SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 7166

PREDVIĐANJE KRETANJA CIJENE KRIPTO VALUTA EVOLUCIJSKIM ALGORITMOM

Marko Benačić

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 7166

PREDVIĐANJE KRETANJA CIJENE KRIPTO VALUTA EVOLUCIJSKIM ALGORITMOM

Marko Benačić

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

ZAVRŠNI ZADATAK br. 7166

Pristupnik:	**Marko Benačić (0036493350)**
Studij:	Računarstvo
Modul:	Računarska znanost
Mentor:	doc. dr. sc. Marko Đurasević

Zadatak: **Predviđanje kretanja cijene kripto valuta evolucijskim algoritmom**

Opis zadatka:

Proučiti problem predviđanja kretanja cijene kripto valuta tijekom vremena. Opisati najčešće korištene metode iz literature koje je moguće primijeniti za procjenu kretanja njihovih cijena. Predložiti metodu baziranu na evolucijskim algoritmima za predviđanje kretanja cijene kripto valuta. Pronaći skup primjera povijesnih kretanja cijena za učenje predložene metode. Ocijeniti učinkovitost predložene metode procjenom zarada na neviđenim podacima. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Rok za predaju rada: 11. lipnja 2021.

Zahvaljujem se mentoru doc. dr. sc. Marku Đurasevi ću na upoznavanju sa ovom predivnom granom računarstva i sestri Ani na iznimnom strpljenju, pripomoći sa gramatičkim greškama i pravljenju kave svako jutro kad se nisam mogao probuditi.

Sadržaj

Uvod

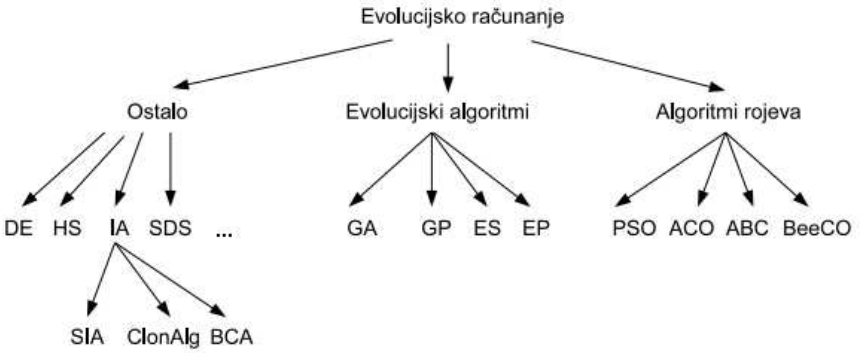
Neki od vrlo poznatih i zahtjevnih problema računarstva su: učenje neuronskih mreža, problem trgovačkog putnika, strategija igara, itd. Kao potencijalni način rješavanja takvih problema predstavljamo genetske algoritme. Genetski algoritmi su heurističke metode optimiranja koje imitiraju prirodni evolucijski proces (Golub, 2004). Oni nam omogućuju pronalaženje dobrih i učinkovitih rješenja koristeći vrlo intuitivni i naizgled jednostavni proces koji vuče inspiraciju iz prirode – evoluciju. Izrečeno računarskim rječnikom, evolucija se može opisati kao robustan proces pretraživanja prostora rješenja.

Cilj ovog završnog rada je objasniti što su genetski algoritmi, gdje se najčešće koriste, nabrojati korištene vrste selekcije, križanja, mutacije te njihovih hiperparametara. Nakon toga, objasniti što su kriptovalute, blockchain tehnologija na kojoj se one temelje, te metode trgovanja kriptovalutama. Objasniti što je simbolička regresija, te implementirati algoritam za trgovanje kriptovalutama koristeći stablo simboličke regresije optimizirano genetskim algoritmom. Završno, prikazati dobivene rezultate korištenjem nekoliko permutacija hiperparametara, te usporediti ih sa poznatom „buy and hold“ metodom trgovanja kriptovaluta.

1. Uvod u genetske algoritme

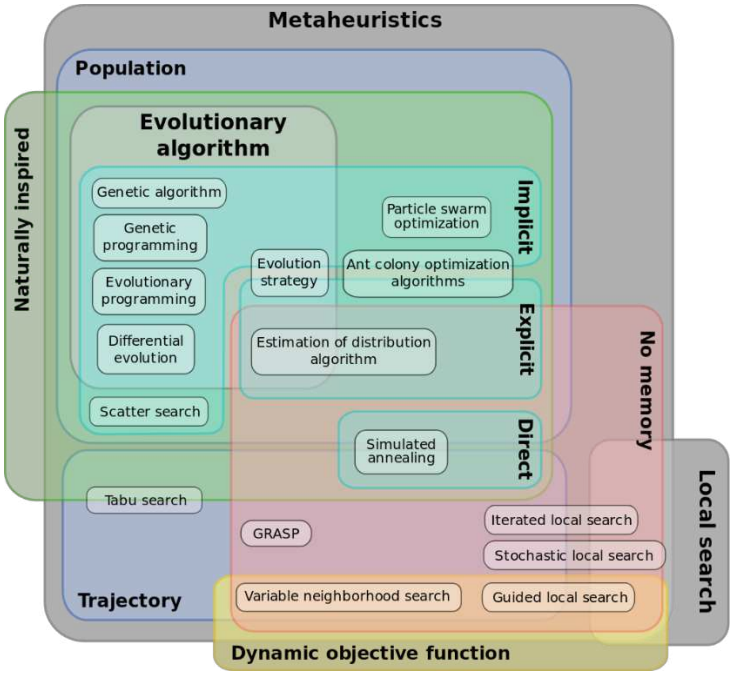
Teorija evolucije je dobro poznata teorija koja objašnjava stvaranje sve kompleksnijeg života na zemlji kroz iznimno dug vremenski period od prve pojave života na zemlji pa do sadašnjosti. Ona počiva na vrlo jednostavnim principima: prirodnoj selekciji, razmnožavanju organizama, te genetskoj mutaciji. Svaki organizam živi u okolini u kojoj je okružen kako hranom koju mora jesti da bi preživio, tako i predatorima od kojih se mora obraniti. Živo biće koje nije prilagođeno okolini u kojoj se nalazi, tj. ne može pronaći hranu, uspješno se razmnožiti, te obraniti od predatora, neće preživjeti u toj okolini. Organizmi koji prežive, te uspješno stvore potomke, osiguravaju da njihov genetski kod nastavi postojati još barem jednu generaciju. Osim toga, priroda je okruženje koje se kontinuirano mijenja, razne vrste izumiru druge se pojavljuju, vremenske prilike na zemlji se kroz dulje vremenske periode konstantno mijenjaju, zbog toga i se sama živa bića moraju prilagođavati novim promjenama. Slijedno tome, organizmi koji se nisu uspjeli prilagoditi tim promjenama neće preživjeti, a ona koja jesu hoće. To je temelj pojma prirodne selekcije. Zbog toga, organizmi moraju imati mogućnost genetske mutacije. Genetska mutacija je nasumično mijenjanje genetskog koda jedinke koji zbog prirodne selekcije kroz mnogobrojne slijedne generacije organizma stvara jedinke koje su sve bolje prilagođene svojoj okolini, mutacija je jedini razlog za ogromnu raznolikost živih bića kojoj možemo posvjedočiti u prirodi. Na temelju navedenog principa evolucije, prirodne selekcije, razmnožavanja i mutacije počivaju genetski algoritmi.

