VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Sami Peltola

KAUPANKÄYNTIROBOTIN SUUNNITTELU, TOTEUTUS JA ARVIOINTI

Liiketalous
2022

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Sami Peltola
Opinnäytetyön nimi	Kaupankäyntirobotin suunnittelu, toteutus ja arviointi
Vuosi	2022
Kieli	suomi
Sivumäärä	41
Ohjaaja	Antti Mäkitalo

Tämän opinnäytetyön tarkoituksena oli tuottaa ohjelmistorobotti, joka kykenee itsenäiseen ja voitolliseen kaupankäyntiin kryptovaluuttamarkkinoilla. Robotin vaatimuksena oli, että se kykenee toteuttamaan ennalta määrättyä strategiaa niin nousu- kuin laskumarkkinassa siten, että toiminta on keskeyttämätöntä. Tavoitteena oli, että kaupankäynti on voitokasta.

Teoriaosuudessa käsitellään lyhyesti finanssimarkkinoita, päiväkaupankäyntiä ja sen eri strategioita. Tämän jälkeen käydään lyhyesti läpi ohjelmointikieli Python ja sen ominaispiirteet sekä kaupankäyntialusta Binance, jota robotti käyttää kaupankäyntiin.

Käytännön osuudessa esitellään robotin toimintalogiikka, jonka jälkeen paneudutaan robotin luokkaan, josta jatketaan datan käsittelyyn, sekä osto- ja myyntifunktioiden läpikäymiseen.

Lopuksi käydään läpi robotin kaupankäynnin tulosta, siihen vaikuttavia tekijöitä ja parannusehdotuksia, joiden avulla robotista saisi enemmän irti.

Avainsanat	ohjelmointi, ohjelmistokehitys, Python, arvopaperikauppa, virtuaalivaluutta

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittely

ABSTRACT

Author	Sami Peltola
Title	Design, implementation, and assessment of a trading bot.
Year	2022
Language	Finnish
Pages	41
Name of Supervisor	Antti Mäkitalo

The purpose of this bachelor's thesis was to develop a bot which is capable of independent and profitable day trading. The set requirement for the bot was for it to be able to implement as set strategy in downtrends and uptrends without interruptions. The aim was for the trading to be profitable.

The theoretical framework consists of financial markets, day trading and its different strategies. Also, Python as a programming language and Binance crypto exchange were studied.

In the practical section of the thesis the bot's functioning logic is introduced after which the source code is reviewed.

The final section of the work consists of assessing the trading results and different factors that may affect them and how the robot could be improved.

Keywords	programming, software development, Python, security trading, crypto currency

SISÄLLYS

KUVIO- JA TAULUKKOLUETTELO

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on tuottaa ohjelmistorobotti, joka kykenee itsenäiseen ja voitolliseen kaupankäyntiin kryptovaluuttamarkkinoilla. Robotin vaatimuksena oli, että se kykenee toteuttamaan ennalta määrättyä strategiaa niin nousu- kuin laskumarkkinassa siten, että toiminta on keskeyttämätöntä. Tavoitteena oli, että kaupankäynti on voitokasta. Tutkimukselle asetettiin kaksi keskeistä tutkimuskysymystä, jotka olivat ”Kykeneekö yksinkertaisin indikaattorein ja vähällä datalla toteutettu robotti arvonluontiin?” ja ”Onko kannattavaa tuottaa robotti siinä tarkoituksessa, että sen varaan voisi tulevaisuudessa luottaa osan sijoituspääomasta?”

Opinnäytetyön henkilökohtainen tavoitteeni oli oppia erilaisista kaupankäyntiin liittyvistä indikaattoreista, strategioista ja yleisesti päiväkaupasta finanssimarkkinoilla.

Algoritminen kaupankäynti on vallannut finanssimarkkinat. Mordor Intelligence nimisen markkinoita tutkivan yrityksen raportin mukaan 60–73 prosenttia Pohjois-Amerikan pörsseissä liikkuvasta pääomasta on robottien hallinnoimaa (Mordor Intelligence, 2021).

Oikean strategian valinta on keskiössä robotin suunnittelussa ja toteutuksessa. Luvussa 2 käydään läpi päiväkaupan perusteita ja sitä, kuinka perinteinen pörssimarkkina eroaa kryptovaluuttamarkkinasta. Lisäksi tarkastellaan kaupankäyntistrategioita ja strategian valintaa.

Luvussa 3 käydään läpi Pythonia ohjelmointikielenä ja alustaa, jolla kauppaa käydään. Ensimmäisenä paneudutaan Pythoniin ohjelmointikielenä, jonka jälkeen tarkastellaan alustaa, jolla kauppaa käydään ja sen tarjoamaa rajapintaa.

Luvussa 4 paneudutaan robotin toteutukseen, jonka jälkeen luku 5 käsittelee robotin testauksesta ja tuloksien arvioinnista. Luvussa 5 käydään läpi testausympä-

ristö ja tulokset, jonka jälkeen tarkastellaan robotin kilpailukykyä vertaamalla tuloksia kaupattavan instrumentin hintakehitykseen. Viimeisessä luvussa 6 on pohdintaa projektin tuloksista ja siitä, mitä tekisin toisin.

