

**SVEUČILIŠTE U RIJECI
POMORSKI FAKULTET**

LOREN KONTIĆ

**KLASIFIKACIJSKI MODEL SMJERA KRETANJA CIJENE
KRIPTOVALUTA WEB STRUGANJEM PODATAKA**

DIPLOMSKI RAD

Rijeka, 2023.

**SVEUČILIŠTE U RIJECI
POMORSKI FAKULTET**

**KLASIFIKACIJSKI MODEL SMJERA KRETANJA CIJENE
KRIPTOVALUTA WEB STRUGANJEM PODATAKA
CLASSIFICATION MODEL OF THE DIRECTION OF
CRYPTOCURRENCY PRICE MOVEMENT BY WEB
SCRAPING DATA
DIPLOMSKI RAD**

Kolegij: Sustavi za podršku odlučivanju

Mentor: izv. prof. dr. sc. Jasmin Ćelić

Komentor: dr. sc. Ivan Panić

Student: Loren Kontić

Studijski smjer: Elektroničke i informatičke tehnologije u pomorstvu

JMBAG: 0112073777

Rijeka, ožujak 2023.

Student: Loren Kontić

Studijski program: Elektroničke i Informatičke tehnologije u pomorstvu

JMBAG: 0112073777

IZJAVA O SAMOSTALNOJ IZRADI DIPLOMSKOG RADA

Kojom izjavljujem da sam diplomski rad s naslovom

KLASIFIKACIJSKI MODEL SMJERA KRETANJA CIJENE KRIPTOVALUTA WEB STRUGANJEM PODATAKA izradio samostalno pod mentorstvom izv. prof. dr. sc. Jasmina Čelića te komentorstvom dr. sc. Ivana Panića.

U radu sam primijenio metodologiju izrade stručnog/znanstvenog rada i koristio literaturu koja je navedena na kraju diplomskog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrizirajući naveo u diplomskom radu na uobičajen, standardan način citirao sam i povezao s fusnotama i korištenim bibliografskim jedinicama, te nijedan dio rada ne krši bilo čija autorska prava. Rad je pisan u duhu hrvatskoga jezika.

Student



Loren Kontić

Student: Loren Kontić

Studijski program: Elektroničke i informatičke tehnologije u pomorstvu

JMBAG: 0112073777

IZJAVA STUDENTA – AUTORA
O JAVNOJ OBJAVI OBRANJENOG DIPLOMSKOG RADA

Izjavljujem da kao student – autor diplomskog rada dozvoljavam Pomorskom fakultetu Sveučilišta u Rijeci da ga trajno javno objavi i besplatno učini dostupnim javnosti u cjelovitom tekstu u mrežnom digitalnom repozitoriju Pomorskog fakulteta.

U svrhu podržavanja otvorenog pristupa diplomskim radovima trajno objavljenim u javno dostupnom digitalnom repozitoriju Pomorskog fakulteta, ovom izjavom dajem neisključivo imovinsko pravo iskorištavanja bez sadržajnog, vremenskog i prostornog ograničenja mog diplomskog rada kao autorskog djela pod uvjetima *Creative Commons* licencije CC BY Imenovanje, prema opisu dostupnom na <http://creativecommons.org/licenses/>

Student– autor

Loren Kontić

SAŽETAK

U okviru ovog diplomskog rada analiziran je klasifikacijski model smjera kretanja cijene *Bitcoin* kriptovalute upotrebom rezultata analize sentimenata teksta, koji je ostrugan iz *Bitcointalk* online foruma. Podaci analize sentimenata teksta upotrebljeni su kao dio ulaznog seta podataka za predviđanje rasta ili pada cijene kriptovalute, što je usporedba dnevne cijene zatvaranja uspoređene s sutrašnjom cijenom zatvaranja, što predstavlja izlaznu varijablu binarne klasifikacije. Binarna klasifikacija izvedena je korištenjem triju različitih modela kako bi se mogla usporediti uspješnost klasifikacije između istih. Uspješnost klasifikacije između modela uspoređena je uporabom ROC krivulja, matrica konfuzije te rezultatima ostalih mjera uspješnosti. Modeli korišteni u problemu binarne klasifikacije jesu: Logistička regresija, Klasifikator Ekstremnih Pojačanja Gradijenta i Klasifikator Potpornih Vektora.

Ključne riječi: analiza sentimenta, *Bitcoin*, binarna klasifikacija, klasifikator ekstremnih pojačanja gradijenta, klasifikator potpornih vektora, logistička regresija, matrica konfuzije, ROC krivulja, mjere uspješnosti

ABSTRACT

In the framework of this thesis, a classification model of the Bitcoin cryptocurrency price direction was analyzed using the results of the sentiment analysis of the text scraped from the *Bitcointalk* online forum. Text sentiment analysis data was used as part of the input data set to predict the rise or fall of the cryptocurrency price, which is a comparison of the day's closing price compared to tomorrow's closing price, which represents the output variable of the binary classification. Binary classification was performed using three different models in order to be able to compare the classification performance between them. The classification performance between the models will be compared using ROC curves, confusion matrices and the results of other performance measures. The models used in the binary classification problem are: Logistic Regression, Extreme Gradient Boost Classifier and Support Vector Classifier

Keywords: Bitcoin, binary classification, confusion matrix, extreme gradient boosting classifier, ROC curve, support vector classifier, logistic regression, performance measures, sentiment analysis

SADRŽAJ

SAŽETAK.....	II
ABSTRACT	III
SADRŽAJ.....	IV
1. UVOD	1
1.1. PROBLEM, PREDMET I OBJEKTI ISTRAŽIVANJA.....	2
1.2. RADNA HIPOTEZA.....	2
1.3. SVRHA I CILJEVI ISTRAŽIVANJA	3
1.4. ZNANSTVENE METODE	3
1.5. STRUKTURA RADA.....	3
2. KRIPTOVALUTE	5
2.1. POVIJEST KRIPTOVALUTA	5
2.2. ARHITEKTURA BITCOIN MREŽE	7
2.3. TEHNOLOGIJA ULANČANIH BLOKOVA	8
2.3.1. Rudarski čvorovi mreže	9
2.3.2. Decentralizirani Konsenzus.....	10
2.3.3. Identifikatori bloka	11
2.3.4. Grananje blokova.....	12
2.3.5. Struktura bloka	14
2.3.6. Izrada zaglavlja bloka	16
2.3.7. Rudarenje bloka.....	18
3. ANALIZA SENTIMENATA TEKSTA	21
3.1. OPIS ANALIZE SENTIMENATA TEKSTA.....	21
3.2. TRANSFORMERS BIBLIOTEKA	21
4. ALGORITMI BINARNE KLASIFIKACIJE	27

4.2. LOGISTIČKA REGRESIJA.....	28
4.3. KLASIFIKATOR POTPORNIH VEKTORA.....	36
4.4. KLASIFIKATOR EKSTREMNOG POJAČANJA GRADIJENATA.....	42
4.4.1. Model pojačanja gradijenata	44
4.4.2. Pобољшanja XGBoost modela u odnosu na model pojačanja gradijenata.....	51
4.5. POKAZATELJI USPJEŠNOSTI KLASIFIKACIJE	52
4.5.1. Matrice konfuzije	52
4.5.2. Točnost klasifikacije modela	53
4.5.3. Preciznost klasifikacije modela	53
4.5.4. Odziv klasifikacije modela.....	53
5. PREDVIĐANJE SMJERA KRETANJA CIJENE BITCOIN-A	
UPOTREBOM PYTHON PROGRAMSKOG JEZIKA.....	58
5.1. STRUGANJE TEKSTA I MANIPULACIJA PODACIMA	58
5.1.1. Opis Bitcointalk foruma	58
5.1.2. Struganje podataka i pročišćavanje teksta.....	61
5.1. ANALIZA SENTIMENATA TEKSTA IZ <i>BITCOINTALK</i> FORUMA.....	66
5.2. PREUZIMANJE POVIJESTI CIJENA KRIPTOVALUTE I DOBIVANJE	
IZLAZNE VARIJABLE	70
5.3. IZRADA FUNKCIJA ZA VALIDACIJU METODOM UTVRĐIVANJA	
PROŠLE USPJEŠNOSTI, PRIKAZ ROC KRIVULJA, MATRICA KONFUZIJE TE	
OSTALIH REZULTATA USPJEŠNOSTI	72
5.4. PREDVIĐANJE SMJERA KRETANJA CIJENA BITCOIN-A.....	76
5.4.1. Traženje optimalnih hiperparametra logističke regresije.....	77
5.4.2. Traženje optimalnih hiperparametra klasifikatora ekstremnih pojačanja	
gradijenata.....	78
5.4.3. Traženje optimalnih hiperparametara klasifikatora potpornih vektora....	78
5.5. USPOREDBA USPJEŠNOSTI KLASIFIKACIJSKIH MODELA.....	80
5.5.1. Rezultati uspješnosti logističke regresije.....	80

5.5.2.	Rezultati uspješnosti klasifikatora ekstremnih pojačanja gradijenata.....	83
5.5.3.	Rezultati uspješnosti klasifikatora potpornih vektora	86
6.	ZAKLJUČAK.....	90
	LITERATURA	91
	KAZALO KRATICA.....	96
	POPIS TABLICA	98
	POPIS SLIKA.....	98
	PRILOG 1	101
	PRILOG 2	102
	PRILOG 3	103
	PRILOG 4	103
	PRILOG 5	104
	PRILOG 6	105
	PRILOG 7	106
	PRILOG 8	108
	PRILOG 9	109
	PRILOG 10	109
	PRILOG 11	110
	PRILOG 12	111
	PRILOG 13	112

1. UVOD

Novac je povijesno sredstvo razmjene za određeno dobro. Dobro je širok pojam koji može predstavljati uslugu, hranu, alat, razonodu, informaciju i slično. Koristeći određenu valutu zadane vrijednosti može se kupiti željeno dobro [1]. U 19. i dijelom 20. stoljeća novčane valute koje bi izdavala središnja ustanova bile su najčešće potkrepljene vrijednosti u zlatu. Godine 1971. ukinut je pojam pokrića američkog dolara u zlatu, koji je do tada bio najdominantnija svjetska valuta, na čijoj su vrijednosti imale pokriće i ostale valute. Tim činom rađa se pojam Fiat valuta, čija vrijednost nije potkrepljena nekim materijalnim dobrom, a vrijednost im definira institucija koja ih izdaje. Prije razdoblja Fiat valuta monetarna politika središnje banke dozvoljavala je izdavanje ograničene količine valute, koja je ovisila o protuvrijednosti zaliha zlata. Uvođenjem Fiat valute navedena prepreka nestaje te je omogućeno izdavanje valute bez određenog pokrića. Zbog takve prakse jedna od mogućih posljedica je hiperinflacija, gdje dolazi do ubrzanog pada vrijednosti valute izvan nadzora državnih tijela [2]. Kriptovalute su digitalne valute, koje nemaju fizičkog oblika, niti središnje ustanove koja ih kontrolira te nisu potkrijepljene nekim materijalnim dobrom [3]. Jedna od karakteristika tih valuta je to što se način stvaranja istih bazira na izvršenju određenog računalnog rada koji verificira transakcije. Budući da je taj izvršeni rad jedini način opskrbe novom valutom ne postoji mogućnost od hiperinflacije zbog izdavanja više valute bez potkrijepe kao kod Fiat valuta [4]. Kriptovalute su izrađene u informatičkom svijetu koje se zasnivaju na blokovima programskog koda, odnosno algoritmima koji grade njihovu mrežu. Tehnologija kriptovaluta omogućuje da transakcije istih budu privatne i bez posrednika. Za razliku od Fiat valuta koje su odavno neprebrojive, u kriptovalute su u potpunosti prebrojive pa se prema tome točno zna koliko digitalnih kovanica ukupno postoji u svakom trenutku [4]. Jedan od načina predviđanja pada ili rasta kriptovaluta je upotrebom strojnog učenja, točnije primjenom modela binarne klasifikacije. Takvi modeli su sposobni iz povijesti cijena određene kriptovalute, s određenom uspješnošću, predvidjeti ili klasificirati binarni izlaz. Poboljšanju uspješnosti izvršene klasifikacije najviše doprinose ulazni podaci koji su relevantni za određenu problematiku [5]. Vrijednost dionice određene tvrtke ovisi o protoku novca odnosno poslovnim prilikama. Trgovanje kriptovalutama predstavlja visoki rizik jer za razliku od dionica kompanija rast ili pad njihove vrijednosti nije ničime potkrepljen i nemoguće ga je sa sigurnošću predvidjeti [6]. Neki od poznatih načina za unaprjeđenje predviđanja kretanja cijena kriptovaluta je dodavanje podataka o analizi sentimenta teksta ulaznom skupu podataka. Analiza sentimentata se izvodi pod pretpostavkom

da ako je stav javnosti pozitivan ili negativan te interes javnosti visok ili nizak prema određenom dobru, dobiti će se određena povratna informaciju o smjeru kretanja cijene tog dobra ili kriptovalute. Stav javnosti prema kriptovaluti može se izvući web struganjem komentara na društvenim mrežama. Web struganje teksta podrazumijeva filtriranje teksta na način da se dohvaća tekst komentara iz programskog koda na osnovu kojeg je izrađena web stranica odabrane društvene mreže [7].

1.1. PROBLEM, PREDMET I OBJEKTI ISTRAŽIVANJA

Problem istraživanja ovog diplomskog rada je binarna klasifikacija koja predstavlja prognozu smjera kretanja cijene *Bitcoin-a* na dnevnoj bazi. Pomoću odabranih algoritama binarne klasifikacije potrebno je što uspješnije klasificirati hoće li sutrašnja cijena zatvaranja (engl. *closing price*) *Bitcoin* kriptovalute rasti ili padati. Kao potpora u klasifikaciji smjera kretanja cijene uz informacije o cijeni *Bitcoin-a*, ulaznom setu podataka dodaju se podaci o analizi sentimenata teksta s *Bitcointalk* foruma.

Predmet istraživanja je programska izvedba web struganja, analiza sentimenata teksta iz *Bitcointalk* foruma, izrada klasifikacijskih algoritama, komparativna analiza modela sa i bez podataka analize sentimenta teksta.

Objekti istraživanja koji se proučavaju u okviru diplomskog rada jesu: analiza sentimenata teksta, primijenjeni algoritmi binarne klasifikacije u problemu predviđanja smjera kretanja cijene *Bitcoin-a*.

1.2. RADNA HIPOTEZA

Na temelju iznesenog problema, predmeta i objekata istraživanja definira se radna hipoteza. Primjena analize sentimenta temeljene na web struganju može unaprijediti klasifikaciju modela za predviđanje smjera kretanja cijene kriptovaluta.

1.3. SVRHA I CILJEVI ISTRAŽIVANJA

Svrha i cilj istraživanja očituju se kroz sljedeće: ispitati može li analiza sentimenta biti korisna pri poboljšanju uspješnosti predviđanja usmjerenja cijene *Bitcoin* kriptovalute, usporediti rezultate uspješnosti kroz tri različita klasifikacijska modela upotrebom ROC krivulja, matrica konfuzije te rezultata mjera točnosti, preciznosti, odziva i AUC parametra modela.

1.4. ZNANSTVENE METODE

Znanstvene metode upotrebljene u ovom diplomskom radu jesu: povijesna metoda, metoda analize i sinteze, metoda deskripcije, metoda komparacije, metoda kompilacije, matematička metoda te metoda modeliranja.

1.5. STRUKTURA RADA

U „Uvodu“, prvom dijelu rada, sažeto je opisan povijesni razvoj valuta kao uvod u temu kriptovaluta, potom su navedeni problem, predmet i objekt istraživanja, radna hipoteza, svrha i ciljevi istraživanja, znanstvene metode te je obrazložena struktura rada.

Naslov drugog dijela rada je „Kriptovalute“. U tome dijelu rada opisana je povijest kriptovaluta te opis tehnologija koji su baza *Bitcoin* kriptovalute.

“Analiza sentimenta teksta” naslov je trećeg dijela rada. U tom dijelu ukratko je opisana analiza sentimenta teksta, opisana je biblioteka *Transformers* programskog jezika *Python* te algoritam koji je primijenjen u rješavanju zadanog problema.

U četvrtom dijelu rada s naslovom „Algoritmi binarne klasifikacije“, opisana je binarna klasifikacija, algoritmi koji su se upotrijebili u problemu predviđanja kretanja cijene *Bitcoin-a* te su opisane upotrebljene mjere uspješnosti klasifikacije uz matrice konfuzije i ROC krivulje.

U petom dijelu naslova „Predviđanje smjera kretanja cijene *Bitcoin-a*“. Opisan je programski kod vezan za struganje i analizu teksta iz *Bitcointalk* online foruma, kao i

programski kod vezan za predviđanje smjera kretanja cijene *Bitcoin*-a, potom su prikazani rezultati kroz ROC krivulje, matrice konfuzije te rezultate uspješnosti točnosti, preciznosti, odziva i AUC parametra modela.

U posljednjem dijelu, „Zaključku“, dana je sinteza rezultata istraživanja kojima se dokazuje zadana radna hipoteza.

2. KRIPTOVALUTE

2.1. POVIJEST KRIPTOVALUTA

Prva kriptovaluta *eCash* razvijena je 1990. godine od strane tvrtke *DigiCash*. Američki kriptograf David Chaum je 1983. godine predložio oblik elektroničke gotovine. Razvio je takozvanu zasljepljujuću formulu korištenu za enkripciju informacija koje se šalju između pojedinaca. Takav oblik novca bi se tako mogao sigurno prenositi između pojedinaca, noseći potpis autentičnosti i mogućnost izmjene bez mogućnosti praćenja. Kako bi svoj patent proveo u praksi, Chaum je odlučio osnovati tvrtku naziva *DigiCash*, što mu je omogućilo stvaranje prvog elektroničkog kriptografskog novca nazvanog *eCash*. *Digicash* je bankrotirao 1998. godine, ali su ideje koje je tvrtka iznijela i neke od njenih formula te alata za kriptografiju odigrali važnu ulogu u razvoju kasnijih digitalnih valuta [8].

Douglas Jackson i Barry Downey stvorili su elektroničku valutu naziva *E-Gold* koja je bila vezana za posjedovanje zlata. Ova digitalna valuta omogućila je korisnicima prijenos vlasništva nad zlatom između korisnika web stranice, koja je nenamjerno ali brzo postala alat za „perache novca“ te druge korisnike koji traže anonimnost u svojim nezakonitim aktivnostima [9].

Koncept nazvan *Bit Gold* je koristio i mnoge slične tehnike koje su implementirane u sustav *Bitcoin-a*. Jedna od najznačajnijih tehnika bila je tehnika ulančanih blokova (engl. *Blockchain*) koja je ukratko baza podataka koja u svom registru sadrži zapise svih transakcija koje su verificirane, te će kasnije biti detaljnije opisana. Najznačajniji aspekt koncepta *Bit Gold* bio njegov odmak od centraliziranog statusa. *Bit Gold* je imao za cilj izbjeći oslanjanje na centralizirane distributere valuta i vlasti. Cilj njegovog tvorca, bio je da *Bit Gold* odražava svojstva pravog zlata, čime korisnicima omogućuje eliminaciju posrednika. *Bit Gold* je, kao i drugi pokušaji, na kraju bio neuspješan. Međutim, on je predstavljao važan temelj za razvoj nadolazećih kriptovaluta [10].

B-money je predložen od strane programera Wei Dai-a. Predstavljao je koncept anonimnog distribuiranog sustava elektroničke gotovine. U sustavu te kriptovalute, digitalni pseudonimi koristili bi se za prijenos valute kroz decentraliziranu mrežu. Sustav je čak uključivao sredstva za provedbu ugovora unutar mreže bez korištenja treće strane. U konačnici, *B-money* kriptovaluta nije nikada uspjela privući dovoljno pažnje za uspješno lansiranje [11].

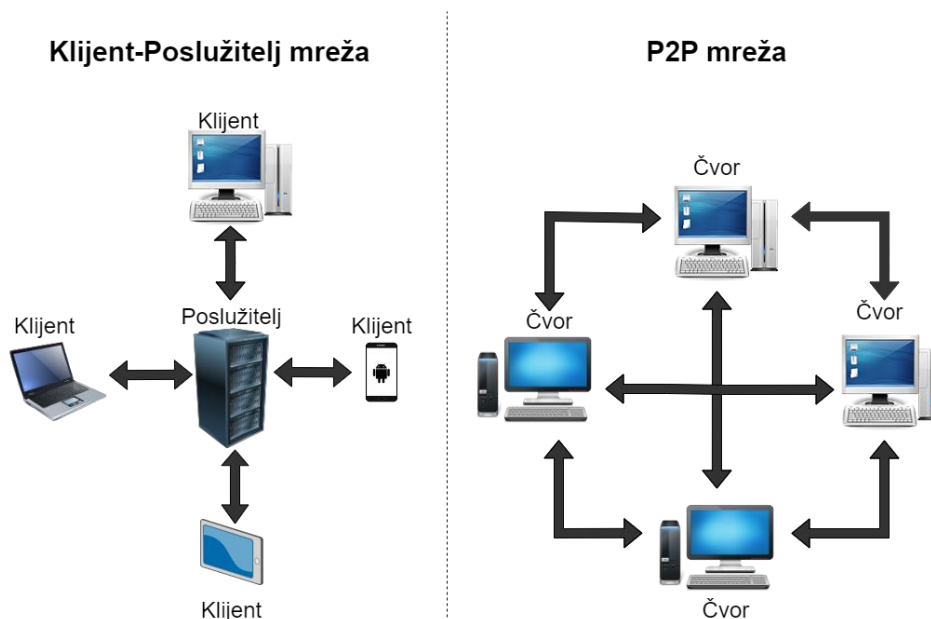
Razvijen sredinom 1990-ih, *Hashcash* je bio jedna od najuspješnijih digitalnih valuta prije *Bitcoin-a*. *Hashcash* je dizajniran za razne svrhe, uključujući smanjenje neželjene elektroničke pošte i sprječavanje DDoS (engl. *Distributed Denial of Service*) napada, koji je podklasa napada uskraćivanjem usluge (dovodi do neraspoloživosti određene usluge). DDoS napad uključuje više povezanih mrežnih uređaja koji se koriste za preplavlivanje ciljne web stranice lažnim prometom. *Hashcash* je otvorio široku ponudu mogućnosti koje su se realizirale tek gotovo nakon dva desetljeća kasnije. Koristio je *Proof-of-Work* (PoW) algoritam kako bi pomogao u stvaranju i distribuciji novih digitalnih kovanica digitalne valute, slično kao i kod suvremenih kriptovaluta. Godine 1997., suočavajući se s povećanom potrebom za procesorskom snagom, *Hashcash* je s vremenom postao sve manje učinkovit. Zbog takvih problema je u konačnici došlo do neuspjeha, ali je kao takav zabilježio značajan stupanj interesa na svom vrhuncu. Kao rezultat toga, mnogi elementi *Hashcash* sustava ušli su i u razvoj *Bitcoin-a* [12].

Bitcoin je prva opće poznata kriptovaluta koja služi kao sredstvo razmjene i skladištenja vrijednosti. Za razliku od Fiat valuta, *Bitcoin* ne kontrolira nijedna država ili središnja banka, niti je podržan bilo kakvom temeljnom imovinom, poput zlata. Umjesto toga, *Bitcoin* je digitalna valuta kojom upravlja P2P (engl. *peer-to-peer*) sustav koji se sastoji od mnogih čvorova. Čvorove čine računala čiji je zadatak provjeravati i potvrđivati transakcije prije nego što mogu postati dio transakcijskog zapisa [4]. P2P je tip računalne mreže gdje svako računalo istovremeno djeluje kao poslužitelj i klijent, što znači da dobavlja i prima datoteke sa određenom mrežom propusnošću (engl. *bandwidth*) i obradom (engl. *processing*) koje su raspoređene između svih članova mreže. Takva decentralizirana mreža koristi resurse učinkovitije od tradicionalne mreže i manje je osjetljiva na sustavne kvarove [13].

Prvi opis funkcionalnosti *Bitcoin-a* objavio je u znanstvenom radu 2008. godine [14] autor pod pseudonimom Satoshi Nakamoto, dok je *Bitcoin* mreža počela s radom godinu dana kasnije. U znanstvenom radu poziva se na kriptovalute *Hashcash* i *B-money* kojima je *Bitcoin* inspiriran.

2.2. ARHITEKTURA BITCOIN MREŽE

Struktura Bitcoin mreže bazira se na *peer-to-peer* (P2P) mrežnoj arhitekturi. To znači da su računala u mreži međusobno ravnopravna i da nema privilegiranih čvorova te da svi čvorovi (eng. *nodes*) dijele teret pružanja mrežnih usluga. Takva arhitektura nema poslužitelja, niti centralne usluge, kao ni hijerarhije unutar mreže. Čvorovi mreže istovremeno pružaju i koriste mrežne servise. Na slici 1 prikazana je usporedba P2P arhitekture sa standardnom klijent-poslužitelj arhitekturom.



Slika 1: Usporedba klijent-poslužitelj i P2P mrežnih arhitektura

Izvor: Adaptirao student prema [15]

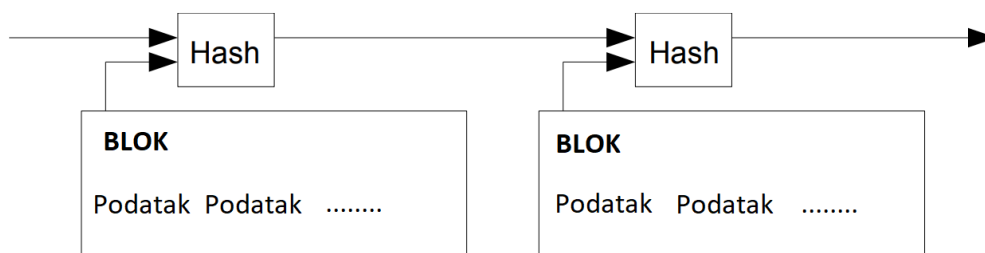
Iako svi čvorovi P2P mreže imaju podjednaku važnost, isti mogu imati drukčije uloge ovisno o funkcionalnosti koju pružaju. Čvor *Bitcoin* mreže je kolekcija funkcija poput: usmjeravanja (engl. *routing*), baze podataka ulančanih blokova, rudarenja te usluga *Bitcoin* novčanika. Potpuni čvor ili jezgra *Bitcoin-a* je kolekcija svih tih usluga, te on sadrži najnoviju kopiju ulančanih blokova. Postoje još čvorovi P2P strukture koji imaju kombinacije usluga poput:

- potpunog čvora ulančanih blokova (ne sadrži usluge rudarenja niti usluge novčanika)
- samostalnog rudarskog čvora (ne sadrži usluge novčanika)

- lagani čvor novčanika (sadrži samo podskup strukture ulančanih blokova, nema usluge rudarenja, odnosno služi samo kao korisnički novčanik).

2.3. TEHNOLOGIJA ULANČANIH BLOKOVA

Ulančani blokovi predstavljaju uređenu strukturu podataka koja predstavlja povratno povezanu listu transakcijskih blokova. Povratno povezana lista označuje da su blokovi povezani unatraske, gdje je svaki blok povezan s prethodnim tvoreći na taj način lanac blokova. Ulančani blokovi se mogu vizualizirati kao vertikalni stog, s blokovima naslaganim na način da je jedan na vrhu drugog, gdje prvi blok služi kao temelj stoga. Pojednostavljeni prikaz strukture ulančanih blokova prikazan je na slici 2.



Slika 2: Simboličan prikaz strukture ulančanih blokova

Izvor: Adaptirao student prema [14]

Svaki blok unutar ulančanih blokova identificira se pomoću kriptirane identifikacijske oznake (engl. *hash*) koja je generirana primjenom *SHA-256* kriptografskog algoritma na zaglavlju bloka. Hash funkcija je matematička funkcija koja pomoću kriptografskog algoritma pretvara ulazni podatak proizvoljne veličine u šifrirani izlaz fiksne veličine. Neovisno o veličini ulaznog podatka, njegova kriptirana identifikacijska oznaka će uvijek biti iste veličine. Kriptirana identifikacijska oznaka odnosno izlaz funkcije se značajno mijenja pri minimalnoj promjeni ulaznog podatka. Prema navedenom, postoji vrlo mala vjerojatnost za otkrivanje ulazne informacije iz kriptiranog izlaza uporabom metode reverznog inženjeringa. Kriptografski algoritam je determinističan, što znači da će za isti ulaz uvijek dobiti identični šifrirani izlaz [4].

Svaki blok se poziva na prethodni blok putem kriptirane identifikacijske oznake, što označuje polje zaglavlja bloka. Prema tome, svaki blok u svom zaglavlju sadrži kriptiranu identifikacijsku oznaku prethodnog bloka. Redoslijed kriptiranih identifikacijskih oznaka kojim se povezuje blok s prethodnim blokom stvara lanac koji se prostire sve do prvog ikad kreiranog bloka (najstarijeg bloka), poznatog kao temeljni (engl. *genesis*) blok.

2.3.1. Rudarski čvorovi mreže

Rudari su računala (čvorovi) u mreži kriptovalute koja otkrivaju valjani dokaz o obavljenom radu (engl. *Proof-of-Work*) ili PoW novonastalih blokova transakcija te ih verificiraju. PoW je podatak koji za pronalazak zahtijeva izvršavanje određenog računalnog rada, pri čemu zahtijeva značajnu računalnu snagu. Kako bi bili nagrađeni, rudari moraju pronaći numeričko rješenje algoritma *SHA-256*, nakon pronalaska kao nagradu dobivaju određenu vrijednost *Bitcoin-a*.

Iako se na rudarenje *Bitcoin-a* gleda samo kao na nagradu za uspješno izvršeni rad, primarni cilj rudarenja nije nagrada ili generacija novih digitalnih kovanica *Bitcoin* kriptovalute. Rudarenje je proces koji podupire pouzdan i siguran rad decentraliziranog sustava, kojim se potvrđuju transakcije. Rudari potvrđuju nove transakcije i spremaju ih u globalni zapisnik (struktura ulančanih blokova). Novi blok, koji sadrži transakcije koje su se dogodile od prijašnjeg bloka nadalje, se rudari u prosjeku svakih 10 minuta, čime se transakcije dodaju u strukturu ulančanih blokova. Transakcije koje su dio bloka i dodane su u ulančane blokove smatraju se potvrđenim, što omogućuje korisnicima trošenje novih vrijednosti *Bitcoin-a* koji su primili tim transakcijama. Rudari za uzvrat pružanja sigurnosti sustavu dobivaju dvije vrste nagrada: nove digitalne kovanice generirane s izradom novog bloka, te provizije svih transakcija uključenih u tom bloku. Kako bi osvojili ove nagrade, rudari se natječu za rješavanje kompleksnog matematičkog problema baziranog na kriptografskom *hash* algoritmu. Dokaz o radu ili *Proof-of-Work* (PoW), dodano je u novi blok i služi kao dokaz da je rudar izvršio značajan računalni rad. Natjecanje za rješenje PoW algoritma u svrhu dobivanja nagrade i pravo spremanja transakcija u ulančane blokove je baza sigurnosti *Bitcoin-ovog* modela.

Samostalni rudarski čvorovi ili potpuni čvorovi mreže koji služe za rudarenje, opisuju se kao rudarski čvorovi, koji kao i ostali čvorovi primaju i propagiraju nepotvrđene transakcije

kroz *Bitcoin* mrežu. Međutim, rudarski čvor uz prikupljanje, prijenos i validaciju transakcija, što je karakteristično i za ostale čvorove, također uključuje te transakcije u blok. Rudar ili rudarski čvor najprije prikuplja transakcije, nakon toga ih validira pa ih potom dodaje u privremeno spremište memorije ili spremište transakcija, gdje su transakcije privremeno pohranjene prije svog uključivanja u blok. Kao i svi ostali čvorovi taj čvor detektira nove blokove koji se propagiraju na *Bitcoin* mreži. Za rudarski čvor, dolazak novog bloka ima poseban značaj. Jedan ciklus natjecanja između rudara efektivno završava propagacijom novog bloka što je najava da je određeni rudar izvršio zadani rad i time pridobio valjani PoW. Rudarima, primanje novog validiranog bloka znači da je neki čvor uspio prikupiti, validirati transakcije i dodati ih u novi blok prije ostalih rudara. Završetkom jednog ciklusa natjecanja, automatski počinje drugi ciklus. Transakcija koja je postavljena na mjesto prve transakcije u bloku naziva se *Coinbase* transakcija, a tamo ju postavlja trenutni rudar koji je pridobio valjani PoW. *Coinbase* transakcija generira nove digitalne kovanice *Bitcoin-a* koje se isplaćuju rudaru kao nagrada za uspješno izvršeno rudarenje [4].

2.3.2. Decentralizirani Konsenzus

Glavni izum stvoritelja *Bitcoin-a* je decentralizirani mehanizam za pojavni konsenzus ili konsenzus u nastajanju (engl. *emergent consensus*). Pošto se ne pojavljuje izričito, ne postoji izbor niti određeni trenutak kada se konsenzus događa. On je pojava struktura asinkronih međudjelovanja tisuća neovisnih čvorova, od kojih svi slijede jednostavna pravila. Sva svojstva *Bitcoin-a* uključujući: valutu, transakcije, plaćanja i model sigurnosti koja ne ovise o izvršnoj vlasti ili povjerenju, proizlaze iz ovog izuma.