Genetski algoritmi su dio skupa evolucijskog računarstva, te njegovog podskupa poznat pod nazivom evolucijski algoritmi. Evolucijsko računanje dijelimo na tri grane: evolucijske algoritme, algoritme rojeva te ostale algoritme (Slika 1.1).



Slika 1.1 Podjela evolucijskih algoritama (Čupić, 2013).

Evolucijsko računanje je grana umjetne inteligencije koja se u generalnoj većini bavi optimizacijskim problemima, ona spada u skupinu metaheuristika (Čupić, 2013). Heuristike su algoritmi koji na ulaznom skupu podataka traže *približno* ili *dovoljno dobro* rješenje, podskup su optimizacijskih algoritama koje zovemo približni algoritmi. Odlikuju se niskom računskom složenošću, te ih dijelimo na *konstrukcijske algoritme* i *algoritme lokalne pretrage*. Slijedno, metaheuristike su skup algoritamskih koncepata koji se bave pronalaženjem i optimiziranjem heuristika, kako bi one funkcionirale na širokom spektru računarskih problema (Slika 1.2).



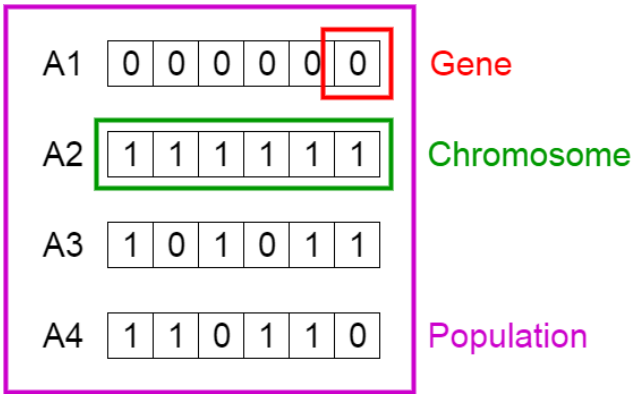
Slika 1.2 Klasifikacija metaheuristika (Wikipedia, 2021).

1.1. Generacijski genetski algoritam

Genetski algoritam kojim se bavimo u ovom radu zove se generacijski genetski algoritam. Zove se tako jer generacijski algoritam iterativno za svaki *n* (*zadani broj generacija*) stvara novu generaciju temeljenu na genetskom kodu prethodne generacije.

1.1.1. Jedinka

Jedna jedinka unutar generacije, općenito govoreći u genetskom algoritmu predstavlja jedno od rješenja za zadani problem koji želimo riješiti. Jedinke mogu biti predstavljene na različite načine, a najjednostavniji način kojim bismo ih predstavili bi bio niz bitova u kojemu svaki (uglavnom različiti) slijed bitova predstavlja neku informaciju. Ta informacija, tj. rješenje unutar jedinke predstavljena je njezinim *kromosomima* (Slika 1.3). Cilj genetskog algoritma je pronaći jedinku koja najbolje dolazi do rješenja nekog zadanog problema. U našoj implementaciji algoritma kromosome će predstavljati jedno simboličko stablo koje se sastoji od jednostavnih matematičkih operacija (više o tome kasnije). Osim kromosoma, svaka jedinka sadrži svoju dobrotu (engl. *fitness*). Dobrota je numerička vrijednost koja nam daje informaciju o tome koliko dobro jedna jedinka rješava zadani problem. Određivanje najboljih jedinki u svakoj generaciji temeljit će se na usporedbi međusobnih vrijednosti dobrote. Funkcija koja određuje vrijednost dobrote naziva se *funkcija cilja* ili *funkcija dobrote* .



Slika 1.3 Prikaz kromosoma slijedom bitova (Mallawaarachchi, 2013)

1.1.2. Općenita struktura genetskog algoritma

Na početku genetskog algoritma stvori se generacija jedinki proizvoljne veličine. Genetski kod prve generacije inicijaliziran je nasumično. Nakon toga, funkcijom dobrote provjerava se koliko dobro svaka jedinka rješava zadani problem korištenjem *funkcije dobrote*. Često korištene funkcije dobrote su:

- *Srednja kvadratna pogreška* (engl. Mean Square Error) – računa se prema izrazu:

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - Y_i)$$

- *Srednja apsolutna pogreška* (engl. Mean Absolute Error) – računa se prema izrazu:

$$\frac{1}{N}\sum_{i=1}^{N}|y_i - Y_i|$$

Gdje N predstavlja broj izlaznih vrijednost, y_i stvarnu vrijednost, a Y_i izračunatu vrijednost.

Nadalje, generacija se sortira na temelju vrijednosti dobrote, sortiranje može biti ulazno ili silazno u ovisnosti o traženom problemu. Nakon toga se stvara nova generacija jedinki. Svaka jedinka u novoj generaciji stvorena je operacijom *križanja* između dva roditelja iz prethodne generacije. Roditelji koji se križaju biraju se operacijom *selekcije*. Nakon njenog stvaranja, novostvorena jedinka se mutira te dodaje u novu generaciju. Nakon uspješnog stvaranja nove generacije, ona zamjenjuje staru, te se cijeli proces ponavlja željeni broj puta (Slika 1.4).

```
Genetski_algoritam
{
   t = 0
   generiraj početnu populaciju potencijalnih rješenja P(0);
   sve dok nije zadovoljen uvjet završetka evolucijskog procesa
   {
        t = t + 1;
        selektiraj P'(t) iz P(t-1);
        križaj jedinke iz P'(t) i djecu spremi u P(t);
        mutiraj jedinke iz P(t);
   }
   ispiši rješenje;
}
```

Slika 1.4 Struktura općeg genetskog algoritma (Golub, 2004)

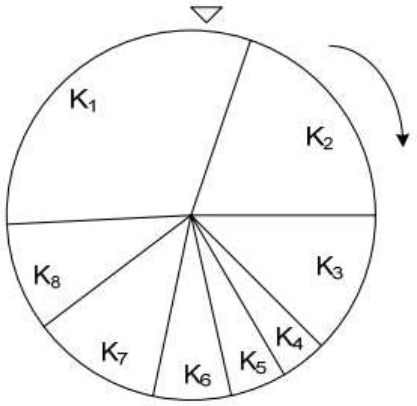
1.1.3. Selekcija, križanje i mutacija

Selekcija – Selekcija je dio genetskog algoritma u kojemu biramo roditelje djeteta kojeg ćemo staviti u novu generaciju jedinki. Postoji puno vrsta selekcije, neke od kojih ćemo nabrojati, ali generalni princip selekcije je: što bolju dobrotu jedinka ima u odnosu na ostatak populacije, to će imati veću šansu biti izabrana za roditelja. Neke od poznatih selekcijskih metoda su:

- **Proporcionalna selekcija** (engl. *Roulette wheel selection*) – Vjerojatnost odabira jedinke je proporcionalna iznosu njene dobrote u usporedbi sa dobrotama ostalih jedinki. Dakle, vjerojatnost biranja jedinke i računa se po formuli:

$$p(i) = \frac{f(i)}{\sum_{j=1}^{N} f(j)}$$

gdje je $f(i)$ dobrota jedinke i, N je veličina generacije (Slika 1.5).