2 PÄIVÄKAUPAN PERUSTEET FINANSSIMARKKINOILLA

Tässä luvussa käydään läpi työn kannalta keskeiset käsitteet finanssimarkkinoihin liittyen. Ensin käydään läpi päiväkaupan käsite, sekä kryptovaluutta- ja pörssimarkkinoiden erot, jonka jälkeen käsitellään lyhyesti kryptovaluutta Bitcoinia ja ominaisuuksia rahoitusinstrumenttina. Luvun toisessa puoliskossa käydään läpi päiväkaupan keskeisiä strategioita ja esitellään tässä työssä käytettävä strategia.

Päiväkauppa tarkoittaa rahoitusinstrumenttien ostoa ja myyntiä tavoitteena hyötyä instrumentin päivänsisäisesti kurssivaihtelusta (Viitala, 2022). Päiväkauppa on tärkeä erottaa perinteisestä sijoittamisesta, joka perustuu pitkäjänteiseen instrumenttien omistamiseen.

2.1 Kryptovaluutta- ja pörssimarkkinoiden erot

Pörssimarkkinoiden toimintaa koostuu liikkeelle laskettujen tuotteiden myymisestä ja ostamisesta. Pörssimarkkinoilla instrumentin hinta koostuu täysin sen kysynnästä ja tarjonnasta (Visma, 2021). Pörssimarkkinoiden tärkein yksittäinen komponentti on siellä myytävät tuotteet, joilla on jokin markkinoilla määräytyvä arvo. Tuotteita on monenlaisia, joista osake on yleisin. Osakkeen omistaja omistaa tietyn osuuden osakeyhtiöstä tai osuuskunnasta (Visma, 2021).

Tässä työssä käydään kauppaa kryptovaluutta Bitcoinilla, joka on vuonna 2009 julkaistu kryptovaluutta (Saastamoinen, Junttila, Kurki, 2014). Kryptovaluutat ovat digitaalisia virtuaalivaluuttoja. Kryptovaluuttoja myydään kryptovaluuttamarkkinoilla, joilla hinnanmuodostukseen vaikuttaa kysynnän ja tarjonnan lisäksi kyseisen valuutan enimmäismäärä, kierrossa oleva valuuttamäärä ja kryptovaluutan louhinnan kustannukset (Joutsen, 2021).

Isoin ero perinteisen pörssimarkkinan ja kryptovaluuttamarkkinan välillä on, että kryptovaluuttamarkkinoilla kauppa käydään suoraan myyjän ja ostajan välillä, toisin kuin perinteisessä pörssissä, jossa myyjä antaa osakkeensa myyntiin välittäjälle (Reed, 2021).

2.2 Miksi Bitcoin?

Bitcoin on äärimmäisen volatiili. Volatiliteetti on rahoitusinstrumentin tuoton keskihajonta annetulla aikahorisontilla (Sijoitustieto, 2018). Bitcoin on mahdollisesti volatiilein omaisuuserä, johon voi sijoittaa. Tämä tarkoittaa käytännössä sitä, että kurssiheilunta on todella suurta (Reed, 2021).

Bitcoinin volatiilisuus yhdistettynä siihen, että kryptovaluuttamarkkinalla ei ole aukioloaikoja, mahdollistaa niin suuret yksittäiset kaupat, kuin kaiken hyödyn irtioton siitä, että robotit eivät väsy.

2.3 Tekninen analyysi, strategiat ja indikaattorit

2.3.1 Tekninen analyysi

Tekninen analyysi on sijoitta-ammattilaisten käyttämä menetelmä, jolla pyritään mallintamaan, sekä etsimään ennustettavissa olevia säännönmukaisuuksia pörssikurssien vaihtelusta. Tekninen analyysi on suuressa osassa erilaisten trendien tunnistamisessa ja niiden hyödyntämisessä sijoituspäätöksissä. Modernissa teknisessä analyysissä hyödynnetään erilaisia tilastollisia indikaattoreita, joiden neljä pääkategoriaa ovat trendi-indikaattorit, momentti-indikaattorit, volatiliteetti-indikaattorit ja tuki- ja vastustasoindikaattorit. (Heikinheimo, 2021)

2.3.2 Lyhyeksi myynti

Lyhyeksi myynti, eli niin sanottu shorttaus tarkoittaa arvopapereiden myyntiä, ilman, että omistaa niitä. Lyhyeksi myyjä odottaa kurssin laskevan ja toivoo voivansa ostaa osakkeet takaisin halvemmalla. Jos kurssit nousevat, lyhyeksi myyjä häviää. Myyjä lainaa tai vuokraa arvopaperit myyntihetkellä välittäjältä tai toiselta sijoittajalta. (Nordnet, 2022)

