Detekcija lažnih blockchain računa primjenom algoritama strojnog učenja

Mustač, David

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Sveučilišni diplomski studij elektrotehnike

Diplomski rad

DETEKCIJA LAŽNIH BLOCKCHAIN RAČUNA PRIMJENOM ALGORITAMA STROJNOG UČENJA

Rijeka, siječanj 2024

David Mustač

0069080003

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Sveučilišni diplomski studij elektrotehnike

Diplomski rad

DETEKCIJA LAŽNIH BLOCKCHAIN RAČUNA PRIMJENOM ALGORITAMA STROJNOG UČENJA

Mentor: prof.dr.sc Zlatan Car

Komentor: v. asist. dr. sc. Nikola Anđelić

Rijeka, siječanj 2024

David Mustač

0069080003

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 12. rujna 2023.

Zavod: **Zavod za automatiku i elektroniku**
Predmet: **Primjena umjetne inteligencije**
Grana: **2.03.06 automatizacija i robotika**

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **David Mustač (0069080003)**
Studij: Sveučilišni diplomski studij elektrotehnike
Modul: Automatika

Zadatak: **Detekcija lažnih blockchain računa primjenom algoritama strojnog učenja**

Opis zadatka:

Napraviti pregled postojećih istraživanja detekcije lažnih blockchain računa primjenom algoritama umjetne inteligencije. Razviti metodu za prikupljanje skupa podataka te primijeniti različite metode za pred obradu skupa podatak koji će se koristiti u algoritmima strojnog učenja. Na prikupljenom skupu podataka napraviti detaljnu statističku analizu. Izvršiti treniranje različitih algoritama strojnog učenja primjenom unakrsne validacije s nasumičnim pretraživanjem hiperparametera. Nakon treniranja algoritama strojnog učenja napraviti procjenu koji od algoritama ima najveću točnost detekcije i odabrane algoritme koristiti za izradu ansambla metode u cilju povećanja točnosti detekcije.

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

Zadatak uručen pristupniku: 20. ožujka 2023.

Mentor:

Prof. dr. sc. Zlatan Car

Dr. sc. Nikola Anđelić (komentor)

Predsjednik povjerenstva za diplomski ispit:

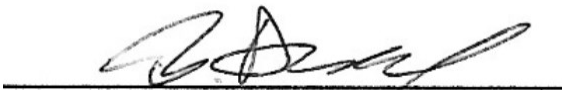
Prof. dr. sc. Dubravko Franković

IZJAVA

"Sukladno članku 9. Pravilnika o diplomskom radu, diplomskom ispitu i završetku diplomskih sveučilišnih studija Tehničkog fakulteta Sveučilišta u Rijeci izjavljujem da sam izradio ovaj diplomski rad samostalno, koristeći vlastito znanje i navedenu literaturu, u razdoblju od datuma zadavanja zadatka do datuma predaje."

Rijeka, siječanj 2024

David Mustač



ZAHVALA

Zahvaljujem se mentoru prof. dr. sc. Zlatanu Caru na pruženoj prilici izrade ovog diplomskog rada. Posebno se zahvaljujem v. asist. dr. sc. Nikoli Anđeliću na izdvojenom vremenu, trudu i pomoći pri izradi diplomskog rada.

Također, zahvaljujem se svojoj obitelji i prijateljima na podršci i potpori tijekom svih godina studiranja.

Sadržaj

1. Uvod

U današnjem informacijskom dobu, blockchain tehnologija se ističe kao iznimno značajna inovacija koja donosi duboke promjene u načinu na koji se obavljaju financijske transakcije i upravlja podacima putem decentraliziranog digitalnog identita. Ova tehnologija omogućuje izravne peer-to-peer transakcije bez posrednika, istovremeno jamčeći sigurnost, transparentnost i decentralizaciju. Međutim, iako donosi brojne prednosti, blockchain tehnologija nije imuna na izazove i rizike koji prijete njezinoj cjelovitosti i pouzdanosti.

Jedan od najozbiljnijih izazova s kojima se suočava blockchain tehnologija je prijevara. Prijevara predstavlja ozbiljnu prijetnju integritetu i reputaciji blockchain sustava, s potencijalom za uzrokovati financijske gubitke, narušavanje povjerenja sudionika te pravne implikacije. Prijevara može uzeti različite oblike i manifestirati se na različitim razinama blockchain mreža, uključujući dvostruke transakcije, neovlaštene promjene transakcija, manipulaciju pametnim ugovorima, krađu identiteta i pranje novca.

Kako bi se suočili s ovim izazovima i očuvali integritet blockchain tehnologije, postoji nužnost razvijanja efikasnih i učinkovitih metoda za otkrivanje i prevenciju prijevarnih aktivnosti. U ovom kontekstu, algoritmi strojnog učenja predstavljaju moćan alat koji može analizirati i izvlačiti uvide iz obilja podataka dostupnih na blockchain mrežama. Ovi algoritmi imaju sposobnost identifikacije obrazaca i ponašanja u transakcijama i računima, otkrivanja anomalija te razlikovanja lažnih i legitimnih slučajeva. Dodatno, integracija algoritama strojnog učenja s blockchain tehnologijom može poboljšati sigurnost i robusnost samog sustava.

U nastavku ovog diplomskog rada, istražit će se primjena algoritama strojnog učenja za detekciju i sprječavanje prijevarnih aktivnosti na blockchain mrežama. Analizirat će se različiti oblici prijevare koji prijete blockchain tehnologiji te kako algoritmi strojnog učenja mogu doprinijeti očuvanju njezine integritete.

Također će se razmotriti mogući izazovi i ograničenja u primjeni ovih tehnika te pružiti pregled relevantnih istraživanja i pristupa u ovom području. To će se postići na način da će se dati pregled postojećih istraživanja na temu detekcije lažnih blockchain računa putem algoritama strojnog učenja, razviti metodu za prikupljanje skupa podataka te primjeniti različite metode za pred obradu skupa podataka koji će se koristiti u algoritmima strojnog učenja.

Nadalje, potrebno je napraviti detaljnu statističku analizu na prikupljenom skupu podataka i izvršiti treninranje različitih algoritama strojnog učenja primjenom unakrsne validacije s nasumičnim pretraživanjem hiperparametara. Nakon treniranja algoritma napraviti će se procjena koji od algoritama daje najveću točnost detekcije i odabrani algoritam koristiti za izradu ansambla metode u cilju povećanja točnosti detekcije.

Ovaj rad ima za cilj doprinijeti boljem razumijevanju važnosti primjene algoritama strojnog učenja za sigurnost blockchain tehnologije i pružiti smjernice za buduća istraživanja i razvoj u ovoj sferi.

2. Pregled literature

Strojno učenje moćan je alat koji se može koristiti za otkrivanje lažnih transakcija. Analizirajući velike količine podataka, algoritmi strojnog učenja mogu identificirati obrasce i anomalije koji mogu ukazivati na prijevaru. To može pomoći tvrtkama da spriječe prijevare i zaštite svoje klijente. Sve je više istraživanja o otkrivanju lažnih transakcija s ML-om. Predloženo je mnoštvo različitih metoda, a područje se još uvijek razvija.

Jedan od najranijih radova o otkrivanju prijevara temeljenom na ML-u bio je Hissu Hyvarinen. Usredotočili su se na otkrivanje porezne prijevare analizom međunarodnih transakcija. Koristili su različite ML algoritme, uključujući vektorske strojeve podrške (engl. Support vector machine) i nasumične šume (engl. Random forest), kako bi identificirali transakcije za koje je vjerojatno da su proizašle iz prijevara. Još jedan rani rad na otkrivanju prijevare temeljenom na ML-u bio je Michał Ostapowicz. Usredotočili su se na otkrivanje lažnih bankovnih računa analizom podataka o transakcijama i karakteristika računa. Koristili su različite ML algoritme, uključujući nasumične šume i XGBoost, kako bi identificirali račune za koje je vjerojatno da su lažni.

Posljednjih godina raste interes za korištenje dubokog učenja za otkrivanje prijevara. Algoritmi dubokog učenja vrsta su ML algoritama koji mogu naučiti složene obrasce iz podataka. Algoritmi dubokog učenja pokazali su se vrlo učinkovitima u otkrivanju lažnih transakcija. Na primjer, Steven Farrugia predložio je model dubokog učenja za otkrivanje nezakonitih računa koji su povezani sa prijevarama na Ethereum blockchainu. Njihov model uspio je otkriti nezakonite račune s točnošću od preko 99%.

T. Pham predložio je model dubokog učenja za otkrivanje sumnjivih transakcija u Bitcoin mreži. Njihov je model uspio otkriti nenormalne transakcije s točnošću od preko 95%.

Kod izrade modela strojnog učenja koristili su nekoliko skupova podataka. Od javno dostupnih koristili su Ethereum Fraud Detection dataset [8] dok su ostali bili privatni, prikupljenih sa njihove strane.

3. Definicija prijevara u blockchain ekosustavu

Prijevara na blockchain mreži je svaka vrsta kriminalne aktivosti koja uključuje korištenje blockchain tehnologije, a to obuhvaća sljedeće aktivnosti [1]:

- **Pranje novca**: korištenje blockchain tehnologije za pranje novca stečenog ilegalnim radnjama,
- **Prijevare sa kriptovalutama**: prijevare ljudi prodajom lažnih ili bezvrijednih kriptovaluta,
- **Krađa kriptovaluta**: hakiranje mjenjačnica kriptovaluta ili krađa kriptovaluta iz novčanika korisnika
- **Izlazne prijevare** (**engl**. Exit scams): pokretanje kripto projekta i zatim njegovo napuštanje, ostavljajući ulagače sa bezvrijednim tokenima, i
- **ICO prijevare (engl**. ICO scams): prijevare sa početnom ponudom kriptovaluta uključuju prikupljanje novca od ulagača prodajom token koji im obećava pristup novoj kriptovaluti ili blockchain platformi, pri čemu je ICO lažan čime investitori gube novac.

Stoga je bitno otkriti lažne blockchain račune koji se mogu iskoristiti za izvođenje raznih kriminalnih aktivnosti kako bi se očuvala sigurnost i integritet blockchain ekosustava.

4. Financijske malverzacije unutar kripto ekosustava

Ekosustav kriptovaluta prostor je koji se brzo, a nove tehnologije i aplikacije se stalno pojavljuju. Iako je ova inovacija donijela mnoge prednosti, stvorila je i nove prilike za financijske malverzacije.

Pranje novca jedna je od najčešćih vrsta financijskih malverzacija u kripto ekosustavu. Uključuje obradu sredstava stečenih nekom vrstom prijevara ili kriminalnih radnji kako bi se prikrilo njihovo nezakonito podrijetlo. To se uglavnom postiže zlouporabom inače legitimnih alata, kao što su kriptovalute za privatnost i usluge pretvaranja gotovine u kripto. Međutim, perači novca također iskorištavaju darknet tržišta i usluge kibernetičkog kriminala, stvarajući višestruki učinak na ukupne nezakonite aktivnosti.

Usluge kriptovalute privlačne su peračima novca zbog mnogih istih čimbenika koje cijene redoviti potrošači: brzi prijenosi, pseudoanonimnost i praktičnost. Kriminalci su zainteresirani za prikupljanje sredstava putem nepovratnog formata koji je gotovo trenutan. Pranje novca koje uključuje kriptovalute uglavnom slijedi isti put kao i konvencionalni pandan, počevši od plasmana, nakon čega slijedi raslojavanje i integracija [2].

Faza plasiranja novca u kripto ekosustav uključuje uvođenje sredstava stečenih kriminalom u sustav kriptovaluta. To se može učiniti različitim metodama koje će biti razrađene u sljedećim poglavljima.

4.1 Plasman novca u kripto ekosustav

Plasman je prva faza pranja novca, a uključuje uvođenje prihoda stečenog kriminalnom radnjom u financijski sustav. U kontekstu kriptovaluta, plasman se može postići na više načina, uključujući:

- kupnja kriptovaluta s fiat valutom putem razmjene gotovine za kripto,
- polaganje kriptovaluta dobivenih krađom, iznudom ili drugom kriminalnom radnjom u mjenjačnicu kriptovaluta, i
- korištenje usluge treće strane ili VASP-a (davatelja usluge virtualne imovine) za pretvaranje kriptovaluta u fiat valutu ili obrnuto.

Konkretna metoda postavljanja koja se koristi ovisit će o vrsti predikatnog zločina i preferencijama perača novca. Na primjer, počinitelji prijevara često koriste razmjenu gotovine u kripto kako bi plasirali svoje prihode, dok počinitelji ransomwarea obično traže od žrtava da koriste uslugu treće strane (engl. Third-party services) ili VASP za plaćanje.

4.1.3 Cash-to-Crypto

Jedna od najbržih metoda pretvaranja fiat valute u kriptovalutu i obrnuto je putem usluga gotovine u kripto (engl. Cash-to-Crypto). Među tim uslugama najviše se koriste kripto bankomati[3]. Ovi automatizirani kiosci omogućuju korisnicima umetanje novčanica, kupnju kriptovalute i prijenos izravno u novčanik bez potrebe za razmjenom ili čak bankovnim računom. U svijetu postoji više od 30 000 kripto bankomata, a više od 90% njih nalazi se u Sjevernoj Americi.

Iako usluge pretvaranja gotovine u kripto, uključujući kripto bankomate, nisu nezakonite, one mogu biti privlačna opcija plaćanja za kibernetičke kriminalce i druge nezakonite aktere. Prema istraživanju koje je proveo TRM Labs, preko 40 milijuna USD prebačeno je na poznate adrese prijevara putem usluga gotovine u kripto 2022.

4.1.2 Davatelji usluge virtualne imovine

Davatelji usluga virtualne imovine (engl. Virtual Asset Service Provider) ovise o infrastrukturi veće burze za pružanje usluga trgovanja digitalnom imovinom korisnicima, često bez svijesti ili autorizacije centralne burze. Protupravni akteri i pojedinci koji podliježu sankcijama mogu iskoristiti parazitske VASP-ove za prijenos svoje nelegitimne dobiti kroz ekosustav kriptovaluta, čime se transakcije čine zakonitima. Parazitske mjenjačnice obično pokazuju slabe ili nepostojeće zahtjeve Know-Your-Customer (KYC) i Anti-Money Laundering (AML), što ih čini omiljenim kanalom za prijenos sredstava kibernetičkih kriminalaca i perača novca.

Prema istraživanju koje je proveo TRM Labs, parazitne mjenjačnice omogućuju značajno više nedopuštenih aktivnosti na blockchainu u usporedbi s njihovim uobičajenim pandanima, s potencijalom da dosegnu do 100 puta više. Znatan dio ovog nezakonitog volumena, koji prelazi polovicu, pripisuje se sredstvima povezanim sa sankcioniranim subjektima.

Ova prevalencija može se djelomično pripisati činjenici da se vjeruje da se gotovo dvije trećine parazitnih mjenjačnica nalazi u Rusiji i Iranu. Osobito su iranske burze bile suočene sa sankcijama zbog svoje jurisdikcije. Naime, SUEX, kripto mjenjačnicu i OTC brokera, sankcionirao je OFAC 2021. jer je djelovao kao parazitska mjenjačnica. Utvrđeno je da je suučesnik u pranju milijuna dolara za ruske ransomware skupine [3].

4.1.4 Procesori plaćanja kriptovalutama

Procesori plaćanja kriptovalutama (engl. Payment Processors) legitimni su subjekti koji nude pomoć pojedincima i tvrtkama u prihvaćanju kriptovalute kao oblika plaćanja. Ti procesori olakšavaju stvaranje adresa za plaćanje za korisnike i pružaju usluge koje im omogućuju izravno primanje plaćanja putem vlastitih web stranica, obično putem API-ja. U zamjenu za njihove usluge naplaćuje se mali postotak vrijednosti transakcije. Međutim, važno je napomenuti da kriminalci mogu iskoristiti procesore plaćanja u svrhu pranja novca, osobito u fazama postavljanja i slojevitosti. Zbog svojih relativno labavih propisa, ti procesori često imaju minimalne ili nikakve zahtjeve za Know Your Customer (KYC). Dopuštajući korisnicima da

generiraju nove adrese za svaku uplatu ili čak ponovno koriste adrese za različite pojedince, procesori plaćanja mogu istražiteljima zakomplicirati praćenje tokova sredstava.

4.1.4 OTC kripto trgovanje

OTC kripto trgovanje (engl. Over the Counter) je metoda trgovanja kriptovalutama izravno između dvije strane, bez korištenja mjenjačnice. U OTC trgovanju, kupac i prodavač pregovaraju o uvjetima trgovine, uključujući cijenu i količinu kriptovalute kojom se trguje. Trgovina se zatim namiruje izvan burze.

OTC kripto trgovanje nudi nekoliko prednosti u odnosu na trgovanje na burzi, uključujući

- trgovanje velikim količinama,
- privatnost, i
- dogovorene cijene.

OTC kripto trgovanje obično koriste institucionalni ulagači, poput hedge fondova i tvrtki rizičnog kapitala. OTC trgovanje također koriste pojedinci visoke neto vrijednosti koji žele trgovati velikim količinama kriptovalute.

4.2 Raslojavanje

Raslojavanje (engl. Layering) je strateška metoda čiji je cilj povećati složenost praćenja nezakonite imovine podvrgavanjem nizu transakcija i korištenjem različitih alata. Mikseri, mostovi, usluge zamjene i spajanja kriptovaluta, koji uključuju više pošiljatelja koji kombiniraju sredstva kako bi prikrili njihovo podrijetlo, obično se koriste u svrhu slojevitosti.

Te su tehnike posebno osmišljene kako bi poboljšale privatnost i ometale istražitelje u njihovim nastojanjima da uđu u trag kretanju sredstava. Dok neki pojedinci mogu jednostavno usmjeriti sredstva na burze za brzo unovčavanje, sofisticirani perači novca mogu koristiti programske tehnike za pranje novca [5].

Kako bi se učinkovito borili protiv pranja novca, od ključne je važnosti da istražitelji posjeduju modele znanosti o podacima koji mogu identificirati različite obrasce pranja novca, poznate kao potpisi. Osim toga, sposobnost demiksiranja transakcija iz miksera i njihovog automatskog praćenja kroz međulančane mostove nezamjenjiva je vještina za istražitelje pranja novca.

