# Szakdolgozat


Miskolci Egyetem

# Genetikus algoritmus alkalmazása a tőzsdei kereskedésben

**Készítette:**

Tamás Csaba

Programtervező informatikus

**Témavezető:**

Dr. Karácsony Zsolt

egyetemi docens

Miskolc, 2021

**Miskolci Egyetem**
Gépészmérnöki és Informatikai Kar
Alkalmazott Matematikai Intézeti Tanszék **Szám:**

# Szakdolgozat Feladat

Tamás Csaba (KO8L9F) programtervező informatikus jelölt részére.

**A szakdolgozat tárgyköre:** Statisztikai elemzés

**A szakdolgozat címe:** Genetikus algoritmus alkalmazása a tőzsdei kereskedésben

**A feladat részletezése:**

A részvényekhez tartozó adatsorok hozzáférhetőségének vizsgálata, ezen adatsorok kinyerése, megfelelő formátumúvá konvertálása a további feldolgozhatóság érdekében. A technikai elemzési technikák megismerése, majd a feldolgozott, letisztított adatsorból gazdasági mutatók felhasználásával döntéseket hozni az árfolyam várható változásáról. Ezen mutatók matematikai képletek, melyek algoritmizálása szükséges. A kutatás tárgya, hogy lehet-e kereskedő szoftvert készíteni, illetve a genetikus algoritmus módszerével hatékonnyá tenni. A felhasznált mutatókat egy grafikonon összesíteni, illetve a vásárlási eladási időpontokat feltüntetni. A program korábbi adatsorokon való tesztelése a működés bebiztosítása érdekében, illetve a hatékonyság mérése.

**Témavezető:** Dr. Karácsony Zsolt, egyetemi docens

**A feladat kiadásának ideje: 2019. szeptember 30.**

. . . . . .
szakfelelős

# Eredetiségi Nyilatkozat

Alulírott **Tamás Csaba**; Neptun-kód: `KO8L9F` a Miskolci Egyetem Gépészmérnöki és Informatikai Karának végzős Programtervező informatikus szakos hallgatója ezennel büntetőjogi és fegyelmi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy *Genetikus algoritmus alkalmazása a tőzsdei kereskedésben* című szakdolgozatom saját, önálló munkám; az abban hivatkozott szakirodalom felhasználása a forráskezelés szabályai szerint történt.

Tudomásul veszem, hogy szakdolgozat esetén plágiumnak számít:

- szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

Alulírott kijelentem, hogy a plágium fogalmát megismertem, és tudomásul veszem, hogy plágium esetén szakdolgozatom visszautasításra kerül.

Miskolc, ..... .év ..... .hó ..... .nap

.....

Hallgató

1.

szükséges (módosítás külön lapon)

A szakdolgozat feladat módosítása

nem szükséges

..... .....

dátum témavezető(k)

2. A feladat kidolgozását ellenőriztem:

témavezető (dátum, aláírás): konzulens (dátum, aláírás):

..... .....

..... .....

..... .....

3. A szakdolgozat beadható:

..... .....

dátum témavezető(k)

4. A szakdolgozat ..... szövegoldalt

..... program protokollt (listát, felhasználói leírást)

..... elektronikus adathordozót (részletezve)

.....

..... egyéb mellékletet (részletezve)

.....

tartalmaz.

..... .....

dátum témavezető(k)

5.

bocsátható

A szakdolgozat bírálatra

nem bocsátható

A bíráló neve: .....

..... .....

dátum szakfelelős

6. A szakdolgozat osztályzata

a témavezető javaslata: .....

a bíráló javaslata: .....

a szakdolgozat végleges eredménye: .....

Miskolc, ..... .....

a Záróvizsga Bizottság Elnöke

# Tartalomjegyzék

# 1. fejezet

# Bevezetés

A modern részvénytársaságok múltja egészen a 17. századig nyúlik vissza. Ekkor alapították az első részvénytársaságot Amszterdamban. Kezdetben csak egyetlen társasággal kereskedtek, az osztalékfizetés pedig néhány évvel később kezdődött el. A tőzsde a vállalatok számára az egyik legfontosabb pénzszerzési módszer. Nyilvánosan kereskedhetnek, és további pénzügyi tőkét gyűjthetnek a terjeszkedéshez azáltal, hogy a vállalat tulajdonrészeit nyilvános piacon értékesítik. A történelem azt mutatja, hogy a részvények és más eszközök árfolyama fontos része a gazdasági tevékenység dinamikájának. Gyakran tekintik egy ország gazdasági erejének és fejlődése elsődleges mutatójának.

1971-ben megtörtént az áttörés, elkezdte a működését a Nasdaq, a világ első elektronikus tőzsdéje. Eleinte csak egy jegyzési rendszer volt, nem adott lehetőséget a kereskedésre. Mivel csökkentette a brókerek nyereségét azáltal, hogy az árkülönbséget csökkentette a kínált és a keresett összegek között, ezért nem kedvelték. Az évek során tovább bővült a Nasdaq, már automatikus kereskedési rendszereket is implementáltak.

Napjainkban egyre nagyobb embertömeget ér el a tőzsdézés. Az internet terjedése platformot kínált a részvények világának, ami nem is annyira veszélyes, mint elsőre gondolnánk. Több weboldal, könyv készült már, mely bemutatja miért és hogyan érdemes kereskedni. Már egy kattintással tulajdonosai lehetünk például a Tesla vállalatnak, és a világon bárhonnan nyomon követhetjük a részvények mozgását, s ez nagyban megkönnyíti a dolgunkat.

A kereskedésnek több stílusa létezik. Az egyik, amikor a számítógép előtt ülünk nyolc órát, és nézzük mikor csökken a részvény, hogy vásároljunk, böngésszük a fórumokat, hogy milyen vállalatokba fektetnek mások. Ez elég monoton és stresszes munka véleményem szerint, hiszen grafikonokat kellene nézegetni nap, mint nap, illetve izgulni, hogy ne lefelé menjen az árfolyam. Ennek apropóján érdemes elgondolkodni azon, hogy a tudást, amit könyvekből, weboldalakról, publikációkból összegyűjtünk, azt egy rendszerbe szervezhetjük, amiből egy automata kereskedő szoftvert lehet készíteni.

Egy 2006-os kutatásban olvastam, hogy a NASA egy genetikus algoritmus segítségével készített egy antennát, melynek a sugárzási mintája optimális a felhasználásukhoz. Ekkor gondoltam, hogy a saját felhasználásomra is tudnám kamatoztatni ezt a technikát. A genetikus algoritmusok a természeti kiválasztás folyamatain alapszanak. Arra használjuk őket, hogy optimalizációs, illetve keresési problémákhoz megbízható megoldást találjunk. Számomra az indikátorok vizsgálatánál kerülhet szóba optimalizáció, amikor el kell dönteni vásárol-e a program, vagy elad.

A szakdolgozatom fő célja, hogy egy automata kereskedő szoftver készítsek el genetikus algoritmus felhasználásával. Én azt vallom, hogy lehetséges a megfelelő hatásfok

elérése, hiszen a mesterséges intelligencia már bemutatta képességeit sok területen, az én esetemben is támaszt nyújthat. Az általam célként kitűzött szimulációhoz elengedhetetlen egy adatbázis, mely a piac tényleges mozgását reprezentálja, így valóságos környezetet tudok szimulálni a stratégiám teszteléséhez. Az olvasó megismerheti a részletes taktikáját a programomnak, a technológiákat, amiket alkalmaztam, és egy átfogó képet kaphat arról, hogyan is használható fel a statisztika a részvényekkel történő kereskedés során.

# 2. fejezet

# Gazdasági háttér

A kereskedelem szerves része az életünknek már az ősidőktől fogva. Alapvető minden társadalomban, hogy áruért, szolgáltatásért valamilyen ellenértékkel szolgáljunk. Régen még nem volt jelen a pénz, mint fizetőeszköz, körülbelül az i.e. 7. század elejétől kezdődött el a felhasználásuk. Ekkoriban a tereken, piacutcákon árusították termékeiket az árusok. Innen ered a piac szó, olaszul piazza, amely térséget jelent.

## 2.1. Piac

A piac az eladók és vevők cserekapcsolatainak összessége, az eladók és vevők kölcsönhatásait fejezi ki; elemei közé tartozik többek között a kereslet, kínálat, ár és jövedelem. A vásárlók és az eladók a piacon lépnek interakcióba egymással, a vevők szükségletei az eladók kínálataiból kielégítésre kerülnek. A piacnak többféle formája létezik, melyeket a javak fajtája különböztet meg. Ezek közül, hogy párat említsek, létezik az

- árupiac, ahol olyan termékekbe fektethetünk, mint például az olaj, arany, kávé,
- a munkaerőpiac, ahol a munkát keresők (eladó) és a munkaerőt kereső (vevő) igénye találkozik,
- és a tőkepiac, ami a befektetések, értékpapírok piaca [7].

## 2.2. Tőzsde

A tőzsde egy nyilvános, központosított piac, ahol többek között az értékpapír kereskedelem történik. A tőzsde a másodlagos piaca az értékpapíroknak, ahol a befektetők között történik a tranzakciók lebonyolítása. Ezt úgy kell elképzelni, hogy egy vállalat kibocsát értékpapírokat, ezzel egyidőben az első tulajdonosukhoz kerülnek, és ha úgy dönt az új tulajdonos, hogy ezeket eladná, akkor ezt a másodlagos piacon teszi meg. Mivel ezek az ügyletek bonyolultak, illetve szigorú eljárási szabályokat alkalmaznak, ezért egy befektetési szolgáltatón keresztül kell ezeket lebonyolítani, ezeket brókercégeknek hívjuk [2].

### 2.2.1. Részvény

Az értékpapír kereskedelem egyik lehetősége a részvényekkel történő adás-vétel. A részvény tulajdonképpen egy vállalat által kibocsátott értékpapír, mely tulajdonjogot teste-

sít meg. Ahhoz, hogy részvényt tudjunk vásárolni, szükség van egy értékpapír számlára, amelyet egy brókercégnél szükséges nyitni. Ezután a brókercégen keresztül lehetőség nyílik a szimpatikus vállalatok részvényeinek megvásárlására. Általában egy online felületen történnek ezek a tranzakciók. A részvényekből kétféleképpen is lehet jövedelemre szert tenni: vagy osztalék kifizetésből, vagy az árfolyamnyereségből.


2.1. ábra. A Coca-Cola részvény záróárai.

### 2.2.2. Osztalék

Az osztalék a vállalat által megtermelt profitnak az a része, amit kifizet a tulajdonosai részére. Egy vállalat működése akkor hatékony, ha teljesülnek a rövid- és hosszútávú nyereségkövetelmények, köztük a kvázi költségek is. Ekkor a többletét kifizetheti a tulajdonosai részére osztalék formában, azonban ez egyáltalán nem kötelező. Ezért érdemes olyan vállalatot kiválasztani, amely bizonyította már az osztalékfizetési szándékát, mint például a Coca-Cola.

| | | | | DGR | DGR | DGR | DGR |
|---|---|---|---|---|---|---|---|
| Cég neve | Ticker | no. Yrs | MR% | 1-yr | 3-yr | 5-yr | 10-yr |
| Coca-Cola Company | KO | 59 | 2.44 | 2.5 | 3.5 | 5.5 | 6.4 |

2.2. ábra. A Coca-Cola osztalékfizetési összefoglalója [10].

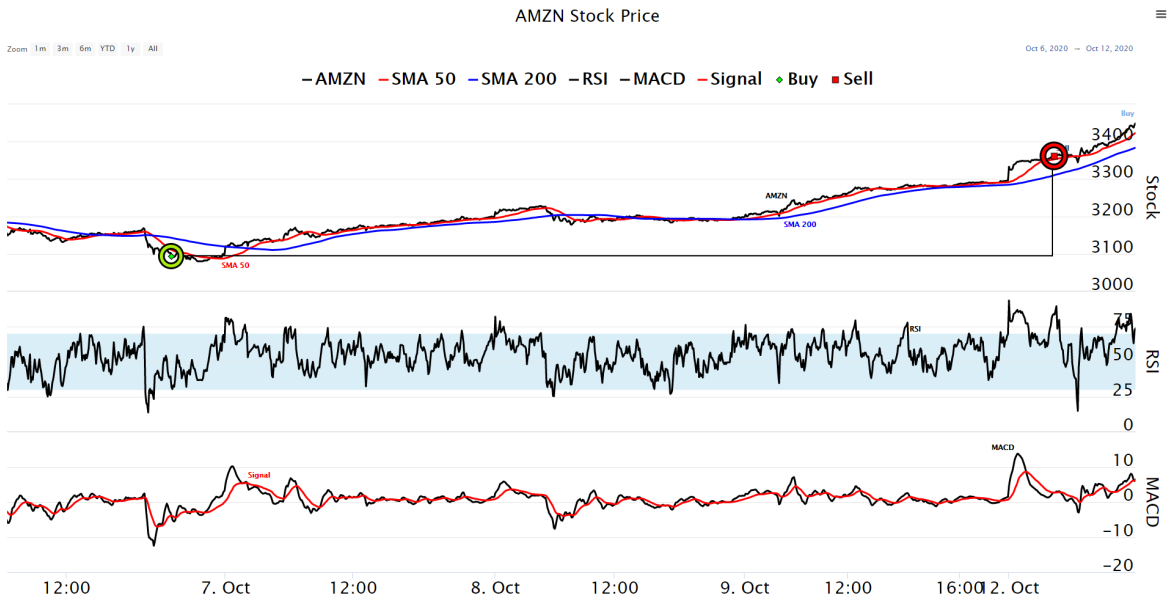
A táblázatban látható adatok jelentése a következő:

- Ticker a cég egyedi azonosítója, mely a tőzsdén azonosítja,
- no. Yrs oszlop mutatja, hogy hány éven keresztül emelte az osztalékát,
- MR% mutatja a legfrissebb emelés mértékét,
- DGR oszlopok pedig azt írják le, hogy az alattuk jelzett időtávokon átlagosan mennyivel emelkedett az osztalék.

Érdemes megnézni egy vállalat osztalékfizetési szokásait vásárlás előtt. Az a cég remek célpont nagyvonalakban, amely folyamatosan emeli az osztalékát, az osztalékfizetési trendjét nézve nem található csökkenés, és a DGR mutatói nem romlanak [14].

