Zahvaljujem mentoru izv. prof. dr. sc. Vladimiru Čepericu na vodstvu tijekom izrade ovog diplomskog rada.

Zahvaljujem roditeljima i sestri na podršci i razumijevanju koju su mi pružili tijekom studiranja.

Sadržaj

1. Uvod

U ekonomiji portfelj se definira kao skupina od dva ili više financijskih instrumenata koje pojedinac posjeduje. Svaki investitor pokušava izgraditi svoj portfelj financijskih instrumenata takav da maksimizira povrat njegovih ulaganja pri određenom stupnju rizika. Važnu ulogu u njegovoj izgradnji ima sama diversifikacija portfelja, odnosno ulaganje u financijske instrumente vodeći računa o njihovim međusobnim korelacijama u svrhu smanjenja rizika. Međutim, diversifikacijom se rizik ne može u potpunosti otkloniti. Također, jedan od velikih problema osim izgradnje portfelja je upravljanje njime, odnosno koliko sredstava će se alocirati u pojedini instrument u portfelju. U današnje vrijeme ulaganje je postalo većini ljudi dostupno zbog nastanka i razvoja online burzi. Online burze, osim što su privukle velike količine novih ulagača zbog jednostavnosti ulaganja, omogućile su prikupljanje i dostupnost podataka za razne analize što je dovelo do razvoja raznih algoritama za ulaganje. Algoritmi iz strojnog učenja i dubokog učenja počeli su se sve više primjenjivati u financijama i problemu optimizacije portfelja.

Ovaj diplomski rad se sastoji od dva dijela. U prvom dijelu je opisana izgradnja portfelja dionica pomoću konvolucijskog autoenkodera i teorije grafova. Portfelj se gradi na način da se nakon svakog određenog perioda biraju dionice koje će činiti portfelj, odnosno od dionica koje pripadaju s&p 500 biramo podskup dionica. U drugom dijelu rada opisano je upravljanje izabranog portfelja dionica koristeći potporno učenje i arhitekturu Transformer-a.

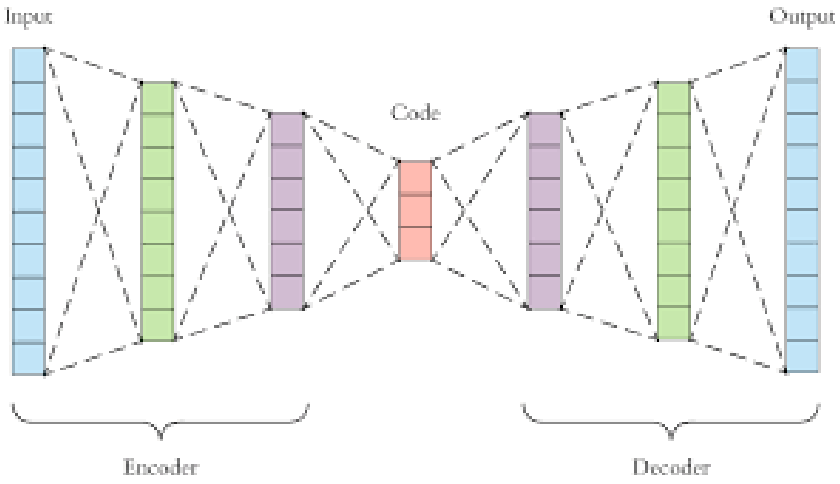
2. Konstrukcija portfelja

Nobelovac Harry Markowitz 1952. godine razvio je modernu teoriju portfelja [16]. Moderna teorija portfelja za glavnu ideju ima diversificirati rizik na način da se konstruira portfelj financijskih instrumenata takav da maksimizira povrat s obzirom na količinu rizika. Kako bi se što bolje diversificirao rizik najbitnije je odrediti koji se financijski instrumenti ponašaju slično. Prema vjerovanju raznih studija dionice koje imaju sličnu promjenu cijene ili povrata biti će visoko korelirane i smatra se da se kreću u istom smjeru, dok one koje su negativno korelirane imaju različiti smjer kretanja. Kako bi odredili kreću li se cijene dionica u istom smjeru koriste se različite mjere korelacije. Problem kod korištenja općenitih mjera korelacija je u tome što prilikom izračuna korelacija pretpostavlja se da su podaci međusobno neovisni što kod vremenskih nizova nije slučaj, jer svaki podatak ovisi o prošlim podacima. Problem s autokorelacijom vremenskih nizova riješen je pretvaranjem vrijednosti dionica u graf sa svijećama (engl. *candelstick chart*) [1]. Svaka svijeća napravljena je korištenjem vrijednosti cijene otvaranja, najviše cijene koju je postigla dionica u promatranom vremenskom periodu, najnižu cijenu u istom periodu i cijenu koju je dionica imala na kraju promatranog vremenskog perioda. Kako bi iz tako generiranog grafa izvukli značajke za svaku pojedinu dionicu koristimo konvolucijski autoenkoder. Konvolucijski autoenkoder kao ulazne vrijednosti prima sliku generiranog grafa sa svijećama (engl. *candelstick chart*), a na izlazu daje sažete najbitnije značajke za pojedinu dionicu.

2.1. Autoenkoder

Autoenkoder je posebna vrsta neuronskih mreža koja se sastoji od kodera i dekodera [2]. Uloga kodera je smanjivanje dimenzionalnosti ulaznih vrijednosti u kod manjih dimanzija, npr. slike u kod, a uloga dekodera je rekonstrukcija ulaznih vrijednosti iz tog koda. Učenje ovakvog tipa neuronske mreže se radi na način da izaberemo funkciju pogreške sličnosti koja će određivati sličnost između izlaznih vrijednosti dekodera, tj. rekonstrukcije i originalnih ulaza. Nakon što istreniramo autoenkoder, ukoliko nas zanima izlaz iz latentnog sloja, tj. sami kod dekoder možemo maknuti.

Izlaz kodera se može koristiti u svrhu kompresije podataka ili za smanjivanje dimenzionalnosti, jer se unutar koda nalaze najbitnije značajke. Postoje razne vrste autoenkodera poput: translatirajućih autoenkodera, autoenkodera za uklanjanje šuma, varijacijski autonekodera i konvolucijskih autoenkodera koji će detaljnije biti opisani u idućem poglavlju [2].



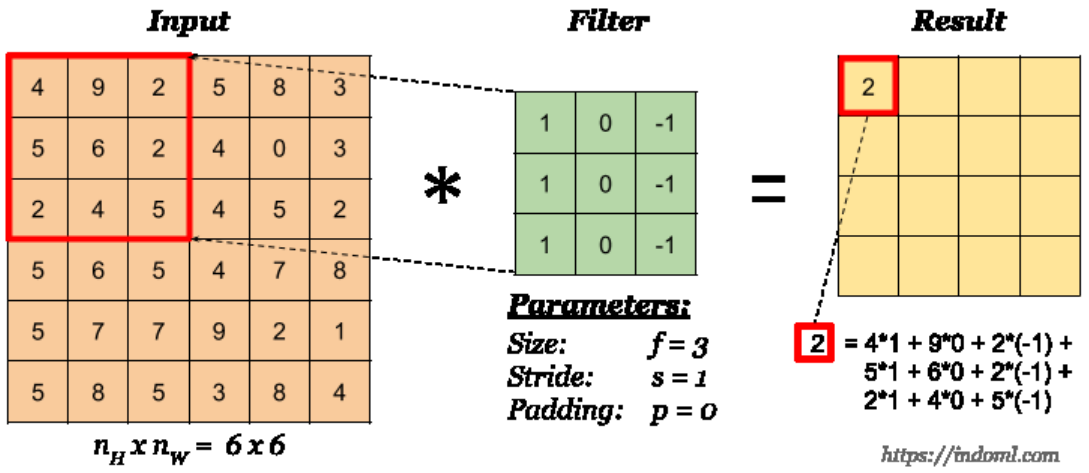
Slika 1. Shematski prikaz arhitekture autoenkodera [3]

2.2. Konvolucijski autoenkoder

Konvolucijski autonekoder sastoji se od konvolucijskih neuronskih slojeva [4]. Svaki konvolucijski autoenkoder na ulazu u koder prima sliku i sažima je u latentni sloj (kod) što kasnije služi kao ulaz u dekoder koji rekonstruira sliku. Ovakva vrsta autoenkodera kao gradivne jedinice kodera i dekodera sadrži konvolucijske slojeve (engl. *convolutional layer*) i slojeve sažimanja (engl. *pooling layer*).