2.3.3 Strategia

Strategia päiväkaupassa ja yleisesti sijoittamisessa tarkoittaa jotain ennalta määrättyä säännöstöä, jonka mukaan markkinoilla toimitaan. Strategiassa määritellään muun muassa instrumentit, joilla kauppaa käydään, säännöt ostamiselle ja myymiselle ja erilaiset säännöt sille, kuinka paljon pääomasta on annetulla hetkellä käytössä kussakin instrumentissa. (Hayes, 2021)

Päiväkauppa vaatii tarkoin määrittelyn strategian. Päiväkauppa ei koostu hetken mielijohteesta tehdyistä kaupoista, vaan tarkoin valikoitujen indikaattorien avulla tunnistetuista mahdollisuuksista. Erilaisia päiväkaupan strategioita ovat muun muassa trendin perusteella sijoittaminen (Trend following), vaihteluvälin perusteella sijoittaminen (Range trading) ja skalppaus (Scalping). Strategioita on lukuisia, eikä tämän työn kontekstissa ole tarkoituksenmukaista läpikäydä jokaista. (CMC Markets, 2022)

2.3.4 Trendin perusteella sijoittaminen

Trendisijoittamisessa pyritään tunnistamaan tarkasteltavan instrumentin trendi, eli suunta, johon instrumentin hinta on menossa ja tekemään sijoituspäätöksiä sen perusteella. Trendisijoittamisessa käytetään liukuvia keskiarvoja, jotka tasoittavat hintakehitystä ja suodattavat ylimääräistä liikettä hintavaihtelusta. Liukuva keskiarvo laskee keskiarvon valitulle aikajaksolle. Liukuva keskiarvo voidaan laskea erilaisilla painotuksilla, joita ovat muun muassa yksinkertainen liukuva keskiarvo, eksponentiaalinen liukuva keskiarvo ja painotettu liukuva keskiarvo. (Mitchell, 2022)

Trendisijoittamisessa trendin tunnistaminen tapahtuu siten, että asetetaan kaksi liukuvaa keskiarvoa tosiaan vasten niin, että toinen keskiarvo reagoi nopeammin hintavaihteluun (lyhyempi aikaväli) ja toinen hitaammin (pidempi aikaväli). Kun

nopeampi keskiarvo leikkaa hitaamman alhaalta ylös, voidaan sitä pitää ostosignaalina. Kun taas leikkaus tapahtuu päin vastaiseen suuntaan, voidaan sitä pitää myyntisignaalina. (Mitchell, 2022)

2.3.5 Vaihteluvälin perusteella sijoittaminen

Vaihteluväliin perustuvassa strategiassa tuotteelle asetetaan hintahistorian perusteella vastustus- ja tukitasoja. Vastustustaso voi olla esimerkiksi alin hinta, johon osuessa tuotteen hinta on menneisyydessä noussut nopeasti ylös tai vastaavasti laskenut rajusti. Tätä strategiaa toteuttava sijoittaja siis pyrkii tunnistamaan vastustasot ja esimerkiksi ostamaan tai myymään lyhyeksi, kun hinta reagoi toivotulla tavalla osuessaan vastustustasoon. (CMC Markets, 2022)

2.3.6 Skalppaus

Skalppaus on kaupankäyntistrategia, jossa sijoittaja etsii hyötyjä pienistä hintavaihteluista avaamalla ja sulkemalla useita eri positioita yhdessä kaupassa. Riski on rajallinen ja voitot eivät kerry yksittäisistä kaupoista, vaan suuren volyymin avulla mahdollisesti sadoista kaupoista päivän aikana. (FxPro-Finland, 2022)

2.3.7 Strategian valinta ja esittely.

Valitsin robotille strategiaksi trendiin perustuvan sijoittamisen. Tarkemmin ottaen valitsin strategian, joka käyttää nopeana indikaattorina eksponentiaalista liukuvaa keskiarvoa viimeisen 9 datapisteen kohdalta (EMA9). Pitkänä indikaattorina käytetään painotettua liukuvaa keskiarvoa viimeisen 30 datapisteen kohdalta (WMA30). (Trading Strategy Guides, 2020)

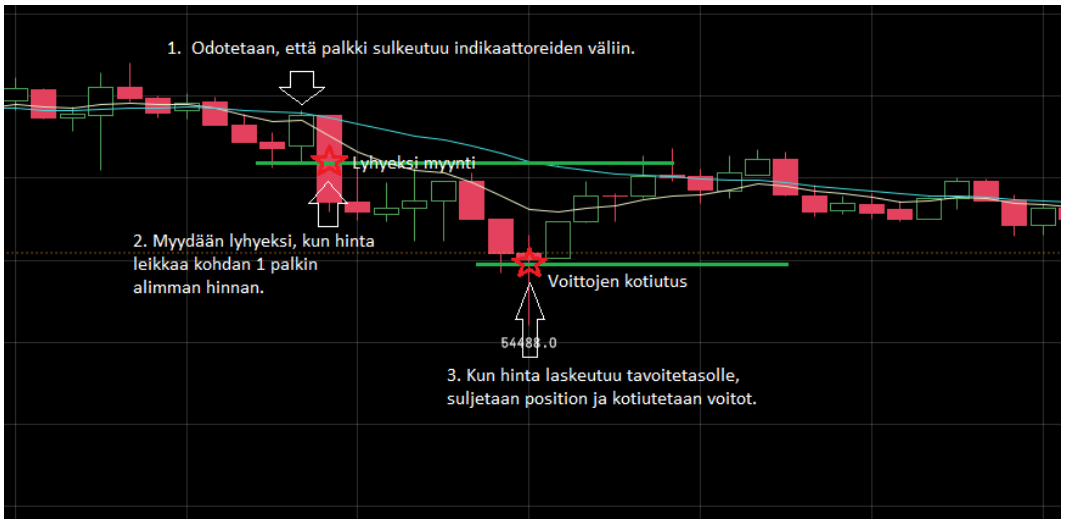
Kuva 1. EMA9 ja WMA30 indikaattorit.

