Strojno učenje za otkrivanje prijevare u kartičnim transakcijama

Široki, Dario

Master's thesis / Diplomski rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:704385

Rights / Prava: Attribution 3.0 Unported/Imenovanje 3.0

Download date / Datum preuzimanja: **2024-07-10**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

zir.nsk.hr

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Dario Široki

STROJNO UČENJE ZA OTKRIVANJE PRIJEVARE U KARTIČNIM TRANSAKCIJAMA

DIPLOMSKI RAD

Varaždin, 2023.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Dario Široki

Matični broj: 0016135842

Studij: Informacijsko i programsko inženjerstvo

STROJNO UČENJE ZA OTKRIVANJE PRIJEVARE U KARTIČNIM TRANSAKCIJAMA

DIPLOMSKI RAD

Mentor:

dr. sc. Bogdan Okreša Đurić

Varaždin, rujan 2023.

Dario Široki

Izjava o izvornosti

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

Autor potvrdio prihvaćanjem odredbi u sustavu FOI Radovi

Sažetak

Otkrivanje prijevara u transakcijama kreditnim karticama možda je jedan od najboljih poligona za testiranje algoritama računalne inteligencije. Ovaj problem uključuje niz izazova kao što su promjena podataka (navike kupaca se mijenjaju, a prevaranti mijenjaju svoje strategije tijekom vremena), neravnoteža kategorija podataka (prave transakcije znatno nadmašuju prijevare) i latencija donošenja zaključaka. Ovaj rad sadrži opis domene problema i korištenog skupa podataka, teorijsku obradu tehnika strojnog učenja koje se koriste za izradu modela u praktičnom dijelu rada te definicije i opise metrika koje se koriste za evaluaciju odabranih tehnika. Praktični dio rada obuhvaća izradu modela temeljenih na raznim tehnikama strojnog učenja i usporedbu njihovih performansi, u kontekstu uspješnosti otkrivanja odabranih tipova prijevara u kartičnim transakcijama.

Ključne riječi: transakcije; kreditne kartice; credit card; transaction; fraud; prijevara; detekcija prijevare; fraud detection

Sadržaj

1. Uvod 1

2. Metode i tehnike rada 2

3. Scenariji prijevare s kreditnom karticom 3

4. Sustav za otkrivanje prijevara s kreditnim karticama 5
4.1. Terminal 5
4.2. Pravila za blokiranje transakcija 5
4.3. Pravila bodovanja 6
4.4. Model vođen podacima 6
4.5. Istražitelji 7

5. Strojno učenje za otkrivanje prijevara s kreditnim karticama 8
5.1. Osnovna metodologija 8
5.2. Izazovi strojnog učenja 10
5.2.1. Neravnomjerne raspodjele klasa 10
5.2.2. Promjena skupa podataka 10
5.2.3. Inženjering značajki 10
5.2.4. Nedostatak podataka 11
5.2.5. Skalabilnost 11
5.2.6. Metrike performansi 11

6. Skup podataka 12
6.1. Generiranje profila kupaca 12
6.2. Generiranje profila terminala 13
6.3. Asocijacija kupaca s terminalima 14
6.4. Generiranje transakcija 16
6.5. Generiranje slučajeva prijevara 16
6.6. Transformacije podataka 17
6.6.1. Transformacije vremena 17
6.6.2. Transformacije jedinstvenog identifikatora kupca 18
6.6.3. Transformacije jedinstvenog identifikatora terminala 18

7. Metrike performansi 23
7.1. Matrica zabune 23
7.2. Mjere temeljene na pragu 24

7.3. Mjere bez praga 25
7.3.1. ROC krivulja 25
7.3.2. PR krivulja 26

8. Odabir modela 28
8.1. Logistička regresija 28
8.2. Stablo odlučivanja 29
8.3. Slučajna šuma 29
8.4. XGBoost 30

9. Izrada modela 31
9.1. Definiranje skupa za treniranje i skupa za testiranje 31
9.2. Skaliranje podataka 32
9.3. Treniranje modela 34
9.4. Usporedba performansi 35

10. Zaključak 39

Popis literature 41

Popis slika 42

Popis tablica 43

Popis isječaka koda 44

1. Uvod

Otkrivanje prijevara s kreditnim karticama *(engl. Credit Card Fraud Detection, CCFD)* izazovan je problem koji zahtijeva analizu velikih količina podataka o transakcijama kako bi se identificirali obrasci prijevara. Velike količine podataka, zajedno s naprednim tehnikama prevaranata, čine nemogućim da ljudski istražitelji učinkovito riješavaju taj problem. U posljednje vrijeme, CCFD sustavi se sve više nadopunjavaju algoritmima strojnog učenja, koji omogućuju pretraživanje i otkrivanje obrazaca iz velikih količina podataka. Algoritmi strojnog učenja pokazali su značajno poboljšanje učinkovitosti sustava za otkrivanje prijevara te pomažu istražiteljima pri pronalasku transakcija koje su vezane uz prijevare.

Metode i tehnike rada, prikazane u drugom poglavlju, pružaju uvid u metode i tehnike koje su korištene pri razradi teme, kako su provedene istraživačke aktivnosti, te koji su programski alati i aplikacije korišteni.

Treće poglavlje, "Scenariji prijevare s kreditnom karticom", donosi dublji uvid u različite oblike prijevara s kreditnim karticama koji su relevantni za rad. Ovaj dio omogućava shvaćanje raznolikosti situacija koje sustav mora detektirati.

Sljedeće poglavlje, "Sustav za otkrivanje prijevara s kreditnim karticama", detaljno razmatra strukturu samog sustava. Pravila za blokiranje transakcija, pravila bodovanja te model vođen podacima čine osnovnu arhitekturu sustava. Također, razmatra se i uloga istražitelja u optimizaciji sustava.

Poglavlje "Sustav za otkrivanje prijevara s kreditnim karticama" dalje detaljno istražuje uporabu strojnog učenja za postizanje ciljeva sustava. Isto tako, suočava se s izazovima kao što su neravnomjerne raspodjele klasa, promjene skupa podataka, inženjering značajki, nedostatak podataka, skalabilnost i metrike performansi.

Poglavlje "Skup podataka" istražuje generiranje profila kupaca, profila terminala, asocijaciju kupaca s terminalima te generiranje transakcija i slučajeva prijevara. Također se istražuju tehnike transformacije podataka u svrhu poboljšanja kvalitete skupa podataka.

Analiza performansi sustava opisana je u poglavlju "Metrike performansi". Matrica zabune, mjere temeljene na pragu te mjere bez praga poput ROC i PR krivulja koriste se za procjenu kvalitete sustava u praktičnom dijelu, stoga su teoretski obrađene u ovom dijelu rada.

U poglavlju "Odabir modela" odabrana su četiri algoritma strojnog učenja koja su kasnije praktično obrađena. Opisana je motivacija za odabir ovih modela koji su zatim detaljnije opisani.

"Izrada modela" istražuje različite modele za otkrivanje prijevara, uključujući stablo odlučivanja, logističku regresiju, slučajnu šumu te XGBoost. Ovi modeli predstavljaju glavne alate za postizanje cilja sustava. Prikazani su i rezultati koje modeli postižu na stvorenom skupu podataka.

2. Metode i tehnike rada

Kao alat za istraživanje korištena je internet tražilica Google Scholar[1].

Praktični dio rada implementiran je u programskom jeziku Python[2]. Uz njega je korišten niz biblioteka: Numpy[3], Matplotlib[4], Scikit-learn[5] i Pandas[6].

Za treniranje modela strojnog učenja korišten je servis Google Colaboratory. To je besplatna platforma za analizu i obradu podataka te izvođenje programskog koda korištenjem Python programskog jezika. Platforma omogućuje korisnicima stvaranje i dijeljenje Jupyter bilježnica koji sadrže tekst, programski kod i izlazne rezultate. [1] Colab je posebno popularan među istraživačima, studentima i programerima jer omogućuje izvršavanje koda na udaljenim Googleovim serverima pomoću snage računalstva u oblaku, što je korisno za zadatke koji zahtijevaju velike resurse ili brzu obradu. Colab također dolazi s podrškom za popularne biblioteke i okvire za strojno učenje, kao što su numpy i scikit-learn, što ga čini idealnim alatom za eksperimentiranje, razvoj i dijeljenje pri izradi modela strojnog učenja.

Za izradu samog rada korišten je servis Overleaf [7] gdje je ovaj dokument stvoren u LaTeX programskom jeziku.

Najprije je obavljeno temeljno istraživanje i stjecanje informacija o tematici, te je nakon toga izveden praktični segment istraživanja. Po dovršetku praktičnog dijela, sastavljen je ovaj tekstualni materijal.

[1] https://scholar.google.com/
[2] https://www.python.org/
[3] https://numpy.org/
[4] https://matplotlib.org/
[5] https://scikit-learn.org/stable/
[6] https://pandas.pydata.org/
[7] https://www.overleaf.com/

3. Scenariji prijevare s kreditnom karticom

Nilsonovo izvješće [2], koje prati industriju plaćanja, objavljeno u prosincu 2022. godine, predviđa da će gubici od kartičnih prijevara u Sjedinjenim Američkim Državama u sljedećih 10 godina iznositi ukupno 165,1 milijardu dolara, odnosno 397,4 milijarde dolara diljem svijeta, pogađajući sve dobne skupine u svakoj državi [3]. Detaljni prikaz gubitaka u svijetu uzrokovanih kartičnim prijevarama od 2014. do 2021. godine prikazan je u tablici 1.

Prema izvješću Savezne trgovačke komisije *(engl. Federal trade commission)* iz veljače 2022. godine [4], u SAD-u je 2021. godine zaprimljeno 389 737 prijava o kartičnim prijevarama. Takav visok broj prijava ukazuje na to da su prijevare s kreditnim karticama i dalje značajan problem s kojim se potrošači suočavaju. Ovaj podatak naglašava važnost rada na sprječavanju i suzbijanju kartičnih prijevara te potrebu za kontinuiranim poboljšanjem sigurnosti i zaštitom podataka kreditnih kartica.

Tablica 1: Gubici uzrokovani kartičnim prijevarama diljem svijeta; prema [2]

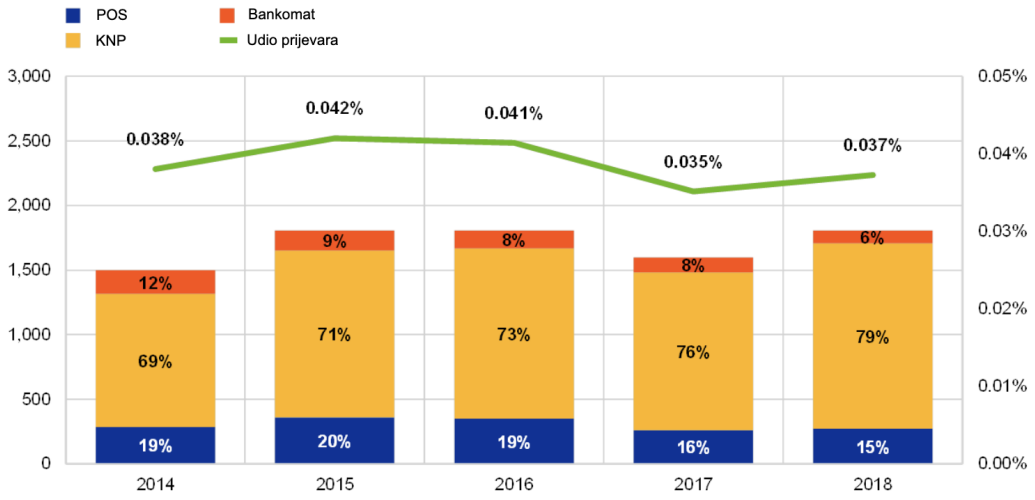
Godina	Gubitak (u milijardama američkih dolara)
2021	32.34
2020	28.43
2019	28.65
2018	27.86
2017	23.97
2016	22.80
2015	21.84
2014	18.11

Postoji veliki broj scenarija koji mogu dovesti do uspješne izvedbe prijevara s kreditnim karticama. Iako ne postoji službena klasifikacija vrsta prijevara s kreditnim karticama, poznati su određeni obrasci koji se često ponavljaju. Također treba napomenuti da je otkrivanje prijevara kao igra mačke i miša, gdje se obrasci prijevara mijenjaju tijekom vremena. Kako tehnologija napreduje, kako u smislu sprječavanja prijevara, tako i u pogledu olakšanog korištenja platnih sustava, mijenjaju se i tehnike koje prevaranti koriste. Oni se prilagođavaju prelaskom sa starih (već zakrpanih) metoda na ranjivosti novijih tehnologija. Također imaju koristi od stalnih promjena u volumenu i karakteristikama stvarnih transakcija.

Prijevare se generalno ipak mogu svrstati u dvije kategorije [5]:

- kartica-prisutna *(engl. card-present)*, kada je kartica fizički prisutna;
- kartica-nije-prisutna *(engl. card-not-present)*, kada kartica nije fizički prisutna;

Korisno je razlikovati ova dva scenarija transakcija. Prvi, nazvan "kartica-prisutna" (nadalje KP), odnosi se na situacije u kojima je potrebna fizička kartica, kao što su transakcije u trgovini (putem POS terminala) ili transakcije na bankomatu. Drugi, nazvan "kartica-nije-prisutna" (nadalje KNP), odnosi se na situacije u kojima se ne mora koristiti fizička kartica, što



Slika 1: Evolucija ukupne vrijednosti kartičnih prijevara korištenjem kartica izdanih unutar SEPA-e (lijeva skala: ukupna vrijednost prijevare (milijuni eura); desna skala: vrijednost prijevare kao udio u vrijednosti transakcija (postoci)); prema [7]

uključuje plaćanja izvršena putem interneta, telefona ili pošte.

