i "Kupon". Najbolji rezultati obično su postignuti pri treniranju modela s 10% označenih podataka. Na 15% označenih podataka rast točnosti se usporio za većinu modela ili čak opao, što je slučaj kod skupa podataka "Banka" gdje je točnost polu-nadziranog modela pala za 1%. Skup podataka "Banka" je jedini skup podataka na kojem su modeli ostvarili najbolje performanse prilikom treniranja na 5% označenih podataka, uzevši u obzir efikasnost treniranja modela. Polu-nadzirani model logističke regresije na skupu podataka "Banka" bolji je 0,356% u odnosu na drugi najbolji polu-nadzirani model izgrađen na algoritmu slučajne šume. Iako je model logističke regresije neznatno bolji po točnosti, ostvaruje bolje rezultate potrošnje vremena i resursa prilikom treniranja modela. Potrošnja modela temeljenom na algoritmu slučajne šume ostvaruje potrošnju od 4625 J CPU-a i 186 s vremena treniranja modela, dok model logističke regresije ostvaruje prosječnu potrošnju 229 J CPU-a i 4,7 s vremena treniranja modela.

Na "Tumor" podacima možemo primijetiti da su svi polu-nadzirani modeli ostvarili bolje performanse predviđanja u usporedbi s nadziranim modelima, kada uspoređujemo modele razvijene na istim klasifikatorima. U prosjeku su polu-nadzirani modeli 1,95% bolji u odnosu na nadzirane modele.

Jedna od bitnijih stvari primijećenih prilikom rada na projektu je to da što su skupovi podataka bili klasno uravnoteženiji i sadržavali manje instanci, to su modeli polu-nadziranog učenja ostvarivali bolje rezultate. Na skupu podataka "Gljiva" kao najbolji model pokazao se polu-nadzirani model najbližeg susjeda koji ostvaruje 2% bolje performanse u odnosu na nadzirani model istog klasifikatora. Ostvaruje 8% veću potrošnju od polu-nadziranog modela logističke regresije ali 33 puta manju potrošnju od polu-nadziranog modela slučajne šume.

Na skupu podataka "Kupon" kao najbolji model pokazao se nadzirani model slučajne šume koji je postigao 8,6% bolje performanse od polu-nadziranog modela istog klasifikatora i 5% bolje performanse od modela logističke regresije. Model slučajne šume u nadziranoj verziji ostvario je najbolje performanse na skupu podataka "50k". Bolji je 1,2% te ostvaruje 230 puta manju potrošnju od polu-nadziranog modela slučajne šume.

Kroz analizu svih skupova podataka, isključujući skup podataka "Tumor", zaključujemo da nadzirani model slučajne šume nadmašuje druge najbolje modele u svim skupovima podataka s razlikom u rasponu od 0,3-5%. Model s najvećom potrošnjom i trajanjem treniranja modela na svim skupovima podataka pokazao se polu-nadzirani model algoritma slučajne šume koji je u

prosjeku ostvario 18 do 66 puta veću potrošnju i 13,4 do 68 puta duže treniranje modela u odnosu na nadzirani model slučajne šume.

Uzimajući u obzir sve rezultate, iako su postignuti dobri rezultati na pojedinim skupovima podataka, ne možemo sa sigurnošću tvrditi da je polu-nadzirani model bolji ili lošiji od modela nadziranog učenja. Analiza različitih skupova podataka i modela pokazuje da postoji optimalni postotak označenih podataka za postizanje najboljih performansi, to je u našem slučaju 10%. Iza ideje i istraživanja postoji potencijal gdje bi možda sljedeći korak bio isprobavanje na većem skupu podataka ili korištenje drugog klasifikatora. Klasifikator slučajne šume se pokazao kao dobar klasifikator, ali je sporiji za trening te zahtijeva dosta resursa.

Važno je istaknuti da rezultati istraživanja variraju ovisno o specifičnim uvjetima i postotku označenih podataka. Nema univerzalnog modela koji će uvijek biti optimalan, stoga je važno uzeti u obzir konkretnu situaciju prilikom odabira modela. Iako polu-nadzirani modeli možda nisu uvijek nadmašivali nadzirane modele u smislu performansi, važno je napomenuti da imaju svoju ulogu i primjenu, posebno u situacijama gdje je prikupljanje označenih podataka teško ili skupo.

LITERATURA

[1] E. Alpaydin: "Introduction to Machine learning third edition", London, England: The MIT Press, 2014.