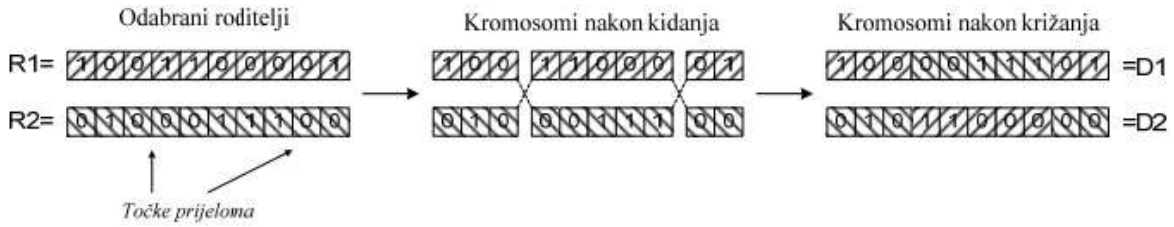
Slika 1.5 Proporcionalna selekcija

- **Turnirska selekcija** (engl. Tournament selection) – Selekcija u kojoj se iterativno ponavlja turnir od k suučesnika. Jedinka s najboljom dobrotom u turniru biva izabrana za razmnožavanje. Najčešća selekcija koja se koristi je *Troturnirska selekcija*.

Križanje – Nakon što su odabrani roditelji za stvaranje potomka, potrebno je na temelju njihovih kromosoma stvoriti novu jedinku. Navedena radnja se obavlja operacijom križanja. Operator križanja vrlo često uz sebe ima i parametar vjerojatnosti da će se križanje dogoditi. Uobičajene vrijednosti tog parametra su između 60% i 90%. Ako taj parametar nije zadan

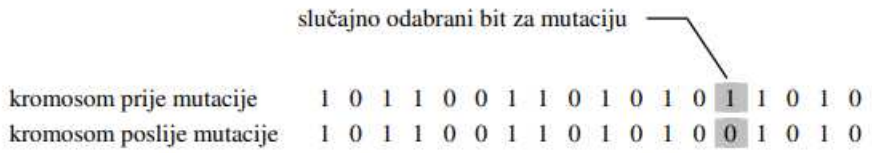
križanje će se uvijek dogoditi. Iz perspektive minimizacije optimizacijske funkcije selekcija i križanje imaju zadaću približavati rješenje minimumu, tj. sužavati područje djelovanja kako bi bilo sve bliže minimumu. Kao i kod selekcije, postoji mnogo vrsta križanja. Neke vrste križanja su:

- **Aritmetička sredina** – ova vrsta križanja koristi se kada su kromosomi predstavljeni nizom decimalnih brojeva. Primjer uporabe aritmetičke sredine jest kod optimizacije neuronskih mreža čije težine (engl. *weight*) predstavljaju kromosome jedne mreže (tj. jedinke).
- **Križanje s jednom točkom prekida** – Nasumično se odabire točka prekida, dijete je stvoreno sa kromosoma prvog roditelja do točke prekida, te nakon točke prekida sa kromosomima drugog roditelja.
- **Križanje s t-točaka prekida** – Općeniti slučaj križanja s točkom prekida, gdje se bira *t* točaka prekida (Slika 1.6).



Slika 1.6 Križanje s više točaka prekida (Čupić, 2013)

Mutacija – Kada je stvoreno novo dijete križanjem roditelja, dijete je potrebno mutirati. Mutacija sa određenom vjerojatnošću p_m mijenja jedan gen, što se ponavlja za svaki gen u kromosomima. Vjerojatnost mutacije obično se nalazi između 1% i 5%, a nekad zna biti i manja (Čupić, 2013). Iz perspektive minimizacije optimizacijske funkcije, mutacija nam služi naš algoritam „pobjegao“ iz lokalnog minimuma u kojem se našao u potrazi za globalnim minimumom (Slika 1.7).



Slika 1.7 Primjer jednostavne mutacije (Golub, 2004)

Elitizam – Nekad se zna dogoditi da prilikom selekcije najbolja jedinka iz trenutne generacije ne bude odabrana za križanje. To znači da će se genetski kod najbolje jedinke izgubiti izmjenom generacija. Kako bi spriječili da dođe do toga, uvodimo elitizam *n* najboljih jedinki. Elitne jedinke trenutne generacije se prekopiraju u novu generaciju prilikom njenog stvaranja.

Odabir koju od ovih operacija koristiti ovisiti će o konkretnom problemu koji želimo riješiti. Zbog nepredvidljivosti genetskih algoritama u praksi se vrlo često testiraju razne varijacije operacija i njihovih parametara, kako bi došli do permutacije koja nam daje najbolje rješenje. Parametri koji se uglavnom optimiziraju su: veličina populacije, vjerojatnost križanja i mutacije te broj jedinki za eliminaciju. Takvo testiranje ćemo obaviti i na problemu ovog rada.

2. Genetsko programiranje i problem simboličke regresije

Genetsko programiranje je podvrsta evolucijskog računanja, te je ona principijalno slična genetskom algoritmu. Jedina razlika je u tome što je genetski kod (tj. kromosomi) jedinke *program* koji je rješenje problema kojeg rješavamo. Cilj genetskog programiranja je automatski graditi programe koji će moći rješavati probleme neovisno o njihovoj domeni. Postoje razne korištene reprezentacije kromosoma, najčešće korištena i ona koja se koristi u ovom radu je sintaksno stablo (engl. *Syntax tree*).

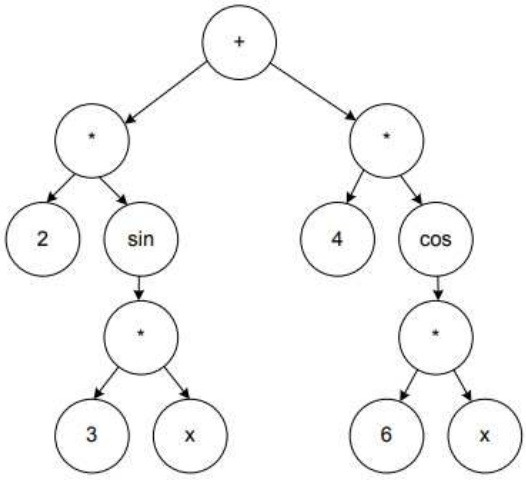
Pošto je genetsko programiranje implementirano tako da ne radi nikakve pretpostavke o modelu podataka koji mu damo nego jednostavno na temelju ulaznih podataka pokušava stvoriti što bolju aproksimaciju izlaznih, ono je po svojoj prirodi idealno za rješavanje vrlo poznatog računarskog problema simboličke regresije. (Poli et al., 2008). Rješavanje problema simboličke regresije je ujedno i jedan od ranijih upotreba genetskog programiranja (Poli et al., 2008).