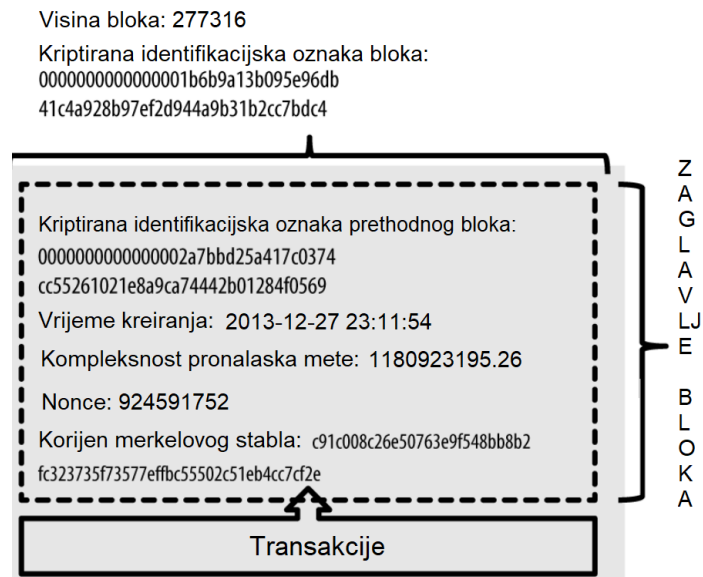
Bitcoin-ov decentralizirani konsenzus proizlazi iz međudjelovanja četiri procesa koji se neovisno odvijaju na čvorovima diljem mreže:

- Neovisna verifikacija svake transakcije, od strane svakog potpunog čvora, na temelju opsežnog popisa kriterija
- Nezavisno prikupljanje tih transakcija u nove blokove od strane rudarskih čvorova, zajedno s demonstriranim izračunom kroz algoritam PoW
- Neovisna provjera novih blokova od strane svakog čvora i sklapanje u lanac blokova

- Neovisni odabir, od strane svakog čvora, lanca s najviše kumulativnih proračuna demonstriranih kroz PoW [4]

2.3.3. Identifikatori bloka

Primarni identifikator bloka je kriptirana identifikacijska oznaka izrađena dvostrukim kriptiranjem kroz *SHA-256* kriptografski algoritam. Kriptirana identifikacijska oznaka veličine 32 bajta identificira blok jedinstveno i nedvosmisleno, a generirana je jednostavnim kriptiranjem zaglavlja bloka. Čvorovi mreže zaduženi su za izračun identifikacijskih oznaka bloka primljenih iz *Bitcoin* mreže [4]. Drugi način identifikacije bloka je koristeći njegovu poziciju unutar ulančanih blokova, koja se naziva visina bloka (engl. *block height*). Temeljni blok odnosno prvi blok ikad kreiran ima visinu bloka jednaku 0. Svaki nadolazeći blok ima visinu bloka za jednu poziciju veću od prethodnog. Primjerice, visina bloka za datum 1.1.2017. iznosi približno 446 000, što znači da je 446 tisuća blokova naslagano na vrh temeljnog bloka kreiranog u siječnju 2009. godine [4]. Za razliku od kriptirane identifikacijske oznake bloka, visina bloka nije jedinstveni identifikator. Iako će zasebni blok uvijek imati nepromijenjenu visinu bloka, ista ne može identificirati jedan određeni blok. Razlog je u tome što dva bloka mogu imati jednaku visinu bloka, dijeleći istu poziciju u strukturi ulančanih blokova, čemu je razlog pojava grananja (engl. *blockchain fork*) [4]. Kriptirana identifikacijska oznaka bloka kao ni visina bloka zapravo nisu uključeni u podatkovnu strukturu bloka, ni pri prijenosu bloka mrežom niti kada je blok pohranjen u trajnoj pohrani čvora kao dio strukture ulančanih blokova. Umjesto toga, kriptiranu identifikacijsku oznaku bloka izračunava svaki čvor, dohvaćajući blok s mreže. Kriptirana identifikacijska oznaka bloka i visina bloka mogu biti pohranjeni u zasebnoj tablici baze podataka kao dio informacija o podacima bloka, kako bi se olakšalo indeksiranje i ubrzao pristup blokovima iz memorije u koju su pohranjeni [4]. Primjeri identifikatora bloka prikazani su na *slici 3*, uz primjer zasebnog bloka strukture ulančanih blokova.



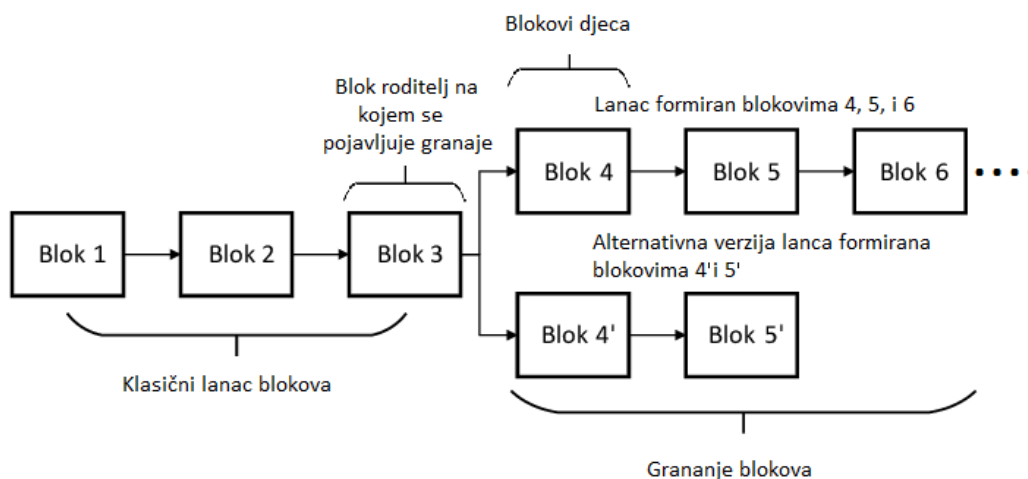
Slika 3: Primjer bloka uz identifikatore bloka

Izvor: Adaptirao student prema [4]

2.3.4. Grananje blokova

Iako se blok povezuje na samo jedan prethodni blok, trenutno može imati više nadolazećih blokova koji se granaju na zajedničkom prethodnom bloku. Više nadolazećih blokova koji se spajaju na jednom bloku stvara se tijekom procesa grananja ulančanih blokova. Grananje je privremena situacija koja se tipično javlja kada su različiti blokovi otkriveni gotovo istovremeno od strane različitih rudara [4], a do istog može doći kao pojava pri sigurnosnim napadima na strukturu ulančanih blokova [16]. Kako su oba rudara gotovo istovremeno otkrila PoW, oba istog časa dijele svoj novoizrađeni blok svojim neposrednim susjednim blokovima (ostalim čvorovima mreže) koji počinju propagirati blok širom mreže. Kandidatski blok označuje blok koji je inicijaliziran od strane određenog rudara te je takav spreman za ispunjavanje listom transakcijama. Ako taj čvor kasnije detektira drugi kandidatski blok koji proširuje isti prethodni blok, povezuje drugog kandidata na sekundarnom lancu blokova. Kao rezultat, neki će rudari detektirati blok prvog kandidata, dok će drugi otkriti blok drugog kandidata, pa će tako postojati dvije verzije ulančanih blokova. Nijedna strana se ne može klasificirati ni kao ispravna ni kao neispravna, budući da su obje verzije valjane perspektive ulančanih blokova. U konačnici će samo jedna verzija ulančanih blokova prevladati, bazirano na tome kako će se pojedina verzija nastaviti širiti dodatnim radom [4]. Problem grananja je gotovo uvijek riješen unutar jednog bloka. Dok je dio mrežnih resursa ili čvorova zadužen za gradnju jedne verzije ulančanih blokova, drugi je dio zadužen za gradnju druge verzije. U

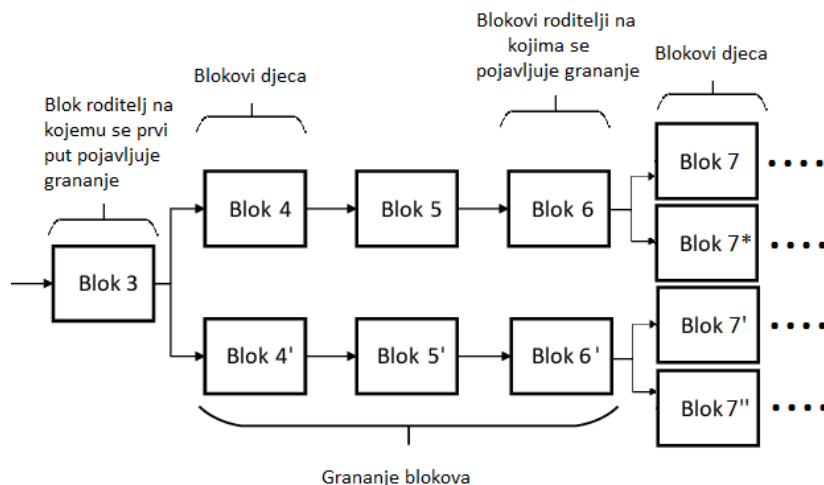
slučaju da su resursi koji se fokusiraju na rješavanje pojedine verzije podjednako podijeljeni, vrlo je vjerojatno da će jedan skup rudara pronaći rješenje i propagirati, prije nego što će drugi skup pronaći bilo kakvo rješenje. Slučaj grananja unutar jednog bloka prikazan je na slici 4.



Slika 4: Grananje unutar jednog bloka

Izvor: Adaptirao student prema [16]

Na slici 4 može se vidjeti da se na bloku 3 stvaraju dvije alternativne verzije ulančanih blokova, ali na kraju pobjeđuje lanac s najviše blokova tako da će se blokovi nastaviti nadograđivati na gornji lanac nakon bloka 6, pa se lanac koji se sastoji od blokova 4' i 5' neće više nadograđivati novim blokovima [4]. Teoretski postoji mogućnost da se grananje nastavi na dva bloka kako je prikazano na slici 5, ako kojim slučajem skupine rudara budu podjednako uspješne pronalasku rješenja, ali su vjerojatnosti za pojavu istog vrlo male [4].



Slika 5: Grananje unutar dva bloka

Izvor: Adaptirao student prema [16]

2.3.5. Struktura bloka

Blok je spremnik koji prikuplja transakcije kako bi iste bile uključene u strukturu ulančanih blokova. Sastoji se od zaglavlja koji sadrži informacije o podacima, nakon čega slijedi dugačak popis transakcija koje čine najveći dio njegove veličine. Veličina zaglavlja bloka je 80 bajta gdje je prosječna transakcija minimalne veličine 250 bajta, a prosječan blok sadrži više od 500 transakcija. Kompletan blok sa svim transakcijama, iz navedenog razloga, je više od 1000 puta veći od zaglavlja bloka [4] .

Zaglavlje bloka se sastoji od tri skupa informacija o podacima bloka. Prvi skup informacija je poveznica na kriptiranu identifikacijsku oznaku prethodnog bloka, koja spaja trenutni blok s prethodnim blokom. Drugi skup informacija predstavljaju: razina složenosti pronalaska mete (engl. *Difficulty Target*), vrijeme kreiranja (engl. *timestamp*) i jednokratna vrijednost (engl. *nonce*). Jednokratna vrijednost je umjetno generirana vrijednost ili broj koji se može koristiti samo jednom i služi kao brojač tijekom procesa rudarenja. Kriptografski *hash* algoritmi i protokoli provjere autentičnosti često koriste jednokratnu vrijednost u kontekstu tehnologije ulančanih blokova. Razina složenosti pronalaska mete predstavlja intenzitet računalnog rada koji je potreban za pronalazak mete PoW algoritma za blok [4], vrijeme

kreiranja je približno vrijeme kreiranja bloka u sekundama iz *Unix Epoch-a*, koji mjeri vrijeme u sekundama počevši od 1.1.1970. godine [17].

Treći skup informacija je struktura podataka pod nazivom korijen merkleovog stabla (engl. *merkle tree root*), koja se koristi za učinkovito sažimanje svih transakcija u bloku. Proces sažimanja se izvodi s namjerom dobivanja jedne sveukupne kriptirane identifikacijske oznake transakcija. Izrada merkelovog stabla je proces rekurzivnog kriptiranja parova pojedinih čvorova, sve dok ne postoji samo jedna kriptirana identifikacijska oznaka koji je sažeta suma svih identifikacijskih oznaka transakcija unutar bloka.

To znači, ako primjerice postoje četiri transakcije naziva A, B, C, D unutar bloka, njihovi identifikacijske oznake (H_i) su dobivene postupkom prikazan formulom [4]:

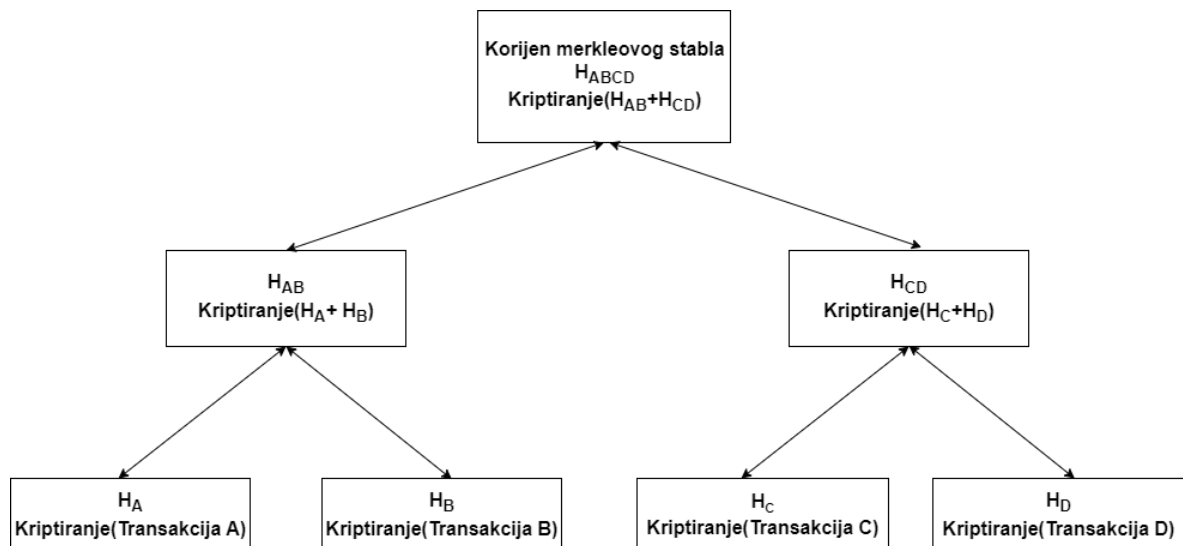
$$H_i = SHA256 ((Transakcija\ i)) \quad (1)$$

Gdje je $i = A, B, C, D$, a oznake $ABCD$ predstavljaju transakcije unutar bloka.

Formula prikazuje da se već kriptirana transakcija ponovo ubacuje unutar *SHA-256* algoritma ili ponovo kriptira. Ovaj proces je poznat kao dvostruko kriptiranje (engl. *double-SHA256*). Kada se dobiju dvostruko kriptirane identifikacijske oznake H_A, H_B, H_C, H_D , spajaju se u parove ($H_A + H_B$ i $H_C + H_D$). H_{AB} se dobiva formulom [4]:

$$H_{AB} = SHA256 (SHA256 (H_A + H_B)) \quad (2)$$

Prema navedenom, H_{CD} dobiva po istom principu. Postupak se nastavlja sve dok se ne dobije jedan korijen na vrhu koji je poznat kao korijen merkleovog stabla (engl. *merkle tree root*) [4]. Rekurzivni proces dobivanja korijena merkleovog stabla prikazan je na slici 6.



Slika 6: Postupak dobivanja korijena merkleovog stabla

Izvor: Adaptirao student prema [4]

2.3.6. Izrada zaglavlja bloka

Za uspješnu izradu zaglavlja bloka rudarski čvor je dužan popuniti 6 polja prikazanih u tablici 1.

Tablica 1: Polja zaglavlja bloka i njihove veličine

Veličina (byte)	Polje
4	inačica
32	hash oznaka prethodnog bloka
32	korijen merkleovog stabla
4	vrijeme kreiranja
4	meta
4	jednokratna vrijednost

Izvor: Izradio student prema [4]

U vremenu kada je bio rudaren određeni blok, broj inačice koji opisuje strukturu bloka bio je 2 te je to prikazano u sljedećem obliku: *0x02000000*. Nadalje, rudarskom čvoru potrebna je kriptirana identifikacijska oznaka prethodnog bloka koja je prikazana u sljedećem obliku: *b6ff0b1b1680a2862a30ca44d346d9e8910d334beb48ca0c0000000000000000*. Sljedeći korak je sažimanje svih transakcija pomoću metode merkleovog stabla. Važno je napomenuti da mora postojati paran broj čvorova koji se granaju u stablu. Da bi se zadovoljio taj uvjet, u slučaju da postoji nepran broj transakcija (uključujući i *coinbase* transakciju) potrebno je kreirati još jedan čvor transakcija koji je duplikat posljednje transakcije [4]. Postupak sažimanja se izvodi dok se ne dobije jedna 32 bajtna vrijednost koja je u sljedećem obliku: *9d10aa52ee949386ca9385695f04ede270dda20810decd12bc9b048aaab31471*. Rudarski čvor će nakon toga dodati 4 bajtni vremenski zapis, baziran na *Unix Epoch* vremenskom zapisu, prikazan u sekundama u slijedećem obliku: *1415239972*. Sljedeći zadatak je popunjavanje polja mete koja definira potrebni PoW kako bi taj blok postao važeći. Poveznica između mete i razine složenosti pronalaska mete je u tome što je vrijednost mete u bitovima niža to je razina složenosti pronalaska iste veća [18]. Razlog navedenom je to što je postavljena brojeva vrijednost niža to je veći izazov pri otkrivanju vrijednosti koja je nižeg iznosa od postavljene mete. Složenost pronalaska mete i meta odnosno trenutna meta povezuju se formulom [18]:

$$C_{mete} = \frac{maks_{meta}}{tren_{meta}} \quad (3)$$

Gdje su:

C_{mete} - složenost pronalaska mete

$maks_{meta}$ - inicijalna vrijednost složenosti pronalaska mete koja postavljena pri prvom pokretanju tehnologije ulančanih blokova

$tren_{meta}$ - trenutna meta (meta)

Ako je meta prikazana vrijednošću 388618029, onda njezina vrijednost pretvorena u heksadecimalni zapis iznosi 0x1729d72d. Meta se dijeli na indeks i koeficijente. Prvi bajt je indeks nakon kojeg slijede 3 bajta koji su koeficijenti. Prema navedenom, indeks mete iznosi 0x17, a koeficijent je 0x29d72d (0x označava heksadecimalni zapis). Posljednje polje je jednokratna vrijednost koje je inicijalizirana na 0. Kako su sva polja zaglavlja bloka ispunjena, isto je spremno za proces rudarenja. Krajnji cilj je pronaći vrijednost jednokratne vrijednosti koja rezultira kriptiranom identifikacijskom oznakom zaglavlja bloka koji je manji od mete.

Rudarski čvor morat će testirati milijarde ili trilijune jednokratnih vrijednosti prije nego što se pronađe jednokratna vrijednost koji će zadovoljiti uvjet [4].

2.3.7. Rudarenje bloka

Rudarenje je proces ponavljajućeg kriptiranja zaglavlja bloka, gdje se mijenja jedan parametar, sve dok se konačna kriptirana identifikacijska oznaka ne podudara sa specifičnom metom. Jedni način dobivanja rezultata vrijednosti kriptirana identifikacijska oznaka koja će se podudarati s specifičnom metom je ponavljanje sve dok se ne pronađe rješenje, nasumičnim modificiranjem ulaza dok se ne pojavi željeni rezultat kriptirane identifikacijske oznake.

Ključna karakteristika kriptografskog *hash* algoritma je to da je računalno neizvedivo pronaći dva različita ulaza koji će dati isti rezultat na izlazu iz kriptografskog algoritma što je poznato kao sudar (engl. *collision*) [4]. Također je nemoguće odabrati ulaz na takav način da proizvede željeni izlaz, osim ubacivanja nasumičnih ulaza kroz algoritam ili kroz postupak višestrukog kriptiranja [19].

Ako se unutar *Python* programskog jezika ubaci programski kod na slici 7, koji uvozom *hashlib* biblioteke omogućuje uporabu *SHA-256* algoritma. Ako se tekstu „*Satoshi*“ doda kao posljednji znak broj od 0 do 19 te se taj tekst ili niz znakova ubaci unutar *SHA-256* algoritma, kao što je već poznato svaki ulaz će dati potpuno drukčiji izlaz iz algoritma. Taj broj koji se svakim prolaskom kroz *for* petlju povećava za 1 predstavlja jednokratnu vrijednost. Prema navedenom, jednokratna vrijednost se koristi za izmjenu izlaza kriptografske funkcije [4].

```
import hashlib

tekst="Satoshi"

for nonce in range(20):
    ulaz=f"{tekst}{nonce}"
    hashirana_vr=hashlib.sha256(ulaz.encode())
    hash=hashirana_vr.hexdigest()
    print(f"{ulaz} => {hash}")
```

Slika 7: Programski kod za kriptiranje pojedinih ulaza

Izvor: Izradio student uporabom Jupyter Notebook IDE-a prema [3]

Izlaz programskog koda za dobivanje kriptirane oznake vrijednosti pojedinih fraza prikazan je na slici 8.

```
Satoshi0 => 5fd20f52062f7549fb2969725353fd8711d56b2b46d814393b12491bba92cf35
Satoshi1 => 498c753629cf8a831f2ba79224871c6bcfac09c12d327069311efdce59bef1ed
Satoshi2 => acc6210a60c9ff10c630222680bb829385bc33c93591b55c10dc026b92946882
Satoshi3 => 2a08ec245aaeadc28ec834f015ba8c044d8b5f35c8ac6beea58d68cd4307926e
Satoshi4 => deb8be0e7c9147f6a9491ea31c74e2c3251c3ca27d83be97c4ffb99e4049f462
Satoshi5 => c7aac11d264bbd757b5f3fc9af75f625bbc9e21f80ec071f86ac96fd380ea9c1
Satoshi6 => 2ce97e3460456621e1dc262646bd074ed1f230aa025d4bb2a31d52220d8fa835
Satoshi7 => ce9361ab655f2a73b05f9322ec7e02737ddb455fc6031c247214e2aaf411cd3b
Satoshi8 => 9de4537a23c78ef826f7a17e51153e688d3b990ace191758fc9d8f718f87101c
Satoshi9 => 3146e75b8aa9de87fac213a199ed49406917cc690bd9c5d827325404e80b3b1e
Satoshi10 => 9ce08fda8dca47a3a6e7318158b48de4cca33fcb2fa3d80a3af5918a3926ce0b
Satoshi11 => 185ab4cba0452d2fa6930a0567fd01a6235e70ef82ebdb5b4188a4e445e0d9e5
Satoshi12 => 8e2122d4056e7499a07cc3b4d39b3a337cf38312b5a27b0455c127fdaf150e6f
Satoshi13 => 45e9b6fdef46e2136d22e8c9573decaff1eb8dd9a5ef68e14fcbabece5a3eae5
Satoshi14 => 97c198fd529b5a5616c8aad7584a355b5b08123e3a6b10d12f8f57aa2cacea34
Satoshi15 => bb22cd0fb59ff833b4d6826b730c7b18379057031ad4cdf26403ba5e26efa23c
Satoshi16 => 03c5723e21ac357d0fa6bc366dffbc3917c5d805ddcc231f40a1f656f0d11096
Satoshi17 => 5ab17472b1f8b4a906594879334d8c93a3b24ec6e57173df187e49d102399117
Satoshi18 => af6d9123dd54b9cec8527903f9eb2579ce15fc7d77021c8689ec4bb44c423b58
Satoshi19 => f1f7d71ffff90684701893a2213dded7ce5d7aa49180d32b99655113eae4001
```

Slika 8: Izlaz programskog koda prikazanog na slici 7

Izvor: Izradio student uporabom Jupyter Notebook IDE-a prema [3]

Ako se primjerice postavi meta koja mora pronaći izraz koji na izlazu *SHA-256* algoritma daje kriptiranu oznaku koja počinje sa nulom. Tada će se relativno brzo otkriti iz *slike 9* da izraz „*Satoshi16*“ daje kriptiranu oznaku jednaku vrijednosti *03c5723e21ac357d0fa6bc366dffbc3917c5d805ddcc231f40a1f656f0d11096*, što se moglo dobiti dodavanjem *if* funkcije kodu na slici 7 kako je prikazano na slici 9 koji zadovoljava uvjet postavljene mete. Cilj mete je pronaći kriptiranu oznaku koja je numerički manja od nje. Ako se smanji vrijednost mete zadatak pronalaska kriptirane oznake koji je manji od mete postaje sve teži [4].

```
import hashlib

tekst="Satoshi"

for nonce in range(20):
    ulaz=f"{tekst}{nonce}"
    hashirana_vr=hashlib.sha256(ulaz.encode())
    hash=hashirana_vr.hexdigest()
    if str(hash)[0]=="0":
        print(f"{ulaz} => {hash}")
```

Satoshi16 => 03c5723e21ac357d0fa6bc366dffb3917c5d805ddcc231f40a1f656f0d11096

Slika 9: Programski kod za kriptiranje u svrhu pronalaska specifične mete uz njegov izlaz

Izvor: Izradio student uporabom Jupyter Notebook IDE-a prema [3]

Kada se algoritam temelji na determinističkoj funkciji kao što je *SHA-256*, sam unos predstavlja dokaz (engl. *proof*) da je obavljena određena količina rada (engl. *work*) kako bi se proizveo rezultat vrijednosti manje od postavljene mete, taj dokaz je PoW. Prethodno izvedeni primjer pogađanja ulaza koji će dati izlaznu kriptiranu oznaku koja počinje s 0, je također PoW, jer postoji dokaz da je uloženi određeni računalni rad za pronalazak jednokratne vrijednosti. *Bitcoin*-ov PoW algoritam je vrlo sličan prethodnom primjeru. Rudar izradi blok kandidata popunjen transakcijama. Zatim, izračunava kriptiranu identifikacijsku oznaku zaglavlja bloka i provjerava da li je manja od trenutne mete. Ako uvjet nije zadovoljen rudar izmjenjuje jednokratnu vrijednost obično povećavajući je za 1, te ponavlja taj postupak sve do uspješnog pronalaska odgovarajuće vrijednosti. Razina složenosti pronalaska mete u *Bitcoin* mreži predstavlja mjeru složenosti pronalaska jednokratne vrijednosti. Pri definiranoj razini složenosti pronalaska mete rudari moraju pokušavati kvadrilijune puta prije uspješnog pronalaska tražene jednokratne vrijednosti [4].

3. ANALIZA SENTIMENATA TEKSTA

3.1. OPIS ANALIZE SENTIMENATA TEKSTA

Analiza sentimenta, poznata i kao rudarenje mišljenja je pristup obradi prirodnog jezika ili NLP (engl. *Natural Language Processing*) koji detektira emocionalni ton u tekstu koji se analizira. Ova metoda uključuje korištenje struganja podataka, strojnog učenja (engl. *Machine Learning*) i umjetne inteligencije za struganje teksta u potrazi za sentimentom i subjektivnim informacijama, poput izražavanja pozitivnih, negativnih ili neutralnih osjećaja. Sustavi analize sentimenta pomažu tvrtkama prikupljanje informacija o raspoloženju kupaca, korisničkom iskustvu te reputaciji robne marke u stvarnom vremenu. Neke od primjena navedenih alata jesu pri analizi elektroničke pošte, web članaka, online recenzija, odgovara na pitanja u anketama i slično. Alati namijenjeni za analizu sentimenta mogu izvući stav pojedinca kojim se dobiva povratna informacija o količini pozitivnosti ili negativnosti iz analiziranog teksta [20]. Kako bi se analizirao tekst potrebno je najprije prikupiti tekst u obliku pročišćenog teksta, odnosno zasebnog komentara odabrane web stranice. Pojam prikupljanja pročišćenog teksta poznat je pod nazivom web struganje. Tu zadaću u pravilu izvršava programski kod čija je namjena automatizirano ponavljajuće struganje teksta iz specifične web stranice. Struganje iz web stranica ili web struganje pojednostavljeno podrazumijeva da se iz hrpe programskog koda, programskih jezika u kojim je izrađena web stranica (CSS, Html, JavaScript, itd.), izvlači tekst koji je komentar korisnika i kojeg je isti ostavio na toj web stranici (online forumu, društvenoj mreži, online trgovini i sl.) [21]. Jedna od metoda analize sentimenta je primjena ugrađenih algoritma biblioteke *Transformers* koja se bazira na neuronskim mrežama kao što je predloženo u radu [22] te je ista korištena u praktičnom dijelu rada kao alat za rješavanje problematike analize sentimenta ostruganog teksta.

3.2. TRANSFORMERS BIBLIOTEKA

Transformers je biblioteka koju podržava programski jezik *Python*. Izrađena je u sklopu *HuggingFace* web stranice i njezina je najpopularnija biblioteka. *HuggingFace* čini veliku zajednicu otvorenog koda (engl. *open source*) koja izrađuje alate koji korisnicima omogućuju

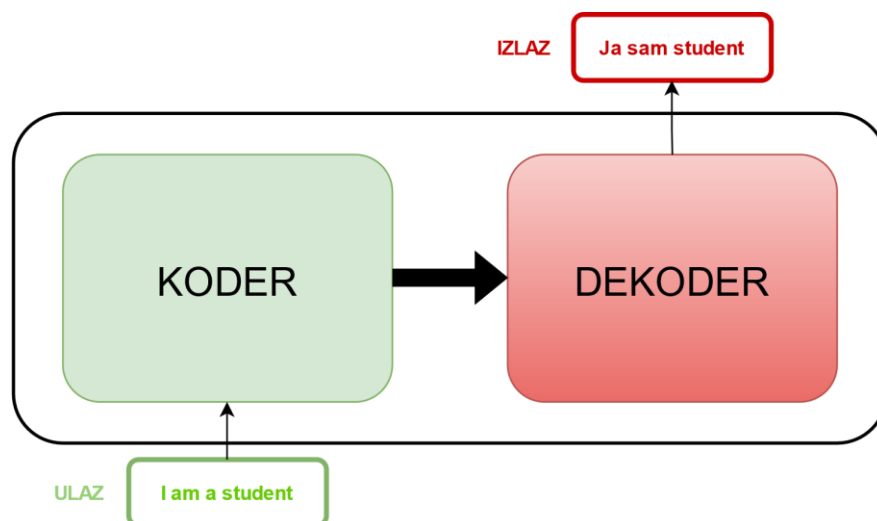
izgradnju, obuku i implementaciju modela strojnog učenja temeljenih na kodu i tehnologijama otvorenog koda [23]. Njezina najpopularnija biblioteka pruža odgovarajuća sučelja tzv. API-e (engl. *Application Programming Interface*), odnosno mehanizme koji omogućuju komunikaciju između dvije softverske komponente, koristeći skup definicija i protokola [24] i alate za jednostavno preuzimanje i treniranje najsuvremenijih pred treniranih (engl. *pre trained*) modela. *Transformers* sadrži mnoštvo modela za široku namjenu poput modela za: višestruku namjenu, računalni vid, NLP, manipulaciju zvučnim zapisom (klasifikaciju zvuka, prepoznavanje govora, pretvaranje teksta u govor i drugih), manipulaciju tablicama, podržano učenje (robotika) [25].

Transformers se bazira na NLP modelima, a to su jezični modeli koji su trenirani na velikoj količini teksta. Funkcionalnost tih modela proizlazi iz samonadziranog (engl. *Self-supervised*) učenja koje je tip treniranja (učenja) modela gdje isti uči u hodu, te ne treba strukturirane podatke. To znači da algoritam mora sam pronaći strukturu u podacima kako bi iz njih učio [23]. U [22] predložena je neuronska mreža *Transformers* s novom arhitekturom koja ima za cilj rješavanje zadataka od niza do niza, a njih čine složeni jezični problemi poput prevođenja, odgovaranja na pitanja i stvaranja *chatbot-a*, a sve to dok upravlja ovisnostima širokog raspona. Neuronska mreža je vrsta strojnog učenja, nazvanog duboko učenje (engl. *Deep learning*), koji koristi međusobno povezane čvorove ili neurone u slojevitoj strukturi koja nalikuje ljudskom mozgu. Osnovna struktura neuronske mreže ima međusobno povezane umjetne neurone u tri sloja i to:

1. ulazni sloj - prima podatke iz vanjskog svijeta, obrađuje ih, analizira i razvrstava.
2. skriveni sloj - prima informacije iz ulaznog sloja ili ostalih skrivenih slojeva. Neuronska mreža može imati veliki broj skrivenih slojeva, gdje će svaki sloj analizirati izlaz prethodnog skrivenog sloja, daljnje obraditi i proslijediti sljedećem sloju.
3. izlazni sloj - na temelju svih prethodnih obrada daje konačni rezultat, primjerice ako se radi o problemu binarne klasifikacije rezultat će biti 1 ili 0 (da ili ne). U slučaju da se radi o više-klasnom problemu klasifikacije, izlazni sloj može sadržavati više izlaznih čvorova [26].

Prijašnji modeli koji su se upotrebljavali kao podloga *Transformers-a* jesu RNN (engl. *Recurrent Neural Networks*) i LSTM (engl. *Long Short-Term Memory Networks*) neuronske

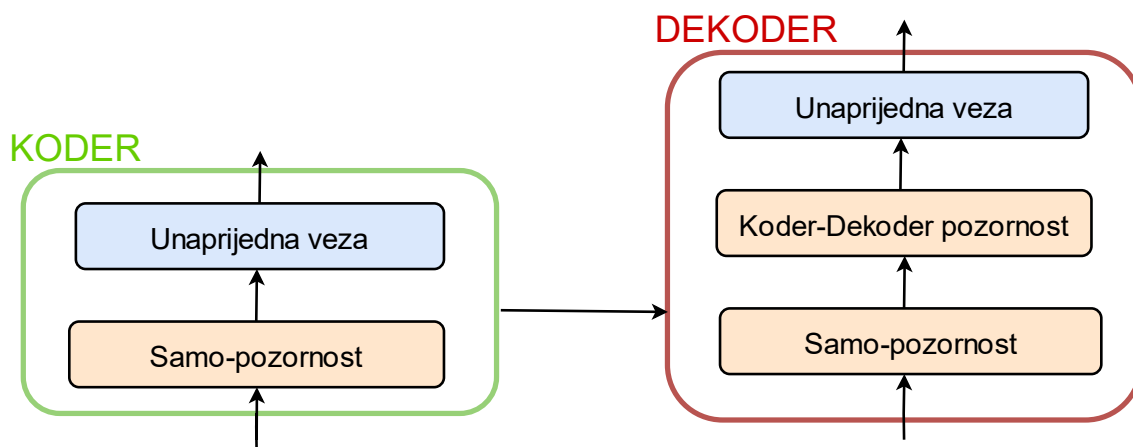
mreže. Umjesto navedenih, koristi se prošireni oblik koder-dekoder arhitektonske strukture za rješavanje navedenih zadataka. Primjer koder-dekoder (engl. *encoder-decoder*) arhitekture prikazan je na slici 10.



Slika 10: Koder-dekoder arhitektura

Izvor: Adaptirao student prema [23]

Prema slici 10 koder prima ulaze i iterativno ih obrađuje kako bi generirao informacije o tome koji su dijelovi ulaza značajni jedni za druge. Model mora biti optimiziran, kako bi što je moguće bolje razumio ulaznu informaciju. Dekoder generira ciljnu sekvencu koristeći reprezentaciju iz kodera i koristi kontekstualne informacije za generiranje izlaza. Ključ ove strukture je dodavanje mehanizama pozornosti (engl. *attention*) i samo-pozornosti (engl. *self-attention*) što je prikazano na slici 11.