[2] E. Burns. TectTarget: "Machine learning", s interneta, https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML, 15, srpanja 2022.

[3] Elements of AI: "Vrste strojnog učenja", s interneta, https://course.elementsofai.com/hr/4/1, 15. srpnja 2022

[4] IBM: "Supervised learning", s interneta, https://www.ibm.com/cloud/learn/supervised-learning, 3. kolovoza 2022

[5] Intellspot: "Supervised vs Unsupervised Learning: Algorithms and Examples", s interneta, https://www.intellspot.com/unsupervised-vs-supervised-learning/, 18. srpnja 2022

[6] Phytonista Planet: "Pros and Cons of Supervised Machine Learning", s interneta, https://pythonistaplanet.com/pros-and-cons-of-supervised-machine-learning/, 18. srpnja 2022

[7] edureka!: "How To Implement Calssification In Machine Learning?", s interneta, https://www.edureka.co/blog/classification-in-machine-learning/#classification, 24. srpnja 2022

[8] Seldon: "Machine Learning Regression Explained", s interneta, https://www.seldon.io/machine-learning-regression-explained, 24. srpnja 2022

[9] D. Soni, Towardsdatascience: "Supervised vs. Unsupervised Learning", s interneta, https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d, 20. Srpnja 2022

[10] Simplilearn: "Supervised Machine Learning - All You Need to Know", s interneta, https://www.simplilearn.com/tutorials/machine-learning-tutorial/supervised-machine-learning, 3. kolovoza 2022

[11] Xiaojin Zhu i Andrew B. Goldberg, "Introduction to Semi-Supervised Learning", Universiti of Wisconsin, Madison: Morgan & Claypool publishers, 2009.

[12] Altexsoft: "Semi-Supervised Learning, Explained with Examples", s interneta, https://www.altexsoft.com/blog/semi-supervised-learning/, 5. kolovoza 2022

[13] V. Mallawaarachchi, Towardsdatascience: "Inductive vs. Transductive Learning", s interneta, https://towardsdatascience.com/inductive-vs-transductive-learning-e608e786f7d, 15. Srpnja 2022

[14] V. Vapnik: "Transductive Inference and Semi-Supervised Learning", The MIT Press, 2006.

[15] Wikipedia contributors: "Transduction (machine learning) - Wikipedia, The Free Encyclopedia ", s interneta, 20. svibnja 2022

[16] J. Tanha; M. Someren; H. Afsarmanesh: "Semi-supervised self-training for decision tree classifiers", International Journal of Machine Learning and Cybernetics, vol 8, br. str. 16, siječanj 2015.

[17] Z. Xiaojin: "Semi-Supervised Learning Literature Survey", Computer Sciences, University of Wisconsin-Madison, 2005.

[18] Wikipedia contributors: "Co-training – Wikipedia, The Free Encyclopedia", s interneta, https://en.wikipedia.org/wiki/Co-training, 12. Listopada 2022.

[19] K. Nigam and R. Ghani: "Analyzing the Effectiveness and Applicability of Co-training", s interneta, http://www.kamalnigam.com/papers/cotrain-CIKM00.pdf, 6 Siječnja 2023.

[20] V. Mallawaarachchi: "Towards Data Science", s interneta, https://towardsdatascience.com/label-propagation-demystified-cd5390f27472, 8 Listopada 2022.

[21] M. Azaouzi and L. B. Romdhane: "An evidential influence-based label propagation algorithm for ", s interneta, https://fardapaper.ir/mohavaha/uploads/2018/06/Fardapaper-An-evidential-influence-based-label-propagation-algorithm-for-distributed-community-detection-in-social-networks.pdf, 6 Siječnja 2023.

[22] S. Menon; D. Chapman; P. Nguyen; Y. Yesha; M. Morris; B. Saboury: "Deep Expectation-Maximization for Semi-Supervised", s interneta, https://arxiv.org/pdf/2010.01173v1.pdf, 23. studeni 2022.

[23] J. Šnajder: "20. Grupiranje II", s interneta, https://www.fer.unizg.hr/_download/repository/SU1-2021-P20-Grupiranje2.pdf, 15. Studeni 2022.

[24] Raman: "Expectation-Maximization Algorithm", s interneta, https://www.geeksforgeeks.org/ml-expectation-maximization-algorithm/, 15 Kolovoza 2023.