### 2.2.3. Árfolyamkülönbség alapú profitálás

A másik lehetséges módja a pénzkeresésnek az, ha olcsón megvesszük a részvényt, majd drágábban eladjuk, számításba véve a kötelező adózási feltételeket. A szakdolgozatom fő célja, hogy megtaláljam azokat az időpontokat, amikor érdemes vásárolni, illetve eladni. Az alábbi ábrán az algoritmus 269$ profitot termelt részvényenként, amely




2.3. ábra. Az Amazon vállalat részvényén történt árfolyamnyereség.

elég szép eredménynek tekinthető. Ezt a profitot később vissza tudja forgatni részvény vásárlásba, azonban ez egyúttal a kockázatot is növeli.

A következő alfejezetekben részletesebben bemutatom a kereskedéshez használt indikátorokat, illetve azokat a jellemzőiket, amelyek segítenek eldönteni, hogy érdemes-e vásárolni a részvényből vagy sem. Ezek az indikátorok közismertek, és előszeretettel használják a befektetők a technikai analízis során. Mindegyik indikátornál más és más fogja jelenteni a vásárlási és eladási jelzést, ezekre bővebben kitérek minden indikátor esetében.

## 2.3. Simple Moving Average (SMA)

Az SMA, avagy az egyszerű mozgóátlag a legkönnyebben használható indikátor, tulajdonképpen egy tartomány átlagát kell venni a kiszámításához.

$$\text{SMA} = \frac{A_1 + A_2 + \ldots + A_n}{n},$$

ahol

$A_i$: részvény záróárai,
$n$: a vizsgált periódus.

Az SMA-t arra használjuk, hogy megnézzük milyen irányba mozog a trend. A 200 periódusú SMA-t szokás használni hosszútávú trend vizsgálatára, az 50 periódusút pedig középtávúnak tekintjük. Általában arra használjuk, hogy az árakat, illetve a technikai indikátorokat kisimítsuk, viszont meg kell találni egy arany középutat, hisz minél nagyobb a periódus, annál nagyobb a simítás mellett a késése is a jelzésnek.


2.4. ábra. Az 50 napos és a 200 napos mozgóátlag grafikonja.

Az ábrán látható, hogy az 50 periódusú SMA mennyivel gyorsabban leköveti az árfolyam mozgását, mint a 200 periódusú társa. Az 50 periódusú és a 200 periódusú SMA együtt nézve felhasználható szignálként.

### 2.3.1. Aranykereszt

Amikor a rövid periódusú SMA alulról keresztezi a hosszabb periódusút, akkor ezt egy vásárlási jelzésként értelmezhetjük. Ezt a találkozási pontot aranykeresztnek nevezzük. Azt mondhatjuk, hogy az aranykereszt egy megbízható jelzés, azonban a kereskedők más indikátorokkal is meg szokták erősíteni a jelzést.

### 2.3.2. Halálkereszt

Halálkeresztről akkor beszélünk, ha a rövidtávú SMA felülről keresztezi a hosszútávút, azaz alábukik, tehát ellentétes irányú, mint az aranykereszt. Ezt a jelzést is figyelembe veszik sokan eladás előtt.

## 2.4. Exponential Moving Average (EMA)

Az EMA, teljes nevén az exponenciális mozgóátlag hasonlít az SMA-hoz, azonban ez súlyozásos technikát alkalmaz. Minél régebbi egy pont, annál kisebb súllyal veszi figyelembe, ami azt eredményezi, hogy hamarabb fogja követni az árfolyam mozgását.

$$\begin{aligned}\mathrm{EMA}_i = &\left(\mathrm{Value}_i * \left(\frac{\mathrm{Smoothing}}{1+n}\right)\right) \\ &+ \mathrm{EMA}_{i-1} * \left(1 - \left(\frac{\mathrm{Smoothing}}{1+n}\right)\right),\end{aligned}$$

ahol

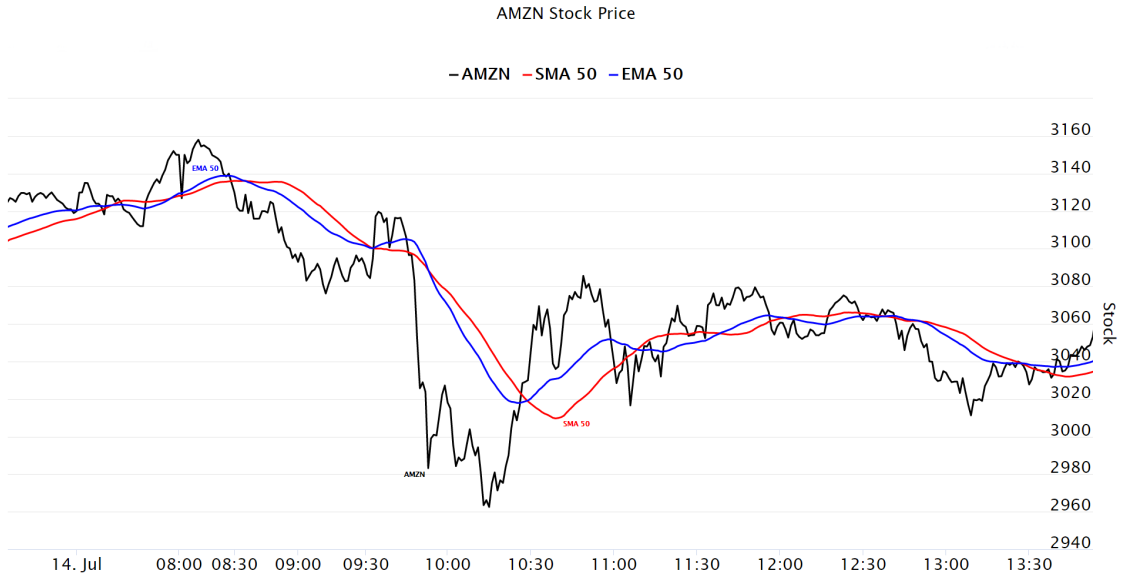
| | |
|---|---|
| $\mathrm{Value}_i$: | a részvény aktuális ára, |
| $\mathrm{EMA}_{i-1}$: | az $(i-1)$. EMA érték, |
| Smoothing: | a simítás értéke, |
| $n$: | a vizsgált periódus. |

A legáltalánosabb, hogy a simítás értékét kettőre állítják be, mely a legfrissebb árnak magasabb súlyozást biztosít. Minél magasabb ez a simítás, annál nagyobb hatással lesz az EMA értékére az utolsó záróár.

Amint láthatjuk, az $i$-edik EMA kiszámításához szükség van az $(i-1)$-edik értékre is, amely nem áll rendelkezésre, ha az első értékét szeretnénk megkapni. Ha a periódust például 16-nak vesszük alapul, akkor a 17. értéktől kezdődően tudjuk számolni az EMA-t a következőképpen:

$$\mathrm{EMA}_{17} = \left(15\$ * \left(\frac{2}{1+16}\right)\right) + \left(\mathrm{SMA}_{16} * \left(1 - \left(\frac{2}{1+16}\right)\right)\right).$$

Mivel nem létezett az $\mathrm{EMA}_{16}$, ezért az előző 16 értéknek a számtani középértékét vettük helyettesítésképp.




2.5. ábra. Az EMA és az SMA átlagok összevetése.

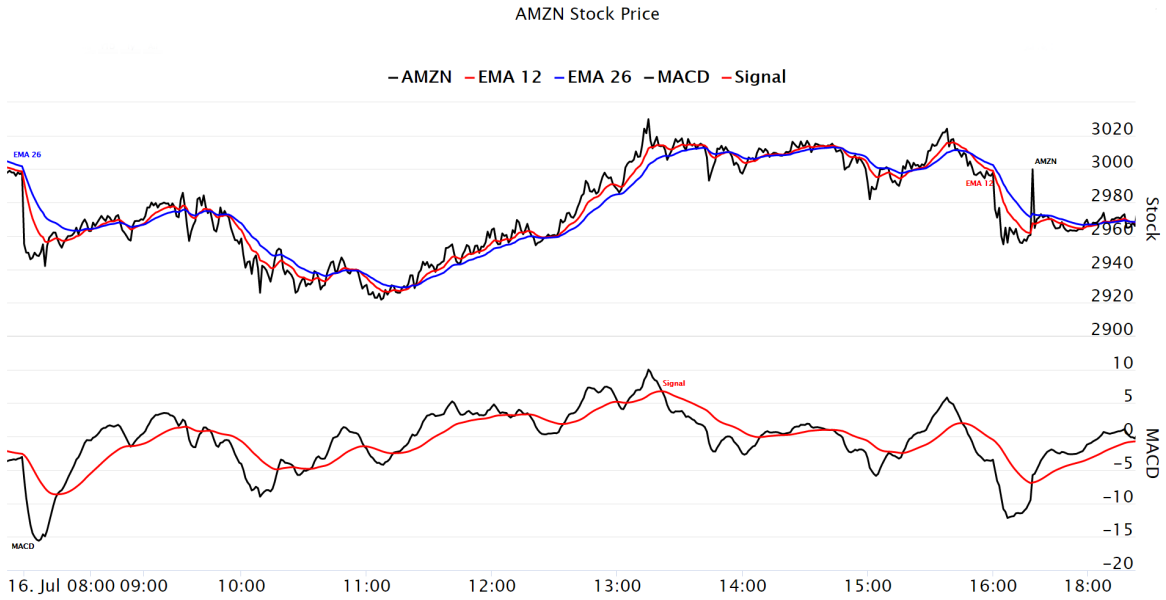
Mint az látható, az EMA gyorsabban reagál a részvény mozgására, mint az ugyanolyan periódusú SMA. Azonban az EMA-t nem ebben a formában fogom hasznosítani, hanem egy következő indikátor számításához fogom felhasználni, amit a következő alfejezetben foglalok össze.

## 2.5. Moving Average Convergence/Divergence (MACD)

Az MACD egy egyszerű és megbízható trend követő indikátor, mely két mozgóátlag közötti kapcsolatot mutatja meg. Az MACD-t úgy számítjuk ki, hogy a 26 periódusú EMA-t kivonjuk a 12 periódusú EMA-ból, és ennek a két mozgóátlagnak a különbsége adja az MACD vonalat, melyhez egy szignál vonalat is szoktak társítani. Az MACD egy nulla alatt és fölött ingadozó függvény minimum és maximum határ nélkül. A szignál vonal a 9 periódusú EMA-ja az MACD-nek, tehát

$$\begin{aligned} \text{MACD} &= \text{EMA}_{12}(\text{záróár}) - \text{EMA}_{26}(\text{záróár}), \\ \text{Szignál} &= \text{EMA}_{9}(\text{MACD}). \end{aligned}$$

A szignál vonal arra való, hogy jelezze a vásárlási és eladási pontokat. Többféleképp értelmezhetjük a függvények mozgását, de a leggyakoribb alakzatok, amit figyelnek a keresztezések, a divergenciák és a hirtelen mozgások.




2.6. ábra. Az MACD és a szignál függvény grafikonja.

A képen látható, ha a 12 periódusú a 26 periódusú felett van, akkor az MACD is a 0 érték fölött mozog, s minél magasabb az MACD értéke, annál nagyobb a két mozgóátlag közötti távolság is. Az MACD és a szignálja, ha keresztezik egymást, akkor ezt egy jelzésnek tekintjük. Amennyiben a szignál felülről keresztezi a MACD vonalát, akkor ezt vásárlási, ha alulról, akkor eladási jelzésként értelmezhetjük.

## 2.6. Relative Strength Index (RSI)

Az RSI egy nagyon népszerű momentum indikátor, mely az árváltozásokat figyeli, és ezáltal megállapíthatóak a túlvett és túladott állapotok. Az RSI az MACD-hez hasonlóan egy oszcillátor, mely két határértékkel rendelkezik: 0 és 100 között mozog az értéke. Hagyományosan 70-től felfelé azt mondjuk, hogy a részvény túlvett és várhatóan

visszafordul a trend, 30 alatt pedig túladott, alulértékelt. Ezt a két szintet szokás jelzésként használni. Számítását tekintve az előzőekhez képest összetettebb, két részben érdemes nézni.

$$1)\ \mathrm{RSI} = 100 - \left[\frac{100}{1 + \frac{\text{Average Gain}}{\text{Average Loss}}}\right],$$

$$2)\ \mathrm{RSI} = 100 - \left[\frac{100}{1 + \frac{(\text{Previous Average Gain} * (\text{period} - 1)) + \text{Current Gain}}{(\text{Previous Average Loss} * (\text{period} - 1)) + \text{Current Loss}}}\right].$$

| Záróár | Változás | Növekedés | Csökkenés | Átlag. Növ. | Átlag. Csök. | RS | RSI |
|---|---|---|---|---|---|---|---|
| 41.99 | | | | | | | |
| 41.92 | -0.07 | | 0.07 | | | | |
| 41.93 | 0.01 | 0.01 | | | | | |
| 41.98 | 0.05 | 0.05 | | | | | |
| 41.93 | -0.05 | | 0.05 | | | | |
| 41.99 | 0.06 | 0.06 | | | | | |
| 41.88 | -0.11 | | 0.11 | 0.02 | 0.0383 | 1.5217 | 34.2857 |
| 41.93 | 0.05 | 0.05 | | 0.15 | 0.1917 | 1.7826 | 43.9024 |

2.7. ábra. Példa az RSI indikátor számítására.

Tekintsük a fenti példa táblázatot, melyre az $\mathrm{RSI}_6$-ot végeztem el. Az első lépésben kiszámítom a 6 napos átlagát a Növekedés oszlop, majd a Csökkenés oszlop elemeinek a második elemtől fogva. E kettő hányadosa adja az RS értéket. Majd az első képletbe behelyettesítve kapom az RSI értékét. A következő átlagokat már a második képlet alapján kell kiszámítani úgy, hogy az előző sor átlagait megszorozzuk öttel, s hozzáadjuk a jelenlegi sorban található növedesést/csökkenést attól függően, hogy merre mozdult el a részvény ára.