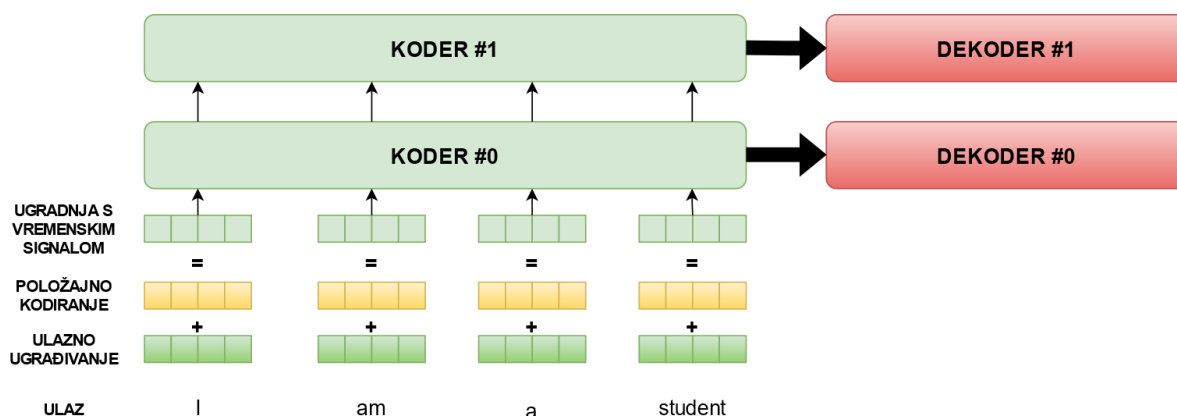


Slika 11: Mehanizmi unutar kodera i dekodera

Izvor: Adaptirao student prema [23]

Kako računala razumiju samo numeričke vrijednosti, u procesu transformacije tekstualnih podataka u numeričke vrijednosti, taj proces mora biti prvi primijenjen na početku većine *Transformers* modela. U svijetu dubokog učenja ili učenja po primjeru na čemu se bazira samonadzirano učenje, taj proces se naziva ugrađivanje riječi (engl. *word embedding*). Ugrađivanje riječi predstavljat će svaku riječ u obliku vektora stvarne vrijednosti koji kodira značenje riječi, tako da se očekuje da riječi bliže u vektorskom prostoru budu slične po značenju.

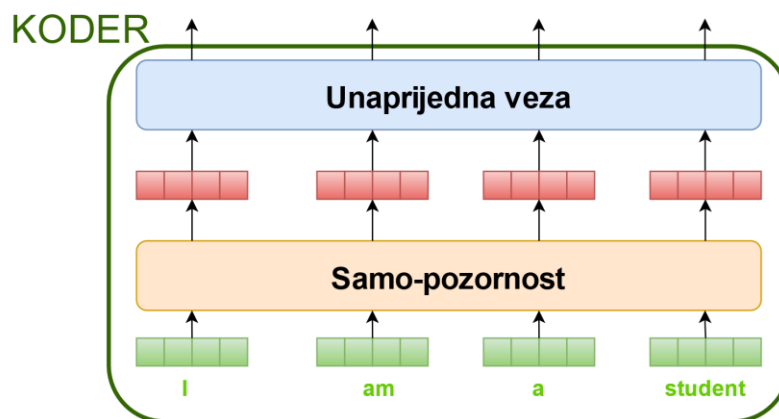
Jedna stvar koja nedostaje ulazu modela, kako je do sada opisan, je način uračunavanja redoslijeda riječi u ulaznom nizu. Kako bi se to riješilo, *Transformers* modelima se dodaje vektor u svako ulazno ugrađivanje. Ovi vektori slijede određeni obrazac koji model nauči, što mu pomaže odrediti položaj svake riječi ili udaljenost između različitih riječi u nizu. Opisan proces dodavanja vektora prikazan je na slici 12, gdje se svakom ulaznom ugrađivanju (engl. *input embeddings*) dodaje vektor položajnog kodiranja (engl. *positional encoding*) kako bi se dobio krajnji rezultat ugradnje s vremenskim signalom (engl. *embedding with time signal*).



Slika 12: Dodavanje vektora pozicijskog kodiranja ulaznom ugrađivanju

Izvor: Adaptirao student prema [23]

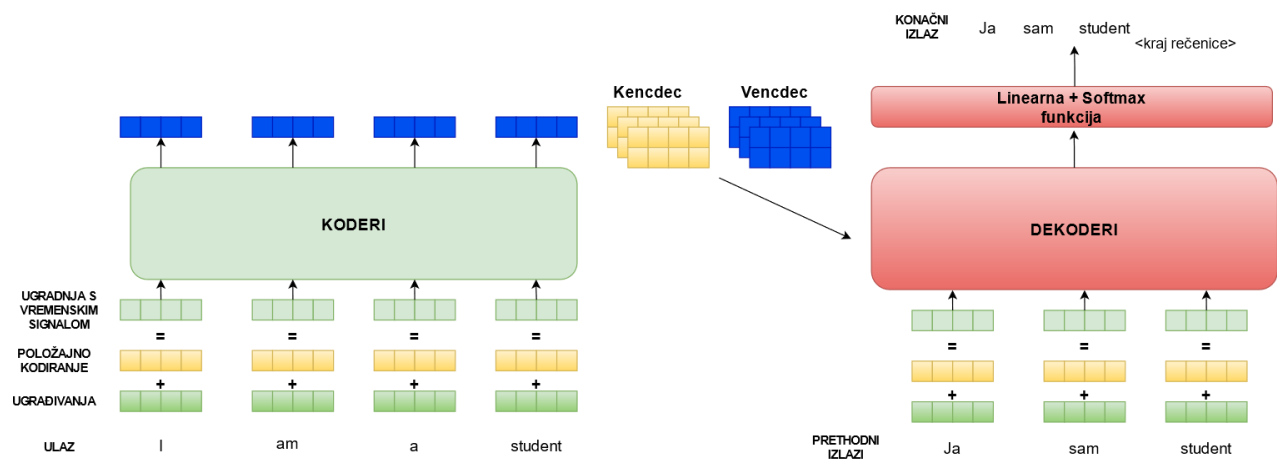
Nakon pretvorbe podataka u razumljiviji format, ugrađeni podaci su preneseni u sljedeći sloj, poznat kao sloj samo-pozornosti, što je prikazano na slici 13. Koristeći sloj samo-pozornosti, model *Transformers-a* je u stanju detektirati udaljene podatkovne relacije. To znači da će dani *Transformers* model proučavati odnose između dviju povezanih riječi, bez obzira da li su te riječi predaleko jedna od druge u danom kontekstu.



Slika 13: Prosljeđivanje podataka u sloj samo-pozornosti

Izvor: Adaptirao student prema [23]

Proces samo-pozornosti predstavlja značajnost specifične riječi u odnosu na susjedne riječi u danoj rečenici. Opisani odnosi predstavljaju takozvani vektor-pozornosti (engl. *attention-vector*). Postoje tri dodatna tipa vektora kreiranih u sloju samo-pozornosti to su: ključni (engl. *key*), upitni (engl. *query*) te vrijednosni (engl. *value*) vektori. Svaki je vektor onda pomnožen s ulaznim vektorom s ciljem dobivanja težinske vrijednosti. Taj proces zatim nastavlja unaprijedna neuronska mreža (engl. *feed-forward neural network*) koja preuzima svaki vektor-pozornosti i transformira ga u razumljivi oblik za sljedeći sloj. Unaprijedna neuronska mreža je tip neuronske mreže koje obrađuje podatke u jednom smjeru, od ulaznog čvora do izlaznog čvora. Svaki čvor u jednom sloju povezan je sa svim čvorovima u sljedećem sloju. Unaprijedna neuronska mreža koristi proces povratne informacije od unaprijedne veze za poboljšanje predviđanja tijekom vremena. Podaci se nakon toga prosljeđuju u sloj dekodera, koji predviđa izlaz danog modela. Princip rada koder-dekoder arhitekture prikazan je na slici 14. Gdje se može vidjeti proces prijevoda ulazne rečenice „I am student“ iz Engleskog na Hrvatski jezik. Nakon dobivanja ugradnje s vremenskim signalom podaci se ubacuju u koder. Izvršenjem obrade u koderu vektori-pozornosti prosljeđuju se u unaprijednu neuronsku mrežu kako bi se isti transformirali u oblik ključa i vrijednosti (*Kencdec* i *Vencdec*), pogodnog za ubacivanje u dekodek. Završetkom procesa obrade u dekoderu iz istog iterativno izlazi riječ po riječ, sve dok se ne dobije potpuni prijevod ulazne rečenice koji je „Ja sam student“.



Slika 14: Princip rada koder-dekoder arhitekture

Izvor: Adaptirao student prema [23]

4. ALGORITMI BINARNE KLASIFIKACIJE

U strojnom učenju klasifikacija je proces razvrstavanja danih informacija na temelju pred definiranih kategorija ili klasa [5]. Klase su ciljevi ili odrednice pojedinih kategorija. Klasifikator (algoritam ili klasifikacijski model) koristi podatke za treniranje kako bi ustanovio kako ulazne varijable ili prediktori pripadaju danoj klasi [27].

Binarna klasifikacija se koristi u slučajevima klasifikacije gdje je potrebno razvrstati dane podatke u dvije pred definirane klase [5]. Kod predviđanja hoće li sutrašnja cijena kriptovalute rasti ili padati za ulazne podatke mogu se uzeti informacije o cijeni kriptovalute kroz duži vremenski period kao podaci za treniranje ili strojno učenje. Klasifikacija spada u kategoriju nadziranog učenja, gdje su klase sadržane u ulaznim podacima.

Uz binarnu klasifikaciju na kojoj se bazira praktičan dio ovog diplomskog rada, postoji i višeklasna, odnosno više kategorijska klasifikacija (engl. *multiclass classification*). Sam naziv naznačuje da postoji više od dvije kategorije, ali u ovom slučaju se koristi više binarnih klasifikatora kako bi se efikasno pronašlo željeno rješenje [27]. Uz klasifikaciju postoje i drugi modeli za različite namjene poput regresije. Regresija kao model strojnog učenja spada također u kategoriju nadziranog učenja [28].

Temeljna razlika regresije u usporedbi s klasifikacijom je u tome što je kod regresije izlazna veličina kontinuirana, a kod klasifikacije diskretna. To znači da se kod regresije predviđa precizan broj, dok se problem klasifikacije rješava predviđanjem specifične klase [29].

4.1. PRISTRANOST I VARIJANCA

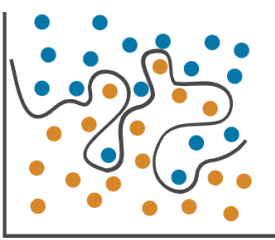
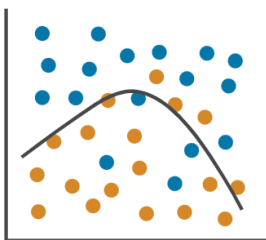
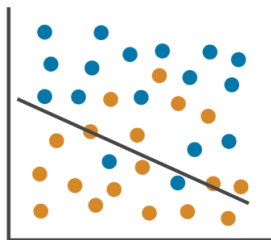
Svaki klasifikacijski model ima tendenciju stvaranja klasifikacijskih pogrešaka koje se mogu podijeliti u dvije glavne kategorije: pogreške uzrokovane pristranošću (engl. *bias*) i pogreške uzrokovane varijancom (engl. *variance*). Na osnovu tih pogrešaka postoje dvije neželjene krajnosti koje se mogu dogoditi pri treniranju određenog skupa podataka. To su pretreniranost (engl. *overfitting*) i podtreniranost (engl. *underfitting*) modela [30].

Pogreške uzrokovane pristranošću opisane su udaljenošću između klasifikacije modela i stvarnih vrijednosti. Kod ovog tipa pogreške, model vodi premalo računa o podacima za treniranje, pa stoga previše pojednostavljuje model i s time nedovoljno nauči potrebne obrasce

ponašanja podataka. Model će ovom pogreškom naučiti krive relacije, ne uzimajući u obzir sve atribute prediktora. Ova se pogreška može opisati kao podtreniranost modela. Pogreške uzrokovane varijancom opisuju se kroz preveliko vođenje računa o podacima za treniranje modela gdje će u tom slučaju model zapamtiti podatke umjesto da iz njih nauči samo najpotrebnije uzorke. Time će doći do odličnog rezultata testiranja na podacima na kojima se model trenirao i vrlo loših rezultata testiranja na podacima koje prvi put detektira, što se može opisati pretreniranošću modela [30].

Usporedbu pretreniranosti, podtreniranosti i prikaza optimalnog balansa između varijance i pristranosti prikazano je u tablici 2.

Tablica 2: Usporedba pretreniranosti, podtreniranosti te ispravne treniranosti klasifikacijskog modela

	Pretreniranost	Ispravna treniranost	Podtreniranost
Opis	Visoka varijanca	Optimalni balans varijance i pristranosti	Visoka pristranost
Ilustracija Klasifikacije			

Izvor: Adaptirao student prema Shrivastava, A. 2020, 'Underfitting Vs Just right Vs Overfitting in Machine learning', Kaggle, online: <https://www.kaggle.com/getting-started/166897> (21.3.2023.)

4.2. LOGISTIČKA REGRESIJA

Logistička regresija je jedan od modela strojnog učenja koji se koristi za rješavanje problema binarne klasifikacije [29]. U statistici, regresijska analiza se sastoji od tehnika za modeliranje veza između izlazne varijable (engl. *output variable*) ili odzivne varijable (engl. *response variable*) i jedne ili više ulaznih varijabli (prediktora) [31]. U regresiji izlazna varijabla je modelirana kao funkcija ulaznih varijabli, regresijskih parametara ili koeficijenta i

termina slučajne pogreške koji predstavljaju varijaciju zavisne varijable neobjašnjenu funkcijom zavisnih varijabli i koeficijenata [31].

Jedan od najjednostavnijih i najpromjenjenijih regresijskih modela je linearna regresija koja se opisuje jednačbom [28]:

$$y(x) = \beta_0 + \beta_1 x + \varepsilon \quad (5)$$

Gdje su:

$y(x)$ - jednačba regresije

β_0 - nagib pravca (koeficijent regresije)

β_1 - sjecište pravca s y osi (koeficijent regresije)

$E(\varepsilon)$ - očekivana vrijednost nasumične pogreške

ε - nasumična pogreška

x - nezavisna (ulazna) varijabla

y - zavisna (izlazna) varijabla

Koeficijenti β_0 i β_1 predstavljaju nepoznanice koje je potrebno odrediti. Nasumična greška predstavlja pogrešku prilikom primjerice izvršenja određenog mjerenja. Pretpostavka je da je $E(\varepsilon) = 0$ s nepoznatom vrijednosti varijance σ^2 [28].

Pri određenom mjerenju očekivana vrijednost funkcije je:

$$E(y|x) = \beta_0 + \beta_1 \cdot x \quad (6)$$

ili kraće:

$$y(x) = \beta_0 + \beta_1 \cdot x \quad (7)$$

Vrijednost varijance je prema tome sljedeći izraz [28]:

$$\sigma^2 = \text{Var}(\beta_0 + \beta_1 \cdot x + \varepsilon) \quad (8)$$

Nepoznanice β_0 i β_1 se određuju metodom najmanjih kvadrata (engl. *least squares method*). Navedena metoda određuje β_0 i β_1 koeficijente na način da minimizira sumu

kvadrata razlike između izmjerenih vrijednosti y_i i predviđene vrijednosti pravca regresije.

Metoda najmanjih kvadrata opisana je formulom [28]:

$$S(\beta_0, \beta_1) = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 X_i)^2 \quad (9)$$

Kako bi se izveli sljedeći izrazi potrebno je svesti (7) na parcijalne diferencijalne jednadžbe. Kao rješenje diferencijalnih jednadžbi dobivaju se skupovi jednadžbi naziva normalne jednadžbe najmanjih kvadrata prikazana sljedećim izrazima [28]:

$$n \cdot \hat{\beta}_0 + \hat{\beta}_1 \cdot \sum_{i=1}^n x_i = \sum_{i=1}^n y_i \quad (10)$$

$$\hat{\beta}_0 \sum_{i=1}^n x_i + \hat{\beta}_1 \cdot \sum_{i=1}^n x_i^2 = \sum_{i=1}^n y_i x_i \quad (11)$$

Rješenje izvoda jednadžbi (8) i (9) je prikazano sljedećim izrazima [28]:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \quad (12)$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n x_i y_i - \frac{(\sum_{i=1}^n x_i)(\sum_{i=1}^n y_i)}{n}}{(\sum_{i=1}^n x_i^2) - \frac{(\sum_{i=1}^n x_i)^2}{n}} \quad (13)$$

Gdje je:

\bar{x} i vrijednost od x_i prema izrazu:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

\bar{y} - srednja vrijednost od y_i prema izrazu:

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

Konačno, model linearne regresije poprima sljedeći izraz:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \cdot x \quad (12)$$

Gdje je \hat{y} predviđena srednja vrijednost izlazne varijable y za željeni ulaz x [28].

Višestruka linearna regresija za razliku od jednostruke linearne regresije koja uspoređuje odnos jedne ulazne varijable s izlaznom varijablom, predstavlja model koji uspoređuje odnose između ulaznih s izlaznom varijablom. Pa je prema izrazu (5) ista aproksimirana sljedećom jednadžbom [29]:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \dots + \beta_n x_n + \varepsilon \quad (13)$$

Gdje su:

y - izlazna varijabla

β_i - koeficijenti regresije ($i = 0, 1, 2, \dots, n$)

x_i – ulazne varijable modela ($i = 0, 1, 2, \dots, n$)

ε - slučajna pogreška

Kod korištenja linearnog te višestrukog linearnog regresijskog modela, podrazumijeva se da postoji linearni odnos između ulaznih i izlazne varijable. Isto tako se podrazumijeva da su ulazne varijable kontinuirane odnosno da se radi o kontinuiranim brojevima umjesto o diskretnim vrijednostima klasa ili kategorija. Prema tome je Linearna regresija model strojnog učenja namijenjen za rješavanje regresijskih problema [29].

Osim linearnog i višestrukog linearnog regresijskog modela postoji i nelinearni regresijski model. On podrazumijeva da je odnos između ulaznih varijabli i izlazne varijable nelinearan. Nelinearni regresijski model opisan je jednadžbom [31]:

$$y = \frac{\alpha}{1 + e^{\beta t}} + \varepsilon \quad (14)$$

Gdje su:

α i β - parametri modela

ε - slučajna pogreška

Budući da model linearne regresije koristi kontinuirani raspon koji se kreće od $-\infty$ do ∞ , potrebno ga je prilagoditi da se kreće od 0 do 1. Taj problem se rješava upotrebom modela logističke regresije koju opisuje sigmoidna funkcija, opisana sljedećim izrazom.

$$\sigma(z_\sigma) = \frac{1}{1 + e^{-(z_\sigma)}} \quad (15)$$

Gdje je z_σ pravac linearne regresije te se opisuje izrazom:

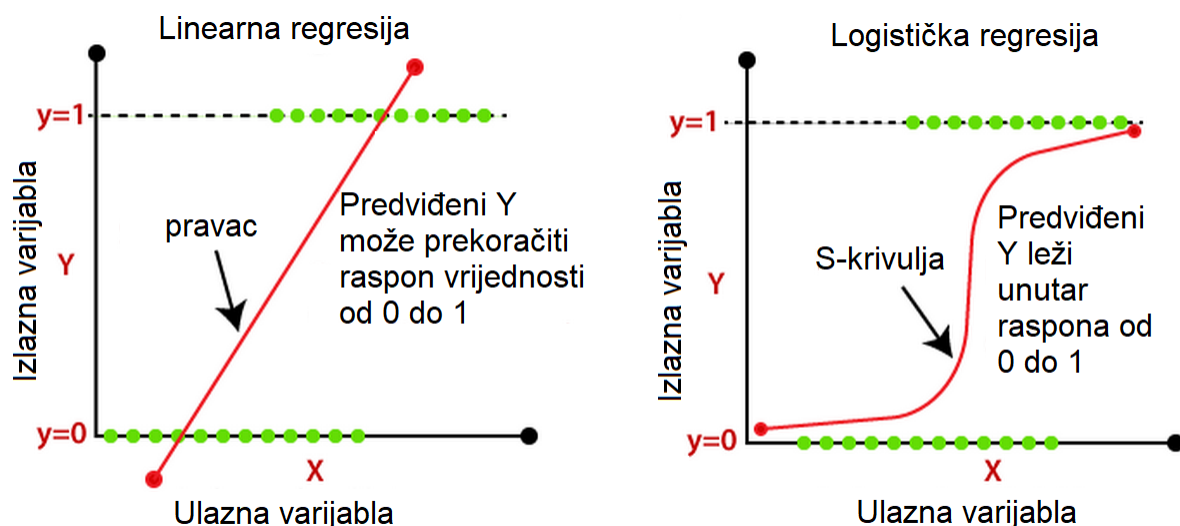
$$z_\sigma = \theta_0 + \theta_1 x_1 + \dots + \theta_n * x_n \quad (16)$$

Gdje su:

θ_i - koeficijenti regresije, prikazuju se i kao β_i ($i = 0, 1, 2, \dots, n$)

x_i – ulazne varijable ($i = 0, 1, 2, \dots, n$)

Pravac linearne regresije z_σ ujedno označuje i izlazne varijable modela. A θ_i označuje koeficijente regresije i njihov doprinos u veličini određene x_i varijable. U slučaju kada su koeficijenti regresije pozitivni postoji veća vjerojatnost da se zadani ishod dogodi. U suprotnom slučaju, odnosno kada su koeficijenti negativni, postoji manja vjerojatnost od zadanog ishoda. Model logističke regresije se može opisati kao linearna regresija za rješavanje klasifikacijskog problema. Za razliku od linearne regresije, logistička regresija ne zahtijeva linearni odnos između ulaznih i izlazne varijable [29]. Na slici 15 prikazana je grafička usporedba linearne regresije i logističke regresije.



Slika 15: Usporedba Linearne i Logističke regresije

Izvor: Adaptirao student prema 'Linear Regression vs Logistic Regression', javaTpoint, online:

<https://www.javatpoint.com/linear-regression-vs-logistic-regression-in-machine-learning>

Iz slike 15 iz grafičke usporedbe modela vidi se da je pravac linearne regresije aproksimiran S-krivuljom koja predstavlja sigmoidnu funkciju (15). Hipoteza linearne regresije prikazana je sljedećim izrazom [32]:

$$h_{\theta(x)} = \beta_0 + \beta_1 x \quad (15)$$

Izrazom (17) definirana je sigmoidna funkcija pomoću hipoteze linearne regresije [32]:

$$\sigma(z_{\sigma}) = \sigma(\beta_0 + \beta_1 x) \quad (16)$$

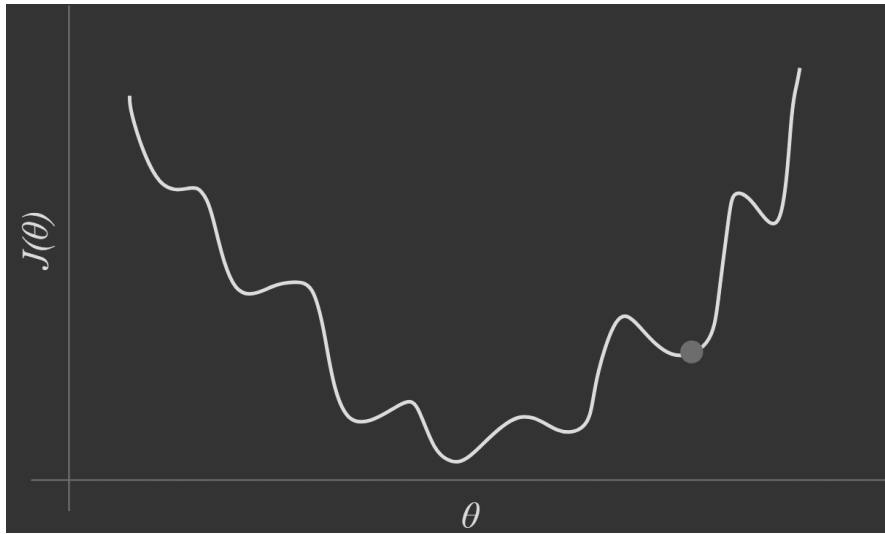
Ubacivanjem (15) u prethodni izraz dobava se hipoteza logističke regresije [32]:

$$h_{\theta(x)} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \quad (17)$$

Funkcijom cilja kod linearne regresije minimizira se suma razlika između predviđene vrijednosti (\hat{y}) i stvarne vrijednosti izlazne varijable (y). Funkcija cilja $J(\theta)$ opisana je izrazom [32]:

$$J(\theta) = \left(\frac{1}{2m}\right) \sum_{i=1}^m h_{\theta}(x_i) - y_i \quad (18)$$

Funkcija (17) nije primjenjiva u problemu logističke regresije jer prilikom aktivacije ista postaje nekonveksna i puna lokalnih minimuma. Nekonveksna $J(\theta)$ funkcija je prikazana na slici 16.



Slika 16: Nekonveksna funkcija $J(\theta)$

Izvor: [32]

Ako se u funkciji (18) unutar sume ubaci 2 koji stoji u nazivniku izraza ispred sume dobiva se sljedeći izraz [32]:

$$C(h_{\theta}(x_i), y_i) = \frac{1}{2} (h_{\theta}(x_i) - y_i)^2 \quad (20)$$

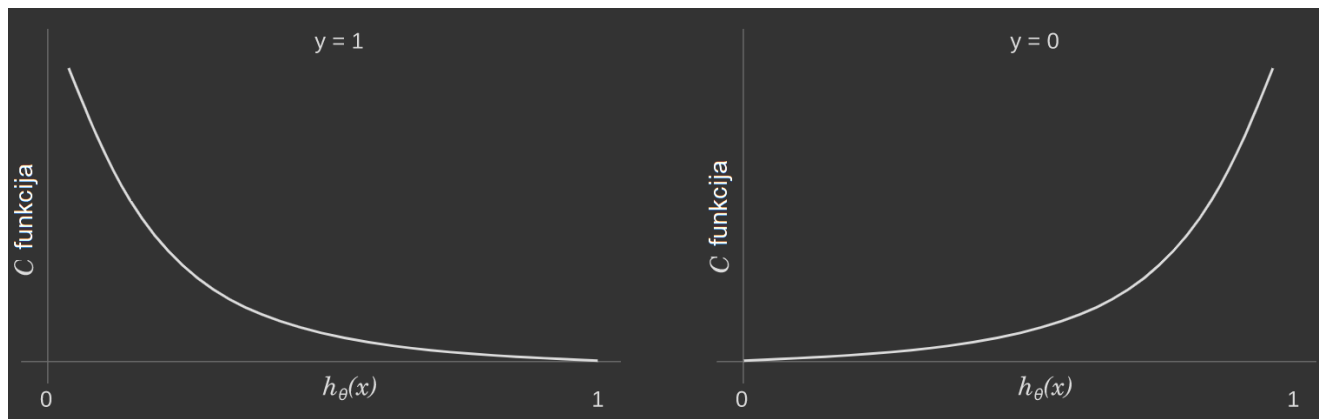
Funkcija C u (19) ima dvije ulazne varijable a to su: $h_{\theta}(x_i) - y_i$. Konačni oblik funkcije $J(\theta)$ se može zapisati na sljedeći način:

$$J(\theta) = \left(\frac{1}{m}\right) \sum_{i=1}^m C(h_{\theta}(x_i), y_i) \quad (21)$$

Logistička regresija zahtijeva upotrebu konveksne funkcije cilja pa je potrebno odabrati drugi izraz unutar sume u (21) $C(h_{\theta}(x_i), y_i)$.

$$C(h_{\theta}(x_i), y_i) = \begin{cases} -\ln(h_{\theta}(x_i)) & \text{za } y = 1 \\ -\ln(1 - h_{\theta}(x_i)) & \text{za } y = 0 \end{cases} \quad (22)$$

Izraz iz sume 21 je prikazan grafički na slici 17, što ujedno predstavlja i grafički prikaz funkcije cilja (C funkcija) za logističku regresiju.



Slika 17: Grafički prikaz funkcije cilja za Logističku regresiju

Izvor: Adaptirao student prema [32]

Funkcija cilja za logističku regresiju može se prikazati u jednoj liniji na sljedeći način:

$$C(h_{\theta}(x_i), y_i) = -y_i \cdot \ln(h_{\theta}(x_i)) - (1 - y_i) \cdot \ln(1 - h_{\theta}(x_i)) \quad (23)$$

Funkciju cilja za logističku regresiju može se zapisati ubacivanjem u (23):

$$J(\theta) = \left(\frac{1}{m}\right) \sum_{i=1}^m -y_i \cdot \ln(h_{\theta}(x_i)) - (1 - y_i) \cdot \ln(1 - h_{\theta}(x_i)) \quad (24)$$

$$J(\theta) = -\left(\frac{1}{m}\right) \sum_{i=1}^m -y_i \cdot \ln(h_{\theta}(x_i)) - (1 - y_i) \cdot \ln(1 - h_{\theta}(x_i)) \quad (25)$$

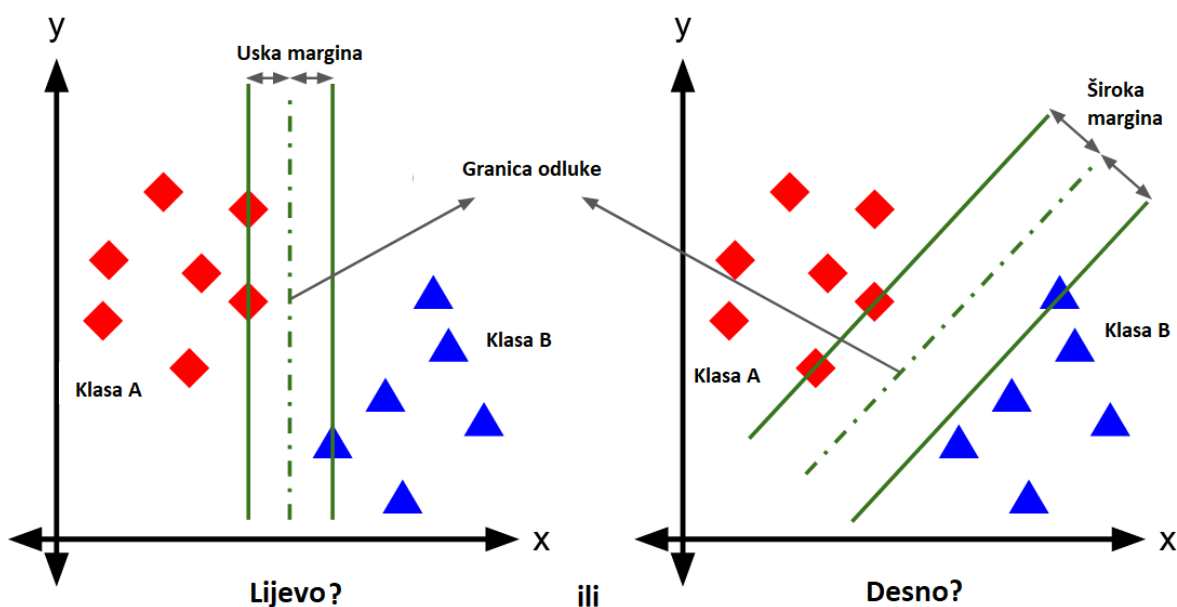
Formalni način zapisa navedene funkcije se izvodi na sljedeći način:

$$\min_{\theta} J(\theta) \quad (26)$$

Jedna od velikih prednosti logističke regresije je jednostavnost implementacije i veća učinkovitost kada je u pitanju manji broj ulaznih varijabli. S druge strane, nedostatak osim smanjenja učinkovitosti s većim brojem varijabli, je i potrebna pretvorba varijabli koje nisu linearnog karaktera [33].

4.3. KLASIFIKATOR POTPORNIH VEKTORA

Klasifikator potpornih vektora ili SVM (eng. *Support-Vector Machine*) klasifikator je algoritam klasifikacije koji spada u grupu nadziranog učenja i kao takav se koristi za probleme binarne klasifikacije [34]. Kao izlaz SVM daje hiper ravninu (eng. *hyper-plane*), koja je višedimenzionalni prostor koji služi za razdvajanje podataka u dvije klase [34]. Primjer hiper ravnine u SVM klasifikaciji prikazan je na slici 18.



Slika 18: Prikaz hiper ravnine SVM klasifikacije

Izvor: Adaptirao student prema [34]

Sad se postavlja pitanje koja je ravnina optimalnija na lijevoj ili desnoj slici. Odgovor je ona koja ima širi prostor ili marginu između klasa, dakle desna slika [34]. Ako se za objašnjenje problema koriste linearno odvojive klase prikazane unutar dvije dimenzije ili osi koordinatnog sustava iste mogu biti odvojene pravcem. Funkciju pravca y ili linije koja odvaja navedene klase opisana je izrazom [35]:

$$y = ax + b \quad (27)$$

Gdje su:

y - funkcija pravca

a – nagib pravca

b – sjecište s y osi

Uvrštavanjem u izraz (26) vrijednosti $X = (x, y)$ i smjer vektora $W = (a, -1)$ [35]:

$$W \cdot X + b = 0 \quad (28)$$

Gdje su:

$W \cdot X$ = skalarni produkt između veličine i smjera vektora

X - veličina vektora (skup ulaznih varijabli ($X = x_1, x_2, \dots, x_n$)), a njegova normala je $||X||$.

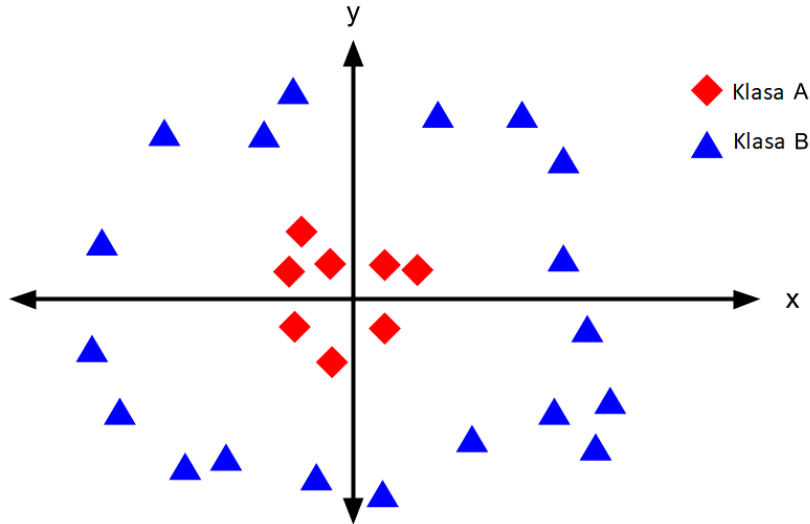
Formula za izračun normale vektora X je [35]:

$$||X|| = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2} \quad (29)$$

W - smjer vektora opisan izrazom [35]:

$$W = \left(\frac{x_1}{||x||}, \frac{x_2}{||x||} \right) \quad (30)$$

Iako je jednačba (27) izvedena iz svojstva dvodimenzionalnih vektora, ista se može primijeniti i u sklopu odvajanja klasa kod bilo kojeg broja dimenzija [35]. U slučaju da su podaci raspoređeni u klase koje su nelinearno odvojive, kao što je prikazano na slici 19, pristup rješavanju problema je drukčiji.