2.2.1. Konvolucijski sloj

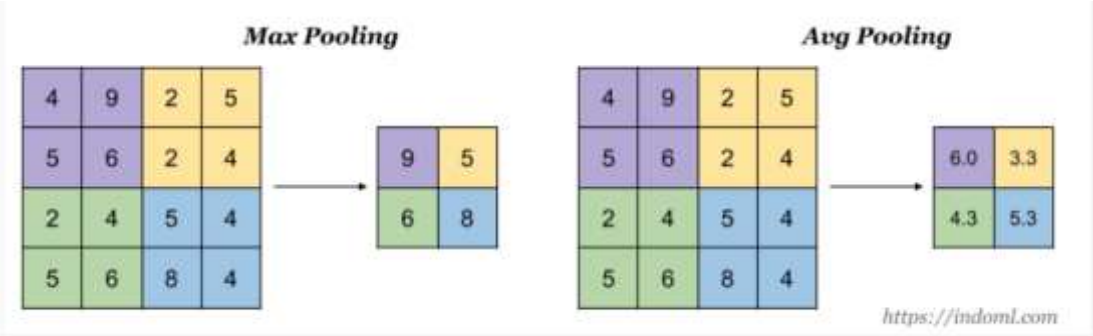
Svaki konvolucijski sloj je definiran jezgrom (engl. *kernel*). Konvolucija se provodi tako da s određenim pomakom pomičemo jezgru po ulaznoj mapi značajki (engl. *feature map*) i na taj način dobivamo izlaznu mapu značajki (slika 2.). U konvolucijskom sloju dolazi do pronalaženja bitnih značajki slike [4].



Slika 2. Ilustracija konvolucije [5]

2.2.2. Sloj sažimanja

Slojevi sažimanja koriste se za smanjivanje dimenzija mapi značajki (engl. *feature map*), a koriste se i za izdvajanje bitnih značajki u pojedinoj regiji ulazne mape. Postoji više vrsta načina sažimanja, a najčešće se koristi maksimalno sažimanje (engl. *max pooling*) i sažimanje srednjom vrijednošću (engl. *average pooling*). Sažimanje maksimalnom vrijednošću se provodi na način da ulaznu mapu značajki podijelimo na određeni broj regija i onda unutar svake regije tražimo najveću vrijednost (slika 3.). Sažimanje srednjom vrijednošću provodi se na isti način, ali se umjesto maksimalne vrijednosti uzima srednja vrijednost (slika 3.) [4].

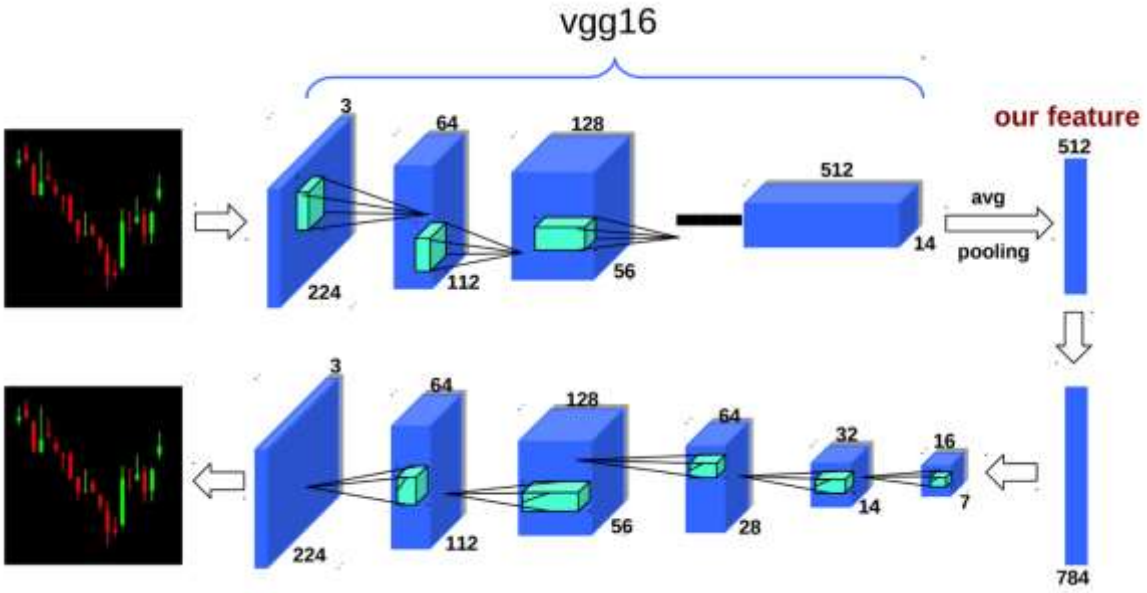


Slika 3. Ilustracija sloja sažimanja maksimalnom vrijednošću [6]

2.2.3. Koder i dekoder

Arhitektura kodera je VGG16 [7] koja je pokazala jako dobre rezultate u vizualnom prepoznavanju (engl. *visual recognition*). Prema [1] korištena je ista arhitektura uz promjenu da je zadnji linearni sloj zamijenjen slojem sažimanja srednjom vrijednošću čiji izlaz sadrži najbitnije značajke. Izlaz iz kodera koristit ćemo kao značajke dionice.

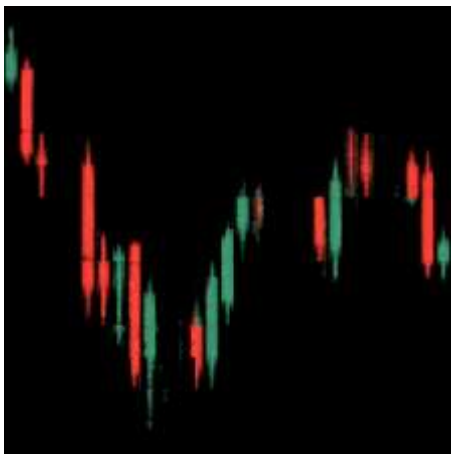
Arhitektura dekodera je ista kao arhitektura kodera samo što izlaz iz kodera prvo prolazi kroz linearni sloj, a zatim radimo dekonvoluciju kako bi dobili rekonstruiranu sliku.



Slika 4. Arhitektura konvolucijsko autoenkodera [1]

Konvolucijski autoenkoder se trenira koristeći MSE funkciju gubitka između originalne slike (slika 5.) i rekonsturirane slike (slika 6.).

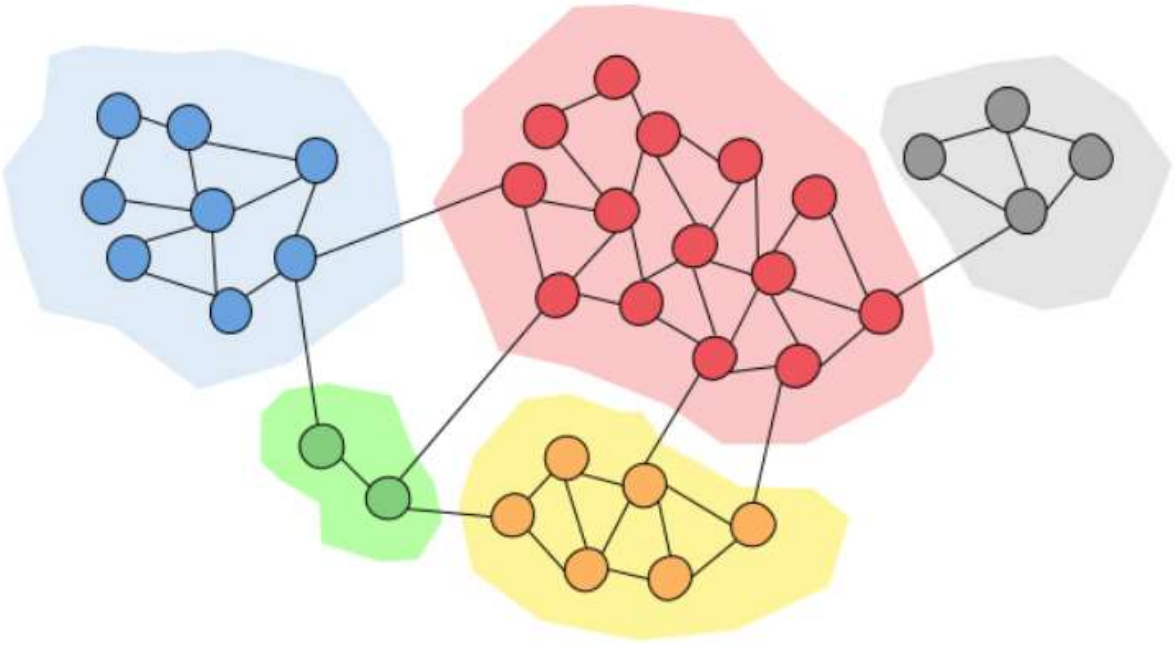
Slika 5. Originalna slika grafa sa svijećama