2.1. Optimiziranje stabala simboličke regresije genetskim algoritmima

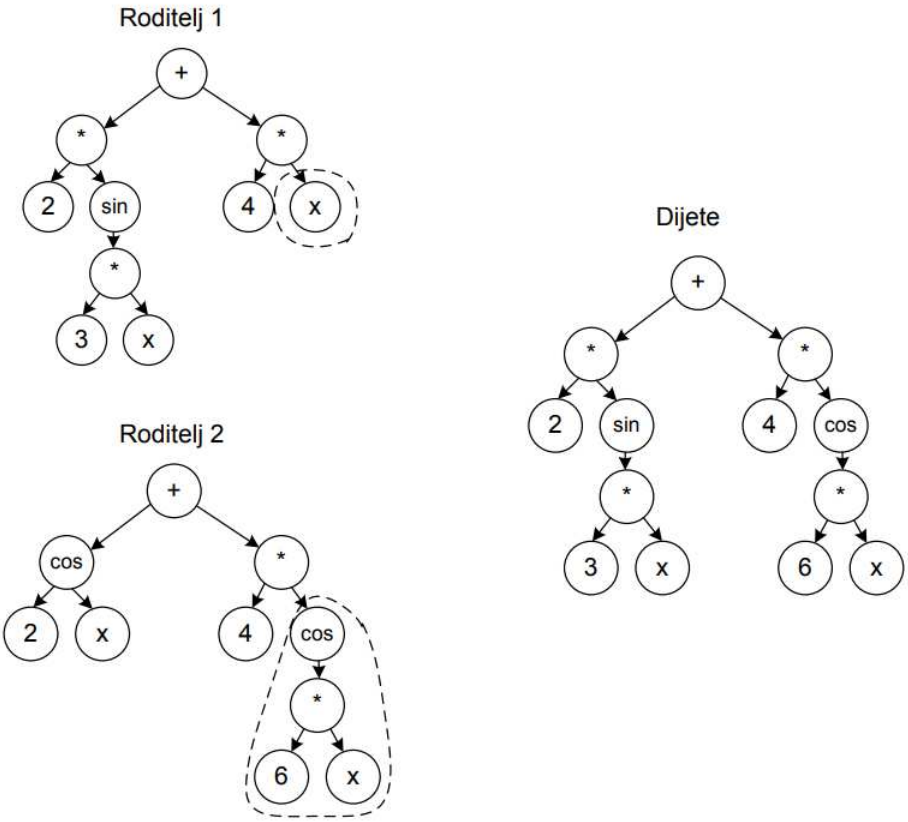
Cilj simboličke regresije je pomoću niza jednostavnih matematičkih operatora pronaći rješenje koje najbolje opisuje (aproksimira) funkciju ili problem koji želimo riješiti. Na primjer: skup podataka čiju funkciju želimo aproksimirati sastoji se od niza ulaznih vrijednosti x, te niza pripadajućih izlaznih vrijednosti $f(x)$. Taj ulazni skup želimo aproksimirati nizom jednostavnih matematičkih operatora. Oni mogu biti: $\sin(x), \cos(x)$, operator množenja, dijeljenja, zbrajanja, oduzimanja itd. Kada smo definirali podatke koje želimo aproksimirati te operatore koje želimo koristiti, potrebno je napraviti genetski algoritam koji će to i napraviti. Stvaramo početnu generaciju jedinki, čiji kromosomi predstavljaju razne varijacije sintaksnih stabala. Stabla će se sastojati od operacija koje smo prethodno zadali. Osim operacija, stablo također sadrži konstante (nasumično odabrani cijeli ili decimalni broj) i varijable (u našem primjeru postoji samo jedna varijabla: x). Ulazni podatci (varijable), konstante i funkcije bez argumenata (primjerice funkcija *random*) predstavljaju skup terminala (engl. *Terminal set*). Prethodno navedene operatore (+, -, /, *, sin, cos itd.) predstavljaju skup funkcija (engl. *Function set*). Skup

terminala i funkcija zajedno nazivamo skup primitiva (engl. *Primitive set*). Osim aritmetičkih operacija i matematičkih funkcija, skup funkcija može sadržavati logičke operacije (i, ili, negacija), kondicionale (if-then-else), petlje itd.

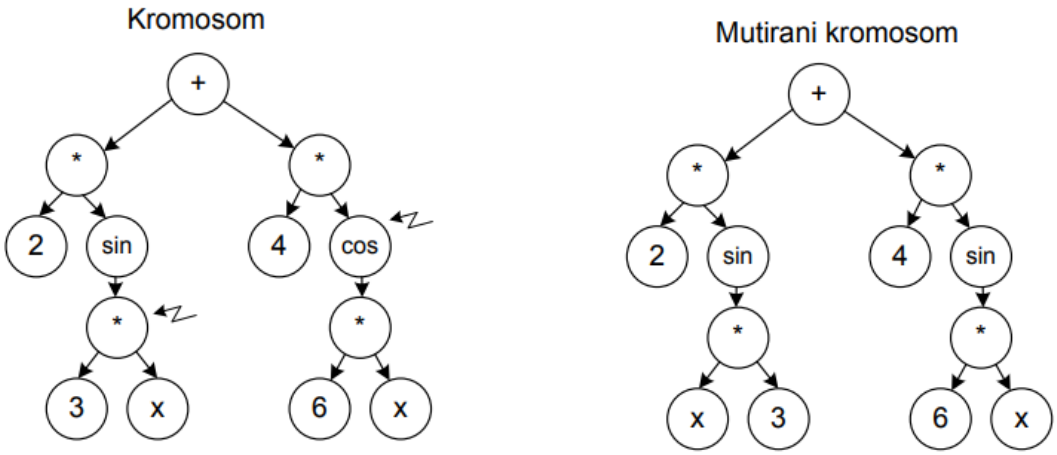
Primjer jednog nasumično generiranog stabla prikazan je na slici (Slika 2.1) (Čupić, 2013). Nadalje, funkciju dobrote jedinki možemo definirati na bilo koji od načina opisanih u prethodnom poglavlju (1.1), a slijedno tome, na isti način i operaciju selekcije. Za operaciju križanja potrebno je odabrati točke križanja na roditeljima, te napraviti zamjenu podstabala u toj točki (Slika 2.2). Operator mutacije možemo implementirati na razne načine: zamjena slučajno odabranih čvorova, zamjena podstabala sa drugim, slučajno generiranim podstablom i slično. Primjer slučajne zamjene čvora prikazan je na slici (Slika 2.3) (Čupić, 2013).