Ova razlika je važna jer se tehnike koje se koriste za zloupotrebu kartice razlikuju, ovisno o tome treba li lažirati fizičku kopiju kartice ili ne. Osim toga, u posljednje vrijeme prevaranti su skloniji iskorištavanju nedostataka KNP scenarija nego KP scenarija, vjerojatno zato što KP scenariji postoje već više od dva desetljeća i postali su prilično otporni na prijevare, posebno zahvaljujući činjenici da se danas uz karticu koristi i PIN za veće transakcije. [6] Tvrdnju da su KNP scenariji učestaliji od KP scenarija potvrđuje i grafikon na slici 1 gdje je prikazan postotak prijevara počinjenih u KP scenarijima (POS i Bankomat) te KNP scenarijima od 2014. do 2018. godine unutar SEPA-e.

4. Sustav za otkrivanje prijevara s kreditnim karticama

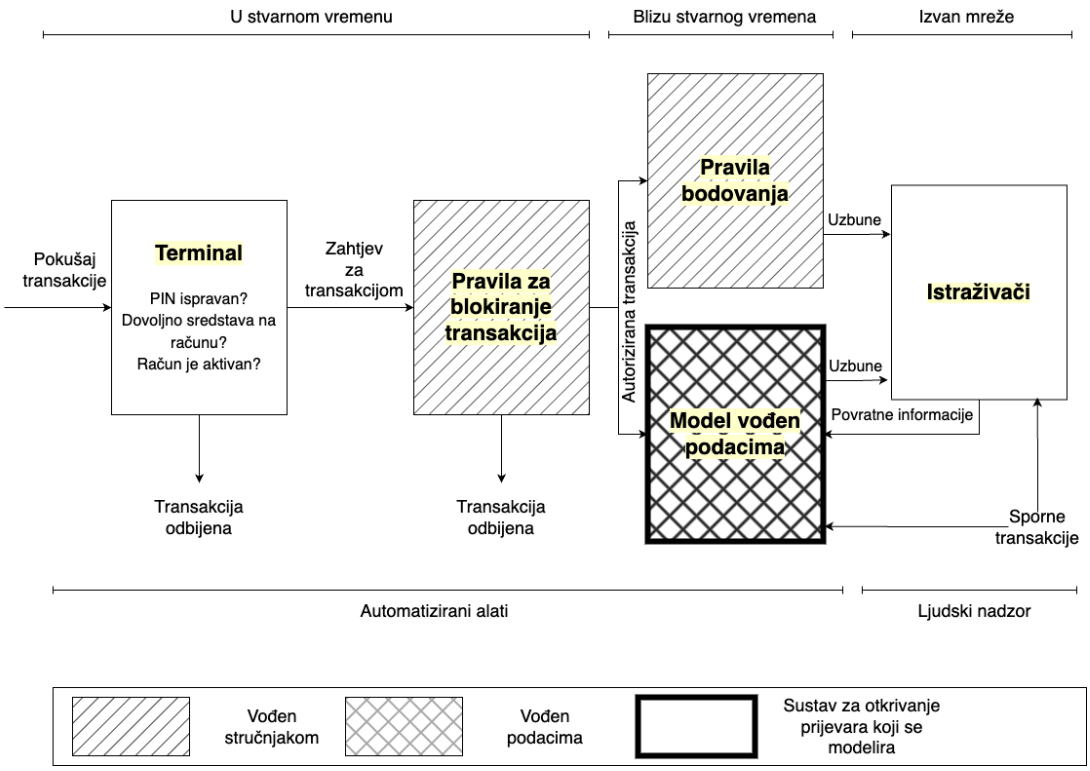
Ovo poglavlje opisuje arhitekturu sustava za otkrivanje prijevara *(engl. Fraud Detection System, FDS)* u pravome svijetu. Slika 2 prikazuje slojeve FDS-a. Prvi sloj sastoji se od automatiziranih alata koji imaju zadatak blokiranja transakcije ako je neko pravilo prekršeno, dok se drugi sloj sastoji od istraživača. Istraživači su ljudi koji ručno provjeravaju mali broj transakcija kako bi donijeli konačnu odluku o tome je li transakcija prijevara ili nije. Kako su resursi ograničeni u drugom sloju, cilj je razviti sustav koji će što bolje iskorištavati resurse u drugom sloju, podižući uzbune sa što većom točnošću kako bi bilo što manje lažnih uzbuna, ali i što veći broj otkrivenih prijevara.

4.1. Terminal

Terminal [8] je prvi sloj kontrole u FDS-u i provodi standardne sigurnosne provjere za sve transakcije. Ove sigurnosne provjere uključuju provjeru PIN koda, broj pokušaja, status kartice (aktivna ili blokirana), dostupno stanje i limit potrošnje. U slučaju mrežnih transakcija, ove provjere se moraju obavljati u stvarnom vremenu, pri čemu terminal šalje upit poslužitelju izdavatelja kartice. Zahtjevi koji ne prođu ove provjere se odbijaju, dok ostali postaju zahtjevi za transakcije koji se obrađuju u drugom sloju kontrole.

4.2. Pravila za blokiranje transakcija

Pravila za blokiranje transakcija [8] razvijaju stručnjaci u tom području i ona blokiraju zahtjeve za transakcije prije njihove autorizacije. Ta pravila djeluju u stvarnom vremenu i vrlo su precizna, odbijajući zahtjeve za transakcije samo ako jasno predstavljaju pokušaj prijevare. Pravila blokiranja su uvjetni izrazi koji povezuju klasu prijevare ili autentične transakcije s određenim uvjetima. Na primjer, jedno pravilo može biti: "ako se transakcije temeljene na PIN-u izvršavaju u Europi i Aziji unutar 5 minuta od istog vlasnika kartice, tada se transakcija odbija", jer je fizički nemoguće da kupac bude prisutan u Europi i Aziji u tako kratkom vremenskom intervalu. Zahtjevi za transakcije koji ne zadovolje nijedno pravilo za blokiranje transakcija su autorizirani i postaju transakcije, što znači da je plaćanje izvršeno. Prije nego što se transakcija prenese na sljedeći sloj FDS-a, obogaćuje se dodatnim podacima (npr. prosječna potrošnja kupca, broj transakcija istog vlasnika kartice istog dana itd.). Rezultirajući skup podataka koji opisuje transakciju potom se analizira putem pravila bodovanja i modela koji koristi podatke. Ove kontrole gotovo se izvode u stvarnom vremenu (obično manje od 6 sekundi), budući da nema potrebe za trenutačnim odgovorom s obzirom da je transakcija već autorizirana.



Slika 2: Shema koja prikazuje arhitekturu sustava za otkrivanje prijevara. Fokus ovog rada je na modelu vođenom podacima; prema [8]

4.3. Pravila bodovanja

Pravila bodovanja [8] također definiraju stručnjaci temeljem iskustva i ta pravila zatim djeluju kao klasifikator. Ona dodjeljuju bodove svakoj autoriziranoj transakciji: što je bod veći, veća je vjerojatnost da je transakcija prijevara. U praksi, pravila bodovanja sadrže jednostavne uvjete razumljive ljudima (npr. ako je transakcija u poreznoj oazi i iznos je veći od 1000 eura, tada bodovi za prijevaru = 0.9) i kao takva ih istražitelji ili drugi stručnjaci mogu lako osmisliti. Ta pravila su jednostavna za razumijevanje i brza za implementaciju, ali zahtijevaju ručno revidiranje kada im performanse padnu. Kao posljedica toga, održavanje tih pravila može biti skupo.

4.4. Model vođen podacima

Model vođen podacima [8] oslanja se na statistički model koji dodjeljuje ocjenu svakoj transakciji. Veća ocjena obično ukazuje na veću vjerojatnost da je transakcija povezana s prijevarom. U ovom sloju sustava obično se koriste algoritmi strojnog učenja koji za svaku transakciju daju procjenu vjerojatnosti da je riječ o prijevari. Ti algoritmi mogu naučiti kompleksne korelacije u podacima koristeći velike količine podataka s visokom dimenzionalnošću. Obično su otporniji od ručno izrađenih pravila, ali većina algoritama strojnog učenja su "crne kutije", što znači da ih nije moguće pretvoriti u pravila koja je lako interpretirati kako bi se razumjelo zašto je određena transakcija označena kao više ili manje rizična. Algoritmi strojnog

učenja mogu uzeti u obzir sve informacije povezane s transakcijom, dok ručno izrađena pravila temelje uvjete samo na nekoliko značajki transakcija.

4.5. Istražitelji

Istražitelji [8] su stručnjaci za prijevare koji provjeravaju upozorenja na prijevare, odnosno transakcije koje automatizirani sustavi prijave kao rizične. Ove sumnjive transakcije prikazuju se u alatu za upravljanje slučajevima gdje istražitelji mogu vidjeti pripadajuću ocjenu prijevare, provjeriti odakle dolazi i označiti ih kao autentične ili prijevare nakon provjere. U stvarnom scenariju istražitelji mogu provjeriti samo nekoliko upozorenja s obzirom na ograničeni broj istražitelja. Iz tog razloga, obično istražuju samo transakcije s najvišom ocjenom prijevare. Uloga istražitelja je kontaktirati vlasnika kartice, provjeriti je li transakcija prijevara ili ne, i označiti je s odgovarajućom oznakom. Ovaj proces može biti dugotrajan i zamoran, a istovremeno je broj transakcija koje treba provjeriti obično vrlo velik, stoga je važno minimizirati lažna upozorenja. Postupak označavanja generira označene transakcije koje zatim konzumira model vođen podacima kako bi se ostvarila povratna veza.

5. Strojno učenje za otkrivanje prijevara s kreditnim karticama

Otkrivanje prijevara s kreditnim karticama je izazovno, no tehnike strojnog učenja pokazale su se učinkovitima u sprečavanju i otkrivanju prijevara s kreditnim karticama. Modeli strojnog učenja mogu analizirati velike količine transakcijskih podataka i prepoznati uzorke u podacima koji ukazuju na prijevaru. Ti modeli mogu učiti na temelju povijesnih podataka i u stvarnom vremenu označavati potencijalno sumnjive transakcije.

Algoritmi strojnog učenja mogu se trenirati na označenim skupovima podataka, gdje su transakcije koje su prijevare unaprijed označene kao takve. Učeći iz tih označenih primjera, modeli mogu generalizirati uzorke i otkriti sumnjivo ponašanje čak i u prethodno neviđenim transakcijama. Inženjering značajki *(engl. Feature engineering)* ima ključnu ulogu u izgradnji kvalitetnih skupova podataka iz kojih modeli mogu učiti.

Ljudski istražitelji također imaju značajnu ulogu u otkrivanju prijevara. Iako modeli strojnog učenja mogu automatizirati početni postupak prepoznavanja i označiti potencijalno sumnjive transakcije, ljudski istražitelji su ključni za daljnju analizu i donošenje odluka. Istražitelji mogu proučiti označene transakcije, provesti detaljnije istrage i provjeriti sumnjive aktivnosti. Mogu iskoristiti dodatne informacije i stručnost iz područja kako bi donijeli informirane prosudbe o tome je li transakcija dio prijevare ili je legitimna.

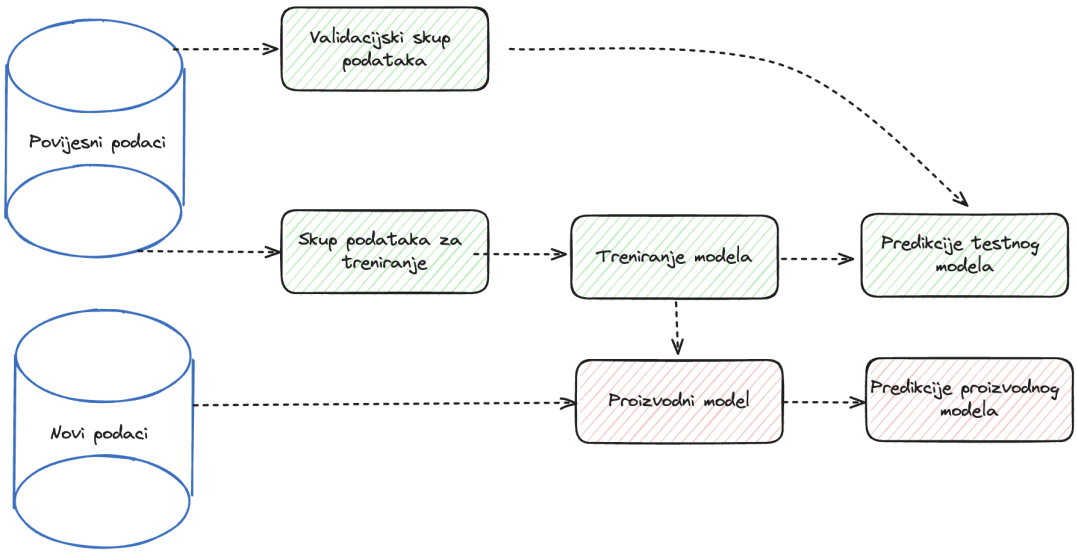
Međutim, s obzirom na velik broj transakcija koje se odvijaju u sustavima kreditnih kartica, bilo bi nepraktično da ljudski istražitelji ručno pregledavaju svaku pojedinu transakciju. Tu modeli strojnog učenja dolaze do izražaja automatiziranjem početnog postupka detekcije i značajno smanjuju broj transakcija koje zahtijevaju ljudsku pažnju. Kombinacijom algoritama strojnog učenja i ljudske stručnosti, financijske institucije mogu poboljšati svoju sposobnost otkrivanja i sprječavanja prijevara s kreditnim karticama.

5.1. Osnovna metodologija

Za rješavanje problema CCFD-a može se koristiti velik broj tehnika strojnog učenja. To se vidi i po velikoj količini radova objavljenih na tu temu u posljednjem desetljeću. Unatoč velikom opsegu istraživačkog rada, većina predloženih pristupa slijedi zajedničku osnovnu metodologiju strojnog učenja koja je prikazana na slici 3. [6]

Proces izrade CCFD sustava može se podijeliti u dvije faze. Prva faza sastoji se od treniranja modela strojnog učenja na skupu označenih povijesnih podataka. Ovaj proces se u strojnom učenju naziva nadziranim učenjem jer je unaprijed poznato jesu li transakcije sumnjive ili legitimne u skupu povijesnih podataka. U praksi obično nastaje više modela, a onaj koji postiže najbolje rezultate postaje proizvodna verzija. Ta proizvodna verzija zatim se koristi u stvarnom sustavu kako bi davala predikcije o legitimnosti pravih transakcija.