Slika 6. Rekonstruirana slika grafa sa svijećama

2.3. Otkrivanje zajednica dionica

Kako bi odredili korelirane zajednice dionica, potrebno je izgraditi potpuno povezani otežani graf. Graf se sastoji od vrhova i bridova [8]. Vrhovi predstavljaju dionice. Svaki vrh sadrži značajke dionica koje su izašle iz kodera, dok su bridovi poveznica između dviju dionica. Svaki brid ima težinu čija je vrijednost mjera kosinusne sličnosti između značajki dviju dionica koje povezuje. Svaki vrh je određen stupnjem, tj. brojem koji predstavlja s koliko je drugih vrhova taj vrh povezan. Grafički se vrhovi prikazuju kao kružići, a bridovi kao crte između ta dva kružića. Nad tako izgrađenim grafom, mogu se provoditi različiti algoritmi za otkrivanje zajednica (slika 7.).



Slika 7. Shematski prikay otkrivanje zajednica u grafu [9]

2.3.1. Leiden-ov algoritam

Otkrivanje zajednica u grafu je problem koji se najčešće koristi kako bi se razumjela struktura velikih i kompleksnih mreža. Jedan od takvih algoritama za otkrivanje struktura zajednica unutar grafa je Leidenov algoritam.
Leidenov algoritam [10] (slika 8.) kroz tri faze u svakoj iteraciji pokušava optimizirati funkciju kvalitete. Modularnost je funkcija kvalitete kojom se pokušava maksimizirati razlika između stvarnog broja bridova u zajednici i očekivanog broja bridova. Za maksimiziranje modularnosti se koristi formula:

$$Q = \frac{1}{2m} \sum_c \left(e_c - \gamma \frac{K_c^2}{2m} \right) \qquad (2.1.)$$

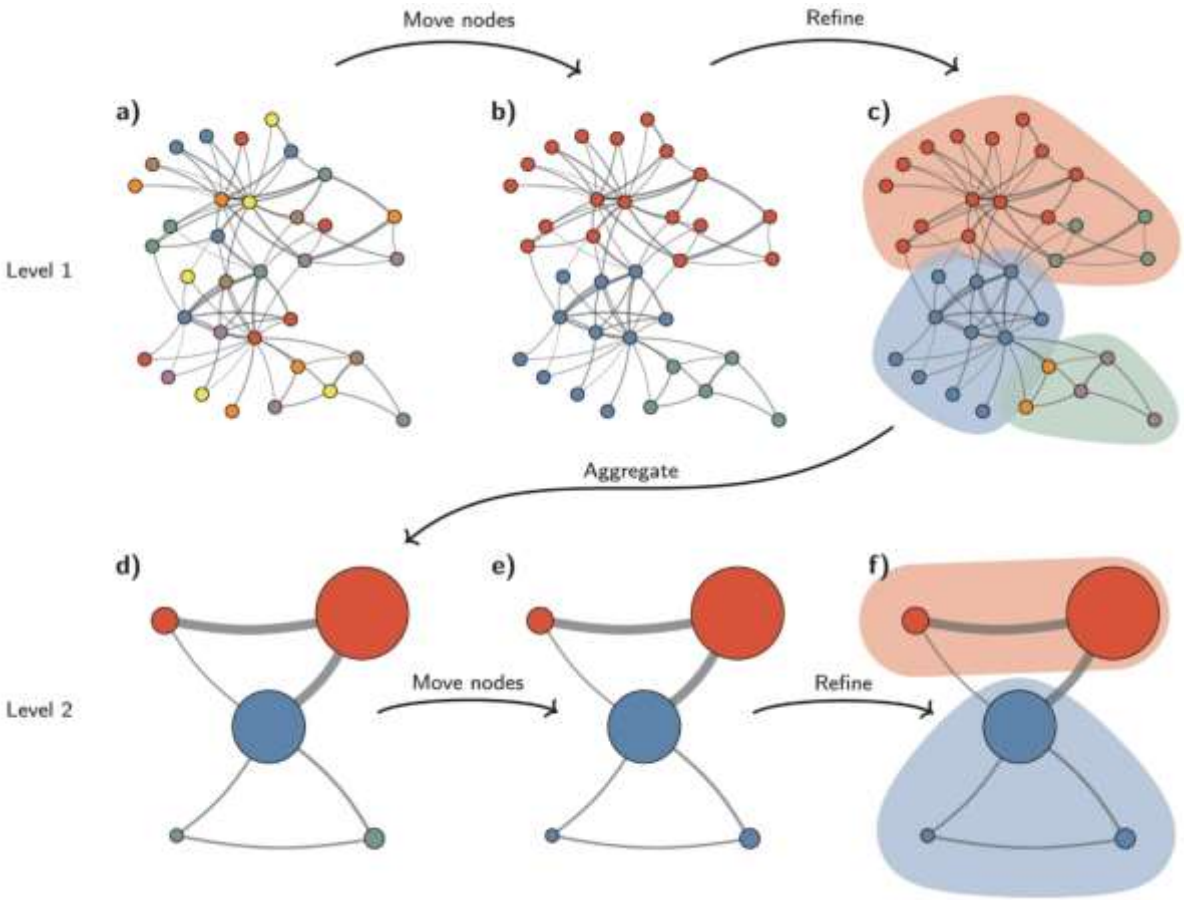
e_c predstavlja broj bridova unutar zajednice c, očekivani broj bridova u zajednici dobije se umnoškom rezolucijskog parametra (γ) i kvadrirane sume stupnjeva svakog vrha u zajednici (K_c^2), a zatim se taj izraz dijeli s dvostrukim brojem ukupnog broja bridova (2m) u cijelom grafu. Rezolucijskim parametrom se kontrolira broj zajednica, a što je vrijednost parametra veća dobiva se veći broj zajednica, a što je vrijednost parametra manja dobiva se manji broj zajednica.

Prva faza u Leidenovom algoritmu je lokalno premještanje vrhova. Premještanje vrhova se radi na način da vrh iz jedne zajednice premjestimo u drugu zajednicu, a pritom maksimalno poboljšavamo modularnost. Nakon što je vrh premješten iz jedne zajednice u drugu, sve vrhove koji su povezani s premještenim vrhom, a pritom se ne nalaze u novoj zajednici premještenog vrha, pokušavamo premjestiti [10].

Druga faza algoritma je poboljšavanje zajednica. U ovoj fazi unutar svake zajednice pokušavamo naći postojanje manjih zajednica. Faza počinje tako da svaki vrh unutar zajednice je sam po sebi zajednica i onda iz takvih

zajednica se rade veće zajednice na način da nasumično povežemo dvije zajednice koje će poboljšati funkciju kvalitete u jednu. Faza završava kada više ne postoje zajednice od jednog člana [10].

Treća i posljednja faza je faza agregacije grafa u kojoj svaka zajednica iz prošle faze postaje vrh. Algoritam se izvršava sve dok se mogu izvršiti poboljšanja funkcije kvalitete. Izlaz Leidenovog algoritma su visoko povezane zajednice [10].