Slika 2.1 Primjer nasumično generiranog stabla



Slika 2.2 Izvedba križanja stabala



Slika 2.3 Mutacija stabla

2.2. DEAP biblioteka za stvaranje stabala

Za izradu programskog rješenja za naš problem predviđanja cijena kriptovaluta simboličkom regresijom koristit ćemo DEAP. DEAP (Distributed Evolutionary Algorithms in Python) je biblioteka (engl. *Library*) implementirana u programskom jeziku Pyhon koja nam daje pristup raznim modulima, strukturama podataka i algoritmima koji se mogu iskoristiti za stvaranje problemskih rješenja genetskim algoritmima. Osim već unaprijed napravljenih genetskih algoritama, biblioteka korisniku daje mogućnost implementacije vlastitih genetskih algoritama pomoću intuitivnih funkcija, klasa i modula kao što su: *Creator*, *Base*, *Toolbox*, *Algorithms*, *Genetic Programming*.

Prva odluka koju je potrebno donjeti prije same implementacije algoritma simboličke regresije jest koje sve primitive želimo koristiti. To će ovisiti o konkretnom problemu kojeg rješavamo te vrsti ulaznih podataka.

Primitive set – Postoje dvije vrste primitiva koji se koriste: Slabo tipizirani (engl. *Loosely typed*) i strogo tipizirani (engl. *Strongly typed*). U strogo tipiziranim implementacijama primitivi i terminali moraju biti točno onog tipa koji smo zadali (npr. Float), dok su slabo tipizirani fleksibilniji, jer implementacija ne forsira specifični tip čvora. Set primitiva incijaliziramo pozivom funkcije PrimitiveSet, kojoj kao argumente prilažemo ime i broj ulaznih varijabli. Nakon toga funkcijom *gp.addPrimitive* i *gp.addTerminal* dodajemo operatore i konstante koje želimo koristiti. Primjer programskog isječka je na slici (Slika 2.4).

```
pset = PrimitiveSet("main", 2)
pset.addPrimitive(max, 2)
pset.addPrimitive(operator.add, 2)
pset.addPrimitive(operator.mul, 2)
pset.addTerminal(3)
```

Slika 2.4 Dodavanje primitiva

Creator – Nakon što je napravljen set primitiva, potrebno je napraviti jedinku, čiji su kromosomi stablo zadanih primitiva. Osim toga, potrebno je definirati funkciju dobrote, te odredili je li maksimizacijska ili minimizacijska. Te dvije zadaće se obavljaju jednostavnim pozivom funkcije *creator.create*. Touple podataka *weights* se postavlja na negativnu vrijednost

kod minimizacije, a na pozitivnu kod maksimizacije. Primjer programskog isječka je na slici (Slika 2.5).

```
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", gp.PrimitiveTree, fitness=creator.FitnessMin)
```

Slika 2.5 Kreiranje funkcije i dobrote jedinke

Toolbox – Pomoću toolbox klase biramo koje ćemo funkcije koristiti za stvaranje populacije, računanje dobrote, selekciju, mutaciju i sl. Primjer postavljanja željenih funkcija i parametara je dan na slikama (Slika 2.6) (Slika 2.7).

```
toolbox = base.Toolbox()
toolbox.register("expr", gp.genHalfAndHalf, pset=pset, min_=1, max_=2)
toolbox.register("individual", tools.initIterate, creator.Individual, toolbox.expr)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("compile", gp.compile, pset=pset)
```

Slika 2.6 Inicijalizacija

```
toolbox.register("evaluate", evalSymbReg, points=[x/10. for x in range(-10,10)])
toolbox.register("select", tools.selTournament, tournsize=3)
toolbox.register("mate", gp.cxOnePoint)
toolbox.register("expr_mut", gp.genFull, min_=0, max_=2)
toolbox.register("mutate", gp.mutUniform, expr=toolbox.expr_mut, pset=pset)

toolbox.decorate("mate", gp.staticLimit(key=operator.attrgetter("height"), max_value=17))
toolbox.decorate("mutate", gp.staticLimit(key=operator.attrgetter("height"), max_value=17))
```

Slika 2.7 Postavljanje parametara selekcije i mutacije

Osim navedenih funkcija, biblioteka sadrži još niz drugih metoda koje olakšavaju spremanje i ispisivanje statistike svake iteracije genetskog algoritma, spremanje najboljih jedinki, benchmarking alata te implementaciju višejezgrenog rada.

Cijela biblioteka i njena dokumentacija nalaze se na: https://github.com/deap/deap.

3. Kriptovalute i trgovanje kriptovalutama

Kriptovalute su vrsta digitalne imovine (digitalnog novca) čija implementacija počiva na tehnologiji blockchain-a. One su dizajnirane u obliku digitalnih kovanica (engl. *coin*) koji informacije o sebi i međusobnim transakcijama kriptografski zaštićuju te ih zapisuju na blockchain. Sustav neke kriptovalute odlikuje se anonimnošću transakcije, te decentraliziranosti svojih valuta i načina plaćanja, za razliku od trenutnog financijskog sustava koji vlada u svijetu, gdje su sve transakcije novca praćene, te prolaze kroz centralizirani sustav banke.

Stvaranje novčanice kriptovalute je također decentralizirano, kriptovalutu stvaraju jedan ili više povezanih računala na temelju *PoW* (Proof of Work) algoritma. *PoW* je koncept koji je omogućio stvaranje prve kriptovalute (Bitcoin), rješavajući problem duple potrošnje (engl. *double-spending*) koji je od pojave interneta onemogućavao pojavu isključivo digitalnih valuta. Osim PoW algoritma postoji još nekoliko efektivnih algoritama za stvaranje novčanica kriptovaluta i potvrđivanja transakcija, ali pošto to nije tema ovog rada, u daljnje detalje se neće ulaziti.

3.1. Trgovanje kriptovalutama

Svaka kriptovaluta ima trenutni broj novčanica koji cirkuliraju sustavom. Taj broj može biti fiksan ili se može povećavati s vremenom. Također gornja granica broja novčanica može biti zadana ali i ne mora. Ako se broj kovanica beskonačno povećava, kriptovaluta je podložna inflaciji te se svojom cijenom ponaša slično pravom novcu. Vrijednost svake kovanice kriptovalute je određena njenom cijenom na nekom od tržišta kriptovaluta. Primjer takvog tržišta je *binance.com*. Dakle bilo koja punoljetna osoba može se pridružiti tržištu i kupiti kovanicu kriptovalute u zamjenu za novac. Primjer cijena jedne kovanice različitih kriptovaluta dana 11.6.2021. prikazan je na slici preuzetoj sa *coinmarketcap.com* (Slika 3.1). Kriptovaluta za koju ćemo pokušati dobiti profit je Cardano. Pošto se cijena kovanica kriptovalute mijenja po zakonu ponude i potražnje, te kupnjom kovanice postajemo njezin vlasnik i možemo ju prodati, smisleno je povući paralelu sličnosti između kriptovaluta i dionica. Predviđanje cijena

dionica je tipičan problem simboličke regresije, te iz tog razloga u ovom se radu koristi genetsko programiranje kako bi stvorili model kretnje cijena kriptovaluta u svrhu stvaranja profita.