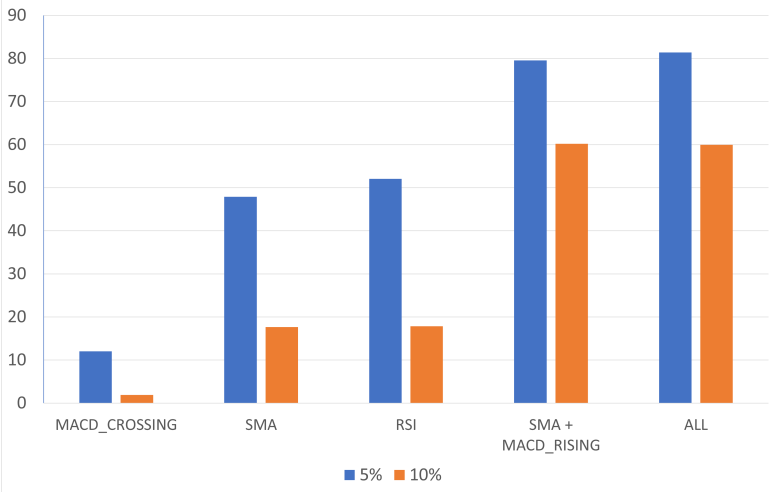

2.8. ábra. Az RSI indikátor mozgása az árfolyammal együtt.

Néhány elemző azonban nem a fent említett szinteket alkalmazza, hanem a középvonal keresztezést nézik. Más szavakkal, ha az RSI nagyobb, mint 50, akkor pozitív jelzésnek, ha kisebb, mint 50, akkor negatív jelzésnek tekintik.

Ha tekintjük a 2.8-as ábrát, akkor azt láthatjuk, hogy több alkalommal jelzett az RSI indikátor. Amikor volt egy nagyobb esés, akkor túladott jelzést mutatott az indikátor, amikor pedig hirtelen ugrások voltak az árfolyamban felfelé, akkor túlvett jelzést kaptunk. Ha az első túlvett jelzést megnézzük, akkor valóban lehetett volna profitot szerezni, azonban ha tovább haladunk a grafikonon akkor látható, hogy az árfolyam nem állt meg, további emelkedést mutatott. Ezért is érdemes az indikátorokat megerősíteni több másik indikátorral.

## 2.7. Indikátorok vizsgálata

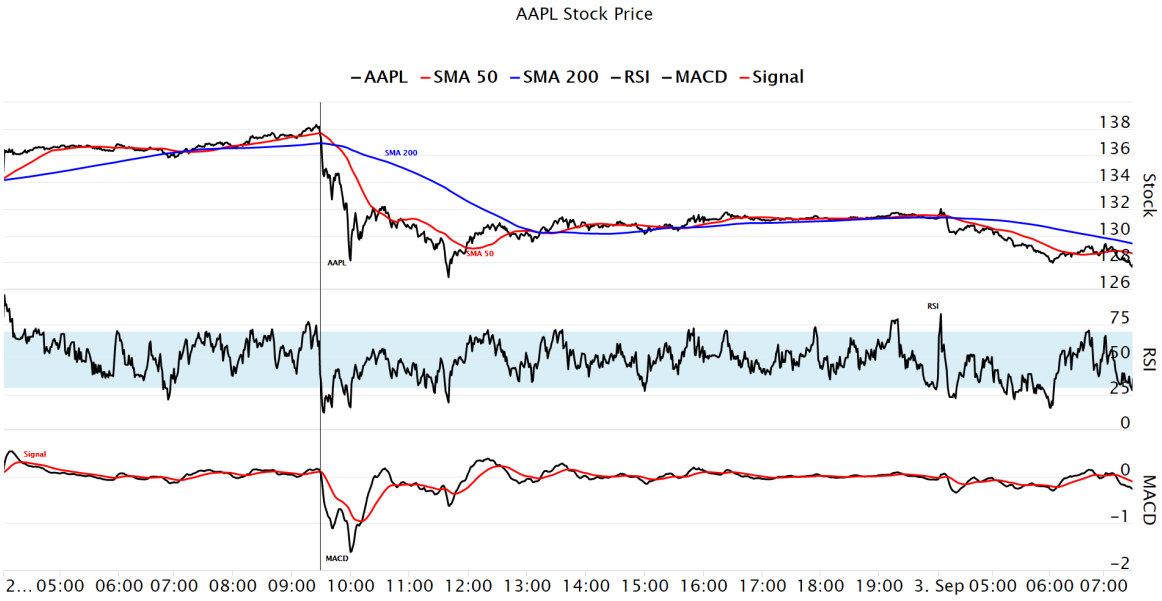
A fent részletezett indikátorokat érdemes górcső alá vetni, s a hatékonyságukat vizsgálni. Az indikátorok feltételeit felhasználva mértem a jelzések számát, illetve, hogy hány esetben növekedett az árfolyam 5, illetve 10%-al, s a hatékonyságukat 0-100-ig tartó skálán pontoztam. Az általam összegyűjtött 44 darab vállalatra lefuttattam a mérést és összegyűjtöttem ezeket egy táblázatban.




2.9. ábra. Az indikátorok hatékonyságának mérése.

A fenti ábráról az olvasható le, hogy önmagukban az indikátorok nem annyira hatékonyak, ezért érdemes őket csoportosan nézni. Ha megnézzük az első mérést, amely az MACD és a szignál függvényének keresztezését figyeli, akkor azt mondhatjuk el, hogy 5%-os emelkedésben kicsit alulteljesít, azonban a 10%-osban nagyon lemarad a többitől. Az RSI és az SMA közel egy szinten vannak, azonban ha elkezdjük őket keresztezni, akkor közel 30%-os emelkedést tapasztalhatunk. Ez abból fakad, hogy az ÉS kapcsolat szigorúbb feltételeket állít, ezért kevesebb lesz a jelzések száma.

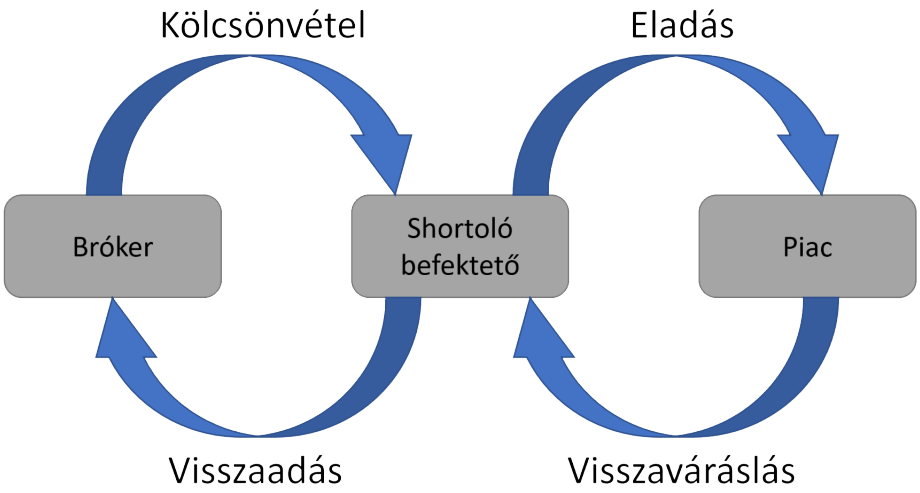
Az indikátorokkal az a legnagyobb probléma, hogy mire jeleznek az indikátorok, addigra már az árfolyam rég elindult pozitív vagy negatív irányba. Emiatt sosem fogjuk tudni a legjobb áron megvenni, illetve eladni a részvényt, ezért nagyon fontos a jelzések pontosítása.




2.10. ábra. A jelzések késése az Apple részvényén.

## 2.8. Részvény shortolása

A kereskedők minden lehetőséget megfognak ahhoz, hogy profitot tudjanak termelni maguknak. A következő technika, amit alkalmaznak is besorolható az árfolyamnyereséges kategóriába, azonban az indikátorokra tett hatása miatt külön dolgozom fel. A piacon lehetőség van arra, hogy kölcsön kérjenek részvényeket befektetők annak reményében, hogy az árfolyam csökkenni fog. Amikor megszerezte a részvényeket eladja, és amikor csökkenésnek indul a részvény kivárja a megfelelő időpontot, és utána visszavásárolja őket, majd visszaadja a részvényeket az eredeti tulajdonosának. Így a kereskedő a különbséget profitként könyveli el, ebből persze lejönnek a költségei a tranzakciónak, illetve bérlési költséget is felszámolnak érte. Ezt a folyamatot shortolásnak nevezik.




2.11. ábra. A shortolás folyamatának ábrája.

### 2.8.1. Short squeeze

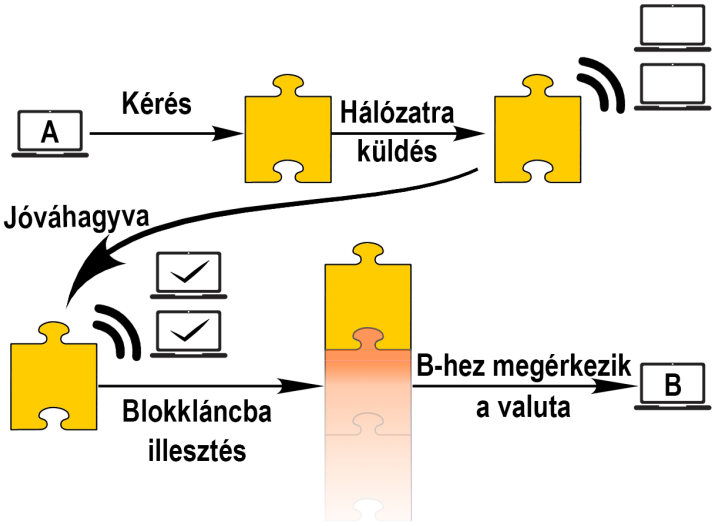
A short squeeze egy részvény értékének meredek emelkedése, amely elsősorban a részvények túlzott shortolásából adódik. Akkor következik be, amikor a részvények iránti kereslet túl nagy, a kínálat pedig túl kevés. Amikor a kereskedők elképzelésével ellentétes irányba, azaz felfelé mozdul el a részvény árfolyama, akkor, hogy csökkentsék a veszteségeiket elkezdik visszavásárolni a részvényeket. Amikor visszavásárolják a részvényeiket, akkor megemelkedik a részvény ára, ezzel még több shortoló befektető kényszerül visszavásárolni a részvényeit. A short squeeze általában azoknál a részvényeknél szokott előfordulni, ahol magas a bérleti költség, hiszen ez még nagyobb nyomást helyez a befektetőkre, hogy a pozíciójukat fedezzék.


2.12. ábra. Short squeeze: 5 perc alatt 230%-os emelkedés az AMC részvényén

## 2.9. A kriptovaluta

A napjaink egyik felkapott témája a kriptovaluta. Ezt az is mutatja, hogy már több mint ezer kriptovaluta közül választhatunk. A leghíresebb kétségek nélkül a Bitcoin, amely a szakdolgozat írásának pillanatában 55 000 $\pm$ 5 000$-ért vásárolható meg. Mi is az a kriptovaluta? A kriptovaluta egy digitális pénznem, amelyet kriptográfiai módszerrel védenek, így közel lehetetlen a hamisítása. A blokklánc (blockchain) technológia tartja fent ezt a védelmet, amelyet egy példán keresztül mutatnék be.

Tegyük fel, hogy kriptovalutát szeretne küldeni az **A**-val jelölt személy a **B**-vel jelölt személynek. **A** egy kérésben jelzi a szándékát, mely egy blokkba kerül beillesztésre. Egy blokk 1MB maximálisan, amely több mint 500 tranzakciót jelent. Ez a blokk minden résztvevő számára közvetítésre kerül, ők fogják validálni a tranzakciókat, és ha jóváhagyták, akkor a blokklánc végére illesztik a blokkot, ami ettől fogva kitörölhetetlen, hiszen akkor megváltozna a hash értéke, és a többi résztvevő visszadobná a blokkot. Ekkor már vége is van egy tranzakció feldolgozásának, átkerült a kívánt összeg a **B** számlájára. Tehát látható, hogy a hálózat védelme egészen hatásos, de miért éri meg bárkinek, hogy egy kriptovaluta tranzakcióit, integritását óvja?




2.13. ábra. Tranzakció végrehajtása blokklánc technológiával.

### 2.9.1. A kriptovaluta bányászata

A kriptovaluta bányászatán azt értjük, hogy a számítógépünket a kriptovaluta hálózathoz kapcsoljuk és a blokkok vizsgálatát végezzük el. Ez egy automatikus folyamat, csupán egy programot kell hozzá elindítani. A számításokhoz a videókártyát használják, hiszen ezzel a leggyorsabb elvégezni a matematikai képleteket. A Bitcoin úgy lett tervezve, hogy a blokkok 10 percenként érkeznek, és ha sikerül megoldani a hashelést, akkor a hozzáadott számítási kapacitás alapján szétosztásra kerül 6.25 BTC, amely körülbelül 105 millió forint értékű. Hogy minél nagyobb számítási kapacitást érjenek el egyre több videókártyát vásárolnak a bányászok, emiatt a készlet lecsökken, és az áruk is az egekbe emelkedik.


2.14. ábra. A Bitcoin árfolyamának alakulása [9].

# 3. fejezet

# Szoftveres technológiák

A következő fejezetben a szoftveres, illetve a gyakorlati függőségekről fogok írni. Ezeket a technológiákat találtam a legjobbnak, szem előtt tartva az egyszerűséget, az ingyenes hozzáférést, illetve a sokoldalú felhasználhatóságot. A felsorolásban leginkább webes technológiák szerepelnek, hiszen így könnyebben lehet több platformra is elérhetővé tenni a programot.

## 3.1. HTML

A HTML, avagy a HyperText Markup Language a W3C (World Wide Web Consortium) által támogatott, szabványosított leíró nyelv. Weboldalak szerkezeti leírására fejlesztették ki az 1993-as években, azóta több verzió is megjelent belőle, melyekben bővítették a készleteit. A HTML nyelv segítségével lehet elkészíteni a weboldalak vázát, melyet tovább szerkeszthetünk funkciók hozzáadásával, megjelenés beállításával. Egy HTML oldal három fő részből áll, HTML, HEAD és BODY részből.