Slika 8. a) zajednice od jednog člana, b) veće zajednice nastale lokalnim premještanjem vrhova u zajednice, c) pronalaženje manjih zajednica unutar zajednice, d) agregacija zajednica u vrhove, e) pomicanje vrhova radi stvaranja novih zajednica (nisu nastale nove zajednice), f) ponavljanje navedenih koraka sve dok se može napraviti neko poboljšanje [10]

2.4. Biranje dionica portfelja

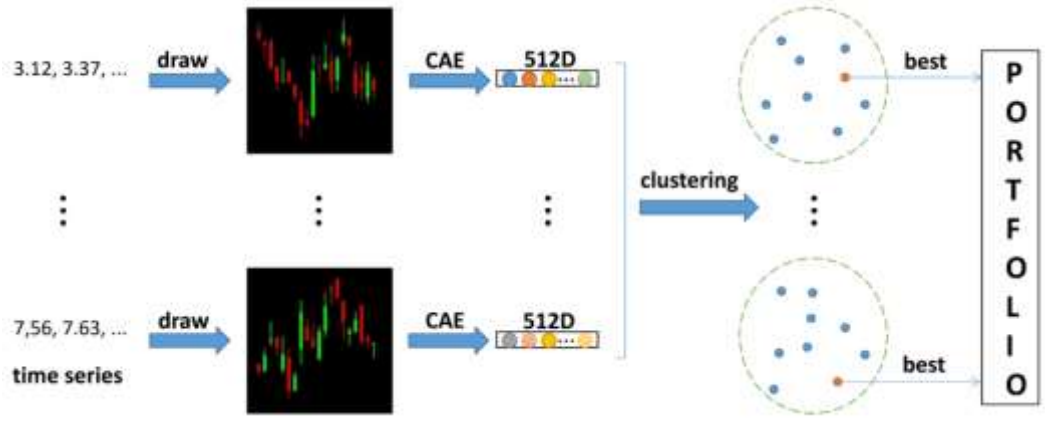
Nakon što smo otkrili zajednice unutar grafa, kako bi konstruirali naš portfelj iz svake zajednice bira se određeni broj dionica. Broj dionica koje se uzimaju iz svake zajednica je određen ukupnim brojem dionica koje čine naš portfelj podijeljen s brojem zajednica u tom trenutku. Iz svake zajednice izaberu se dionice s najboljim Sharpe omjerom.

Sharpe omjer je mjera koja u omjer stavlja prosječan prinos iznad nerizične kamatne stope i standardnu devijaciju prinosa u promatranom vremenskom periodu kao što je prikazano formulom 2.2 [11].

$$SR = \frac{r_p - r_F}{\sigma_P} \qquad (2.2.)$$

U formuli 2.2, SR predstavlja Sharpe omjer, r_p prinos portfelja, r_f nerizičnu kamatnu stopu, a σ_P standardnu devijaciju prinosa portfelja.

Dionice s višim Sharpe omjerom imaju povoljniji odnos prinosa i rizika. Ukoliko u portfelj nije izabrano dovoljno dionica, od svih ne izabranih dionica biraju se one s najboljim Sharpe omjerom.



Slika 9. Potpuni shematski prikaz procesa konstrukcije porfelja [1]

U idućem poglavlju, opisan je način raspodijele imovine i upravljanje portfeljom.

3. Upravljanje portfeljom

Upravljanje portfeljom je proces donošenja odluka o kontinuiranoj preraspodjeli kapitala u određeni broj izabranih financijskih instrumenta. Tradicinalne metode upravljanja portfeljom su ”*Follow-the-Winner*”, ”*Follow-the-Loser*” i ”*Pattern-Matching*” [12].

U metodi ”*Follow-the-Winner*” sredstva se iz financijskih instrumenata koji imaju loše rezultate redistribuiraju u financijske instrumente koji imaju bolje rezultate, jer se vjeruje da će ti instrumenti nastaviti donositi dobre rezultate [12].

U metodi ”*Follow-the-Loser*” sredstva se iz financijskih instrumenata koji imaju bolje rezultate redistibuiraju u financijske instrumente koji imaju lošije rezultate, jer se vjeruje da će instrumenti koji su imali lošiji period napraviti obrat i početi donositi dobre rezultate [12].

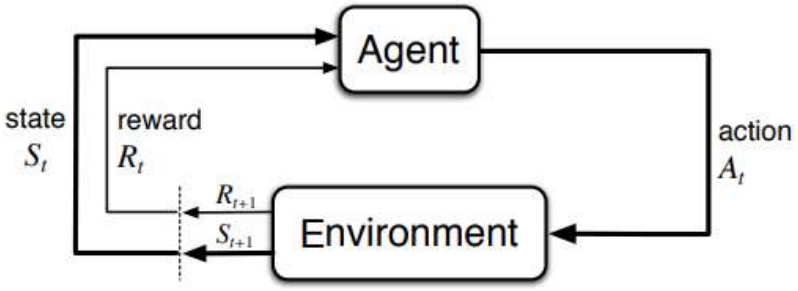
U metodi ”*Pattern-Matching*” zbog uvjerenja da se isti uzorci ponavljaju, traže se slični uzorci u povijesnim podacima i prema tome dolazi do redistribucije imovine [12].

Komplicirani, nelinearni uzorci i količine šuma u podacima te sama dinamika financijskih tržišta neki su od razloga zbog kojih algoritmi dubokog potpornog učenja postaju sve popularniji. Algoritmi dubokog potpornog učenja su pokazali dobre rezultate u drugim dinamičnim područjima kao što su robotika, autonomna vožnja i igranje igara.

3.1. Duboko potporno učenje

Duboko potporno učenje je dio dubokog učenja temelji se na izgradnji i treniranju agenta koji se u interakciji s okolinom postepeno pobošljava sa strategijom pokušaja i pogreške. Okruženje u kojem se agent nalazi predstavlja se kao Markovljev proces odlučivanja [13].

U svakom trenutku t agent promatra trenutno stanje okruženja S_t i primjećene informacije koristi kako bi napravio akciju A_t kojom će preći u stanje S_{t+1} i pritom dobiti nagradu R_{t+1} za napravljenu akciju [13].



Slika 10. Dijagram modela Markovljevog proces odlučivanja [14]

Cilj algoritma potpornog učenja je odabrati optimalnu politiku (engl. *policy*) koja maksimizira očekivanu sniženu kumulativnu nagradu koja se još naziva i povrat. Snižena kumulativna nagrada se računa prema formuli u kojoj je faktor sniženja označen γ.

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \qquad (3.1.)$$

Optimalnu politiku koju agent pokušava naučiti označavamo izrazom π^*. Optimalna politika uvijek postoji, ali ne mora nužno biti jedinstvena, a predstavljena je formulom:

$$\pi^*(s) = arg_\pi \max E_\pi \, [G_t] \qquad (3.2.)$$

U svakom stanju ***s*** agent bira akciju ***a*** prema politici i za nju dobiva povrat čija je vrijednost definirana formulom:

$$q_\pi(s,a) = E_\pi [G_t \mid S_t = s, A_t = a] \qquad (3.3.)$$

U svakom stanju ***s*** bira se akcija koja maksimizira formulu 3.3 i time je optimalna politika je definirana izrazom:

$$\pi^* (s) = arg_a \max q_* (s,a) \qquad (3.4.)$$

U svakom trenutku okruženje koje agent promatra su vrijednosti financijskih instrumenata koje se trenutno nalaze u portfelju. Trenutno stanje okruženja se sastoji od matrice vrijednosti portfelja financijskih instrumenata čije su dimenizije $k\ X\ m\ X\ d$, gdje ***m*** predstavlja broj dionica koji se nalazi u portfelju, ***n*** predstavlja koliko vrijednosti prethodnih vremenskih perioda će naše trenutno stanje sadržavati i ***d*** predstavlja broj značajki financijskog instrumenta koje uzimamo.

Značajke financijskog instrumenta koje uzimamo su najviša cijena, najniža cijena, cijena koju je dionica imala prilikom zatvaranja u tom vremenskom periodu. Sve navedene vrijednosti su normalizirane prema cijeni zatvaranja iz proslog vremenskog perioda, dok je cijena otvaranja normalizirana prema cijeni otvaranja iz prošlog vremenskog perioda. Osim opisane matrice vrijednosti stanju pripada i vektor težina portfelja iz prošlog perioda. Vektor težina portfelja $\boldsymbol{a}_t$ je akcija koju agent generirara za svaki idući period i sastoji se od ***m+1*** vrijednosti pri čemu svaka vrijednost predstavlja koliko bi naših sredstava trebalo biti uloženo u određeni financijski instrument i jedne vrijednosti koja predstavlja koliko bi sredstava trebali ostaviti neuloženo [13].

Nagradu za svaku akciju koju je naš agent generirao računamo prema formuli:

$$R_t = \ln(\boldsymbol{a}_t^{\mathrm{T}} \boldsymbol{y}_t\ (1 - c_t)) \qquad (3.5.)$$

U formuli se uzima logaritam vrijednosti umnoška cijene zatvaranja y_t, vektora težina portfelja i transakcijskog troška c_t.

Postoje različite implementacije arhitekture agenta u dubokom potpornom učenju, a u ovom radu implementirana je arhitektura Transformera [15].