Rank	Name	Symbol	Market Cap	Price
1	Bitcoin	BTC	$687,398,854,040	$36,696.53
2	Ethereum	ETH	$286,361,755,681	$2,463.51
3	Tether	USDT	$62,638,937,160	$1.00
4	Binance Coin	BNB	$54,394,428,822	$354.52
5	Cardano	ADA	$48,779,518,966	$1.53
6	Dogecoin	DOGE	$42,242,357,781	$0.325
7	XRP	XRP	$40,100,338,482	$0.8682
8	USD Coin	USDC	$23,241,507,432	$1.00
9	Polkadot	DOT	$21,861,697,541	$22.98
10	Uniswap	UNI	$13,300,032,055	$23.13

Slika 3.1 Cijene kriptovaluta

4. Predviđanje kretanja cijena kriptovaluta

Kako bi uspješno predvidjeli kretanje cijene neke kriptovalute, prvo je potrebno utvrditi koje ćemo indikatore koristiti kao ulazni skup podataka. Indikatori su uglavnom brojčane vrijednosti koje predstavljaju stanje neke valute na tržištu. Indikatori koje ćemo koristiti su:

- *High* – Najveća zabilježena cijena tijekom jednog dana.
- *Low* – Najniža zabilježena cijena tijekom jednog dana.
- *Open* – Cijena zabilježena tijekom otvaranja tržišta.
- *Close* – Cijena zabilježena tijekom zatvaranja tržišta.
- *Volume* – Broj kovanica koje su prošle kroz tržište taj dan.
- *Marketcap* – Sveukupna vrijednost kriptovalute na tržištu, računa se umnoškom cijene jedne kovanice i količinom kovanica u opticaju.

Osim navedenih indikatora, postoje i puno kompleksniji tržišni indikatori koji se obično izvode iz prethodno navedenih. Neki od njih su:

- *Rate of change* (ROC) – računa se po formuli:

$$ROC(n) = \left(\frac{Cijena\ zatvaranja\ prethodnog\ dana}{Cijena\ zatvaranja\ prije\ n\ dana} - 1\right) * 100$$

- *Relative strength index* (RSI) – računa se po formuli:

$$RSI(n) = 100 - (\frac{100}{1 + RS(n)})$$

Gdje je

$$RS(n) = \frac{\sum_{i \in D^+(n)} r_i}{-\sum_{i \in D^-(n)} r_i}$$

Gdje je $D^+(n)$ skup od *n* dana u kojima je cijena rasla, a $D^-(n)$ skup od *n* dana u kojima je cijena padala.

U ovom radu koristit će se ROC indikator, koji će uspoređivat cijene zatvaranja prethodna dva dana. Ulazni skup podataka su dnevne cijene i ostali indikatori u razdoblju od 2.10.2017. do 27.2.2021.

4.1. Implementacija

Radi treniranja i evaluacije našeg modela, koristit ćemo 70/30 razdiobu ulaznih podataka, gdje prvih 70% podataka koristimo za trening modela, dok idućih 30% koristimo za evaluaciju našeg rješenja.

Ulazni podatci zadani su u csv formatu, kod za parsiranje podataka naveden je u nastavku (Slika 4.1):

```
def parse_data(data_path):
    variables = list()
    train_data = list()
    test_data = list()

    all_data = list()

    with open(data_path, newline='') as csvfile:
        csv_reader = csv.reader(csvfile, delimiter=',')
        for row in csv_reader:
            lista = list()
            for x in row:
                try:
                    fl = float(x)
                    lista.append(fl)
                except ValueError:
                    lista.append(x)
            all_data.append(lista)

    variables = all_data.pop(0)
    datacount = len(all_data)

    datacount_train = math.ceil(TRAIN_SPLIT * datacount)
    datacount_test = datacount - datacount_train

    train_data = all_data[:datacount_train]
    test_data = all_data[datacount_train:]

    return train_data, test_data, variables
```

Slika 4.1 Parsiranje ulaznog skupa podataka

Genetski algoritam implementiran je na način koji je prikazan u poglavlju 2.2. Skup ulaznih argumenata, primitiva i konstanti koji su korišteni prikazan je na slici (Slika 4.2).

```
self.pset.renameArguments(ARG0='High',ARG1='Low',ARG2='Open',ARG3='Close',ARG4='Volume',ARG5
='Marketcap', ARG6 = 'ROC')
self.pset.addPrimitive(operator.add, 2)
self.pset.addPrimitive(operator.sub, 2)
self.pset.addPrimitive(operator.mul, 2)
self.pset.addPrimitive(protectedDiv, 2)
self.pset.addPrimitive(operator.neg, 1)
self.pset.addPrimitive(math.cos, 1)
self.pset.addPrimitive(math.sin, 1)
self.pset.addPrimitive(operator.abs, 1)
self.pset.addPrimitive(min, 2)
self.pset.addPrimitive(max, 2)
self.pset.addEphemeralConstant("rand101", lambda: random.randint(-1, 1))
```

Slika 4.2 Ulazni argumenti i korišteni primitivi

Nakon određivanja ulaznih argumenata, te dodavanja primitiva i postavljanja parametara genetskog algoritma, potrebno je implementirati evaluacijsku funkciju, tj odlučiti kako ćemo odrediti dobrotu. U ovom radu dobrotu određujemo kao razliku novca kojeg smo zaradili našom strategijom kupnje na cijelom skupu podataka s čime oduzimamo vrijednost koju bi zaradili da smo koristili „*buy and hold*“ metodu.

Strategija kupnje – Naša implementirana strategija kupnje je jednostavna, ali dovoljna za testiranje algoritma, ona glasi:

„*Na temelju ulaznih podataka jučerašnjeg dana, procijeni cijenu zatvaranja (engl. Closing price) valute za sutrašnji dan. Ako je procijenjena cijena idućeg dana veća nego trenutna, kupi danas i prodaj sutra za profit.*“

Za svaku se kupnju troši imaginarnih 1 dolar, dok se idući dan sav taj iznos prodaje za profit ili gubitak.

Buy and hold - poznata metoda kupnji kriptovaluti u kojoj se novčanice u jednom trenutku kupe, te pod pretpostavkom da će kriptovalute nastaviti rast u cijeni kroz dulji vremenski period, prodaju tek kad cijena dosegne neku ciljnu cijenu ili kraj testnog perioda.