Kuvassa 1 näkyvät strategiassa käytettävät indikaattorit. Kuvassa yksi palkki, niin sanottu kynttilä, kuvaa yhden tunnin hintavaihtelua. Palkin paksumpi osuus on yhden tunnin aukeamis- ja sulkuhinnan välinen erotus. Kapeat viivat kuvaavat yhden tunnin alinta ja korkeinta hintaa. Jos palkki on vihreä, on hinta noussut. Jos taas palkki on punainen, on hinta tarkastelujaksolla laskenut. Kuvassa indikaattorit kuvaavat nousumarkkinaa, eli tilannetta, jossa EMA9-indikaattori on WMA30-indikaattorin yläpuolella. Laskumarkkinassa EMA9-indikaattori on WMA30-indikaattorin alapuolella. (Trading Strategy Guides, 2020)

Kuva 2. EMA9/WMA30-strategian toteutus nousumarkkinassa.

Valitun strategia toteutus nousumarkkinassa tapahtuu seuraavasti:

1. Odotetaan, että palkki sulkeutuu indikaattoreiden väliin.
2. Odotetaan, että hinta nousee kyseisen palkin huippuun ja ostetaan (Kuvassa 2 vasen punainen tähti).
3. Myydään, kun asetettu hintatavoite saavutetaan (Kuvassa 2 oikea punainen tähti). Hintatavoite voi olla prosentuaalinen osuus ostohinnasta tai esimerkiksi trendin muutoskohta, eli seuraava indikaattoreiden leikkauspiste.



Kuva 3. EMA9/WMA30-strategian toteutus laskumarkkinassa.

Valitun strategian toteutus laskumarkkinassa tapahtuu seuraavasti:

1. Odotetaan, että palkki sulkeutuu indikaattoreiden väliin.
2. Odotetaan, että hinta laskeutuu kyseisen palkin alimpaan tasoon ja avataan positio lyhyeksi myyntiä varten.
3. Myydään, kun hinta laskeutuu tavoitetasolle. Tällöin palautetaan osakkeet lainaajalle ja voitto-osuus on myynti- ja ostohinnan erotus.

Käydään seuraavaksi läpi tämän strategian valintaan johtaneet syyt.

Ohjelmoinnilliset syyt. Trendisijoittaminen vaikutti näistä sellaiselta, että sen toteuttaminen luotettavasti robotille olisi todennäköisintä. Kaupankäyntialustalla on esimerkiksi rajoituksia rajapinnalta kyselemisen tiheyteen, joka olisi saattanut olla ongelma skalppauksen kohdalla. Mitä taas tulee vaihteluväliin perustuvaan strategiaan, sen tukitason eivät välttämättä pidä, joten sääntöjen rakentaminen sille, milloin tukitaso pitää ja milloin ei, voisi olla haasteellista.

Strategian yksinkertaisuus ja ymmärrettävyys. On yleisesti todella tärkeää, että ymmärtää strategia. Ymmärrettävyys helpottaa myös strategian toteutusta ohjelmointivaiheessa. EMA9/WMA30-strategia tarjosi selkeät raamit, joiden sisällä kauppoja tulee tehdä.

3 PYTHON JA KAUPANKÄYNTIALUSTA BINANCE

Tässä luvussa paneudutaan ohjelmointikieli Pythoniin ja sen keskeisiin ominaisuuksiin. Tämän jälkeen käydään lyhyesti läpi kaupankäyntialusta Binance. Lopuksi käydään lyhyesti läpi käsite ohjelmointirajapinta.

3.1 Python ohjelmointikielenä

Valitsin opinnäytetyössä käytettäväksi kieleksi Pythonin. Pythonille on saatavilla sopivat kirjastot robotin toteuttamiseen ja kieli on itselleni hyvin tuttu.