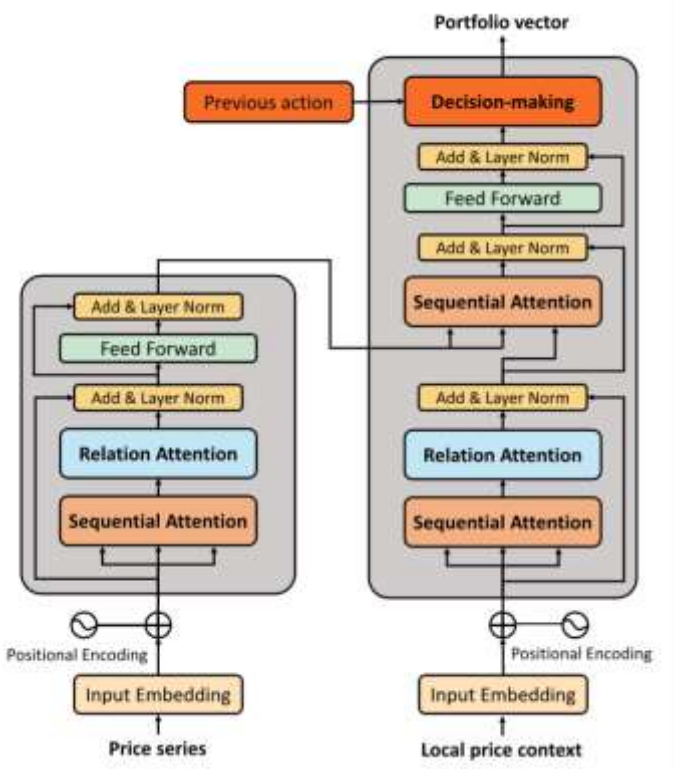
3.2. Arhitektura agenta

U novije vrijeme, Transformer, koji se oslanja na mehanizam pažnje (engl. *attention*), pokazao je veliki potencijal u shvaćanju sekvencijalnih uzoraka s vrlo dugotrajnim ovisnostima .

Arhitektura Transformera se sastoji od enkodera i dekodera koji koriste višeglavnu pažnju (engl. *multi-headed attention*) [15]. Međutim, originalna arhitektura Transformera se ne može izravno primijeniti na upravljanje portfeljom iz dva razloga.
Prvi razlog zbog kojega se Transformer arhitektura ne može izravno primjeniti je zbog toga što se vrijednosti u slojevima višeglavne pažnje računaju na temelju pojedinih vrijednosti, a to ga čini nesposobnim za hvatanje informacija lokalnog konteksta i lako ga je zbuniti lokalnim vrijednostima koje sadrže šum. Štoviše, kako je Transformer prvenstveno dizajniran za prijevod sekvenci riječi u području prirodne obrade jezika, nije u stanju uhvatiti korelacije imovine koje su korisne u donošenju odluka o portfelju. Zbog navedenih problema arhitektura originalnog Transformera je izmijenjena [16].

Arhitektura izmijenjenog Transformera (slika 11.) se također sastoji od enkodera i dekodera. Enkoder se koristi za ekstrakciju sekvencijalnih značajki i na ulazu prima trenutni niz cijena. Dekoder na ulazu prima lokalni kontekst cijena, dok izlaz dekodera je odluka, tj. vektor težina portfelja. Promjena napravljena u enkoderu i dekoderu je ta da se umjesto obične

višeglavne pažnje koristi sekvencijonalna višeglavna pažnja i relacijska višeglavna pažnja te je promjenjen izlaz dekodera. Ulazne vrijednosti prije nego što uđu u enkoder i dekoder kao i u originalnoj arhitekturi prolaze kroz pozicijsko enkodiranje [16].



Slika 11. Arhitektura izmjenjenog Transformer-a [16]

3.2.1. Pozicijsko enkodiranje

Pozicijsko enkodiranje se koristi kako bi model mogao dobiti više informacija o poziciji i rasporedu ulaznih vrijednosti u sekvenci. Ulazne vrijednosti prvo prolaze kroz linearni sloj nakon kojeg dobivamo vektore vrijednosti i potom provodimo pozicijsko enkodiranje. Pozicijsko enkodiranje se provodi na način da se svakom ulaznom vektoru ovisno o njegovoj poziciji u ulaznoj sekvenciji nadoda vrijednost sinusa na svakoj parnoj poziciji unutar vektora izračunatoj prema formuli 3.6. ili kosinusa ukoliko je pozicija unutar vektora neparna izračuna prema formuli 3.7 [15].

$$PE_{pos,2i} = \sin\left(\frac{pos}{10000^{2i/dm}}\right) \qquad (3.6.)$$

$$PE_{pos,2i+1} = \cos\left(\frac{pos}{10000^{2i/dm}}\right) \qquad (3.7.)$$

3.2.2. Pažnja

Slojevi pažnje za svaku ulaznu vrijednost daju izlazne vrijednosti jednake veličine, a služe kako bi se odredila važnost podatka u trenutnom kontekstu na način da se promatrani podatak uspoređuje s drugim podacima u sekvenci. Usporedba između elemenata se radi koristeći skalarni produkt i rezultat tih usporedbi se naziva ocjena (engl. *score*). Koristeći samo originalne ulazne vrijednosti, nije moguće odrediti kako bi pojedini ulazi mogli doprinijeti u duljim sekvencama. Kako bi se riješio ovaj problem, u slojevima pažnje uvode se tri nove matrice težine gdje se težine uče prilikom treniranja. Svaka matrica koristi za linearnu transformciju ulaznih podataka u rezultirajuće vrijednosti pojedine uloge [15].

Upit (engl. *query*) je jedna od uloga koja predstavlja trenutni fokus pažnje dok se uspoređuje sa svim prethodnima ulazima.
Ključ (engl. *key*) je jedna od uloga koja predstavlja vrijednost prethodnog ulaza s kojom se uspoređuje trenutni fokus pažnje.
Vrijednost (engl. *value*) je uloga koja se koristi za računanje izlaza trenutnog fokusa.
Vektor vrijednosti svake uloge za pojedini ulazni vektor dobivamo koristeći iduće formule.

$$q_i = W^Q x_i \qquad k_i = W^K x_i \qquad v_i = W^V x_i \qquad (3.8.)$$

Ocjenu usporedbe između vektora trenutnog fokusa x_i i vektora iz prethodnog konteksta x_j je sastavljena od vektorskog umnoška trenutnog vektora upita q_i i vektora ključa k_j iz prethodnog konteksta.

$$score\left(x_i, x_j\right) = q_i \cdot k_j \qquad (3.9.)$$

Kako bi dobili izlazni vektor težina α_{ij} koje prikazuju proporcionalnu značajnost na ocjene sličnosti je potrebno primjeniti funkciju softmax:

$$\alpha_{ij} = softmax\left(score\left(x_i, x_j\right)\right) \qquad \forall j \leq i \qquad (3.10.)$$

$$= \frac{\exp\left(score\left(x_i, x_j\right)\right)}{\sum_{k=1}^{i} \exp\left(score(x_i, x_k)\right)} \qquad \forall j \leq i$$

Kako bi dobili konačne izlazne vrijednosti y_i koristi se formula:

$$y_i = \sum_{j \leq i} \alpha_{ij} x_j \qquad (3.11.)$$

Za izračunavanje pojedinih elemenata u vektoru ocjene koristi se skalarni produkt čije vrijednosti mogu biti proizvoljno velike što može uzrokavati numeričke probleme prilikom korištenja softmax funkcije (eksponenciranje velikih negativnih vrijednosti rezultiraju 0, a pozitivnih prevelikim brojem). Kako bi se ovaj problem izbjegao potrebno je modificirati formulu tako da rezultat skalarnog produkta podijelimo s korijenom veličine dimenzije matrice ključa [15].