Ovakvi predikcijski modeli su matematički modeli koji se koriste za predviđanje budućih vrijednosti ili ponašanja temeljem ulaznih podataka. Ovakvi modeli konstruiraju se na teme-



Slika 3: Strojno učenje za CCFD, skica osnovne metodologije; prema [6]

lju analize i obrade velikog skupa povijesnih podataka, čime stiču sposobnost prepoznavanja obrazaca i znanja iz prošlih događaja. Kada je model potpuno obučen, može se primijeniti na nove, neviđene podatke kako bi se donijele predikcije ili procjene o budućim događajima ili vrijednostima.

Predikcijski modeli mogu biti različiti u svom obliku i kompleksnosti, uključujući linearne regresije, neuronske mreže, stabla odlučivanja *(engl. decision tree)*, slučajne šume *(engl. random forest)*, i mnoge druge. Svaki od ovih modela ima svoje prednosti i ograničenja, te odabir odgovarajućeg modela ovisi o prirodi problema i dostupnim podacima.

Kada se model primijeni na nove podatke, generira izlaz koji može biti u obliku numeričkih vrijednosti (npr. cijena nekretnine, broj prodanih proizvoda) ili kategorija (npr. klasifikacija e-maila kao spam ili ne-spam). Važno je naglasiti da predikcijski modeli nisu uvijek stopostotno točni, već se temelje na statističkim i matematičkim analizama te pružaju vjerojatnost da će predviđanje biti ispravno.

Modeli za CCFD obično pružaju predikcije u obliku vrijednosti koja pripada otvorenom intervalu (0, 1). Ova vrijednost može se interpretirati kao vjerojatnost da je transakcija povezana s prijevarom. Što je vrijednost bliža 1, veća je vjerojatnost da je transakcija sumnjiva, dok vrijednost bliža 0 sugerira da je transakcija vjerojatno legitimna. U matematičkom smislu, ovi modeli se često izražavaju korištenjem funkcija koje daju vrijednost između 0 i 1, poput sigmoidne funkcije. Binarni klasifikatori, kao što su ovi modeli, imaju ključnu ulogu u razvrstavanju ulaznih podataka u dvije klase, što omogućuje precizno razlikovanje sumnjivih i legitimnih transakcija.

5.2. Izazovi strojnog učenja

Strojno učenje za otkrivanje prijevara je izuzetno zahtjevan problem. Ovo potpoglavlje opisuje neke od uobičajenih izazova s kojima se treba suočiti prilikom razvoja rješenja.

5.2.1. Neravnomjerne raspodjele klasa

Ovaj problem prisutan je u svakoj primjeni strojnog učenja za otkrivanja prijevara s kreditnim karticama. Otkrivanje prijevara je kao traženje igle u plastu sijena. Postotak prijevara obično je manji od 0,4%. Drugim riječima, osnovni klasifikator s točnošću od 99,6% je neprihvatljiv i treba ga poboljšati. Učenje iz neuravnoteženih skupova podataka [9] je težak zadatak jer većina sustava za učenje nije pripremljena za suočavanje s velikom razlikom između broja slučajeva koji pripadaju svakoj klasi. Taj se problem rješava na razini podataka ili algoritamskoj razini. Na razini podataka, koriste se razne prilagođene tehnike uzorkovanja (prekomjerno i smanjeno). Obično se ili nasumično uklanjaju transakcije koje nisu sumnjive ili se stvaraju novi primjeri sumnjive klase. Postoje i hibridni pristupi gdje se smanjuje broj transakcija većinske klase i povećava broj manjinske klase.

Na algoritamskoj razini, vrše se modifikacije osnovnih klasifikacijskih algoritama. One se mogu podijeliti na učenje s neuravnoteženim skupovima podataka i učenje s osjetljivošću na trošak [9]. Prve povećavaju točnost manjinske klase, a potonje smanjuju povezani trošak. Treba napomenuti da je, u otkrivanju prijevara, trošak različit za lažno pozitivne i lažno negativne rezultate.

5.2.2. Promjena skupa podataka

Taktike prijevare se mijenjaju s vremenom. [9] Neke metode prijevare postaju zastarjele zbog stalne igre mačke i miša između prevaranata i stručnjaka, a nove tehnologije pokazuju slabosti koje mogu biti iskorištene od strane prevaranata. Osim toga, ponašanje vlasnika kreditnih kartica se mijenja tijekom vremena. Kombinacija navedenih čimbenika čini tipične algoritme zastarjelima ili nevažnima. Algoritmi u stvarnom svijetu moraju biti dizajnirani tako da se redovito ažuriraju ili na način da se otkrije smanjenje učinkovitosti.

5.2.3. Inženjering značajki

Sami transakcijski podaci nisu dovoljni za dobivanje kvalitetnog rješenja. Transakcijski podaci se često obogaćuju agregiranim podacima iz prošlih transakcija, a ako su dostupni, i dodatnim podacima iz drugih izvora (npr. demografski podaci).[9] Primjerice, jedan takav izvedeni atribut može biti prosječni iznos potrošen u prethodnim transakcijama u određenom vremenskom okviru. Ovi agregirani atributi zajedno se odnose na profil kupca, to jest njegovo ponašanje. Nije neobično izgraditi nekoliko stotina takvih varijabli.

5.2.4. Nedostatak podataka

Postoji nedostatak javno dostupnih visokokvalitetnih skupova podataka. [9] Mnoge relevantne informacije su prikrivene ili izostavljene zbog povjerljivosti, kao što su definicije agregiranih varijabli. Neki istraživači čak koriste sintetičke skupove podataka. Nedostatak javno dostupnih skupova otežava usporedbu modela i olakšava donošenje krivih zaključaka do kojih se može doći korištenjem skupa podataka koji ima određene nedostatke. Osim toga, ovo područje se suočava s visokom volatilnošću, jer se potrošačke navike mijenjaju tijekom vremena (npr. razdoblje pandemije), a skupovi podataka postaju zastarjeli. Također, uzorci korištenja se razlikuju na različitim tržištima.

5.2.5. Skalabilnost

Sustavi za otkrivanje prijevara moraju biti sposobni brzo otkriti sumnjive transakcije. Podaci o kreditnim karticama su veliki podaci *(engl. Big data)*. Također ih se može smatrati podacima koji se obično kreću kroz tokove *(engl. stream)*. Potrebno je obraditi veliki broj transakcija u sekundi, a procjene prijevara moraju biti izračunate u milisekundama. Izazovno je dizajnirati takav skalabilan sustav. [9]

5.2.6. Metrike performansi

Standardne mjere za klasifikacijske sustave, poput srednje greške klasifikacije *(engl. mean classification error)*, nisu prikladne za probleme otkrivanja anomalija zbog neravnoteže broja primjera sumnjivih i legitimnih transakcija u skupovima podataka. [9] Sustav za otkrivanje prijevara trebao bi maksimizirati otkrivanje sumnjivih transakcija istovremeno minimizirajući broj pogrešno predviđenih prijevara (lažno pozitivnih rezultata). Često je potrebno razmotriti više mjera kako bi se ocijenila ukupna učinkovitost sustava za otkrivanje prijevara. Unatoč ključnoj ulozi u dizajnu sustava za otkrivanje prijevara, trenutno ne postoji suglasnost o tome koje mjere učinkovitosti treba koristiti.

6. Skup podataka

Za skup podataka koji će se u kasnijim poglavljima koristiti pri treniranju raznih modela odabran je pristup koji su koristili Le Borgne i sur. (2022) [6]. U njihovom radu opisuje se simulator koji generira skupove transakcija, kupaca i terminala.

Proces generiranja transakcija sastoji se od pet koraka: [6]

- **Generiranje profila kupaca** - Svaki kupac se razlikuje u svojim navikama trošenja. To će se simulirati definiranjem nekih svojstava za svakog kupca. Glavna svojstva bit će njihova geografska lokacija, učestalost trošenja i iznosi transakcija.
- **Generiranje profila terminala** - Svojstva terminala sastojat će se samo od geografske lokacije.
- **Asocijacija kupaca s terminalima** - Iskoristit će se pretpostavka da kupci obavljaju transakcije samo na terminalima koji se nalaze unutar radijusa r od njihove geografske lokacije. To znači da kupci obavljaju transakcije na terminalima koji su geografski blizu njihove lokacije. Ovaj korak će uključivati dodavanje atributa 'lista terminala' svakom profilu kupca, koji sadrži skup terminala koje kupac može koristiti.
- **Generiranje transakcija** - Simulacija će iterirati kroz skup profila kupaca i generirati transakcije prema njihovim svojstvima (učestalosti i iznosima trošenja, te dostupnim terminalima).
- **Generiranje slučajeva prijevara** - Posljednji korak označit će transakcije kao legitimne ili sumnjive. To će se postići uz pomoć tri različita scenarija prijevare koji su opisani kasnije.

6.1. Generiranje profila kupaca

Svakog kupca definiraju sljedeće varijable:

- **id_kupac** - Jedinstveni identifikator kupca.
- **(x_poz, y_poz)** - Koordinate iz prostora veličine 100x100 koje predstavljaju geografsku lokaciju kupca, brojevi su izvučeni iz jednolike raspodjele.
- **(mean_iznos, std_iznos)** - Srednji iznos i standardna devijacija svih transakcija kupca. Srednji iznos je broj između 5 i 100, izvučen iz jednolike raspodjele, a standardna devijacija je zadana kao polovica srednjeg iznosa.
- **prosjek_transakcija_po_danu** - Prosječan broj transakcija po danu za kupca. Broj između 0 i 4, izvučen iz jednolike raspodjele.

Isječak programskog koda koji je zadužen za generiranje profila kupaca prikazan je isječkom koda 1, a tablica koja prikazuje pet uzoraka iz stvorenog skupa podataka prikazana je tablicom 2. Funkcija vraća podatke u Pandas okviru koji olakšavaju rad sa podacima. Pandas okviri će se učestalo koristiti i dalje kroz ovaj rad.

```
def generiraj_kupce(broj_kupaca):
  kupci = []

  for id_kupac in range(broj_kupaca):
    x_poz = np.random.uniform(0,100)
    y_poz = np.random.uniform(0,100)

    mean_iznos = np.random.uniform(5,100)
    std_iznos = mean_iznos/2

    prosjek_transakcija_po_danu = np.random.uniform(0,4)

    kupci.append([id_kupac, x_poz, y_poz, mean_iznos, std_iznos,
    ↪  prosjek_transakcija_po_danu])

  return pd.DataFrame(kupci, columns=["id_kupac", "x_poz",
  ↪  "y_poz","mean_iznos", "std_iznos",
  ↪  "prosjek_transakcija_po_danu"])
```

Isječak koda 1: Isječak programskog koda koji generira profile kupaca; prema [6]

Tablica 2: Primjer kupaca generiranih za skup podataka. PTPD predstavlja prosjek transakcija po danu. Vlastita izrada

id_kupac	x_poz	y_poz	mean_iznos	std_iznos	PTPD
0	28.51	60.53	40.68	20.34	3.99
1	69.14	55.05	44.02	22.01	3.69
2	35.51	61.86	22.09	11.04	2.80
3	45.98	34.56	60.15	30.07	1.06
4	91.32	1.09	9.03	4.51	1.91

6.2. Generiranje profila terminala

Svaki terminal definiraju sljedeće varijable:

- **id_terminal** - Jedinstveni identifikator terminala.
- **(x_poz, y_poz)** - Koordinate iz prostora veličine 100x100 koje predstavljaju geografsku lokaciju terminala, brojevi su izvučeni iz jednolike raspodjele.

Isječak programskog koda koji je zadužen za generiranje profila terminala prikazan je isječkom koda 2, a tablica koja prikazuje pet uzoraka iz stvorenog skupa podataka prikazana

je tablicom 3.

```
def generiraj_terminale(broj_terminala):
  terminali = [
      [
        id_terminal,
        np.random.uniform(0,100), # x_poz
        np.random.uniform(0,100) # y_poz
      ]
      for id_terminal in range(broj_terminala)
  ]

  return pd.DataFrame(terminali, columns=['id_terminal', 'x_poz',
   ↪  'y_poz'])
```

Isječak koda 2: Isječak programskog koda koji generira profile terminala; prema [6]

Tablica 3: Primjer terminala generiranih za skup podataka; vlastita izrada

id_terminal	x_poz	y_poz
0	77.05	14.69
1	7.95	8.96
2	67.20	24.53
3	42.05	55.73
4	86.05	72.70

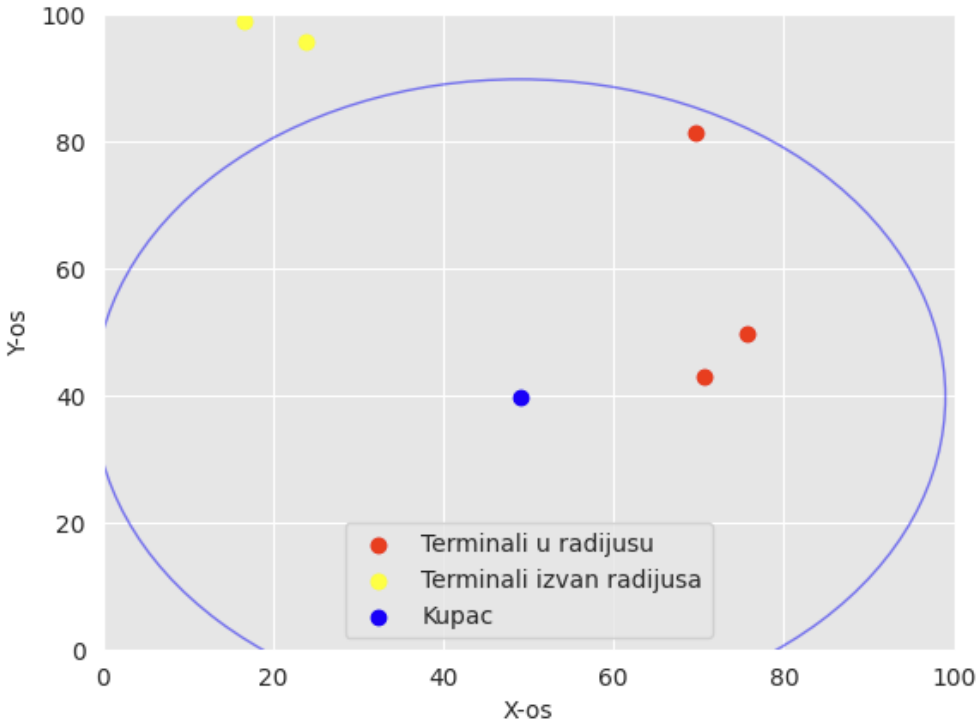
6.3. Asocijacija kupaca s terminalima

Sada je potrebno povezati terminale s profilima kupaca. Kupci mogu obavljati transakcije samo na terminalima koji su unutar radijusa r od njihovih geografskih lokacija.