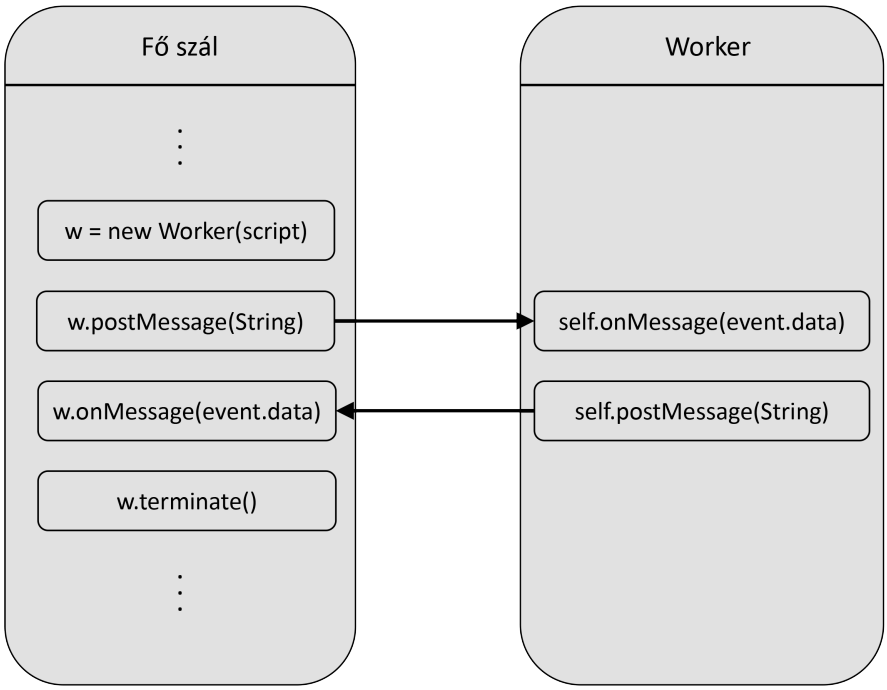
```
<!DOCTYPE HTML>
<html lang="hu">
    <head>
        <meta charset="UTF-8">
        <title>Oldal cime</title>
        <link rel="stylesheet" href="css/style.css">
    </head>
    <body>
        <h2>Ez egy pelda oldal</h2>
        <p>Ez egy bekezdes</p>
        <script src="js/script.js"></script>
    </body>
</html>
```

A szöveges fájl egy típusdefinícióval kezdődik, mely információ a böngészőnek szól, hogy tudja milyen formátumú. Ezután következhet a fejléc rész, mely a `<head>` tagek között található. Itt rögzíthetünk olyan paramétereket, amelyeket a böngésző nem fog megjeleníteni, például a szerző, a weboldal címe, vagy éppen a stílusleíró fálj elérési útvonala. Majd végül a törzs része jön az oldalnak, ami a `<body>` tagek között szerepel. Ez már a megjelenítendő részt tartalmazza, viszont például `<script>` tag is szerepelhet

benne, ami nem feltétlen jelenít meg tartalmat, viszont a weboldallal együtt töltődik be, nem pedig előtte, ami jobb élményt nyújthat a felhasználóknak.

## 3.2. JavaScript

A JavaScript egy programozási nyelv, melyet nagyon sok fontos alkalmazásban használnak manapság, elsősorban weboldalak fejlesztésére, de megjelenik a JavaScript a mobilalkalmazás fejlesztésben is. A JavaScript egy multiparadigmájú nyelv, mely támogatja az imperatív, deklaratív és objektumorientált stílusokat is, szintaxisát tekintve a Java nyelvhez hasonlítható. Osztályok helyett prototípusokat alkalmaz, ezzel megvalósítva az objektumorientált programozást. A JavaScript továbbá támogatja a párhuzamos feldolgozást az úgynevezett Web Worker technológián keresztül [11].




3.1. ábra. JavaScript Web Worker technológiájának működése.

- A Worker konstruktornak meg kell adni az elérési útját annak a JavaScript fájlnak, amiben a párhuzamosítandó kód szerepel.
- Az objektum postMessage függvényén keresztül lehetséges az üzenet küldése mindkét irányba.
- Az onMessage esemény figyeli, hogy küldtek-e üzenetet vagy sem.
- A terminate függvényhívással törölhetjük a workert a böngészőből.

## 3.3. CSS

A weboldal vázát, ha megszerkesztettük a HTML leíró nyelvvel, akkor az alapértelmezett stílusleírást fogja megkapni minden elem. Ahhoz, hogy a kreativitásunknak ne szabjanak határt, létrehozták a CSS stílusleíró nyelvet hat évvel a HTML megjelenése után. Ez egy olyan erőteljes eszközt ad a kezünkbe, amellyel bármilyen weboldal elkészíthető. A stílusok beállítására hivatkozni szükséges a módosítandó elemekre melynek egyszerűbb módjai:

1. elem (tag) kiválasztás

```
p {
    background-color: yellow;
}
```

2. egyedi azonosító (id) kiválasztás

```
#title { ... }
```

3. csoport (class) kiválasztás

```
.paragraphs { ... }
```

A stílusok megadásának három lehetséges megadási formája létezik, ezek érvényesülés sorrendjében a:

1. Külső (external) stílus

   Egy külön .css fájlban definiáljuk az elemek tulajdonságait. A CSS fájlt a HTML `<head>` részében szükséges importálni, melynek módja:

   ```
   <link rel="stylesheet" href="style/style.css">
   ```

2. Belső (internal) stílus

   A HTML dokumentum `<head>` részében is létrehozhatunk egy `<style>` elemet, melyben ugyanúgy kell megadni a stílusokat, mint a külső változatánál.

   ```
   <style>
   body{
       background-color: lightblue;
   }
   </style>
   ```

3. Sorközi (inline) stílus

   Közvetlenül az elemek style attribútumában is lehet rögzíteni a stílusokat.

   ```
   <h1 style="color: blue; text-align: center;">Ez egy cimsor</h1>
   ```

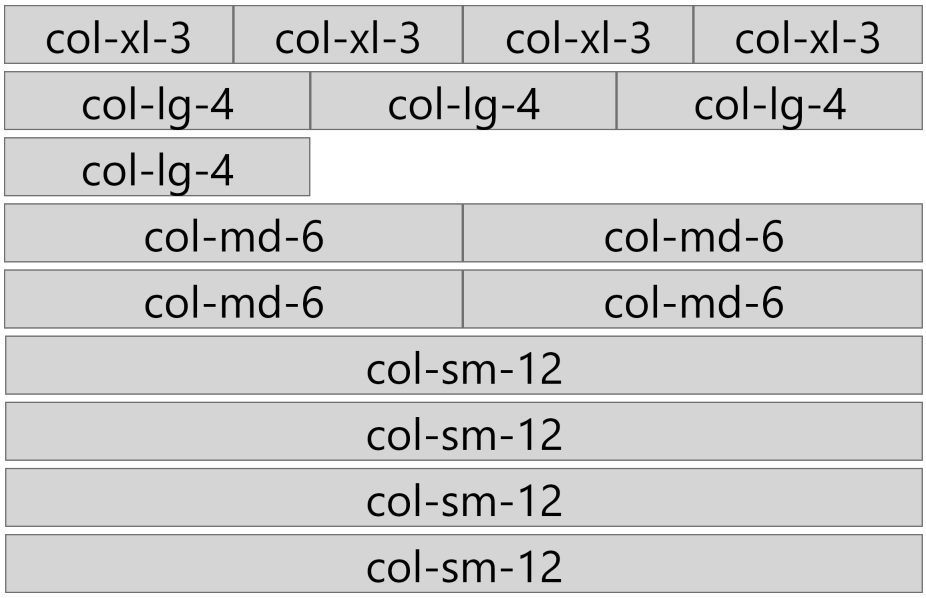
## 3.4. Bootstrap 4

Az internet fiatal korában többféle könyvtárat használtak interfészek készítéséhez, mely megnehezítette a karbantartását a weboldalaknak. Mark Otto, a Twitter egyik fejlesztője rukkolt elő azzal az ötlettel, hogy megalkot egy rendszert, mely kielégíti a kor követelményeit a webfejlesztés terén, így jelent meg 2011-ben a Bootstrap.

A Bootstrap egy ingyenes, nyílt forráskódú CSS keretrendszer, amely a front-end fejlesztést segíti azáltal, hogy támogatást nyújt reszponzív felületek létrehozásához. Elsősorban CSS alapú, de tartalmaz JavaScript-re épülő sablonokat is. Definiáltak például űrlapokat, gombokat, menürendszert és még sok mást. Ezeket a sablonokat úgy lehet alkalmazni, hogy a megfelelő class-szal ellátjuk az elemet. Például:

```
<button class="btn btn-primary btn-block"></button>
```

A Bootstrap az elemek csoportokba rendezéséhez konténereket definiált több méretben. Hat lehetőség közül lehet választani, ezek között az a különbség, hogy milyen képernyőszélesség alatt töltik ki az oldalt teljes szélességében. Ezeket a konténereket feldarabolhatjuk több részre, és így egy rácsszerkezetet hozhatunk létre a weboldalon. Tizenkét darab oszlop hozható létre maximálisan, viszont ezt úgy töltik ki a benne foglalt mezők, ahogy szeretnénk. Egy mezőnek meg lehet adni, hogy hány egységet töltsön ki a tizenkét oszlopból. Ezt az értéket az összes konténer méretre lehet rögzíteni, melyet a következő példán szemléltetek.

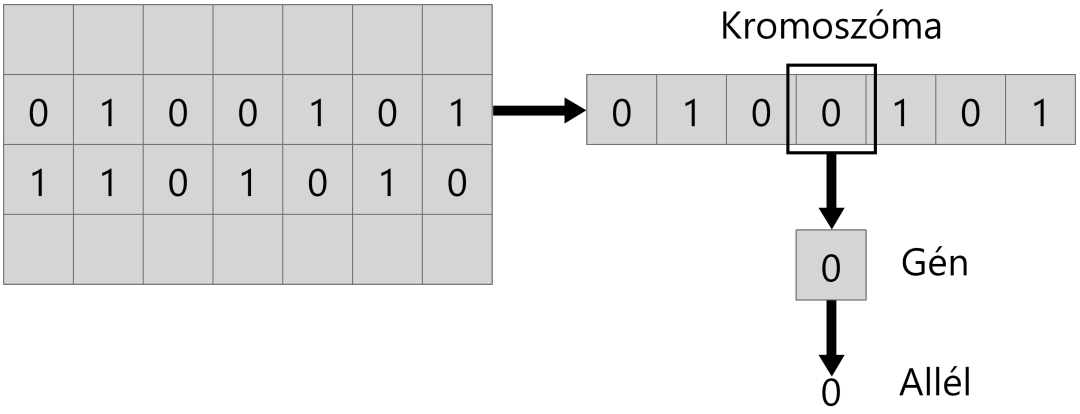



3.2. ábra. Bootstrap 4 rácsszerkezete.

Ha teljes méretben tekintjük meg a weboldalt, akkor egymás mellett elférnek a mezők, viszont minél inkább csökken a weboldal szélessége, annál több helyet vesz fel egy-egy mező, s elkezdi új sorba rendezni a fennmaradókat. A mezők azt a class értéket tartalmazzák a 3.2 ábrán, amivel beállíthatóak ezek az méretek [4].

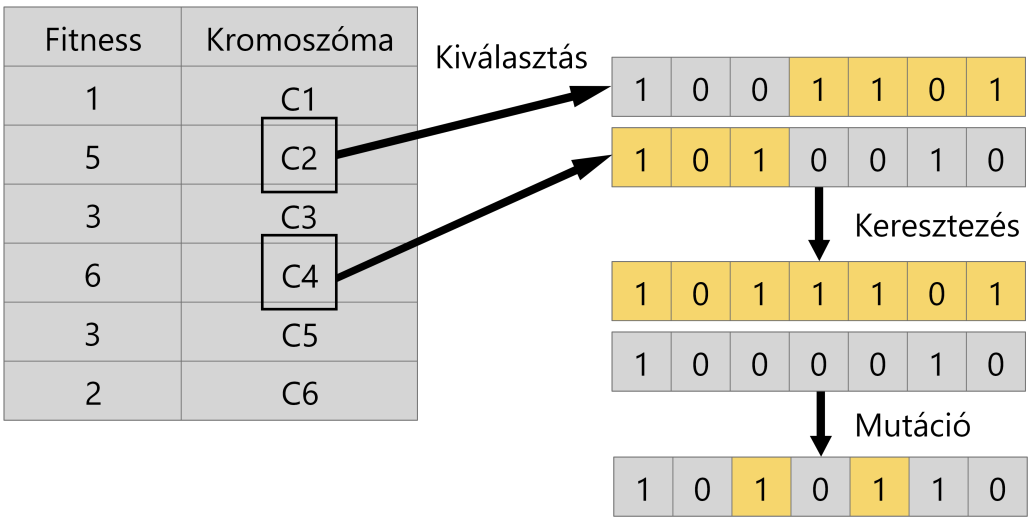
## 3.5. Genetikus algoritmus

A számítástechnikában a genetikus algoritmusok optimalizációs eljárások, melyek a biológia evolúciós elvekeit követik, mint a mutáció, keresztezés, és a kiválasztás. Abban az esetben megfelelő választás, amikor az optimum egy közelítése is elegendő a megoldáshoz, mert sok esetben előfordulhat, hogy nagyon sok munka lenne előállítani a tökéletes megoldást. A genetikus algoritmus építőelemei a kromoszómák, a fitness alapú kiválasztás és a biológiából átvett műveletek. A kromoszóma tipikusan egy bináris string formátumot vesz fel. A kromoszómákon belül minden egyes lokusznál (ahol a gén található) kettő lehetséges allél létezik: 0 és 1.




3.3. ábra. Populáció felépítése.

A kromoszóma tulajdonképpen a megoldás halmaz egy pontja. Ezeket a kromoszómákat dolgozzuk fel a genetikus operátorokkal. Azt, hogy mennyire elfogadható a megoldás, az $f(x) \rightarrow \mathbb{R}$ fitness függvény mutatja meg, mely a valós számok halmazába képez le. A biológiai operátorok közé tartozik a kiválasztás, a mutáció és a keresztezés. A kiválasztásnál a fitness függvény alapján választunk ki kromoszómákat. A keresztezés operátor két kromoszóma esetén egy-egy véletlen lokuszt választ ki, mely mentén két részre osztódnak a kromoszómák, s az utolsó szeletüket felcseréljük. Mutációkor a kromoszóma egy-egy génjében felcserélődik az allél.