Python on vuonna 1990 julkaistu tulkattava, korkean tason ohjelmointikieli (Arora, 2022). Tulkattavuus tarkoittaa, että Pythonilla kirjoitetut ohjelmat ovat valmiita ajettaviksi välittömästi, kääntämättä niitä ensin. Tämä mahdollistaa ohjelman nopean ohjelmoinnin ja testaamisen, mutta tarkoittaa myös sitä, että Python-ohjelmat eivät ole yhtä suorituskykyisiä kuin käännetyillä kielillä kirjoitetut ohjelmat. Korkean tason ohjelmointikielet puolestaan ovat kieliä, jotka tarjoavat vahvan abstraktion tietokoneen toiminalle ja eivät ole rajattuja tietokoneen mukaan. Voisi ajatella, että korkean tason abstraktion tarjoavat kielet ovat lähempänä ihmiselle luonnollista ajattelua. (Beal, 2022) Pythonin ympärille on kasautunut suuri yhteisö, joka luo ja tarjoaa Pythonille avoimen lähdekoodin kirjastoja, jotka nopeuttavat kehitystyötä. (Arora, 2022)

Pythonin suosio on viime vuosina noussut merkittävästi. Pythonin tarjoama korkea abstraktion taso tekee siitä hyvin aloittelijaystävällisen. Tämän lisäksi python soveltuu hyvin koneoppis-, massadata- ja pilvipalveluprojektien toteuttamiseen. Myös kirjastojen laaja valikoima helpottaa erilaisten projektien aloittamista nimenomaan Pythonilla. (Dev, 2020)

3.2 Kaupankäyntialusta Binance

Binance on alun perin kiinalainen, vuonna 2017 perustettu yhtiö, joka tarjoaa yksityisiä pankkipalveluita, sekä ylläpitää kryptovaluuttapörssiä. Binancessa Fiat-

raha muutetaan vaakavaluutaksi, jolla sitten käydään kauppaa muilla kryptovaluutoilla (Peters, 2021). Fiat-raha tarkoittaa valuuttaa, jolla ei ole itsearvoa, vaan sen arvo perustuu hallinnon luomiin säädöksiin sen olemisesta laillinen maksuväline. Esimerkkejä Fiat-rahasta ovat muun muassa euro ja Yhdysvaltain dollari (Chen, 2021). Vakaavaluutta taas on kryptovaluutta, jonka arvo on sidottu johonkin Fiat-valuuttaan, yleensä Yhdysvaltain dollariin (Hayes, 2022). Valitsin Binancen kaupankäyntialustaksi, koska se tarjoaa sopivan rajapinnan ja pienimmät kulut.

3.3 Rajapinnoista yleisesti

Ohjelmointirajapinta, eli API (Application programming interface), on raja komponenttien välillä ohjelmoitavassa järjestelmässä. Ohjelmointirajapinta mahdollistaa tiedonvälityksen kahden laitteiston tai ohjelman välillä (Hoffman, 2021). Tässä työssä Binancen tarjoamalle rajapinnalle lähetetään erilaisia GET- ja POST-kyselyitä. GET-kyselyillä pyydetään dataa rajapinnalta tietoa esimerkiksi Bitcoinin hintatilanteesta. POST-kyselyillä halutaan tehdä jotain järjestelmässä. Esimerkki POST-kutsusta voisi olla tilauksen tekeminen.