Slika 19: Prikaz nelinearno odvojivih klasa podataka

Izvor: Adaptirao student prema [34]

U ovom slučaju se ne može povući pravac, odnosno hiper ravnina koja bi odvojila dvije klase. Stoga, kako bi se riješio zadani problem potrebno je koristiti Kernelovu metodu transformacije. Kernelova metoda se bazira na Mercerovom teoremu koji je definiran uvjetom da ako funkcija $K(a, b)$ zadovoljava sve uvjete koja se nazivaju Mercerova ograničenja u tom slučaju postoji funkcija koja preslikava a i b u višu dimenziju. Navedena funkcija je opisana izrazom [36]:

$$K(a, b) = \Phi(a)^T \cdot \Phi(b)$$

Gdje je Φ Kernel.

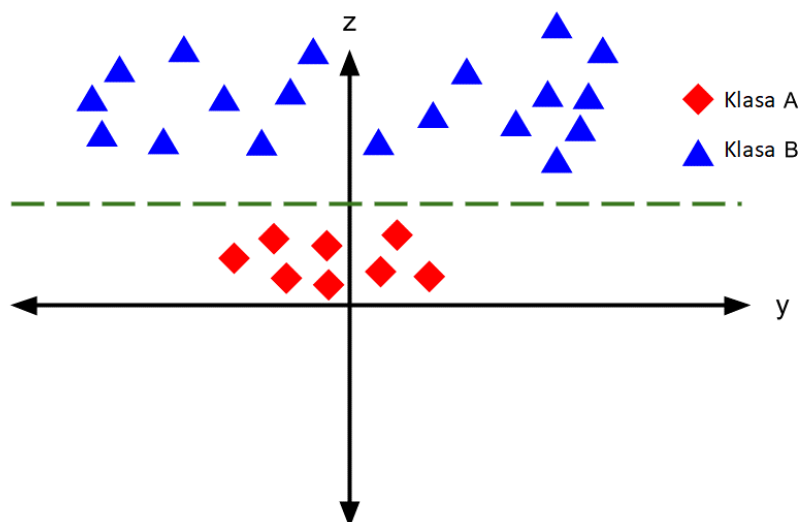
$$\text{Max } l_d = \sum_{i=1}^l \lambda_i \frac{\lambda^T \lambda K}{2} \text{ prema } 0 \leq \lambda \leq C, \sum_{i=1}^l \lambda_i y_i = 0 \quad (31)$$

Gdje je $K(i, j) = y_i y_j x_i x_j$, budući da je $y_i y_j = 1$, pa je $K(i, j) = x_i x_j$. U izrazu (30) radi se o linearnom Kernelu pa je u općem obliku prikazan izrazom [36]:

$$K(x_i, x_j) = \Phi(x_i)^T \cdot \Phi(x_j) \quad (32)$$

Gdje je Φ Kernel .

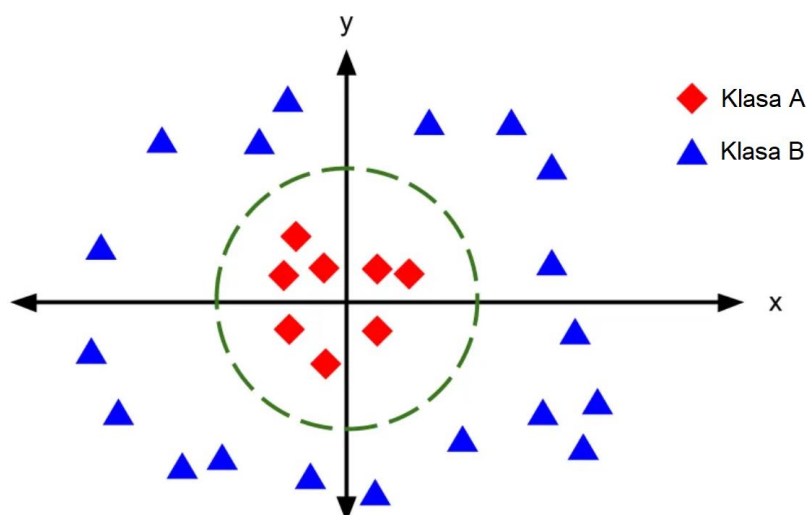
Kernelova metoda transformacije podrazumijeva dodavanje treće dimenzije, odnosno z osi. Dodavanjem z osi omogućuje se projekcija iz x - y prikazano na slici 19 u y - z koordinatni sustav prikazan na slici 20.



Slika 20: Transformacija u 3D sustav, te projekcija u y - z koordinatni sustav

Izvor: Adaptirao student prema [34]

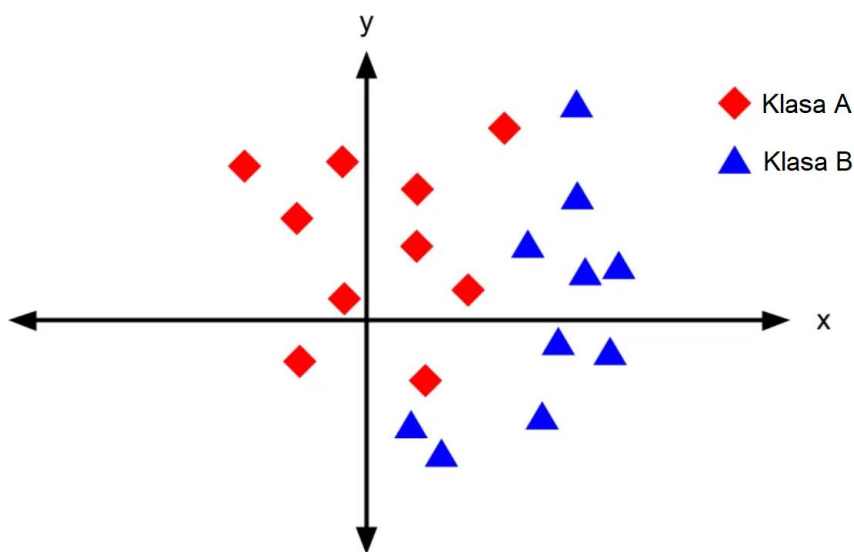
Projekcijom u y - z koordinatni sustav može se povući linija između dviju klasa, koja označuje hiper ravninu. Nakon pronalaska hiper ravnine potrebno je ponovo prebacivanje natrag u dvodimenzionalni sustav, gdje se može vidjeti kružna granica, prikazana na slici 21, koja odvaja klase [34].



Slika 21: Transformacija u 2D sustav gdje se linija transformira u krug

Izvor: Adaptirao student prema [34]

U slučaju preklapanja klasa kako je prikazano na slici 22 postoji više mogućnosti rješavanja problema.



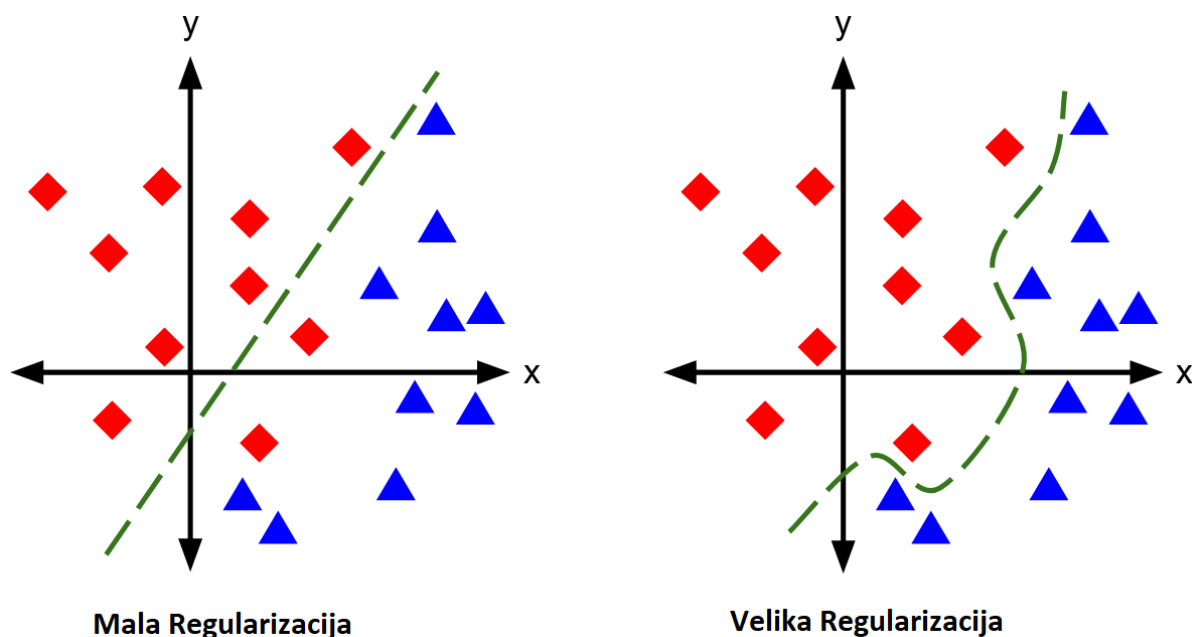
Slika 22: Preklapanje klasa

Izvor: Adaptirao student prema [34]

Prva je mogućnost tolerancija odstupanja od pojedine klase (engl. *outlier*) što podrazumijeva toleranciju na pogrešnu klasifikaciju pojedinih uzoraka klase. Drugi način je nulta tolerancija na odstupanje, što znači maksimalna točnost klasifikacije. Veća točnost klasifikacije može se postići namještanjem pojedinih hiperparametara modela poput: kernela, regularizacije, *gamma*-e i margine [34].

Kernel podrazumijeva metodu korištenja linearnog klasifikatora za rješavanje ne-linearnog klasifikacijskog problema. To znači transformiranje linearno nerazdvojivih klasa (slika 19) u linearno razdvojive (slika 20), već opisanim postupkom. On SVM algoritmu određuje mjeru izbjegavanja ili mjeru za izbjegavanje krive klasifikacije na pojedinom uzorku treniranja [34].

Za male vrijednosti regularizacije, algoritam će odabrati šire margine, čak i ako ta hiper ravnina pogrešno klasificira veći broj uzoraka. S druge strane, velika vrijednost regularizacije učini će da algoritam traži manju marginu koja razdvaja hiper ravninu ako ta hiper ravnina učini bolji zadatak klasifikacije. Primjer velike i male regularizacije je prikazan na slici 23 [34].



Slika 23: Usporedba male i velike regularizacije

Izvor: Adaptirao student prema [34]

Γ parametar definira koliko daleko doseže utjecaj pojedinog trening uzorka (male vrijednosti znače daleki utjecaj, veće vrijednosti bliži utjecaj). Drugim riječima, mala vrijednosti Γ -e znači da uzorci daleko od vjerojatne linije razdvajanja klasa imaju utjecaj pri izračunu krajnje linije razdvajanja. Dok veće vrijednosti Γ -e znače da su uzorci bliže vjerojatoj liniji razdvajanja uzeti u obzir pri kalkulaciji krajnje linije [34].

Margina je udaljenost između separacijske linije i najbližeg uzorka klase. Optimalna margina je ona kojoj je najveća udaljenost između obje linije odvajanja i najbližeg uzorka obje klase. To omogućuje da uzorci budu u svojim zadanim klasama, bez pojave preklapanja s susjednom klasom [34].

Prednosti SVM algoritma jesu: visoke performanse s jasnom separacijom margina, uspješnost pri radu s velikim brojem dimenzija, efikasnost u slučajevima kad je broj dimenzija veći od broja uzoraka te se koriste podskup trening skupa u funkciji odluke pa su učinkovitiji pri upotrebi memorije. Nedostaci se očituju u: lošim performansama kad je u pitanju veći set podataka jer se vrijeme treniranja značajno povećava, nedostatak uspješnosti u raspoznavanju klasa u slučajevima kad skup podataka ima puno šuma [34].

4.4. KLASIFIKATOR EKSTREMNOG POJAČANJA GRADIJENATA

Model ekstremnog pojačanja gradijenata poznatiji kao *XGBoost* (eng. *eXtreme Gradient Boosting*), je algoritam baziran na spoju algoritama baziranim na stablima odlučivanja s algoritmom pojačanja gradijenata. U problemima predviđanja koji se zasnivaju na nestrukturiranim podacima poput slika, teksta i sl., neuronske mreže imaju tendenciju nadmašivanja svih ostalih algoritma. Međutim, kada se radi o malom do srednjem skupu tabličnih strukturiranih podataka, modeli koji se baziraju na stablima odlučivanja trenutno se smatraju najboljima u klasi [37]. U tablici 3 prikazana je evolucija algoritma baziranih na stablima odlučivanja.

Modeli strojnog učenja	Opis modela
Stabla odlučivanja (engl. <i>Decision Trees</i>)	Grafička reprezentacija mogućih rješenja koja je bazirana na uvjetima odlučivanja ovisna o prijašnjim odgovorima na pitanja, od kojih je građena struktura stabla odlučivanja.
Bagging	Složeni algoritam koji kombinira predviđanja iz višestrukih stabla odlučivanja.
Random Forest	Algoritam baziran na <i>Bagging</i> metodi gdje je pri svakoj izradi stabla odlučivanja nasumično odabran podskup značajki prediktora kako bi se izgradila kolekcija stabla odlučivanja.
Pojačavanje (engl. <i>Boosting</i>)	Pojedina stabla odlučivanja spajaju se sekvencijalno kako bi se minimizirale pogreške izgradnje prethodnih stabla, dok se istovremeno pojačava utjecaj od najuspješnije izrađenih stabla odlučivanja.
Pojačanje gradijenata (engl. <i>Gradient Boosting</i>)	Uvodi se algoritam spuštanja gradijenata kako bi se smanjile pogreške klasifikacije kod sekvencijskih modela stabla odlučivanja.
Ekstremno pojačanje gradijenata (engl. <i>Extreme Gradient Boosting</i>)	Pojačanje gradijenata s sklopovskim i programskim poboljšanjima algoritma.

Tablica 3: Usporedba algoritama baziranih na stablima odlučivanja

Izvor: Adaptirao student prema [37]

Iz tablice 3 vidljivo je da je svaki sljedeći model počevši od stabla odlučivanja napredniji od prethodnog/prethodnih kombinirajući prijašnje algoritme s naprednijim tehnologijama. Primjerice *Random Forest* algoritam je spoj stabla odlučivanja s *Bagging* algoritmom. *Random Forest* uzima samo podskup značajki odabranih nasumično kako bi kreirali šumu (engl. *Forest*) ili kolekciju stabla odlučivanja. *Bagging* algoritam kombinira predikcije iz više stabala odlučivanja kroz većinski mehanizam glasanja. Stabla odlučivanja su grafička reprezentacija mogućih rješenja odluke temeljene na određenim uvjetima. Poboljšanje *Random Forest* modela je *Boosting* model gdje se model uči na svojim prethodnim pogreškama napravljenim na prethodnim stablima odlučivanja, pa se nastoji ispraviti pri gradnji nadolazećeg stabla. *Gradient Boosting* je algoritam na kojem se minimiziraju pogreške upotrebom *Gradient Descent* algoritma [37].

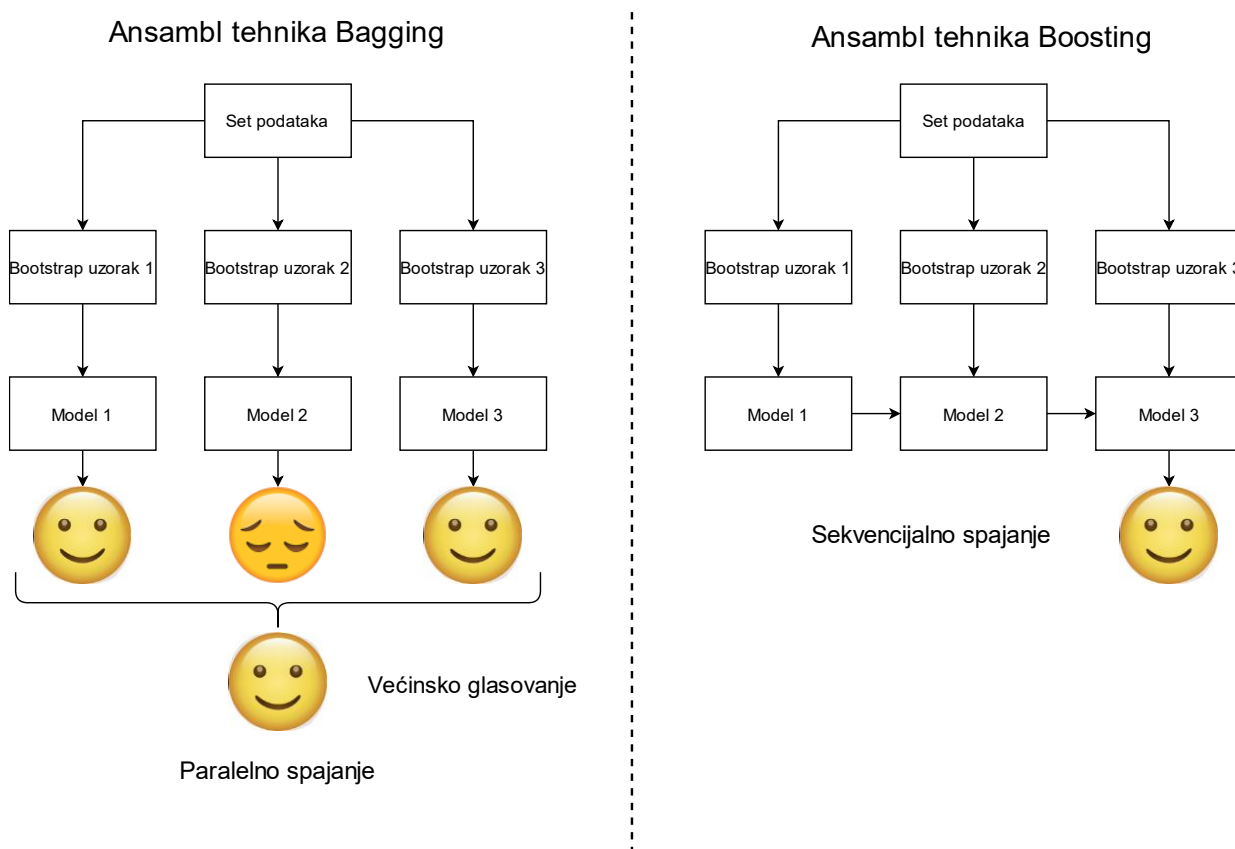
Kako bi se objasnile optimizacije algoritama potrebno je objasniti princip rada algoritma na kojemu se bazira *XGBoost*, a to je model pojačanja gradijenta.

4.4.1. Model pojačanja gradijenata

Model pojačanja gradijenata (engl. *Gradient Boosting*) je algoritam koji koristi ansambl tehnike (engl. *Ensemble Techniques*). Složene tehnike se baziraju na kolekciji slabijih prediktora (engl. *Weak Learners*), koji u kombinaciji daju značajno bolje rezultate nego pojedinačni slabiji prediktor. Postoje dva tipa složenih tehnika to su *Bagging* i *Boosting*.

Bagging ili prikupljanje *Bootstrap-a* je algoritam strojnog učenja u kojem se niz neovisnih prediktora gradi uzimanjem uzoraka sa zamjenom. Pojedinačni ishodi se zatim kombiniraju prosjekom (regresija) ili većinskim glasovanjem (klasifikacija) kako bi se dobilo konačno predviđanje. Često korišten algoritam u ovom području je *Random Forest*. *Boosting* je algoritam strojnog učenja u kojem se slabi prediktori pretvaraju u snažne prediktore. Slabi prediktori su klasifikatori koji uvijek imaju malo bolju izvedbu od slučajnosti, bez obzira na distribuciju podataka o obuci. U *Boosting-u*, predviđanja su sekvencijalna pri čemu svaki sljedeći prediktor uči iz pogrešaka prethodnih prediktora. Stabla pojačanja gradijenata (engl. *Gradient Boosting Trees*) ili GBT često je korištena metoda u ovoj kategoriji [38]. Usporedba *Bagging* i *Boosting* tehnika prikazana je na slici 25. Gdje se može vidjeti da se *Bagging* zasniva

na paralelnom procesu spajanja stabla odlučivanja, dok se *Boosting* tehnika temelji na sekvencijalnom povezivanju.



Slika 24: Usporedba tehnika bagging i boosting

Izvor: Adaptirao studen prema [39]

Bagging ima mnogo nekoreliranih stabala u konačnom modelu što pomaže u smanjenju varijance. *Boosting* će smanjiti varijance u procesu izgradnje sekvencijalnih stabala. U isto vrijeme, njegov fokus ostaje na premošćivanju razlike između stvarnih i predviđenih vrijednosti smanjenjem rezidualne vrijednosti (engl. *residual*) pa se time također smanjuje i pristranost [38].

Primjer klasifikacije korištenjem GBT modela izvest će se na setu podataka u kojem se predviđa ima li osoba bolest srca (engl. *Heart Disease*) bazirano na ulaznim podacima koji su: bol u prsima (engl. *Chest Pain*), dobra cirkulacija krvi (engl. *Good Blood Circulation*) te začepljene arterije (engl. *Blocked Arteries*) [38]. Tablica podataka prikazana je na slici 26.

Bol u prsima	Dobra cirkulacija krvi	Začepljene arterije	Bolest srca
Ne	Ne	Ne	Ne
Da	Da	Da	Da
Da	Da	Ne	Da
Da	Ne	Ne	Da
Da	Ne	Da	Ne
Ne	Da	Ne	Ne

Slika 25: Skup podataka za predviđanje bolesti srca

Izvor: [38]

U početnoj predikciji počinje se s podatkom koji predstavlja inicijalnu predikciju za svakog pojedinca. Za klasifikaciju to će biti jednako logaritmu vjerojatnosti da neko ima bolest srca. Budući da 4 osobe imaju bolest srca, dok preostale 2 osobe nemaju bolest srca. To se može prikazati formulom:

$$\log(P_{srce}) = \log\left(\frac{4}{2}\right) = 0.6931 \quad (32)$$

Gdje je P_{srce} vjerojatnost od pojave bolesti srca.

Tu vjerojatnost je potrebno pretvoriti koristeći logističku funkciju prikazanu formulom:

$$\frac{\exp^{\log(P_{srce})}}{1 + \exp^{\log(P_{srce})}} = \frac{\exp^{\log(\frac{4}{2})}}{1 + \exp^{\log(\frac{4}{2})}} = 0.6667 \quad (33)$$

Ako se uzme granica odluke za vjerojatnost 0.5, to znači da je inicijalna predikcija da svi ispitanici imaju bolest srca, što nije slučaj. Nakon toga je potrebno izračunati rezidualne vrijednosti za svaku predikciju. Rezidualne vrijednosti se računaju uporabom formule:

$$e = y - \hat{y} \quad (34)$$

Gdje su:

e - Rezidualna vrijednost

y - Stvarna vrijednost

\hat{y} - Predviđena vrijednost

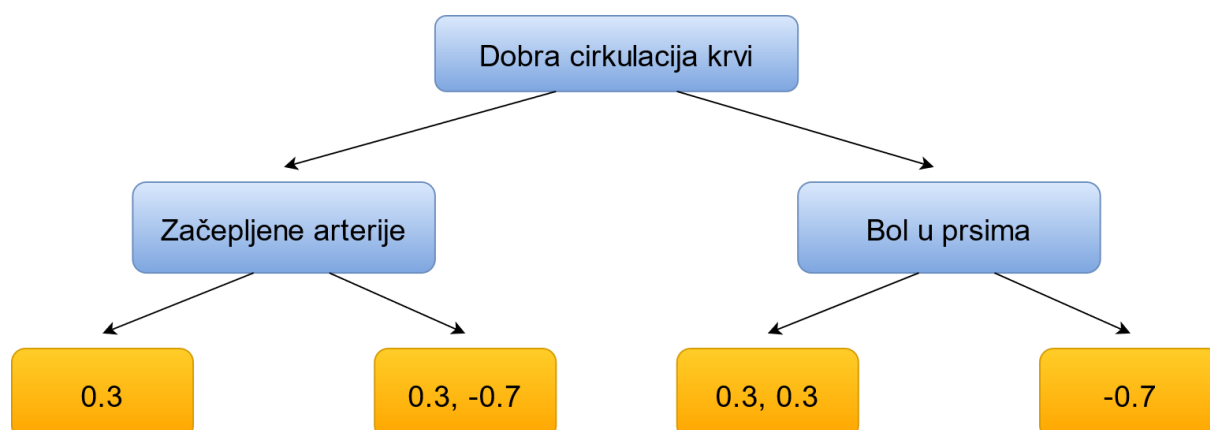
U slučaju da su $y = 1$ te $\hat{y} = 0.7$ (zaokružena na jednu decimalu) onda će rezidualna vrijednost $e = 0.3$. Tablica podataka nakon izračuna rezidualnih vrijednosti prikazana je na slici 26.

Bol u prsima	Dobra cirkulacija krvi	Začepljene arterije	Bolest srca	Stvarne vrijednosti (y)	Rezidualne vrijednosti (e=y- \hat{y})
Ne	Ne	Ne	Ne	0	-0,7
Da	Da	Da	Da	1	0,3
Da	Da	Ne	Da	1	0,3
Da	Ne	Ne	Da	1	0,3
Da	Ne	Da	Ne	1	0,3
Ne	Da	Ne	Ne	0	-0,7

Slika 26: Tablica nakon izračuna rezidualnih vrijednosti

Izvor: Adaptirao student prema [38]

Sljedeći korak je predviđanje rezidualnih vrijednosti koji uključuje izradu stabla odlučivanja kako bi se predvidjele rezidualne vrijednosti koristeći ulazne varijable. Stablo odlučivanja prikazano je na slici 27.



Slika 27: Izrada stabla odlučivanja na temelju ulaznih varijabli

Izvor: Adaptirao student prema [38]

Iz konstrukcije stabla odlučivanja, vidi se da mu je maksimalna dubina (engl. *max depth*) jednaka 2. Kako je broj rezidualnih vrijednosti veći od broja listova (engl. *leaves*), može se vidjeti da neki rezidualne vrijednosti završavaju na istom listu.

Nadalje, potrebna transformacija kako bi se dobile predviđene rezidualne vrijednosti u smislu inicijalne formule (33). Predviđena rezidualna vrijednost može se izračunati sljedećom formulom:

$$\hat{y}_e = \frac{\sum e_i}{\sum [In_i * (1 - In_i)]} \quad (35)$$

Gdje su:

$$i = 1, 2, 3, \dots, n$$

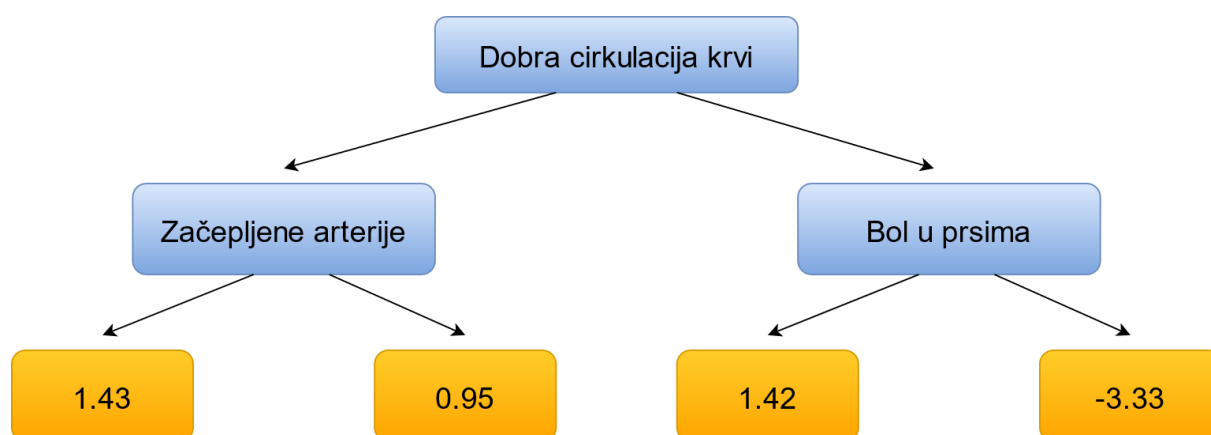
e_i – Rezidualna vrijednost pojedinog uzorka

In_i – Inicijalna vjerojatnost

Ako se formula (35) primjeni na prvi list dobije se:

$$\frac{0.3}{0.7 * (1 - 0.7)} \approx 1.43$$

Na isti način se izračunavaju vrijednosti rezidualne vrijednostima ostalih listova stabla odlučivanja, što je i prikazano na slici 28.



Slika 28: Modificirano stablo odlučivanja

Izvor: [24]

Izračun nove vjerojatnosti od bolesti srca je sljedeći korak u kojemu je potrebno proslijediti svaki uzorak seta podataka, kroz čvorove novo formiranog stabla. Predviđene rezidualne vrijednosti dobivene za svako opažanje bit će dodane prethodnom predviđanju kako bi se utvrdilo ima li osoba srčanu bolest. One će biti pomnožene sa zadanom stopom učenja (engl. *Learning Rate*) i dodane prijašnjoj predikciji.

Stopa učenja je hiperparametar koji sprječava pojavu pretreniranosti modela. Implementacija stope učenja zahtijeva gradnju više stabla odlučivanja, kako bi se uzeli manji koraci do krajnjeg rezultata. Ti mali koraci pomažu u postizanju optimalnog balansa između pristranosti i varijance.

Izračun nove vjerojatnosti $\log(P_{srce})$ dobiva se sljedećom formulom:

$$\log(P_{srce}) = In + Lr * \check{y}_e \quad (36)$$

Gdje su:

In - Inicijalna predikcija

Lr - Stopa učenja

\check{y}_e – Predviđeni rezidualne vrijednosti

Ako se (6) primjeni na prvi list dobiva se:

$$\log(P_{srce}) = 0.7 + (0.2 * 1.43) = 0.99 \quad (37)$$

Pretvaranjem $\log(P_{srce})$ u vrijednost vjerojatnosti dobiva se sljedeća formula:

$$\frac{\exp^{\log(P_{srce})}}{1 + \exp^{\log(P_{srce})}} = \frac{\exp^{0.99}}{1 + \exp^{0.99}} \approx 0.73 \quad (38)$$

Ako se isti postupak izračuna primjeni na ostale listove stabla odlučivanja, dobivaju se rezultati prikazani u stupcu *nove predviđene vrijednosti*, te se iz tih novih predviđanja mogu izračunati nove rezidualne vrijednosti prikazane u susjednom stupcu (slika 29).

Bol u prsima	Dobra cirkulacija krvi	Začepljene arterije	Bolest srca	Stvarne vrijednosti (y)	Nove predviđene vrijednosti (\hat{y})	Rezidualne vrijednosti ($e=y-\hat{y}$)
Ne	Ne	Ne	Ne	0	0,51	-0,51
Da	Da	Da	Da	1	0,73	0,27
Da	Da	Ne	Da	1	0,73	0,27
Da	Ne	Ne	Da	1	0,72	0,28
Da	Ne	Da	Ne	1	0,72	0,28
Ne	Da	Ne	Ne	0	0,63	-0,63

Slika 29: Dodavanje izračuna novih predikcija za dobivanje novih rezidualnih vrijednosti

Izvor: Adaptirao student prema [38]

Sad kada postoje izračuni za nove rezidualne vrijednosti, iste se mogu koristiti za izradu sljedećeg stabla odlučivanja po postupku koji je prikazan na slici 26.

Potrebno je ponavljati postupke od slike 26 do slike 29 sve dok izračunate rezidualne vrijednosti ne poprime vrijednost koja je približno jednaka nuli ili dok broj ponavljanja (iteracija) odgovara vrijednostima zadanog hiperparametra, koji označuje broj stabla odlučivanja.

Nakon izračuna izlaznih vrijednosti za sva stabla odlučivanja, potrebno je ubaciti $\log(P_{srce})$ u logističku funkciju (34). Prema zadanoj granici odluke 0.5, ako je vjerojatnost veća od 0.5 osoba je klasificira kao osoba s srčanom bolesti, u suprotnom se osoba klasificira kao osoba bez bolesti srca.

Najvažniji hiperparametri ovog modela jesu: broj stabla odlučivanja (engl. *Number of Estimators*), stopa učenja, maksimalna dubina pojedinog stabla odlučivanja (engl. *Maximum Depth*). Maksimalna dubina limitira broj čvorova pojedinog stabla [23].

4.4.2. Poboljšanja XGBoost modela u odnosu na model pojačanja gradijenata

Optimizacije i poboljšanja *XGBoost-a* u odnosu na GBT model omogućene su zbog ugradnje: algoritama za pronalazak podjela, paraleliziranog učenja, svjesnosti prorijeđenosti, svjesnosti pristupa priručnoj memoriji, izvan jezgrene obrade, regularizacije te obrezivanja stabla [40].

Za pronalazak najbolje podjele unutar kontinuirane varijable (značajke) podaci moraju biti sortirani i u potpunosti se uklopiti u memoriju što je problem kod velikih skupova podataka. Za rješavanje tog problema koriste se algoritmi za pronalazak podjela, točnije aproksimativni algoritam. Kod tog algoritma, kandidatne točke razdvajanja predlažu se na temelju postotka distribucije značajke. Kontinuirane značajke grupiraju se u spremnike koji se dijele na temelju kandidatnih točaka razdvajanja. Najbolje rješenje za kandidatne točke razdvajanja odabire se iz prikupljene statistike unutar spremnika [40].

Paralelizirano učenje je izvedeno na način da se podaci pohranjuju u jedinice u memoriji koje se nazivaju blokovi. Svaki blok ima podatkovne stupce poredane prema odgovarajućoj vrijednosti značajke. Ovo izračunavanje potrebno je napraviti samo jednom prije treninga i kasnije se može ponovno koristiti. Sortiranje blokova može se odvojeno izvesti te biti raspoređeno između paralelnih dretva (engl. *Threads*) CPU-a [40].

Svjesnost prorijeđenosti definira se kao svojstvo algoritma da raspozna uzorke u podacima koji sadrže praznine (nedostajuće vrijednosti) odnosno *NaN* (engl. *Not a Number*) vrijednosti. *XGBoost* algoritam raspoznaje uzorke prorijeđenosti u podacima i posjećuje samo zadani smjer (unos koji ne nedostaju) u svakom čvoru. Svjesnost pristupa priručnoj memoriji (engl. *cache*) izvedena je s ciljem prevencije promašaja priručne memorije CPU-a, odabire se ograničen broj uzoraka po bloku. Za podatke koji ne stanu u memoriju, dijele se na više blokova, te se svaki blok sprema na disk. Pojedini blok se komprimira po stupcima pa se dekomprimira u hodu zasebnom dretvom za vrijeme čitanja diska. Ta tehnika naziva se izvan jezgrene obrada [40].

Dodavanje regularizacije *XGBoost* algoritmu omogućuje se kontrola kompleksnosti modela te se time preventira pretreniranost modela [40]. Kriterij za prestanak razdvajanja stabla odlučivanja je definiran unutar hiperparametra zaduženog za maksimalnu dubinu stabla. Kod *XGBoost* algoritma je ta značajka nazvana *obrezivanje stabla* jer je skraćivanje čvorova unutar

stabla izvedeno unatraske (od najdonjeg lista prema gore) time se postiže značajno brža izvedba algoritma [37].