[25] S. Ding, Z. Zhu and X. Zhang: "An overview on semi-supervised support vector machine", s interneta, https://www.researchgate.net/publication/284233833_An_overview_on_semi-supervised_support_vector_machine, 3. Prosinca 2022.

[26] Z.-H. Zhou: "Machine Learning", Singapore: Springer, 2021.

[27] E. García-Martín, N. Lavesson, H. Grahn, E. Casalicchio and V. Boeva; "How to Measure Energy Consumption in Machine Learning Algorithms" Springer, Švicarska, 2018.

[28] R. Shaikh; "Cross Validation Explained: Evaluating estimator performance", s interneta, https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85, 24. Srpnja 2023.

[29] V. Lyashenko; "Cross-Validation in Machine Learning: How to Do It Right", s interneta, https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right, 24. Srpnja 2023.

[30] K. Muralidhar; "What is Stratified Cross-Validation in Machine Learning?", s interneta, https://towardsdatascience.com/what-is-stratified-cross-validation-in-machine-learning-8844f3e7ae8e, 24. Srpnja 2023.

[31] N. Bressler; "How to Check the Accuracy of Your Machine Learning Model ", s interneta, https://deepchecks.com/how-to-check-the-accuracy-of-your-machine-learning-model/, 24. Srpnja 2023.

[32] J. Korstanje; "The F1 score", s interneta, https://towardsdatascience.com/the-f1-score-bec2bbc38aa6, 26. Srpnja 2023.

[33] byjus, "Geometric Mean", s interneta, https://byjus.com/maths/geometric-mean/, 26. Srpnja 2023.

[34] deepchecks; "ROC (Receiver Operating Characteristic) Curve", s interneta, https://deepchecks.com/glossary/roc-receiver-operating-characteristic-curve/, 27. Srpnja 2023.

[35] S. Narkhede; "Understanding AUC - ROC Curve", s interneta, https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5, 27. Srpnja 2023.

[36] Anthony Corbo; "How Is Python Used in Machine Learning?", s interneta, https://builtin.com/machine-learning/python-machine-learning, 1. Kolovoza 2023.

[37] Anaconda.org; "Anaconda spyder", s interneta, https://anaconda.org/anaconda/spyder, 1. Kolovoza 2023.

[38] W. contributors; "NumPy", s interneta, https://en.wikipedia.org/wiki/NumPy, 1. Kolovoza 2023.

[39] tutorialspoint; "Scikit Learn - Introduction", s interneta, https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm, 1. Kolovoza 2023.

[40] A. Gupta; " An Introduction to Scikit-Learn: Machine Learning in Python ", s interneta, https://www.simplilearn.com/tutorials/python-tutorial/scikit-learn, 1. Kolovoza 2023.

[41] K. Jain; " Scikit-learn(sklearn) in Python – the most important Machine Learning tool I learnt last year! ", s interneta, https://www.analyticsvidhya.com/blog/2015/01/scikit-learn-python-machine-learning-tool/, 1. Kolovoza 2023.

[42] geeksforgeeks; "Introduction to Pandas", s interneta, https://www.geeksforgeeks.org/introduction-to-pandas-in-python/, 2. Kolovoza 2023.

[43] H. Sharma; "Comparing Python Libraries for Visualization", s interneta, https://medium.com/mlearning-ai/comparing-python-libraries-for-visualization-b2eb6c862542, 2. Kolovoza 2023.

[44] INRIA; "Welcome to pyJoules's documentation!", s interneta, https://pyjoules.readthedocs.io/en/latest/, 23. Srpnja 2023.

[45] INRIA; "Welcome to pyRAPL's documentation!", s interneta, https://pyrapl.readthedocs.io/en/latest/, 23. Srpnja 2023.

[46] G. Rodola; "psutil documentation", s interneta, https://psutil.readthedocs.io/en/latest/, 23. Srpnja 2023.

[47] J. Montantes; "Best Processors for Machine Learning", s interneta, https://james-montantes-exxact.medium.com/best-processors-for-machine-learning-b1fe46561c31, 2. Kolovoza 2023.

[48] Intel; "Intel® Core™ i5-9300H Processor", s interneta, https://www.intel.com/content/www/us/en/products/sku/191075/intel-core-i59300h-processor-8m-cache-up-to-4-10-ghz/specifications.html, 2. Kolovoza 2023.