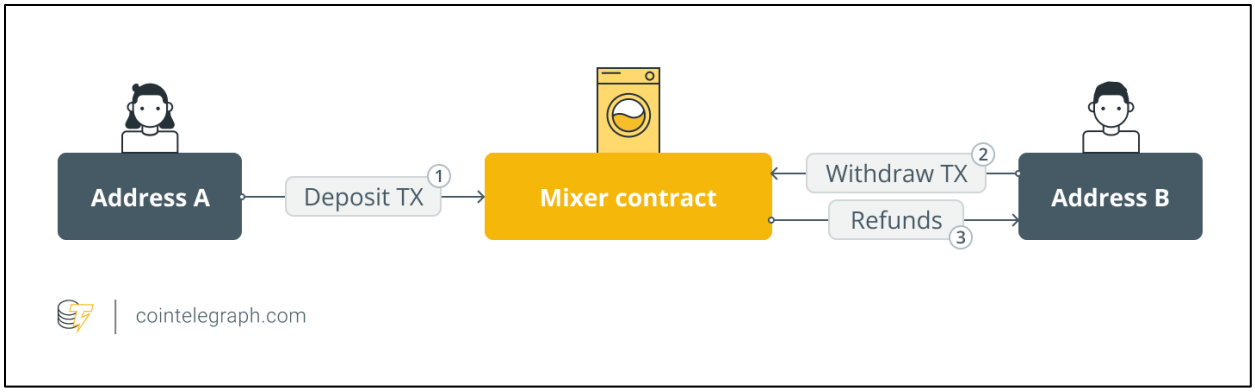
4.3 Mikseri

Kripto mikseri (engl. Mixers/Tumblers) funkcioniraju tako da kombiniraju kriptovalute od više korisnika u jedan skup i zatim ih redistribuiraju na različite adrese što otežava postupak praćenja izvornog izvora sredstava. Na primjer, Bitcoin explorer, koji prati sve BTC transakcije, pokazat će da je pojedinac A prebacio Bitcoin u mikser i da je pojedinac B primio BTC iz miksera. Ovaj proces osigurava da nitko ne zna tko je poslao BTC i kome, čime se peru „prljavi“ Bitcoini u procesu kriptomješanja.

Kripto mikseri rade kombiniranjem vaše kriptovalute s velikim fondom druge kriptovalute prije nego što vrate manje jedinice kriptovalute na adresu po vašem izboru, uz odbitak od 1-3% od ukupnog iznosa koji ste položili.

Tvrtka za miješanje novčića obično ostvaruje profit od 1-3%, što je njihov izvor prihoda. Miješanje kriptovaluta je slično pranju novca, što je kazneno djelo. Međutim, sudjelovanje u miješanju novčića ne znači nužno da netko čini kazneno djelo. Zanimljivo je da mikseri nisu ilegalni, niti se koriste isključivo za nedopuštene radnje. Na primjer, mnogi se mikseri promoviraju kao sredstvo za poboljšanje privatnosti i anonimnosti na internetu.

Prema Nacionalnoj procjeni rizika od pranja novca Ministarstva financija SAD-a za 2022., mikseri "olakšavaju kriminalcima prikrivanje kretanja ili podrijetla sredstava, stvarajući tako dodatne prepreke za istražitelje".



Slika 4.1. Semantički prikaz neskrbničkih miksera na Ethereum mreži [1]

Korisnici mogu potvrditi ugovoru o mješanju da su položili sredstva bez izlaganja transakcije depozita koju su izdali korištenjem jedne od nekoliko dostupnih kriptografskih tehnika kao što su prstenasti potpisi (engl. Ring signatures) i zk-SNARK-ovi u transakciji povlačenja.

5. Spriječavanje prijevara

Sprječavanje lažnih blockchain računa ključno je za rast i razvoj blockchain ekosustava. Koraci koji su se poduzeli u smislu očuvanja integriteta su sljedeći:

- Implementacija jakih procedura protiv pranja novca(engl. AML/KYC) koji su dizajnirani za provjeru indentiteta korisnika, međutim tim postupkom djelomično narušava osnovno svojstvo blokchaina pseudoanonimnost. Te postupke bi trebale provoditi mjenjačnice kriptovaluta i sve tvrtke koje posluju istima.
- Algoritmi strojnog učenja mogu se koristiti za otkrivanje lažnih blockchain računa analizom uzoraka u podacima o transakcijama i ponašanju korisnika. Mjenjačnice kriptovaluta i druge tvrtke trebale bi koristiti strojno učenje za označavanje sumnjivih računa radi daljnje istrage.
- Mjenjačnice kriptovaluta i druge tvrtke trebale bi pratiti transakcije u potrazi za neobičnim uzorcima, kao što su velike količine transakcija s jednog računa ili transakcije na poznate nedopuštene račune(engl. Blacklisted accounts)

- Korisnici bi trebali biti upoznati rizicima prijevare i kako se zaštititi. To uključuje podučavanje korisnika kako prepoznati prijevare i kako zaštititi svoje privatne ključeve.

Postoji niz tvrtki koje nude rješenja za detekciju lažnih blockchain računa. Najistaknutije tvrtke su Chainalysis, Elliptic i CipherTrace kao vodeći pružatelj softvera za podatkovnu analizu blockchaina. Njihov softver koriste agencije za provođenje zakona, financijske institucije i druge organizacije za istraživanje i sprječavanje kriminala povezanog s kriptovalutama.

Obzirom da smo mi fokusirani na izgradnju algoritma strojnog učenja u daljnjim koracima ćemo razmatrati taj segment, odnosno detekciju lažnih blockchain računa putem algoritma strojnog učenja.

6. Opis skupa podataka

Korišteni skup podataka sadrži redove poznatih prijevara i valjanih transakcija izvršenih putem Ethereuma. U danoj tablici su izlistane varijable i njihov opis. Razlika između ERC20 i ETH je u tome što ETH, često simboliziran kao Ether, predstavlja izvornu kriptovalutu Ethereum blockchaina. Služi kao gorivo koje pokreće mrežu, omogućuje transakcije, nagrađuje rudare i potiče sudjelovanje u mreži. Baš kao što je zlato standardna valuta u mnogim gospodarstvima, ETH igra sličnu ulogu unutar Ethereum ekosustava. ERC20, akronim za Ethereum Request for Comment 20, predstavlja tehnički standard koji upravlja stvaranjem zamjenjivih tokena na Ethereum blockchainu. Zamjenjivost podrazumijeva da su ti tokeni međusobno zamjenjivi, što znači da je jedan ERC20 token identičan po vrijednosti i korisnosti bilo kojem drugom ERC20 tokenu.

Tablica 6.1. Popis varijable i njihovog značenja

Variable Name	Variable Description
Index	The index number of a row
Address	The address of the Ethereum account
FLAG	Whether the transaction is fraud or not

Variable Name	Variable Description
Avg min between sent tnx	Average time between sent transactions for account in minutes
Avg_min_between_received_tnx	Average time between received transactions for account in minutes
Time_Diff_between_first_and_last(Mins)	Time difference between the first and last transaction
Sent_tnx	Total number of sent normal transactions
Received_tnx	Total number of received normal transactions
Number_of_Created_Contracts	Total number of created contract transactions
Unique_Received_From_Addresses	Total unique addresses from which the account received transactions
Unique_Sent_To_Addresses20	Total unique addresses to which the account sent transactions
Min_Value_Received	Minimum value in Ether ever received
Max_Value_Received	Maximum value in Ether ever received
Avg_Value_Received5	Average value in Ether ever received
Min_Val_Sent	Minimum value of Ether ever sent
Max_Val_Sent	Maximum value of Ether ever sent
Avg_Val_Sent	Average value of Ether ever sent
Min_Value_Sent_To_Contract	Minimum value of Ether sent to a contract
Max_Value_Sent_To_Contract	Maximum value of Ether sent to a contract
Avg_Value_Sent_To_Contract	Average value of Ether sent to contracts
Total_Transactions(Including_Tnx_to_Create_Contract)	Total number of transactions
Total_Ether_Sent	Total Ether sent for account address
Total_Ether_Received	Total Ether received for account address
Total_Ether_Sent_Contracts	Total Ether sent to contract addresses
Total_Ether_Balance	Total Ether balance following enacted transactions
Total_ERC20_Tnxs	Total number of ERC20 token transfer transactions

Variable Name	Variable Description
ERC20_Total_Ether_Received	Total ERC20 token received transactions in Ether
ERC20_Total_Ether_Sent	Total ERC20 token sent transactions in Ether
ERC20_Total_Ether_Sent_Contract	Total ERC20 token transfer to other contracts in Ether
ERC20_Uniq_Sent_Addr	Number of ERC20 token transactions sent to unique account addresses
ERC20_Uniq_Rec_Addr	Number of ERC20 token transactions received from unique addresses
ERC20_Uniq_Rec_Contract_Addr	Number of ERC20 token transactions received from unique contract addresses
ERC20_Avg_Time_Between_Sent_Tnx	Average time between ERC20 token sent transactions in minutes
ERC20_Avg_Time_Between_Rec_Tnx	Average time between ERC20 token received transactions in minutes
ERC20_Avg_Time_Between_Contract_Tnx	Average time between ERC20 token transactions
ERC20_Min_Val_Rec	Minimum value in Ether received from ERC20 token transactions for account
ERC20_Max_Val_Rec	Maximum value in Ether received from ERC20 token transactions for account
ERC20_Avg_Val_Rec	Average value in Ether received from ERC20 token transactions for account
ERC20_Min_Val_Sent	Minimum value in Ether sent from ERC20 token transactions for account
ERC20_Max_Val_Sent	Maximum value in Ether sent from ERC20 token transactions for account
ERC20_Avg_Val_Sent	Average value in Ether sent from ERC20 token transactions for account
ERC20_Uniq_Sent_Token_Name	Number of unique ERC20 tokens transferred
ERC20_Uniq_Rec_Token_Name	Number of unique ERC20 tokens received
ERC20_Most_Sent_Token_Type	Most sent token for account via ERC20 transaction

Variable Name	Variable Description
ERC20_Most_Rec_Token_Type	Most received token for account via ERC20 transactions

Priprema podataka je ključni korak u strojnom učenju. Ovaj korak uključuje čišćenje podataka od grešaka, nevažećih podataka i duplih podataka. Također uključuje transformaciju podataka u format koji je pogodniji za analizu. U ovom slučaju, prva dva stupca, "Index" i "Address", izostavljeni su iz skupa podataka. To je učinjeno jer ovi stupci nisu relevantni za analizu.

Nakon toga, izvukle su se informacije o podacima. To je učinjeno pomoću funkcije describe() u Pythonu. Ova funkcija daje informacije o skupu podataka, kao što su prosječna vrijednost, standardna odstupanja, minimum i maksimum. Objektne varijable su pretvorene u kategoričke varijable dtype radi učinkovitijeg procesa računanja. To je učinjeno pomoću funkcije astype() u Pythonu. Ova funkcija pretvara varijablu u određeni dtype.

Metoda df.info() u Pandasu daje sažeti prikaz DataFramea, uključujući njegovu vrstu indeksa, oznake stupaca, vrste podataka stupaca, korištenje memorije, indeks raspona i broj ćelija u svakom stupcu (vrijednosti koje nisu null). To je koristan alat za brzo razumijevanje strukture i karakteristika DataFramea.

```
 #   Column                                                Non-Null Count  Dtype
---  ------                                                --------------  -----
 0   FLAG                                                  9841 non-null   int64
 1   Avg min between sent tnx                              9841 non-null   float64
 2   Avg min between received tnx                          9841 non-null   float64
 3   Time Diff between first and last (Mins)               9841 non-null   float64
 4   Sent tnx                                              9841 non-null   int64
 5   Received Tnx                                          9841 non-null   int64
 6   Number of Created Contracts                           9841 non-null   int64
 7   Unique Received From Addresses                        9841 non-null   int64
 8   Unique Sent To Addresses                              9841 non-null   int64
 9   min value received                                    9841 non-null   float64
 10  max value received                                    9841 non-null   float64
 11  avg val received                                      9841 non-null   float64
 12  min val sent                                          9841 non-null   float64
 13  max val sent                                          9841 non-null   float64
 14  avg val sent                                          9841 non-null   float64
 15  min value sent to contract                            9841 non-null   float64
 16  max val sent to contract                              9841 non-null   float64
 17  avg value sent to contract                            9841 non-null   float64
 18  total transactions (including tnx to create contract  9841 non-null   int64
 19  total Ether sent                                      9841 non-null   float64
 20  total ether received                                  9841 non-null   float64
 21  total ether sent contracts                            9841 non-null   float64
 22  total ether balance                                   9841 non-null   float64
 23   Total ERC20 tnxs                                     9012 non-null   float64
 24   ERC20 total Ether received                           9012 non-null   float64
 25   ERC20 total ether sent                               9012 non-null   float64
 26   ERC20 total Ether sent contract                      9012 non-null   float64
 27   ERC20 uniq sent addr                                 9012 non-null   float64
 28   ERC20 uniq rec addr                                  9012 non-null   float64
 29   ERC20 uniq sent addr.1                               9012 non-null   float64
 30   ERC20 uniq rec contract addr                         9012 non-null   float64
 31   ERC20 avg time between sent tnx                      9012 non-null   float64
 32   ERC20 avg time between rec tnx                       9012 non-null   float64
 33   ERC20 avg time between rec 2 tnx                     9012 non-null   float64
 34   ERC20 avg time between contract tnx                  9012 non-null   float64
 35   ERC20 min val rec                                    9012 non-null   float64
 36   ERC20 max val rec                                    9012 non-null   float64
 37   ERC20 avg val rec                                    9012 non-null   float64
 38   ERC20 min val sent                                   9012 non-null   float64
 39   ERC20 max val sent                                   9012 non-null   float64
 40   ERC20 avg val sent                                   9012 non-null   float64
 41   ERC20 min val sent contract                          9012 non-null   float64
 42   ERC20 max val sent contract                          9012 non-null   float64
 43   ERC20 avg val sent contract                          9012 non-null   float64
 44   ERC20 uniq sent token name                           9012 non-null   float64
 45   ERC20 uniq rec token name                            9012 non-null   float64
 46   ERC20 most sent token type                           9000 non-null   object
 47   ERC20_most_rec_token_type                            8990 non-null   object
dtypes: float64(39), int64(7), object(2)
memory usage: 3.7+ MB
```

Slika 6.1. Sažeti prikaz podataka iz data framea

Izračunavanje varijance numeričkih stupaca u DataFrameu je uobičajena praksa u analizi i istraživanju podataka i služi za:

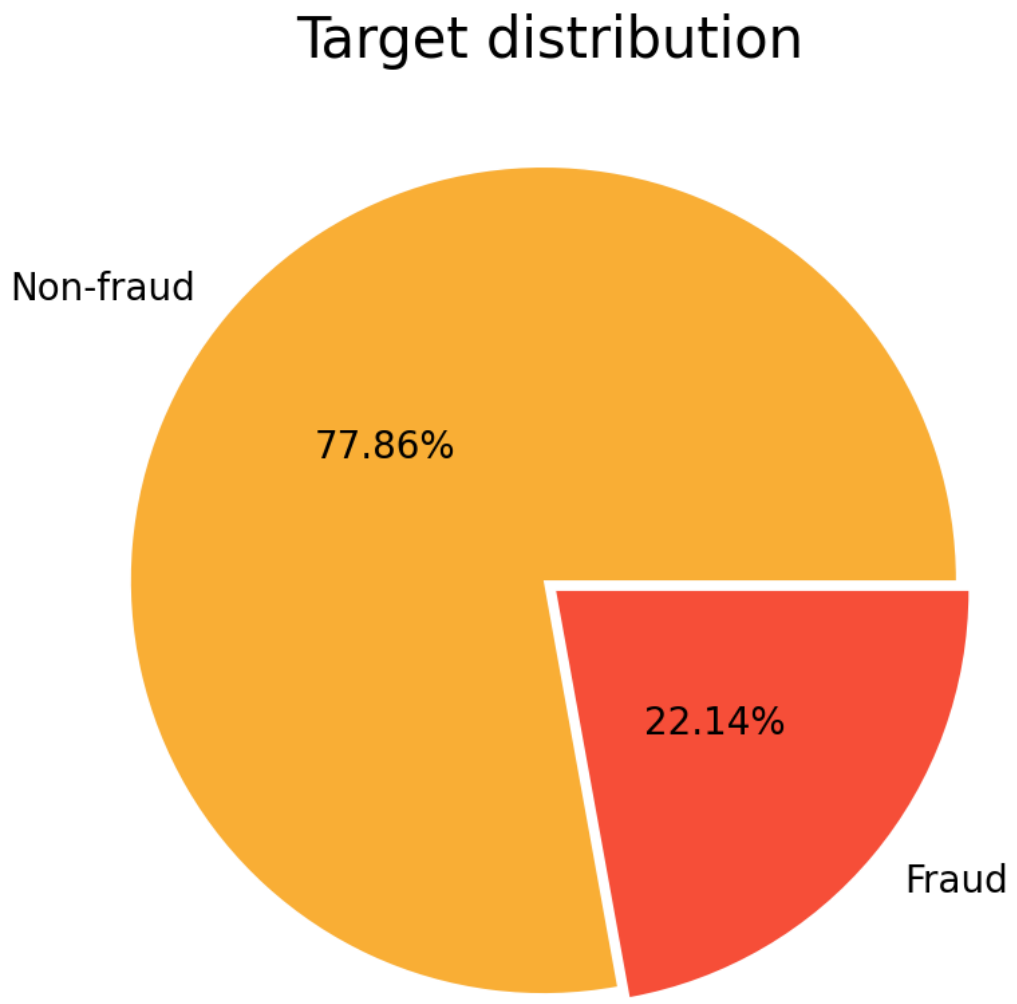
- Procjenu varijabilnosti podataka: Varijanta mjeri raspršenja podataka oko srednje vrijednosti, ukazujući na to koliko se vrijednosti u stupcu fluktuiraju. Visoka varijanca sugerira širi raspon vrijednosti, dok niska varijanca sugerira gušću koncentraciju vrijednosti oko srednje vrijednosti,
- Identificiranje odstupanja: Odstupanja su podatkovne točke koje se značajno razlikuju od ukupnog obrasca raspodjele. Izračunavanje varijance može pomoći u identificiranju potencijalnih odstupanja, jer ona obično značajno povećavaju ukupnu varijancu,

- Odabir značajki: U strojnom učenju, varijanca se može koristiti za odabir značajki, što uključuje identifikaciju najrelevantnijih značajki za određeni zadatak. Značajke s visokom varijantom često pružaju više informacija i doprinose većoj snazi modela predviđanja,
- Standardizaciju i skaliranje: Varijanta igra ulogu u tehnikama standardizacije i skaliranja podataka. Ove tehnike transformiraju numeričke značajke tako da imaju srednju vrijednost 0 i standardnu devijaciju 1, čineći ih usporedbom na zajedničkoj skali, i
- Razumijevanje distribucije podataka: Varijanta, zajedno s drugim mjerama poput standardnih odstupanja, pomaže u razumijevanju distribucije podataka, ukazujući na to da li je normalno raspoređena, iskrivljena ili ima druge značajne karakteristike.

```
FLAG                                                  1.724110e-01
Avg min between sent tnx                              4.616718e+08
Avg min between received tnx                          5.327656e+08
Time Diff between first and last (Mins)               1.042889e+11
Sent tnx                                              5.733918e+05
Received Tnx                                          8.851734e+05
Number of Created Contracts                           2.000685e+04
Unique Received From Addresses                        8.917457e+04
Unique Sent To Addresses                              6.960121e+04
min value received                                    1.062298e+05
max value received                                    1.692294e+08
avg val received                                      8.323238e+06
min val sent                                          1.921264e+04
max val sent                                          4.394646e+07
avg val sent                                          5.715935e+04
min value sent to contract                            5.080371e-08
max val sent to contract                              2.660652e-07
avg value sent to contract                            1.046096e-07
total transactions (including tnx to create contract  1.828997e+06
total Ether sent                                      1.283952e+11
total ether received                                  1.326451e+11
total ether sent contracts                            2.660625e-07
total ether balance                                   5.877009e+10
 Total ERC20 tnxs                                     2.002821e+05
 ERC20 total Ether received                           1.110618e+20
 ERC20 total ether sent                               1.393321e+18
 ERC20 total Ether sent contract                      3.756017e+07
 ERC20 uniq sent addr                                 1.107809e+04
 ERC20 uniq rec addr                                  6.694262e+03
 ERC20 uniq sent addr.1                               4.316210e-03
 ERC20 uniq rec contract addr                         2.974444e+02
 ERC20 avg time between sent tnx                      0.000000e+00
 ERC20 avg time between rec tnx                       0.000000e+00
 ERC20 avg time between rec 2 tnx                     0.000000e+00
 ERC20 avg time between contract tnx                  0.000000e+00
 ERC20 min val rec                                    2.850451e+08
 ERC20 max val rec                                    1.110370e+20
 ERC20 avg val rec                                    4.584705e+16
 ERC20 min val sent                                   1.110004e+12
 ERC20 max val sent                                   1.392176e+18
 ERC20 avg val sent                                   3.498443e+17
 ERC20 min val sent contract                          0.000000e+00
 ERC20 max val sent contract                          0.000000e+00
 ERC20 avg val sent contract                          0.000000e+00
 ERC20 uniq sent token name                           4.536185e+01
 ERC20 uniq rec token name                            2.781759e+02
dtype: float64
```

Slika 6.2. Prikaz varijance značajki

Analiza ciljne distribucije u skupu podataka ključna je za razumijevanje njegovih nijansi i potencijalnog utjecaja na modele strojnog učenja. Otkriva uvide o ciljnoj varijabli, pomažući u prepoznavanju izazova i mogućnosti modeliranja. Otkriva učestalost različitih vrijednosti, procjenjujući ravnotežu skupa podataka, dominantne klase i utjecaj neravnoteže klasa. Razumijevanje oblika distribucije, bilo normalnog, iskrivljenog ili jedinstvenog, ključno je za odabir modela i interpretaciju rezultata. Otkrivanje izvanrednih vrijednosti ključno je jer mogu poremetiti izvedbu modela. Rješavanje neravnoteže u distribuciji razreda ključno je za izbjegavanje pristranosti modela.