Opisani proces koristi se za izračunavanje jednog izlaz u pojedinom trenutku, ali kako se svi izlazi mogu računati nezavsno cijeli proces se može poboljšati i paralelizirati koristeći formule:

$$Q = W^Q X \qquad K = W^K X \quad V = W^V X \qquad (3.12.)$$

U formulama dolazi do grupiranja pojedinih vektora ulaza u matricu **X**.
Koristeći ovako dobivene matrice **Q**, **K**, **V** izlaz iz sloja pažnje možemo dobiti istovremeno koristeći formulu:

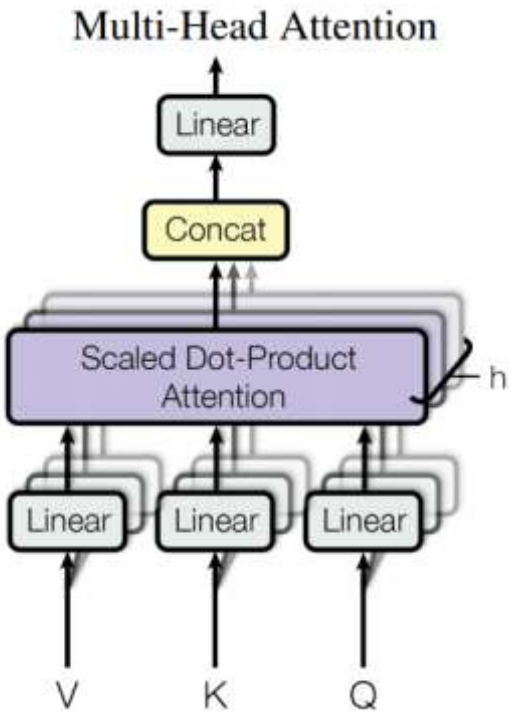
$$Attention\,(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad (3.13.)$$

Razlika između pažnje i samopažnje je u tome što se u pažnji matrica upita može dobiti iz različite sekvence od one iz koje su dobivene matrice ključa i vrijednosti, dok kod samopažnje sve tri matrice su dobivene iz iste sekvence.

3.2.3. Višeglavna pažnja

Pojedini ulazi u sekvenci mogu imati više različitih odnosa odjednom i zbog toga je teško da jedan sloj pažnje nauči sve različite vrste tih odnosa. Ovaj problem se rješava na način da se uvedu slojevi višeglavne pažnje (engl. *multihead attention*).

Sloj višeglavne pažnje (slika 12.) se sastoji od skupa slojeva pažnje, gdje se svaki takav sloj pažnje naziva glava (engl. *head*). Sve glave se nalaze na istoj dubini modela i svaka sadrži svoje parametre težina. Originalne matrice upita, ključa i vrijednosti dijele se ovisno o broju glava, a na taj način svaka glava dobiva svoju matricu upita, ključa i vrijednosti s kojima izvršava operacije. Izlazi svih glava se nakraju povežu konkatenacijom i potom na tako konkatenirani izlaz primjenimo linearni sloj kako bi dobili originalnu dimenziju izlaza. Kako bi bolje iskoristili infomarcije koje nam daju sam kontekst sekvenci i odnosi u portfelju napravljeni su modificirani slojevi višeglavne pažnje [15].



Slika 12. Vizualni prikaz višeglavne pažnje [15]

3.2.4. Sekvencijalna višeglavna pažnja

Kako bi vrijednosti upita i ključeva napravili što otpornijima na šum u lokalnim nizovima cijena umjesto samo trenutne cijene $P_t \in R^{k\,x\,m\,x\,d}$ kao ulazne vrijednosti koristimo lokalni kontekst cijena $P_t^l \in R^{l\,x\,m\,x\,d}$ čija je veličina definirana vrijednošću *l*. Vrijednosti matrice ključa (**K**) i upita (**Q**) dobivamo tako da radimo 2D konvoluciju nad ulaznim lokalnim kontektsom cijena, a vrijednosti matrice vrijednosti (**V**) dobivamo tako da na ulazni kontekst cijena primjenimo linearni sloj. Izlaz iz ovog sloja se dobiva korištenjem formule 3.13. Ukoliko bi vrijednost *l* iznosila 1, sekvencijalna višeglavna pažnja se degradira u standardnu višeglavnu pažnju [16].

3.2.5. Relacijska višeglavna pažnja

Kako bi vrijednosti nakon sekvencijalne višeglavne pažnje dodatno obogatili korelacijama između financijskih instrumenata u portfelju, na izlazne vrijednosti iz sloja sekvencijalne višeglavne pažnje potrebno je provesti još jednu višeglavnu pažnju [16].

3.2.6. Sloj donošenja odluke

Zadnja promjena koju je potrebno napraviti kako bi arhitektura Transformera bila primjenjiva na naš problem je dodati sloj donošenja odluke. Ulaz u sloj donošenja odluke je izlaz iz dekodera konkateniran s vektorom težina portfelja iz prošlog vremenskog perioda. Sloju se predaje vektor težine portfelja iz prošlog vremenskog perioda kako bi spriječili model da ne radi prevelike promjene u težinama, a time se smanjuje transakcijski trošak. Sloj donošenja odluke se sastoji od tri linearna sloja. Izlaz iz prvog linearnog sloja predstavlja inicijalni vektor težina portfelja $\hat{a}_t$, izlaz iz drugog linearnog sloja se naziva vektor kratke prodaje (engl. *short sale vector*) $\hat{a}_t^s$, a izlaz iz trećeg linearnog sloja se naziva vektor reinvestiranja $\hat{a}_t^r$ [16].

Vektor kratke prodaje služi kako bi model imao mogućnost zarade ukoliko cijena nekog instrumenta krene gubiti na vrijednosti. Nad izlazom svakog od linearnih slojeva se primjenjuje funkcija softmax, a zatim se konačni vektor težina portfelja izračunava prema formuli:

$$a_t = \hat{a}_t - \hat{a}_t^s + \hat{a}_t^r \qquad (3.14.)$$

Težine unutar portfelja poprimaju vrijednosti između $(-1,2)$, a pritom mora vrijediti:

$$\sum_{i=1}^{m} a_t, i = 1 \qquad (3.15.)$$

4. Rezultati

Za dobivanje rezultata korišteni su vremenski nizovi cijena 250 dionica koje pripadaju *Standard and Poors 500* (S&P 500) indeksu. Za treniranje, validaciju i testiranje koristili su se podaci iz vremenskog razdobolja od 1.1.2005. do 17.5.2022.

4.1. Implementacija

Sustav je implementiran u programskom jeziku *Python* i korišteni su vanjski paketi *pandas, numpy, igraph, PyTorch* i *empyrical*. Sustav je organiziran u 4 klase. Prva klasa je Environment koja se koristi za učitavanje i obradu podataka. Druga klasa je PortfolioConstruct. U ovoj klasi dolazi do kreiranja slika grafa sa svijećama koje se potom dovode na ulaz konvolucijskog autonekodera nakon čega se prema kodiranim značajkama kreira graf i koristeći Leidenov algoritam dobivamo zajednice iz kojih se biraju dionice koje će činiti naš portfelj. Treća klasa je RelationAwareTransformer u kojoj se

nalazi model pomoću kojeg upravljamo portfeljom. Četvrta klasa je Backtest u kojoj se računaju povrati i koliko je model loše donosio odluke i nad tim vrijednostima se primjenjuje funkcija gubitka po kojoj se model trenira. Za treniranje konvolucijskog autoenkodera i transformera koristio se *Adam* optimizator.

4.2. Hiperparametri

Hiperparametri i njihove vrijednosti koji su se koristili za generiranje rezultata prikazani su u tablici 1.

Tablica 1. Popis hiperarametara i njihovih vrijednosti

Hiperparametri	**Vrijednost**	**Opis**
resolution_parametar	0,8	Rezolucijski parametar u Leiden-ovom algoritmu
n_iteration_leiden	5	Broj iteracija u Leiden-ovom algoritmu
n_candles_image	20	Broj svijeća na grafu svijeća (engl. *candlestick chart*)
cae_encoder_output_size	512	Veličina izlaznog sloja kodera u konvolucijskom autoenkoderu
cae_encoder_input_linear_size	784	Veličina ulaznog sloja dekodera u konvolucijskom autoenkoderu
portfolio_stock_number	25	Broj dionica koji čine portifelj
n_trading_days	20	Broj dana nakon kojih se ponovno radi konstruiranje portfelja

n_selected_features	4	Broj ulaznih vrijednosti u model
trading_period	1d	Vremenski period između ulaganja
window_size	30	Broj dana od kojih se sastoji svaka ulazna vrijednost u model
batch_size	128	Broj uzoraka tijekom treniranja modela potpornog učenja
total_steps	25 000	Broj koraka tijekom treniranja modela potpornog učenja
commision_rate	0,25%	Stopa provizije koja se primjenjuje na svaku transakciju
learning_rate	1×10^{-5}	Stopa učenja tijekom treniranja modela potpornog učenja
learning_rate_cae	1×10^{-3}	Stopa učenja tijekom treniranja konvolucijskog autoenkodera
local_context_length	7	Veličina lokalnog konteksta u slojevima sekvencijalne pažnje
n_attention_heads	3	Broj glava u slojevima pažnje
model_dim_encoder_decoder	24	Veličina enkodera i veličina dekodera u Transformeru
buffer_bias_ratio	4×10^{-5}	Parametar geometrijske razdiobe pri odabiru uzoraka