4.5. POKAZATELJI USPJEŠNOSTI KLASIFIKACIJE

4.5.1. Matrice konfuzije

Točnost klasifikacije može se prikazati pomoću matrica konfuzije. Matrice konfuzije predstavljaju mjeru točnosti za potrebe klasifikacije kod strojnog učenja gdje izlazne vrijednosti mogu biti dvije ili više klasa [41]. To je tablica sa 4 različite kombinacije predviđenih i stvarnih vrijednosti [25], kako je i prikazano na slici 30.

		Predviđene klase	
		Negativna 0	Pozitivna 1
Stvarne klase	Negativna 0	TN	FP
	Pozitivna 1	FN	TP

Slika 30: Pojašnjenja značenja pojedinog dijela matrice konfuzije

Izvor: Adaptirao student prema [41]

Opis i pojašnjenje značenja natpisa na slici 29:

1. TN- ispravno negativni (engl. *True Negative*), predviđeno=0 i stvarno=0
2. FP- lažno pozitivni (engl. *False Positive*), predviđeno=1 i stvarno=0
3. FN- lažno negativni (engl. *False Negative*), predviđeno=0 i stvarno=1
4. TP- ispravno pozitivni (engl. *True Positive*), predviđeno=1 i stvarno=1

Na slici 29 TP označuje ispravno predviđanje pozitivne klase, u tom slučaju znači da su predviđena klasa i stvarna klasa uzoraka 1. TN označuje broj ispravnih predviđanja negativne klase, što bi značilo da su stvarna i predviđena jednake 0 [41].

S druge strane, FP ima značenje lažne ili neispravne klasifikacije pozitivnih uzoraka, dok FN znači neispravna klasifikacija negativnih uzoraka. U djelu matrice gdje se prikazuje FP, prikazan je broj netočno klasificiranih uzoraka koji su klasificirani kao pozitivni, a zapravo su negativni. FN označuje broj pozitivnih uzoraka koji su klasificirani kao negativni [41].

4.5.2. Točnost klasifikacije modela

Točnost klasifikacije modela (engl. *accuracy score*) je najčešći mjera pri evaluaciji modela dobiva se omjerom broja točno klasificiranih uzoraka i ukupnog broja uzoraka. Pomoću podataka dobivenih iz matrice konfuzije formula za točnost izvedena je na sljedeći način [42]:

$$\text{Točnost} = \frac{TP + FN}{TP + FP + TN + FN} \quad (38)$$

4.5.3. Preciznost klasifikacije modela

Preciznost (engl. *precision score*) se može definirati kao mjera koliko je točno pozitivnih predviđanja u odnosu na zbroj točno pozitivnih i lažno pozitivnih predviđanja. Što se može opisati sljedećom formulom [42].

$$\text{Preciznost} = \frac{TP}{TP + FP} \quad (39)$$

4.5.4. Odziv klasifikacije modela

Odziv (engl. *recall*) je mjera koja indicira koliko je uzoraka pozitivne klase ispravno klasificirano kao pozitivni. Ako je mjera odziva jednaka 1.0 (100%), to znači da model pri predviđanju nije proizveo ni jedan lažno negativni rezultat. Dobiva se dijeljenjem ispravno

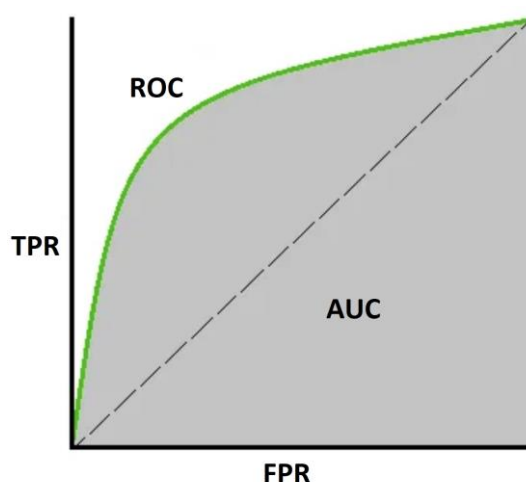
pozitivnih s ukupnim brojem uzoraka koji ispravno pripadaju pozitivnoj klasi, što je prikazano u formuli:

$$\text{Odziv} = \frac{TP}{TP + FN} \quad (40)$$

4.5.5. ROC krivulje

Drugi način grafičkog prikaza uspješnosti klasifikacije pored matrica konfuzije su ROC krivulje. ROC krivulja je prema definiciji mjera, odnosno prikaz uspješnosti klasifikacije kod strojnog učenja za različite postavke granice odluke (engl. *Threshold*). Što je ROC krivulja „viša“ uspješnija je u predviđanju klasa, odnosno preciznije će predvidjeti nule kao nule i jedinice kao jedinice [43].

ROC krivulja je grafički prikaz stope ispravno pozitivnih (Y-os) naspram stope točno negativnih predviđanja (X-os). Na slici 31 prikazan je primjer ROC krivulje.



Slika 31: Primjer ROC krivulje

Izvor: [43]

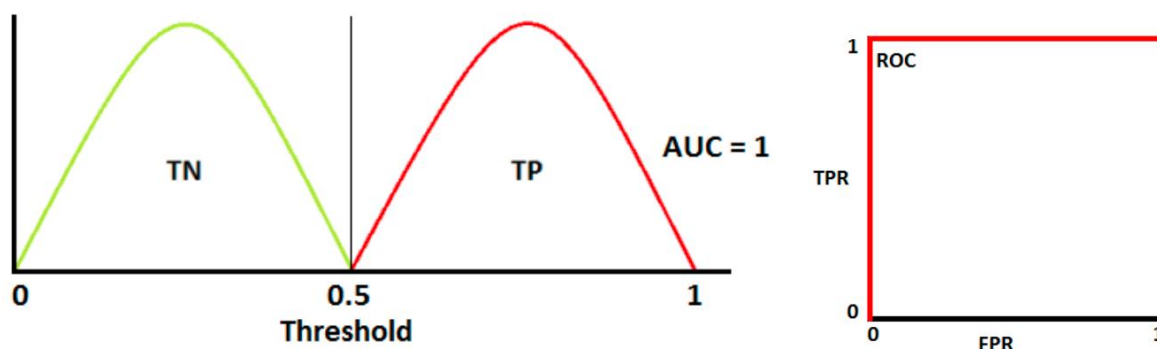
Zelenom bojom je prikazana ROC krivulja, AUC (engl. *Area Under Curve*) je siva površina ispod krivulje. TPR (engl. *True Positive Rate*) je stopa ispravno pozitivnih predviđanja opisana formulom [43]:

$$\text{TPR} = \frac{TP}{TP + FP} \quad (41)$$

Mjera FPR (engl. *False Positive Rate*) ili stopa lažno pozitivnih je opisana formulom [43]:

$$FPR = \frac{FP}{TN + FP} \quad (42)$$

Idealna situacija je kada je krivulja na maksimalnoj visini i širini odnosno ima oblik obrnutog slova L. To znači da nema preklapanja između pozitivne i negativne klase i površina ispod krivulje AUC je 1.0, odnosno 100%. Idealna ROC krivulja je prikazana na slici 32 desno [43].

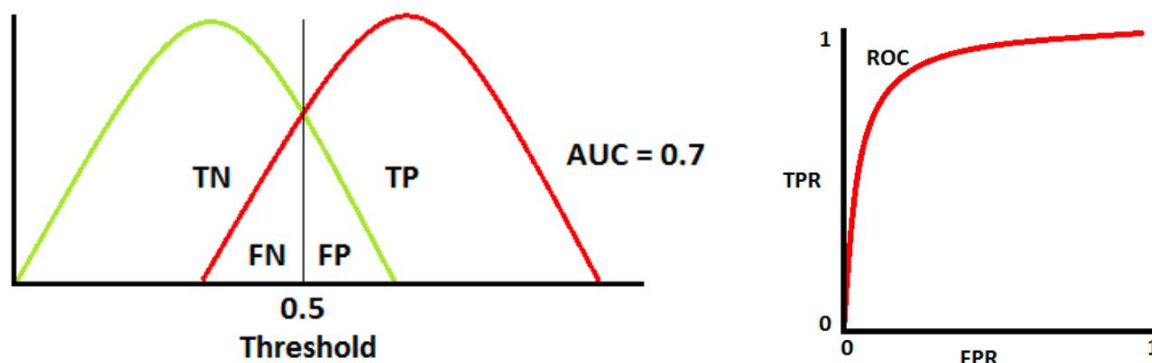


Slika 32: Idealna ROC krivulja

Izvor: [43]

Na slici 32 lijevo prikazane su dvije klase TN i TP koje se nimalo ne preklapaju, to znači da je model u stanju sa 100 postotnom točnošću predvidjeti klase, odnosno predviđeni i stvarni podaci su jednaki [43].

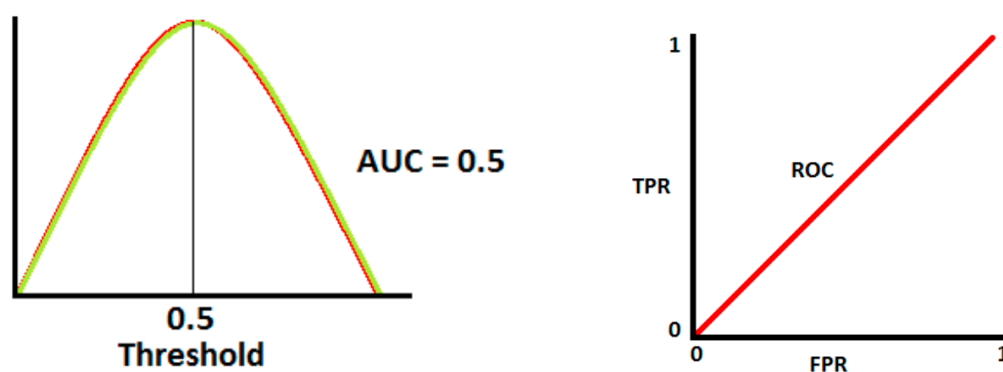
Kada se pozitivna (TP) i negativna klasa (TN) preklapaju, što je najčešći slučaj u stvarnosti, dolazi do grešaka u predviđanju, odnosno određene jedinice su predviđene kao nule i obrnuto. Kad je AUC jednak 0.7 to znači da je model u stanju sa 70 % vjerojatnošću predvidjeti jedinice (pozitivnu klasu), odnosno nule (negativnu klasu). Taj slučaj prikazan je na slici 33 [43].



Slika 33: ROC krivulja kod slučaja $AUC=0.7$

Izvor: [43]

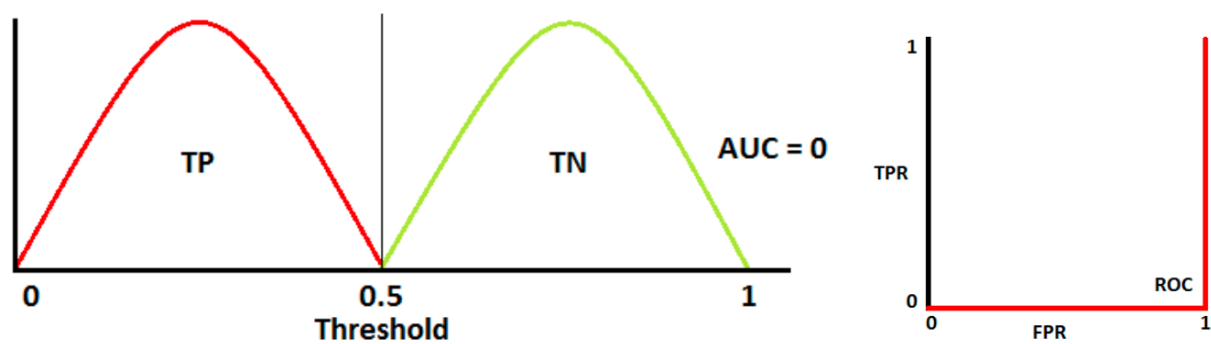
Najgori mogući slučaj je kada dolazi do potpunog preklapanja pozitivne i negativne klase. Takva ROC krivulja naznačuje da je model klasifikacije nije u stanju razaznati između pozitivne i negativne klase, te u tom slučaju pogađa izlaznu vrijednost. To znači da je AUC jednak 0.5, taj je slučaj prikazan na slici 34.



Slika 34: ROC krivulja nerazpoznavanja klasa ($AUC=0.5$)

Izvor: [43]

U slučaju kada je $AUC=0$, tada dolazi do komplementarnog predviđanja klasa, odnosno negativna klasa je klasificirana kao pozitivna i obrnuto [43]. Ovaj slučaj je prikazan na slici 35.



Slika 35: Prikaz ROC krivulje sa $AUC=0$

Izvor: [43]

5. PREDVIĐANJE SMJERA KRETANJA CIJENE BITCOIN-A UPOTREBOM PYTHON PROGRAMSKOG JEZIKA

Unutar ovog poglavlja opisan je praktični dio Diplomskog rada. Sav programski kod napisan je izveden upotrebom Python programskog jezika, unutar *Jupyter Notebook* IDE-a (engl. *Integrated Development Environment*) koji je dio *anaconda3* programskog paketa. Rješenje zadanog problema djelomično je izvedeno korištenjem određenih biblioteka koje su podržane unutar *Python* programskog jezika.

5.1. STRUGANJE TEKSTA I MANIPULACIJA PODACIMA

5.1.1. Opis Bitcointalk foruma

Razlog odabira ovog foruma kao izvora podataka za analizu sentimenta teksta je jer je iz njega omogućeno struganje povijesti podataka koji su stari nekoliko godina. Iako bi najidealniji izvori za rješavanje ove problematike bile popularne društvene mreže poput: *Twitter-a*, *Facebook-a*, te poznatijih foruma poput *Reddit-a*. Isti nažalost sadrže politiku koja limitira preuzimanje njihovih podataka. Ovaj forum nije popularan poput navedenih društvenih mreža, ali daje dovoljno dobre povratne informacije od pojedinaca koji imaju određeni interes prema ovoj kriptovaluti.








Korisnik ovog foruma može biti bilo tko zainteresiran za *Bitcoin*, ostale kriptovalute bazirane na tehnologiji ulančanih blokova, određene tehnologije te investiranje. Neki od njih su stručni programeri te vlasnici tvrtki koji najčešće posluju i komuniciraju pod pseudonimima. Tu su i osoblje te moderatori koji održavaju forum urednim, moderiraju razgovore i nude tehničku podršku za pitanja koja postavljaju korisnici. Zahvaljujući ogromnoj bazi korisnika (više od 2 milijuna članova zajednice), *Bitcointalk* je postao vrijedan izvor prometa za sve što je povezano s *Bitcoin* kriptovalutom. Valja napomenuti da je kreator ovog foruma Satoshi Nakamoto koji je ujedno i kreator *Bitcoin-a* [44].

Na forumu url-a: <https://Bitcointalk.org/> može se pronaći više odjeljka unutar *Bitcoin* prozora, to *Bitcoin Discussion, Development & Tehnical Discussion, Mining, Bitcoin Tehnical Support, Project Development*.

Za problematiku ovog rada odabran je odjeljak *Bitcoin Discussion* čiji opis preveden iz engleskog jezika glasi: „*Opća rasprava o Bitcoin ekosustavu koja ne pristaje bolje drugdje. Vijesti, Bitcoin zajednica, inovacije, opće okruženje itd. Rasprava o određenim uslugama povezanim s Bitcoinom obično pripada drugim odjeljcima* „. Prema opisu u ovaj odjeljak se objavljuju teme opće teme rasprave o *Bitcoin-u*, što najviše odgovara tekstu koji bi bio pogodan za analizu sentimenta, te uz to spada u najpopularniji odjeljak ovog web foruma.

Otvaranjem *Bitcoin Discussion* odjeljka na forumu ispod ploče *Child Boards* prikazuju se stranice (*Pages*) pomoću kojih je omogućeno listanje prethodno komentiranih tema. Ispod stranica nalazi se jezgra foruma prikazana tablicom koja sadrži stupce po nazivima: *Subject, Started by, Replies, Views* te *Last post*. Za problematiku analize sentimenta zanimljiv je stupac *subject* koji prikazuje naziv teme objavljene teme te stupac *Last post* u kojem je naveden datum posljednje objave za zadanu temu, što je vidljivo na slici 36.

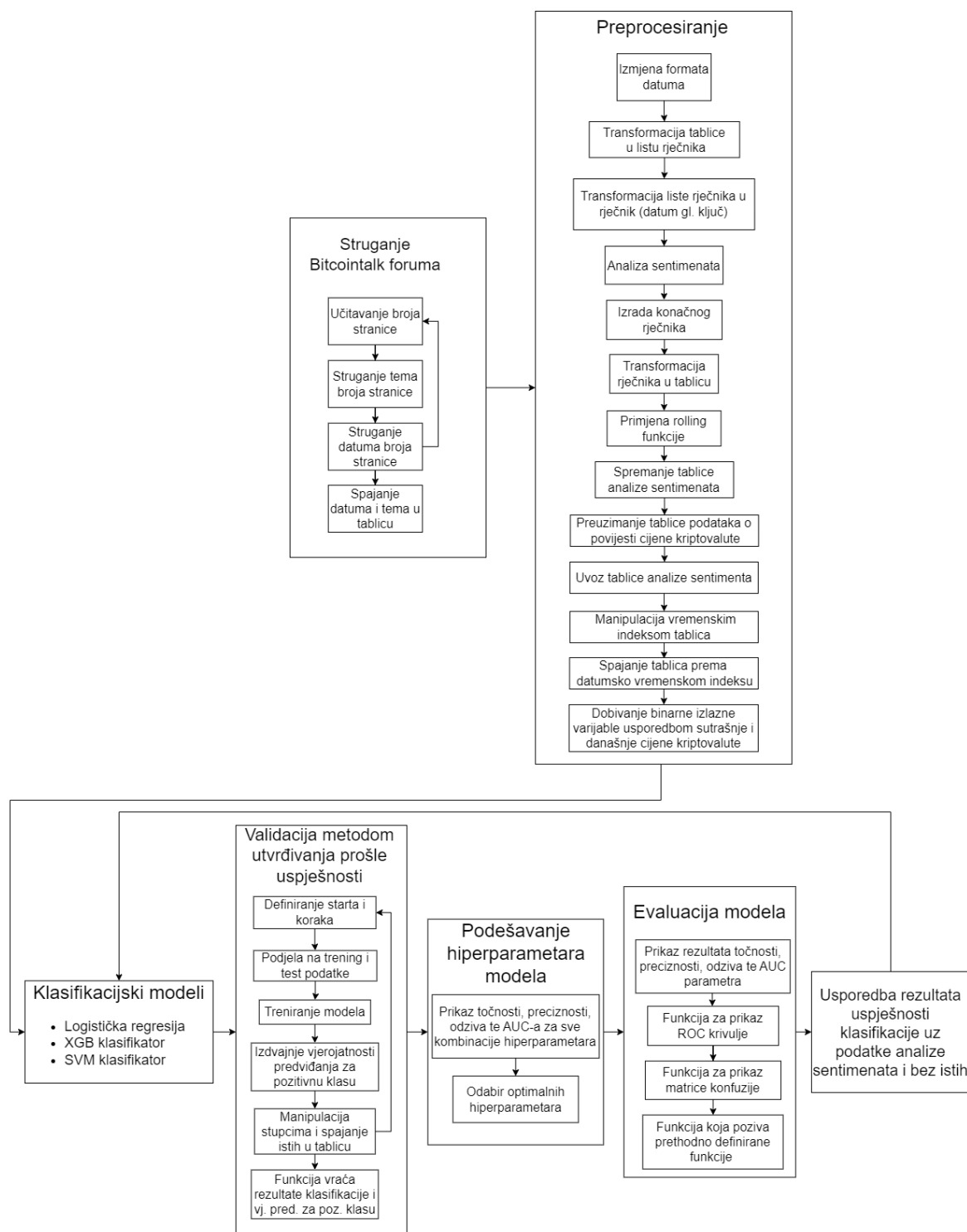
Pages: [1] 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 ... 1473

	Subject	Started by	Replies	Views	Last post
	Bitcoin Core 24.0.1 Released « 1 2 All »	achow101	30	2445	Today at 05:14:16 PM by Mr.Member
	Forum moderation policy « 1 2 ... 43 44 »	sirius	878	875937	January 19, 2023, 06:30:40 PM by Welsh
	Low quality topics do not belong here.	hilariousandco	0	69253	October 15, 2016, 10:51:58 AM by hilariousandco
	I still believe I am on the right path « 1 2 All »	Kalchef	25	117	Today at 10:42:33 PM by Adbitco
	Public Perspective towards Bitcoin	TitanGEL	8	43	Today at 10:33:55 PM by serjent05
	Thanks to bitcoin « 1 2 3 4 All »	Y3shot	72	420	Today at 10:30:50 PM by Myleschetty
	What other ways are available for purchasing	KiaKia	68	380	Today at 10:29:09 PM by livingfree

Slika 36: Odjeljak Bicoiin Discussion unutar *Bitcointalk* foruma

Izvor: *Bitcoin* Forum, online: <https://Bitcointalk.org/index.php?board=1.0> (28.3.2023.)

Programski kod koji je prikazan u prilogima može se prikazati blok dijagramom na slici 37.



Slika 37: Blok dijagram programskog koda

Izvor: Pripremio student

5.1.2. Struganje podataka i pročišćavanje teksta

Prilog 1 sadrži programski kod cjelokupnog postupka struganja teksta iz odabranog foruma. Prvih nekoliko linija koda tog dijela sadrži naredbe za uvoz potrebnih biblioteka te određenih klasa iz tih biblioteka.

Korištenjem biblioteke *requests*, omogućen je dohvat sadržaja s danog *url-a*, a *Python* klasa *BeautifulSoup* iz *bs4* biblioteke pomaže da se taj sadržaj analizira te iz njega dohvate detalji (tekst) od značaja. *BeautifulSoup* se može koristiti za dohvat podataka koristeći *html tag*, *class*, *id*, *css* te ostale selektore [45].

Pandas biblioteka je najpoznatija po svojim mogućnostima za manipulaciju podataka u tablicama. Biblioteka *datetime* omogućuje manipulaciju i konverziju podataka vremenskog zapisa odnosno datuma i vremena. Biblioteka *re* je zadužena za traženje određenih tekstualnih podataka pod zadanom *regex* naredbom.

Prije definiranja *for* petlje koja će sakupljati zadane podatke potrebno je definirati prazne liste gdje će se pojedini podaci spremati. Lista *subject* služi kao spremnik teksta iz svake teme koja se nalazi na pojedinoj stranici unutar foruma, dok lista *dat* služi za spremanje datuma posljednje objave vezane za tu temu.

Budući da na svakoj sljedećoj stranici *Bitcoin Discussion* odjeljka foruma broj na kraju *url* oznake (nakon *...?board1.*) stoji broj koji se povećava za 40, prema tome se definira *for* petlja koja će prelistavati stranice unutar foruma, počevši od 0 do 25000 (do 625 stranice) za korak 40. Unutar varijable *page* je najprije pomoću funkcije *get* dohvaćen sav sadržaj *url-a* pa je potom atributom *text* taj sadržaj prikazan u obliku teksta. Unutar objekta *soup* unutar *BeautifulSoup* klase kao prvi parametar ubačena je varijabla *page* te potom oznaka *'html.parser'* koja omogućuje razvrstavanje i kronološku konstrukciju *html* programskog koda.

Kako je postupak dohvaćanja *html* koda web stranice završen i spremljen u objekt *soup* iz njega je moguće izvući određene oznake. Desnim klikom, na određenu temu *Bitcoin Discussion* odjeljka te odabirom na *Inspect Element Google Chrome* preglednika, prikazuje se *html* kod te stranice te točan dio koda u kojem se nalazi tekst te teme, kako je i prikazano na slici 38.

[illegible]

Slika 38: Html kod unutar Bitcoin Discussion odjeljika foruma

Izvor: *Bitcoin* Forum, online: <https://Bitcointalk.org/bord.php?board=1.0> (30.3.2023.)

Dio *html* koda u kojem se nalazi tekst označen je plavom bojom. Vidljivo je da tekst pripada *windowbg html* klasi, pa se pomoću metode *find_all* nad objektom *soup* pronalaze se svi elementi te *html* klase te je to spremljeno u varijablu *post*. Isti postupak vrijedi i za pronalazak datuma posljednje objave, samo se u tom slučaju traži *html* klasa *windowbg2 lastpostcol* te je to spremljeno u varijablu *date_element*.

Metoda *find_all* sve elemente pronađene klase sprema u listu. Jedini način prolaska kroz listu je korištenjem *for* petlje, pa se za svaki element *liste (date)* traže potomci (engl.

descendants) tog elementa. Atribut *descendants* razdjeljuje element na listu elemenata pa je potrebno unutar istoimene varijable *date* to definirati (*list(date.descendants)*). Datum je u nekim slučajevima spremljen u dva elementa te liste, točnije u 5. i 6. element, pa je potrebno dva elementa spojiti u jedan i sve to pretvoriti u *string* (*str* format) odnosno znakovni format.

Varijablu *date* se potom obradi unutar *if-else* uvjeta. Unutar *if* uvjeta je definirano da ako je varijabla *date* veća od 20 znakova (što znači da sadrži puni datum), tu varijablu je potrebno skratiti da počne od 20. znaka (*date[20:]*), kako bi se filtrirali početni nepotrebni znakovi te počelo od oznake datuma. Varijabla *date* uz datum sadrži i vrijeme koje je nepotrebno, pa se zaseban datum izvlači pomoću biblioteke *re* funkcijom *search* u koju se pod prvi argument ubacuje *regex* naredba, a u drugi varijabla u kojoj se traži dio teksta odnosno *date*. *Regex* funkcija *search* za izlaz daje više elemenata koji daju dodatne informacije o traženju teksta, potreban je samo prvi element koji je rezultat traženja teksta (*group(0)* funkcija) te je traženi datum sljedećeg oblika: npr. *February 22, 2023*. Pošto taj oblik sadrži zarez (,) nakon oznake dana (22), potrebo je istog ukloniti što se izvršava upotrebom funkcije *sub* biblioteke *re*. Za prvi argument funkcije *sub* unosi se *regex* oznaka za zarez (\ ,), u drugi zamjena za taj simbol (prazan znak) te u treći varijabla u kojoj se to izvršava (*date*). Unutar *else* dijela (ako *date* ima manje ili jednako 20 znakova), što označuje da je u *date* spremljena oznaka današnjeg datuma (*//n...Today*), u istu varijablu će se bibliotekom *datetime*, točnije *datetime.now()* metodom koja dohvaća današnji datum pomoću *strftime* metode, definirati zadani format tog datuma (*"%B %d %Y"*), tako da je identičnog oblika kao i ostali datumi (npr. *February 22 2023*). Nakon navedenog postupaka filtriranja teksta dobivenu konačnu varijablu *date* dodaje se u listu *dat* naredbom *append(date)*.

Za pronalazak teksta postavljenih tema potreban je malo kraći postupak. Unutar *for* petlje koja prelistava svaki post unutar pojedine stranice odjeljka foruma, u *text* varijablu metodom *find* pronalazi se prvi element, unutar pojedinog elementa *posts* varijable (*post*) , *'a'* što se može vidjeti na slici 38 (tekst je spremljen unutar *<a>*). Pošto element *'a'* uz tekst objave sadrži i *url* (*href=...*) za pristup istoj, to se može filtrirati atributom *string* (*text.string*).

Međutim, u tom dijelu se nailazi na pogrešku pri izvođenju programskog koda, budući da su prve tri objave prve stranice odjeljka pravila prikvačena na vrh tablice objava objavljenih od strane moderatora foruma, iste će pri izvršenju metode *find* dati izlaz *None* (prazna/nepostojeća vrijednost). Taj problem može se riješiti *try-except* uvjetima za pronalazak i uklanjanje grašaka programskog koda. U *try* dio se unosi programski kod koji se želi izvršiti,

dok se u *except* unosi naredba ili programski kod koji će se izvršiti u slučaju pojave greške pri izvođenju programa *try* dijela. Na kraju te *for* petlje pojedina *text* varijabla se dodaje unutar *subject* liste.

Dobivene liste nakon svih prolaska *for* petlje je potrebno pretvoriti u stupce metodom *Series* iz *pandas* biblioteke. Kako bi objave bile usklađene s njihovim datumom potrebno je ukloniti prvi element, iz određenog razloga postoje 4 *None* vrijednosti na početku *subject* liste, od kojih prva nema datum, pa je i ona višak. Nakon pretvorbe u stupce, isti se mogu spojiti funkcijom *concat*, koja omogućuje dodjelu naziva pojedinom stupcu, te je potrebno definirati *axis=1* parametar kako bi metoda raspoznala da se radi o spajanju stupaca. Kako bi se izradila konačna tablica potrebno je još izdvojiti prva tri retka tablice *df* naredbom *iloc[3:]*. Prikaz prvih 5 redaka tablice spremljene u *df* varijablu, prikazan je na slici 39.

	Date	subject
3	April 14 2023	Federal government and banks are threats...
4	April 14 2023	What does it take to become a Bitcoin investor
5	April 14 2023	Bitcoin is racist
6	April 14 2023	Warren Buffett Blasts Bitcoin as 'Gambling Tok...
7	April 14 2023	What is the situation of Bitcoin today?

Slika 39: Dio tablice izrađene struganjem teksta iz *Bitcointalk* foruma

Izvor: Izradio student upotrebom IDE-a Jupyter Notebook

U *Prilogu 2* prikazan je kod namijenjen za transformaciju datuma u potpuno brojevi format odvojen crticama upotrebom biblioteke *time*. Prvi prozor sadrži *for* petlju u rasponu od 3 do veličine stupca *Date*, čemu je dodano 3. To je napravljeno iz razloga jer *df* tablica počinje od 4. elementa (brojevne oznake redaka počinju od nule), odnosno prvi indeks tablice je 3. Namjena ove *for* petlje je da svaki red stupaca *date* (datum) pretvori iz jednog formata u drugi. To je izvedeno pomoću biblioteke *time* te njezine funkcije *strptime* u kojoj je potrebno najprije pretvoriti zadani format u *time.struct_time tuple* zapis. To je datum zapisan unutar *Phyton tuple-a*, u kojem je svaki element dio vremenskog zapisa. Iz navedenog formata može se uspješno pretvoriti u željeni format koji je sljedećeg oblika npr. *2017-10-1*. Kako bi se efikasno poništili brojevni indeksi koji počinju od broja 3, postavlja se kao glavni indeks stupac *Date* (*set_index('Date')*) te potom tablica *df* okreće naopako tako da počinje od najstarijeg datuma (*df.iloc[::-1]*). Tablicu je nužno pretvoriti u listu naredbom *to_dict('records')*, što se sprema u *df_list* varijablu, kojoj je pojedini element zasebni rječnik (engl. *dictionary*), koji ima 2 ključa

(engl. *keys*) prema prethodnoj tablici to su *Date* i *subject*. Svaki ključ ima dodijeljenu vrijednost prema redovima u *df* tablici kako je i prikazano na slici 40.

```
[{'Date': '2023-03-25',  
  'subject': 'How we lived when we had no choice until Bitcoin came'},  
 {'Date': '2023-03-25',  
  'subject': 'Advantage of holding Bitcoin in time of crisis '},  
 {'Date': '2023-03-25', 'subject': 'Quoting sales prices in Bitcoin'},  
 {'Date': '2023-03-25', 'subject': 'How Bitcoin changed my life '},  
 {'Date': '2023-03-26',  
  'subject': 'Is this making any sense or a misunderstanding '},
```

Slika 40: Prikaz dijela sadržaja *df_list* varijable

Izvor: Izradio student upotrebom Jupyter Notebook IDE-a

Prilog 3 sadrži programski kod zaslužen za svrstavanje teksta pod datum kojem pripadaju. To je izvršeno na način da je zadan datum kao glavni ključ rječnika, a njegova vrijednost je unutrašnji (ugniježdjeni) rječnik koji ima ključ *teme_list*, vrijednost ugniježdenog rječnika je lista teksta tema kojim se posljednja objava dogodila na taj dan. Programski je to izvedeno na način da se najprije kreira prazni rječnik naziva *teme*. Kako bi se u *teme* spremali podaci po određenom datumu nužno je kreirati *for* petlju koja će prolaziti kroz *df_list* listu. Budući da svaki rječnik u *df_list* listi sadrži dva ključa, prvi ključ *date* se sprema u varijablu *datum* kodom *datum=dic["Date"]*. Taj datum potrebno je inicijalizirati unutar *teme* rječnika pomoću *if* uvjeta, čija je namjena provjera nalazi li se taj datum unutar rječnika. Ako datum unutar rječnika *teme* ne postoji, isti se dodaje kao glavni ključ koji za svoju vrijednost ima unutrašnji rječnik. Nakon provjere i/ili izvršenja *if* uvjeta dodaje se tekst teme listi koja je vrijednost unutrašnjeg rječnika s ključem *teme_list*, što je prikazano u posljednje dvije linije *Priloga 3*. Sadržaj rječnika prikazan je na slici 41.

```
{'2023-03-25': {'teme_list': ['How we lived when we had no choice until Bitcoin came',
'Advantage of holding Bitcoin in time of crisis ',
'Quoting sales prices in Bitcoin',
'How Bitcoin changed my life ']}},
'2023-03-26': {'teme_list': ['Is this making any sense or a misunderstanding ',
'Bitcoin can do better job opportunities if empower by government',
'Bitcoin market changes over the past 15 years.',
'$1M Bitcoin Valuation until the end of this Bull Run',
'[CHARITY] Improving lives through Bitcoin - Bitcoin gathering in church ',
'Stay humble and go your own way (invest in yourself) - the best strategy of 2022',
'how much money is considered as "invest only what you can afford to lose"',
'Who has/had the oldest mined Bitcoin?',
'How do you feel selling your BTC against your plans?'],
'Bitcoin as a Mainstream payment method'}},
'2023-03-27': {'teme_list': ['Legal side to promoting bitcoin.'],
```

Slika 41: Prikaz dijela sadržaja rječnik-a teme

Izvor: Izradio student upotrebom Jupyter Notebook IDE-a

5.1. ANALIZA SENTIMENATA TEKSTA IZ BITCOINTALK FORUMA

U *Prilogu 4* je prikazan dio programskog koda koji je zadužen za izvršenje analize sentimenata teksta. Najprije je potrebno uvesti potrebnu funkciju naziva *pipeline* iz biblioteke *transformers*, zatim se unutar parametra te funkcije definira koji će se predtrenirani *deep learning* model koristiti za rješavanje zadanog problema ("*sentiment-analysis*").