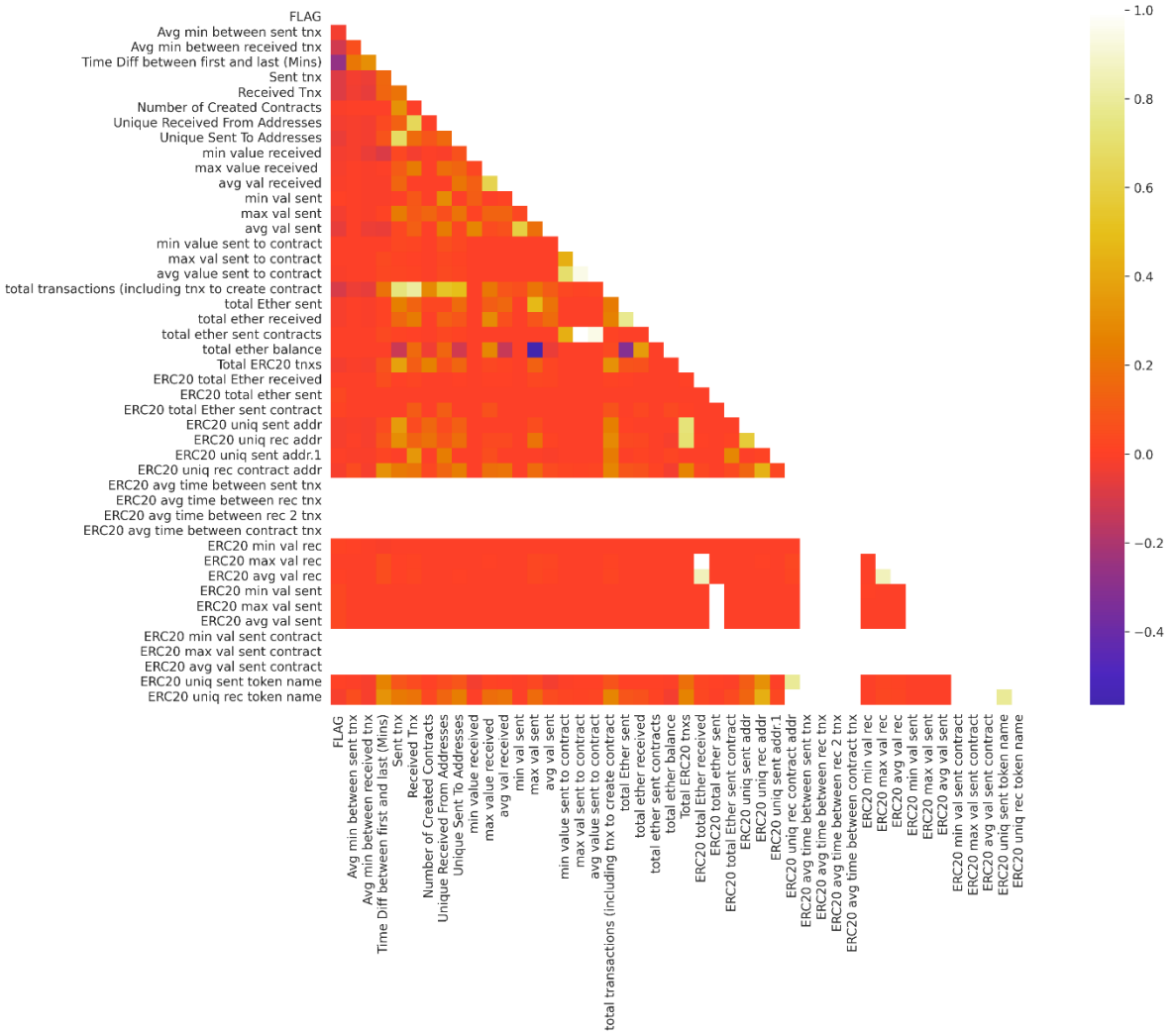


Slika 6.3 Distribucija ciljnih varijabli

Korelacijska matrica je alat koji prikazuje varijabilne odnose, pomažući u otkrivanju multikolinearnosti i identificiranju suvišnih značajki. U odabiru značajki, on točno određuje relevantne varijable koje su u korelaciji s ciljem ili jedna s drugom. Poboljšava interpretabilnost modela pokazujući kako promjene značajke utječu na predviđanja. Matrica je kvadratna koja sadrži koeficijente korelacije između svih parova varijabli u skupu podataka. Koeficijenti korelacije mjere snagu i smjer linearne veze između dvije varijable. Koeficijent korelacije od 1 ukazuje na savršenu pozitivnu korelaciju, koeficijent korelacije od -1 ukazuje na savršenu negativnu korelaciju, a koeficijent korelacije od 0 ukazuje na to da nema linearne veze između dvije varijable.

Neophodan je u istraživačkoj analizi podataka, otkrivanju skrivenih obrazaca i uvida. Osim toga, pomaže u čišćenju podataka identificiranjem problematičnih značajki. Matrica vodi odabir modela, a promjene u njoj mogu signalizirati probleme s izvedbom modela.

Ono što je najvažnije znati o matričnim korelacijama jest da mjere samo linearne odnose. To znači da možda neće moći uhvatiti sve odnose između varijabli, posebno ako su odnosi nelinearni. Osim toga, matrične korelacije mogu biti zavaravajuće ako postoje problemi s multikolinearnošću, što znači da postoje dvije ili više varijabli koje su visoko korelirane jedna s drugom. Unatoč tim ograničenjima, matrične korelacije mogu biti vrijedan alat za analizu podataka. Mogu se koristiti za identificiranje potencijalnih odnosa između varijabli, za smanjenje broja varijabli u skupu podataka i za procjenu ukupne kvalitete skupa podataka.



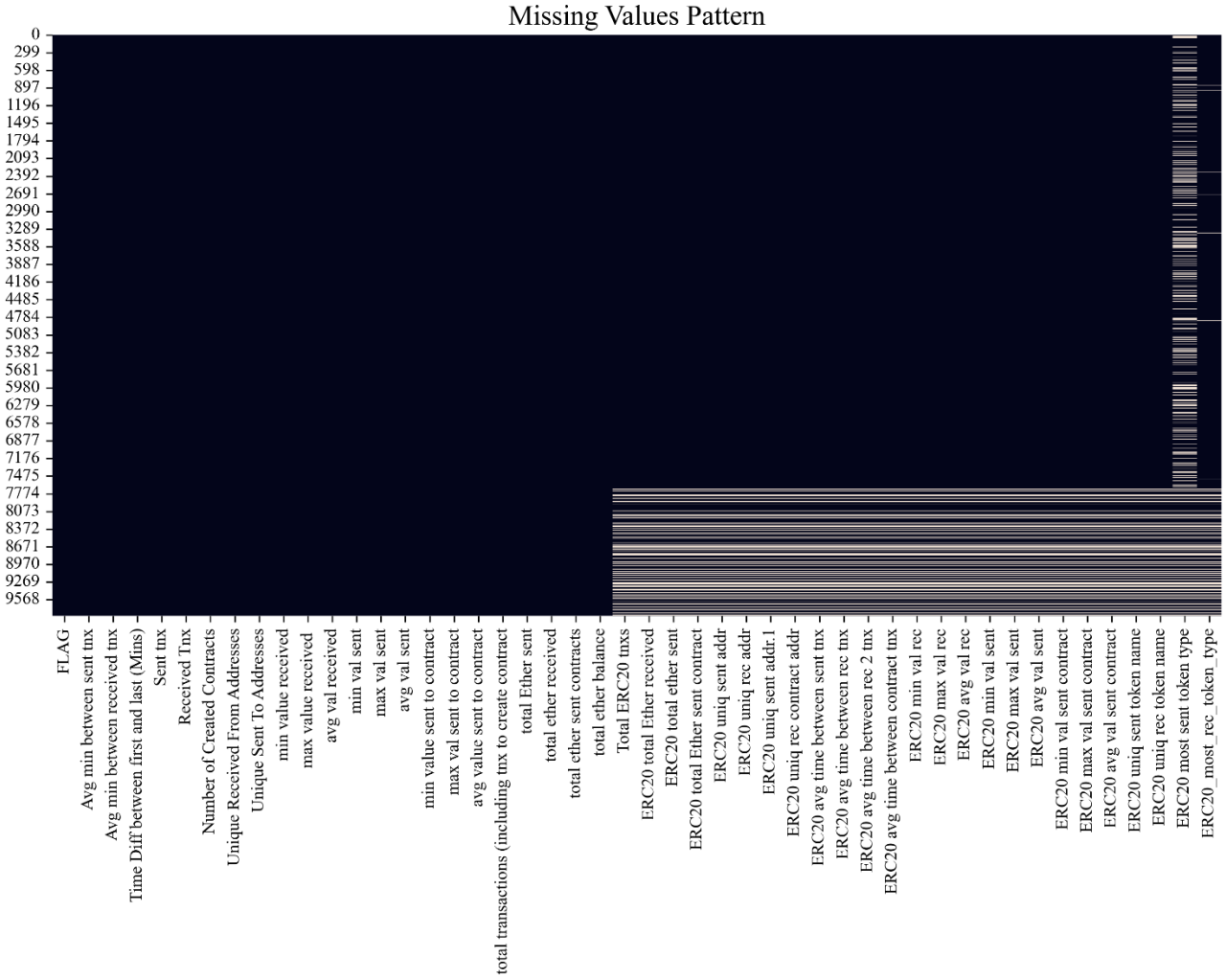
Slika 6.4 Prikaz korelacijske matrice

7. Čišćenje podataka

Heatmap u Seaborn-u je vizualizacija dvodimenzionalne matrice podataka, često korištena za prikazivanje korelacije između varijabli, iako se može primijeniti i za druge vrste podataka. Boje u heatmapu se koriste za prikazivanje korelacije: tamnije boje označavaju pozitivnu korelaciju, svjetlije boje označavaju negativnu, dok bijela ukazuje na odsustvo korelacije.

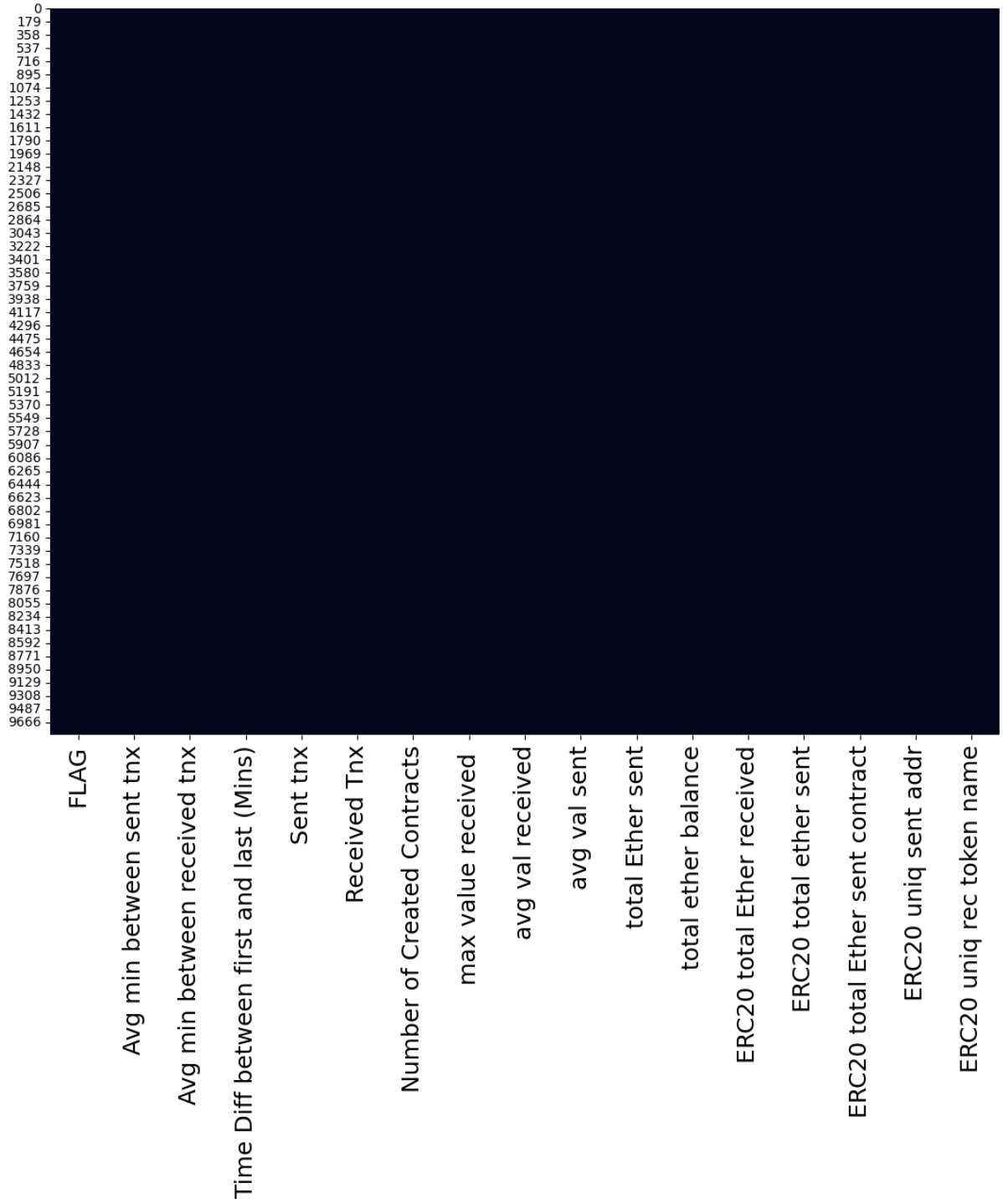
Vrijednosti u legendi heatmapa predstavljaju korelacijske koeficijente, koje sežu od -1 do 1. Vrijednost 0 označava odsustvo korelacije, dok vrijednosti bliže 1 ili -1 ukazuju na jaču korelaciju. Na primjer, tamna boja u gornjem lijevom kutu heatmapa ukazuje na jaku pozitivnu korelaciju između "x" i "y" varijabli, dok svijetla boja u donjem desnom kutu ukazuje na jaku

negativnu korelaciju između istih varijabli. Bijela boja u središtu heatmapa ukazuje na odsustvo korelacije između varijabli "x" i "y".



Slika 7.1. Dvodimenzionalni prikaz korelacije podataka

Budući da kategoričke značajke mogu biti teške za rukovanje s nedostajućim vrijednostima, često ih je bolje potpuno izbaciti iz skupa podataka. To je osobito istinito ako kategorička obilježja nisu jako važna za analizu. Za numeričke značajke možete zamijeniti vrijednosti koje nedostaju s medijanom odgovarajuće značajke. Ovo je uobičajeni pristup jer je medijan robusna mjera središnje tendencije na koju ne utječu outlieri. Nakon ovo čišćenja ponovo provjerimo stanje heatmapa.

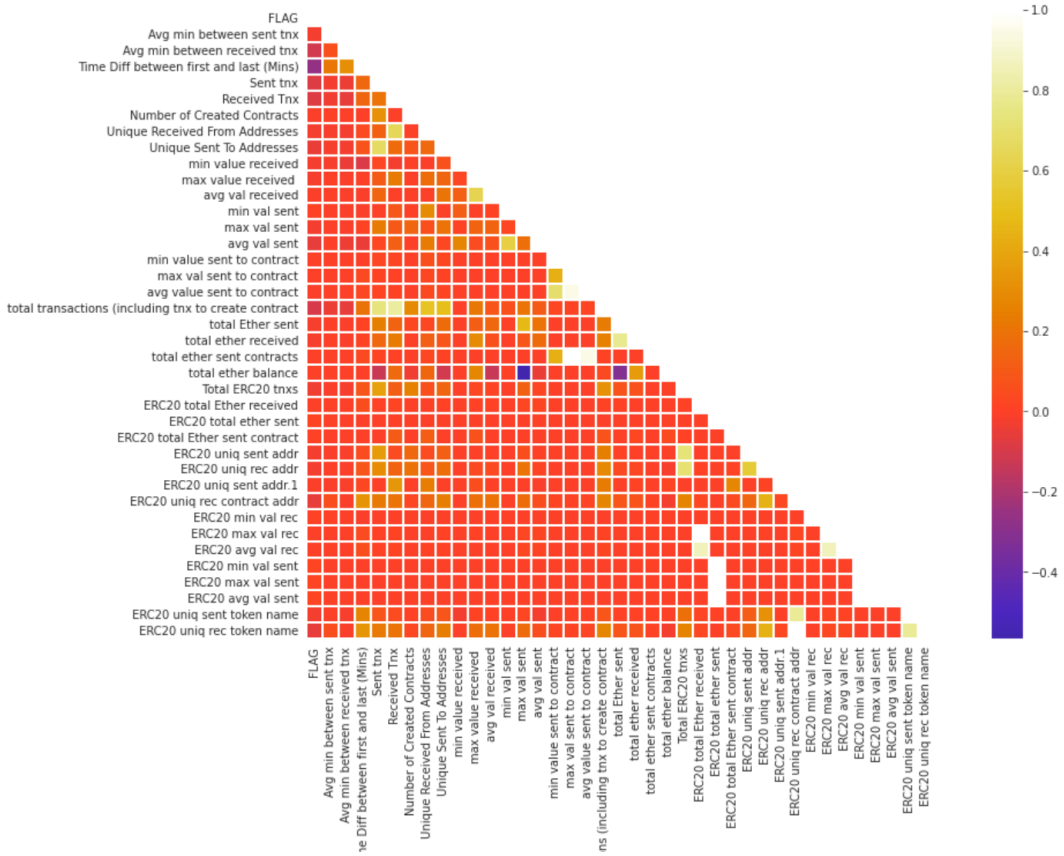


Slika 7.2 Dvodimenzionalni prikaz korelacije podataka nakon ukanjanja dviju kategoričkih značajki i zamjena numeričkih vrijednosti koje nedostaju sa medijanom

Istražujući varijancu značajki, uočeno je da postoje neke značajke s varijancom = 0, Značajke s varijancom 0 ne pružaju nikakve korisne informacije i mogu se ukloniti iz skupa podataka. To će smanjiti dimenzionalnost skupa podataka i olakšati njegovu analizu.