4.3. Povrati i usporedba

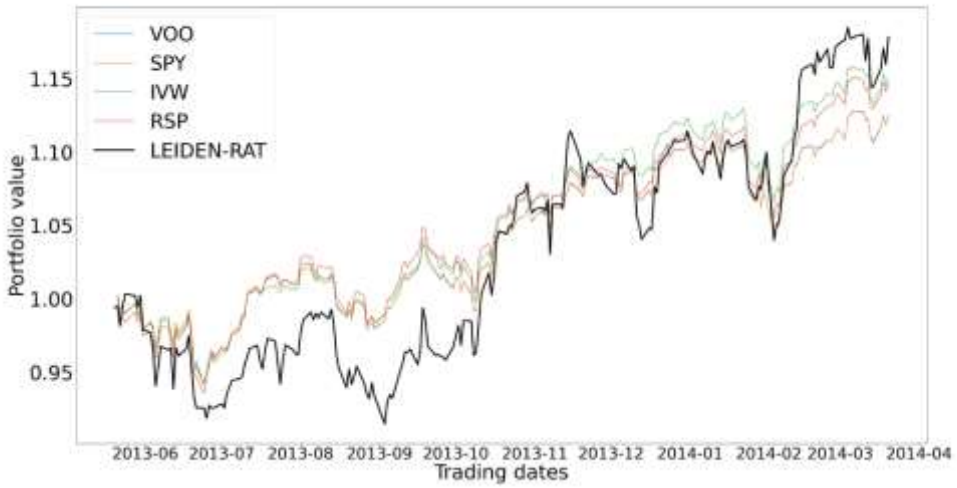
Sustav koji je opisan u prošlim poglavljima testiran je na način da su izabrana 4 vremenska perioda u kojima su podaci bili podijeljeni na skup podataka za treniranje, na validacijske podatke i na podatke za testiranje. Model koji je imao najbolje rezultate na validacijskom skupu je izabran i korišten za dobivanje krajnjih rezultata na testnom skupu podataka. Rezultati modela na testnom skupu podataka uspoređeni su sa burzovno trgovanim fondovima (engl. *Exchange Traded Fund*). ETF je fond kojim upravlja kompanija koja se bavi investiranjem, a sastoji se od različitog broja dionica ili nekih drugih financijskih instrumenata i takvim fondom se može trgovati na burzi. ETF-ovi izabrani za uspredbu su : *Vanguard S&P 500 ETF* (VOO), *SPDR S&P 500 ETF* (SPY), *iShares S&P 500 Growth ETF* (IVW), *Invesco S&P 500 Eql Wght ETF* (RSP).

Vremenska razdoblja i podjela podataka prikazana je u tablici 2.

Tablica 2. Podjela skupa podataka po vremenskim razdobljima

Skup podataka za trening	**Skup podataka za validaciju**	**Skup podataka za testiranje**
01.01.2005. - 01.12.2011.	01.01.2012. – 01.01.2013.	13.05.2013. – 20.03.2014.
01.01.2008. – 25.04.2018.	01.06.2018. – 07.05.2019.	10.09.2019. – 28.04.2020.
01.01.2007. – 01.08.2020.	01.11.2020. – 07.06.2021.	21.12.2021. – 17.05.2022.

Na grafu prikazanom na slici 13. prikazano je kretanje vrijednosti portfelja napravljenog i upravljanim pomoću našeg sustava (linija crne boje) u usporedbi sa izabranim ETF-ima u razdoblju od 13.05.2013. do 20.03.2014.



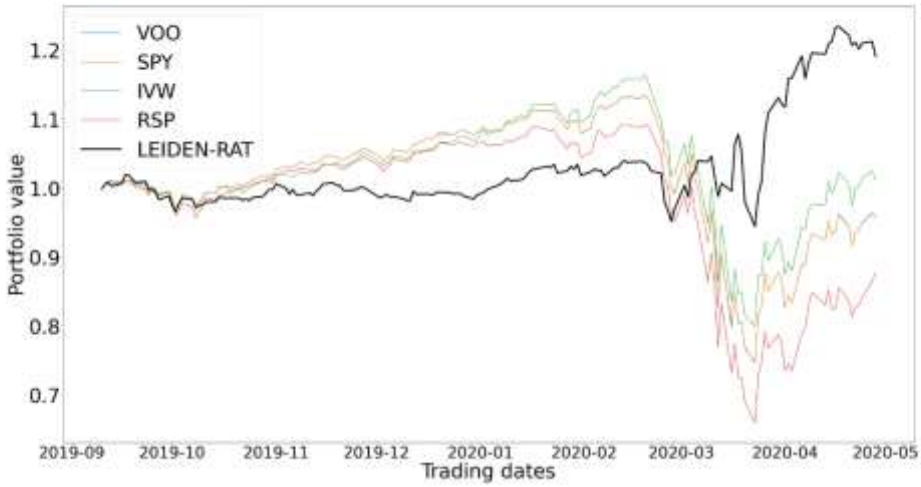
Slika 13. Kretanje vrijednosti portfelja u razdoblju od 13.05.2013. do 20.03.2014.

Usporedba vrijednosti za vremensko razdoblje od 13.05.2013. do 26.03.2014. prikazana je u tablici 3.

Tablica 3. Usporedba portfelja za razbolje od 13.05.2013 do 26.03.2014

Naziv	**Povrat**	**Annualized Sharpe Ratio**	**Annualized Calmar Ratio**
VOO	12.5%	1.16	2.07
SPY	12.4%	1.18	2.06
IVW	14.9%	1.36	2.48
RSP	14.6%	1.28	2.22
LEIDEN – RAT	**18.1%**	**1.2**	**2.10**

Na grafu prikazanom na slici 14. prikazano je kretanje vrijednosti portfelja napravljenog i upravljanim pomoću našeg sustava (linija crne boje) u usporedbi sa izabranim ETF-ima u razdoblju od 10.09.2019. do 28.04.2020.



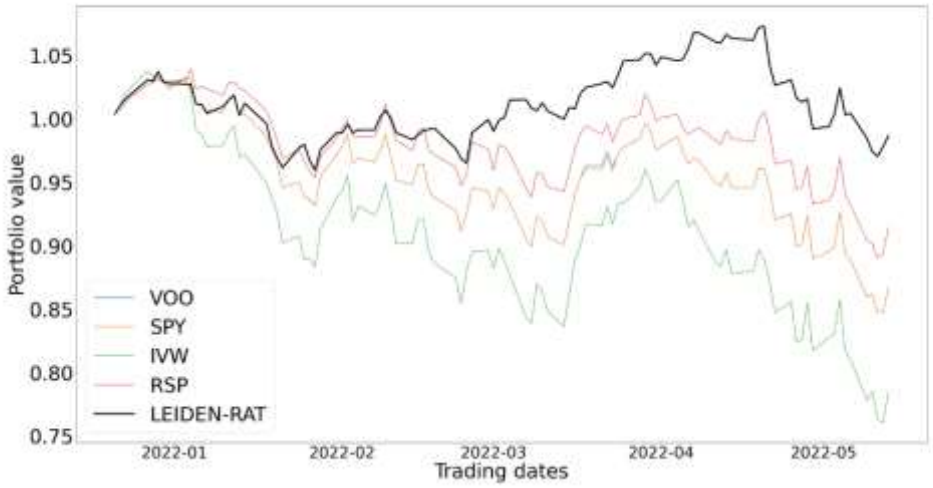
Slika 14. Kretanje vrijednosti portfelja u razdoblju od 10.09.2019. do 28.04.2020.

Usporedba vrijednosti za vremensko razdoblje od 10.09.2019. do 28.04.2020. prikazana je u tablici 4.

Tablica 4. Usporedba portfelja za razbolje od 10.09.2019 do 28.04.2020

Naziv	**Povrat**	**Annualized sharpe ratio**	**Annualized calmar ratio**
VOO	-4.1%	0.02	-0.12
SPY	-4.2%	0.01	-0.13
IVW	1.2%	0.20	0.04
RSP	-12%	-0.24	-0.31
LEIDEN – RAT	**19.1%**	**1.02**	**1.55**

Na grafu prikazanom na slici 15. prikazano je kretanje vrijednosti portfelja napravljenog i upravljanim pomoću našeg sustava (linija crne boje) u usporedbi sa izabranim ETF-ima u razdoblju od 21.12.2021. do 17.05.2022.