Kad se u funkciju tog modela ubaci određeni tekst, tom tekstu će se dodijeliti određeni izlazni rezultat i polaritet. To je pri izvršenju prikazano kao lista s rječnikom koji je jedini element te liste. Taj rječnik sadrži dva ključa, a to su: *label* (polaritet) i *score* (rezultat) unesenog teksta. Polaritet može imati dvije moguće vrijednosti: *POSITIVE* ili *NEGATIVE*, dok se *score* vrijednost kreće od 0 do 1. Kako bi se to pojednostavilo napravljena je funkcija koja će iz te liste rječnika izvući jedan numerički rezultat. U *Prilogu 4* je prikazano definiranje funkcije *analiza_sent* u koju se kao argument može ubaciti određeni tekst. Taj tekst se unutar te funkcije ubacuje u zadani model za analizu teksta, analiza teksta staje na 100 simbola kako bi se ubrzalo izvršenje funkcije te je uz to definirano da se izvlači prvi element te liste (jer samo prvi i postoji). Unutar *if* uvjeta u tijelu funkcije zadano je da kao je polaritet negativan rezultat će se pomnožiti s brojem -1. Na taj način se osigurava da ako je polaritet pozitivan na izlazu funkcije vraća se samo vrijednost ključa *score*, u suprotnom će ta vrijednost biti negativna, usporedba ubačene riječi „*hate*“ bez upotrebe *analiza_sent* funkcije i s upotrebom iste prikazana je na slici 42.

```
sent_pipeline("hate")  
[{'label': 'NEGATIVE', 'score': 0.9996899366378784}]  
  
analiza_sent('hate')  
-0.9996899366378784
```

Slika 42: Usporedba izlaza pri ubacivanju negativnog teksta sa i bez upotrebe funkcije `analiza_sent`

Izvor: Izradio student upotrebom Jupyter Notebook IDE-a

Kako bi se konačno ubacio izvučen tekst iz foruma u zadani model za analizu sentimenta, potrebno je definirati prazni rječnik (*rez_transf*) što je i prikazano u *Prilogu 5*. Tekst iz prijašnjeg rječnika *teme* izvlači se prelistavnjem korištenjem *for* petlje kroz svaki ključ unutar istog. Naredbom *teme[dat]["teme_list"]* dohvaća se vrijednost ključa *teme_list* koja se sprema u varijablu *lista_tema*. Rječniku *rez_transf* se dodjeljuje se glavni ključ, što je datum iz prijašnjeg rječnika, a kao njegova vrijednost inicijaliziran je rječnik s dva ključa: *broj_tema* te *rezultati*, što je izvršeno kodom *rez_transf[dat]=dict(broj_tema=len(lista_tema), rezultati=list())*. Dnevni broj tema se pronalazi prebrojavanjem ukupnog broja elemenata varijable *lista_tema* (funkcija *len*), dok je za vrijednost ključa *rezultati* potrebno definirati praznu listu, navedeno je izvršeno kodom *rez_transf[dat]=dict(broj_tema=len(lista_tema), rezultati=list())*. Za dodavanje rezultata analize teksta u listu koja je vrijednost ključa *rezultati*, unutrašnjeg rječnika, potrebno je prelistati kroz sve elemente varijable *lista_tema*. Konačno, lista koja je vrijednost ključa *rezultati*, nadopunjuje se rezultatima koje daje funkcija *analiza_sent* pri ubacivanju teksta *tema* u istu kao argument. Dio sadržaja *rez_transf* rječnika prikazan je na slici 43.

```
{'2023-03-25': {'broj_tema': 4,
  'rezultati': [-0.9843512773513794,
    0.963616669178009,
    -0.9519705176353455,
    0.9498694539070129]}},
{'2023-03-26': {'broj_tema': 10,
  'rezultati': [-0.9995962977409363,
    -0.9970706701278687,
    -0.9489948749542236,
    -0.9970603585243225,
    0.9077235460281372,
    0.9997840523719788,
    -0.9934604167938232,
    -0.9934388995170593,
    -0.9994494318962097,
    -0.9549924731254578]}},
```

Slika 43: Dio sadržaja rez_transf rječnik-a

Izvor: Izradio student upotrebom IDE-a Jupyter Notebook

U drugom prozoru programskog koda u *Prilogu 5* definirane su funkcije koje za argument primaju listu iz koje izdvajaju pozitivne ili negativne brojeve, ovisno o kojoj se funkciji radi. Krajnji rječnik prije pretvorbe u tablicu, naziva *end_transf*, inicijaliziran je u trećem prozoru. Rječnik *end_transf* služi za spremanje broja dnevnih tema (ključ *broj_tema*), prosjeka liste rezultata analize sentimenata (ključ *avg_rezultata*) te postotaka pozitivnih i negativnih tema (ključ *posto_poz* i ključ *posto_neg*) na dnevnoj bazi. Navedeni ključevi su dio rječnika koji je vrijednost glavnog ključa (*k*) *end_transf* rječnika, čija vrijednost sadrži datum prijašnjeg *end_transf* rječnika. Do željenog ključa koji za vrijednost sadrži datum dolazi se korištenjem *for* petlje, a dohvat vrijednosti ključa sprema se u varijablu *rez*. U novi rječnik spremaju se vrijednost broja dnevnih tema te tri nova ključa (*avg_rezultata*, *posto_poz*, *posto_neg*) čije je vrijednosti potrebno definirati kao decimalne brojeve (*float* vrsta podataka), koje će se dobiti nadolazećim formulama. Vrijednost ključa *avg_rezultata* opisuje formula $\text{sum}(\text{rez})/\text{len}(\text{rez})$, koja zbraja sve vrijednosti unutar liste *rezultati* te potom taj broj dijeli s ukupnim brojem elemenata liste. Vrijednosti ključa *posto_poz* dobivaju se prebrojavanjem svih pozitivnih rezultanata koji su se izvukli upotrebom funkcije *pozitivni*, što se potom dijeli s ukupnim brojem elementa liste *rezultati*. Vrijednosti ključa *posto_neg* se dobiva na sličan način, razlika je u tome što se za izdvajanje negativnih rezultata koristi funkcija *negativni*. Prikaz djelomičnog sadržaja *end_transf* rječnika prikazan je na slici 44.

```
{'2023-03-25': {'broj_tema': 4,
'avg_rezultata': -0.00570891797542572,
'posto_poz': 0.5,
'posto_neg': 0.5},
'2023-03-26': {'broj_tema': 10,
'avg_rezultata': -0.5976555824279786,
'posto_poz': 0.2,
'posto_neg': 0.8},
'2023-03-27': {'broj_tema': 26,
'avg_rezultata': -0.42927881387563854,
'posto_poz': 0.2692307692307692,
'posto_neg': 0.7307692307692307},
'2023-03-28': {'broj_tema': 15,
'avg_rezultata': -0.728974453608195,
'posto_poz': 0.1333333333333333,
'posto_neg': 0.8666666666666667},
'2023-03-29': {'broj_tema': 13,
'avg_rezultata': -0.5201487724597638,
'posto_poz': 0.23076923076923078,
'posto_neg': 0.7692307692307693},
```

Slika 44: Dio sadržaja `end_transf` rječnik-a

Izvor: Izradio student upotrebom Jupyter Notebook IDE-a

Nakon analize sentimenta i izrade svih potrebnih izračuna, podaci iz posljednjeg rječnika `end_transf` mogu se prebaciti u tablicu što je prikazano prvim prozorom *Priloga 6*. Transformacija iz rječnika u tablicu izvršava se metodom `pandas` biblioteke `from_dict` u koju je potrebno definirati argumente, od koji je prvi rječnik koji se želi transformirati te definirati argument parametra `orient` koji je `"index"`. Kada je `orient="index"`, znači da će svaki glavni ključ rječnika (datum) biti zaseban red u tablici te biti definiran kao njezin indeks (engl. *index*). Kako bi nadolazeći rezultati predviđanja bili svakog puta replikabilni, nužno je definirati krajnji datum do kojeg će se vršiti klasifikacija, u ovom slučaju to je *23.1.2023.*. Uz krajnji datum definiran je početni datum od kojeg će se podaci analize teksta uzeti u obzir, to je *9.10.2017.*, kako je i definirano kodom u trećem prozoru *Priloga 6*. Iako su cijene *Bitcoin* valute dostupne još od 2009. godine, podaci *Bitcointalk* foruma su od tada vrlo diskontinuirani, a čak i u slučaju da nisu, prikupljanje i analiza tako velike baze podataka uz dugotrajnost, načinilo bi samo po sebi diskontinuitet nad podacima jer bi se događale sve veće pogreške kod struganja podataka zadanim bibliotekama. Navedene granice početnog i krajnjeg datuma spremljene su unutar varijable `datumi`. Varijabla `datumi` poslužit će za promjenu postojećeg indeksa odnosno proces reindexiranja koji će postojeće datume (indekse) zamijeniti neprekinutim datumima. Taj problem rješava funkcija `reindex` (4. prozor) koja za prvi argument prihvaća varijablu u koju su spremljeni novi datumi te u slučaju da postoji praznina ili diskontinuitet taj datum će biti ispunjen nulom (`fill_value=0`) za sve vrijednosti stupaca tog retka u tablici. Prije procesa reindexiranja potrebno je pretvoriti postojeće indekse koji su datumi u datumsko-vremenske

indekse ugrađenom funkcijom biblioteke *pandas* naziva *to_datetime*, pri čemu se za argument zadaje indeks postojeće tablice, što je prikazano u drugom prozoru ili tekstnom okviru *Priloga 6*.

Diskontinuitet podataka može se provjeriti pomoću *for* petlje koja će u *df_reindex* tablici provjeravati da li je vrijednost stupaca *broj_tema* jednaka 0, ako je u tom stupcu vrijednost za određeni datum 0, znači da za taj dan nije bilo objava na određenu temu te da su i svi ostali stupci tog datuma jednaki 0.

Zbog postojećeg diskontinuiteta dolazi do upotrebe funkcije *rolling* koja uz pomoć naredbe *mean* iz svih stupaca tablice, prikuplja podatke od *posljednih 7* dana te u 7. red bilježi njihovu srednju vrijednost, za vrijednost 8. reda uzimaju se podaci od 2. do 8. reda te se u 8. red sprema rezultat, taj postupak ponavlja do kraja tablice. Prvih 7 redova (datuma) tablice biti će prazno jer ne postoje prethodni podaci od kojih bi se mogla računati srednja vrijednost, već će oni biti iskorišteni za računanje osmog reda. Pošto je prvih 7 redova prazno, popunjeni su *NaN* vrijednosti potrebno ih je ukloniti pomoću *dropna* naredbe.

Ovime procedura struganja podataka i analize sentimenta završava te se dobiveni podaci mogu spremati iz tablice u *csv* datoteku funkcijom *to_csv* te definiranjem proizvoljnog naziva datoteke s nastavkom *csv*, koja će se kasnije iskoristiti kao dio ulaznog seta podataka za problem predviđanja izlazne varijable.

5.2. PREUZIMANJE POVIJESTI CIJENA KRIPTOVALUTE I DOBIVANJE IZLAZNE VARIJABLE

Unutar ulaznog seta podatka osim podataka analize sentimenta koriste se i podaci povijesti vrijednosti *Bitcoin* kriptovalute u usporedbi s američkim dolarom (*BTC-USD*). U prvom prozoru *Priloga 7*, ta povijest preuzeta je upotrebom biblioteke *yfinance* (*yf*), koja omogućuje preuzimanje podataka o kretanju cijena dionica kao i kriptovaluta iz *Yahoo Finance* web stranice. Tablica podataka o cijenama *Bitcoin-a* preuzeta je funkcijom *download* iz *yf* biblioteke, kojoj se definiraju argumenti kriptovalute u usporedbi sa cijenom valute kojom se uspoređuje ("*BTC-USD*") te prema potrebi datum od kojeg će se početi preuzimati podaci. Nakon preuzimanja dobiva se tablica s 7 stupaca, od koji je *Date* indeks stupac. Uočava se da su vrijednosti cijena u stupcima *Close* i *Adj Close* identične tako da se to i provjerava dvostrukim

izjednačavanjem vrijednosti stupaca te spremanjem istog u varijablu *eq* uz provjeru rezultata te varijable naredbom *value_counts*, koja za izlaz daje da su sve vrijednosti *eq* varijable *True*, što potvrđuje pretpostavku jednakosti stupaca. Kada je to dokazano, stupac *Close* se uklanja funkcijom *drop* te se postavlja argument parametara *inplace=True*, što znači da se izmjena trenutno sprema, bez potrebe za spremanjem u istoimenu varijablu. Kako bi se tablica mogla spojiti s tablicom u kojoj su vrijednosti analize sentimenta, definira se funkcija koja pretvara indeks tablice u *datetime* indeks. U slučaju da su u tablicu prikupljene informacije o vremenskoj zoni to se uklanja metodom *tz_localize(None)* što je također definirano unutar funkcije naziva *datetime_tz*, dio sadržaja tablice *btc* varijable prikazan je na slici 45.

	Open	High	Low	Close	Adj Close	Volume
Date						
2017-10-09	4614.520020	4878.709961	4564.250000	4772.020020	4772.020020	1968739968
2017-10-10	4776.209961	4922.169922	4765.100098	4781.990234	4781.990234	1597139968
2017-10-11	4789.250000	4873.729980	4751.629883	4826.479980	4826.479980	1222279936
2017-10-12	4829.580078	5446.910156	4822.000000	5446.910156	5446.910156	2791610112
2017-10-13	5464.160156	5840.299805	5436.850098	5647.209961	5647.209961	3615480064
2017-10-14	5643.529785	5837.700195	5591.640137	5831.790039	5831.790039	1669030016
2017-10-15	5835.959961	5852.479980	5478.609863	5678.189941	5678.189941	1976039936
2017-10-16	5687.569824	5776.229980	5544.209961	5725.589844	5725.589844	2008070016
2017-10-17	5741.580078	5800.350098	5472.720215	5605.509766	5605.509766	1821570048
2017-10-18	5603.819824	5603.819824	5151.439941	5590.689941	5590.689941	2399269888

Slika 45: Tablica i povijesti BTC-USD cijena

Izvor: Izradio student upotrebom IDE-a Jupyter Notebook

Nakon uvoza i manipulacije tablicom *BTC-USD* cijena potrebno je ubaciti tablicu analize sentimenta teksta spremljenu u *csv* formatu funkcijom *read_csv* biblioteke *pandas*. Unutar argumenta funkcije definiran je naziv *csv* datoteke ("*Transf_roll7.csv*") te parametar *indeks_col* čiji je argument *0* čime se definira da se brojevni indeks stupac ne označuje u tablici. Budući da indeks stupac uvezene tablice varijable *transf* nije definiran, istog se definira naredbom *index.name='Date'*, izvršenom nad navedenom tablicom. Tablicu *transf* se ubacuje kao argument *datetime_tz* funkcije kako bi indeks tablice bio identičnog formata kao kod tablice *btc*, koja je ubačena u istu funkciju. Funkcijom *type* provjerava se jesu li tablice jednakog formata, što je nužno jer je spajanje tablica prema indeksu moguće samo u slučaju identičnih odnosno podudarajućih indeksa. Tablice *btc* i *transf* se spajaju funkcijom *merge* u čijim je parametrima potrebno definirati da se indeksi obje tablice spoje u jedan jedinstveni indeks

(*right_index = True, left_index = True*). Spajanje se provodi kodom *btc.merge(transf, right_index=True, left_index=True)*, što znači da će stupci tablice *btc* biti prvi te će se njima pridodati stupci tablice *transf*. Konačna tablica spremljena je u *btc_transf* varijablu.

Trenutna tablica sadrži samo ulazne varijable koje će poslužiti za strojno učenje, kako bi se dobila binarna izlazna varijabla, potrebno je usporediti „sutrašnju“ cijenu s „današnjom“ cijenom *Bitcoin*-a. Kako bi se to izvršilo, potrebno je pomaknuti cijenu zatvaranja (*Adj Close*) za jedan red u nazad naredbom *shift(-1)* te tu vrijednost spremiti u novi stupac naziva *btc_sutra*, što je prikazano u zadnjem prozoru *Priloga 7*. Budući da *btc_sutra* sadrži „sutrašnju“ cijenu *Bitcoin*-a, uspoređuje se jesu li vrijednosti stupca *btc_sutra* veće od vrijednosti *Adj Close* stupca iz istog reda. Kao rezultat usporedbe cijena vraća se *True* ako je uvjet zadovoljen, a u slučaju da nije izlaz je *False*. Kako bi se dobivene vrijednosti iz *bool (True/False)* tipa pretvorile u numeričke, potrebno je koristiti funkciju *astype* te njoj definirati argument *int*. Dobivena izlazna varijabla spremljena je u stupac naziva *izlaz*.

5.3. IZRADA FUNKCIJA ZA VALIDACIJU METODOM UTVRĐIVANJA PROŠLE USPJEŠNOSTI, PRIKAZ ROC KRIVULJA, MATRICA KONFUZIJE TE OSTALIH REZULTATA USPJEŠNOSTI

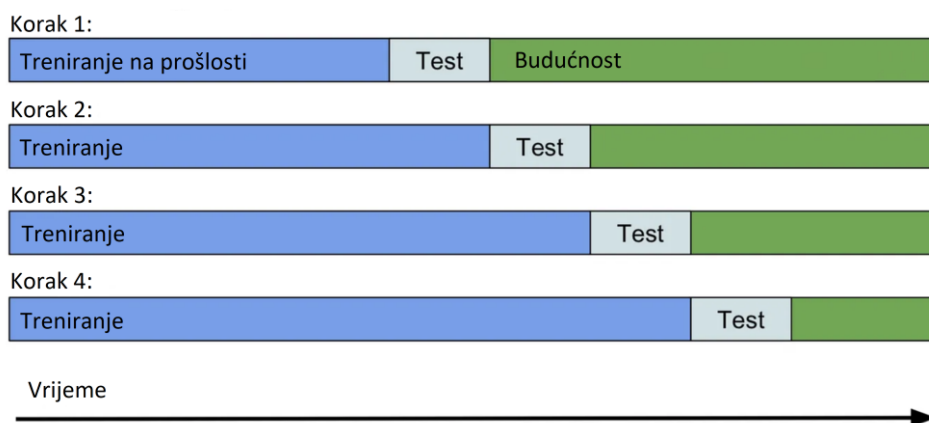
Pouzdan rezultat pri testiranju pojedinog modela strojnog učenja potvrđuje se u većini slučajeva križnom validacijom (engl. *Cross Validation*). Križana validacija podrazumijeva uzimanje određenog omjera nasumično odabranih redova tablice od kojih se veći dio primjerice 80 % uzima za treniranje modela, te ostalih 20% za testiranje uspješnosti modela (mogu se uzeti i drugi omjeri) [46].

Set podataka na kojem se bazira ovaj diplomski rad je vremensko serijskog tipa (engl. *time series data*) te je važno da se pri testiranju daje li model pouzdane rezultate uspješnosti ili validaciji, zadrži originalni redoslijed podataka. U slučaju primjene križne validacije na takvom setu podataka došlo bi do nerealnih scenarija, gdje bi se primjerice koristili podaci iz 2022. za predviđanje podataka iz 2018. godine. Osim navedenog, došlo bi do pojave curenja podataka (engl. *Data Leakage*) iz razloga jer bi model pri testiranju saznao dio podataka iz budućnosti, pa je primjerice veća vjerojatnost da će model točnije pogoditi sutrašnji rezultat ako mu se da u setu podataka za treniranje točan podatak cijene koji će biti za 10 dana. U navedenom slučaju

validacija modela bila bi u potpunosti nevaljana, što može za rezultat pri validaciji na test podacima dati vrlo dobre rezultate, dok bi u stvarnosti rezultati uspješnosti bili značajno lošiji [46].

Najpouzdanija validacija kod vremensko serijskog tipa podatka je takozvana metoda utvrđivanja prošle uspješnosti (engl. *Backtesting*). Kod toga tipa validacije poštuje se redosljed podataka, a pritom se uvijek uzimaju drugi podaci za treniranje i testiranje. To se postiže tako da se pomiče za zadani korak kroz redove tablice podataka od početka do kraja iste. Pri tome veličina podataka za trening raste svakim sljedećim prolazom *for* petlje, dok je veličina test podataka uvijek ista samo se pomiče u skladu s trening podacima [46].

To je u ovom slučaju izvedeno na način da ako je definirano da je korak svake iteracije 30 dana, te početna veličina trening podataka 1095 dana (približno 3 godine). Dakle, u prvom će se koraku uzeti 1095 dana (redova) podataka za treniranje, te od 1096 do 1125 (30 dana, jer se uključuju početna i krajnja granica) dana se vrši testiranje modela. U drugom koraku je veličina trening podataka jednaka 1125 dana, te se od 1126 do 1155 vrši testiranje. Treći korak treniran je na 1155 dana, te testiran od 1156 do 1185 dana, taj postupkom se ponavlja sve do stizanja do posljednjeg reda u tablici. Postupak metode utvrđivanja prošle uspješnosti prikazan je na slici 46.



Slika 46: Postupak validacije metodom utvrđivanja prošle uspješnosti

Izvor: Adaptirao student prema [46]

Implementacija metode utvrđivanja prošle uspješnosti za analizu uspješnosti modela, primjenom rezultata preciznosti, točnosti, odziva te AUC parametra nalazi se unutar funkcije *back_testing*, čiji je programski kod prikazan u *Prilogu 8*. Funkcija *back_testing* prihvata 5 mogućih parametara od kojih *start* i *korak* imaju unaprijed definiranu vrijednost argumenta.

Prvi parametar *df* rezerviran je varijablu tablice na čijim podacima se izvodi strojno učenje, drugi parametar *X* namijenjen je za listu ulaznih varijabli te parametar *model* rezervira mjesto za objekt modela strojnog učenja.

Jezgra *back_testing* funkcije je u dijelu gdje se definira podjela na ulazne i izlazne varijable definirane tablice te je to potrebno vrši po prolazima *for* petlje, kao što je to prethodno objašnjeno. Raspon *for* petlje kreće od *start* parametra (*start=1095*) do posljednjeg reda tablice (*len(df)*), za zadani korak (*korak=30*). Varijable *X_train* i *X_test* odnose se na ulazne varijable, *y_train* i *y_test* varijable predstavljaju izlazne varijable za treniranje i testiranje modela strojnog učenja. Funkcijom *iloc* koja označuje brojevni indeks reda u tablici (1. red tablice ima *iloc* indeks jednak nula, jer brojanje počinje od nule) definira se pomoću sjeckanja (engl. *slicing*) podataka tablice definiranje početnog i krajnjeg reda koji će se koristiti za treniranje ili testiranje. Kod naredbe *iloc[0:i]* na trening podacima, u prvom koraku, unutar 0 označuje početni red, dok je *i=1095*, ali se ne uključuje 1095 (1096. red tablice), već do 1094. Te se onda na test podacima primjenjuje *iloc[i:(i+korak)]* naredba jer je potrebno vršiti testiranja na podacima od *i=1095* (1096. red tablice) do 1124 indeksa tablice (1125. reda).

Treniranje modela vrši se metodom *fit* koja za parametre ima definirana mjesta za ulazne i izlazne trening podatke (*X_train* i *y_train*). Kada su podaci o treniranju modela pohranjeni u memoriji, mogu se izvesti ostale naredbe nad odabranim modelom.

Kako bi se izračunale TPR i FPR vrijednosti koje su potrebne za analizu uspješnosti kroz AUC parametar, potrebno izvršiti predikciju modela upotrebom funkcije *y_probs*, koja vraća vektor (engl. *array*) s dva stupca od kojih je prvi vjerojatnost da će model pri vršenju klasifikacije za rješenje odabrati negativnu klasu ili 0, dok drugi stupac označuje isto samo za pozitivnu klasu ili 1. Za svrhu izračuna FPR i TPR parametara potrebne su samo vjerojatnosti pozitivne klase, pa se prema tome koristi samo drugi stupac (*[:,1]*). Budući da se pri svakom prolasku *for* petlje, dobivaju novi rezultati, kako bi se isti mogli spojiti u jedan cjelokupni stupac potrebno ih je pretvoriti u tablični oblik funkcijom *Series* kojoj je za argumente osim vektorskog reda potrebno definirati i indeks koji se mora podudarati s brojem elemenata vektora, u tom slučaju podudara se indeks ulaznih test podataka. U takvom se obliku vjerojatnosti za odabir pozitivne klase (*y_proba_1*) mogu dodati u predefiniranu listu *y_probs_poz*. Funkcijom *concat*, pri svakom prolasku petlje, spajaju se zasebni elementi liste u jedan cjelokupni stupac.

Funkcija *predict* za argument prima ulazne *test podatke* te kao izlaz vraća vektor predviđenih vrijednosti izlaza za dane ulazne varijable, što je spremljeno u varijablu *y_pred*.

Nad varijablom *y_pred* se izvodi postupak pretvorbe *Series* funkcijom u stupac, kako bi se omogućilo spajanje dva stupaca, funkcijom *concat*, spajanjem podatka izlazne test podatke (*y_test*) te predviđene (*y_pred*), što je spremljeno u varijablu *spojeno_izlaz_pred*. Spojeni stupci varijable *spojeno_izlaz_pred* dodaju se u predefiniranu listu *predikcije*, da bi se svi elementi te liste spajali funkcijom *concat* pri svakom prolasku petlje, što je spremljeno u varijablu *predikcije_df*. Ta varijabla vraća 2 vrijednosti to su: *predikcije_df*, *y_proba1_df*, što znači da će se rezultat izvršene funkcije spremiti u 2 zasebne varijable, pa će pri pozivu funkcije, istu biti potrebno izjednačiti s navedenim brojem varijabli.

Prvi prozor *Priloga 9* sadrži funkciju za analizu uspješnosti predviđanja metodom utvrđivanja prošle uspješnosti. Funkcija *backtest_eval* za argumente parametra prihvata varijablu u koju su spremljeni podaci klasifikacijskog modela te varijablu u koju vraća *back_testing* funkcija koja sadrži vjerojatnosti za predviđanje pozitivne klase (parametar *y_probs*). Funkcija *roc_curve* omogućuje izračun FPR-a i TPR-a pomoću izlaznih test podataka i argumenta *y_probs* parametra, ali za izlaz osim FPR-a i TPR-a daje i treću varijablu (*thresholds*) koja neće biti upotrjebljena.

Funkcija pri pozivu ispisuje četiri mjere uspješnosti modela. Mjere točnosti, preciznosti i odziva dobivaju se upotrebom funkcija *accuracy_score*, *precision_score* te *recall_score* *sklearn* biblioteke iz modula *metrics*. U tim funkcijama potrebno je dati 2 argumenta, to su izlazni test podaci te predviđeni test podaci modela kojeg se evaluira metodom utvrđivanja prošle uspješnosti. Posljednji rezultat uspješnosti koji ispisuje ta funkcija je AUC parametar, upotrebom funkcije *auc* iz iste biblioteke te definiranjem parametara FPR i TPR unutar funkcije. Naredbe *:.2f* i *:.4f* služe za zaokruživanje dobivenog rezultata na dvije ili četiri decimale.

U drugom prozoru *Priloga 9* prikazan je postupak izrade funkcije za prikaz ROC krivulje. Kako bi se plot mogao prikazati potrebno je uvesti biblioteku *matplotlib* te njezinu klasu *pyplot*, uz to kako bi se mogao prikazati plot krivulje neophodno je definirati magičnu funkciju *%matplotlib inline*. Izrađena funkcija *plot_roc_curve* za argumente parametra prima varijablu klasifikacijskog modela uz vjerojatnosti da će se predvidjeti pozitivna klasa (*y_probs*). U tijelu funkcije se poziva *roc_curve* funkcija na isti način kao i u prethodnoj funkciji, s ciljem dobivanja FPR i TPR vrijednosti. U funkciju *plot* se za svrhu prikaza ROC krivulje definiraju parametri FPR, TPR, boja krivulje (*color="red"*) te oznaka unutar legende (*label="ROC"*). ROC krivulju treba usporediti s granicom odluke AUC-a jednakog 0.5 (znači da model nagađa

izlaz), kako bi se dobila predodžba o sposobnosti modela pri razlikovanju klasa. Granica odluke označena je isprekidanim crtama, tamno plave boje te ima koordinate (0,1) za obje osi plot, što je vidljivo iz koda `plt.plot([0,1],[0,1],color="darkblue",linestyle="--", label="granica odluke")`. Uz prikaz krivulje još se i definiraju natpisi za pojedinu os (`xlabel`, `ylabel`), naslov plot (title) te naredbe za prikaz legende i prikaz cjelokupnog plot.

Treći prozor *Priloga 9* sadrži kod za prikaz matrice konfuzije, u tijelu izrađene funkcije `conf_matrix`. Funkcija `confussion_matrix` uvezena iz *sklearn* biblioteke, omogućuje prikaz matrice konfuzije koja je u *array* formatu. Za grafički prikaz matrice konfuzije, uz prethodno uvezenu klasu *pyplot*, potrebno je uvesti i biblioteku *seaborn*. Izrađena funkcija `conf_matrix` za vrijednosti parametara prima izlazne test (`y_test`) i predviđene izlazne test podatke (`y_pred`) kako bi se isti mogli ubaciti u funkciju `confussion_matrix`. Funkcija `heatmap` omogućuje prikaz matrice konfuzije u grafičkom obliku pomoću boja koje predstavljaju određeni raspon vrijednosti, a unutar parametara je potrebno definirati varijablu u kojoj je spremljena matrica konfuzije u *array* formatu (`cm`), zadati argument parametra `annot` na *True* kako bi se prikazale brojeve vrijednosti unutar matrice te je potrebno definirati format tih brojeva u standardni (`fmt="d"`), jer se inače prikazuju u eksponencijalnom obliku. Posljednje, kao i kod prethodne funkcije, definiraju se naslov i natpisi pojedine osi matrice konfuzije.

5.4. PREDVIĐANJE SMJERA KRETANJA CIJENA BITCOIN-A

Za svrhu predviđanja cijena *Bitcoin-a* odnosno binarne klasifikacije upotrijebljena su 3 modela strojnog učenja: logistička regresija, klasifikator ekstremnih pojačanja gradijenata te klasifikator potpornih vektora. Kod svakog modela je posebno testirana uspješnost s upotrebom varijabli koje su rezultat analize sentimenta teksta te bez njih (samo podaci o cijeni *Bitcoin-a*). Kako bi se dobio što bolji rezultat predviđanja, kod obje varijante (sa i bez podataka ulaznih podataka o analizi sentimenta) izveden je postupak podešavanja hiperparametara pojedinog modela. Kod sva tri modela odabrana su dva hiperparametra koji prema danim izvorima daju najveći značaj u rezultatima uspješnosti predviđanja. Podešavanje hiperparametra se može poistovjetiti s podešavanjem frekvencije radio prijamnika, gdje će određena frekvencija dati najmanje šuma pri emitiranju signala određene radio stanice.

5.4.1. Traženje optimalnih hiperparametra logističke regresije

Model logističke regresije se uvozi iz biblioteke *sklearn*, točnije iz njezine klase *linear_model*. Unutar varijable *X_transfl* definira se lista stupaca koji će poslužiti kao ulazne varijable pri predviđanju sa svim ulaznim varijablama, dok lista *X_btc* ne sadrži stupce koji sadrže podatke o analizi sentimenta (drugi prozor *Priloga 11*).

Prema [47], za podešavanje odabrani su hiperparametri: *solver* i *C*. Hiperparametar *solver* označuje vrstu algoritma koji se koristi kod problema optimizacije, zadani algoritam je *lbfgs*. Definiranje inverzne snage regularizacije postiže se *C* hiperparametrom, kao i kod metoda potpunih vektora manje vrijednosti označuju snažniju regularizaciju (zadana vrijednost $C=1.0$). Ovisno koja kombinacija argumenta u pojedinom hiperparametru daje najbolje rezultate uspješnosti kroz mjere: točnosti, preciznosti, odziva te AUC parametra, bazirano na toj postavci će se prikazati svi rezultati uspješnosti, ROC krivulja te matrica konfuzije. Prikaz navedenih stavki koje su suma funkcija prethodnih funkcija za prikaz uspješnosti klasifikacije, izvedeno je pomoću funkcije *eval_ROC_cm* koja u svom tijelu sadrži poziv prethodne 4 izrađene funkcije, točnije funkcija *back_testing*, *backtest_eval*, *plot_roc_curve* i *conf_matrix*, kako je i prikazano u *Prilogu 10*.

Osim tih hiperparametra podesti će se i *random_state* parametar (postavljen je na 1, a zadano je *None*), koji omogućuje dobivanje replikabilnih rezultata jer fiksira generator nasumičnih brojeva. Kako bi se ubrzala brzina izvođenja programskog koda modela valja podesiti *n_jobs* parametar na -1 (zadano je 1), što znači da će se pri predviđanju upotrijebiti sve jezgre procesora na računalu kojem se isto izvršava.

Treći prozor *Priloga 11* sadrži postupak traženja najbolje kombinacije *solvers* i *c_values* argumenata hiperparametra uporabom svih ulaznih varijabli. To se vrši stvaranjem dvije liste (*solvers* i *c_values*), koje sadrže vrijednosti argumenta za pojedini hiperparametar. Ubacivanje vrijednosti izvodi se upotrebom dvije *for* petlje (jedna unutar druge), kako bi se mogao dobiti rezultat najoptimalniji rezultat kombinacije navedenih hiperparametra. Tim postupkom višestrukim kombinacijama, pri pojedinim prolazima *for* petlje, utvrđeno je da algoritam *solver-a 'newton-cg'* daje najprihvatljivije rezultate, kada je uz to vrijednost *C* hiperparametra jednaka *0.1*.

Peti prozor *Priloga 11* sadrži identični postupak samo je umjesto *X_transfl* argumenta unutar *back_testing* funkcije u unutrašnjoj *for* petlji za drugi argument parametra, kojim se zadaje popis stupaca, zadana lista *X_btc*. Pri završetku tog postupaka, najbolje rezultate proizvodi kombinacija parametara modela logističke regresije *solver='newton-cg'* i *C=1*.

5.4.2. Traženje optimalnih hiperparametara klasifikatora ekstremnih pojačanja gradijenata

Klasifikator *XGBoost* modela, uvezen je iz biblioteke *xgboost*, klasom *XGBClassifier*. Odabrani hiperparametri za podešavanje jesu: *n_estimators* i *learning_rate* prema [23]. Parametar *n_estimators* označuje broj pojedinačnih stabla odlučivanja koji se pri pojavljuju pri problematici treniranja modela, dok je *learning_rate* već opisana stopa učenja modela. Zadani hiperparametri jesu: *n_estimators=100* te *learning_rate=0.3*.

Rješenje traženja najbolje kombinacije hiperparametara upotrebom svih prediktora je kad su hiperparametri: *n_estimators=200* i *learning_rate=0.3*. Dok najpovoljnije rješenje kombinacije hiperparametara upotrebom prediktora bez rezultata analize teksta, daje jednaki broj stabla odlučivanja te stopa učenja jednaka 0.1. Navedeno je dobiveno sličnim postupkom, te uporabom funkciji kao i kod modela logističke regresije.