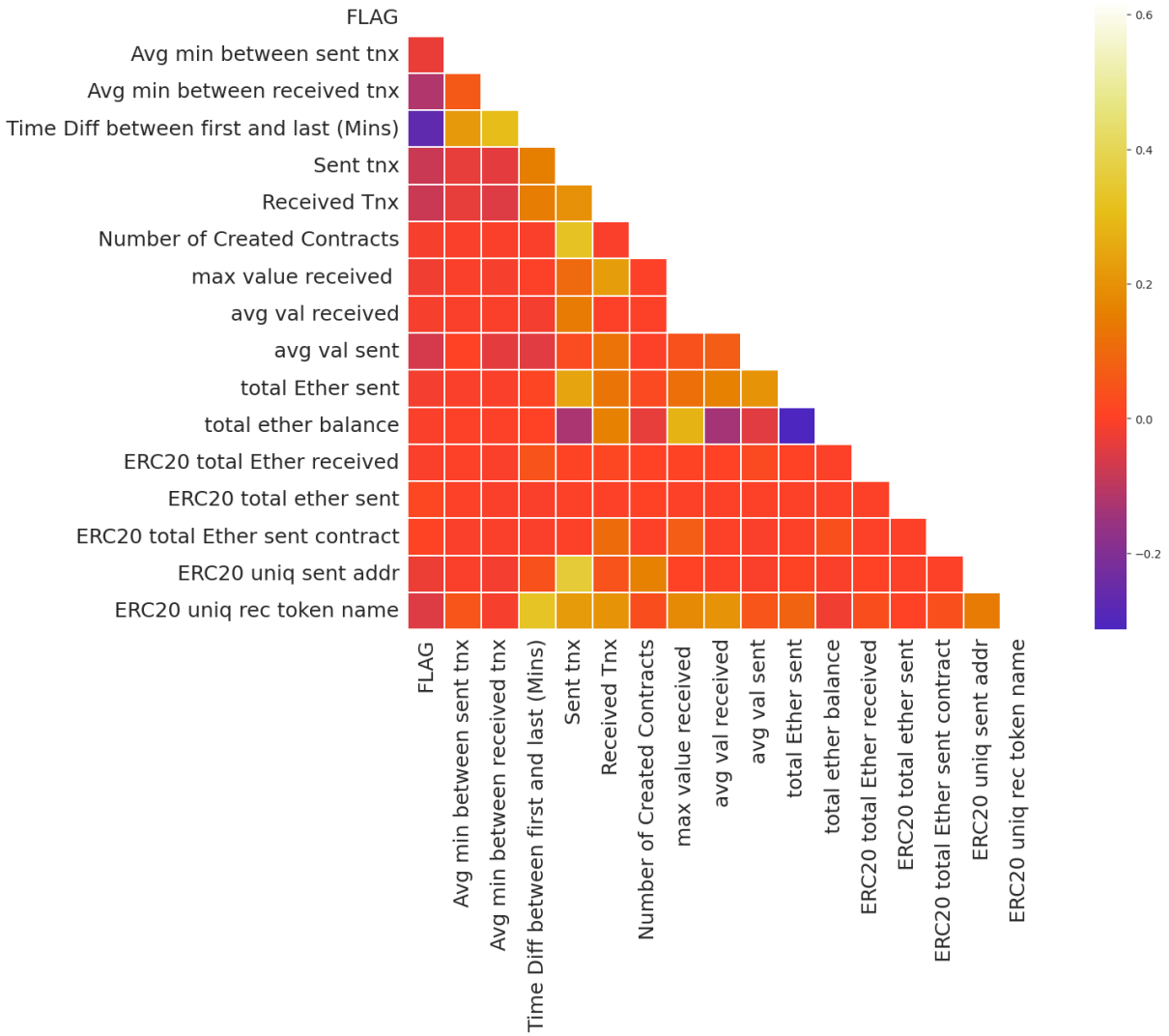
Tablica 7.1 Uklonjene značajke sa varijancom 0

ERC20 avg time between sent tnx	0
ERC20 avg time between rec tnx	0
ERC20 avg time between rex 2 tnx	0
ERC20 avg time between contract tnx	0
ERC20 min val sent contract	0
ERC20 max val sent contract	0
ERC20 avg val sent contract	0



Slika 7.3 Korelacijska matrica nakon uklanjanja značajki sa varijancom 0

Ponovo smo provjerili korelacijsku matricu. Cilj je identificirati značajke koje su u visokoj korelaciji jedna s drugom, što može ukazivati na redundanciju. Visoko korelirane značajke ćemo ukloniti i nastavljamo proces.

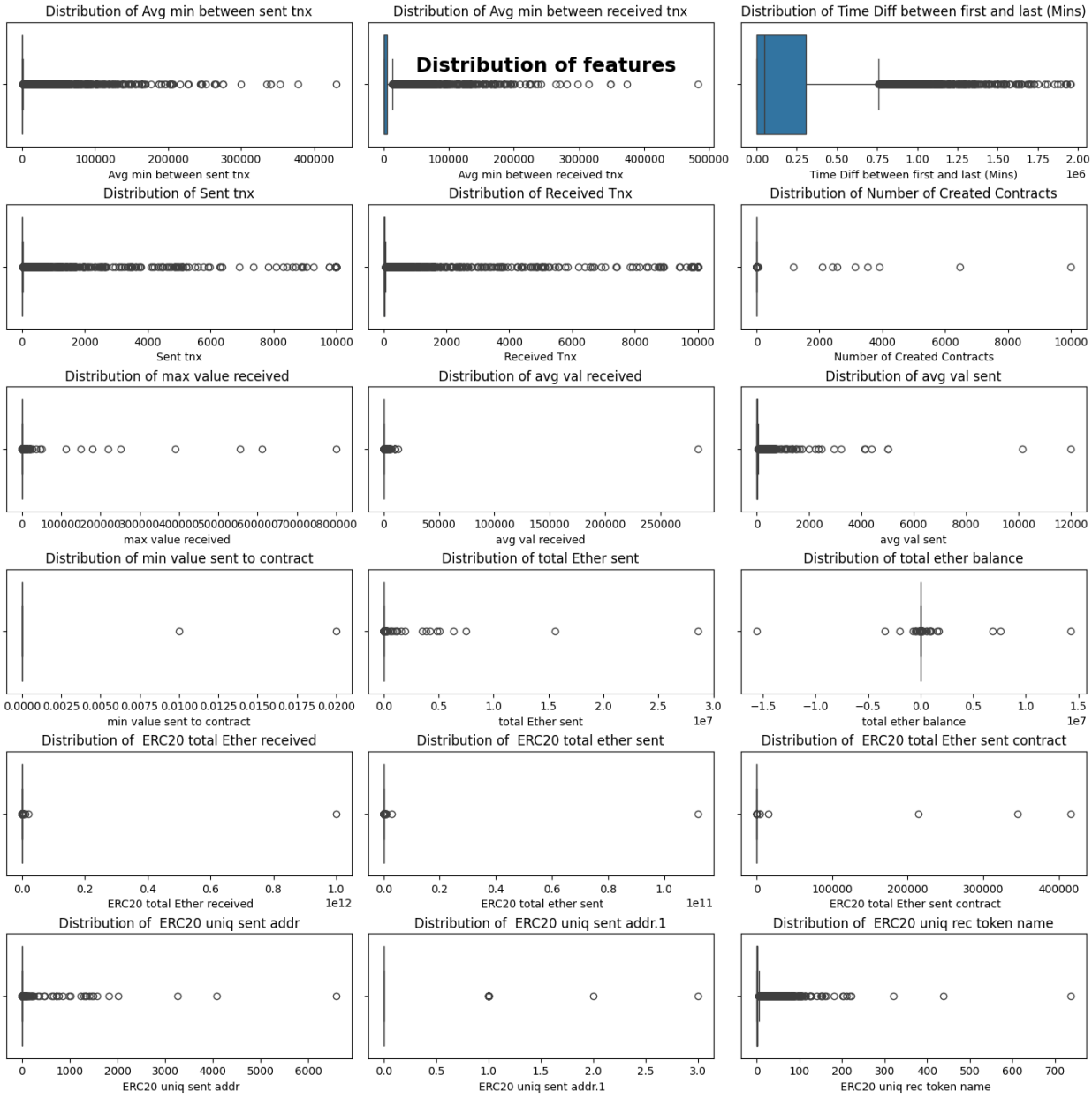


Slika 7.4 Korelacijska matrica nakon uklanjanja visoko koreliranih značajki

Boxplot vizualno sažima numeričke podatke prikazujući ključne statistike: minimum, prvi kvartil (Q1), medijan (Q2), treći kvartil (Q3) i maksimum. Uključuje okvir od Q1 do Q3, liniju za medijan i dvije linije koje dosežu minimalnu i maksimalnu vrijednost. Ovaj dijagram daje sažeti pregled distribucije podataka, ističući središnju tendenciju (medijan) i širenje (interkvartilni raspon), te identificira potencijalne izvanredne vrijednosti izvan dosega dviju linija.

Boxplot se sastoji od pet glavnih komponenata:

- Kutija (engl. Box) sama predstavlja srednjih 50% podataka. Graniči se s prvim kvartilom (Q1) i trećim kvartilom (Q3). Mediana (Q2) je linija koja kutiju dijeli na dva jednaka dijela,
- Brkovi(engl. Whiskers) se protežu od krajeva kutije kako bi pokazali raspon podataka. Obično se produžuju do 1,5 puta interkvartilnog opsega (IQR), što je razlika između Q3 i Q1,
- Linija na vrhu brka je najveća vrijednost u skupu podataka, isključujući outliere,
- Linija na dnu brka je najmanja vrijednost u skupu podataka, isključujući outliere,
- Svakoj točki koja padne izvan brka smatra se outlierom. To su podatkovne točke koje su ili vrlo visoke ili vrlo niske u odnosu na ostatak podataka.



Slika 7.5. Boxplot prikaz značajki

Boxplot pokazuje da je raspodjela svih četiri značajke pomaknuta udesno, što znači da ima više izvanrednih vrijednosti u gornjem repu distribucije. Medijanske vrijednosti za sve četiri značajke relativno su niske, dok su maksimalne vrijednosti mnogo veće.

Tumačenje značajki je sljedeće:

1. Avg min između poslanih tx: Ova značajka pokazuje prosječno vrijeme potrebno da se transakcija potvrdi nakon što je poslana. Medijanska vrijednost za ovu značajku je 1,5 minute, ali maksimalna vrijednost je preko 100 minuta. To sugerira da ima nekoliko izdvojenih vrijednosti gdje je za potvrdu transakcije trebalo vrlo dugo vremena,
2. Avg min između primljenih tx: Ova značajka pokazuje prosječno vrijeme potrebno za potvrdu transakcije nakon što je primljena. Medijanska vrijednost za ovu značajku je jedna minuta, ali maksimalna vrijednost je preko 60 minuta. To sugerira da ima nekoliko izdvojenih vrijednosti gdje je trebalo vrlo dugo vremena potvrditi primljenu transakciju, i
3. Time Diff između prvog i posljednjeg (minuti): Ova značajka pokazuje razliku u vremenu između prve i posljednje transakcije u setu. Medijanska vrijednost za ovu značajku je 10 minuta, ali maksimalna vrijednost je preko 1000 minuta. To sugerira da ima nekoliko izdvojenih vrijednosti gdje je bilo vrlo dugo vremena između prve i posljednje transakcije u setu.

Boxplot sugerira da postoji velika varijacija u vremenu potrebnom transakcijama za potvrdu i da postoje nekoliko izdvojenih vrijednosti gdje je za potvrdu transakcija trebalo vrlo dugo vremena. To bi moglo biti posljedica nekoliko čimbenika, kao što su zagušenje mreže, složenost transakcije ili prioritet transakcije. Brkovi se protežu do 1,5 puta IQR-a, što je razlika između Q3 i Q1. To znači da brkovi hvataju 95% podataka.

Sve točke koje se nalaze izvan brkova smatraju se izvanrednim vrijednostima. Postoje nekoliko izvanrednih vrijednosti u gornjem repu distribucije za sve četiri značajke. Može se uočiti da su vrijednosti ovih dviju varijabli većinom nule. Stoga će obje značajke biti odbačene jer neće biti od pomoći za naš model.

8. Priprema podataka

Prvi korak uključuje podjelu skupa podataka u dva podskupa: skup za treniranje (80% podataka) i skup za testiranje (preostalih 20%). Ova je podjela ključna kako bi se spriječio overfitting, gdje model postaje pretjerano prilagođen podacima o obuci i ne uspijeva se generalizirati na nove podatke. Testni skup koristi se za procjenu izvedbe modela usporedbom njegovih predviđanja sa stvarnim oznakama.

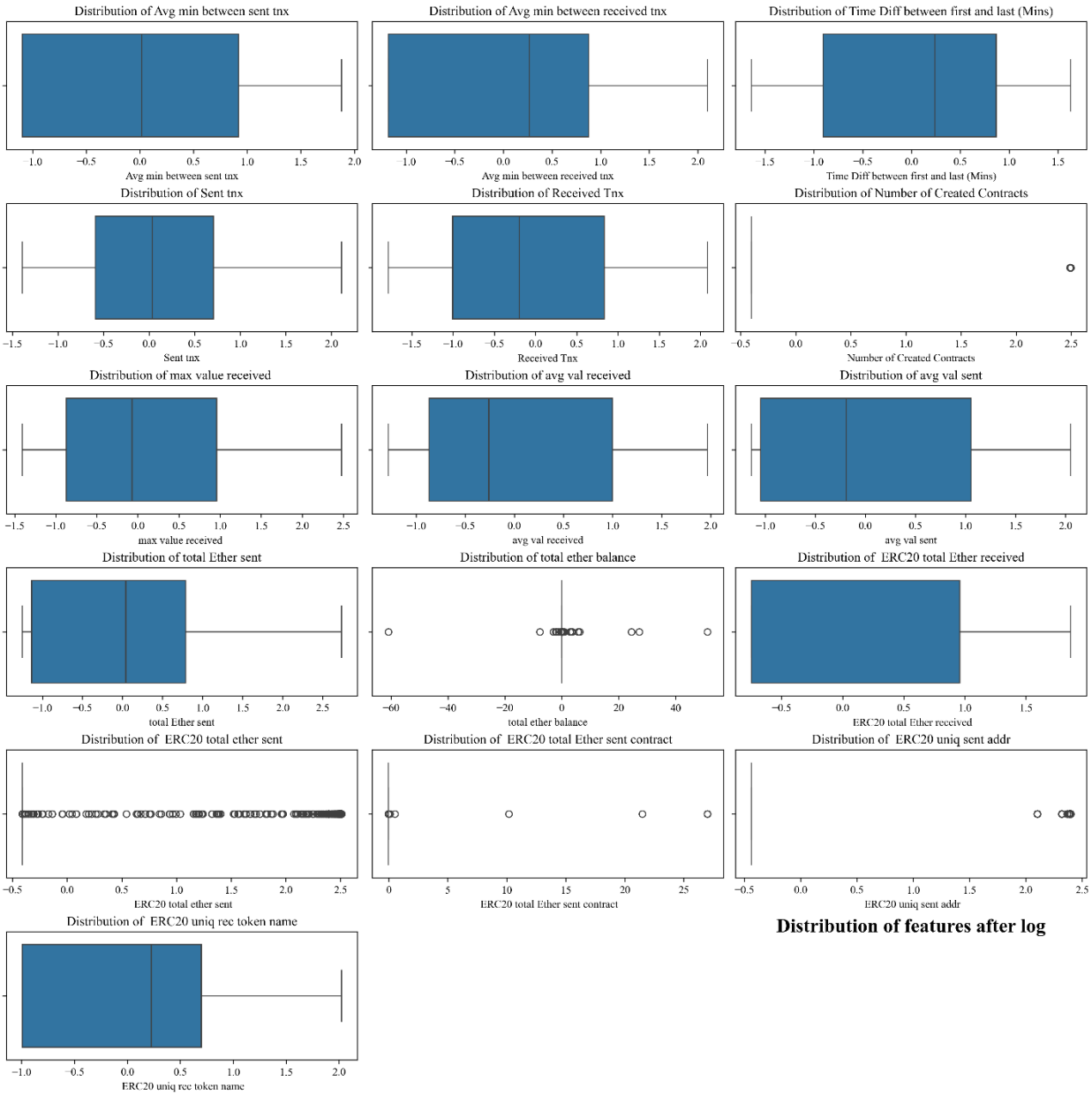
Normalizacija podataka ima za cilj skalirati značajke na zajednički raspon, olakšavajući algoritmima strojnog učenja učenje iz podataka. Dvije uobičajene tehnike normalizacije su min-max skaliranje, koje preslikava značajke u raspon [0, 1], i normalizacija z-rezultata, koja značajkama daje srednju vrijednost od 0 i standardnu devijaciju od 1. To osigurava da sve značajke su slično skalirane, sprječavajući da bilo koja značajka dominira procesom učenja.

Transformacija distribucije s logaritamskom transformacijom: logaritamska transformacija je matematička operacija koja se primjenjuje na značajke za rješavanje asimetrije u distribuciji podataka. Asimetrija se javlja kada podaci nisu normalno raspoređeni. Uzimajući logaritam vrijednosti svake značajke, transformacija sažima veće vrijednosti i širi manje, čineći podatke sličnijima normalnoj distribuciji. Mnogi algoritmi strojnog učenja pretpostavljaju da su podaci normalno distribuirani, a transformacija dnevnika pomaže u ispunjavanju te pretpostavke.