Dakle, cilj našeg algoritma jest maksimizirati izraz:

$$razlika = zarađen\ iznos - buy\ and\ hold$$

Implementacija evaluacijske funkcije prikazana je u nastavku (Slika 4.3):

```
def evalSymbReg(self, individual, points):
    func = self.toolbox.compile(expr=individual)
    money_made = 0
    for i in range(len(points)):
        if i > 0:
            roc = 1     # rate of change u usporedbi sa prethodnim danom
            if i > 1:
                roc = (points[i - 1][7] / points[i - 2][7] - 1) * 100
            calculated_value = func(High=points[i-1][4],
                                    Low=points[i-1][5],
                                    Open=points[i-1][6],
                                    Close=points[i-1][7],
                                    Volume=points[i-1][8],
                                    Marketcap=points[i-1][9],
                                    ROC=roc)
            if calculated_value > points[i - 1][7]:
                money_made += points[i][7] - points[i - 1][7]  # kupi ovaj dan i prodaj iduci

    buy_and_hold = points[len(points) - 1][7] - points[1][7] #kolko bi zaradili da smo kupili i drzali
    return (money_made - buy_and_hold),
```

Slika 4.3 Evaluacijska funkcija

4.2. Usporedba modela sa metodom buy and hold

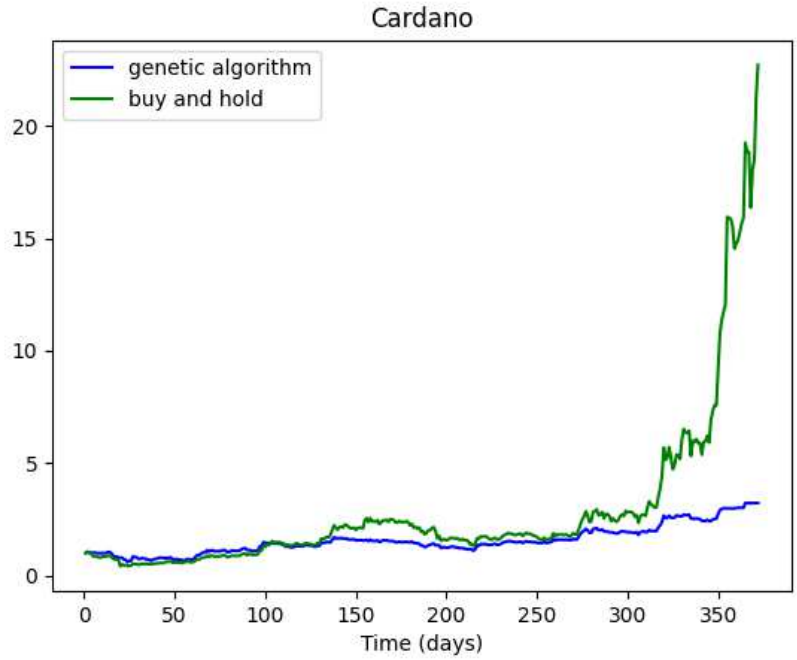
Rezultati su dobiveni uporabom turnirske selekcije, križanja s jednom točkom, te uniformnom mutacijom.

Tablica 1: Tablica profita u usporedbi sa raznim hiperparametrima

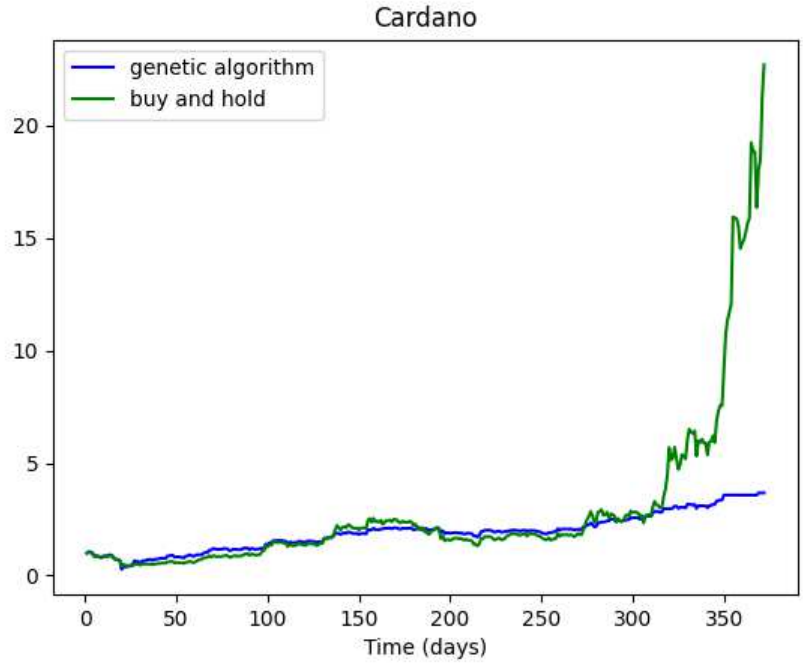
PARAMETRI	**PROFIT**
BROJ GENERACIJA = 10 POPULACIJA = 10	4.0202165809573702
BROJ GENERACIJA = 100 POPULACIJA = 10	3.418107366222499
BROJ GENERACIJA = 10 POPULACIJA = 100	3.2229569182872915
BROJ GENERACIJA = 50 POPULACIJA = 300	3.086063890175791
BROJ GENERACIJA = 100 POPULACIJA = 300	2.3605716381189639
BROJ GENERACIJA = 500 POPULACIJA 300	2.9151711501747122
BROJ GENERACIJA = 500 POPULACIJA 10	4.1063704533370347

Profit predstavlja ostvarenu dobit za ulaganje s $1 (jednim dolarom). Kao što se vidi iz tablice; najveći profit ostvarila je zadnja permutacija parametara (veličina populacije = 10 jedinki, broj generacija = 500). Dakle, svaki puta kada je algoritam smatrao da će idući dan biti veća cijena od prethodnog , kupio bi $1 vrijednosti te idući dan prodao kupljeno. Tom strategijom u našem slučaju najveći ostvareni profit je 3.1 dolara. Na slikama (slika 4.4, 4.5, 4.6) prikazana je usporedba zarade genetskog algoritma sa zaradom metode buy and hold. X os predstavlja vrijeme koje je prošlo od prve kupnje, pa do zadnjeg dana skupa ulaznih podataka. Vrijednost na y osi predstavlja multiplikator zarade u usporedbi s početnim ulogom. Dakle, buy-and-hold metoda je s početnih uloženih $1 ostvarila profit od više od $20. To je daleko bolji rezultat od