5.4.3. Traženje optimalnih hiperparametara klasifikatora potpornih vektora

Kao dio *svm* klase *sklearn* biblioteke je *SVC* funkcija koja definira klasifikacijski model Potpornih Vektora. Parametri funkcije *SVC* koji se podešavaju jesu *C* i *kernel*. Parametar *kernel* ima slično značenje kao hiperparametar *solver* prethodnog modela, dok *C* ima jednako značenje kao i kod Logističke Regresije. Razlika je u tipu algoritma *kernel-a* u usporedbi s *solver-om* od koji je zadani *'rbf'*. Funkcija *SVC* ne podržava parametar *n_jobs* te je potrebno definirati parametar *probability* na vrijednost *True*, jer se u suprotnom javlja greška pri izvođenju programskog koda koja sugerira da se taj parametar omogući.

Najbolje rezultate pri traženju kombinacije hiperparametara upotrebom svih prediktorima daju postavke hiperparametara: *C=0.1* te *kernel='sigmoid'*. U slučaju predviđanja

bez rezultata analize sentimenata najbolje rezultate proizvode identične postavke, što je vidljivo u *Prilogu 13*.

Iako je iz dijela izlaza koda u drugom prozoru iz *Priloga 13* vidljivo, prikazanog na slici 47 da su odzivi vrijednosti prvog predviđanja, točnije da 1. i 5. rezultat pri određenoj kombinaciji hiperparametara daju maksimalnu vrijednost odziva u kombinaciji s najuspješnijim AUC parametrom (pri usporedbi s ostalim rezultatima). Ti rezultati su zanemareni jer je u tim slučajevima došlo do podtreniranosti SVM klasifikatora, koji je pri treniranju prepoznao da postoji više pozitivnih u usporedbi s negativnim klasama te je u tom slučaju je proizveo da su sve vrijednosti ili velika većina predviđanja jednake 1. Prema tome će odziv dati lažnu predodžbu o uspješnosti modela, jer se u opisnom slučaju ne proizvodi ni jedna lažno negativna vrijednost pa će odziv težiti rezultatu *1.0*.

```
Kernel:sigmoid,C:0.01
accuracy_score: 51.08%
precision_score: 51.08%
recall: 1.0000
AUC:0.5300
-----
Kernel:sigmoid,C:0.1
accuracy_score: 54.21%
precision_score: 53.63%
recall: 0.7647
AUC:0.5113
-----
Kernel:sigmoid,C:1
accuracy_score: 52.64%
precision_score: 52.72%
recall: 0.7059
AUC:0.4766
-----
Kernel:poly,C:0.01
accuracy_score: 50.96%
precision_score: 51.02%
recall: 0.9976
AUC:0.5245
-----
Kernel:poly,C:0.1
accuracy_score: 50.96%
precision_score: 51.02%
recall: 0.9976
AUC:0.5337
```

Slika 47: Dio izlaznog koda pri traženju optimalnih hiperparametara SVM klasifikatora

Izvor: Izradio student pomoću Jupyter Notebook IDE-a

5.5. USPOREDBA USPJEŠNOSTI KLASIFIKACIJSKIH MODELA

U ovom podpoglavlju uspoređeni su rezultati uspješnost klasifikacijska modela. Uspoređeni su rezultati uspješnosti pojedinog modela korištenjem svih prediktora pri predviđanju smjera kretanja cijene s rezultatima uspješnosti kada se za ulazni set podataka uzme dio koji ne sadrži rezultate analize sentimenta.

5.5.1. Rezultati uspješnosti logističke regresije

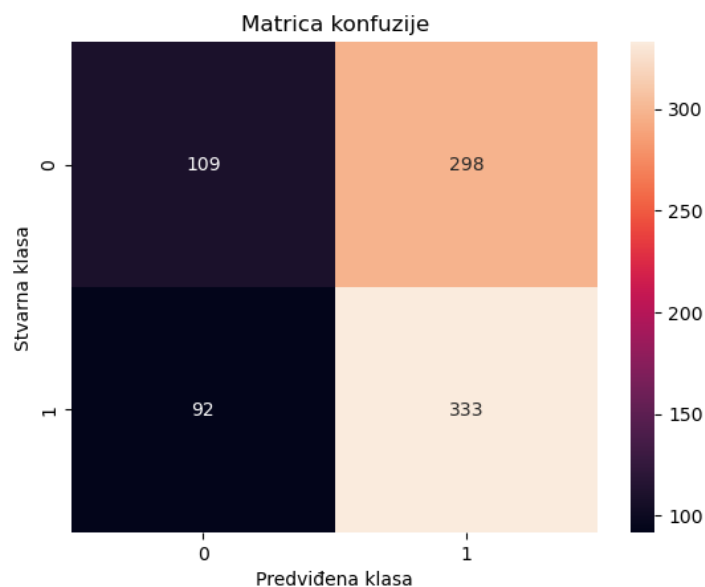
Rezultati uspješnosti modela logističke regresije pri uporabi svih ulaznih varijabli prikazani su u tablici 4.

Tablica 4: Rezultati uspješnosti logističke regresije upotrebom svih prediktora

Točnost	Preciznost	Odziv	AUC
53.12%	52.77%	0.7835	0.5292

Izvor: Izradio student

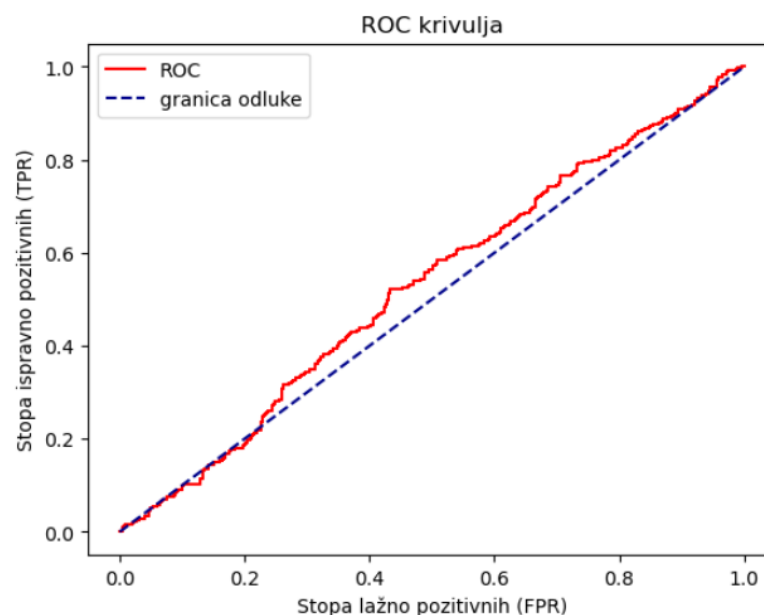
Iz pregleda matrice konfuzije na slici 47, od ukupno 832 predviđanja broj ispravno negativnih predviđanja (TN) iznosi 109, broj lažno negativnih predviđanja klase (FN) iznosi 298, dok je 92 lažno pozitivnih predviđanja (FP) te 333 ispravno pozitivnih predviđanja (TP).



Slika 48: Matrica konfuzije modela logističke regresije upotrebom svih prediktora

Izvor: Izradio student upotrebom Jupyter Notebook IDE-a

Prikaz ROC krivulje logističke regresije upotrebom svih prediktora, čiji parametar AUC iznosi 0.5292 prikazana je na slici 49.



Slika 49: ROC krivulja modela Logističke Regresije upotrebom svih prediktora

Izvor: Izradio student upotrebom Jupyter Notebook IDE-a

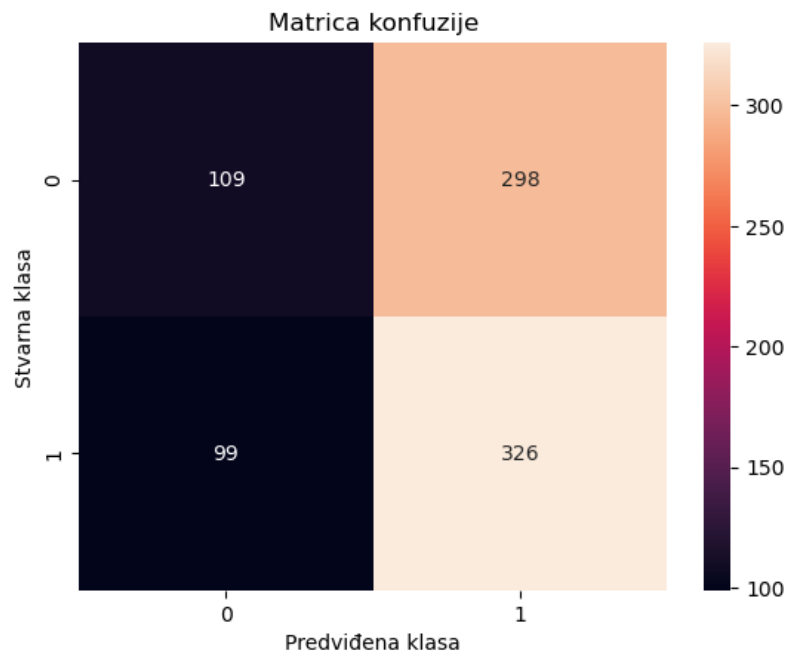
Rezultati uspješnosti modela logističke regresije, bez uporabe ulaznih varijabli analize sentimenata, prikazani su u tablici 5.

Tablica 5: Rezultati uspješnosti Logističke regresije bez analize sentimenata

Točnost	Preciznost	Odziv	AUC
52.28%	52.24%	0.7671	0.5271

Izvor: Izradio student

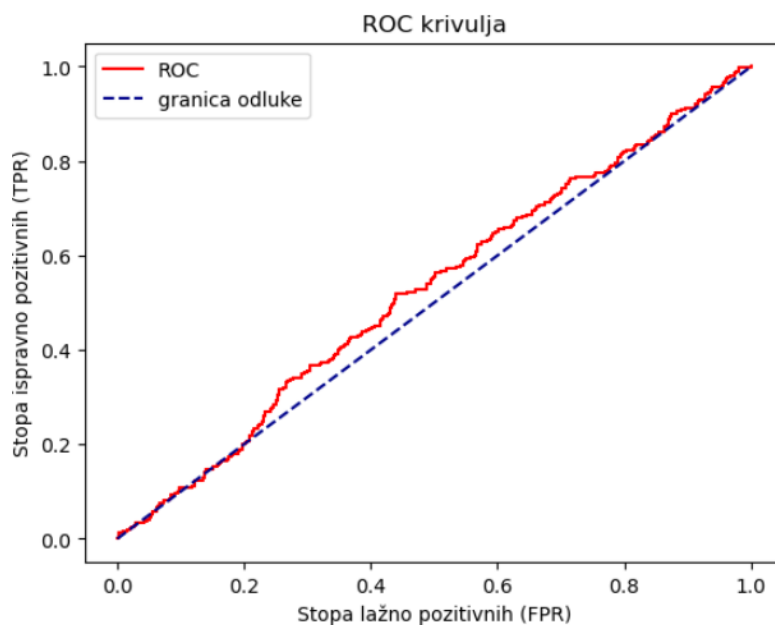
Iz pregleda matrice konfuzije na slici 50, od ukupno 832 predviđanja broj ispravno negativnih predviđanja (TN) iznosi 109, broj lažno negativnih predviđanja klase (FN) iznosi 298, dok je 99 lažno pozitivnih predviđanja (FP) te 326 ispravno pozitivnih predviđanja (TP). U odnosu na usporedbu predviđanja sa svim prediktorima brojevi TN i FN predviđanja je ostao isti, dok se povećao broj FP-a te smanjio broj TP-a, što je lošije u odnosu na prethodni rezultat. To je dokazano usporedbom tablice 4 i tablice 5, gdje je u tablici 4 prisutno povećanje točnosti iznosi 0.84 %, povećanje preciznosti iznosi 0.53%, *odziv* je povećan za 1.64 % te AUC parametar za 0.21 %.



Slika 50: Matrica konfuzije logističke regresije bez upotrebe analize sentimenata kao prediktora

Izvor: Izradio student upotrebom Jupyter Notebook IDE-a

ROC krivulja logističke regresije pri predviđanju bez prediktora analize sentimenata, čiji AUC parametar iznosi 0.5271, prikazana je na slici 51.



Slika 51: ROC krivulja Logističke Regresije pri predviđanju bez analize sentimenata

Izvor: Izradio student upotrebom Jupyter Notebook IDE-a

5.5.2. Rezultati uspješnosti klasifikatora ekstremnih pojačanja gradijenata

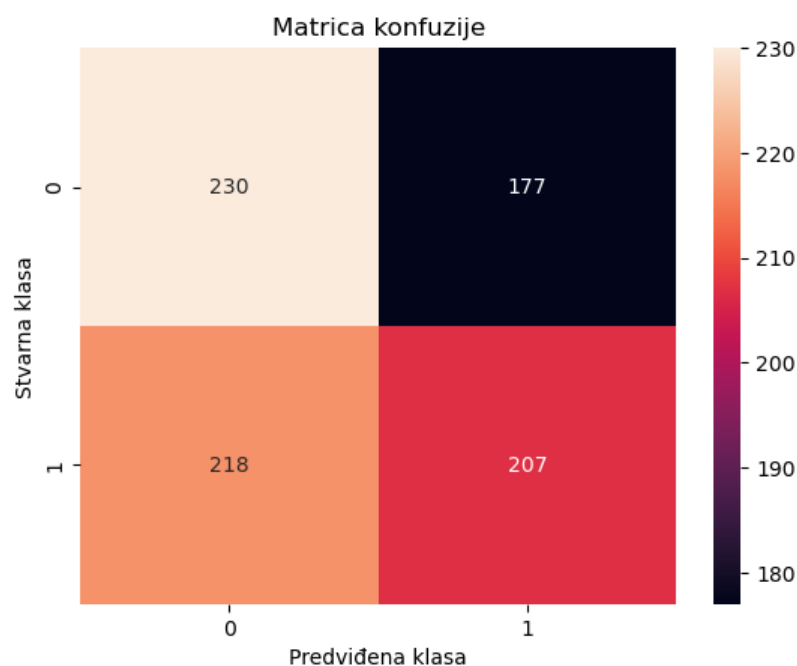
Rezultati uspješnosti klasifikatora ekstremnih pojačanja gradijenata pri uporabi svih ulaznih varijabli prikazani su u tablici 6.

Tablica 6: Rezultati uspješnosti XGBoost klasifikatora predviđanjem upotrebom svih prediktora

Točnost	Preciznost	Odziv	AUC
52.52%	53.91%	0.4871	0.5076

Izvor: Izradio student

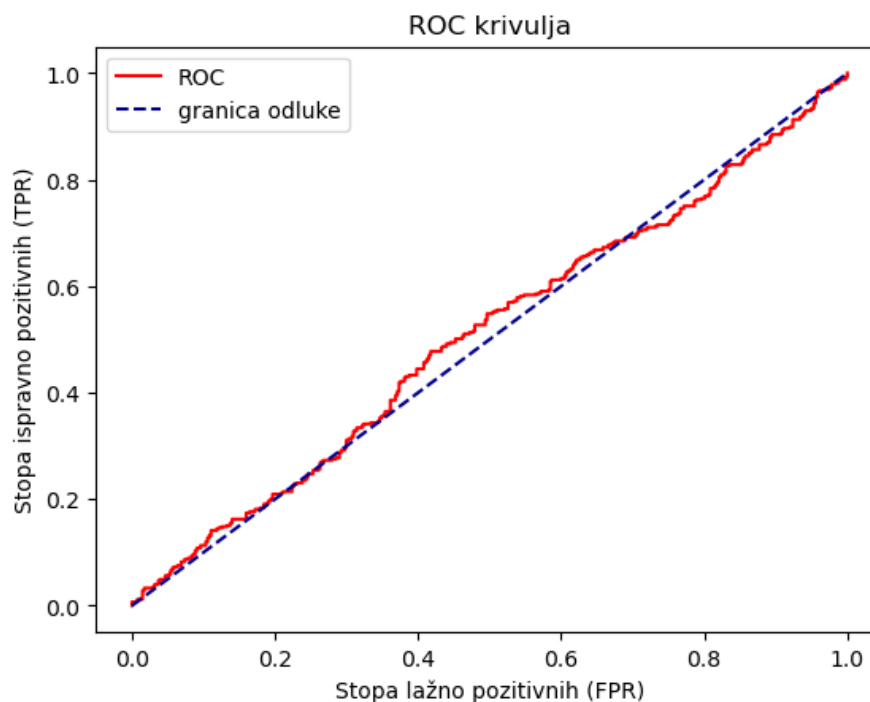
Iz pregleda matrice konfuzije na slici 52, od ukupno 832 predviđanja broj ispravno negativnih predviđanja (TN) iznosi 230, broj lažno negativnih predviđanja klase (FN) iznosi 177, dok je 218 lažno pozitivnih predviđanja (FP) te 207 ispravno pozitivnih predviđanja (TP).



Slika 52: Matrica konfuzije XGBoost klasifikatora upotrebom svih prediktora

Izvor: Izradio student upotrebom Jupyter Notebook IDE-a

ROC krivulja XGBoost klasifikatora pri upotrebi svih prediktora, čiji AUC parametar iznosi 0.5076, prikazana je na slici 53.



Slika 53: ROC krivulja XGBoost klasifikatora upotrebom svih prediktora

Izvor: Izradio student upotrebom Jupyter Notebook IDE-a

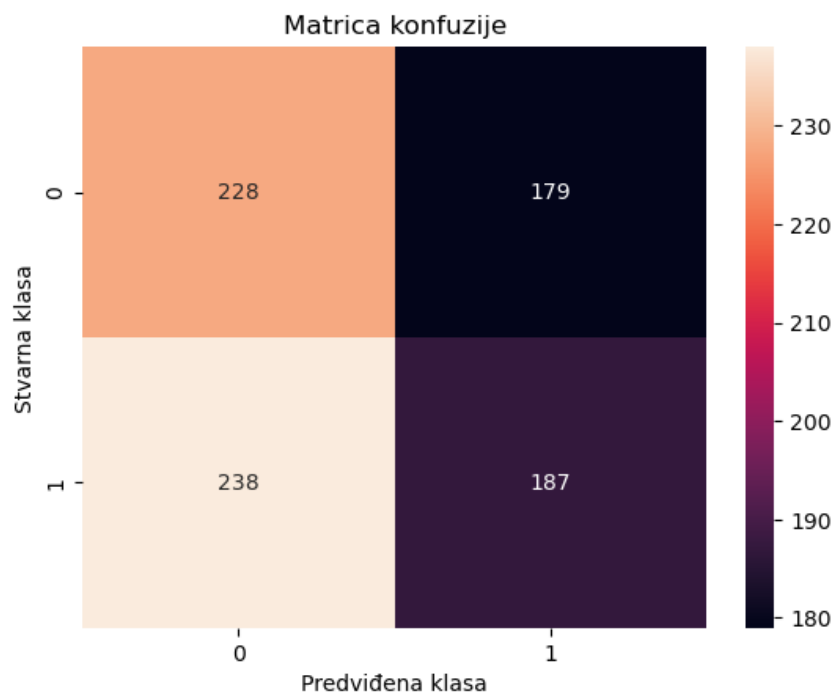
Rezultati uspješnosti modela Klasifikatora Ekstremnih Pojačanja Gradijenata bez uporabe ulaznih varijabli o analizi sentimenta prikazani su u tablici 7.

Tablica 7: Rezultati uspješnosti XGBoost klasifikatora predviđanjem bez prediktora analize sentimenta

Točnost	Preciznost	Odziv	AUC
49.88%	51.09%	0.4400	0.4742

Izvor: Izradio student

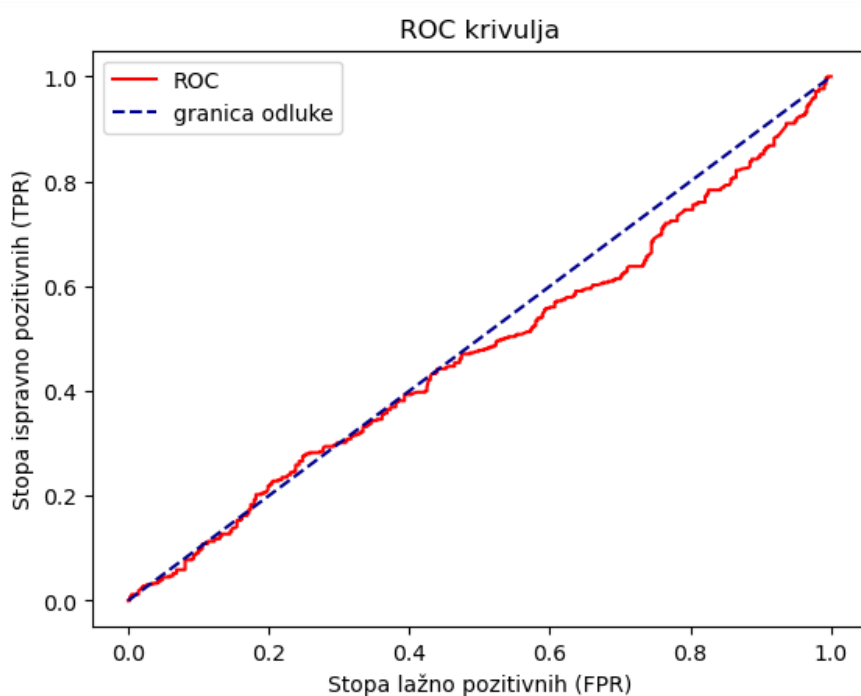
Iz pregleda matrice konfuzije na slici 54, od ukupno 832 predviđanja broj ispravno negativnih predviđanja (TN) iznosi 228, broj lažno negativnih predviđanja klase (FN) iznosi 179, dok je 238 lažno pozitivnih predviđanja (FP) te 187 ispravno pozitivnih predviđanja (TP). Navedeni rezultati imaju, u odnosu na predviđanje sa svim prediktorima, za 2 predviđanja lošiji TP, za 2 predviđanja veći FP, za 20 predviđanja lošiji TP te za isti broj veći FP. Poboľšanje je jasno vidljivo iz tablice rezultata uspješnosti gdje se očituje povećanje točnosti za 3.64%, povećanje preciznosti za 2.82%, odziva za 4.71 % te AUC parametra za 3.34 %, u korist upotrebe svih prediktora, usporedbom tablice 6 s tablicom 7.



Slika 54: Matrica konfuzije XGBoost klasifikatora bez analize sentimenata

Izvor: Izradio student upotrebom Jupyter Notebook IDE-a

ROC krivulja *XGBoost* klasifikatora pri predviđanju bez analize sentimenata, čiji AUC parametar iznosi 0.4742, prikazana je na slici 55.



Slika 55: ROC krivulja XGBoost klasifikatora pri predviđanju bez analize sentimenata

Izvor: Izradio student upotrebom Jupyter Notebook IDE-a

5.5.3. Rezultati uspješnosti klasifikatora potpornih vektora

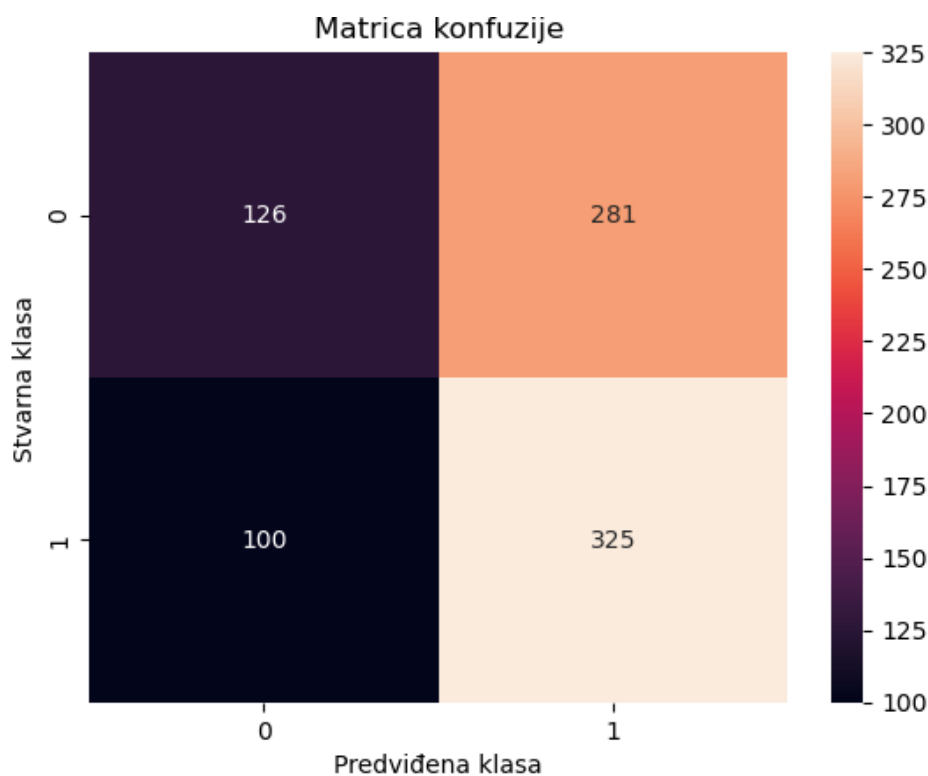
Rezultati uspješnosti modela klasifikacije potpornih vektora upotrebom svih ulaznih varijabli prikazani su u tablici 8.

Tablica 8: Rezultati uspješnost SVM klasifikatora uporabom svih prediktora

Točnost	Preciznost	Odziv	AUC
54.21%	53.63%	0.7647	0.5113

Izvor: Izradio student

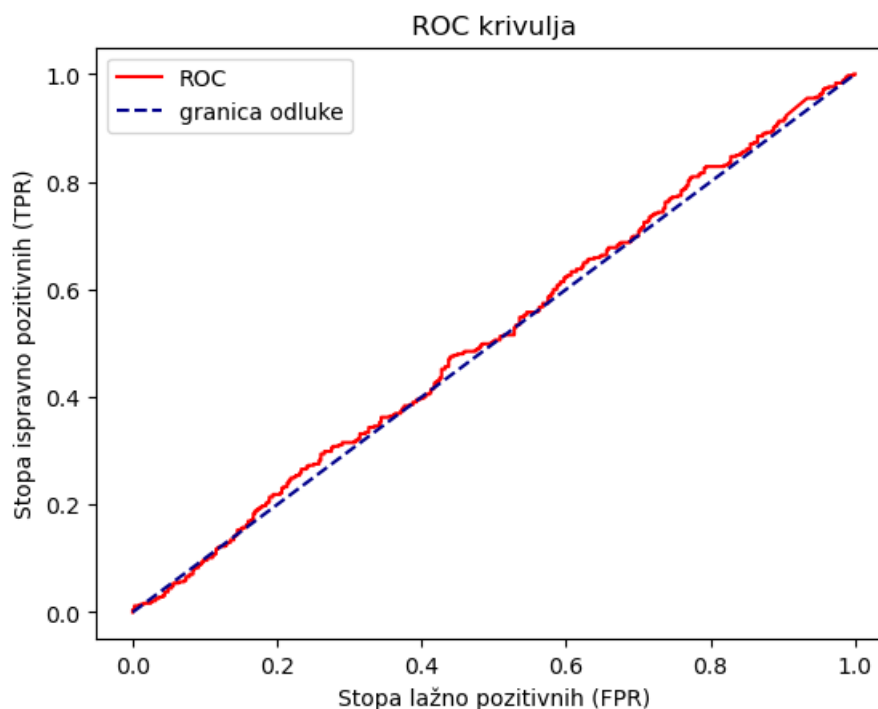
Iz pregleda matrice konfuzije na slici 56, od ukupno 832 predviđanja broj ispravno negativnih predviđanja (TN) iznosi 126, broj lažno negativnih predviđanja klase (FN) iznosi 281, dok je 100 lažno pozitivnih predviđanja (FP) te 325 ispravno pozitivnih predviđanja (TP).



Slika 56: Matrica konfuzije SVM klasifikatora uporabom svih prediktora

Izvor: Izradio student upotrebom Jupyter Notebook IDE-a

Prikaz ROC krivulje SVM klasifikatora upotrebom svih prediktora, čiji parametar AUC iznosi 0.5113 prikazana je na slici 57.



Slika 57: ROC krivulja SVM klasifikatora upotrebom svih prediktora

Izvor: Izradio student upotrebom Jupyter Notebook IDE-a

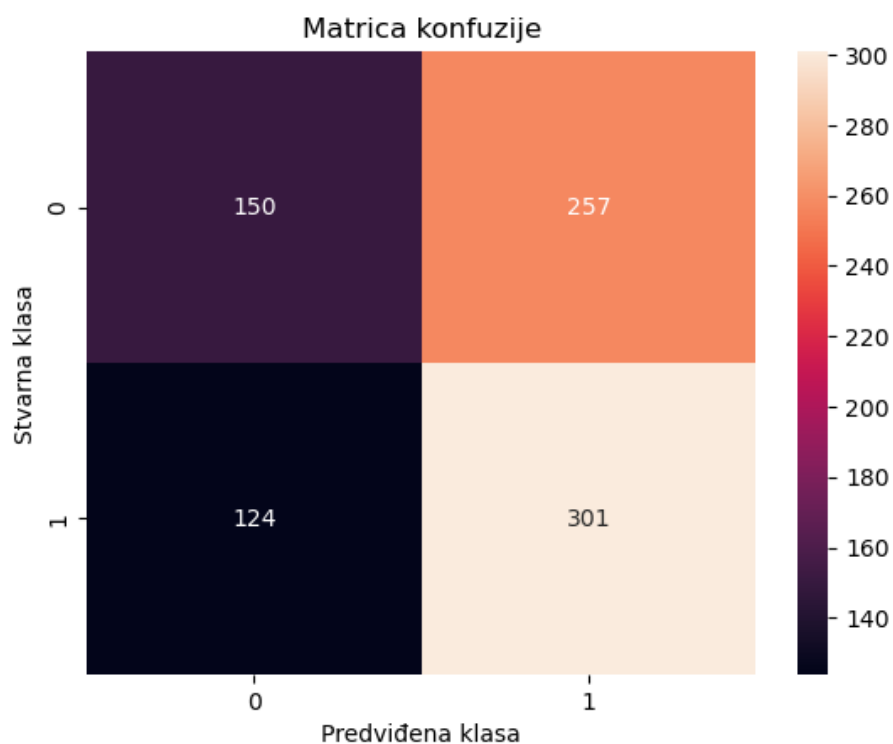
Rezultati uspješnosti modela klasifikatora ekstremnih pojačanja gradijenata bez uporabe ulaznih varijabli o analizi sentimenta prikazani su u tablici 9.

Tablica 9: Rezultati uspješnost SVM klasifikatora bez prediktora analize sentimenta

Točnost	Preciznost	Odziv	AUC
54.21%	53.94%	0.7082	0.4956

Izvor: Izradio student

Iz pregleda matrice konfuzije na slici 58, od ukupno 832 predviđanja broj ispravno negativnih predviđanja (TN) iznosi 150, broj lažno negativnih predviđanja klase (FN) iznosi 257, dok je 124 lažno pozitivnih predviđanja (FP) te 301 ispravno pozitivnih predviđanja (TP). Navedeni rezultati imaju u odnosu na predviđanje sa svim prediktorima za 24 predviđanja veći TP te za 24 predviđanja manji FP, ali uz to također za 24 predviđanja manji TP te za isti broj veći FP.

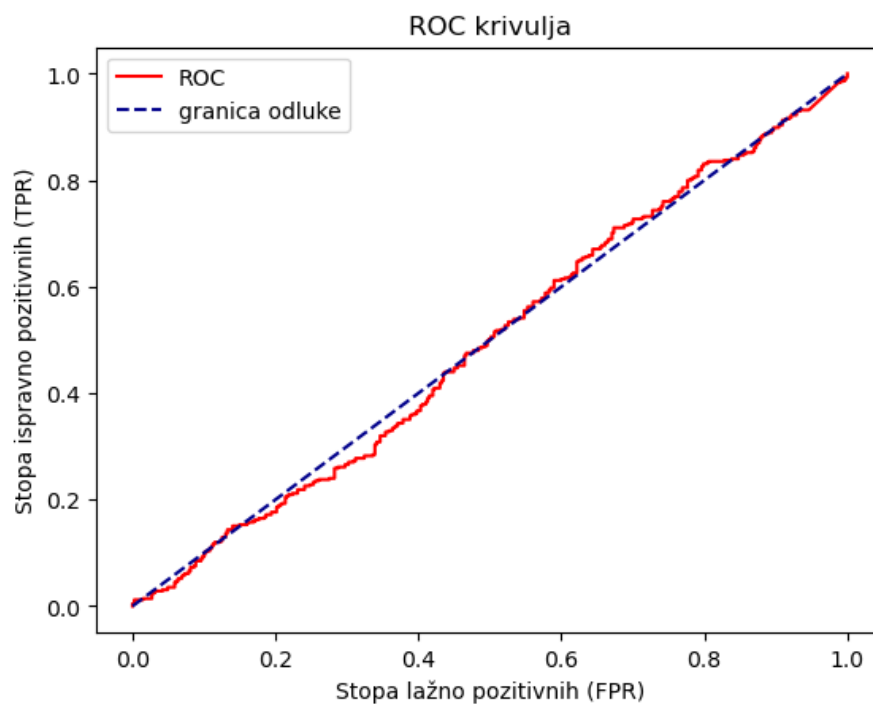


Slika 58: Matrica konfuzije SVM klasifikatora bez prediktora analize sentimenta

Izvor: Izradio student upotrebom Jupyter Notebook IDE-a

Prema rezultatima matrice konfuzije vidljivo je da je ovaj slučaj specifičan, iz razloga jer su neki rezultati bolji pri usporedbi predviđanja sa svim prediktorima. Pa su prema tome rezultati točnosti u oba slučaja jednaki te iznose 54.21%, rezultat preciznosti blago ide u korist rezultatu predviđanja bez analize sentimenta za 0.31%. Mjera odziva veća je za 5.65% kod predviđanja s svim prediktorima te je AUC parametar veći za 1.57 % kod istog slučaja.

ROC krivulja SVM klasifikatora pri predviđanju bez analize sentimenta, čiji AUC parametar iznosi 0.4956, prikazana je na slici 59.



Slika 59: ROC krivulja SVM klasifikatora bez prediktora analize sentimenta

Izvor: Izradio student upotrebom Jupyter Notebook IDE-a

6. ZAKLJUČAK

Dodavanje rezultata analize sentimenta teksta, u manjoj mjeri te u većini slučajeva povećava točnost i preciznost modela, dok je povećanje odziva i AUC parametra prisutno kod sva tri modela. Mjera AUC-a uz mjere točnosti i preciznosti imaju najveći značaj pri analizi uspješnosti modela jer daju pregled koliko je određeni model sposoban razlikovati klase binarne klasifikacije. Najlošiju predodžbu, ali ne i manje važnu, daje mjera odziv jer sama po sebi može dati lažan dojam uspješnosti modela, gdje ako model raspoznaje da postoji više pozitivnih klasa, jednostavno postavi da su sva predviđanja klase 1. U tom slučaju će mjera odziva biti 1.0 ili 100%, zato što pri predviđanju pojave podtreniranosti modela, isti nije predvidio niti jednu negativnu klasu ili 0. Takav slučaj uočljiv je jednostavnom usporedbom ostalih mjera poput preciznosti i točnosti s mjerom odziva, ako navedene mjere daju značajno lošiji rezultat od mjere odziva koja teži ili je jednaka vrijednosti 1.0.