3.4. ábra. Kromoszómákon végzett műveletek.

## 3.6. Alpha Vantage

Az Alpha Vantage egy adatszolgáltató cég, weboldal. Több nagyobb vállalattal partnerséget kötve jött létre ez az adattár, mely ingyenesen biztosít kutatási és egyéb célra napi, havi, vagy akár napközben rögzített részvény adatokat. Nemcsak cégek tőzsdei értékeivel foglalkoznak, hanem kriptovalutákra is található itt adat, amellyel kiegészülve széles spektrumúnak tekinthető a szolgáltatás. A használatához szükség van egy API key-re, mely azonosítja a felhasználót a lekérdezések során. Az ingyenes kulcs mellett korlátozásokat fogalmazott meg a cég, napi 500, illetve percenként 5 a maximum hívások száma, ami megengedett. Ez kibővíthető, vannak prémium opciók, viszont ez az adatokra semmilyen pozitív hatással nincsen.

Az Alpha Vantage API több lehetőséget is kínál, részletesen a napi kereskedéshez kapcsolódó információkat mutatom be. A **TIME_SERIES_INTRADAY** lekérdezés során lehetőség van 1-2 hónap adatait kinyerni, amely gyorsabb letöltést tesz lehetővé, illetve, ha csak kisebb mennyiségű adatra van szükség, akkor ezt célszerű használni. A lekérdezéshez több kapcsoló tartozik, melyeket részletezek a következő sorokban, vastagon szedve a kötelező paramétereket.

- **function**: itt adható meg, hogy melyik lekérdezési formát szeretnénk használni, ez esetben TIME_SERIES_INTRADAY.
- **symbol**: ez a kapcsoló adja meg a tickert, a részvény azonosítóját, például: GOOG, AAPL, TSLA.
- **interval**: két egymást követő adat között eltelt időt határozza meg, ez lehet 1, 5, 15, 30 és 60 perces.
- adjusted: ez a kapcsoló határozza meg, hogy korrigált legyen-e a részvény értéke vagy sem. Alapértelmezésképp ez igaz, hiszen, ha a részvényt például kettéosztják (split), az értéke hirtelen a felére csökken, s ha nem vagyunk tudatában annak, hogy ez történt, akkor tévesen hihetjük azt, hogy csökkent az érték, el kell adni a papírokat.
- outputsize: lehet compact, vagy full. A compact az alapértelmezett érték, így csak 100 értéket kapunk vissza az oldaltól. Ha a full kapcsolót használjuk, akkor a teljes méretű adatsort kapjuk vissza.
- datatype: kettő visszatérési forma közül lehet választani, a JSON és a CSV. A JSON elterjedt a webes alkalmazások körében, ezért ez az alapértelmezett, viszont ezzel szemben a CSV egyszerű, s kompatibilis a széles körben használt Excel programmal.
- **apikey**: a kapott kulcsot itt kell megadni.

Az én felhasználásomhoz egy bővebb adathalmazra van szükség, melyre egy másik függvény, paraméter meghívása szükséges a weboldalon, a **TIME_SERIES_INTRADAY_EXTENDED**. Ez a függvény visszatér két évre visszamenőleg az adatokkal, amelyet ők is ajánlanak szimulációkhoz, vizualizációhoz. Az előző függvénytől nem tér el sokban, kikerült a datatype kapcsoló, hiszen a CSV kevésbé memóriaigényes mint a JSON, s a helyére a slice kapcsoló került be, mellyel a különböző darabjait tudjuk letölteni az adatsornak. A két év minden hónapját külön tudjuk lekérdezni, példa a

használatra: slice=year1month1. Ennek a függvénynek a visszatérési fájlja a következő adatokat tartalmazza:

- Date: az részvény árának mintavételi dátuma,
- Open: a nyitóár,
- High: a maximum érték,
- Low: a minimum érték,
- Close: a záróár,
- Volume: a részvények mennyisége.

Egy példa lekérdezés az alábbiak szerint hívható meg.

```
https://www.alphavantage.co/query?
function=TIME_SERIES_INTRADAY_EXTENDED&symbol=IBM
&interval=15min&slice=year1month1&apikey=demo
```

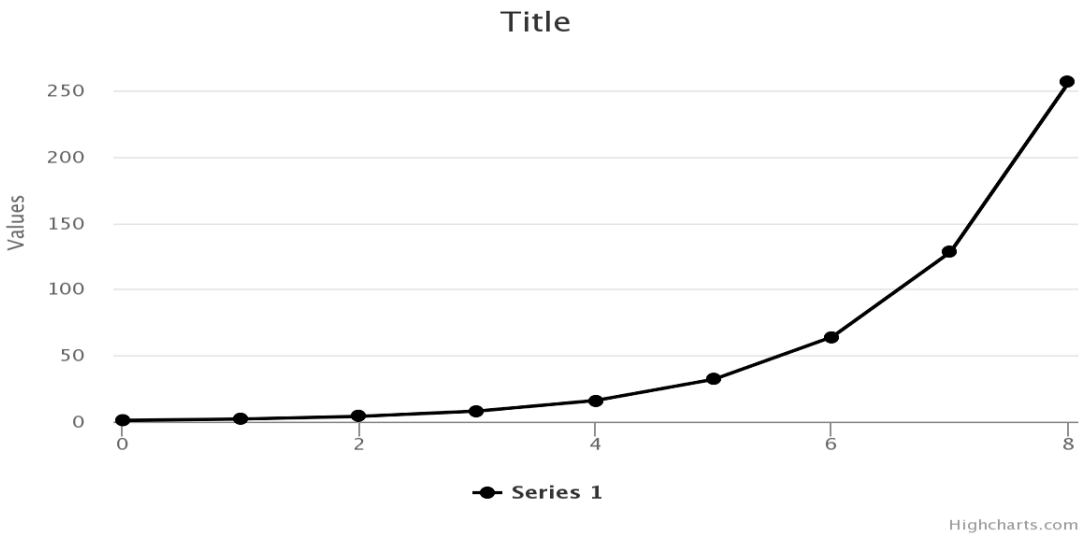
Paramétereit tekintve elmondható, hogy az IBM vállalat részvényeit adja vissza az elmúlt 30 napból, 15 perces időközökkel [1].

## 3.7. Highcharts

A szimulációk egyik fontos eleme a megfelelő grafikai szemléltetés. Sokkal érthetőbb lesz az eredmény, hogyha valamilyen ábrát, táblázatot, grafikont jelenítünk meg. Az elemzéseim megjelenítéséhez a Highcharts termékét választottam, a Stock-ot. Ez egy alapvetően fizetős fejlesztői könyvtár, azonban egyetemistáknak ingyenesen is rendelkezésre bocsátják, csupán egy egyszerű űrlapot kell kitölteni és máris küldik emailben a letöltési linket. A weboldalon rengeteg demo programot lehet megtekinteni, szinte mindenre található valamilyen példa. A táblázat megadása objektum formában történik meghatározott paraméterek alapján. Például egy egyszerű vonaldiagram megadható a következő módon.

```
Highcharts.chart('container', {
    title: {
        text: 'Title'
    },
    series: [{
        data: [1, 2, 4, 8, 16, 32, 64, 128, 256],
        pointStart: 1
    }]
});
```

A függvényhívás első paramétere lesz az a `<div>` ahova fogja kirajzolni a grafikont, a következő paraméter pedig egy objektum, amiben különféle paramétereket adhatunk meg. Jelen esetben két paraméter szerepel benne, a title, vagyis a címe a grafikonnak, illetve a series, amiben az adatokat szükséges megadni. Ha szeretnénk egy újabb vonalat ugyanerre az ábrára rajzolni, akkor a series tömbhöz egy újabb objektumot szükséges hozzáadni [5].




3.5. ábra. Vonaldiagram a Highchart környezetben.

Amennyiben részvényspecifikus ábrákat szeretnénk megjeleníteni, arra is lehetőséget ad a Highcharts. Kis módosítással létrehozhatunk egy gyertyadiagramot, ahol egyszerre több adatot is megtudhatunk a részvény áráról.


3.6. ábra. Gyertyadiagram a Highchart környezetben.

```
Highcharts.stockChart('container', {
    title: {
      text: 'AAPL'
    },series: [{
      type: 'candlestick',
      name: 'AAPL',
      data: data}]
});
```

# 4. fejezet

# TradeMe

A TradeMe egy olyan szimulációs szoftver, mely bemutatja, hogy a részvényekkel történő kereskedés lehetséges, kezdőknek adhat iránymutatást milyen feltételek mellett vásárol a program, és a megszerzett tudást átültethetik a gyakorlatba. A cél, hogy olcsón tudjon vásárolni, és később többszörös áron el tudja adni a tulajdonrészét, viszont ez nem ennyire egyszerű, a piacot sok faktor befolyásolja. A program ezeket a nehézségeket hivatott kiküszöbölni a technikai analízis eszközeit felhasználva. A szimulációhoz a limitációk miatt csak NASDAQ alá tartozó, amerikai vállalatok adatai tölthetők le.

## 4.1. Adatkinyerés

A különböző tickerekhez tartozó adatok kinyerése nem egyszerű feladat, hiszen a tőzsde világában ezek az adatsorok kincset érnek. Nagyon sok szolgáltató az adatbázisát kisebb-nagyobb összegekért árulja, vagy valamilyen korlátozáshoz köti a letöltést, ezért az egyik fő feladatom, hogy olyan adatbázishoz férjek hozzá, mely ingyenes, illetve egy API-n keresztül lehetővé teszi a letöltést. Ez azért fontos, mert programozói eszközökkel könnyedén automatizálható ez a folyamat, illetve szükséges is, hogy az alrendszerek közötti egységességet biztosítani tudjam. Több lehetőséget is számításba vettem, viszont később vagy fizetőssé tették a szolgáltatást, vagy nem lett volna egyszerű a letöltési folyamat.

### 4.1.1. Szoftveres támogatás

Ahhoz, hogy az Alpha Vantage oldalról le tudjak tölteni adatokat, szükség van egy API kulcsra, amit a weboldalon egy egyszerű regisztrációt követően ki is küldenek. Ezután már csak annyi dolgom van hátra, hogy megfogalmazzam a lekérdezést, melyre az Alpha Vantage alfejezetben bővebben kitértem.

Java nyelven készült egy kiegészítő program, amely elvégzi a megfelelő lekérdezések meghívását, letölti a 24 darab szeletet, ezeket összefűzi, majd a weboldal részéhez hozzácsatolja. Erre azért volt szükség, mert egy-egy részvény adatainak a letöltése több percet vehet igénybe, s ezzel a megoldással a háttérben futtatható a letöltés, míg a weboldalon elemezzük a részvényeket. Mivel az Alpha Vantage oldalon az ingyenes tagság korlátozásokkal jár, mégpedig azzal, hogy öt hívás lehetséges percenként, illetve egy nap maximum 500, ezért a programban szükséges rögzíteni mikor történtek a kérések. Ezt egy egyszerű listával oldottam meg, aminek `long` típusú elemei vannak. Minden letöltés után lementem az aktuális időt, s hogyha a lista mérete elérte az ötöt, akkor

megnézem, hogy a legrégebbi bejegyzés óta eltelt-e 60 másodperc. Amennyiben igen, akkor a listát eltolom balra eggyel, egyéb esetben várakozik a program.

Egy lekérdezés implementációja az alábbi formában látható.

```
String url = "https://www.alphavantage.co/query?function=" +
    "TIME_SERIES_INTRADAY_EXTENDED&symbol=" + symbol +
    "&interval=1min&slice=" + date +
    "&apikey=" + apikey;
URL website = new URL(url);
ReadableByteChannel rbc = Channels.newChannel(
    website.openStream());
FileOutputStream fstream = new FileOutputStream(
    dlDir.getAbsolutePath() + "\\" +
    symbol + "-" + date + ".csv");
fstream.getChannel().transferFrom(rbc, 0, Long.MAX_VALUE);
fstream.close();
```

Létrehozok egy url változót, ami a lekérdezést tartalmazza. A változóban három dinamikus rész szerepel: a symbol, a date, és az apikey (bővebben a 3.6 fejezetben). Majd egy csatornát hozok létre a weboldal és a programkód között. Ezt a csatornát megkapja a FileOutputStream, amely ki tudja végül írni a lemezre a webszervertől kapott adatokat. Egy **for** ciklussal végig iterálok két év hónapjain, s így kapok 24 darab csv fájlt, amely további feldolgozásra szorul.

A második lépésben a program beolvassa egyesével a fájlokat, majd egy csv fájlba kimenti az összesnek a tartalmát. Mivel minden fájl tartalmazott fejléc részt, ezt kivágom belőlük és egy fejléc részt rögzítek a fájl első sorában.

A harmadik és utolsó lépésben a program elmozgatja abba a mappába az összefűzött állományt, amiből a weboldal alapértelmezetten kiolvassa őket, illetve az ideiglenes fájlokat törli a lemezről.

## 4.2. Részvény adatok feldolgozása

A weboldal számára előkészített részvény adatokat valamilyen struktúrába szükséges rendezni, hogy később fel lehessen használni az adatokat. A részvény adatokat a program a **stocks** mappából olvassa ki a következő módon.

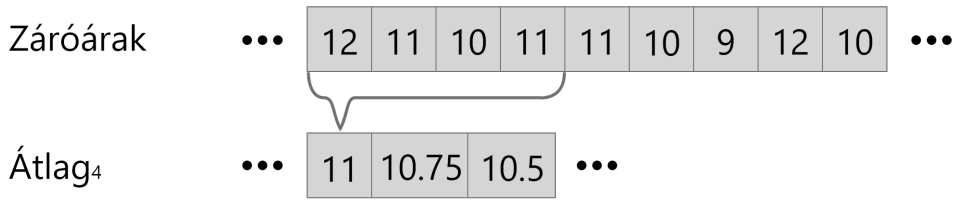
```
$(data).find("td > a").each(function () {
    let file = $(this).attr("href").substr(1);
    file = file.substr(file.indexOf("/") + 1, file.lastIndexOf("/") -
    file.indexOf("/") - 1);

    if (file == ".." || file == "")
        return;

    tickers.push(file);
    $("#stocks").append("<option>" + file + "</option>");
});
```

A beolvasott sorokon végig iterál a program és kiválogatja belőle a dátumokat és a hozzájuk tartozó záróárakat. Mivel az adatsorok több százezer bejegyzést is tartalmazhatnak, ezért úgy döntöttem, hogy külön tömbben tárolom a dátumokat és a záróárakat. Ez azért is fontos, mert később szükség lesz a grafikon készítésénél a dátumra külön, hiszen hozzá kell rendelni minden adatot a megfelelő időponthoz.