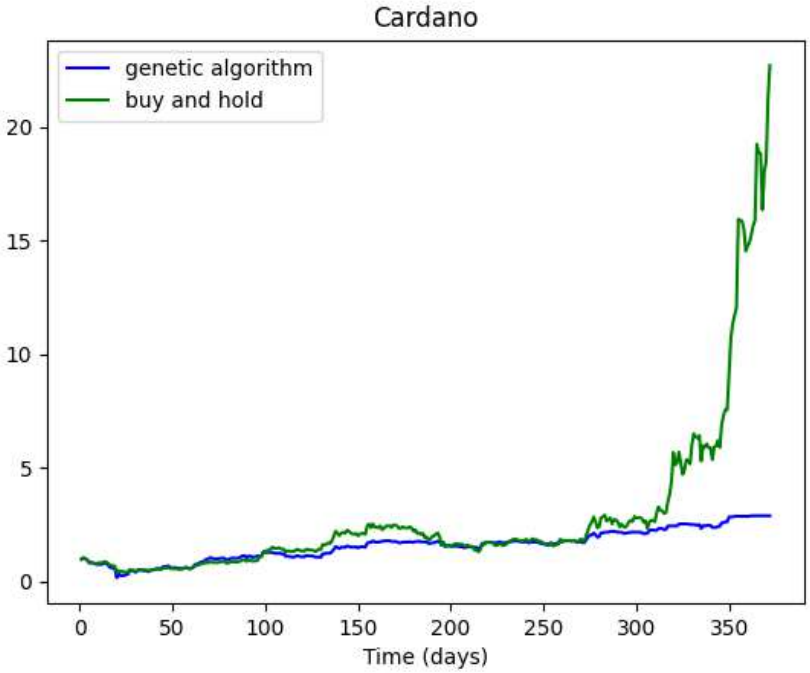
onog kojeg je ostvario naš genetski algoritam, ali treba uzeti u obzir da je završni period našeg skupa podataka također period u kojemu su kriptovalute počele eksponencijalno rasti u cijeni. Takav porast u cijeni gotovo je pa nemoguće predivjeti bez stručnog znanja i uvida u kretanje svjetskog tržišta kriptovaluta. Takva nepredvidljivost naglog porasta našem algoritmu ne odgovara, jer on je naučen generalizirati dosad viđene podatke, ali nema nikakav uvid u svjetske trendove i razvoj tehnologije.



Slika 4.4 Broj generacija = 100, veličina populacije = 40



Slika 4.5 Broj generacija = 100, veličina populacije = 100



Slika 4.6 Broj generacija = 400, veličina populacije = 20

5. Upute za pokretanje programa

Prije pokretanja programa potrebno je imati instaliran python 3.7, te *pip*. Pomoću *pip-a* instalirati navedene libraryje:

pip install matplotlib

pip install deap

pip install networkx

pip install graphviz

Program se pokreće pozivanjem naredbe:

python main.py --dataset datasets/Cardano.csv

Gdje argument –dataset predstavlja putanju ulaznih podataka u CSV(Comma separated value) formatu.

Zaključak

Za razliku od modernih metoda strojnog učenja, genetski algoritmi relativno su neistraženi aspekt u području predviđanja cijena dionica i kritpovaluta. Iako naši rezultati pokazuju mogućnost zarade našom implementiranom metodom, metoda buy and hold pokazuje znatno bolje rezultate. Osim toga, povećanjem kompleksnosti sintaktičkog stabla te povećanjem broja generacija i veličine populacije, nismo primjetili znatno poboljšanje u preformansama algoritma. Superiornost metode buy and hold može biti do toga što je za testni skup podataka odabran period naglog rasta kriptovaluta, pa samim time se buy and hold metoda pokazala vrlo profitabilnom (čak 20x početnog uloga). Pretpostavka koju donosim je da bi naš implementirani genetski algoritam bio puno efektivniji na manje volatilnim tržištima u kojima cijena nema toliko veliku fluktuaciju. Osim toga, implementacija koja u sebi sadržava indikatore koji prate svjetske trendove (npr. analiza spominjanja kriptovaluta na Twitteru, Facebook, mainstream medijima i novinama itd.) bi mogla poboljšati sposobnost algoritma da predvidi nagli porast cijene. Nadalje, smatram da genetski algoritmi mogu dobro poslužiti kao provjera i dopuna već implementiranih modela strojnog učenja a ne kao čista metoda predviđanja cijene. Razlog tome može biti što je tržište dionica i kriptovaluta vrlo kompleksno, te ono ovisi o mnogo više parametara pored onih kojima smo imali pristup u ovom radu.

Literatura

1. Čupić, Marko. Prirodom inspirirani optimizacijski algoritmi. Metaheuristike. 2013.
2. Golub, Marin. Genetski algoritam. 2004.
3. Poli, Riccardo, et al. A field guide to genetic programming. Lulu. com, 2008.
4. Potvin, J. Y., Soriano, P., & Vallée, M. Generating trading rules on the stock markets with genetic programming. Computers & Operations Research, 31(7), 1033-1047. 2004.
5. Sebastião, H., Godinho, P. Forecasting and trading cryptocurrencies with machine learning under changing market conditions. Financ Innov 7, 3, 2021.
6. Wikipedia. Metaheuristika. Poveznica: https://en.wikipedia.org/wiki/Metaheuristic; pristupljeno 8. lipnja 2021.
7. Wikipedia. Kriptovalute. Poveznica: https://en.wikipedia.org/wiki/Cryptocurrency; pristupljeno 10. lipnja 2021.
8. Deap.readthedocs.io. DEAP documentation — DEAP 1.3.1 documentation. Poveznica: https://deap.readthedocs.io/en/master/index.html; pristupljeno 11. lipnja 2021.
9. Viljini Mallawaarachchi, Introduction to Genetic Algorithms — Including Example Code, 8. srpnja 2017, Poveznica: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3; pristupljeno 8. lipnja 2021.

Sažetak

Problem predviđanja cijena dionica i kriptovaluta te profitabilno trgovanje njima je jedno od unosnijih i težih zadataka umjetne inteligencije. U ovom radu se pomoću genetskog programiranja pokušava napisati program koji će donositi ispravne odluke o kupnji i prodaji određene kriptovalute na temelju javno dostupnih indikatora tržišta. Potom se rezultati ostvareni genetskim programiranjem i isprobavanjem raznih permutacija hiperparametara uspoređuju sa poznatom metom trgovanja kriptovaluta „buy and hold“. Završno, dana su razmatranja dobivenih rezultata te daljnji potencijalni načini poboljšanja programskog rješenja.

Ključne riječi: Genetski algoritmi, genetsko programiranje, simbolička regresija, kriptovalute, predviđanje cijena kriptovaluta, trgovanje kriptovalutama

Summary

The problem of forecasting stock and cryptocurrency prices and profitable trading is one of the most lucrative and difficult tasks in Artificial inteligence. The goal of this papir is writing a program that makes correct decision between buying and selling a given cryptocurrency on the basis of publicly available market indicators. Consequent results of our genetic programming algorithm using different permutations of hyperparameters are then compared against a well known method of crypto trading called "buy and hold". Finally, further considerations of aquired results and possible future algorithm improvements are given.

Keywords: Genetic algorithm, genetic programming, symbolic regression, cryptocurrency, cryptocurrency price forecasting, cryptocurrency trading