Funkcija nazvana $terminali\_u\_radijusu$, prikazana isječkom koda 3, pronalazi te terminale za profil kupca. Funkcija će kao ulaz primiti profil prethodno generiranog kupca, polje koje sadrži listu svih terminala i radijus r te vratiti popis terminala unutar radijusa r za tog kupca. Kako bi se ubrzalo izvršavanje ove funkcije, koriste se funkcije iz biblioteke Numpy. Brzina izvršavanja će kasnije biti bitna kod generiranja cijelog skupa podataka kako nebi trajalo predugo.

Na slici 4 je prikazana vizualizacija terminala koji su unutar radijusa kupca i terminala koji su izvan radijusa.

Nakon što se svakom kupcu dodaju terminali koji su u njegovom radijusu, dobije se tablica podataka kao što je prikazano na slici 5.



Slika 4: Vizualizacija terminala u radijusu kupca; vlastita izrada

kupac_id	x_poz	y_poz	mean_iznos	std_iznos	prosjek_transakcija_po_danu	terminali_u_radijusu
0	48.969252	39.779171	50.064983	25.032492	3.933224	[0, 1, 2]
1	45.357625	78.919597	77.496086	38.748043	1.087858	[0, 1, 2, 3, 4]
2	24.273542	82.785711	65.425063	32.712532	1.403603	[2, 3, 4]
3	83.536031	27.501544	65.425894	32.712947	0.610086	[0, 1]
4	8.544722	32.115524	46.901558	23.450779	2.065237	[]

Slika 5: Kupci sa pridruženim terminalima u radijusu; vlastita izrada

```
def terminali_u_radijusu(kupac, terminali, radijus):
  x_y_kupac = kupac[['x_poz','y_poz']].values.astype(float)

  kvadrati_razlike = np.square(x_y_kupac - terminali)
  daljine_x_y = np.sqrt(np.sum(kvadrati_razlike, axis=1))

  return list(np.where(daljine_x_y < radijus)[0])
```

Isječak koda 3: Isječak programskog koda koji generira profile terminala; prema [6]

6.4. Generiranje transakcija

Transakcija se sastoji od iznosa koji je kupac platio trgovcu u određeno vrijeme. Šest glavnih značajki koje opisuju transakciju su sljedeće: [6]

- **id_transackije** - Jedinstveni identifikator transakcije.
- **id_kupac** - Jedinstveni identifikator kupca.
- **id_terminal** - Jedinstveni identifikator trgovca (ili preciznije, terminala). Svaki terminal ima jedinstveni identifikator.
- **vrijeme_transakcije** - Datum i vrijeme kada se transakcija dogodila.
- **iznos** - Iznos transakcije.
- **prijevara** - Oznaka prijevare: Binarna varijabla, s vrijednošću 0 za autentičnu transakciju ili vrijednošću 1 za transakciju koja je prijevara.

Isječak programskog koda koji je zadužen za generiranje transakcija prikazan je isječkom koda 4, a tablica koja prikazuje pet uzoraka iz stvorenog skupa podataka prikazana je tablicom 4.

6.5. Generiranje slučajeva prijevara

U zadnjem koraku simulacije, prikazanom isječkom koda 5, dodaju se transakcije koje su prijevare u skup podataka. Funkcija će transakcijama dodati dva nova atributa, $prijevara$ i $prijevara\_scenarij$. Koriste se sljedeći scenariji prijevara, prema [6]:

- **Scenarij 1** - Bilo koja transakcija čiji je iznos veći od 220 smatra se prijevarom. Ovaj scenarij ne odgovara uzorku u skupu podataka koji bi se koristio u stvarnome svijetu, no pruža očigledan uzorak prijevare koji bi trebao otkriti bilo koji osnovni detektor prijevara. To će biti korisno za analizu tehnika otkrivanja prijevara.
- **Scenarij 2** - Za svaki dan se nasumično izvlači lista od dva terminala. Sve transakcije na tim terminalima u sljedećih 28 dana označene su kao prijevare. Ovaj

Tablica 4: Primjer transakcija generiranih za skup podataka; vlastita izrada

id	datum	id_kupac	id_terminal	iznos	sekunda	dan
0	2022-01-01 07:59:21	1	2	74.75	28761	0
1	2022-01-01 06:28:13	1	1	16.65	23293	0
2	2022-01-02 07:42:02	1	3	181.80	114122	1
3	2022-01-02 06:26:53	1	1	105.11	109613	1
4	2022-01-02 14:49:18	1	4	69.08	139758	1

scenarij simulira kriminalnu uporabu terminala. Otkrivanje ovog scenarija bit će moguće dodavanjem atributa koji prate broj prijevarnih transakcija na terminalu.

- **Scenarij 3** - Za svaki dan se nasumično izvlači lista od 3 korisnika. U sljedećih 14 dana, 1/3 njihovih transakcija ima iznos pomnožen s 5 i označen je kao prijevara. Ovaj scenarij simulira prijevaru s karticom koja nije prisutna, gdje su podaci korisnika procurili. Korisnik nastavlja s transakcijama, a transakcije većih vrijednosti obavlja prevarant koji pokušava maksimizirati svoj dobitak.

6.6. Transformacije podataka

Iako su atributi koje skup podataka sadrži jednostavni za shvaćanje čovjeku, modeli strojnog učenja ih neće moći prihvatiti jer oni uglavnom rade sa numeričkim varijablama koje imaju neko značenje. U generiranom skupu podataka jedini atributi koji su numerički su $iznos$ i $je\_prijevara$. Jedinstveni identifikatori terminala, kupaca i transakcije jesu numerički, no algoritmi bi mogli donositi krive zaključke na temelju identifikatora, kao npr. da je ID 200 veći od ID-a 300 i ID bi zatim mogao utjecati na rezultate, dok je ID zapravo u potpunosti nekoreliran sa činjenicom je li transakcija prijevara ili nije. Ne postoji standardna procedura za transformiranje ovih atributa u numeričke varijable, no postoji niz tehnika koje se mogu primijeniti. Prije nego se pristupi izradi modela strojnog učenja, potrebno je obaviti neke transformacije podataka.

6.6.1. Transformacije vremena

Prva vrsta transformacije uključuje vrijeme transakcija i stvaranje dvije nove značajke koje označavaju da li se transakcija dogodila tijekom radnog tjedna ili vikenda (atribut je_vikend)

te da li je bila tijekom dana ili noći (atribut je_noc). Ove značajke su korisne jer se u stvarnom svijetu primjećuju različiti oblici prijevara tijekom radnih dana i vikenda, kao i tijekom dnevnih i noćnih sati. [6] Isječak koda koji prethodno navedena dva atributa pridružuje skupu podataka prikazan je isječkom koda 6.

6.6.2. Transformacije jedinstvenog identifikatora kupca

Druga vrsta transformacije odnosi se na identifikatore kupaca. Ovdje se stvaraju nove značajke koje opisuju potrošačke navike kupca. Prati se prosječan iznos potrošnje i broj transakcija za svakog kupca u različitim vremenskim intervalima. Generirat će se šest novih atributa koji prate potrošačke navike:

- $id\_kupac\_br\_transakcija\_prozor\_ < interval > d$ - Broj transakcija u vremenskom prozoru, gdje je $< interval >$ 1, 7 ili 30 dana.
- $id\_kupac\_prosjecan\_iznos\_prozor\_ < interval > d$ - Prosjecan iznos transakcija u vremenskom prozoru, gdje je $< interval >$ 1, 7 ili 30 dana.

6.6.3. Transformacije jedinstvenog identifikatora terminala

Treća vrsta transformacije odnosi se na identifikatore terminala. Stvaraju se nove značajke koje opisuju rizik povezan s terminalom. Rizik se definira kao prosječan broj prijevara zabilježenih na tom terminalu u različitim vremenskim razdobljima (prozorima).

Za razliku od transformacija identifikatora kupaca, vremenski prozori neće izravno prethoditi određenoj transakciji. Umjesto toga, bit će pomaknuti unatrag za određeni vremenski period. Taj period pomaka uzima u obzir činjenicu da se u praksi prijevare obično otkrivaju tek nakon istrage o prijevari ili prigovora kupca. Zato su oznake prijevara, koje su potrebne za izračunavanje ocjene rizika, dostupne tek nakon tog perioda pomaka. Gruba procjena za taj period pomaka iznosi jedan tjedan. [6]

```
def dodaj_sekunde_datumu(datum_str, seconds):
  datum_objekt = datetime.datetime.strptime(datum_str, '%Y-%m-%d')

  offset = datetime.timedelta(seconds=seconds)

  datum = datum_objekt + offset

  rezultat = datum.strftime('%Y-%m-%d %H:%M:%S')

  return rezultat

dan_u_sekundama = 24 * 60 * 60
pet_sati_u_sekundama = 5 * 60 * 60

def generiraj_transakcije(kupac, pocetni_datum, broj_dana):
  if len(kupac["lista_terminala"]) == 0: return pd.DataFrame([])

  transakcije = []
  random.seed(int(kupac["id_kupac"]))
  np.random.seed(int(kupac["id_kupac"]))

  id_transakcija = 0

  for dan in range(broj_dana):
    for i in range(np.random.poisson(kupac["prosjek_transakcija_po_d
    ↪  anu"])):

      # Oko podneva, standardna devijacija od 5 sati
      vrijeme_transakcije = round(np.random.normal(dan_u_sekundama /
      ↪  2, pet_sati_u_sekundama))

      if 0 <= vrijeme_transakcije < dan_u_sekundama:
        iznos = np.random.normal(kupac["mean_iznos"],
        ↪  kupac["std_iznos"])

        if iznos < 0:
          iznos = np.random.uniform(0, kupac["mean_iznos"] * 2)

        id_terminal = random.choice(kupac["lista_terminala"])
        offset = vrijeme_transakcije + dan * dan_u_sekundama
        vrijeme_transakcije_apsolutno =
        ↪  dodaj_sekunde_datumu(pocetni_datum, offset)

        transakcije.append([id_transakcija,
        ↪  vrijeme_transakcije_apsolutno, kupac["id_kupac"],
        ↪  id_terminal, round(iznos, 2), offset, dan])
        id_transakcija += 1

  transakcije = pd.DataFrame(transakcije, columns=['id_transakcija',
  ↪  'datum_vrijeme','id_kupac', 'id_terminal', 'iznos','sekunda',
  ↪  'dan'])

  return transakcije
```


Isječak koda 4: Isječak programskog koda koji generira profile terminala; prema [6]

```
def dodaj_prijevare(kupci, terminali, transakcije):
  transakcije["prijevara"] = 0
  transakcije["prijevara_scenarij"] = 0

  # Prvi scenarij
  transakcije.loc[transakcije["iznos"] > 220, "prijevara"] = 1
  transakcije.loc[transakcije["iznos"] > 220, "prijevara_scenarij"]
   ↪  = 1

  # Drugi scenarij
  for dan in range(transakcije["dan"].max()):
    ugrozeni_terminali = terminali["id_terminal"].sample(n=2)

    prijevarne_transakcije = transakcije[
        (transakcije["dan"] >= dan) &
        (transakcije["dan"] < dan + 28) &
        (transakcije["id_terminal"].isin(ugrozeni_terminali))
    ]

    transakcije.loc[prijevarne_transakcije.index, "prijevara"] = 1
    transakcije.loc[prijevarne_transakcije.index,
     ↪  "prijevara_scenarij"] = 2

  # Treci scenarij
  for dan in range(transakcije["dan"].max()):
    ugrozeni_kupci = kupci["id_kupac"].sample(n=3,
     ↪  random_state=dan).values

    prijevarne_transakcije = transakcije[
        (transakcije["dan"] >= dan) &
        (transakcije["dan"] < dan + 14) &
        (transakcije["id_kupac"].isin(ugrozeni_kupci))
    ]


    random.seed(dan)
    indeksi_prijevara =
     ↪  random.sample(list(prijevarne_transakcije.index.values),
     ↪  k=round(len(prijevarne_transakcije) / 3))

    transakcije.loc[indeksi_prijevara, "iznos"] =
     ↪  transakcije.loc[indeksi_prijevara, "iznos"] * 5
    transakcije.loc[indeksi_prijevara, "prijevara"] = 1
    transakcije.loc[indeksi_prijevara, "prijevara_scenarij"] = 3

  return transakcije
```

Isječak koda 5: Isječak programskog koda koji generira prijevare; prema [6]

```
transakcije["datum_vrijeme"] =
↪  pd.to_datetime(transakcije["datum_vrijeme"])
transakcije['je_vikend'] = transakcije["datum_vrijeme"].apply(lambda
↪  x: int(x.weekday() >= 5))
transakcije['je_noc'] = transakcije["datum_vrijeme"].apply(lambda x:
↪  x.hour <= 6)
```