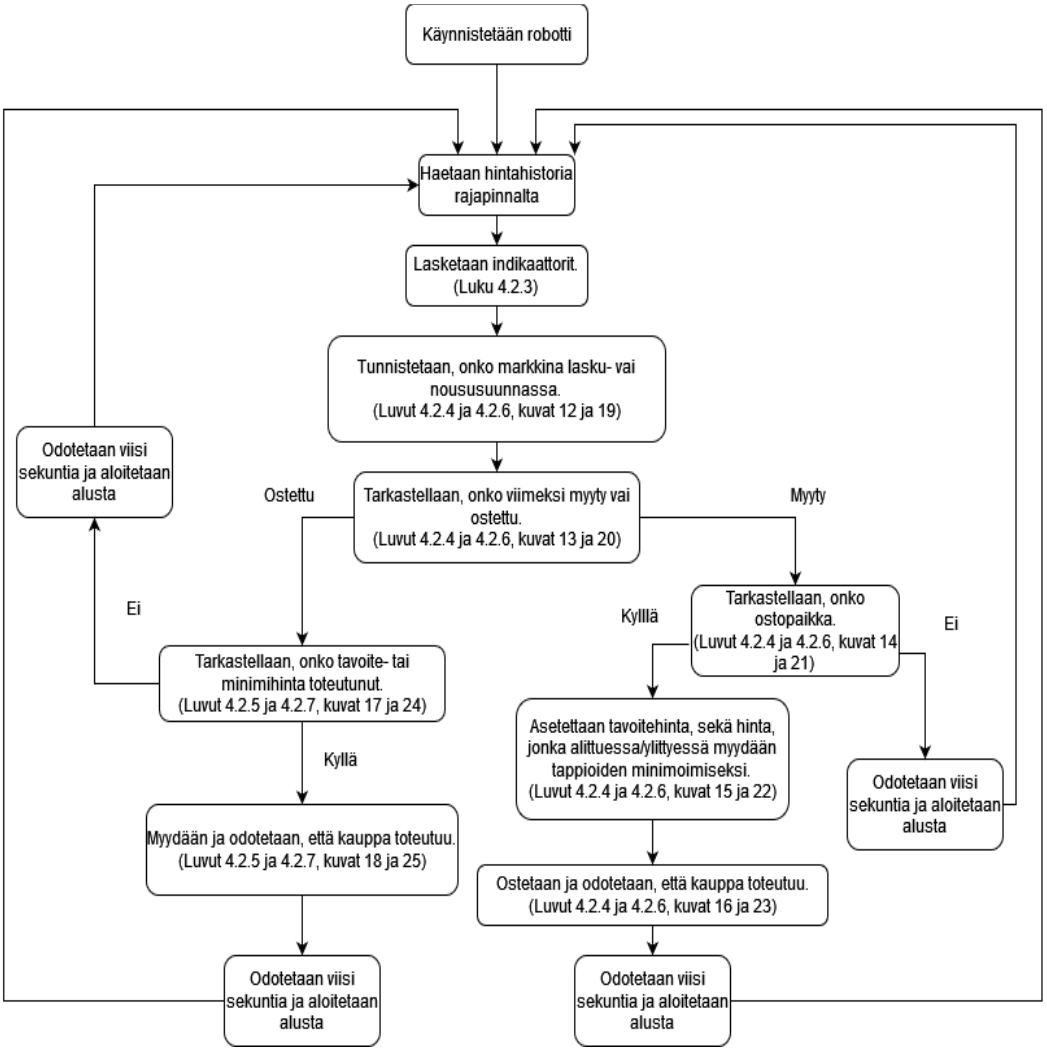
4 ROBOTIN RAKENNE JA TOTEUTUS

Tässä luvussa paneudutaan robotin käytännön toteutukseen. Ensin käydään läpi robotin toimintalogiikka, jonka jälkeen siirrytään koodin läpikäymiseen.

4.1 Robotin toimintalogiikka

Robotin toimii hyvin suoraviivaisesti. Kuvassa 4 (s. 19) kuvataan robotin toimintaa. Ensin haetaan Bitcoinin hintahistoria rajapinnalta ja lasketaan indikaattorit datan perusteella. Indikaattoreiden avulla tunnistetaan, onko lasku- vai nousumarkkina. Kun markkinan trendi on tiedossa, tarkastellaan, onko viimeisin kauppa ollut osto- vai myyntitapahtuma. Mikäli viimeisin tapahtuma on ollut myyntitapahtuma, tarkastellaan, onko tällä hetkellä markkinoilla ostopaikkaa. Jos taas viimeisin tapahtuma on ostotapahtuma, tarkastellaan, onko markkinoilla myyntipaikkaa. Mikäli myynti- tai ostopaikkaa ei ole, aloitetaan kierros uudelleen. Jos viimeisin tapahtuma on myyntitapahtuma ja markkinoille on avautunut ostomahdollisuus, avataan ostotarjous ja odotetaan, että se täyttyy. Tämän jälkeen asetetaan hintatavoite, jonka täyttyessä omistus myydään. Jos taas viimeisin tapahtuma on ostotapahtuma ja markkinoille on avautunut myyntimahdollisuus, avataan myyntitarjous tavoitehintaan ja odotetaan sen päättymistä.

Kaupankäynnissä voi tulla tilanne, että hintakehitys onkin strategian vastaista. Tällaisia tilanteita varten päivitetään stop loss-arvoa, jonka ylittyessä tai alittuessa omistus myydään tappioiden minimoimiseksi. Kun kyseessä on perinteinen osta-myy-kauppa, stop loss toteutuu, kun hinta alittaa WMA30-indikaattorin. Lyhyeksi myydessä stop loss toteutuu, kun hinta ylittää WMA30-indikaattorin.



Kuva 4. Robotin toimintalogiikka

4.2 Robotin rakenne, luokat ja metodit

4.2.1 Robotin initialisointi

Robotti koostuu yhdestä luokasta, nimeltä trading_bot. Kuvassa 5 näkyy luokan initialisointimetodi. Initialisoinnin yhteydessä haetaan API-avaimet keys-nimisestä JSON-tiedostosta (Kuva 6) ja alustetaan Binance-kirjaston Client-luokka. API-avaimet luodaan Binancen verkkosivuilla.

```
22  class trading_bot():
23      """
24      Trading bot. Strategy is to use EMA9 as a short indicator and WMA30 as a long indicator.
25      """
26      def __init__(self, ):
27
28          # DEFINING API
29          self.api_key, self.private_key = self.load_keys()
30          self.api = Client(self.api_key, self.private_key)
31
32          # TRADING VARIABLES
33          self.currency = "BTCBUSD" # This is used in public queries
34          self.currency_private = "BUSD" # This is used in private queries
35          self.fund_currency = "BUSD" # This is used in currency queries
36          self.uptrend = False
37          self.downtrend = False
38          self.trend_change = False
39          self.loan_open = False
40          self.win_target = 1.01
41          self.short_win_target = 0.99
42          self.stop_loss_value = 0
43          self.available_funds_spot = None
44          self.available_funds_margin = None
45          self.long_buy_price = 0
46          self.short_buy_price = 0
47          self.target_price = 0
48          self.available_btc = 0
49          self.last_bought_price = 0
50          self.shorted_amount = 0
```

Kuva 5. Trading_bot-luokan initialisointi.

```
{
    "api_key" : "vcEZ                                                    ",
    "private_key" : "ZFt                                                                  "
}
```

Kuva 6. keys.JSON-tiedosto.