Iako su ROC krivulje "glatke", što znači da se površine ispod krivulje kreću oko vrijednosti 0.5, što je granica pogađanja (nema raspoznavanja klasa), dodavanjem prediktora analize sentimenta omogućuje se povećanje AUC parametra, što je direktan pokazatelj poboljšanja prikaza ROC krivulje. Najbolje rezultate AUC površine ispod ROC krivulje daje logistička regresija, dok najgore daje model ekstremnog pojačanja gradijenata. Zanimljivo zapažanje je to što je klasifikator potpornih vektora imao jednako dobre rezultate točnosti, zanemarivo bolje rezultate preciznost, dok rezultati mjera odziva i AUC-a pokazuju veću uspješnost pri korištenju svih ulaznih varijabli.

Navedeni modeli predviđanja bi prema pretpostavci autora mogli dati bolje rezultate kad bi se analizirao sentiment teksta s javnosti poznatijih društvenih mreža poput *Twitter-a*, *Facebook-a*, *Reddit-a* i sličnih, jer bi u tom slučaju postojalo više objava vezanih za *Bitcoin* kriptovalutu na dnevnoj bazi, što bi doprinijelo rezultatima uspješnosti predviđanja. Navedeno bi i značajno zakompliciralo postupak struganja podataka s takvih stranica, jer njihova politika posjeduje ograničenje za takav način dohvata informacija.

Dakle, bez obzira na loše rezultate, zadana hipoteza je po mišljenju autora dokazana jer je usporedbom više klasifikacijskih modela dokazano da analiza sentimenta teksta može doprinijeti predviđanju kretanja cijene *Bitcoin* kriptovalute na dnevnoj bazi.

LITERATURA

- [1] Tikkanen, A., 'A Brief (and Fascinating) History of Money', Britannica, online: <https://www.britannica.com/story/a-brief-and-fascinating-history-of-money> (6.5.2023.)
- [2] Kokanović, J. 2023., 'Fiat novac - Što je i koji su njegovi nedostaci?', Centar Zlata, online: <https://www.centarzlata.com/fiat-novac/> (6.5.2023.)
- [3] *Bitcoin* (BTC)', online: [https://www.investing.com/education/terms/Bitcoin-\(btc\)-200430790?gl_campaign_id](https://www.investing.com/education/terms/Bitcoin-(btc)-200430790?gl_campaign_id) (4.3.2023.)
- [4] Antonopoulos, A. 2017, *Mastering Bitcoin: Programming the Open Ulančani blokovi*, 2.izdanje, O'Reilly Media, 1005 Gravenstein Highway North, Sebastopol
- [5] Kumari, R. S. 2017, *Machine Learning: A Review on Binary Classification* International Journal of Computer Applications, vol. 160-No 7
- [6] Loo, A. et. al. 2023, 'Cryptocurrency vs Stocks - Overview, Similarities, Differences', online: <https://corporatefinanceinstitute.com/resources/cryptocurrency/cryptocurrency-vs-stocks/> (7.5.2023.)
- [7] Barney, N. 2023, 'sentiment analysis (opinion mining)', TechTarget, online: <https://www.techtarget.com/searchbusinessanalytics/definition/opinion-mining-sentiment-mining> (18.3.2023.)
- [8] Pascual, J. 2018, 'What is HashCash?', bit2me, online: <https://academy.bit2me.com/en/what-is-hashcash/> (18.3.2023.)
- [9] E-Gold, stanford.edu, online: <https://cs.stanford.edu/people/eroberts/cs201/projects/2010-11/Bitcoins/e-gold.html> (15.5.2023.)
- [10] Moskov, P. 2018., 'What Is Bit Gold? The Brainchild of Blockchain Pioneer Nick Szabo', CoinCentral, online: <https://coincentral.com/what-is-bit-gold-the-brainchild-of-blockchain-pioneer-nick-szabo/> (15.5.2023.)
- [11] Reiff, N. 2021, 'B-Money: Overview, Goals, Differences From *Bitcoin*', Investopedia, online: <https://www.investopedia.com/terms/b/bmoney.asp>

- [12] Reiff, N. 2022, 'What Was the First Cryptocurrency?', online: <https://www.investopedia.com/tech/were-there-cryptocurrencies-Bitcoin/> (4.3.2023.)
- [13] Gaur, A. et. al. 2022, 'P2P', Encyclopedia Britannica, online: <https://www.britannica.com/technology/P2P> (18.3.2023.)
- [14] Nakamoto, S. 2008, *Bitcoin: A Peer-to-Peer Electronic Cash System*
- [15] Rosencrance, L. 2022, 'What is peer-to-peer (P2P)?', online: <https://www.techtarget.com/searchnetworking/definition/peer-to-peer> (13.3.2023.)
- [16] Wang K, et. al. 2020, 'Defending Blockchain Forking Attack by Delaying MTC Confirmation', Research Gate, online: https://www.researchgate.net/publication/342017074_Defending_Blockchain_Forking_Attack_by_Delaying_MTC_Confirmation (9.3.2023.)
- [17] 'Block Chain', Bitcoin Developer, online: https://developer.Bitcoin.org/reference/block_chain.html (9.3.2023.)
- [18] Son, D. 2019, Bitcoin#6: Target and Difficulty. How to calculate, Medium, online: <https://medium.com/@dongha.sohn/bitcoin-6-target-and-difficulty-ee3bc9cc5962> (15.7.2023.)
- [19] VasuDev4, 'hashlib module in Python', GeeksforGeeks, online: <https://www.geeksforgeeks.org/hashlib-module-in-python/> (24.5.2023.)
- [20] Barney, N. 2023, 'What Is Sentiment Analysis (Opinion Mining)?', TechTarget, online: <https://www.techtarget.com/searchbusinessanalytics/definition/opinion-mining-sentiment-mining> (24.5.2023.)
- [21] Breuss M., 'Beautiful Soup: Build a Web Scraper With Python', Real Python, online: <https://realpython.com/beautiful-soup-web-scraper-python/> (23.5.2023.),
- [22] Vaswani A., et. al. 2017, *Attention is All You Need*, Advances in Neural Information Processing Systems
- [23] Shah, A. 2023, 'An Introduction To HuggingFace Transformers for NLP', Weights & Biases,, online: https://wandb.ai/int_pb/huggingface/reports/An-Introduction-To-HuggingFace-Transformers-for-NLP--VmlldzoyOTgzMjI5 (18.3.2023.)

- [24] 'What Is An API (Application Programming Interface)?', AWS <https://aws.amazon.com/what-is/api/> (18.3.2023.)
- [25] Transformers', HuggingFace, online: <https://huggingface.co/docs/transformers/index> (18.3.2023.)
- [26] 'What Is A Neural Network?', AWS, online: <https://aws.amazon.com/what-is/neural-network/> (20.3.2023.)
- [27] Chandola, Y. 2021, 'Binary Classification - an overview', ScienceDirect Topics, online: <https://www.sciencedirect.com/topics/computer-science/binary-classification> (24.5.2023.)
- [28] Džalto S., Gusić I. 2017, *Simulacija jednostavne linearne regresije*, Kemija u industriji/Journal of Chemists and Chemical Engineers
- [29] Belyadi H., Haghighat A. 2021, *Supervised Learning*, 'Machine Learning Guide for Oil and Gas Using Python'
- [30] Hali, B. 2019, 'Understanding Bias-Variance Trade-Off in 3 Minutes', Towards Data Science, online: <https://towardsdatascience.com/understanding-bias-variance-trade-off-in-3-minutes-c516cb013513> (21.3.2023.)
- [31] Su, G. 2009, Linear Regression Analysis: Theory and Computing
- [32] Triangles, 2019, 'The cost function in logistic regression - Internal Pointers', online: <https://www.internalpointers.com/post/cost-function-logistic-regression> (29.5.2023.)
- [33] Agrawal, A. , 'Logistic Regression. Simplified.. After the basics of Regression, it's...', Data Science Group, IITR, Medium online: <https://medium.com/data-science-group-iitr/logistic-regression-simplified-9b4efe801389> (29.5.2023.)
- [34] Maulana, B 2019, 'Support Vector Machine: Classification' , IT Paragon, Medium, online: <https://medium.com/it-paragon/grid-search-f24a73a8a0ac> (20.3.2023.)
- [35] Fan, S. 2018, 'Understanding the mathematics behind Support Vector Machines Fan, online: <https://shuzhanfan.github.io/2018/05/understanding-mathematics-behind-support-vector-machines/> (30.5.2023.)

- [36] MLMath.io, Math Behind SVM(Kernel Trick). This is PART III of SVM Series, Medium, online: <https://ankitnitjsr13.medium.com/math-behind-svm-kernel-trick-5a82aa04ab04> (30.5.2023.)
- [37] Morde, M. 2019, 'XGBoost Algorithm: Long May She Reign!', Towards Data Science, online: <https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d> (20.3.2023.)
- [38] Pal, A. , 'Gradient Boosting Trees For Classification: A Beginner's Guide', Affine, online: <https://affine.ai/gradient-boosting-trees-for-classification-a-beginners-guide/> (21.3.2023)
- [39] Polamuri , S. 2020, 'Ensemble Methods - Bagging Vs Boosting Difference', Dataaspirant, online: <https://dataaspirant.com/ensemble-methods-bagging-vs-boosting-difference/> (14.7.2023.)
- [40] Shaw, R. 2017, 'XGBoost: A Concise Technical Overview', KD Nuggets, online: <https://www.kdnuggets.com/2017/10/xgboost-concise-technical-overview.html> (27.3.2023.)
- [41] Narkhede, S. 2018, ' Understanding Confusion Matrix ', Towards Data Science, online: <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62> (27.3.2023)
- [42] Anello, E 2021, 'How to Evaluate your Model using the Confusion Matrix', Towards AI, online: <https://pub.towardsai.net/deep-understanding-of-confusion-matrix-6ab1f88a267e> (27.3.2023.)
- [43] Narkhede, S. 2018, 'Understanding AUC - ROC Curve', Towards Data Science, online: <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5> (27.3.2023.)
- [44] Bandy, P. 2020, 'BitcoinTalk: The Absolute Guide to Acquiring Ultra-Targeted Traction for Your Crypto Start-Up', BUNCH, online: <https://www.meetbunch.com/the-Bitcoin-talk-guide> (27.3.2023)
- [45] 'Requests - Web Scraping using Requests', tutorialspoint, online: https://www.tutorialspoint.com/requests/requests_web_scraping_using_requests.htm (30.3.2023.)

- [46] Osipenko, A 2018, 'Backtesting Time Series models — Weekend of a Data Scientist' , Medium, online: <https://medium.com/@subpath/backtesting-time-series-models-weekend-of-a-data-scientist-92079cc2c540> (6.4.2023.)
- [47] Brownlee, J. 2019, 'Tune Hyperparameters for Classification Machine Learning Algorithms', Machine Learning Mastery, online: <https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/> (8.4.2023.)
- [48] Pandey, P. 2019, 'Simplifying the ROC and AUC metrics' , Towards Data Science, online: <https://towardsdatascience.com/understanding-the-roc-and-auc-curves-a05b68550b69> (20.3.2023.)

KAZALO KRATICA

Kratika	Puni naziv na stranom jeziku	Tumačenje na hrvatskom jeziku
AUC	engl. <i>Area Under Curve</i>	Površina pod ROC krivuljom
BTC	engl. <i>Bitcoin</i>	Kriptovaluta <i>Bitcoin</i>
FN	engl. <i>False Negative</i>	Lažno negativna predviđanja klase
FP	engl. <i>False Positive</i>	Lažno pozitivna predviđanja klase
FPR	engl. <i>False Positive Rate</i>	Stopa lažno pozitivnih rezultata predviđanja
GBT	engl. <i>Gradient Boosted Trees</i>	Model gradijentom poboljšanih stabla odlučivanja
IDE	engl. <i>Integrated Development Enviroment</i>	Softverska aplikacija koja pomaže programeru u učinkovitom razvoju softverskog koda
P2P	engl. <i>Peer to Peer</i>	Tip arhitekture mreže gdje se informacije izmjenjuju po principu svaki sa svakim
PoW	engl. <i>Proof of Work</i>	Dokaz da je rudar obavio određeni računalni rad pri verifikaciji transakcija
ROC	engl. <i>Reciever Operating Characteristic</i>	Radna krivulja prijammnika
SHA256	engl. <i>Secure Hash Algorithm</i>	Sigurnosni algoritam kriptiranja koji dani ulaz na izlazu kodira na veličinu 256 bita
SVM	engl. <i>Support Vector Machine</i>	Model potpornih vektora
TN	engl. <i>True Negative</i>	Ispravno negativna predviđanja klase

TP	engl. True Positive	Ispravno pozitivna predviđanja klase
TPR	engl. <i>True Positive Rate</i>	Stopa ispravno pozitivnih rezultata predviđanja
XGBoost	engl. <i>eXtreme Gradient Boost</i>	Model Ekstremnih Pojačanja Gradijenata

POPIS TABLICA

Tablica 1: Polja zaglavlja bloka i njihove veličine	16
Tablica 2: Usporedba pretreniranosti, podtreniranosti te ispravne treniranosti klasifikacijskog modela.....	28
Tablica 3: Usporedba algoritama baziranih na stablima odlučivanja	43
Tablica 4: Rezultati uspješnosti logističke regresije upotrebom svih prediktora	80
Tablica 5: Rezultati uspješnosti Logističke regresije bez analize sentimenata	81
Tablica 6: Rezultati uspješnosti XGBoost klasifikatora predviđanjem upotrebom svih prediktora	83
Tablica 7: Rezultati uspješnosti XGBoost klasifikatora predviđanjem bez prediktora analize sentimenata	84
Tablica 8: Rezultati uspješnost SVM klasifikatora uporabom svih prediktora.....	86
Tablica 9: Rezultati uspješnost SVM klasifikatora bez prediktora analize sentimenata	87

POPIS SLIKA

Slika 1: Usporedba klijent-poslužitelj i P2P mrežnih arhitektura	7
Slika 2: Simboličan prikaz strukture ulančanih blokova	8
Slika 3: Primjer bloka uz identifikatore bloka	12
Slika 4: Grananje unutar jednog bloka.....	13
Slika 5: Grananje unutar dva bloka	14
Slika 6: Postupak dobivanja korijena merkleovog stabla	16
Slika 7: Programski kod za kriptiranje pojedinih ulaza	18
Slika 8: Izlaz programskog koda prikazanog na slici 7	19
Slika 9: Programski kod za kriptiranje u svrhu pronalaska specifične mete uz njegov izlaz	20
Slika 10: Koder-dekoder arhitektura	23
Slika 11: Mehanizmi unutar kodera i dekodera.....	23
Slika 12: Dodavanje vektora pozicijskog kodiranja ulaznom ugrađivanju.....	24
Slika 13: Prosljeđivanje podataka u sloj samo-pozornosti.....	25
Slika 14: Princip rada koder-dekoder arhitekture	26
Slika 15: Usporedba Linearne i Logističke regresije	33
Slika 16: Nekonveksna funkcija $J\theta$	34
Slika 17: Grafički prikaz funkcije cilja za Logističku regresiju	35
Slika 18: Prikaz hiper ravnine SVM klasifikacije.....	36
Slika 19: Prikaz nelinearno odvojivih klasa podataka	38
Slika 20: Transformacija u 3D sustav, te projekcija u y-z koordinatni sustav	39

Slika 21: Transformacija u 2D sustav gdje se linija transformira u krug	39
Slika 22: Preklapanje klasa	40
Slika 23: Usporedba male i velike regularizacije	41
Slika 24: Usporedba tehnika bagging i boosting	45
Slika 25: Skup podataka za predviđanje bolesti srca	46
Slika 26: Tablica nakon izračuna rezidualnih vrijednosti	47
Slika 27: Izrada stabla odlučivanja na temelju ulaznih varijabli	47
Slika 28: Modificirano stablo odlučivanja	48
Slika 29: Dodavanje izračuna novih predikcija za dobivanje novih rezidualnih vrijednosti	50
Slika 30: Pojašnjenja značenja pojedinog dijela matrice konfuzije	52
Slika 31: Primjer ROC krivulje	54
Slika 32: Idealna ROC krivulja	55
Slika 33: ROC krivulja kod slučaja $AUC=0.7$	56
Slika 34: ROC krivulja neraspознаvanja klasa ($AUC=0.5$)	56
Slika 35: Prikaz ROC krivulje sa $AUC=0$	57
Slika 36: Odjeljak Bitcoin Discussion unutar Bitcointalk foruma	59
Slika 37: Blok dijagram programskog koda	60
Slika 38: Html kod unutar Bitcoin Discussion odjeljka foruma	62
Slika 39: Dio tablice izradene struganjem teksta iz Bitcointalk foruma	64
Slika 40: Prikaz dijela sadržaja <code>df_list</code> varijable	65
Slika 41: Prikaz dijela sadržaja rječnik-a teme	66
Slika 42: Usporedba izlaza pri ubacivanju negativnog teksta sa i bez upotrebe funkcije <code>analiza_sent</code>	67
Slika 43: Dio sadržaja <code>rez_transf</code> rječnik-a	68
Slika 44: Dio sadržaja <code>end_transf</code> rječnik-a	69
Slika 45: Tablica i povijesti BTC-USD cijena	71
Slika 46: Postupak validacije metodom utvrđivanja prošle uspješnosti	73
Slika 47: Dio izlaznog koda pri traženju optimalnih hiperparametara SVM klasifikatora	79
Slika 48: Matrica konfuzije modela logističke regresije upotrebom svih prediktora	80
Slika 49: ROC krivulja modela Logističke Regresije upotrebom svih prediktora	81
Slika 50: Matrica konfuzije logističke regresije bez upotrebe analize sentimenta kao prediktora	82
Slika 51: ROC krivulja Logističke Regresije pri predviđanju bez analize sentimenta	82
Slika 52: Matrica konfuzije XGBoost klasifikatora upotrebom svih prediktora	83

Slika 53: ROC krivulja XGBoost klasifikatora upotrebom svih prediktora	84
Slika 54: Matrica konfuzije XGBoost klasifikatora bez analize sentimenata	85
Slika 55: ROC krivulja XGBoost klasifikatora pri predviđanju bez analize sentimenata	85
Slika 56: Matrica konfuzije SVM klasifikatora upotrebom svih prediktora	86
Slika 57: ROC krivulja SVM klasifikatora upotrebom svih prediktora	87
Slika 58: Matrica konfuzije SVM klasifikatora bez prediktora analize sentimenata.....	88
Slika 59: ROC krivulja SVM klasifikatora bez prediktora analize sentimenata.....	89

PRILOG 1

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import datetime
import re
subject=[]
dat=[]
for i in range(0,25000,40):
    # stranice imaju url pomaknut za 40 gledajući znak poslije ...board=1.
    url = f'https://bitcointalk.org/index.php?board=1.{i}'
    page=requests.get(url).text
    soup = BeautifulSoup(page, 'html.parser')
    date_element= soup.find_all(class_='windowbg2 lastpostcol')
    posts = soup.find_all(class_='windowbg')
    # find_all funkcija vraća listu elemenata iz kojih se izvlači datum
    for date in date_element:
        # 2 elementa liste spremljena u jednu str varijablu
        date=str(list(date.descendants)[5:7])
        # ako element ima više od 20 znakova sadrži datum i vrijeme
        if len(date)>20:
            date=date[20:]
            # search funkcija za pronalazak datuma (npr. February 22, 2023)
            date=re.search('^[A-Z][a-z]{2,8}\s\d{2}\s\d{4}',date).group(0)
            #datum sadrži zarez(,) potrebno ga je ukloniti sub funkcijom
            date=re.sub("\s", "", date)
            # ako ne, radi se o oznaci današnjeg datuma //n...<b>Today<b>
            else:
                #pri čemu se koristi datetime biblioteka za definiranje trenutnog
                datuma
                date=datetime.datetime.now().strftime("%B %d %Y")
            dat.append(date)
    # prolaz kroz svaki element klase 'windowbg'
    for post in posts:
        text=post.find('a')
        # pošto su neki elementi za izlaz daju "None" vrijednost
        try:
            text=text.string
            # ako je text==None
        except:
            continue
        # dodavanje text var u subject listu
        subject.append(text)
```

```
# dat i subject liste se pretvaraju u Series stupce
dat=pd.Series(dat)
# 1. stranica daje jedan element None viška, pa se počinje od 2. reda
subject=subject[1:]
subject=pd.Series(subject)
df=pd.concat({"Date": dat , "subject": subject}, axis=1)
df=df.iloc[3:]
```

PRILOG 2

```
import time
# od 3 jer tablica počinje od 3 indexa do len +3
for i in range(3,len(df.Date)+3):
    # najprije u time.struct_time tuple format
    df.Date[i]=time.strptime(df.Date[i], "%B %d %Y")
    # pa u %Y-%m-%d format
    df.Date[i]=time.strftime("%Y-%m-%d",df.Date[i])
```

```
df=df.set_index('Date')
```

```
df=df.iloc[::-1]
```

```
# potrebno je opet zadati brojevne indexe jer ako su
# datumi postavljeni za index stupac ne prepoznaju se
# kao zasebni stupac
df=df.reset_index()
```

```
# pretvaranje dobivene tablice u listu
df_list = df.to_dict('records')
```

PRILOG 3

```
teme={}
for dic in df_list:
    datum=dic["Date"]
    if datum not in teme:
        teme[datum]=dict(teme_list=list())
    tema=dic["subject"]
    teme[datum]["teme_list"].append(tema)
```

PRILOG 4

```
from transformers import pipeline
sent_pipeline = pipeline("sentiment-analysis")
```

```
def analiza_sent(tekst):
    tema=sent_pipeline(tekst[:100])[0]
    rezultat=tema["score"]
    polaritet=tema["label"]
    if polaritet=="NEGATIVE":
        rezultat*=-1
    return rezultat
```


PRILOG 5

```
rez_transf={}
for dat in teme:
    lista_tema = teme[dat]["teme_list"]
    rez_transf[dat]=dict(broj_tema=len(lista_tema), rezultati=list())
    for tema in lista_tema:
        rez_transf[dat]["rezultati"].append(analiza_sent(tema))
```

```
# funkcije za odvajanje pozitivnih od negativnih tema
def pozitivni(lista):
    poz=[]
    for br in lista:
        if br >= 0:
            poz.append(br)
    return poz

def negativni(lista):
    neg=[]
    for br in lista:
        if br < 0:
            neg.append(br)
    return neg
```

```
end_transf={}

for k in rez_transf:
    rez=rez_transf[k]["rezultati"]
    end_transf[k]=dict(broj_tema=rez_transf[k]["broj_tema"],
                       avg_rezultata=float, posto_poz=float, posto_neg=float)
    end_k=end_transf[k]
    end_k["avg_rezultata"]=sum(rez)/len(rez)
    end_k["posto_poz"]=len(pozitivni(rez))/len(rez)
    end_k["posto_neg"]=len(negativni(rez))/len(rez)
```

PRILOG 6

```
# pretvaranje end_transf rječnika u tablicu
df_transf=pd.DataFrame.from_dict(end_transf,orient="index")
```

```
# definiranje varijable koja će postati novi index
# postavljanje početnog i završnog datuma
datumi=pd.date_range(start="10-09-2017",end="01-23-2023")
```

```
# reindexiranje i ispunjavanje nepostojećih datuma s vrijednostima 0
df_reindex=df_transf.reindex(datumi,fill_value=0)
```

```
# provjera indeksa koji su dodani pri reindexiranju
null=[]
for i in range(len(df_reindex)):
    if df_reindex["broj_kom"].iloc[i]==0:
        null.append(i)
print(f"Index vrijednosti nula = {null}")0
```

```
# izbacivanje NaN vrijednosti koje su kreirane rolling funkcijom
rolling_transf=rolling_transf.dropna()
```

```
# Prebacivanje konačne tablice u .csv dokument
rolling_transf.to_csv("Transf_roll7.csv")
```

PRILOG 7

```
# uvoz yfinance biblioteke koja omogućuje preuzimanje podataka o
kriptovalutama
import yfinance as yf
# filtriranje nepotrbnih .copy() upozorenja
import warnings
warnings.filterwarnings("ignore")
```

```
btc=yf.download("BTC-USD",start="2017-10-09")
```

```
# provjera jesu li iste tablice Adj Close i Close
eq=btc["Close"]==btc["Adj Close"]
eq.value_counts()
```

```
#Brisanje nepotrebnih Close kolona
btc.drop(columns=["Close"],inplace=True)
```

```
# importiranje biblioteke pandas i definiranje parametara za pregled outputa
import pandas as pd
pd.set_option('display.max_rows', 100)
pd.set_option('display.max_columns', 50)
```

```
import pandas as pd
def datetime_tz(tablica):
    tablica.index=pd.to_datetime(tablica.index)
    tablica.index = tablica.index.tz_localize(None)
```

```
datetime_tz(btc)
```

```
# index_col=0- bez index kolone desno od Date-a
transf=pd.read_csv("Transf_roll17.csv", index_col=0)
# Dodavanje naziva index stupcu
transf.index.name='Date'
# primjena prethodno izrađene datetime_tz funkcije
datetime_tz(transf)
```

```
# provjera jesu li tablice istog tipa
type(transf.index),type(btc.index)
```

```
btc_transf["btc_sutra"]=btc_transf["Adj Close"].shift(-1)
btc_transf["izlaz"]=btc_transf["btc_sutra"]>btc_transf["Adj Close"]
btc_transf["izlaz"]=btc_transf["izlaz"].astype(int)
```

```
btc_transf=btc.merge(transf,right_index=True,left_index=True)
```

PRILOG 8

```
# parametri funkcije: df-tablica, X-prediktori
def back_testing(df, X, model, start=1095, korak=30):
    #start definira koliko se podataka prvi put koristi za predviđanje
    predikcije = []
    y_proba_1=[]
    # Prolazak for petlje u koracima, do kraja tablice range(početak,kraj,korak)
    for i in range(start, len(df), korak):
        # Podjela na ulazne(X) i izlazne(y) podatke
        ulaz = df[X]
        izlaz = df["izlaz"]
        #Podjela na trening test split
        X_train = ulaz.iloc[0:i]
        X_test = ulaz.iloc[i:(i+korak)]
        y_train = izlaz.iloc[0:i]
        y_test = izlaz.iloc[i:(i+korak)]

        # Treniranje Klafikacijskog model-a
        model.fit(X_train, y_train)

        ## ZA izračun TPR i FPR
        # potrebno je izdvojiti vjerojatnosti predviđanja za pozitivnu klasu
        y_probs=model.predict_proba(X_test)
        #vjerojatnosti su odvojene u 2 stupca: 1. neg klasa(0), 2. poz klasa (1)
        #za izradu ROC krivulje potrebna je samo poz klasa (2 stupac)
        y_probs_poz=y_probs[:,1]
        y_probs_poz=pd.Series(y_probs_poz, index=X_test.index)
        y_proba_1.append(y_probs_poz)
        y_proba1_df=pd.concat(y_proba_1)

        # mora se pretvoriti iz array tipa u Series tip, inače se
        # ne može spojiti sa stupcem iz tablice(df)
        y_pred = model.predict(X_test)
        y_pred = pd.Series(y_pred, index=X_test.index)

        # Spajanje tablica izlaznih test varijabli i predviđanja,
        # axis=1 jer se spajaju stupci
        spojeno_izlaz_pred= pd.concat({"Izlaz": y_test,"Predviđeno": y_pred},
axis=1)

        predikcije.append(spojeno_izlaz_pred)
        predikcije_df=pd.concat(predikcije)
    # funkcija nakon izvršavanja vraća 2 vrijednosti
    """Returns predikcije_df,y_proba1_df"""

    return predikcije_df,y_proba1_df
```

PRILOG 9

```
from sklearn.metrics
import accuracy_score, precision_score, recall_score, auc, roc_curve
# Funkcija za evaluaciju backtesta
def backtest_eval(backtest_model, y_probs):
    fpr, tpr, thresholds=roc_curve(backtest_model['Izlaz'], y_probs)
    print(f"accuracy_score:
{accuracy_score(backtest_model['Izlaz'], backtest_model['Predviđeno'])*100:.2f}%
")
    print(f"precision_score:
{precision_score(backtest_model['Izlaz'], backtest_model['Predviđeno'])*100:.2f}
%")
    print(f"recall:
{recall_score(backtest_model['Izlaz'], backtest_model['Predviđeno']):.4f}")
```

PRILOG 10

```
import matplotlib.pyplot as plt
%matplotlib inline
# funkcija za prikaz ROC krivulje
def plot_roc_curve(backtest_model, y_probs):
    # dobivanje vrijednosti FPR-a te TPR-a pomoću funkcije roc_curve sklearn
    biblioteke
    fpr, tpr, thresholds=roc_curve(backtest_model['Izlaz'], y_probs)
    #Plot ROC krivulje
    plt.plot(fpr, tpr, color="red", label="ROC")
    #Plot linije bez mogućnosti predviđanja (baseline), linija pogađanja klase
    plt.plot([0,1], [0,1], color="darkblue", linestyle="--", label="granica
odluke")

    # Dodavanje naslova, te naziva pojedine osi, prikaz legende
    plt.xlabel("Stopa lažno pozitivnih (FPR)")
    plt.ylabel("Stopa ispravno pozitivnih (TPR)")
    plt.title("ROC krivulja")
    plt.legend()
```

```

from sklearn.metrics import confusion_matrix
# omogućuje prikaz matrice konfuzije u grafičkom obliku
import seaborn as sns
def conf_matrix(y_test, y_pred):
    cm = confusion_matrix(y_test, y_pred)
    # annot=True- omogućuje prikaz brojeva unutar matrice konfuzije
    # fmt="d" - omogućuje prikaz brojeva u standardnom obliku (zadano:
    eksponencijani oblik)
    sns.heatmap(cm, annot=True, fmt="d")
    plt.title('Matrica konfuzije')
    plt.ylabel('Stvarna klasa')
    plt.xlabel('Predviđena klasa')
    plt.show()

```

```

def eval_ROC_cm(X,model):
    # poziv back_testing funkcije
    backtest,y_probs=back_testing(btc_transf,X,model)
    # prikaz rezultata uspješnosti modela
    backtest_eval(backtest,y_probs)
    #plot ROC krivulje
    plot_roc_curve(backtest,y_probs)
    #plot matrice konfuzije
    conf_matrix(backtest["Izlaz"],backtest["Predviđeno"])

```

PRILOG 11

```

from sklearn.linear_model import LogisticRegression

```

```

X_transf1=['Open', 'High', 'Low', 'Adj Close', 'Volume',
           'broj_tema','avg_rezultata', 'posto_poz', 'posto_neg']
X_btc=['Open', 'High', 'Low', 'Adj Close','Volume']

```

```

solvers = ['newton-cg','lbfgs', 'liblinear']
c_values = [0.01,0.1,1,10,100]
for solver in solvers:
    for val in c_values:
        # definiranje modela
        log_model=LogisticRegression(solver=solver,C=val,random_state=1,n_jobs=-
1)

        print(f"solver={solver}, C={val}")
        #poziv funkcije za backtest
        backtest_log,y_probs=back_testing(btc_transf,X_transf1,log_model)
        #poziv funkcije za evaluaciju
        backtest_eval(backtest_log,y_probs)

```

```
log_model=LogisticRegression(solver='newton-cg',C=0.1,random_state=1,n_jobs=-1)
eval_ROC_cm(X_transf1,log_model)
```

```
for solver in solvers:
    for val in c_values:
        log_model=LogisticRegression(solver=solver,C=val,random_state=1,n_jobs=-1)

        print(f"solver={solver}, C={val}")
        backtest_log,y_probs=back_testing(btc_transf,X_btc,log_model)
```

```
log_model=LogisticRegression(solver='newton-cg',C=1,random_state=1,n_jobs=-1)
eval_ROC_cm(X_btc,log_model)
```

PRILOG 12

```
from xgboost import XGBClassifier
```

```
#zadani hiperparametri n_estimators=100,learning_rate=0.3
n_estimators=[100,200,300]
learn_rates=[0.01,0.03,0.05,0.07,0.1,0.3,0.5]
for est in n_estimators:
    for rate in learn_rates:
        XGB_model = XGBClassifier(n_estimators=est, learning_rate=rate, n_jobs=-1, random_state=1)

        print(f"n_estimators={est}, learning_rate={rate}")
        backtest_XGB,y_probs=back_testing(btc_transf,X_transf1,XGB_model)
        backtest_eval(backtest_XGB,y_probs)
```

```
XGB_model =
XGBClassifier(n_estimators=200,learning_rate=0.3,random_state=1,n_jobs=-1)
eval_ROC_cm(X_transf1,XGB_model)
```

```
n_estimators=[100,200,300]
learn_rates=[0.01,0.03,0.05,0.07,0.1,0.3,0.5]
for est in n_estimators:
    for rate in learn_rates:
        XGB_model = XGBClassifier(n_estimators=est, learning_rate=rate, n_jobs=-1, random_state=1)

        print(f"n_estimators={est}, learning_rate={rate}")
        backtest_XGB,y_probs=back_testing(btc_transf,X_btc,XGB_model)
        backtest_eval(backtest_XGB,y_probs)
```



```
XGB_model = XGBClassifier(n_estimators=200, learning_rate=0.1, n_jobs=-1,
random_state=1)
eval_ROC_cm(X_btc,XGB_model)
```

PRILOG 13

```
from sklearn.svm import SVC
```

```
#https://medium.com/grabngoinfo/support-vector-machine-svm-hyperparameter-
tuning-in-python-a65586289bcb
C_values=[0.01,0.1,1]
kernels=['sigmoid','poly','rbf']
for ker in kernels:
    for c_val in C_values:
        SVC_model=SVC(C=c_val, kernel=ker,probability=True,random_state=1)
        print(f"Kernel:{ker},C:{c_val}")
        backtest_SVC,y_probs=back_testing(btc_transf,X_transf1,SVC_model)
        backtest_eval(backtest_SVC,y_probs)
```

```
SVC_model=SVC(C=0.1,kernel='sigmoid',probability=True,random_state=1)
eval_ROC_cm(X_transf1,SVC_model)
```

```
for ker in kernels:
    for c_val in C_values:
        SVC_model=SVC(C=c_val, kernel=ker,probability=True,random_state=1)
        print(f"Kernel:{ker},C:{c_val}")
        backtest_SVC,y_probs=back_testing(btc_transf,X_btc,SVC_model)
        backtest_eval(backtest_SVC,y_probs)
```

```
SVC_model=SVC(C=0.1,kernel='sigmoid',probability=True,random_state=1)
eval_ROC_cm(X_btc,SVC_model)
```