Isječak koda 6: Isječak programskog koda koji generira atribute je_vikend i je_noc; prema [6]

```
def transformacije_kupca(transakcije_kupca,
↪  velicine_prozora=[1,7,30]):
    transakcije_kupca =
    ↪  transakcije_kupca.sort_values("datum_vrijeme")
    transakcije_kupca.index = transakcije_kupca["datum_vrijeme"]

    for velicina_prozora in velicine_prozora:
        prozor_zbroj_transakcija = transakcije_kupca["iznos"].rollin
        ↪  g(window=f"{velicina_prozora}d").sum()
        prozor_br_transakcija = transakcije_kupca["iznos"].rolling(w
        ↪  indow=f"{velicina_prozora}d").count()

        prozor_prosjek_transakcije = prozor_zbroj_transakcija /
        ↪  prozor_br_transakcija

        transakcije_kupca[f"id_kupac_br_transakcija_prozor_{velicina
        ↪  _prozora}d"] =
        ↪  list(prozor_br_transakcija)
        transakcije_kupca[f"id_kupac_prosjecan_iznos_prozor_{velicin
        ↪  a_prozora}d"] =
        ↪  list(prozor_prosjek_transakcije)

    transakcije_kupca.index = transakcije_kupca["id_transakcija"]
    return transakcije_kupca
```

Isječak koda 7: Isječak programskog koda koji generira karakteristike kupca; prema [6]

```
def transformacije_terminala(transakcije_terminala,
↪  razdoblje_odgode, velicine_prozora):
    transakcije_terminala =
    ↪  transakcije_terminala.sort_values("datum_vrijeme")
    transakcije_terminala.index =
    ↪  transakcije_terminala["datum_vrijeme"]

    odgodjeno_br_prijevara = transakcije_terminala["prijevara"].roll
    ↪  ing(f"{razdoblje_odgode}d").sum()
    odgodjeno_br_transakcija = transakcije_terminala["prijevara"].ro
    ↪  lling(f"{razdoblje_odgode}d").count()

    for velicina_prozora in velicine_prozora:
        # op = odgodjeni prozor, razdoblje odgode + velicina_prozora
        op_br_prijevara = transakcije_terminala["prijevara"].rolling
        ↪  (f"{razdoblje_odgode+velicina_prozora}d").sum()
        op_br_transakcija = transakcije_terminala["prijevara"].rolli
        ↪  ng(f"{razdoblje_odgode+velicina_prozora}d").count()

        prozor_br_prijevara = op_br_prijevara -
        ↪  odgodjeno_br_prijevara
        prozor_br_transakcija = op_br_transakcija -
        ↪  odgodjeno_br_transakcija

        prozor_rizik = prozor_br_prijevara / prozor_br_transakcija

        transakcije_terminala[f"id_terminal_br_transakcija_prozor_{v
        ↪  elicina_prozora}d"] =
        ↪  list(prozor_br_transakcija)
        transakcije_terminala[f"id_terminal_rizik_prozor_{velicina_p
        ↪  rozora}d"] =
        ↪  list(prozor_rizik)

    transakcije_terminala.index =
    ↪  transakcije_terminala["id_transakcija"]
    transakcije_terminala = transakcije_terminala.fillna(0)
    return transakcije_terminala
```

Isječak koda 8: Isječak programskog koda koji obavlja transformacije terminala; prema [6]

7. Metrike performansi

Ovo poglavlje opisuje kako procijeniti učinkovitost sustava za otkrivanje prijevara. U strojnom učenju ne postoji savršeno uputstvo za odabir metrika jer ih je potrebno prilagoditi domeni problema. Idealni sustav za otkrivanje prijevara trebao bi maksimizirati broj točnih klasifikacija i otkriti sve prijevarne transakcije. U problemima strojnog učenja često se koristi preciznost (omjer stvarno pozitivnih predikcija u odnosu na sve pozitivne predikcije) i primamljivo je pomisliti da je to metrika koju treba optimizirati. Jednostavan način za ilustraciju toga koliko je to loša ideja je primijetiti da će model koji sve transakcije klasificira kao autentične, u skupu podataka s 0,1% prijevarnih transakcija, pružiti vrlo visoku točnost od 0,99. To je široko prepoznato u literaturi o otkrivanju prijevara, pa se stoga često koriste druge metrike performansi [8] [10] [11].

Mjere točnosti, osjetljivosti, preciznosti i F-mjera, poznate kao mjere temeljene na pragu [6], imaju poznata ograničenja zbog ovisnosti o pragovima odlučivanja koji su teški za odrediti u praksi i snažno ovise o kontekstu problema. One se često nadopunjuju mjernim veličinama površine *(engl. area under the curve, AUC)* ispod ROC *(engl. Receiver operating characteristic)* krivulje i prosječnom preciznošću *(engl. average precision, AP)*. Mjere AUC ROC i AP ciljaju na procjenu učinkovitosti za sve moguće pragove odlučivanja i nazivaju se mjere bez praga. Prag određuje potrebnu sigurnost modela u predikciji prije označavanja instance kao pozitivne ili negativne. Trenutno je AUC ROC de facto mjera za procjenu točnosti otkrivanja prijevara [6]. Međutim, nedavna istraživanja pokazala su da je ova mjera također zavaravajuća za procjenu problema s visokim neuravnoteženjem, kao što je otkrivanje prijevara, i preporučuje se korištenje preciznost-odziv *(engl. precision-recall, PR)* krivulje i mjere AP umjesto toga. [6]

U sljedećim potpoglavljima ove metrike su opisane zajedno sa njihovim prednostima i nedostatcima.

7.1. Matrica zabune

Matrica zabune (slika 5) je alat koji se koristi u problemima klasifikacije u strojnom učenju kako bi se vizualno predstavili rezultati testiranja modela na stvarnim podacima. Ova matrica pomaže u procjeni performansi modela klasifikacije analizirajući točno klasificirane primjere i one koji su pogrešno klasificirani. [12] Rezultat se klasificira kao pozitivan u slučajevima kada klasifikator previdi vrijednost iznad praga t koji je broj u otvorenom intervalu (0, 1).

Matrica zabune ima četiri osnovne komponente [12]:

- **True Positive (TP):** Broj primjera koji su ispravno klasificirani kao pozitivni (stvarno pripadaju toj klasi).
- **True Negative (TN):** Broj primjera koji su ispravno klasificirani kao negativni (stvarno ne pripadaju toj klasi).
- **False Positive (FP):** Broj primjera koji su pogrešno klasificirani kao pozitivni, iako

stvarno ne pripadaju toj klasi (lažno pozitivni rezultati).

- **False Negative (FN):** Broj primjera koji su pogrešno klasificirani kao negativni, iako stvarno pripadaju toj klasi (lažno negativni rezultati).

Tablica 5: Matrica konfuzije; vlastita izrada

Stvarno / Predviđeno	**Pozitivno**	**Negativno**
Pozitivno	*TP*	*FN*
Negativno	*FP*	*TN*

7.2. Mjere temeljene na pragu

Na temelju podataka iz matrice zabune, moguće je izračunati različite metrike performansi modela kao što su [12]:

- **Točnost *(engl. Accuracy)*:** $(TP + TN)/(TP + TN + FP + FN)$ - omjer ispravno klasificiranih primjera prema ukupnom broju primjera.
- **Osjetljivost *(engl. Sensitivity, Recall, True positive rate)*:** $TP/(TP+FN)$ omjer ispravno pozitivno klasificiranih primjera prema svim stvarno pozitivnim primjerima.
- **Preciznost *(engl. Precision)*:** $TP/(TP + FP)$ - omjer ispravno pozitivno klasificiranih primjera prema svim pozitivno klasificiranim primjerima.
- **F1 mjera *(engl. F1-score)*:** Harmonijska sredina preciznosti i osjetljivosti, često korištena metrika koja uzima u obzir oba aspekta performansi.

Točnost je intuitivna metrika koja mjeri ukupan postotak ispravno klasificiranih primjera. Međutim, treba je koristiti s oprezom, posebno u scenarijima s neravnoteženim klasama, jer visoka točnost može biti postignuta čak i ako model nudi neprihvatljivo nisku sposobnost identifikacije manje zastupljenih klasa. Točnost pruža sveobuhvatan pogled, ali može zanemariti suptilne performanse modela na pojedinim klasama.

Osjetljivost pruža dublji uvid u sposobnost modela da uhvati stvarne pozitivne primjere. Visoka osjetljivost sugerira da model uspješno prepoznaje veći udio stvarno pozitivnih primjera, minimizirajući lažno negativne rezultate (FN). Ova metrika je od značaja u situacijama gdje nedostatak identifikacije pozitivnih primjera može imati ozbiljne posljedice.

Preciznost je osjetljiva metrika koja daje uvid u sposobnost modela da identificira istinito pozitivne primjere. Visoka preciznost ukazuje na to da model rijetko označava negativne primjere kao pozitivne (FP), što je ključno u situacijama gdje lažno pozitivni rezultati imaju ozbiljne implikacije. Međutim, preciznost može biti zavaravajuća kada postoji niska stopa stvarnih pozitivnih primjera (niska osjetljivost), jer čak i mali broj lažno pozitivnih rezultata može smanjiti preciznost.

F1 mjera, kao harmonijska sredina preciznosti i osjetljivosti, nudi kompromis koji je posebno koristan kada je ravnoteža između ove dvije metrike presudna. Ova metrika je prikladna kada se suočava s neuravnoteženim klasama, gdje visoka preciznost može biti povezana s niskom osjetljivošću i obrnuto. F1 mjera naglašava potrebu za uravnoteženim performansama na obje strane klasifikacijskog spektra.

Sustav s visokom osjetljivošću i niskom preciznošću generira mnogo upozorenja, no većina tih upozorenja je netočna. Sustav s visokom preciznošću i niskom osjetljivošću je upravo suprotno; vraća vrlo malo upozorenja, ali većina tih upozorenja je ispravna. Idealni sustav s visokom preciznošću i visokom osjetljivošću generira mnogo upozorenja koja su ispravna. Za potpuno automatizirani sustav blokiranja poželjna je visoka preciznost. Za sustav s drugim slojem ljudske verifikacije visoka osjetljivost je korisna jer će lažno pozitivna upozorenja u svakom slučaju biti odbačena od strane istražitelja.

Iako se F1 mjera čini kao dobar izbor, nju je moguće izračunati tek kada je dostupna matrica zabune. Problem sa računanjem matrice zabune je taj da njeno računanje ovisi o pragu t koji je potrebno samostalno definirati, a to često nije lak zadatak.

7.3. Mjere bez praga

7.3.1. ROC krivulja

Tipična metrika performansi pri različitim postavkama praga za detekciju prijevara je AUC ROC. [8]. ROC je krivulja vjerojatnosti, a AUC predstavlja površinu ispod krivulje. Što je veći AUC, to je model bolji u razlikovanju između prijevarnih i autentičnih transakcija. [13]

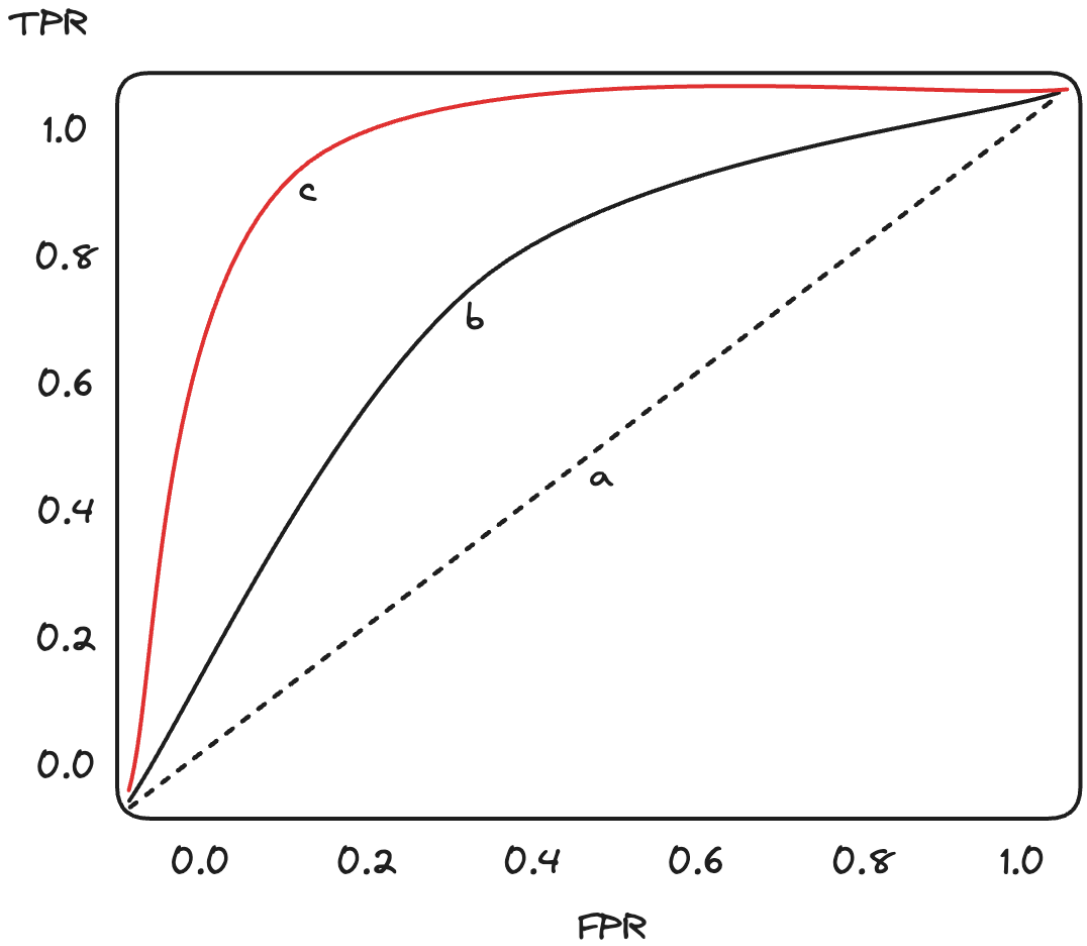
Na slici 6, ROC krivulja je nacrtana s TPR-om *(engl. True Positive Rate)* nasuprot FPR-u *(engl. False Positive Rate)*, gdje je TPR na y-osi, a FPR na x-osi.

$$TPR(osjetljivost) = TP/(TP + FN)$$

$$FPR = FP/(TN + FP)$$

Krivulja a predstavlja nasumični klasifikator sa AUC vrijednosti 0.5, AUC vrijednost krivulje b je približno 0.7, dok je AUC vrijednost krivulje c približno 0.9. Idealni klasifikator nalazio bi se u točki (0, 1) i predstavlja klasifikator koji nema lažno negativnih niti lažno pozitivnih predviđanja. Kaže se da je klasifikator a bolji od klasifikatora b ako se njegova krivulja u potpunosti nalazi iznad krivulje klasifikatora b. U slučajevima kada jedna krivulja ne dominira potpuno nad drugom, računa se AUC vrijednost. [6]

Izvrsni model ima AUC vrijednost blizu 1, dok slab model ima AUC vrijednost blizu 0. AUC vrijednost od 0.5 znači da model nema prediktivnu moć za razlikovanje između pozitivnih i negativnih primjera. To znači da je model jednako dobar kao nasumično odabiranje i nije sposoban donositi informirane odluke o klasifikaciji. Kada je AUC vrijednost 0.7, to znači da



Slika 6: ROC krivulja; vlastita izrada

postoji 70% vjerojatnosti da će model donijeti ispravnu odluku. [13]

ROC krivulje su dobre kako bi se stekao osjećaj za performanse klasifikatora preko cijelog raspona mogućih FPR-a. Međutim, njihova važnost za otkrivanje prijevara je ograničena jer je važan cilj otkrivanja prijevara održavanje vrlo niskog FPR-a zbog ograničenog broja istraga koje se u praksi mogu dnevno provoditi.