## 4.3. Statisztikák számítása

A kereskedésnél a technikai elemző stílust választottam, amely indikátorokat is vizsgál, és az ismereteit összegezve hoz döntést a részvényeiről. Az én szoftverem is az indikátorokat használja a vásárlás/eladás döntés meghozatalához, viszont az elkészítésüknél több problémába is ütköztem.

A statisztikai számításokban megjelenik a periódus paraméter, mely meghatározza hány adatból kalkuláljon az algoritmus egyet. Ha egyszerűen két `for` ciklust készítünk hozzá nem törődve az iránnyal, hamis eredményt kaphatunk.




4.1. ábra. Átlag számítása 4-es periódus értékkel előrefelé.

Hogyha előrefelé haladunk a ciklussal, akkor azt tapasztaljuk a futtatásnál, hogy hitetetlen összegeket tud gyűjteni a program, és ez azért van, mert jövőbeni adatokat veszünk figyelembe. Ha történik egy hirtelen irányváltás a részvény árában, akkor az indikátorok ezt mindig később fogják jelezni, ahogyan a 2.4-es ábrán látható. Ezért, ha előrefelé haladunk, akkor ezeket a jelzéseket is előrébb hozzuk, ami életszerűtlen. A szimulációs környezet miatt, illetve a futási idő javítása érdekében a statisztikák előre elkészülnek, viszont a való életben úgy frissülnek az indikátorok, ahogyan érkeznek az adatok.

Második probléma, ami előbukkant a tesztelések során, az a futási idő. Összesen 8 darab indikátort számít ki a program, amelyből a legegyszerűbbet emelem ki, az SMA-t. Az SMA-nak a számításához két `for` ciklus szükséges, ami összesen $((\text{adatmennyiség} - \text{periódus} - 1) * \text{periódus})$ számítást végez el, s az alábbi módon implementáltam.

```
function WcalcSMA(arr, end, period) {
    let tmp = [];
    for (let point = 0; point < end; point++) {
        let sum = 0;

        if (point < period - 1) {
            tmp.push(0);
        } else {
```

```
            for (let i = point; i > point - period; i--) {
                sum += arr[i];
            }
            tmp.push(parseFloat((sum / period).toFixed(6)));
        }
    }
    return tmp;
}
```

Ez nem nagy feladat a processzornak, viszont érdemes megfontolni az iteráció helyét. Az árfolyamok nagysága miatt érdemesebb a ciklust az adott statisztikát számító függvénybe helyezni, mert így egy függvényhívással elvégezhető egy indikátor számítása. Ezzel szemben, ha egy ciklusban hívom meg az összes függvényt, amik egy pontot számítanak ki, akkor a hatalmas mennyiségű függvényhívás lelassíthatja a futást.

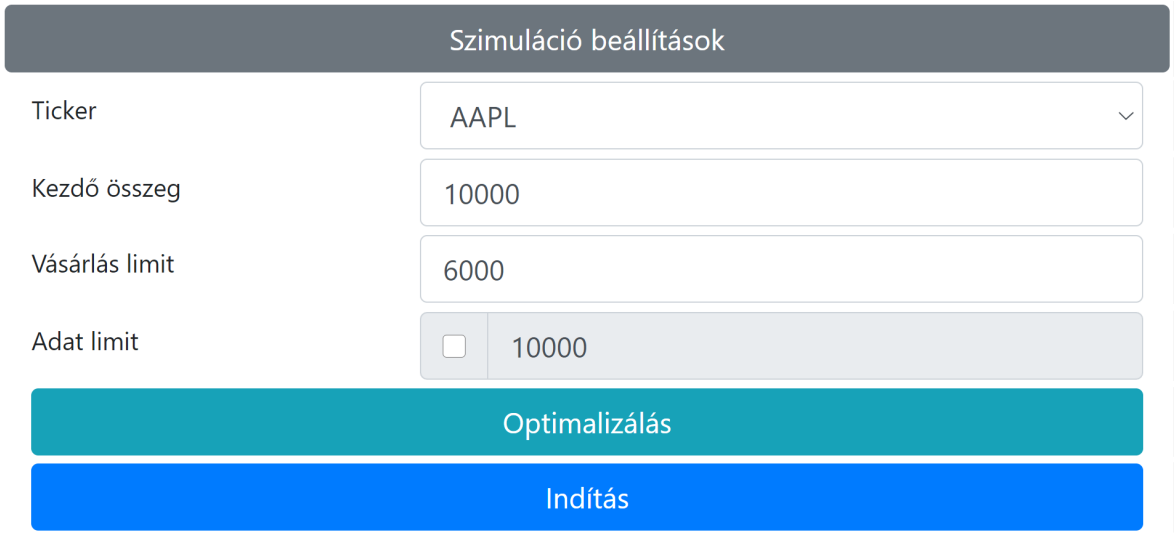
A statisztikákat külön szálon számítom ki, hogy a felhasználó számára ne okozzon negatív élményt, ne fagyjon le addig a grafikus felület kezelője. Egy statisztikának a kiszámítása az alábbiak szerint néz ki.

```
function createWorker(i) {
    return new Promise(function (resolve) {
        let w = new Worker('/js/worker.js');

        w.postMessage(i);
        w.onmessage = function (event) {
            w.terminate();
            resolve(event.data);
        };
    });
}

message = {type: "RSI", data: [closePrices, endPoint, 14]};
promises.push(createWorker(message));
```

Létrehozok egy új Workert, aminek egy üzenetet kell küldeni, amit az ő feldolgozója értelmezni tud. Meg kell adni a típusát az indikátornak, illetve a paraméterlistáját, ez esetben a záróárak tömbjét, a feldolgozandó elemek mennyiségét, illetve az indikátor periódusát. Ez alapján meghívja a megfelelő függvényt, ami egy tömbben összesíti az eredményeket, és ezt visszaküldi a csatornán (3.1-es ábrán szemléltetve). A visszaérkező üzenetek olyan sorrendben jönnek, mint amilyen sorrendben küldtem őket, ezért nem kell ügyelni az esetleges keveredésekre.

## 4.4. A kereskedés menete

A weboldal az alábbi felülettel fogad megnyitáskor. A program automatikusan felismeri a helyesen elhelyezett részvényeket és betölti, s egy legördülő menüből választhat a felhasználó közülük. A kezdő összeg mezővel beállítható, hogy mennyi pénzzel kezdjen el gazdálkodni a program. Alapértelmezettként 10 000$-t adtam meg, hiszen például az Interactive Brokers cég minimális számlanyitási összegét így állapították meg, azonban eltérő lehet cégenként [14]. A vásárlási limit beállítás arra való, hogy korlátozza az egy

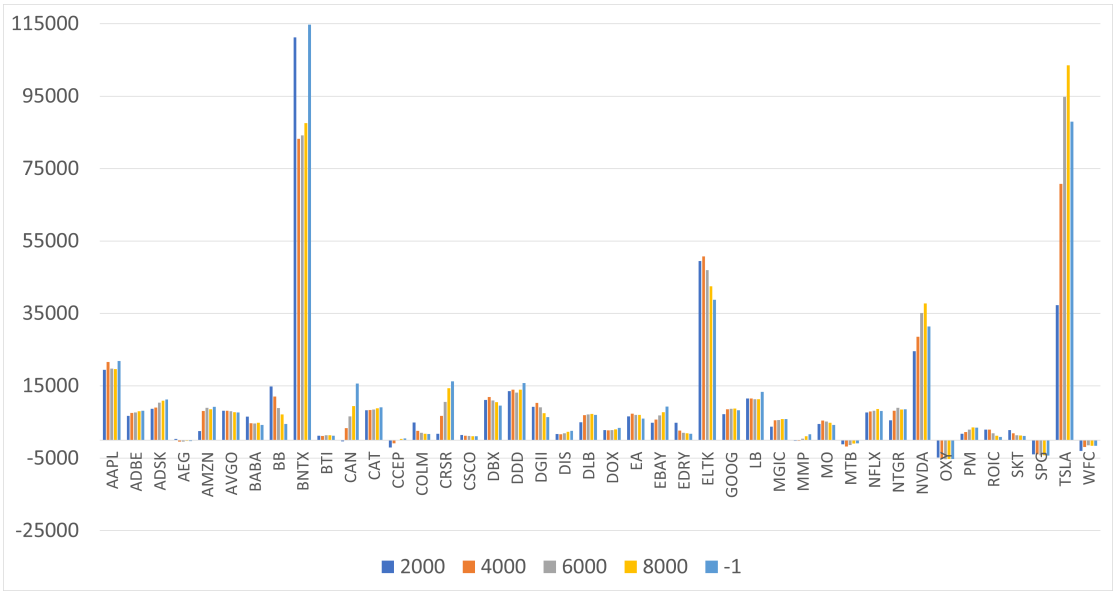



4.2. ábra. A webes felület nyitóoldala.

tranzakcióra eső elkölthető pénz mennyiségét. Létezik egy joker lehetőség, ha $(-1)$-et ad meg a használó, akkor mindig elkölti az összes pénzt részvényekre. Az adat limit paraméter leginkább arra használható, hogy leválasszon a program $x$ darab záróárat, így kis adatbázissal gyorsabb lekérdezést, illetve gyorsabb grafikont lehet elérni.

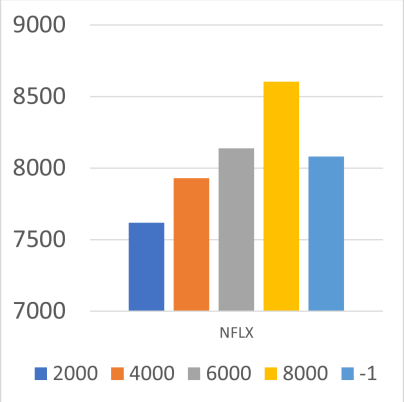
### 4.4.1. Vásárlási limit beállítása

A vásárlási limit egy olyan beállítás, ami nagyban befolyásolhatja a program vásárlási szokásait. Meghatározza, hogy mekkora profittal zárja a kereskedést, hogy mennyi tranzakciót végez. Ezt az adatot érdemes valamilyen módszerrel az optimumhoz közelíteni, hiszen ekkor keres a legtöbbet vele a program. Az alábbi ábrán látható, hogy a különböző részvények a 10 000$-os induló összegüktől mennyivel térnek el a szimuláció végére. Ha megnézzük például a NFLX, azaz a Netflix eredményeit, akkor azt láthatjuk,




4.3. ábra. A részvények profitjai vásárlási limit függvényében.

hogy egészen közel vannak az eredményei egymáshoz, a maximumát a 8 000-es limitnél éri el (4.4-es ábra). Mindegyik beállításnál profitot termelt, azonban vannak olyan




4.4. ábra. A Netflix profitja különböző vásárlási limitekkel.

részvények is, amik a limit növelésével a profit, illetve olyanok is, amik a veszteség irányába változnak. Összességében az mondható el, hogy a profitok összege a maximumát a $(-1)$ esetben éri el, azaz amikor az összes pénzét elkölti a program egy-egy vásárlásnál (4.3-es ábra).

### 4.4.2. Kereskedési feltételek

A szakdolgozatom egyik legnehezebb része a vásárlási, illetve az eladási feltételek meghatározása volt. Az indikátorok megléte még önmagában nem garantálja a sikert, meg kell mondani az algoritmusnak, hogy melyek azok az alakzatok, amelyek bebiztosítják a sikert. Összesen 5 féle alakzatot definiáltam a programban vásárlási és eladási célra, amelyekek súlyozását utána a genetikus algoritmus segítségével optimalizáltam. A három legmeghatározóbb alakzat között szerepel az MACD növekedése és csökkenése. Ekkor 20 periódussal nézem visszafelé az adatokat, és hogyha nem tapasztalok egymás után kétszer csökkenést, akkor a függvény irányát felfelé haladónak tekintem. Ha teljesül a növekedés feltétele, vagy az MACD éppen 0 fölötti értéken szerepel, akkor a vásárlást eldöntő változóhoz hozzáadom a súlyozási értékét az alábbiak szerint.

```
if (macdArr[i] > 0 || macdRising(i)) {
    decision.buy += mutant[4];
}
```

A genetikus algoritmus ezt a jelzést 9.15 pontra értékelte a maximum 10-ből.

A következő mutató, amit szeretnék megemlíteni az RSI-vel van összefüggésben. Az RSI-nél két határ szoktak nézni, a jelzésnél a 30-as határt vizsgálom éppen, a következő módon.

```
if (rsiArr[i] < 30) {
    decision.buy += mutant[6];
}
```

Ha az RSI 30 alá menne, akkor alulértékelt a részvény, tehát érdemes megfontolni a vásárlást. Az optimalizáció során 6.35 pontot kapott, azaz a második leghasznosabb indikátor a gyűjteményből.

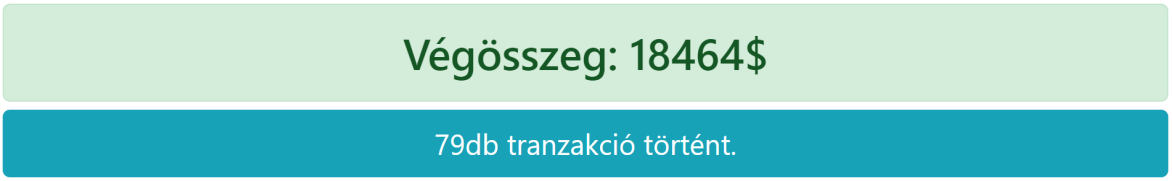
A harmadik legtöbb pontot kapott indikátor már az eladási jelzések közé tartozik. Azt vizsgálja, hogy az MACD negatív szám-e, vagy csökkenést mutat-e az MACD. Ez az alakzat az elsőnek a megfordításából adódik, azaz a következőképp írható le.

```
if (macdArr[i] < 0 || !macdRising(i)) {
    decision.sell += mutant[5];
}
```

A következő felsorolásban a többi általam implementált jelzést foglalom össze röviden.