	Avg min between sent tnx	Avg min between received tnx	Time Diff between first and last (Mins)	Sent tnx	Received Tnx	Number of Created Contracts	max value received	avg val received	avg val sent	total Ether sent	total ether balance	ERC20 total Ether received	ERC20 total ether sent	ERC20 total Ether sent contract	ERC20 uniq sent addr	ERC20 uniq rec token name
0	1.294061	1.151313	1.393751	1.591951	1.017881	-0.401539	1.170696	0.988757	0.651223	1.416308	-0.007274	1.815951	2.508169	-0.038483	2.398649	1.831406
1	-1.096066	-1.184221	-1.638410	-1.391726	-1.785005	-0.401539	-1.407378	-1.283886	-1.138468	-1.252291	-0.006836	-0.746114	-0.410600	-0.038483	-0.437145	0.226082
2	-0.006354	0.213137	1.103220	1.970707	1.876994	-0.401539	0.613575	-0.995919	-0.869934	1.171479	-0.006819	-0.746114	-0.410600	-0.038483	-0.437145	-0.994019
3	-1.096066	1.220438	0.195684	-1.391726	-0.618856	2.490307	-0.871196	-0.822389	-1.138468	-1.252291	-0.006832	-0.746114	-0.410600	-0.038483	-0.437145	-0.994019
4	0.628503	-1.121221	-0.904665	-0.206187	-0.618856	-0.401539	0.889875	1.277099	1.332663	0.787323	-0.006836	-0.746114	-0.410600	-0.038483	-0.437145	-0.994019
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7867	-1.096066	0.249518	-0.705997	-0.590325	-0.618856	-0.401539	-0.535113	-0.269750	0.183532	-0.463079	-0.006836	-0.746114	-0.410600	-0.038483	-0.437145	0.226082
7868	1.128623	0.633621	0.895636	0.707356	1.432439	-0.401539	0.339924	-0.354586	1.043745	1.010645	-0.006835	1.611838	-0.410600	-0.038483	-0.437145	1.238838
7869	-1.096066	-1.184221	-1.154101	-0.590325	-1.004903	-0.401539	-1.275895	-1.091970	-0.960939	-1.159157	-0.006835	-0.746114	-0.410600	-0.038483	-0.437145	-0.994019
7870	1.385084	0.988959	0.715180	0.707356	0.556810	-0.401539	1.858790	1.531808	1.671553	1.592460	-0.006835	1.514388	-0.410600	-0.038483	-0.437145	1.122262
7871	-1.096066	0.565250	0.761464	-1.391726	1.419117	2.490307	-0.956946	-0.971335	-1.138468	-1.252291	-0.006675	0.014594	-0.410600	-0.038483	-0.437145	0.956416

7872 rows × 16 columns

Slika 8.1 Skup podataka nakon normalizacije



Slika 8.2 Distribucija značajki nakon primjene logaritamske transformacije

9. Uravnoteženje skupa podataka

Podjela 80/20 uobičajena je i jednostavna metoda za dijeljenje podataka u skupove za obuku i testiranje. Uključuje podjelu podataka u dva skupa, gdje se 80% podataka koristi za obuku, a 20% podataka koristi se za testiranje. Podjela 80/20 dobra je metoda opće namjene za dijeljenje podataka, ali može biti problematično ako su podaci neuravnoteženi. Neuravnoteženi skup podataka je skup podataka u kojem je jedna klasa mnogo češća od druge klase. Na primjer, u skupu podataka za otkrivanje prijevare, klasa lažnih transakcija može biti puno manja od klase valjanih transakcija. SMOTE je metoda za rješavanje neuravnoteženih skupova podataka.

Tehnika sintetskog manjinskog preuzorkovanja (engl. SMOTE) tehnika je predobrade podataka osmišljena za borbu protiv neravnoteže klasa u strojnom učenju. Klasna neravnoteža nastaje kada postoji značajna razlika u broju instanci između većinske (ciljane) klase i manjinske (neciljane) klase, što predstavlja izazov za algoritme strojnog učenja koji se često bore da učinkovito nauče manjinsku klasu.

SMOTE rješava ovaj problem generiranjem sintetičkih uzoraka za manjinsku klasu. Ovi sintetički uzorci stvoreni su interpolacijom između postojećih instanci manjinske klase, uvođenjem raznolikosti bez repliciranja identičnih točaka podataka.

Prije oversamplinga početni skup podataka ima neravnoteže u klasi, s većinom uzoraka koji nisu prijevare (6115) i manjinom uzoraka prijevara (1757). Ova neravnoteža predstavlja izazov za modele strojnog učenja jer bi mogli naučiti favorizirati većinsku klasu i zanemariti manjinsku klasu. To može dovesti do loše izvedbe, osobito kada je u pitanju prepoznavanje slučajeva prijevare.

Nakon preuzorkovanja (engl. Oversampling), broj uzoraka koji nisu prijevare i prijevare izjednačen je na 6115 svaki. To znači da je distribucija klasa sada uravnotežena, a model ne bi trebao biti pristran ni prema jednoj klasi. Sada je vjerojatnije uhvatiti temeljne obrasce u podacima i točno identificirati slučajeve prijevare i slučajeve koji nisu prijevare.

10. Algoritmi

Kod strojnog učenja, klasifikacijski zadaci imaju ogromnu važnost, omogućavajući nam da podatkovne točke kategoriziramo u unaprijed definirane skupine na temelju njihovih karakteristika. Ovi zadaci pronalaze primjenu u raznim područjima, od medicinske dijagnoze do otkrivanja prijevara i filtriranja neželjene pošte. Među vodećim klasifikacijskim algoritmima ističu se logistička regresija (engl. Logistic regression), Slučajna šuma (engl. Random forest) i XGBoost.

10.1 Logistička regresija

Logistička regresija, utemeljena na statističkoj vjerojatnosti, djeluje procjenom vjerojatnosti događaja koji se događaju. Za razliku od linearne regresije, koja se bavi kontinuiranim ishodima, logistička regresija usredotočuje se na predviđanje binarnih ishoda, kao što je je li e-pošta neželjena pošta ili će klijent odustati. Jedna od značajki logističke regresije leži u njezinoj interpretabilnosti. Pregledom koeficijenata povezanih s svakom varijablom predikatora, možemo lako razumjeti utjecaj svake značajke na predviđenu vjerojatnost. Primjerice, ako je koeficijent za varijablu predikatora pozitivan, to znači da je povećanje vrijednosti te značajke povezano s povećanom vjerojatnošću predviđenog ishoda. Nasuprot tome, negativan koeficijent implicira suprotan odnos.

Iako logističkom regresijom se može predvidjeti s nevjerojatnom preciznošću, važno je riješiti potencijalnu prenaučenost, fenomen u kojem model postaje previše specifičan za podatke za učenja, što dovodi do loše izvedbe na neviđenim podacima. Za ublažiti prenaučenost mogu se koristiti tehnike regularizacije, koje penaliziraju model za složene odnose, potičući ga bolje generalizira.

Glavni hiperparametri koji se mogu podesiti za logističku regresiju su:

1. Snaga regularizacije (C): Regularizacija (engl. Regularization strength) je tehnika koja pomaže spriječiti prekomjerno prilagođavanje kažnjavanjem modela za složene odnose. Parametar C kontrolira snagu regulacije, pri čemu više vrijednosti dovode do manje regulacije i potencijalno većeg pretjeranog prilagođavanja,
2. Kazna (L1 ili L2): Parametar kazne (engl. Penalty L1 or L2) određuje vrstu regularizacije koja se koristi. L1 regulacija smanjuje koeficijente manje važnih značajki prema nuli, dok L2 regulacija proporcionalno smanjuje sve koeficijente,
3. Rješivač: Algoritam rješavača (engl. Solver) određuje metodu za optimizaciju parametara modela. Uobičajeni rješavači uključuju Newton-Raphson, BFGS i stohastički gradijentni spust,
4. Max_iter: Maksimalni broj ponavljanja kojima će rješavač pokušati optimizirati model. Povećanje ovog parametra može poboljšati konvergenciju, ali također može dovesti do pretjeranog prilagođavanja, i
5. tol: Parametar tolerancije specificira minimalnu promjenu u funkciji cilja koja se smatra značajnom. Manja tolerancija dovest će do više ponavljanja i točnijih rezultata, ali također može povećati rizik od prekomjernog opremanja.

10.2 Random forest

Random Forest popularan je algoritam učenja u ansamblu koji se koristi za zadatke klasifikacije i regresije. Kombinira predviđanja višestrukih stabala odlučivanja kako bi se napravila preciznija i robusnija predviđanja. Pripada obitelji algoritama za učenje ansambla. Skupno učenje kombinira predviđanja više pojedinačnih modela kako bi se poboljšala ukupna prediktivna izvedba.Osnovna ideja iza skupnog učenja je da združivanje predviđanja više modela često može dovesti do boljih rezultata u usporedbi s korištenjem jednog modela

Izgrađena je na stablima odlučivanja. Stabla odlučivanja hijerarhijski su modeli koji donose niz binarnih odluka na temelju ulaznih značajki kako bi došli do predviđanja. Svako stablo odlučivanja u nasumičnoj šumi zasebni je klasifikator ili regresor koji neovisno uči iz podataka o obuci. Slučajnost je ključna značajka Random Foresta. Uvodi varijabilnost u proces obuke modela, što pomaže smanjiti prekomjerno opremanje i poboljšati generalizaciju.

Random Forest uvodi slučajnost na dva glavna načina:

1. Slučajni odabir podskupa: Umjesto korištenja cijelog skupa podataka za obuku za obuku svakog stabla odlučivanja, za obuku se odabire slučajni podskup podataka (bootstrap uzorak). Ovaj nasumični odabir podskupa unosi raznolikost među stable, i

2. Podprostor nasumičnih značajki: Na svakom čvoru stabla odlučivanja razmatra se razdvajanje nasumičnog podskupa značajki. Ovaj potprostor nasumičnih značajki dodatno povećava raznolikost među stablima.

Glavni hiperparametri koji se mogu podesiti za random forest su:

1. Broj stabala: Ovaj hiperparametar kontrolira broj stabala odlučivanja koja se koriste u skupu. Veći broj stabala općenito će dovesti do bolje točnosti, ali također može povećati računalne troškove obuke i predviđanja (engl. Number of trees),
2. Maksimalna dubina stabla: Ovaj hiperparametar kontrolira maksimalnu dubinu svakog stabla odlučivanja u skupu. Veća maksimalna dubina omogućit će stablima da nauče složenije odnose, ali također može povećati rizik od prekomjernog opremanja (engl. Maximum depth of tree),
3. Minimalni broj uzoraka po listu: Ovaj hiperparametar kontrolira minimalni broj uzoraka koji moraju biti prisutni u čvoru lista svakog stabla odlučivanja. Veći minimalni broj uzoraka spriječit će stabla u učenju pretjerano granularnih uzoraka u podacima, ali također može smanjiti točnost modela (engl. Minimum number of samples per leaf), i
4. Slučajnost odabira značajki: Ovaj hiperparametar kontrolira stupanj do kojeg su značajke nasumično odabrane prilikom dijeljenja čvorova u svakom stablu odlučivanja. Viši stupanj slučajnosti pomoći će spriječiti da stabla postanu pristrana prema određenim značajkama, ali također može smanjiti točnost načina rada.

10.3 XGBoost

U području strojnog učenja, XGBoost (Extreme Gradient Boosting) pojavio se kao istaknuti algoritam koji pripada široj klasi algoritama za povećanje gradijenta, koji obuhvaćaju obitelj skupnih metoda koje kombiniraju višestruka stabla odlučivanja za poboljšanje prediktivne izvedbe. XGBoost se razlikuje po tome što ih stalno nadmašuje na različitim referentnim skupovima podataka, pokazujući iznimnu točnost i učinkovitost.

XGBoost radi na temeljnom principu gradijentnog povećanja, iterativno obučavajući pojedinačna stabla odlučivanja kako bi se smanjila ukupna funkcija gubitka. Svako se stablo usredotočuje na minimiziranje gradijenta funkcije gubitka u odnosu na ostatke prethodnog stabla. Ovaj iterativni proces dovodi do skupa stabala, kolektivnog usavršavanja predviđanja i postizanja vrhunske izvedbe.

Glavni hiperparametri koji se mogu podesiti za XGBoost uključuju:

1. Broj krugova pojačanja: ovaj hiperparametar kontrolira broj izvedenih ponavljanja pojačanja gradijenta. Veći broj krugova općenito će dovesti do bolje točnosti, ali također može povećati računalne troškove obuke i predviđanja (engl.Number of boosting rounds),
2. Stopa učenja (eta): Ovaj hiperparametar kontrolira veličinu koraka tijekom svakog koraka pojačanja. Manja stopa učenja dovest će do sporije konvergencije, ali također može smanjiti prekomjerno opremanje (engl. Learning rate),
3. Minimalno smanjenje gubitaka (gama): Ovaj hiperparametar kontrolira minimalno smanjenje gubitaka potrebno za dijeljenje čvora. Manja gama omogućit će veće razdvajanje, što može dovesti do bolje točnosti, ali također može povećati rizik od pretjeranog uklapanja. (engl. Minimum loss reduction),
4. Minimalna veličina dijeljenja (min_split_size): Ovaj hiperparametar kontrolira minimalni broj uzoraka potrebnih za dijeljenje čvora. Veća min_split_size spriječit će prekomjerno prilagođavanje sprječavanjem cijepanja stabala na vrlo male podgrupe podataka, ali također može smanjiti točnost modela. (engl. Minimum split size),

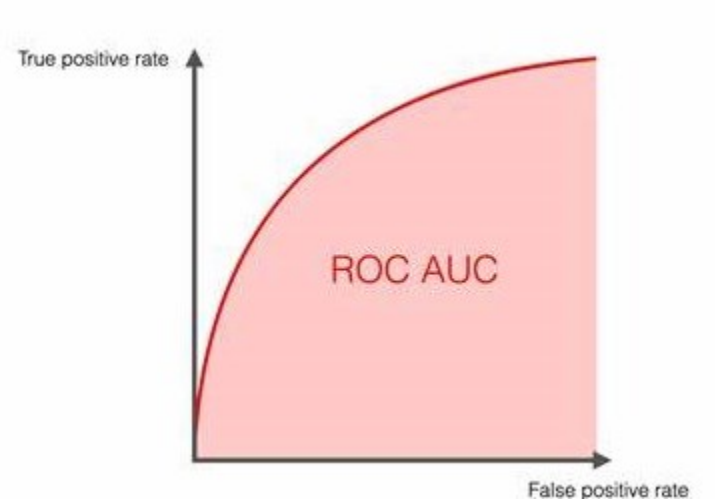
5. Maksimalna dubina stabla (max_depth): Ovaj hiperparametar kontrolira maksimalnu dubinu svakog stabla odlučivanja u skupu. Veća max_depth omogućit će stablima da nauče složenije odnose, ali također može povećati rizik od prekomjernog opremanja (engl. Maximum tree depth),
6. Poduzorak: Ovaj hiperparametar kontrolira udio podataka za obuku koji se koristi za obuku svakog stabla odlučivanja. Manji poduzorak smanjit će rizik od prekomjernog prilagođavanja sprječavajući da svako stablo više puta vidi iste podatke, ali također može smanjiti točnost modela (engl. Subsample), i
7. Stopa uzorkovanja stupca (colsample_bytree): Ovaj hiperparametar kontrolira udio značajki koje se koriste za obuku svakog stabla odlučivanja. Manji colsample_bytree također će smanjiti rizik od prekomjernog uklapanja sprječavajući da svako stablo vidi sve značajke više puta, ali također može smanjiti točnost modela (engl. Column sample rate).

11. Evaluacijske metrike

1. Točnost (engl. Accuracy) je široko korištena metrika za procjenu klasifikacijskih modela. Mjeri udio ispravno klasificiranih instanci u odnosu na ukupni broj instanci u skupu podataka. Drugim riječima, točnost pokazuje koliko se često predviđanja modela podudaraju sa stvarnim oznakama. Izračunava se kao omjer stvarnih pozitivnih i pravih negativnih rezultata prema ukupnom broju predviđanja.

$$Accuracy = \frac{True\ positives + True\ negatives}{True\ positives + Trues\ negatives + False\ positives + False\ negatives}$$

2. Krivulja radnih karakteristika prijemnika (ROC) i rezultat područja ispod krivulje (AUC) metrike su procjene koje se obično koriste u zadacima binarne klasifikacije. ROC krivulja ilustrira kompromis između prave pozitivne stope (osjetljivost) i lažno pozitivne stope (1-specifičnost) za različite klasifikacijske pragove. AUC rezultat kvantificira ukupnu izvedbu modela izračunavanjem površine ispod ROC krivulje. Viši AUC rezultat ukazuje na bolju izvedbu modela, s rezultatom 1 koji predstavlja savršen klasifikator.



Slika 11.1 Roc auc score

3. Preciznost (engl. Precision score) je metrika koja se fokusira na udio točno predviđenih pozitivnih instanci među svim instancama predviđenim kao pozitivne. Mjeri sposobnost modela da izbjegne lažna pozitivna predviđanja. Preciznost se izračunava kao omjer pravih pozitivnih rezultata i zbroja pravih pozitivnih i lažno pozitivnirezultata.

$$Precision = \frac{True\ positives}{True\ positives + False\ positives}$$

4. Prisjećanje, poznato i kao osjetljivost ili stvarna pozitivna stopa (engl. Recall_score) mjeri udio točno predviđenih pozitivnih instanci u odnosu na sve stvarne pozitivne instance. Kvantificira sposobnost modela da ispravno identificira sve pozitivne instance, a da nijednu ne propusti. Prisjećanje se izračunava kao omjer istinskih pozitivnih rezultata i zbroja pravih pozitivnih i lažno negativnih rezultata.

$$Recall = \frac{True\ positives}{True\ positives + False\ negatives}$$

5. F1 rezultat (engl. F1 Score) je harmonijska sredina preciznosti i prisjećanja, pružajući uravnoteženu mjeru performansi modela. U obzir uzima i lažno pozitivne i lažno negativne rezultate. Rezultat F1 izračunava se kao ponderirani prosjek preciznosti i prisjećanja, pri čemu se obama daje jednaka važnost. Korisno je u scenarijima u kojima želite pronaći ravnotežu između preciznosti i prisjećanja.

$$F1\ score = 2 \times \frac{Precision\ \times Recall}{Precision + Recall}$$

11.1 Podešavanje hiperparametara

Podešavanje hiperparametara je postupak prilagođavanja parametara modela strojnog učenja kako bi se optimizirala njegova izvedba. Ovi se parametri obično ne uče tijekom procesa obuke, već se postavljaju prije početka obuke. Vrijednosti ovih hiperparametara mogu imati značajan utjecaj na performanse modela, stoga je važno pažljivo ih podesiti.