7.3.2. PR krivulja

PR krivulja, poznata i kao krivulja preciznosti-odziva u hrvatskoj literaturi ili precision-recall curve u stranoj literaturi, pruža korisne informacije o performansama klasifikatora u odnosu na različite pragove klasifikacije. Ona se konstruira iscrtavanjem TPR i osjetljivosti za različite pragove. Njena glavna prednost leži u sposobnosti da istakne klasifikatore koji istovremeno postižu visoku osjetljivost i visoku preciznost, odnosno visoki TPR i niski FPR. [14]

Metrika nazvana prosječna preciznost *(engl. average precision, AP)* često se koristi za kvantificiranje ukupnih performansi modela pri različitim pragovima. AP izračunava površinu ispod PR krivulje i pruža jednu vrijednost koja odražava sposobnost modela da uspostavi ravnotežu između preciznosti i odziva. AP se računa sljedećom formulom: [14]

$$\mathsf{AP} = \sum_{i=1}^{n} P(r_i) \cdot \Delta r_i$$

- n je broj različitih pozitivnih primjera u skupu podataka.
- $P(r_i)$ je preciznost za i-ti pozitivan primjer.
- Δr_i predstavlja promjenu osjetljivosti između i-tog i $(i+1)$-tog primjera. Drugim riječima, to je razlika u osjetljivosti između dva uzastopna pozitivna primjera.

Formula sumira produkt preciznosti i promjene osjetljivosti za svaki pozitivan primjer, čime se dobiva ukupna prosječna preciznost modela.

PR krivulje se razlikuju od ROC krivulja na nekoliko ključnih načina. Za razliku od ROC krivulja, PR krivulje nisu nužno monotone; preciznost može varirati kako se osjetljivost mijenja. Unatoč tome, postoji jedan-na-jedan odnos između PR i ROC krivulja te krivulja koja dominira u PR prostoru nužno dominira i u ROC prostoru. Važno je napomenuti da AUC za PR krivulje nema statističku interpretaciju kao AUC za ROC krivulje. [6]

U određenim scenarijima, poput detekcije prijevara, TPR za niske vrijednosti FPR je od ključne važnosti. U kontekstu detekcije prijevara broj kartica koje mogu biti ručno pregledane od strane istražitelja obično je vrlo ograničen. PR krivulje omogućuju precizniju procjenu performansi klasifikatora u takvim slučajevima.

Dok ROC krivulje pružaju vizualno atraktivan prikaz performansi klasifikatora u širokom rasponu specifičnosti, njihova interpretacija u kontekstu neravnoteženih skupova podataka može biti zavaravajuća. PR krivulje, s druge strane, pružaju gledatelju precizniju predikciju budućih performansi klasifikatora, jer evaluiraju udio pravih pozitivnih predviđanja među pozitivnim predviđanjima.

Iako su ROC krivulje najpopularnija metoda za evaluaciju binarnih klasifikatora, interpretacija ROC krivulja zahtijeva poseban oprez prilikom korištenja s neravnoteženim skupovima podataka.

Moguće je zaključiti da PR krivulje pružaju dublji uvid u performanse klasifikatora, posebno u situacijama s neravnoteženim podacima. Njihova sposobnost razdvajanja klasifikatora s visokom osjetljivošću i preciznošću ih čini važnim alatom u analizi binarnih klasifikacija, pogotovo u kontekstu problema ovog rada.

8. Odabir modela

Odabir odgovarajućih modela za detekciju prijevara u kartičnim transakcijama ključan je korak u osiguravanju uspješnog rješavanja ovog problema. Modeli koji su odabrani za ovaj rad su stablo odlučivanja, logistička regresija, slučajna šuma i XGBoost. Dio motivacije za baš ovim odabirom dolazi iz rada Priscille i Padme [10] koji su u svojem preglednom radu iz 2020. objavili tablicu učestalosti korištenja određenih tehnika strojnog učenja u znanstvenim radovima koji se bave detekcijom prijevara uz pomoć strojnog učenja. Upravo su odabrani modeli bili oni koji su bili najčešće korišteni.

8.1. Logistička regresija

Logistička regresija je statistička metoda koja se koristi u analizi podataka u kojoj je cilj opisati vezu izmedu zavisne varijable i jedne (ili više) nezavisnih varijabli. [15] Ova metoda se često koristi u kontekstu binarne klasifikacije, gdje se želi predvidjeti pripadnost nekog podatka jednoj od dvije moguće klase (npr. "da" ili "ne", "1" ili "0", "pozitivno" ili "negativno").

Matematički izraženo, logistička regresija modelira vjerojatnost da će se događaj pripisati jednoj od dvije klase kao logaritam omjera vjerojatnosti ($\log \frac{P}{1-P}$), gdje je P vjerojatnost da događaj pripada prvoj klasi, a $1 - P$ je vjerojatnost da pripada drugoj klasi. Model se obično izražava formulom [16]:

$$\log\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p$$

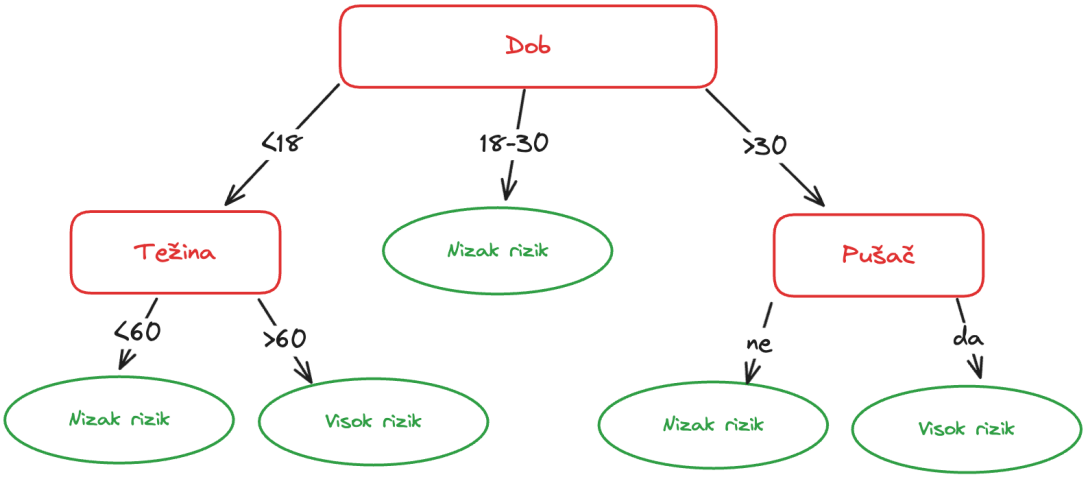
Gdje:

- P predstavlja vjerojatnost da događaj pripada prvoj klasi.
- $X_1, X_2, X_3, \ldots, X_p$ predstavljaju nezavisne varijable (atribute) koje se koriste za predviđanje vjerojatnosti.
- β_0 je odsječak na y-osi (intercept).
- $\beta_1, \beta_2, \beta_3, \ldots, \beta_p$ su koeficijenti koji se procjenjuju kako bi se odredila veza između nezavisnih varijabli i vjerojatnosti P.

Važno je napomenuti da izlaz logističke regresije nije stvarna vjerojatnost, već logaritam omjera vjerojatnosti. Da bi se dobila stvarna vjerojatnost, primjenjuje se sigmoidna funkcija (logistička funkcija) na izlaz modela:

$$P = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p)}}$$

Ova funkcija ograničava izlazni raspon između 0 i 1, što je idealno za modeliranje vjerojatnosti.



Slika 7: Stablo odlučivanja; vlastita izrada

8.2. Stablo odlučivanja

Stablo odlučivanja klasificira podatke postavljajući niz pitanja o značajkama koje su povezane s tim podacima. [17] Slika 7 prikazuje stablo odlučivanja koje daje odgovor na pitanje rizika od srčane bolesti za neku osobu. Svako pitanje sadržano je u čvoru, a svaki čvor pokazuje na jedno dijete za svaki mogući odgovor na svoje pitanje. Pitanja tako tvore hijerarhiju koje čine stablo. U najjednostavnijem obliku, postavljaju se pitanja na koja se može odgovoriti s "da" ili "ne", i svaki unutarnji čvor ima "da" dijete i "ne" dijete. Podatak se svrstava u određenu klasu tako da slijedi put od vršnog čvora, korijena, do čvora bez djece, lista, prema odgovorima koji se odnose na podatak koji se razmatra. Podatak se dodjeljuje klasi koja je povezana s listom na kojeg dolazi. Stabla odlučivanja su interpretabilnija od drugih klasifikatora jer kombiniraju jednostavna pitanja o podacima na razumljiv način. [17]

Stabla odlučivanja rastu dodavanjem čvorova pitanja postupno, koristeći označene primjere učenja kako bi usmjerili odabir pitanja. Idealno, jedno jednostavno pitanje bi savršeno podijelilo primjere učenja u njihove klase. Ako ne postoji pitanje koje omogućava takvo savršeno razdvajanje, bira se pitanje koje primjere razdvaja što je bolje moguće. [17]

8.3. Slučajna šuma

Iako pojedinačna stabla odlučivanja mogu biti izvrsni klasifikatori, često se postiže veća točnost kombiniranjem rezultata kolekcije stabala odlučivanja. Ansambli stabala odlučivanja ponekad su među najboljim vrstama klasifikatora. Slučajne šume i pojačavanje *(engl. boosting)* su dvije strategije za kombiniranje stabala odlučivanja. [17]

U pristupu slučajnih šuma, mnogo različitih stabala odlučivanja se gradi pomoću algoritma za nasumično izgrađivanje stabala. Skup podataka za treniranje se uzorkuje s ponavljanjem kako bi se stvorio modificirani skup za obuku iste veličine kao i originalni, ali s nekim elementima za treniranje koji su uključeni više puta. Osim toga, pri odabiru pitanja u svakom čvoru, razmatra se samo mali, slučajno odabrani podskup značajki. S ovim dvama modifika-

cijama, svako izvođenje može rezultirati malo drugačijim stablom. Predviđanja rezultirajućeg ansambla stabala odlučivanja kombiniraju se tako da se uzme najčešće predviđanje. Održavanje kolekcije dobrih hipoteza, umjesto obveze za jedno stablo, smanjuje šansu da će se novi primjer krivo klasificirati tako da mu se dodijeli pogrešna klasa od strane mnogih stabala. [17]

8.4. XGBoost

XGBoost (ekstremno pojačavanje gradijenta, engl. *extreme gradient boost*) je iznimno moćan i popularan algoritam za strojno učenje. [18] Algoritam pripada obitelji modela koji koriste metodu ansambla, što znači da se temelji na kombinaciji više slabih modela kako bi se stvorio snažniji model. XGBoost se ističe po svojoj izuzetnoj učinkovitosti, preciznosti i sposobnosti za rad s velikim skupovima podataka.

Osnovna ideja XGBoosta je stvaranje niza stabala odlučivanja, poznatih i kao pojačana stabla *(engl. boosted trees)*. Svako stablo se gradi iterativno, a svaka iteracija se usredotočuje na ispravljanje pogrešaka koje su napravljene u prethodnim stablima. Ova korekcija se postiže tako da se više pažnje posvećuje primjerima koji su bili pogrešno klasificirani u prethodnim iteracijama.

Jedna od ključnih karakteristika XGBoosta je njegova sposobnost rukovanja različitim vrstama značajki, uključujući numeričke, kategoričke i tekstualne. Također, algoritam ima ugrađenu podršku za rad s nedostajućim vrijednostima, što ga čini robusnim za stvarne skupove podataka.

XGBoost se također ističe po svojoj optimiziranoj implementaciji koja koristi tehnike kao što su pojačavanje gradijenta *(engl. gradient boosting)* i regularizacija *(engl. regularization)* kako bi se smanjila prekomjerna specijalizacija modela i povećala stabilnost. Osim toga, koristi tehniku obrezivanja *(engl. pruning)* stabla kako bi se osiguralo da stabla ne rastu prekomjerno, što pomaže u smanjenju potrošnje resursa i ubrzanju treniranja. [18]

Ovaj algoritam također omogućuje korisnicima da podešavaju različite hiperparametre kako bi postigli optimalne rezultate za svoj problem strojnog učenja. To uključuje parametre kao što su stopa učenja, dubina stabala, broj stabala u ansamblu i mnoge druge.