- Amikor a $SMA_{200}$ felülről metszi az $SMA_{50}$-et, akkor aranykeresztről beszélünk, ellentétes esetben halálkeresztről, előbbi a vásárlási, utóbbi az eladást jelző alakzat.
- Ha az MACD keresztezi a szignálját felülről, akkor vásárlási, egyébként eladási jelzőként szolgál.
- Az RSI-nél egy szigorúbb feltételt is szerepeltettem a 20-as és a 80-as határokra, hogy erősítsem az indikátor jelzéseit nagyobb kilengés esetén.

Ezeket az adatokat a **decision** objektumban összegzem. Ha a **sell**, vagy a **buy** paramétere nagyobb lesz mint 10, akkor az annak megfelelő függvényt meghívom. Ha végig fut a program az összes értéken, akkor visszatérek a végösszeggel, s ha normál módban van a program, akkor kiírja a végeredményt. Ha profitot termelt a program, akkor zöld színnel, ha bukott, akkor piros színnel jelenítem meg a végösszeget. A végösszeg nem csak a profitot tartalmazza, hanem a kezdőösszeget is. Az alábbi képen látható, hogyan néz ki egy végállapot. A tranzakciós tájékoztató egy interaktív elem, gombnyomásra




4.5. ábra. Pénztárca tartalma, illetve a tranzakciók száma a szimuláció végén.

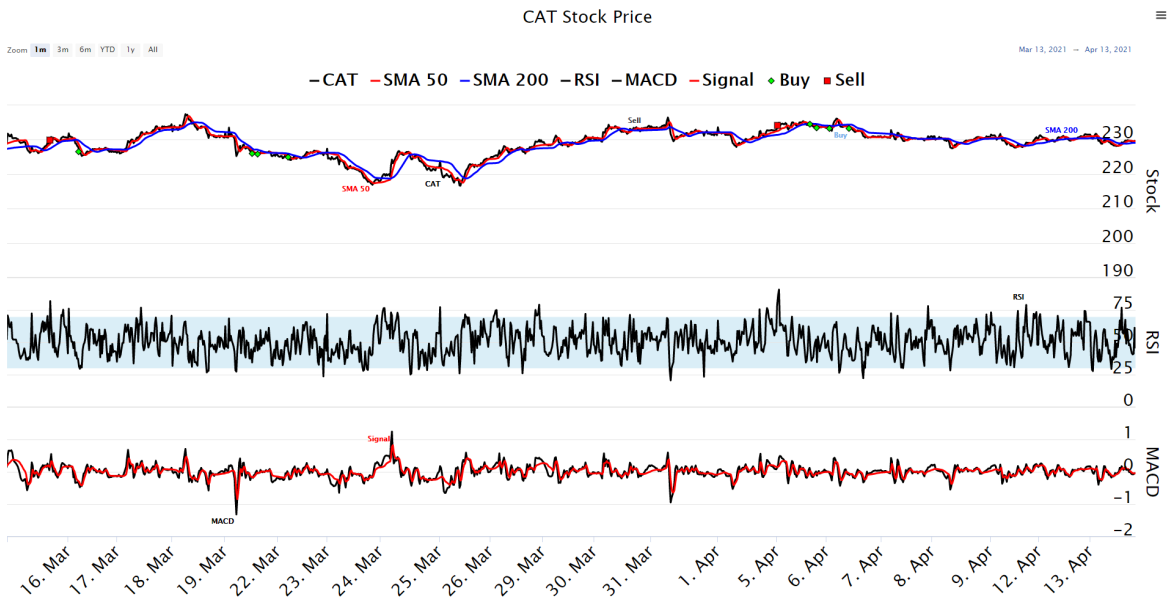
| # | Dátum | Típus | Darab | Záróár | Összeg |
|---|---|---|---|---|---|
| 1 | 2019. 02. 19 15:49 | Vásárlás | 46 | 128.526419 | 5912.22 |
| 2 | 2019. 03. 06 11:53 | Vásárlás | 31 | 127.987465 | 3967.61 |
| 3 | 2019. 06. 06 15:35 | Eladás | 77 | 117.84967 | 9074.42 |
| 4 | 2019. 06. 07 13:50 | Vásárlás | 50 | 118.4885 | 5924.42 |
| 5 | 2019. 06. 26 08:11 | Eladás | 50 | 128.412812 | 6420.64 |

4.6. ábra. A szimuláció végén generálódó tranzakciós lista.

lenyílik, s kiírja az összes tranzakciót, ami történt (4.6. ábra). Látható erről az öt adatról is, hogy a vásárlási limitet 6 000-re állítottam be, hiszen az első és negyedik vásárlás nem lépte túl ezt a mértéket. Továbbá, ha megnézzük az adatokat, akkor azt mondhatjuk, hogy az első eladáskor 805$ deficit keletkezett, azonban a következő tranzakció páros on már részvényenként 10$ hozamot ért el a program. A következő kódrészlet írja le azt, ahogyan megjelenik a tranzakciós lista.

```
$("#transactionsTable").append(
    '<tr><th scope="row">' + (counter++) + '</th>' +
    '<td>' + formatDate(new Date(tr.date)) +
    '<td>' + ((tr.type == "buy") ? 'Vasarlas' : 'Eladas') +
    '<td>' + tr.amount +
    '<td>' + parseFloat(tr.stock_price.toFixed(6)) +
    '<td>' + parseFloat((tr.stock_price * tr.amount).toFixed(6)) +
    '</tr>');
```

Ha tovább görgetünk a weboldalon, akkor a legalján egy grafikon található. A grafikonon megjelenítem a záróárakat, az 50, illetve 200 periódusú SMA-kat, az RSI indikátort a 30-as és 70-es határértékeivel, az MACD-t és a szignál függvényét, végezetül a vásárlási és eladási pontokat. A grafikonon megjelenő adatok ki és bekapcsolhatók a menüsor közepén található feliratokkal. Az adatok mennyisége miatt érdemes kikapcsolni addig bizonyos függvényeket, míg navigálunk a grafikonon, hiszen nagyon erőforrásigényes tud lenni nagy adatsorokon.




4.7. ábra. A szimuláció végén generálódó grafikon.

## 4.5. Legjobb kromoszóma megtalálása

A szimuláció önmagában haszontalan, hiszen az indikátorok megfelelő súlyozását ki kellene találni, vagy ha szerencsénk van egyből a legjobbat találjuk meg, azonban ennek kicsi az esélye. Ennek a kiküszöbölésére vezettem be a genetikus algoritmust, mely optimalizálja nekem a problémát.

A kezdeti populáció 10 elemű és 11 darab gént tartalmaznak, amely megegyezik a feltételek számával a kereskedésnél. Minden gén inicializálásra kerül, egy 0 és 10 közötti számot kapnak kezdeti értéknek. Ezután meghívom a **newGeneration** függvényt, mely először kiszámítja a generált populációra a fitness értékeket. A fitness értékek a következő számításon alapszanak.

```
function calcFitness(start_money, end_money) {
    return end_money / start_money;
}
```

Miután kiszámította a fitness értékeket, megkeresem közülük a maximumot, hiszen a legjobbat nagyobb valószínűséggel szeretném megtartani. Ezért a következő szakaszban végig iterálok a populáción, és a fitness függvény alapján sokszorozom a kromoszómákat. Minél nagyobb egy kromoszóma fitness értéke, annál többször szerepel a gyűjteményben. Majd ebből a gyűjteményből véletlenszerűen kiválasztok két kromoszómát, és keresztezem őket szimmetrikusan.

```
function crossover(a, b) {
    let len = a.length;
    let midpoint = Math.floor(Math.random() * len);

    let child = a.slice(0, midpoint).concat(b.slice(midpoint, len));

    return child;
}
```

A kapott kromoszómára még egy mutáló függvényt meghívok, ekkor lesz kész egy új eleme az új populációnak.

```
function mutate(child) {
    return child.map(e =>
        Math.random() < 0.1 ? getRandomNumber() : e
    )
}
```

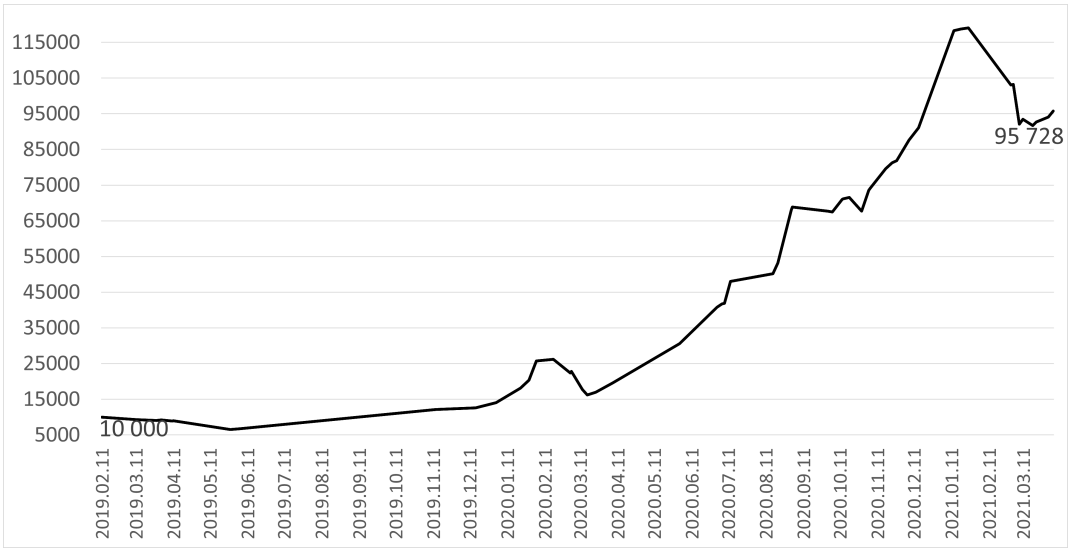
Miután meglett az új populáció, újra meghívom a függvényt, és kezdődik előről a ciklus. Mindaddig fut, amíg el nem éri a 100 iterációt úgy, hogy nem javult semennyit a fitness függvény.

A végeredményt manuálisan szükséges eltárolni a **weights.js** fájlban, viszont már alapból a legjobb fitness függvényű kromoszóma szerepel benne a letöltött adatokon vizsgálva. Amennyiben szeretnénk saját magunk elkészíteni az optimalizálást, abban az esetben ki kell választani egy részvényt, és az optimalizálást gombot megnyomni. Ekkor lefut a genetikus algoritmus, majd kiírja a fitness függvény értékét. Ezt leokézva a következő ablakban a letöltés indul el, a weboldal mappáján belül a **js** mappába szükséges menteni.

## 4.6. A tőke alakulása a szimuláció alatt

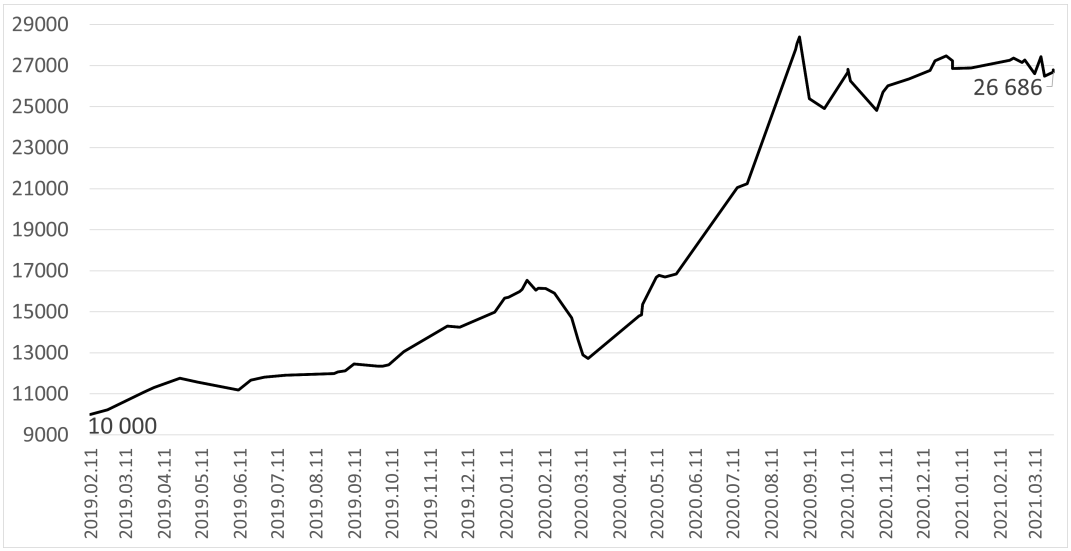
A kereskedés ideje alatt érdekes információkat tudhatunk meg abból, hogyan változik a tőke összege. Nem mindegy, hogy elbukjuk az induló tőkét, és visszaszerezzük, vagy éppen egész jó profit volt végig és a végén elbuktuk az egészet. A következő grafikonokon három részvényt mutatok be: egy jól, egy közepesen, és egy rosszul teljesítő részvényt.

A jól teljesítő részvényhez a Tesla részvényét választottam. Mint látható, 10 000$-ból 95 728$-ig jutott el. Eleinte lefelé indult, a veszteség irányába, minimum értéke 6 480$ volt, azonban hamar megfordult a görbe és meredeken döntötte a profitokat.