4.2.2 API-kyselyt tietojen hakemiseen.

Datan, kuten käytettävissä olevan pääoman tai historian, kyseleminen on rajapintakirjaston avulla yksinkertaista. Kuvassa 7 on kuvattu kaksi erilaista metodia eri datan hakemiseen ja käsittelyyn.

Get_available_funds_spot -metodi pyytää rajapinnalta tietoa siitä, kuinka paljon BUSD-vakaavaluuttaa on käytettävissä. Rajapinta palauttaa Dictionary-muotoista tietoa, joka on kokoelma avain-arvo-pareja. Get_available_funds_spot -metodi palauttaa kyselyn tuloksena tulleesta Dictionary-muotoisesta muuttujasta free-avaimen arvon.

```
488     def get_available_btc_spot(self):
489         api_callback = None
490         time.sleep(1)
491         while True:
492             try:
493                 api_callback = self.api.get_asset_balance(asset="BTC")
494                 funds = float(api_callback["free"])
495                 return funds
496             except Exception as e:
497                 print(api_callback)
498                 print(e)
499                 time.sleep(1)
500                 pass
501 
502     def get_available_funds_spot(self):
503         time.sleep(1)
504         api_callback = None
505         while True:
506             try:
507                 api_callback = self.api.get_asset_balance(asset="BUSD")
508                 funds = float(api_callback["free"])
509                 return funds
510             except Exception as e:
511                 print(api_callback)
512                 print(e)
513                 time.sleep(1)
514                 pass
```

Kuva 7. Esimerkkejä rajapintakyselyn tekevistä metodeista.

4.2.3 Indikaattoreiden tuottaminen

Historiadatan avulla tuotetaan strategian mukaiset indikaattorit. Painotettu liukuva keskiarvo (WMA) on tässä työssä tuotettu itse, kun taas eksponentiaalisen liukuvan keskiarvon tuottamiseen käytetään pandas-kirjastoa.

Kuvassa 9 (s. 23) näkyy eksponentiaalisen liukuvan keskiarvon tuottava metodi. Tässä metodissa huomion arvoista on, että keskiarvon tuottamiseen käytetään pandas-kirjastoa ja data käsitellään siinä DataFrame-muotoisena.

Eksponentiaalinen liukuva keskiarvo painottaa voimakkaasti uusimpia muutoksia hinnassa. Niiden painottaminen tapahtuu käyttämällä laskennassa painokertoimia. Kertoimien avulla vanhempien hintojen painotus laskee eksponentiaalisesti. (Kohti taloudellista riippumattomuutta, 2010)

$$\mathrm{EMA} = \mathrm{Price}_t \times k + \mathrm{SMA}_y \times (1 - k)$$

where:

$t = \mathrm{Today}$

$$k = \frac{2}{\mathrm{Number\ of\ days\ in\ period} + 1}$$

$\mathrm{SMA} = \mathrm{Simple\ Moving\ Average\ of\ closing\ price}$
$\mathrm{for\ the\ number\ of\ days\ in\ the\ period}$

$y = \mathrm{Yesterday}$

Kuva 8. Eksponentiaalisen liukuvan keskiarvon laskentakaava.

```
def calculate_ema9(self, data):
    period = 9
    needed_prices_list = []
    index = 0

    for i in list(reversed(data)):
        if index <= 29:
            needed_prices_list.append( float(i[4]) )
            index += 1
        else:
            break

    values = DataFrame(list(reversed(needed_prices_list)))
    ema9s = values.ewm(span=period, adjust=False).mean()
    all_ema9 = list(reversed(ema9s.values.tolist()))
    ema9 = []
    for i in range(0, period):
        ema9.append(all_ema9[i][0])

    return ema9
```

Kuva 9. Eksponentiaalisen liukuvan keskiarvon tuottaminen.

Kuvassa 10 on kuvattu painotetun keskiarvon laskentakaava, jossa n-merkkaa laskennan aikajaksoa. Kuvassa 11 (s. 24) näkyy painotetun liukuvan keskiarvon tuottava metodi. Myös keskiarvo painottaa uusia hintoja, mutta siten, että vanhempien hintojen merkitys laskee lineaarisesti. (Kohti taloudellista riippumattomuutta, 2010)

$$\mathrm{WMA} = \frac{\mathrm{Price}_1 \times n + \mathrm{Price}_2 \times (n-1) + \cdots \mathrm{Price}_n}{\frac{n \times (n+1)}{2}}$$

Kuva 10. Painotetun liukuvan keskiarvon laskentakaava.