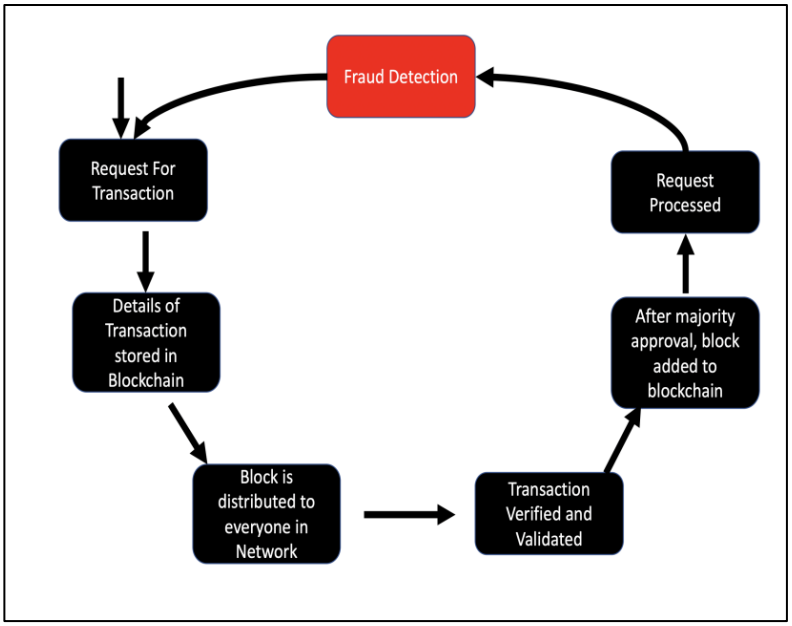
Strategije za podešavanje parametara:

1. Pretraživanje mreže (engl. Grid Search): Uključuje iscrpno isprobavanje svih kombinacija vrijednosti parametara unutar određenog raspona. To može biti računski skupo za složenije modele,
2. Nasumično pretraživanje (engl. Random search),: Uključuje nasumično uzorkovanje kombinacija vrijednosti parametara iz navedenog raspona. Ovo može biti brže od pretraživanja mreže, ali možda neće tako temeljito istražiti cijeli prostor parametara i
3. Bayesova optimizacija (engl. Bayesian optimization).: koristi probabilistički model za vođenje potrage za optimalnim vrijednostima parametara. To može biti učinkovitije od mreže ili nasumičnog pretraživanja, posebno za složene modele.

12. Treniranje modela

Obuka modela uključuje korištenje skupa podataka za obuku kako bi se naučio model strojnog učenja da daje predviđanja ili uči uzorke iz ulaznih podataka. Skup podataka za obuku sastoji se od označenih primjera gdje su poznate i ulazne značajke (X) i njihove odgovarajuće ciljne varijable (y). Predstavlja povijesne podatke ili prethodno opažene slučajeve.

Tijekom obuke, model prilagođava svoje unutarnje parametre ili težine na temelju ulaznih značajki i ciljnih varijabli. Algoritam učenja ima za cilj minimizirati pogrešku ili razliku između predviđenih rezultata i stvarnih ciljanih vrijednosti.



Slika 12.1 Tijek primjene provjere za otkrivanje prijevare

Cilj modela je pronaći optimalne vrijednosti njegovih parametara koji mu omogućuju dobru generalizaciju na nove, dosad nepoznate podatke. Specifični cilj optimizacije ovisi o vrsti modela koji se trenira (npr. minimiziranje srednje kvadratne pogreške za regresiju ili maksimiziranje vjerojatnosti za klasifikaciju).

Uvježbavanje modela često je iterativni proces, gdje se skup podataka uvježbavanja predstavlja modelu više puta (epohe) kako bi se poboljšali njegovi parametri i poboljšala izvedba. Svaka se epoha sastoji od prolaza naprijed (računanje predviđanja) i prolaza unatrag (ažuriranje parametara korištenjem gradijentnog spuštanja ili drugih tehnika optimizacije). Proces obuke se nastavlja sve dok se ne ispuni kriterij zaustavljanja, kao što je postizanje maksimalnog broja epoha ili postizanje željene razine poboljšanja performansi.

12.1 Logistička regresija

Logistička regresija je algoritam klasifikacije koji predviđa vjerojatnost binarnog ishoda. U kontekstu otkrivanja prijevare, utvrđuje vjerojatnost da je transakcija lažna ili legitimna. Algoritam analizira različite značajke transakcije, kao što su iznos transakcije, mjesto i vrijeme, te svakoj transakciji dodjeljuje ocjenu vjerojatnosti. Transakcije s višim ocjenama vjerojatnosti označene su kao potencijalno lažne, što zahtijeva daljnju istragu. Primjena kreće od obučavanja logističkog modela na pripremljenom skupu podataka nakon čega se izrađuju predikcije na temelju testnih podataka te na kraju ocjenjujem izvedbu modela koristeći matricu konfuzije i izvješće o klasifikaciji.

Tablica 12.1 Opis transakcija

FLAG	Broj transakcija	OPIS
0	1547	Valjanje transakcije
1	422	Prijevare

12.1.1 Podešavanje hiperparametara za logističku regresiju

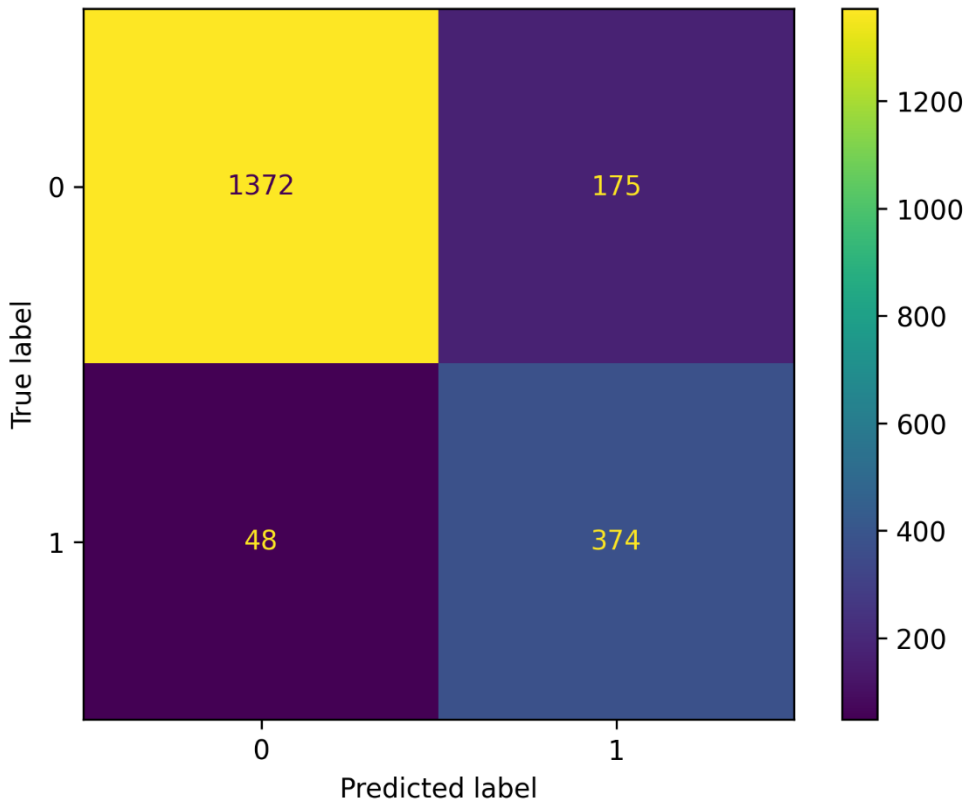
Best Hyperparameters	'C'	'Penalty'	'Solver'
	1	L2	liblinear

C parametar kontrolira snagu regularizacije. Viša vrijednost C će strože kažnjavati velike koeficijente, što će model učiniti manje sklonim pretjeranju. Niža vrijednost C će omogućiti modelu da se bolje prilagodi trenirnim podacima, ali može i učiniti model sklonijim pretjeranju.

Penalty parametar određuje vrstu regularizacije koju treba koristiti. l2 kažnjava zbroj kvadrata koeficijenata. To ima tendenciju da koeficijente smanji na nulu, što pomaže u sprječavanju pretjeranja.

Solver parametar određuje optimizacijski algoritam koji se koristi za treniranje modela. Optimizator liblinear je brz i učinkovit optimizator koji je dobro prilagođen malim do srednjim skupovima podataka.

U kontekstu algoritama logističke regresije, ovi hiperparametri podešavaju model kako bi se postigla ravnoteža između njegove sposobnosti točnog razvrstavanja novih podataka i njegove otpornosti na pretjerivanje na treniranim podacima.



Slika 12.2 Prikaz matrice konfuzije za logističku regresiju

Istinski pozitivno (TP): Model je točno identificirao 374 slučaja PRIJEVARE od 422 (tj. 88,8%). To ukazuje da je model učinkovit u identificiranju stvarnih lažnih transakcija.

Lažno pozitivno (FP): Model je netočno označio 175 transakciju kao PREVARU od 1547, dok one zapravo NISU PREVARE. To ukazuje na to da model može označiti neke legitimne transakcije kao lažne, što dovodi do nepotrebnih istraga i neugodnosti za korisnike.

Lažno negativan (FN): Model nije uspio identificirati 48 slučajeva PRIJEVARE, tretirajući ih kao NE-prijevare (tj. 11,6%). Ovo je aspekt izvedbe modela koji najviše zabrinjava jer znači da neke lažne transakcije nisu otkrivene, što potencijalno dovodi do financijskih gubitaka za organizaciju.

Baveći se scenarijem otkrivanja prijevare, važnije su transakcije koje su zapravo bile PREVARE, ali koje je naš model tretirao kao NISU PRIJEVARE (FN - 48) GREŠKA VRSTE II. Takve greške je potrebno maksimalno smanjiti, a to ćemo pokušati sa sljedećim modelima.

Tablica 12.2 Prikaz izvješća o klasifikaciji za logističku regresiju

Matrica konfuzije	Precision	Recall	F1- score	Support
0	0.97	0.89	0.93	1547
1	0.68	0.89	0.77	422
Accuracy			0.89	1969
Macro avg	0.82	0.89	0.85	1969
Weighted avg	0.91	0.89	0.89	1969

Accuracy	0.8842
ROC AUC Score	0.8841
Precision	0.6757
Recall	0.8839
F1 Score	0.7659

12.2 Random forest

Random Forest popularan je algoritam učenja u ansamblu koji se koristi za zadatke klasifikacije i regresije. Kombinira predviđanja višestrukih stabala odlučivanja kako bi se napravila preciznija i robusnija predviđanja. Osnovna ideja iza skupnog učenja je da združivanje predviđanja više modela često može dovesti do boljih rezultata u usporedbi s korištenjem jednog modela.

12.2.1 Podešavanje hiperparametara za random forest algoritam

Best Hyperparameters	'max_depth'	'max_features'	'min_samples_leaf'	'min_samples_split'	n_estimators'
	‘None’	‘Auto’	1	2	100

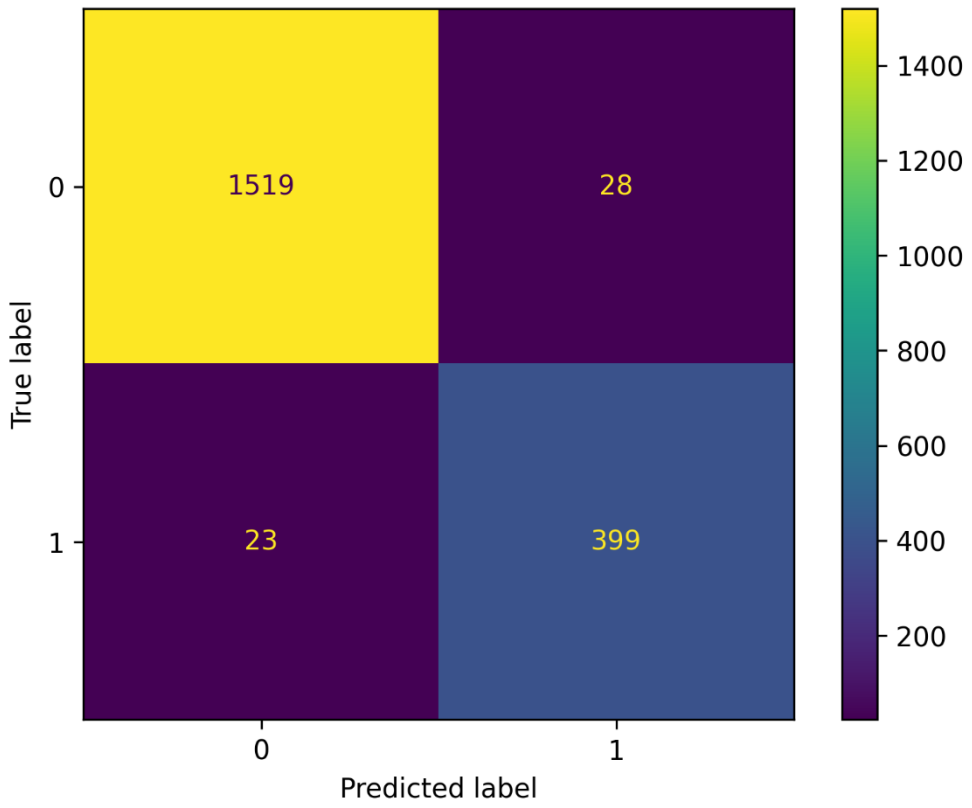
Parametar max_depth određuje maksimalnu dubinu stabla odlučivanja u šumi. Drvo s većom dubinom moći će naučiti složenije odnose između značajki i cilja, ali će također biti sklonije prekomjernom opremanju. Zadana vrijednost None znači da će stablo rasti što je dublje moguće dok ne dosegne kriterij zaustavljanja, kao što je minimalni broj uzoraka po čvoru lista.

Max_features određuje broj značajki koje se uzimaju u obzir prilikom podjele na svakom čvoru u stablu. Veća vrijednost max_features omogućit će stablu da nauči složenije odnose između značajki i cilja, ali će također učiniti stablo računalno skupljim za treniranje. Zadana vrijednost auto znači da će se broj značajki odabrati automatski na temelju broja značajki u skupu podataka.

Min_samples_leaf određuje minimalni broj uzoraka koji moraju biti u čvoru lista kako bi se stablo dalje dijelilo. Viša vrijednost min_samples_leaf spriječit će stablo od pretjeranog prilagođavanja podacima za obuku. Zadana vrijednost od 1 znači da se listni čvor može podijeliti ako uopće ima uzoraka.

Parametar min_samples_split određuje minimalni broj uzoraka koji moraju biti u čvoru prije nego što se čvor podijeli. Veća vrijednost min_samples_split spriječit će prečesto dijeljenje stabla i pretjerano prilagođavanje podacima za obuku. Zadana vrijednost 2 znači da čvor mora imati najmanje 2 uzorka kako bi se mogao podijeliti.

N_estimators određuje broj stabala u šumi. Veća vrijednost n_estimators obično će poboljšati izvedbu šume, ali će također učiniti šumu računalno skupljom za treniranje. Zadana vrijednost od 100 dobra je polazna točka za većinu aplikacija.



Slika 12.3 Prikaz matrice konfuzije za random forest classifier

Random forest classifier (RF) pokazuje značajno poboljšanje u performansama otkrivanja prijevara u usporedbi s modelom logističke regresije. I lažno pozitivni (FP) i lažno negativni (FN) znatno su smanjeni, što dovodi do viših rezultata prisjećanja i preciznosti. To ukazuje da je RF učinkovitiji u ispravnom identificiranju lažnih transakcija dok smanjuje broj legitimnih transakcija koje su pogrešno označene kao lažne.

RF klasifikator smanjuje broj lažno pozitivnih (FP) sa 177 na 28, što predstavlja značajno smanjenje od 82%. To implicira da je RF selektivniji u označavanju transakcija kao lažnih, smanjujući neugodnosti za legitimne kupce i radno opterećenje za istražitelje prijevara.

RF klasifikator također smanjuje broj lažno negativnih rezultata (FN) sa 48 na 23. To znači da je RF učinkovitiji u prepoznavanju stvarnih lažnih transakcija, sprječavajući da ostanu neotkrivene i uzrokuju financijske gubitke.

Rezultat preciznosti, koji mjeri udio pozitivnih predviđanja koja su zapravo točna, povećava se sa 68% na 93% s RF-om. To ukazuje da je RF sigurniji u svoja predviđanja lažnih transakcija i da je veći udio njegovih označenih transakcija doista lažan.

Rezultat prisjećanja, koji mjeri udio stvarno pozitivnih slučajeva koji su ispravno identificirani, povećava se s 88% na 95% s RF-om. To ukazuje da je RF učinkovitiji u identificiranju svih lažnih transakcija, smanjujući broj neotkrivenih lažnih slučajeva.

Tablica 12.3 *Prikaz izvješća o klasifikaciji za random forest*

Matrica konfuzije	Precision	Recall	F1- score	Support
0	0.99	0.99	0.99	1547
1	0.95	0.96	0.95	422
Accuracy			0.98	1969
Macro avg	0.97	0.97	0.97	1969
Weighted avg	0.98	0.98	0.98	1969

Accuracy	0.974
ROC AUC Score	0.967
Precision	0.926
Recall	0.955
F1 Score	0.941

12.3 XGB Classifier

XGBoost ili Extreme Gradient Boosting algoritam je strojnog učenja koji koristi skupne metode (engl. Ensamble methods) za treniranje modela. XGBoost se može koristiti za zadatke klasifikacije postavljanjem parametra objektiva na "multi:softmax" ili "binary:logistic" ovisno o tome je li problem klasifikacije multi-class ili binarni. Nakon što se XGBoost klasifikator obuči, može se koristiti za predviđanje novih podataka pozivanjem metode predict(). Metoda predict() vraća vektor predviđenih oznaka klasa za nove podatke.

XGBoost radi treniranjem slijeda stabala odlučivanja, pri čemu je svako stablo osposobljeno za ispravljanje pogrešaka prethodnog stabla. Ovaj se postupak ponavlja sve dok model ne postigne željenu razinu performansi.

12.3.1 Podešavanje hiperparametara za XGB klasifikator

Best Hyperparameters	'colsample_bytree'	'learning_rate'	'max_depth'	'n_estimators'	'subsample'
	0.5	0.5	4	200	0.5

colsample_bytree kontrolira frakciju stupaca za slučajni odabir za svako stablo u modelu XGBoost. Vrijednost od 0.5 znači da će se za svako stablo slučajno odabrati 50% stupaca. To može pomoći u smanjenju pretjerivanja sprječavajući model da se pretjerano prilagodi određenim značajkama.