Slika 15. Kretanje vrijednosti portfelja u razdoblju od 21.12.2021. do 17.05.2022.

Usporedba vrijednosti za vremensko razdoblje od 21.12.2021. do 17.05.2022. prikazana je u tablici 5.

Tablica 5. Usporedba portfelja za razbolje od 21.12.2021 do 17.05.2022

Naziv	Povrat	Annualized sharpe ratio	Annualized calmar ratio
VOO	-10.4%	-0.92	-1.53
SPY	-10.7%	-0.93	-1.37
IVW	-21.5%	-1.15	-0.81
RSP	-8.5%	-0.64	-0.59
LEIDEN - RAT	**-1.3%**	**-0.09**	**-0.14**

Iz grafova na slikama 13, 14, i 15 vidljivo je da opisani sustav funkcionira i može se prilagoditi trenutnoj situaciji na tržištu. Ipak, u određenim vremenskim razdobljima sustav daje slabije rezultate, a mogući razlog slabijih rezulata je taj što prilikom treniranja modela za upravljanje portfeljom se pokušava maksimizirati profit bez uključivanja nekih mjera rizika. Također, bolji rezultati osim drugačije funkcije gubitka za model upravljanja portfeljom mogli bi se postići i dugotrajnijim i detaljnijim pretraživanjem hiperparametara, drugačijim načinom validacije npr. *Regular Day Forward Chaining* i biranjem drugačijih ulaznih vrijednosti u model (npr. tehnički indikatori).

U ovom radu tijekom testiranja vrijedile su slijedeće pretpostavke:

1. *Zero slippage*: Likvidnost svih financijskih instruemnata je dovoljno velika da se svaka kupovina odnosno prodaja izvršava odmah pri zadnjoj cijeni.
2. *Zero market impact*: Kapital koji je investiran od strane sustava za trgovanje nema utjecaja na samo tržište.

5. Zaključak

U ovom diplomskom radu prikazan je sustav za izgradnju diversificarnog portfelja i upravljenje tako izgrađenim portfeljom. Za izgradnju diversificiranog portfelja koristio se konvolucijski autoenkoder za ekstrakciju značajki iz slike za svaku pojedinu dionicu. Pomoću tako dobivenih značajki i korištenjem Leiden-ovog algoritma za otkrivanje zajednica grupiraju se dionice u visoko korelirane zajednice iz kojih se zatim biraju dionice koje će činiti portfelj. Za upravljanje portfeljom korištene su metode iz dubokog potpornog učenja. Arhitektura agenta je izmijenjena arhitektura Transformera koja je dosad korištena za primjenu na problemima u području prirodne obrade jezika. Ovakav sustav pokazuje obečavajuće rezultate. Kako bi dobili bolje rezultate od prikazanih potrebno je provesti daljnja istraživanja i poboljšanja kao što su promjena funkcije gubitka prilikom treniranja modela za donošenje odluka, drugačije ulazne vrijednosti i detaljnije pretraživanje hiperparametara.

Literatura

[1] Hu G., Hu Y., Yang K., Yu Z., Sung F., Zhang Z., Miemie, Q. (2018). Deep Stock Representation Learning: From Candlestick Charts to Investment Decisions, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[2] Jordan J. (2018). Introduction to autoencoders.

[3] Slika 1. URL: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

[4] Stanford University (2015). Cs231n: Convolutional neural networks for visual recognition, URL http://cs231n.github.io/

[5] Slika 2. URL: https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/?fbclid=IwAR0t5ZJJq9s-vzsQcZinPo1DbSNPfX04ij8XsGCSAR2BlezN-sL2G1QCV7I, pristupljeno: 20.lipanj 2022.

[6] Slika 3. https://medium.com/@achoulwar901/the-art-of-convolutional-neural-network-abda56dba55c , pristupljeno: 20.lipanj 2022.

[7] Simonyan K., Zisserman A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition, Computer Vision and Pattern Recognition

[8] Pavčević M.O., Nakić A. (2013). Uvod u teoriju grafova, Zagreb: Element, 2013

[9] Slika 7. URL: https://towardsdatascience.com/community-detection-algorithms-9bd8951e7dae, pristupljeno: 20.lipanj 2022.

[10] Traag V.A., Waltman L., Eck N.J. (2019). From Louvain to Leiden: guaranteeing well-connected communities, Social and Information Networks (cs.SI); Physics and Society (physics.soc-ph)

[11] Sharpe Ratio vs. Sortino vs. Calmar – Risk Adjusted Return (2022)., URL: https://www.optimizedportfolio.com/risk-adjusted-return/?fbclid=IwAR1-L2tPO7CF_YvAUOfZ3hKQFAJFozXagPmeQXX23_Spro7Lntg-_Trh0LQ

[12]http://www.doyensahoo.com/types-of-strategies.html?fbclid=IwAR242C10olM4z1gsfQv3miqc8kltYn3fR4dMHqA3Yw9SSQYXkUwl3QJ9fRk, pristupljeno: 18.svibanj 2022.

[13] Gašperov B., Begušić S., Posedel Šimović P., Kostanjčar Z. (2021). Reinforcement Learning Approaches to Optimal Market Making, Mathematics 2021, 9, 2689. https://doi.org/10.3390/math9212689

[14] Slika 10. URL: https://towardsdatascience.com/introduction-to-reinforcement-learning-markov-decision-process-44c533ebf8da, pristupljeno: 20.lipanj 2022.

[15] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser L., Polosukhin I. (2017). Attention Is All You Need

[16] Xu K., Zhang Y., Ye D., Zhao P., Tan M. (2020). Relation-Aware Transformer for Portfolio Policy Learning, Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20) Special Track on AI in FinTech

[17] Jiang Z., Xu D., Liang J. (2017). A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem

[18] Zhang Y., Zhao P, Wu Q., Li B., Huang J., Tan M. (2020). Cost-Sensitive Portfolio Selection via Deep Reinforcement Learning

[19] López de Prado M. (2016). Building Diversified Portfolios that Outperform Out-of-Sample, Journal of Portfolio Management

[20] Tolaab V., Lillocde F., Gallegatia M., Mantegnacd R.N. (2008). Cluster analysis for portfolio optimization, Journal of Economic Dynamics and Control, Volume 32, Issue 1, Pages 235-258

[21] DeepMind x UCL RL Lecture Series (2021)., URL: https://www.youtube.com/playlist?list=PLqYmG7hTraZDVH599EItlEWsUOsJbAodm, pristupljeno 18. svibanj 2022.

[22] Harry Markowitz's Modern Portfolio Theory: The Efficient Frontier (2021)., Guided Choice, URL: https://www.guidedchoice.com/video/dr-harry-markowitz-father-of-modern-portfolio-theory/?fbclid=IwAR2aXcZ1wx1vWI_PI4pA6gtqsZQ_YWhBnyIh5kvhC7AfOMRfoMogr50O7DU#:~:text=What%20is%20MPT%3F,resulting%20in%20the%20ideal%20portfolio.&text=MPT%20works%20under%20the%20assumption,a%20given%20level%20of%20return , pristupljeno 25.ožujak 2022.

Sažetak

U ovom diplomskom radu opisan je sustav za konstruiranje i upravljanje portfeljom dionica. Konstruiranje diversificiranog portfelja dionica je napravljeno korištenjem konvolucijskog autoenkodera i Leiden-ovog algoritma za otkrivanje zajednica. Upravljanje konstruiranim portfeljom dionica napravljeno pomoću metoda iz dubokog potpornog učenja. Arhitekutra agenta koji je donosio odluke je bila izmjenjena arhitektura Transformera. Opisani sustav je ispitan na vremenskim nizovima cijena dionica koje pripadaju S&P 500 indexu.

Ključne riječi: konvolucijski autoenkoder, teorija grafova, Leiden-ov algoritam, upravljanje portfeljom, duboko potporno učenje, arhitektura Transformer.

Abstract

This paper describes a system for constructing and managing a stock portfolio. The construction of a diversified stock portfolio was done using a convolutional autoencoder and Leiden's community detection algorithm. Stock portfolio managment was done using methods from deep reinforcement learning. The architecture of agent in deep reinforcement learning was modified architecture of the Transformer. The described system has been tested on stock price time series belonging to the S&P 500 index.

Keywords: convolutional autoencoder, graph theory, Leiden algorithm, portfolio managment, deep reinforcement learning, Transformer architecture