[49] PureStorage ; "CPU vs. GPU for Machine Learning", s interneta, https://blog.purestorage.com/purely-informational/cpu-vs-gpu-for-machine-learning/, 2. Kolovoza 2023.

[50] UCI repozitorij skupova podataka, s interneta, https://archive.ics.uci.edu/datasets, 5 Kolovoza 2023.

[51] T. Renee; "A look at sklearn’s new estimator, the SelfTrainingClassifier", s interneta, https://medium.com/geekculture/a-look-at-sklearns-new-estimator-the-selftrainingclassifier-5a8628276abc, 6. kolovoza 2023.

SAŽETAK

Cilj ovog diplomskog rada bio je proučiti metode polu-nadziranog učenja te opisati njih i njihove prednosti i mane. Izraditi model predviđanja polu-nadziranog učenja i usporediti ga s modelom nadziranog učenja kada je dostupna mala količina označenih podataka. Vrednovati odabrane modele te izmjeriti utrošak energije, memorije i vremena potrebnog za treniranje modela za scenarije nadziranog i polu-nadziranog učenja. Za mjerenje utroška CPU-a, RAM-a i vremena treniranja modela korištena je pyRAPL knjižnica koja pruža mogućnost mjerenja samo određenog dijela programskog koda. Prilikom razvoja modela izvedeno je nekoliko mjerenja za svaki model, gdje je prilikom svakog mjerenja povećavan postotak označenih podataka kako bi se istražio omjer performansi i učinkovitosti u ovisnosti o količini označenih podataka u fazi treniranja. Najbolji rezultati obično su postignuti pri treniranju modela s 10% označenih podataka. Osim polu-nadziranih metoda u diplomskom radu opisane su i metrike vrednovanja korištene za vrednovanje modela. Također, opisan je i razvoj modela i unakrsne provjere za polu-nadzirano učenje. Osim klasičnog testiranja gdje se testiranje odvija na skupu za testiranje u unakrsnoj provjeri, u svrhu istraživanja u ovom diplomskom radu odrađeno je i testiranje na podacima namijenjenima za pseudo-označavanje. Rezultati istraživanja prikazani su u posljednjem poglavlju. Nakon analize svih skupova podataka, osim "Tumor" skupa, nadzirani model slučajne šume ima bolje rezultate u usporedbi s drugim najboljim modelima na svim skupovima podataka s razlikom u rasponu od 0,3-5% te se pokazao kao najbolji omjer performansa i učinkovitosti. Model s najvećom potrošnjom i trajanjem treniranja modela na svim skupovima podataka pokazao se polu-nadzirani model algoritma slučajne šume koji je u prosjeku ostvario 18 do 66 puta veću potrošnju CPU-a i 13,4 do 68 puta duže treniranje modela u odnosu na nadzirani model slučajne šume.

Ključne riječi **– polu-nadzirani model, pyRAPL, unakrsna provjera, učinkovitost modela**

ABSTRACT

The objective of this thesis was to learn about semi-supervised learning methods and describe them and their advantages and disadvantages. Build a semi-supervised learning predictive model and compare it to a supervised learning model when a small amount of labeled data is available. Evaluate the selected models and measure the consumption of energy, memory and time needed to train the model for supervised and semi-supervised learning scenarios. To measure CPU consumption, RAM and model training time, the pyRAPL library is used, which provides the

ability to measure only a certain part of the program code. During the development of the model, several measurements were performed for each model, where during each measurement the percentage of labeled data was increased in order to investigate the ratio of performance and efficiency depending on the amount of labeled data in the training phase. The best results are achieved when training the model with 10% labeled data. In addition to semi-supervised methods, evaluation metrics used for model evaluation are also described in the thesis. Model development and cross-validation for semi-supervised learning are also described. In addition to classic testing, where testing takes place on a test set in cross-validation, for the purpose of research in this thesis, testing was also done on data intended for pseudo-labeling. The results of the research are presented in the last chapter. After analyzing all datasets, except for the "Tumor" set, the supervised Random Forest model has better results compared to the other second-best models in all datasets with a difference in the range of 0.3-5% and proved to be the best ratio of performance and efficiency. The model with the highest consumption and duration of model training on all datasets turned out to be a semi-supervised model based on a Random Forest algorithm, which achieved an average of 18 to 66 times higher CPU consumption and 13.4 to 68 times longer model training compared to the supervised model Random Forest.

Keywords **– semi-supervised model, pyRAPL, cross-validation, model efficiency**