9. Izrada modela

Izrada modela strojnog učenja u ovom poglavlju uključuje četiri koraka:

- **Definiranje skupa za treniranje i skupa za testiranje:** Skup za treniranje predstavlja podskup transakcija koji se koristi za treniranje modela strojnog učenja. Skup za testiranje je podskup transakcija koji služi za procjenu performansi modela.
- **Skaliranje podataka**: Vrijednosti u podacima se skaliraju kako bi se numeričke vrijednosti standardizirale. Ovo pomaže pri pronalasku optimalnih parametara modela tijekom treniranja.
- **Treniranje modela:** Ovaj korak uključuje korištenje skupa za treniranje kako bi se podesili parametri modela. Za ovaj zadatak koristi se Python biblioteka scikit-learn koja nudi funkcije za treniranje modela.
- **Evaluacija performansi modela za predviđanje:** Performanse modela za predviđanje procjenjuju se pomoću skupa za provjeru (novi podaci).

9.1. Definiranje skupa za treniranje i skupa za testiranje

Za skup za treniranje koriste se transakcije od 1.3.2018. do 31.5.2018., a za skup za testiranje transakcije od 7.6.2018. do 14.6.2018. Transakcije skupa za testiranje odvijaju se tjedan dana nakon posljednje transakcije u skupu za treniranje s razlogom. U kontekstu otkrivanja prijevara, razdoblje koje razdvaja skupove za treniranje i testiranje naziva se razdoblje kašnjenja ili vremensko kašnjenje [8]. Kašnjenje uzima u obzir činjenicu da je u stvarnom sustavu za otkrivanje prijevara autentičnost transakcije poznata tek nakon pritužbe klijenta ili rezultata istrage. U ovom radu postavlja se pretpostavka da je autentičnost transakcije poznata točno tjedan dana nakon što su se dogodile. U praksi, kašnjenje može biti kraće kada korisnici brzo prijavljuju prijevare, ili puno duže u slučajevima gdje prijevare ostaju neotkrivene dulje vrijeme. Iz skupa za testiranje izbacuju se sve transakcije koje uključuju klijente koji su bili prevareni u skupu za treniranje. Na taj način se osigurava bolja procjena performansi modela i njegova sposobnost da otkriva nove prijevare. Isječak koda za razdvajanje skupova podataka prikazan je isječkom koda 9.

Nakon razdvajanja skupova, u skupu za treniranje se nalazi 891 720 transakcija, od kojih je 7 855 prijevara. Proporcija prijevara i autentičnih transakcija je 0.009. U skupu za testiranje nalazi se 26 654 transakcija, od kojih je 156 prijevara što daje proporciju od 0.006. Broj transakcija koje su označene kao prijevare po danu u testnom skupu prikazan je slikom 8. Prosječan broj prijevara u danu je 22.

```
def odvoji_skupove(transakcije, pocetni_datum_trening,
↪  zavrsni_datum_trening, pocetni_datum_test, zavrsni_datum_test):
    trening_skup = transakcije[
        (transakcije["datum_vrijeme"] >= pocetni_datum_trening) &
        (transakcije["datum_vrijeme"] <= zavrsni_datum_trening)
    ]

    prevareni_klijenti =
    ↪  set(trening_skup.loc[trening_skup["prijevara"] == 1,
    ↪  "id_kupac"])

    testni_skup = transakcije[
            (transakcije["datum_vrijeme"] >= pocetni_datum_test) &
            (transakcije["datum_vrijeme"] <= zavrsni_datum_test) &
            (~transakcije["id_kupac"].isin(prevareni_klijenti))
    ]


    trening_skup = trening_skup.sort_values("id_transakcija")
    testni_skup = testni_skup.sort_values("id_transakcija")

    return trening_skup, testni_skup

(trening_skup, testni_skup) = odvoji_skupove(
    transakcije,
    pocetni_datum_trening=datetime.datetime.strptime("1.3.2018.",
    ↪  "%d.%m.%Y."),
    zavrsni_datum_trening=datetime.datetime.strptime("31.5.2018.",
    ↪  "%d.%m.%Y."),
    pocetni_datum_test=datetime.datetime.strptime("7.6.2018.",
    ↪  "%d.%m.%Y."),
    zavrsni_datum_test=datetime.datetime.strptime("14.6.2018.",
    ↪  "%d.%m.%Y.")
)
```

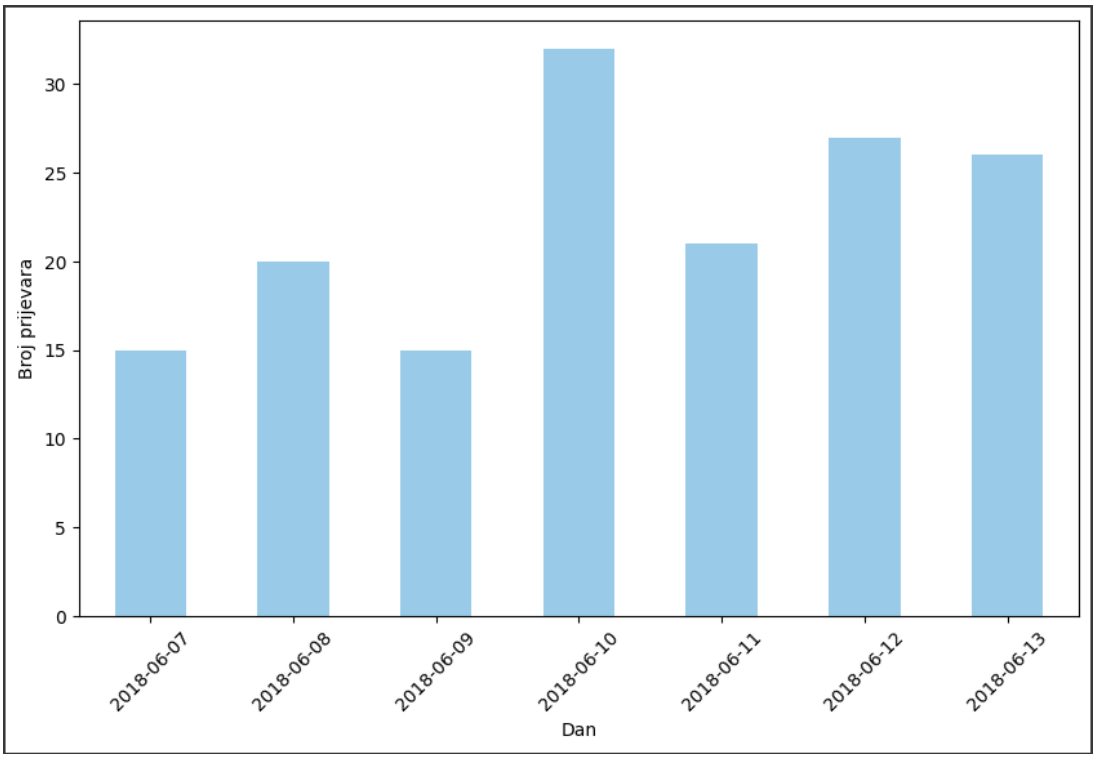
Isječak koda 9: Isječak programskog koda koji razdvaja skupove za treniranje i testiranje; vlastita izrada

9.2. Skaliranje podataka

Prije pristupanja treniranju modela, definiraju se parametri koji će biti ulazne vrijednosti modela strojnog učenja i parametar koji će se koristiti za predviđanje. Isječak koda 10 prikazuje taj postupak.

Nadalje, definira se pomoćna funkcija nazvana $skaliraj\_skupove$ koja se koristi za skaliranje određenih parametara u skupovima podataka za treniranje i testiranje. Funkcija je prikazana isječkom koda 11.

Prvo se inicijalizira objekt putem klase $StandardScaler$ iz biblioteke Scikit-learn. Ova



Slika 8: Broj transakcije koje su prijevara po danu u skupu podataka za testiranje; vlastita izrada

```
ulazni_parametri = [
  "iznos",
  "je_vikend",
  "je_noc",
  "id_kupac_br_transakcija_prozor_1d",
  "id_kupac_prosjecan_iznos_prozor_1d",
  "id_kupac_br_transakcija_prozor_7d",
  "id_kupac_prosjecan_iznos_prozor_7d",
  "id_kupac_br_transakcija_prozor_30d",
  "id_kupac_prosjecan_iznos_prozor_30d",
  "id_terminal_br_transakcija_prozor_1d",
  "id_terminal_rizik_prozor_1d",
  "id_terminal_br_transakcija_prozor_7d",
  "id_terminal_rizik_prozor_7d",
  "id_terminal_br_transakcija_prozor_30d",
  "id_terminal_rizik_prozor_30d",
]


izlazni_parametar = "prijevara"
```

Isječak koda 10: Definiranje ulaznih i izlaznih parametara za modele strojnog učenja; vlastita izrada

klasa pruža jednu od metoda skaliranja iz navedene biblioteke koja centrira i standardizira numeričke podatke tako da imaju srednju vrijednost 0 i standardnu devijaciju 1.

Pozivom $scaler.fit\_transform(trening\_skup[atributi])$ računa se standardna devijacija i srednja vrijednost parametara na odabranim parametrima u skupu podataka za treniranje. Nakon što su izračunati, svaki parametar se prilagođava formulom:

$$vrijednost\_parametra = (vrijednost\_parametra - srednja\_vrijednost)/standardna\_devijacija$$

Postupak se ponavlja sa skupom podataka za testiranje pozivom $scaler.transform$ nad skupom za testiranje, no ovoga puta se standardna devijacija i srednja vrijednost skupa ne računaju ponovno, već se koriste vrijednosti dobivene iz skupa za treniranje. Nad svim parametrima se primjenjuje prethodno navedena formula kako bi se podaci dosljedno transformirali.

Funkcija na kraju vraća dva skalirana skupa podataka, odnosno skalirani skup za treniranje i skalirani skup za testiranje. Ovi skalirani skupovi mogu se dalje koristiti u procesu strojnog učenja kako bi se osigurala bolja stabilnost i usporedivost između različitih parametara.

Skaliranje je često korisno pri radu s algoritmima strojnog učenja jer pomaže algoritmima da bolje konvergiraju i daje im bolje performanse.

```
def skaliraj_skupove(trening_skup, testni_skup, atributi):
    scaler = sklearn.preprocessing.StandardScaler()
    trening_skup[atributi] =
     ↪  scaler.fit_transform(trening_skup[atributi])
    testni_skup[atributi] = scaler.transform(testni_skup[atributi])

    return (trening_skup, testni_skup)
```

Isječak koda 11: Isječak programskog koda koji skalira skupove podataka za trening i testiranje; vlastita izrada

9.3. Treniranje modela

Kako bi model naučio korisne informacije, konačno se pristupa treniranju modela. Funkcija koja prima instancu klasifikatora iz biblioteke Scikit-learn, skupove za treniranje i testiranje te ulazne i izlazne parametre prikazana je isječkom koda 12. Skupovi za treniranje i testiranje prvo se skaliraju uz pomoć prethodno opisane funkcije za skaliranje skupova podataka. Zatim se definira rječnik, varijabla $rezultat$, u koji se akumuliraju rezultati treniranja. Model se trenira pozivom metode klasifikatora fit koja kao prvi argument prima skup za treniranje sa ulaznim parametrima. Drugi argument je ciljna varijabla, tj. oznake povezane s podacima za treniranje. Ova ciljna varijabla sadrži ishode ili vrijednosti koje model treba predvidjeti na temelju ulaznih parametara. Glavni zadatak modela je upravo predviđanje te ciljne varijable. Nadalje, u rječnik se sprema vrijeme koje je bilo potrebno za treniranje modela za kasniju analizu i vrše se predikcije nad skupovima za treniranje i testiranje. Rezultati se također spremaju u rječnik.

Nadalje, isječak koda 13 koristi funkciju $treniraj$ kako bi istrenirao nekoliko različitih modela strojnog učenja. Tu spadaju logistička regresija, stablo odlučivanja (jedno dubine tri

```
def treniraj(
    klasifikator,
    trening_skup,
    testni_skup,
    ulazni_parametri,
    izlazni_parametar
):
  (trening_skup, testni_skup) = skaliraj_skupove(trening_skup,
  ↪  testni_skup, ulazni_parametri)

  rezultat = {"klasifikator": klasifikator}

  pocetak = time.time()
  klasifikator.fit(trening_skup[ulazni_parametri],
  ↪  trening_skup[izlazni_parametar])
  rezultat["vrijeme_trening"] = time.time() - pocetak

  rezultat["predikcije_testni_skup"] =
  ↪  klasifikator.predict_proba(testni_skup[ulazni_parametri])[:, 1]

  rezultat["predikcije_trening_skup"] =
  ↪  klasifikator.predict_proba(trening_skup[ulazni_parametri])[:,
  ↪  1]

  return rezultat
```

Isječak koda 12: Funkcija koja trenira model; vlastita izrada

i jedno neograničene dubine), slučajna šuma i XGBoost. Rezultati treniranja svakog modela spremljeni su u rječnik.

9.4. Usporedba performansi

Isječak koda 14 prikazuje funkciju koja računa performanse klasifikatora istreniranih u prethodnom koraku. Funkcija $izracunaj\_performanse$ prima rezultate treniranja i skup podataka za testiranje kao svoje parametre. AUC ROC i prosječna preciznost se akumuliraju u varijablu $performanse$. Za svaki klasifikator se izrađuje novi okvir podataka koji je baziran na skupu podataka za testiranje. Dodaje mu se nova kolumna zvana predikcije. U toj kolumni se nalaze predikcije klasifikatora za svaki uzorak u skupu podataka za testiranje. Predikcija je broj u otvorenom intervalu (0, 1). Što je vrijednost veća, to je veća šansa da je transakcija prijevara. Taj skup podataka se zatim prosljeđuje funkciji roc_auc_score biblioteke Scikit-learn kako bi se izračunala AUC ROC vrijednost. Isti postupak se ponavlja za prosječnu preciznost uz pomoć funkcije $average\_precision\_score$. AUC ROC i prosječna preciznost se zaokružuju na tri decimale i akumuliraju se, uz vrijeme izvršavanja za treniranje, u varijablu $performanse$.