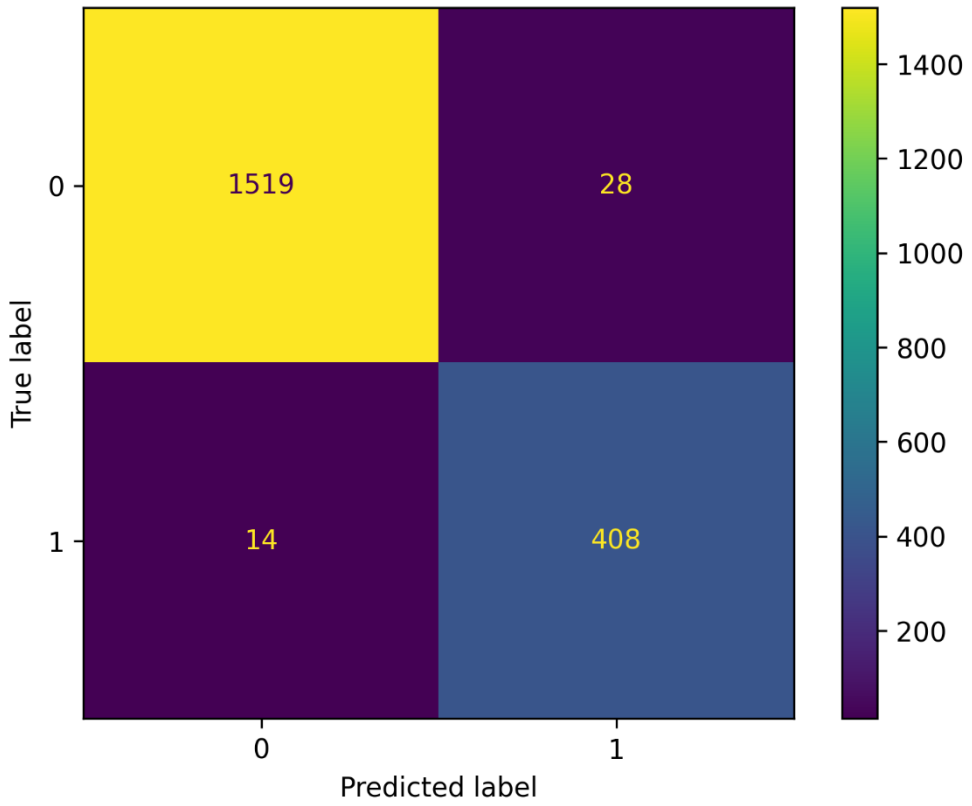
learning_rate kontrolira veličinu koraka algoritma gradijentnog pada koji se koristi za treniranje modela XGBoost. Viša vrijednost stope učenja znači da će se model agresivnije ažurirati, što može dovesti do bržeg konvergencije. Međutim, veća stopa učenja može model učiniti osjetljivijim na pretjerivanje.

max_depth kontrolira maksimalnu dubinu stabala u modelu XGBoost. Dublje stablo će omogućiti modelu da nauči složenije obrasce u podacima, ali to može također dovesti do pretjerivanja.

n_estimators kontrolira broj stabala za izgradnju u modelu XGBoost. Veći broj stabala općenito će dovesti do točnijeg modela, ali može također učiniti model računski zahtjevnijim.

Subsample kontrolira frakciju primjeraka za obuku za povlačenje za svako ponavljanje algoritma gradijentnog pada. Vrijednost od 0.5 znači da će se za svako ponavljanje koristiti 50%

primjeraka za obuku. To može pomoći u smanjenju pretjerivanja sprječavajući model da se pretjerano prilagodi specifičnim primjercima za obuku



Slika 12.4 Prikaz matrice konfuzije za XGBoost classifier

Model je točno identificirao 408 slučaja PRIJEVARE od 422. To ukazuje da je model učinkovit u identificiranju stvarnih lažnih transakcija.

Lažno pozitivno (FP): Model je netočno označio 28 transakciju kao PREVARU od 1547, što pokazuje napredak u odnosu na prethodni model

Lažno negativan (FN): Model nije uspio identificirati 14 slučajeva PRIJEVARE, tretirajući ih kao NE-prijevare, što je isto poboljšanje na prethodni model

Usporedba klasifikatora Random Forest i XGBoost za otkrivanje prijevara prikazuje da oba klasifikatora imaju vrlo slične performanse. Oba imaju istu preciznost, recall i F1 Score, a također imaju isti broj TP i FN rezultata. Jedina razlika je u tome što klasifikator XGBoost ima nešto veći broj TN rezultata i nešto manji broj FP rezultata.

To sugerira da klasifikator XGBoost je selektivniji u prepoznavanju prijevarnih transakcija, dok klasifikator Random Forest je više osjetljiv. Drugim riječima, klasifikator XGBoost je manje vjerojatno da će pogrešno klasificirati legitimne transakcije kao prijevarne, dok klasifikator Random Forest je vjerojatnije da će ispravno identificirati prijevarne transakcije.

Odabir klasifikatora koji je bolji za otkrivanje prijevara ovisi o specifičnim potrebama primjene. Ako je važno minimirati broj lažno pozitivnih rezultata (tj. legitimnih transakcija koje su pogrešno klasificirane kao prijevarne), klasifikator XGBoost može biti bolji izbor. Ako je važno maksimizirati broj istinitih pozitivnih rezultata (tj. prijevarnih transakcija koje su ispravno identificirane), klasifikator Random Forest može biti bolji izbor

Tablica 12.4 Prikaz izvješća o klasifikaciji za XGBoost

Matrica konfuzije	*Precision*	*Recall*	*F1- score*	*Support*
0	*0.99*	*0.99*	*0.99*	*1547*
1	*0.95*	*0.96*	*0.95*	*422*
Accuracy			*0.98*	*1969*
Macro avg	*0.97*	*0.97*	*0.97*	*1969*
Weighted avg	*0.98*	*0.98*	*0.98*	*1969*

Accuracy	*0.9802*
ROC AUC Score	*0.9727*
Precision	*0.9484*
Recall	*0.9597*
F1 Score	*0.9541*

12.4 Usporedba preformansi pojedinačnih algoritama

	Model	*Accuracy*	*Precision*	*Recall*	*F1 Score*	*ROC AUC Score*
0	*Random Forest*	*0.974606*	*0.928571*	*0.954976*	*0.941589*	*0.967469*
1	*XGBoost*	*0.980193*	*0.948478*	*0.959716*	*0.954064*	*0.972747*
2	*Logistic Regression*	*0.887252*	*0.683824*	*0.881517*	*0.770186*	*0.885167*

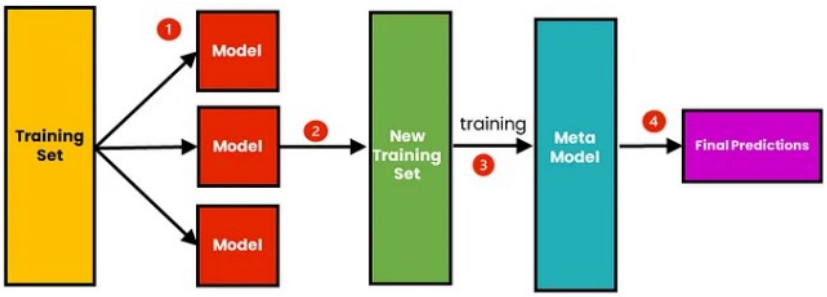
Na osnovi odabranih evaluacijskih metrika prikazanih u tablici najbolje preformanse pokazuje XGBoost algoritam.

13. Stacking ansambl

Slaganje je snažna skupna strategija učenja (engl. Ensabmle learning strategy) u strojnom učenju koja kombinira predviđanja brojnih osnovnih modela kako bi se dobilo konačno predviđanje s boljom izvedbom. Također je poznat kao naslagana generalizacija (engl. Stacked generalization).

Slaganje je strategija strojnog učenja koja kombinira predviđanja brojnih osnovnih modela, također poznatih kao modeli prve razine (engl. First-level models), kako bi se dobilo konačno predviđanje. To uključuje obuku brojnih osnovnih modela na istom skupu podataka za obuku, zatim unošenje njihovih predviđanja u model više razine, također poznat kao meta-model ili model druge razine, kako bi se napravilo konačno predviđanje. Glavna ideja iza slaganja je kombiniranje predviđanja različitih osnovnih modela kako bi se dobila izvanrednija prediktivna izvedba od korištenja jednog modela [13].

U radu su se kao osnovni modeli koristili logistička regresija, random forest i xgboost algoritmi, te se na temelju njih napravio stacking ansambl. Kombinacijom predviđanja logističke regresije, radnom foresta i XGBoosta, skup slaganja (engl. Stacking ansamble) često može postići bolju izvedbu od bilo kojeg pojedinačnog modela. To je zato što meta-model može naučiti kombinirati prednosti svakog modela i dati točnija predviđanja.



Slika 13.1 Prikaz stacking ansambla [2]

13.1 Preformanse meta modela

Tablica 13.1 Preformanse stacking ansambla sa logističkom regresijom kao meta modelom

Meta model	Accuracy	Precision	Recall	F1 Score	ROC AUC Score
Logistic regression	0.974099	0.926437	0.954976	0.940490	0.967146

Tablica 13.2 Preformanse stacking ansambla sa random forest algoritmom kao meta modelom

Meta model	Accuracy	Precision	Recall	F1 Score	ROC AUC Score
Random forest	0.980193	0.954869	0.952607	0.953737	0.970162

Tablica 13.3 Preformanse stacking ansambla sa random forest algoritmom kao meta modelom

Meta model	Accuracy	Precision	Recall	F1 Score	ROC AUC Score
XGBoost	0.980193	0.954869	0.952607	0.953737	0.970162

13.2 Usporedba ansambla algoritama sa pojedinačnim

Nakon izrade stacking algoritma sa tri meta modela, dati ćemo usporedbu spomenutog algoritma sa XGBoost modelom koji je dao najbolje rezultate kod pojedinačnih modela.

Vrste algoritma/metode	Meta model	Accuracy	Precision	Recall	F1 Score	ROC AUC Score
Stacking ansamble	XGBoost	0.980193	0.954869	0.952607	0.953737	0.970162
XGBoost	-	0.980193	0.942263	0.966825	0.954386	0.975332

Pojedinačni XGBoost model ima bolje pamćenje, F1 rezultat i ROC AUC rezultat od skupa za slaganje s XGBoost meta modelom. To je zato što model XGBoost može naučiti složenije obrasce u podacima, što mu omogućuje učinkovitije identificiranje pozitivnih i negativnih slučajeva. Međutim, ako je ukupna prediktivna izvedba važnija, tada je model Stacking ensemble bolji izbor.

Ansambl slaganja s XGBoost meta modelom moćna je tehnika koja može poboljšati izvedbu višestrukih osnovnih modela. Međutim, važno je napomenuti da ova tehnika ne nadmašuje uvijek jedan, dobro podešen model. U ovom slučaju, model XGBoost jednostavno je u stanju naučiti obrasce u podacima učinkovitije nego što to može skup slaganja.

14. Zaključak

U ovom radu istražila se primjenu algoritama strojnog učenja za detekciju i određivanje prijevarnih aktivnosti na blockchain mrežama. Analizirali su različite oblike prijevara koje prijete blockchain tehnologiji te istražio kako algoritmi strojnog učenja mogu doprinijeti očuvanju njezinog integriteta. Također, razmotrili su se mogući izazovi i ograničenja u primjeni ovih tehnika te pružio pregled relevantnih istraživanja i pristupa u ovom području. Nakon detaljne statističke analize na prikupljenom skupu podataka i izvršenog treniranja različitih algoritama strojnog učenja primjenom unakrsne validacije s nasumičnim pretraživanjem hiperparametara, procijenilo se koji od algoritama daje najveću točnost detekcije i odabrani algoritam se koristio za izradu metoda ansambla u cilju povećanja točnosti detekcije.

Model XGBoost ima više vrijednosti za metriku prisjećanja, F1 scorea i ROC AUC. To sugerira da je model XGBoost nešto bolji u predviđanju pozitivnih primjera i razlikovanju između pozitivnih i negativnih primjera. Međutim, model Stacking ensemble ima više vrijednosti za metriku točnosti I preciznosti. To sugerira da je model Stacking ensemble nešto bolji u ukupnoj prediktivna izvedbi. U konačnici, izbor modela koji se koristi ovisi o specifičnim zahtjevima aplikacije. Ako su preciznost i ROC AUC osobito važni, tada je model XGBoost možda bolji izbor. Međutim, ako je ukupna prediktivna izvedba važnija, tada je model Stacking ensemble možda bolji izbor.

Sljedeće što bih volio sa ovim algoritmom jest dati mu neku komercijalnu svrhu, odnosno razviti neki komercijalni proizvod. Razmišljao sam o primjeni u zdravstvenoj zaštiti za detekciju prijevara kako bi se spriječilo prekomjerno naplaćivanje računa kao i prijevare sa osiguranjem.

Konačno, ovaj rad ima za cilj doprinijeti boljem razumijevanju važnosti primjene algoritama strojnog učenja za sigurnost blockchain tehnologije i pružiti smjernice za buduća istraživanja i razvoj u ovoj sferi.

15. Literatura

[1] Cryptocurrency fraud, Članak sa Interneta, Constantine Canon

[2] Tahmid Hasan Pranto, Kazi Tamzid Akhter Md Hasib, Tahsinur Rahma, AKM Bahalul Haque, A.K.M. Najmul Islam and Rashedur M. Rahman, Blockchain and Machine Learning for Fraud Detection: A Privacy-Preservingand Adaptive Incentive Based Approach, IEEE

[3] TRM, Illicit Crypto Ecosystem Report. Comprehensive Guide to Illicit Finance Risks in Crypto, TRM Labs

[4]Cointelegraph, What is a cryptocurrency mixer and how does it work

[5] Thanh Tam Nguyen, Example-based explanations for streaming fraud detection on graphs, ScienceDirect

[6]Ethereum Fraud Detection Dataset, Kaggle, https://www.kaggle.com/datasets/vagifa/ethereum-frauddetection-dataset

[7] Madhuparna Bhowmik, Tulasi Sai Siri Chandana, Dr. Bhawana Rudra, Comparative Study of Machine Learning Algorithms for Fraud Detection in Blockchain, IEEE

[8] Dejan Varmedja, Mirjana Karanovic, Srdjan Sladojevic, Marko Arsenovic, Andras Anderla, Credit Card Fraud Detection - Machine Learning methods,

[9] Chainalysis, The 2023 Crypto Crime Report, Chainalysis

[10] Massimo Bartoletti, Barbara Pes, Sergio Serusi, Massimo Bartoletti, Barbara Pes, Sergio Serusi, Arxiv

[11] Baran Kılıc ,, Alper Sen and Can Ozturan, Fraud Detection in Blockchains using Machine Learning, IEEE

[12] Anuska Rakshit, Fraud Detection: A review on Blockchain, Research Gate

[13] Stacking to Improve Model Performance: A Comprehensive Guide on Ensemble Learning in Python Medium, Medium članak

16. Popis slika

17. Sažetak

U ovom diplomskom radu istražena je primjena algoritama strojnog učenja za detekciju i sprječavanje prijevarnih aktivnosti na blockchain mrežama. Autor je analizirao različite oblike prijevare koji prijete blockchain tehnologiji te istražio kako algoritmi strojnog učenja mogu doprinijeti očuvanju njezine integritete. Nakon detaljne statističke analize i treniranja različitih algoritama strojnog učenja, autor je procijenio koji od algoritama daje najveću točnost detekcije i odabrani algoritam koristio za izradu ansambla metode u cilju povećanja točnosti detekcije. Konačno, ovaj rad ima za cilj doprinijeti boljem razumijevanju važnosti primjene algoritama strojnog učenja za sigurnost blockchain tehnologije i pružiti smjernice za buduća istraživanja i razvoj.

Ključne riječi: blockchain, strojno učenje, detekcija prijevare, sigurnost, ansambl metoda, XGB klasifikator, random forest klasifikator

18. Summary

In this thesis, the application of machine learning algorithms for the detection and prevention of fraudulent activities on blockchain networks is investigated. The author analyzed the various forms of fraud that threaten blockchain technology and investigated how machine learning algorithms can contribute to preserving its integrity. After a detailed statistical analysis and training of different machine learning algorithms, the author evaluated which of the algorithms gives the highest detection accuracy and used the selected algorithm to create an ensemble of methods in order to increase the detection accuracy. Finally, this paper aims to contribute to a better understanding of the importance of applying machine learning algorithms to the security of blockchain technology and to provide guidance for future research and development.

Keywords: blockchain, machine learning, fraud detection, security, ensemble method, XGB classifier, random forest classifier

DODATAK A

A1. Programski kod

```
LIBRARIES IMPORT

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PowerTransformer
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix, roc_auc_score, roc_curve, auc,
classification_report, ConfusionMatrixDisplay
from sklearn.model_selection import GridSearchCV
import pickle

DATA READING
df = pd.read_csv('transaction_dataset.csv', index_col=0)
print(df.shape)
df.head()
# Ommit first two columns (Index, Adress)
df = df.iloc[:,2:]



DATA INFORMATION AND EXPLORATION
df.info()

# Turn object variables into 'category' dtype for more computation efficiency
categories = df.select_dtypes('O').columns.astype('category')
df[categories]
# Inspect categoricals
for i in df[categories].columns:
    print(f'The categorical column --{i}-- has --{len(df[i].value_counts())}-
- unique values')
# Inspect numericals
numericals = df.select_dtypes(include=['float','int']).columns
df[numericals].describe()
# Inspect features variance
df[numericals].var()
# Inspect target distribution
print(df['FLAG'].value_counts())

pie, ax = plt.subplots(figsize=[15,10])
labels = ['Non-fraud', 'Fraud']
```

```
colors = ['#f9ae35', '#f64e38']
plt.pie(x = df['FLAG'].value_counts(), autopct='%.2f%%', explode=[0.02]*2,
labels=labels, pctdistance=0.5, textprops={'fontsize': 14}, colors = colors)
plt.title('Target distribution')
plt.show()
numeric_df = df.select_dtypes(include=[np.number])
corr = numeric_df.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
with sns.axes_style('white'):
    fig, ax = plt.subplots(figsize=(18, 10))
    sns.heatmap(corr, mask=mask, annot=False, cmap='CMRmap', center=0,
square=True)

DATA CLEANING

# Visualize missings pattern of the dataframe
plt.figure(figsize=(12,6))
sns.heatmap(df.isnull(), cbar=False)
plt.show()
# Drop the two categorical features
df.drop(df[categories], axis=1, inplace=True)
# Replace missings of numerical variables with median
df.fillna(df.median(), inplace=True)
# Visualize missings pattern of the dataframe
print(df.shape)
plt.figure(figsize=(12,6))
sns.heatmap(df.isnull(), cbar=False)
plt.show()
# Filtering the features with 0 variance
no_var = df.var() == 0
print(df.var()[no_var])
print('\n')

# Drop features with 0 variance --- these features will not help in the
performance of the model
df.drop(df.var()[no_var].index, axis = 1, inplace = True)
print(df.var())
print(df.shape)
df.info()

# Recheck the Correlation matrix
corr = df.corr()

mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)]=True
with sns.axes_style('white'):
    fig, ax = plt.subplots(figsize=(18,10))
    sns.heatmap(corr,  mask=mask, annot=False, cmap='CMRmap', center=0,
linewidths=0.1, square=True)
#Drop one of those highly correlated features
drop = ['total transactions (including tnx to create contract', 'total ether
sent contracts', 'max val sent to contract', ' ERC20 avg val rec',
        ' ERC20 avg val rec',' ERC20 max val rec', ' ERC20 min val rec', '
ERC20 uniq rec contract addr', 'max val sent', ' ERC20 avg val sent',
        ' ERC20 min val sent', ' ERC20 max val sent', ' Total ERC20 tnxs',
'avg value sent to contract', 'Unique Sent To Addresses',
```

```
        'Unique Received From Addresses', 'total ether received', ' ERC20
uniq sent token name', 'min value received', 'min val sent', ' ERC20 uniq rec
addr' ]
df.drop(drop, axis=1, inplace=True)

# Recheck the Correlation matrix
corr = df.corr()

mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)]=True
with sns.axes_style('white'):
    fig, ax = plt.subplots(figsize=(18,10))
    sns.heatmap(corr,  mask=mask, annot=False, cmap='CMRmap', center=0,
linewidths=0.1, square=True)
columns = df.columns
columns
# Investigate the distribution of our features using boxplots
b=20

fig, axes = plt.subplots(6, 3, figsize=(14, 14), constrained_layout =True)
plt.subplots_adjust(wspace = 0.7, hspace=0.8)
plt.suptitle("Distribution of features",y=0.95, size=18, weight='bold')

ax = sns.boxplot(ax = axes[0,0], data=df, x=columns[1])
ax.set_title(f'Distribution of {columns[1]}')

ax1 = sns.boxplot(ax = axes[0,1], data=df, x=columns[2])
ax1.set_title(f'Distribution of {columns[2]}')

ax2 = sns.boxplot(ax = axes[0,2], data=df, x=columns[3])
ax2.set_title(f'Distribution of {columns[3]}')

ax3 = sns.boxplot(ax = axes[1,0], data=df, x=columns[4])
ax3.set_title(f'Distribution of {columns[4]}')

ax4 = sns.boxplot(ax = axes[1,1], data=df, x=columns[5])
ax4.set_title(f'Distribution of {columns[5]}')

ax5 = sns.boxplot(ax = axes[1,2], data=df, x=columns[6])
ax5.set_title(f'Distribution of {columns[6]}')

ax6 = sns.boxplot(ax = axes[2,0], data=df, x=columns[7])
ax6.set_title(f'Distribution of {columns[7]}')

ax7 = sns.boxplot(ax = axes[2,1], data=df, x=columns[8])
ax7.set_title(f'Distribution of {columns[8]}')

ax8 = sns.boxplot(ax = axes[2,2], data=df, x=columns[9])
ax8.set_title(f'Distribution of {columns[9]}')

ax9 = sns.boxplot(ax = axes[3,0], data=df, x=columns[10])
ax9.set_title(f'Distribution of {columns[10]}')

ax10 = sns.boxplot(ax = axes[3,1], data=df, x=columns[11])
ax10.set_title(f'Distribution of {columns[11]}')

ax11 = sns.boxplot(ax = axes[3,2], data=df, x=columns[12])
```

```
ax11.set_title(f'Distribution of {columns[12]}')

ax12 = sns.boxplot(ax = axes[4,0], data=df, x=columns[13])
ax12.set_title(f'Distribution of {columns[13]}')

ax13 = sns.boxplot(ax = axes[4,1], data=df, x=columns[14])
ax13.set_title(f'Distribution of {columns[14]}')

ax14 = sns.boxplot(ax = axes[4,2], data=df, x=columns[15])
ax14.set_title(f'Distribution of {columns[15]}')

ax15 = sns.boxplot(ax = axes[5,0], data=df, x=columns[16])
ax15.set_title(f'Distribution of {columns[16]}')

ax16 = sns.boxplot(ax = axes[5,1], data=df, x=columns[17])
ax16.set_title(f'Distribution of {columns[17]}')

ax17 = sns.boxplot(ax = axes[5,2], data=df, x=columns[18])
ax17.set_title(f'Distribution of {columns[18]}')

plt.show()
# Some features present a small distribution
for i in df.columns[1:]:
    if len(df[i].value_counts()) < 10:
        print(f'The column {i} has the following distribution:
\n{df[i].value_counts()}')
        print('=====================================')
drops = ['min value sent to contract', ' ERC20 uniq sent addr.1']
df.drop(drops, axis=1, inplace=True)
print(df.shape)
df.head()
```