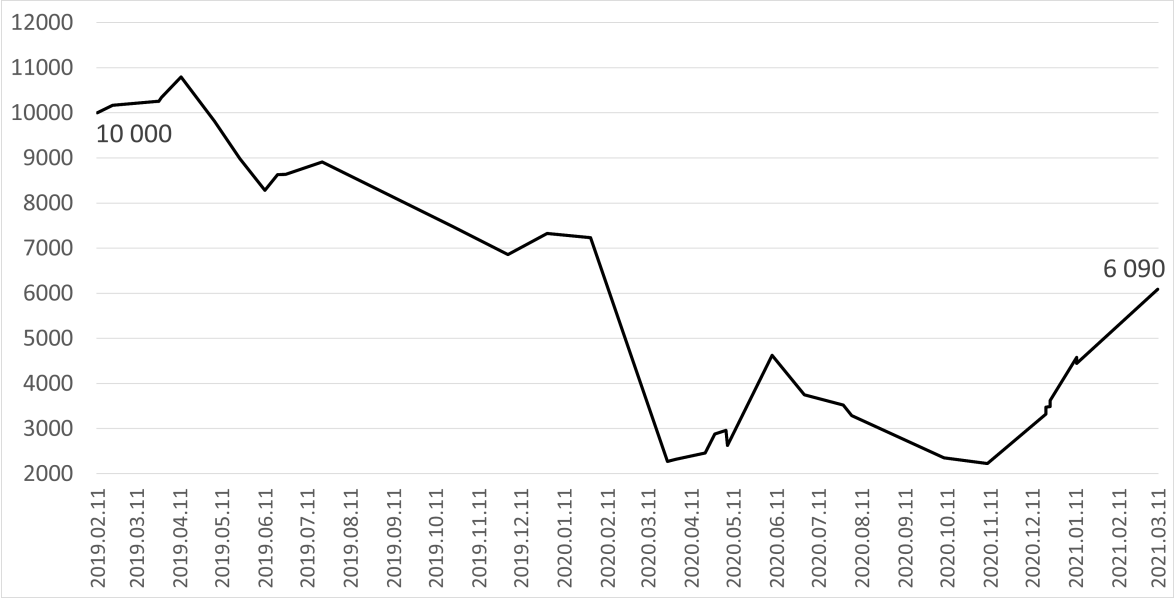

4.8. ábra. A tőke ingadozása a Tesla részvényén.

A következő részvény, melyet vizsgálat alá vontam, az az Apple részvénye. 26 686$-os végösszegével átlagosnak mondható, ha az összes részvényt tekintjük (vizsgált részvények a 4.3 ábrán). Ennél a részvénynél nem tapasztalható negatív szakasz, azaz sosem ment 10 000$ alá.




4.9. ábra. A tőke ingadozása az Apple részvényén.

Az utolsó részvény a listából a rosszul teljesítő kategóriába esik, az OXY részvénye. Egy rövid felfelé ívelés után folyamatos bukás jellemzi a görbéjét. Később próbált rendeződni az tőke, elképzelhető, hogy egy év múlva visszatérne 10 000$ fölé.




4.10. ábra. A tőke ingadozása az OXY részvényén.

Ha megnézzük ezeket a részvényeket, egy közös pontot találhatunk bennük. Március 16-án a részvények nagy része hatalmasat csökkent, miután a világjárvány komollyá vált. 1987-óta nem történt ekkora esés a részvényeknél. Szerencsére sikerült kilábalni a gödörből, azonban látható, hogy az automata profitját is sikerült lecsökkenteni. A következő fejezetben olyan tényezőkről szeretnék szót ejteni, amiket az automata nem tud időben felfedezni, és nem garantált a nyereségessége.

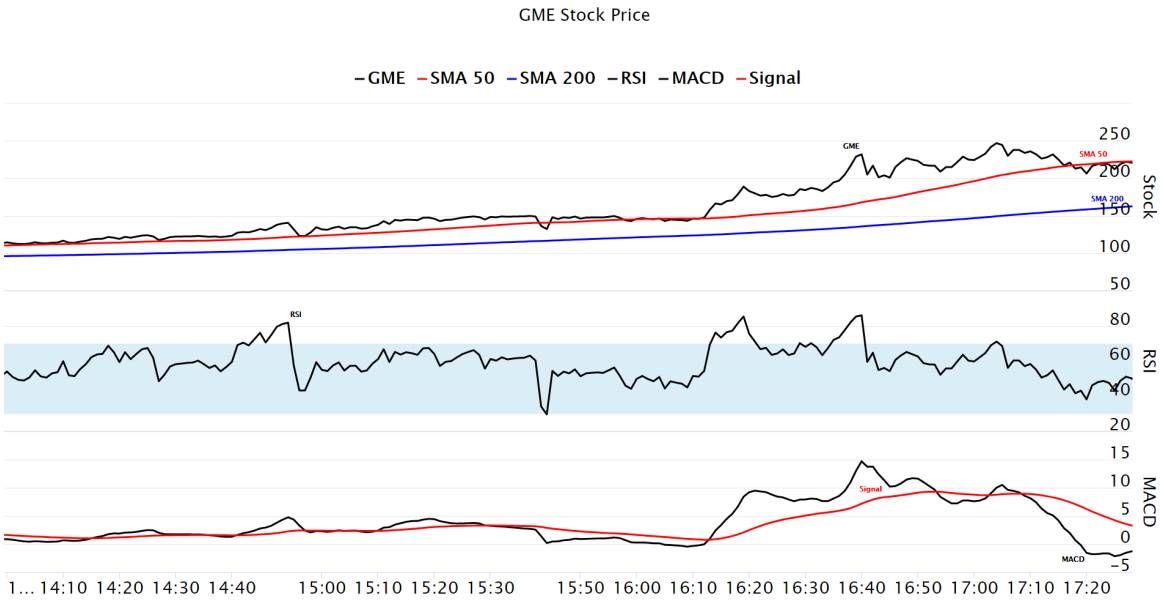
# 5. fejezet

# Kivételes esetek

Sajnálatos módon az indikátorokkal történő kereskedés sem jelent 100%-os hozamot, hiszen nem tud minden tényezővel számolni az algoritmus. Az indikátorok jellemzően késnek, hamarabb következik be a változás, mint az indikátor jelzése. Ebben a fejezetben szeretnék bemutatni néhány tényezőt grafikonokkal, amelyek megnehezítik az automata dolgát.

## 5.1. GameStop short squeeze

A GameStop egy elavult üzleti modellel bíró, csökkenő árbevételt elérő vállalat. Ennek az a magyarázata, hogy boltokban értékesítenek számítógépes játékokat, holott már a platformok nagy része online adja el őket. Ez volt az oka annak, hogy elég sokan sholtolták a céget, hiszen egyre többen bíztak abban, hogy csődközeli állapotba fog térni a vállalat. Azonban egy internetes közösségi oldalon, a Redditen lévő felhasználók short squeeze-t váltottak ki. A csúcspontján, január 28-án a részvényeinek árfolyama 500$ fölött jártak, ami a hónap eleji 17$-hoz képest 30-szoros emelkedés. Látható az ábrán,



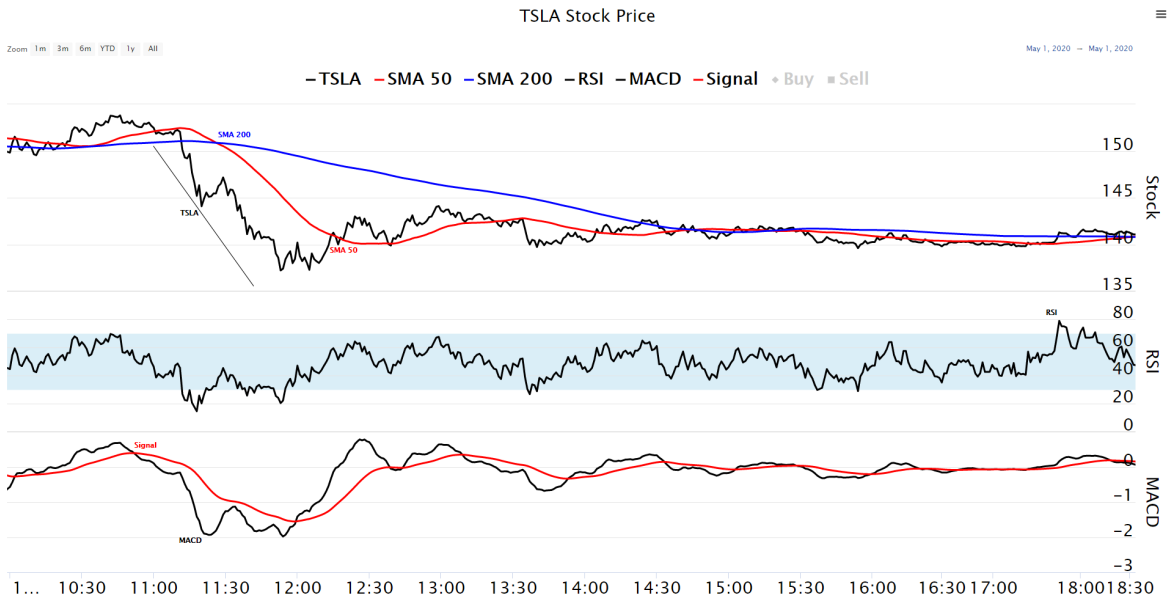

5.1. ábra. GameStop részvény short squeeze közben.

hogy az indikátorok milyen meredeken kezdtek el emelkedni, az MACD is korábban

nem látott magasságokba ugrott. Az indikátorok közül az MACD jelzi egyedül, hogy vásárolni kellene, azonban 27$-t ugrott már mire az indikátor bejelzett.

## 5.2. Hírérzékenység

Az internet fejlődése lehetővé tette a hírek gyors terjedését, ezért manapság egy cégtulajdonos, ha negatív hírt tesz közzé az üzenőfalán, akkor nagyon megmozdulhat az árfolyama a cégének. Például a Tesla vezetője Elon Musk a kedvenc social media platformjára, a Twitterre május elsején kiírta, hogy szerinte túl magas a Tesla árfolyama. Erre reagálva a befektetők adták el a részvényeiket amíg tudták, és egy óra alatt majdnem 15$-os különbséget generáltak.




5.2. ábra. A Tesla részvény esése egy megjelent vélemény miatt.

# 6. fejezet

# Összefoglalás

A kutatásom során egy új témával ismerkedhettem meg, mely eddig ismeretlen volt számomra. A szakdolgozatomban bemutatásra került, hogy a technikai analízis felhasználható automatikus kereskedő szoftver készítésére. A bemutatott példa egy lehetőség, azonban ez egy annyira komplex probléma, mely további kutatást igényel, amennyiben még jobb eredmények elérése a cél.

A dolgozat készítése során az első probléma az volt, hogy milyen adatok kerüljenek felhasználásra, honnan tölthetem le őket. Az általam használt Alpha Vantage oldal jó választásnak bizonyult, azonban kizárólag két évre visszamenőleg lehet a napi adatokat letölteni. Ez némileg nehezítheti a következtetés levonását, viszont így is több százezer rekordos adatbázisokkal dolgoztam, mely elegendőnek bizonyult.

A szakdolgozatom elkészítéséhez két programot készítettem el. Azért döntöttem így, mert a webes felületet egyszerűnek gondoltam, és több eszközről érhető el. Viszont szükség volt egy adatletöltő alkalmazásra, amit Java nyelven írtam meg. Így a két rendszer nem használja el az erőforrást egymástól, a letöltés nem függ a weboldaltól, illetve a letöltés az egy lassabb folyamat. Míg az tart, nyugodtan nézegethetőek az eddig letöltött részvények.

A program által felhasznált statisztikák a kereskedők ajánlásai alapján kerültek kiválasztásra. Összesen 11 darab feltételt alkalmaztam a kereskedés eldöntéséhez. Egy további kutatás témájaként a több ezer indikátor közül érdemes lehet több másik indikátort kipróbálni, hogyan viselkednek az adott környezetben. A feltételeket súlyozással láttam el, és a genetikus algoritmus dönti el egy részvény alapján, hogy milyen súlyokat kapjon. Ezeket a súlyokat a többi részvényre is lehet használni, választástól függően jobb, rosszabb eredményt ér el.

A szoftver megjelenítési mechanizmusa véleményem szerint a gyengepontja a dolgozatomnak, hiszen erőforrásigényessége miatt lassabb számítógépeken nehezebb lehet a használata. Kicsit gyorsabbá lehet tenni, ha a megjelenítést leveszi a felhasználó a mozgatás erejéig.

Összességében az elkészített program megoldotta a kitűzött célt, és sikerült egy nyereséges kereskedő automatát elkészíteni genetikus algoritmus segítségével. A méréseim alapján azt tudom elmondani, hogy a vizsgált részvényeken 83.7%-ban sikerült profitot termelnie a programnak. A feltételek bővítésével elképzelhető, hogy ez növelhető, azonban ha egy létező rendszerbe integrálásra kerülne bővítményként az algoritmusom, úgy gondolom hosszútávon még jobb számot érne el.

# Irodalomjegyzék


[1] Alpha Vantage. https://www.alphavantage.co. [hozzáférés 2021.01.12.].

[2] Az én pénzem. https://penziranytu.hu/. [hozzáférés 2021.04.05.].

[3] Bitcoin. https://bitcoin.org. [hozzáférés 2021.04.13.].

[4] Bootstrap. https://getbootstrap.com. [hozzáférés 2021.01.15.].

[5] Highcharts. https://www.highcharts.com. [hozzáférés 2021.02.03.].

[6] Investopedia. https://www.investopedia.com. [hozzáférés 2021.01.05.].

[7] Közgazdasági ismeretek. https://dtk.tankonyvtar.hu/xmlui/bitstream/handle/123456789/3401/kozgazdasagi_ismeretek_elmeleti_jegyzet.pdf?sequence=1&isAllowed=y, 2013. [hozzáférés 2021.04.12.].

[8] Osztalékportfólió. https://osztalekportfolio.com. [hozzáférés 2021.04.12.].

[9] Tradingview. https://www.tradingview.com. [hozzáférés 2021.04.21.].

[10] U.S. Dividend Champions. https://www.dripinvesting.org/tools/U.S.DividendChampions.pdf, 2021. [hozzáférés 2021.04.16.].

[11] Web technology for developers. https://developer.mozilla.org/en-US/docs/Web. [hozzáférés 2021.01.12.].

[12] Benjámin Graham. *Az intelligens befektető*. T.bálint Kiadó, Törökbálint, 2011.

[13] Daniel Shiffman. *Nature of Code*. 2012.

[14] Sólyomi Dávid. *Osztalékból szabadon*. Bioenergetic, Budapest, 2017.

# Adathordozó használati útmutató

A dolgozathoz tartozó melléklet a következőket tartalmazza.

- `dolgozat.pdf`: a dolgozat PDF formátumban
- `/szakdolgozat`: a dolgozat LaTeX forráskódját tartalmazó jegyzék
- `/programok`: az elkészített programok jegyzéke

A program használatához szükség lesz a Node.js telepítésére, mely a következő linken tölthető le: `https://nodejs.org/en/download/`.
A webszerver használata előtt szükséges azt telepíteni, ehhez a következő parancsot szükséges kiadni egy terminálban:

```
> npm install --global http-server
```

A programok használatát a `/programok` jegyzékben lévő felhasználói dokumentumban részletezem.