Nakon što je model istreniran, rezultati se prosljeđuju funkciji za računanje performansi. Rezultat je vidljiv u tablici 6. Poznato je da bi slučajni klasifikator dao AUC ROC vrijednost od

```
modeli = {
    "Logisticka regresija":
    ↪  sklearn.linear_model.LogisticRegression(random_state=0),
    "Stablo odluivanja - dubina=3":
    ↪  sklearn.tree.DecisionTreeClassifier(max_depth=3,
    ↪  random_state=0),
    "Stablo odluivanja - dubina=neogranicena":
    ↪  sklearn.tree.DecisionTreeClassifier(random_state=0),
    "Random forest":
    ↪  sklearn.ensemble.RandomForestClassifier(random_state=0,
    ↪  n_jobs=-1),
    "XGBoost": xgboost.XGBClassifier(random_state=0, n_jobs=-1),
}

rezultati = {}

for naziv_modela, model in modeli.items():
    rezultat = treniraj(
        model,
        trening_skup,
        testni_skup,
        ulazni_parametri,
        izlazni_parametar,
    )

    rezultati[naziv_modela] = rezultat
```

Isječak koda 13: Isječak programskog koda koji pokreče trening različitih klasifikatora; vlastita izrada

0.5 i prosječnu preciznost od 0.006 (udio prijevara u testnom skupu). Dobivene vrijednosti su znatno više, potvrđujući sposobnost klasifikatora da pruži mnogo bolje rezultate od slučajnog modela.

Klasifikatori koji koriste metode ansambla, slučajna šuma i XGBoost, očigledno pružaju bolje performanse od logističke regresije i stabla odlučivanja. Yousefi i sur. [19] navode da je logistička regresija široko korištena tehnika koja je često korištena u početnim studijama otkrivanja prijevara. Iako su metode dobro razumljive, jednostavne za implementaciju i imaju dobro uspostavljenu povijest u otkrivanju prijevara, imaju ograničenu snagu kada je riječ o obradi nelinearnih podataka, što ih čini neprikladnima za kompleksne probleme otkrivanja prijevara. Moglo bi se zaključiti da su ovi navodi istiniti jer slučajna šuma i XGBoost daju bolje rezultate, dok je algoritam ipak uspio nadmašiti rezultate stabla odlučivanja. Yousefi i sur. također potvrđuju ovakve rezultate o odnosu performansi algoritama logističke regresije i stabla odlučivanja. Iako se rezultati stabla odlučivanja daju lako interpretirati, mogu biti nestabilni i izrazito osjetljivi na neujednačene distribucije klasa što je ovdje slučaj.

Za detekciju prijevara u transakcijama važno je imati visoku točnost u identificiranju stvarnih prijevara (visoku osjetljivost) kako bi se smanjili lažno negativni rezultati. Također,

```
def izracunaj_performanse(rezultati, testni_skup):
  performanse = pd.DataFrame()

  for naziv_algoritma, rezultat in rezultati.items():
      predikcije = testni_skup
      predikcije["predikcije"] = rezultat['predikcije_testni_skup']

      aucroc = metrics.roc_auc_score(predikcije["prijevara"],
      ↪  predikcije["predikcije"]).round(3)
      ap = metrics.average_precision_score(predikcije["prijevara"],
      ↪  predikcije["predikcije"]).round(3)

      performanse_modela = pd.DataFrame([[aucroc, ap,
      ↪  round(rezultat["vrijeme_trening"])]], columns=['AUC ROC',
      ↪  'Average precision', "Vrijeme izvravanja (s)"])
      performanse_modela.index = [naziv_algoritma]

      performanse = pd.concat([performanse, performanse_modela])

  return performanse
```

Isječak koda 14: Isječak programskog koda koji računa AUC ROC i Average Precision; vlastita izrada

budući da postoji ograničenje broja istraga koje se dnevno mogu provesti, važno je minimizirati lažno pozitivne rezultate kako bi se smanjio broj neopravdanih istraga. Iz tog razloga, moguće je zaključiti da je XGBoost algoritam koji se čini najboljim za problem otkrivanja prijevara u transakcijama.

I na kraju, kao što je za pretpostaviti, vrijeme izvođenja za treniranje klasifikatora koji koriste metodu ansambla (slučajna šuma i XGBoost) znatno je duže nego za pojedinačne modele (stablo odlučivanja i logistička regresija).

Tablica 6: Performanse istreniranih klasifikatora nad testnim skupom podataka.
**Za dubinu=3.*
***Za neograničenu dubinu.*
Vlastita izrada.

Klasifikator	**AUC ROC**	**Prosječna preciznost**	**Vrijeme izvršavanja (s)**
Logistička regresija	0.833	0.602	2
Stablo odlučivanja*	0.768	0.503	3
Stablo odlučivanja**	0.806	0.354	33
Random forest	0.842	0.647	274
XGBoost	*0.851*	*0.666*	87

10. Zaključak

Ovaj rad donosi dublje razumijevanje problema spriječavanja prijevara te predstavlja arhitekturu sustava koji postiže visoku razinu pouzdanosti i točnosti u identifikaciji sumnjivih transakcija.

Istraživanje pokazuje da je pristup temeljen na strojnom učenju učinkovit za prepoznavanje raznovrsnih oblika prijevara s kreditnim karticama. Analiza različitih modela, kao što su stablo odlučivanja, logistička regresija, slučajna šuma i XGBoost, ukazuje na na njihove sposobnosti da se prilagode različitim tipovima prijevara i donesu precizne odluke.

Kroz poglavlje "Metrike performansi" istaknuto je da je važno uzeti u obzir više mjera evaluacije kako bi se pravilno procijenila učinkovitost sustava. Matrica zabune, ROC i PR krivulje pružaju cjelovit uvid u sposobnost sustava da razlikuje između autentičnih transakcija i sumnjivih transakcija.

Poboljšanje skupa podataka kroz generiranje profila kupaca, terminala te transakcija omogućilo je bolje treniranje modela i veću generalizaciju na stvarne slučajeve. Transformacije podataka, posebno one koje su se odnosile na vremenske i identifikatorske atribute, demonstrirale su važnost pripreme podataka za daljnju obradu.

Važnost istražitelja u optimizaciji sustava ne smije se zanemariti. Njihova sposobnost da dublje razumiju prirodu prijevara i pravilno konfiguriraju sustav igra ključnu ulogu u postizanju visokih performansi.

Ovaj diplomski rad doprinos je razumijevanju i razvoju sustava za otkrivanje prijevara s kreditnim karticama korištenjem strojnog učenja. Rad otvara vrata daljnjim istraživanjima i unaprjeđenjima te naglašava važnost kontinuirane optimizacije i prilagodbe kako bi se sustav uspješno nosio s evoluirajućim prijetnjama u svijetu digitalnih transakcija.

Popis literature

[1] T. Carneiro, R. V. M. Da Nóbrega, T. Nepomuceno, G.-B. Bian, V. H. C. De Albuquerque i P. P. Reboucas Filho, „Performance analysis of google colaboratory as a tool for accelerating deep learning applications," *IEEE Access*, sv. 6, str. 61 677–61 685, 2018.

[2] T. N. R. Organization. „2023 credit card fraud report," Nilson Report. (2022.), adresa: https://nilsonreport.com/newsletters/1232/ (pogledano 5. 7. 2023.).

[3] J. Egan. „Credit card fraud statistics," Bankrate. (12. 1. 2023.), adresa: https://www.bankrate.com/finance/credit-cards/credit-card-fraud-statistics/ (pogledano 5. 7. 2023.).

[4] „Consumer sentinel network," Federal Trade Commission. (2022.), adresa: https://www.ftc.gov/system/files/ftc_gov/pdf/CSN%20Annual%20Data%20Book%202021%20Final%20PDF.pdf (pogledano 5. 7. 2023.).

[5] J. Akhilomen, „Data mining application for cyber credit-card fraud detection system," *Advances in Data Mining. Applications and Theoretical Aspects*, P. Perner, ur., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013., str. 218–228, ISBN: 978-3-642-39736-3.

[6] Y.-A. Le Borgne, W. Siblini, B. Lebichot i G. Bontempi, *Reproducible Machine Learning for Credit Card Fraud Detection - Practical Handbook*. Université Libre de Bruxelles, 2022.

[7] „Sixth report on card fraud," European Central Bank. (2020.), adresa: https://www.ecb.europa.eu/pub/cardfraud/html/ecb.cardfraudreport202008~521edb602b.en.html (pogledano 5. 7. 2023.).

[8] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi i G. Bontempi, „Credit card fraud detection: A realistic modeling and a novel learning strategy," *IEEE transactions on neural networks and learning systems*, sv. 29, br. 8, str. 3784–3797, 2017.

[9] I. Mekterović, L. Brkić i M. Baranović, „A systematic review of data mining approaches to credit card fraud detection," *WSEAS Trans. Bus. Econ*, sv. 15, str. 437, 2018.

[10] C. V. Priscilla i D. P. Prabha, „Credit card fraud detection: A systematic review," *Intelligent Computing Paradigm and Cutting-edge Technologies: Proceedings of the First International Conference on Innovative Computing and Cutting-edge Technologies (ICICCT 2019), Istanbul, Turkey, October 30-31, 2019 1*, Springer, 2020., str. 290–303.

[11] C. Elkan, „The foundations of cost-sensitive learning," *International joint conference on artificial intelligence*, Lawrence Erlbaum Associates Ltd, sv. 17, 2001., str. 973–978.

[12] D. Valero-Carreras, J. Alcaraz i M. Landete, „Comparing two SVM models through different metrics based on the confusion matrix," *Computers & Operations Research*, sv. 152, str. 106 131, 2023.

[13] S. Narkhede, „Understanding auc-roc curve," *Towards Data Science*, sv. 26, br. 1, str. 220–227, 2018.

[14] M. Zhu, „Recall, precision and average precision," *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, sv. 2, br. 30, str. 6, 2004.

[15] A. Šarlija, „Logistička regresija u predikciji odljeva korisnika mobilne igre," disertacija, University of Split. University of Split, Faculty of science. Department of . . ., 2022.

[16] S. Sperandei, „Understanding logistic regression analysis," *Biochemia medica*, sv. 24, br. 1, str. 12–18, 2014.

[17] C. Kingsford i S. L. Salzberg, „What are decision trees?" *Nature biotechnology*, sv. 26, br. 9, str. 1011–1013, 2008.

[18] S. R. „Boost your machine learning with xgboost!" LinkedIn. (2023.), adresa: `https://www.linkedin.com/pulse/boost-your-machine-learning-xgboost-santhiya-r/` (pogledano 1. 9. 2023.).

[19] N. Yousefi, M. Alaghband i I. Garibay, „A Comprehensive Survey on Machine Learning Techniques and User Authentication Approaches for Credit Card Fraud Detection," *International Journal of Computer and Information Engineering*, sv. 15, br. 11, str. 599–612, 2021.

Popis slika

1. Evolucija ukupne vrijednosti kartičnih prijevara korištenjem kartica izdanih unutar SEPA-e (lijeva skala: ukupna vrijednost prijevare (milijuni eura); desna skala: vrijednost prijevare kao udio u vrijednosti transakcija (postoci)); prema [7] 4

2. Shema koja prikazuje arhitekturu sustava za otkrivanje prijevara. Fokus ovog rada je na modelu vođenom podacima; prema [8] 6

3. Strojno učenje za CCFD, skica osnovne metodologije; prema [6] 9

4. Vizualizacija terminala u radijusu kupca; vlastita izrada 15

5. Kupci sa pridruženim terminalima u radijusu; vlastita izrada 15

6. ROC krivulja; vlastita izrada 26

7. Stablo odlučivanja; vlastita izrada 29

8. Broj transakcije koje su prijevara po danu u skupu podataka za testiranje; vlastita izrada 33

Popis tablica

1. Gubici uzrokovani kartičnim prijevarama diljem svijeta; prema [2] 3

2. Primjer kupaca generiranih za skup podataka. PTPD predstavlja prosjek transakcija po danu. Vlastita izrada 13

3. Primjer terminala generiranih za skup podataka; vlastita izrada 14

4. Primjer transakcija generiranih za skup podataka; vlastita izrada 17

5. Matrica konfuzije; vlastita izrada 24

6. Performanse istreniranih klasifikatora nad testnim skupom podataka. **Za dubinu=3. **Za neograničenu dubinu.* Vlastita izrada. 38

Popis isječaka koda

1. Isječak programskog koda koji generira profile kupaca; prema [6] 13
2. Isječak programskog koda koji generira profile terminala; prema [6] 14
3. Isječak programskog koda koji generira profile terminala; prema [6] 16
4. Isječak programskog koda koji generira profile terminala; prema [6] 19
5. Isječak programskog koda koji generira prijevare; prema [6] 20
6. Isječak programskog koda koji generira atribute je_vikend i je_noc; prema [6] . . 21
7. Isječak programskog koda koji generira karakteristike kupca; prema [6] 21
8. Isječak programskog koda koji obavlja transformacije terminala; prema [6] 22
9. Isječak programskog koda koji razdvaja skupove za treniranje i testiranje; vlastita izrada 32
10. Definiranje ulaznih i izlaznih parametara za modele strojnog učenja; vlastita izrada 33
11. Isječak programskog koda koji skalira skupove podataka za trening i testiranje; vlastita izrada 34
12. Funkcija koja trenira model; vlastita izrada 35
13. Isječak programskog koda koji pokreče trening različitih klasifikatora; vlastita izrada 36
14. Isječak programskog koda koji računa AUC ROC i Average Precision; vlastita izrada 37