DATA PREPARATION

```
y = df.iloc[:, 0]
X = df.iloc[:, 1:]
print(X.shape, y.shape)
# Split into training (80%) and testing set (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 123)
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
# Normalize the training features
norm = PowerTransformer()
norm_train_f = norm.fit_transform(X_train)
norm_df = pd.DataFrame(norm_train_f, columns=X_train.columns)
norm_df
# Distribution of features after log transformation

b=20

fig, axes = plt.subplots(6, 3, figsize=(14, 14), constrained_layout =True)
plt.subplots_adjust(wspace = 0.7, hspace=0.8)
axes[-1, -1].axis('off') # hide axes
```

```
axes[-1, -2].axis('off') # hide axes
plt.suptitle("Distribution of features after log",y=0.95, family='Sherif',
size=18, weight='bold')

ax = sns.boxplot(ax = axes[0,0], data=norm_df, x=norm_df.columns[0])
ax.set_title(f'Distribution of {norm_df.columns[0]}')

ax1 = sns.boxplot(ax = axes[0,1], data=norm_df, x=norm_df.columns[1])
ax1.set_title(f'Distribution of {norm_df.columns[1]}')

ax2 = sns.boxplot(ax = axes[0,2], data=norm_df, x=norm_df.columns[2])
ax2.set_title(f'Distribution of {norm_df.columns[2]}')

ax3 = sns.boxplot(ax = axes[1,0], data=norm_df, x=norm_df.columns[3])
ax3.set_title(f'Distribution of {norm_df.columns[3]}')

ax4 = sns.boxplot(ax = axes[1,1], data=norm_df, x=norm_df.columns[4])
ax4.set_title(f'Distribution of {norm_df.columns[4]}')

ax5 = sns.boxplot(ax = axes[1,2], data=norm_df, x=norm_df.columns[5])
ax5.set_title(f'Distribution of {norm_df.columns[5]}')

ax6 = sns.boxplot(ax = axes[2,0], data=norm_df, x=norm_df.columns[6])
ax6.set_title(f'Distribution of {norm_df.columns[6]}')

ax7 = sns.boxplot(ax = axes[2,1], data=norm_df, x=norm_df.columns[7])
ax7.set_title(f'Distribution of {norm_df.columns[7]}')

ax8 = sns.boxplot(ax = axes[2,2], data=norm_df, x=norm_df.columns[8])
ax8.set_title(f'Distribution of {norm_df.columns[8]}')

ax9 = sns.boxplot(ax = axes[3,0], data=norm_df, x=norm_df.columns[9])
ax9.set_title(f'Distribution of {norm_df.columns[9]}')

ax10 = sns.boxplot(ax = axes[3,1], data=norm_df, x=norm_df.columns[10])
ax10.set_title(f'Distribution of {norm_df.columns[10]}')

ax11 = sns.boxplot(ax = axes[3,2], data=norm_df, x=norm_df.columns[11])
ax11.set_title(f'Distribution of {norm_df.columns[11]}')

ax12 = sns.boxplot(ax = axes[4,0], data=norm_df, x=norm_df.columns[12])
ax12.set_title(f'Distribution of {norm_df.columns[12]}')

ax13 = sns.boxplot(ax = axes[4,1], data=norm_df, x=norm_df.columns[13])
ax13.set_title(f'Distribution of {norm_df.columns[13]}')

ax14 = sns.boxplot(ax = axes[4,2], data=norm_df, x=norm_df.columns[14])
ax14.set_title(f'Distribution of {norm_df.columns[14]}')

ax15 = sns.boxplot(ax = axes[5,0], data=norm_df, x=norm_df.columns[15])
ax15.set_title(f'Distribution of {norm_df.columns[15]}')

plt.show()
HANDLING THE IMBALANCE
oversample = SMOTE()
print(f'Shape of the training before SMOTE: {norm_train_f.shape,
y_train.shape}')
```

```
x_tr_resample, y_tr_resample = oversample.fit_resample(norm_train_f, y_train)
print(f'Shape of the training after SMOTE: {x_tr_resample.shape,
y_tr_resample.shape}')
# Target distribution before SMOTE
non_fraud = 0
fraud = 0

for i in y_train:
    if i == 0:
        non_fraud +=1
    else:
        fraud +=1

# Target distribution after SMOTE
no = 0
yes = 1

for j in y_tr_resample:
    if j == 0:
        no +=1
    else:
        yes +=1

print(f'BEFORE OVERSAMPLING \n \tNon-frauds: {non_fraud} \n \tFauds:
{fraud}')
print(f'AFTER OVERSAMPLING \n \tNon-frauds: {no} \n \tFauds: {yes}')

MODELING

**Logistic Regression**

LR = LogisticRegression(random_state=42)
LR.fit(x_tr_resample, y_tr_resample)

# Transform test features
norm_test_f = norm.transform(X_test)

preds = LR.predict(norm_test_f)
print(y_test.shape)
y_test.value_counts()

from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.model_selection import GridSearchCV


y_pred = LR.predict(norm_test_f)
disp = ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
disp.plot()
plt.show()

accuracy = accuracy_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)
```

```
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("ROC AUC Score:", roc_auc)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

# Define the parameter grid
param_grid = {
    'penalty': ['l1', 'l2'],
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'solver': ['liblinear', 'saga']
}

# Create the GridSearchCV object
grid_search = GridSearchCV(estimator=LogisticRegression(random_state=42),
param_grid=param_grid, cv=5)

# Fit the grid search to the training data
grid_search.fit(x_tr_resample, y_tr_resample)

# Get the best hyperparameters
best_params = grid_search.best_params_
print("Best Hyperparameters:", best_params)


**Random Forest Classifier**

RF = RandomForestClassifier(random_state=42)
RF.fit(x_tr_resample, y_tr_resample)
preds_RF = RF.predict(norm_test_f)
y_pred = LR.predict(norm_test_f)
disp = ConfusionMatrixDisplay.from_predictions(y_test, preds_RF)
disp.plot()
plt.show()

print(classification_report(y_test, preds_RF))
print(confusion_matrix(y_test, preds_RF))

accuracy_RF = accuracy_score(y_test, preds_RF)
precision_RF = precision_score(y_test, preds_RF)
recall_RF = recall_score(y_test, preds_RF)
f1_RF = f1_score(y_test, preds_RF)
roc_auc_RF = roc_auc_score(y_test, preds_RF)

print("Accuracy (RF):", accuracy_RF)
print("Precision (RF):", precision_RF)
print("Recall (RF):", recall_RF)
print("F1 Score (RF):", f1_RF)
print("ROC AUC Score (RF):", roc_auc_RF)
```

```
# Define the parameter grid
param_grid_RF = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2']
}

# Create the GridSearchCV object
grid_search_RF =
GridSearchCV(estimator=RandomForestClassifier(random_state=42),
param_grid=param_grid_RF, cv=5)

# Fit the grid search to the training data
grid_search_RF.fit(x_tr_resample, y_tr_resample)

# Get the best hyperparameters
best_params_RF = grid_search_RF.best_params_
print("Best Hyperparameters (RF):", best_params_RF)

**XGB Classifier**

xgb_c = xgb.XGBClassifier(random_state=42)
xgb_c.fit(x_tr_resample, y_tr_resample)
preds_xgb = xgb_c.predict(norm_test_f)

disp = ConfusionMatrixDisplay.from_predictions(y_test, preds_xgb)
disp.plot()
plt.show()

accuracy_xgb = accuracy_score(y_test, preds_xgb)
precision_xgb = precision_score(y_test, preds_xgb)
recall_xgb = recall_score(y_test, preds_xgb)
f1_xgb = f1_score(y_test, preds_xgb)
roc_auc_xgb = roc_auc_score(y_test, preds_xgb)

print("Accuracy (XGB):", accuracy_xgb)
print("Precision (XGB):", precision_xgb)
print("Recall (XGB):", recall_xgb)
print("F1 Score (XGB):", f1_xgb)
print("ROC AUC Score (XGB):", roc_auc_xgb)

print(classification_report(y_test, preds_xgb))
print(confusion_matrix(y_test, preds_xgb))

**Hyperparameters tuning for XGB Classifier**

params_grid = {'learning_rate':[0.01, 0.1, 0.5],
               'n_estimators':[100,200],
               'subsample':[0.3, 0.5, 0.9],
                'max_depth':[2,3,4],
                'colsample_bytree':[0.3,0.5,0.7]}

grid = GridSearchCV(estimator=xgb_c, param_grid=params_grid,
scoring='recall', cv = 10, verbose = 0)
```

```
grid.fit(x_tr_resample, y_tr_resample)
print(f'Best params found for XGBoost are: {grid.best_params_}')
print(f'Best recall obtained by the best params: {grid.best_score_}')

preds_best_xgb = grid.best_estimator_.predict(norm_test_f)
accuracy_best_xgb = accuracy_score(y_test, preds_best_xgb)
precision_best_xgb = precision_score(y_test, preds_best_xgb)
recall_best_xgb = recall_score(y_test, preds_best_xgb)
f1_best_xgb = f1_score(y_test, preds_best_xgb)
roc_auc_best_xgb = roc_auc_score(y_test, preds_best_xgb)

print("Accuracy (Best XGB):", accuracy_best_xgb)
print("Precision (Best XGB):", precision_best_xgb)
print("Recall (Best XGB):", recall_best_xgb)
print("F1 Score (Best XGB):", f1_best_xgb)
print("ROC AUC Score (Best XGB):", roc_auc_best_xgb)

print(classification_report(y_test, preds_best_xgb))
print(confusion_matrix(y_test, preds_best_xgb))

disp = ConfusionMatrixDisplay.from_predictions(y_test, preds_best_xgb)
disp.plot()
plt.show()

# Plotting AUC for untuned XGB Classifier
probs = xgb_c.predict_proba(norm_test_f)
pred = probs[:,1]
fpr, tpr, threshold = roc_curve(y_test, pred)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(12,8))
plt.title('ROC for tuned XGB Classifier')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0,1], [0,1], 'r--')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.grid(True)
plt.savefig('roc_plot.png', dpi=300, bbox_inches='tight')  # Export the plot
as a PNG file
plt.show()

import pandas as pd
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, roc_auc_score

# Logistic Regression
lr_preds = LR.predict(norm_test_f)
lr_accuracy = accuracy_score(y_test, lr_preds)
lr_precision = precision_score(y_test, lr_preds)
lr_recall = recall_score(y_test, lr_preds)
lr_f1_score = f1_score(y_test, lr_preds)
lr_roc_auc = roc_auc_score(y_test, lr_preds)

# Random Forest
rf_preds = RF.predict(norm_test_f)
rf_accuracy = accuracy_score(y_test, rf_preds)
```

```
rf_precision = precision_score(y_test, rf_preds)
rf_recall = recall_score(y_test, rf_preds)
rf_f1_score = f1_score(y_test, rf_preds)
rf_roc_auc = roc_auc_score(y_test, rf_preds)

# XGBoost
xgb_preds = xgb_c.predict(norm_test_f)
xgb_accuracy = accuracy_score(y_test, xgb_preds)
xgb_precision = precision_score(y_test, xgb_preds)
xgb_recall = recall_score(y_test, xgb_preds)
xgb_f1_score = f1_score(y_test, xgb_preds)
xgb_roc_auc = roc_auc_score(y_test, xgb_preds)

# Create a dictionary with the performance metrics
data = {
    'Model': ['Random Forest', 'XGBoost', 'Logistic Regression'],
    'Accuracy': [rf_accuracy, xgb_accuracy, lr_accuracy],
    'Precision': [rf_precision, xgb_precision, lr_precision],
    'Recall': [rf_recall, xgb_recall, lr_recall],
    'F1 Score': [rf_f1_score, xgb_f1_score, lr_f1_score],
    'ROC AUC Score': [rf_roc_auc, xgb_roc_auc, lr_roc_auc]
}

# Create a DataFrame from the dictionary
df = pd.DataFrame(data)

# Display the DataFrame
Df

**Stacking ansamble**

# Normalize the test features
norm_test_f = norm.transform(X_test)

# Oversampling using SMOTE
oversample = SMOTE(random_state=123)
X_train_res, y_train_res = oversample.fit_resample(norm_train_f, y_train)

**Using a XGBoost as meta model**

# Initialize the base models
model1 = LogisticRegression(random_state=123)
model2 = RandomForestClassifier(random_state=123)
model3 = XGBClassifier(random_state=123)

# Initialize the meta-model
meta_model = XGBClassifier(random_state=123)

# Initialize the StackingCV classifier
stacking_model = StackingCVClassifier(classifiers=[model1, model2, model3],
                                      meta_classifier=meta_model,
                                      random_state=123)

# Fit the model to the oversampled training data
stacking_model.fit(X_train_res, y_train_res)
```

```
# Make predictions
stacking_preds = stacking_model.predict(norm_test_f)

# Evaluate the model
stacking_accuracy = accuracy_score(y_test, stacking_preds)
stacking_precision = precision_score(y_test, stacking_preds)
stacking_recall = recall_score(y_test, stacking_preds)
stacking_f1_score = f1_score(y_test, stacking_preds)
stacking_roc_auc = roc_auc_score(y_test, stacking_preds)

# Create a DataFrame with the evaluation metrics
data = {
    'Metric': ['Accuracy', 'Precision', 'Recall', 'F1 Score', 'ROC AUC
Score'],
    'Stacking': [stacking_accuracy, stacking_precision, stacking_recall,
stacking_f1_score, stacking_roc_auc]
}

df_metrics = pd.DataFrame(data)
df_metrics

**Using Logistic regression as meta model**

# Initialize the meta-model
meta_model = LogisticRegression(random_state=123)

# Initialize the StackingCV classifier
stacking_model = StackingCVClassifier(classifiers=[model1, model2, model3],
                                      meta_classifier=meta_model,
                                      random_state=123)

# Fit the model to the oversampled training data
stacking_model.fit(X_train_res, y_train_res)

# Make predictions
stacking_preds = stacking_model.predict(norm_test_f)

# Evaluate the model
stacking_accuracy = accuracy_score(y_test, stacking_preds)
stacking_precision = precision_score(y_test, stacking_preds)
stacking_recall = recall_score(y_test, stacking_preds)
stacking_f1_score = f1_score(y_test, stacking_preds)
stacking_roc_auc = roc_auc_score(y_test, stacking_preds)

# Create a DataFrame with the evaluation metrics
data = {
    'Metric': ['Accuracy', 'Precision', 'Recall', 'F1 Score', 'ROC AUC
Score'],
    'Stacking': [stacking_accuracy, stacking_precision, stacking_recall,
stacking_f1_score, stacking_roc_auc]
}

df_metrics = pd.DataFrame(data)
df_metrics
```

```
**Using Random forest as meta model**

# Initialize the meta-model
meta_model = RandomForestClassifier(random_state=123)

# Initialize the StackingCV classifier
stacking_model = StackingCVClassifier(classifiers=[model1, model2, model3],
                                      meta_classifier=meta_model,
                                      random_state=123)

# Fit the model to the oversampled training data
stacking_model.fit(X_train_res, y_train_res)

# Make predictions
stacking_preds = stacking_model.predict(norm_test_f)

# Evaluate the model
stacking_accuracy = accuracy_score(y_test, stacking_preds)
stacking_precision = precision_score(y_test, stacking_preds)
stacking_recall = recall_score(y_test, stacking_preds)
stacking_f1_score = f1_score(y_test, stacking_preds)
stacking_roc_auc = roc_auc_score(y_test, stacking_preds)

# Create a DataFrame with the evaluation metrics
data = {
    'Metric': ['Accuracy', 'Precision', 'Recall', 'F1 Score', 'ROC AUC
Score'],
    'Stacking': [stacking_accuracy, stacking_precision, stacking_recall,
stacking_f1_score, stacking_roc_auc]
}

df_metrics = pd.DataFrame(data)

df_metrics
```