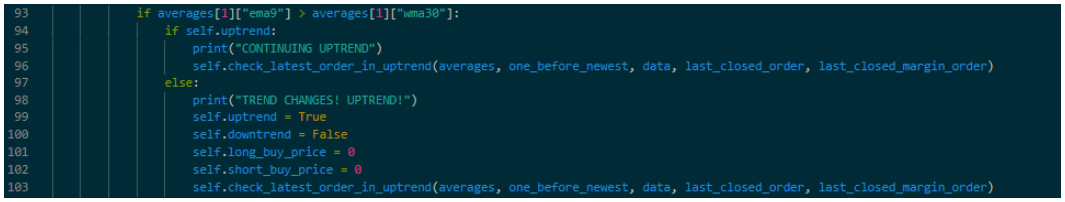
```
def calculate_wma30(self, data):
    reversed_data = list(reversed(data))
    wma30s = []
    for i in range(0,30): # Values from 0 to 30
        wma30_values = 0
        y=i+31
        wma_weight = 30
        #print(f"Calculating wma for {i}")
        for n in range(i,y):# Values from i + 30 to 1
            wma30_values += float(reversed_data[n][4]) * wma_weight
            wma_weight -= 1
            #print(n)

        denominator = 30 * 31 / 2
        wma30 = wma30_values / denominator
        wma30s.append(wma30)
    return wma30s
```

Kuva 11. Painotetun liukuvan keskiarvon tuottaminen

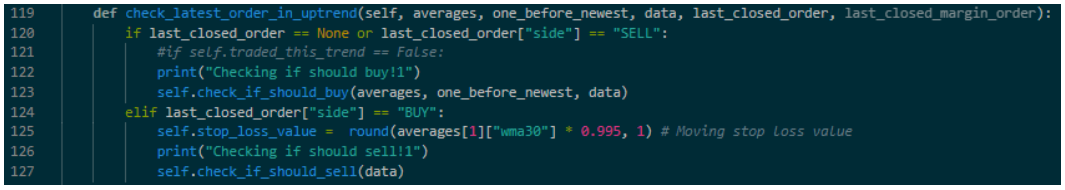
4.2.4 Ostaminen nousumarkkinassa

Kuten strategialuvussa todettiin, nousumarkkina tarkoittaa tilannetta, jossa nopeampi indikaattori on hitaamman yläpuolella. Kuvan 12 (s. 24) ensimmäisellä rivillä tarkastellaan viimeisimmän sulkeutuneen kynttilän tilannetta strategian mukaisesti. Jos toisen rivin ehto toteutuu, ollaan oltu nousumarkkinassa jo ainakin yhden tarkastelukierroksen ajan. Jos taas toisen rivin ehto ei toteudu, nopea indikaattori on juuri leikannut hitaamman ja ollaan saavuttu nousumarkkinaan. Kuvan 13 (s. 25) riveillä 120 ja 124, sekä kuvan 20 (s. 30) riveillä 130 ja 134 tarkastellaan toimintalogiikan mukaisesti edellistä tapahtumaa. Mikäli edellinen tapahtuma on ostotapahtuma, tarkastetaan, voidaanko myydä ja toisin päin.



```
93              if averages[1]["ema9"] > averages[1]["wma30"]:
94                  if self.uptrend:
95                      print("CONTINUING UPTREND")
96                      self.check_latest_order_in_uptrend(averages, one_before_newest, data, last_closed_order, last_closed_margin_order)
97                  else:
98                      print("TREND CHANGES! UPTREND!")
99                      self.uptrend = True
100                     self.downtrend = False
101                     self.long_buy_price = 0
102                     self.short_buy_price = 0
103                     self.check_latest_order_in_uptrend(averages, one_before_newest, data, last_closed_order, last_closed_margin_order)
```

Kuva 12. Nousutrendin tunnistaminen ja sen perusteella toimiminen.



```
119     def check_latest_order_in_uptrend(self, averages, one_before_newest, data, last_closed_order, last_closed_margin_order):
120         if last_closed_order == None or last_closed_order["side"] == "SELL":
121             #if self.traded_this_trend == False:
122             print("Checking if should buy!1")
123             self.check_if_should_buy(averages, one_before_newest, data)
124         elif last_closed_order["side"] == "BUY":
125             self.stop_loss_value =  round(averages[1]["wma30"] * 0.995, 1) # Moving stop loss value
126             print("Checking if should sell!1")
127             self.check_if_should_sell(data)
```

Kuva 13. Edellisen tapahtuman perusteella toimiminen nousutrendissä.

```
217     def check_if_should_buy(self, averages, data_from_latest_candle, data):
218         trend_strength = round(averages[0]["ema9"] / averages[0]["wma30"], 4)
219         trend_is_strong = False
220         if trend_strength >= 1.003:
221             trend_is_strong = True
222         if float(data_from_latest_candle[4]) > averages[1]["wma30"] and float(data_from_latest_candle[4]) < averages[1]["ema9"]:
223             self.long_buy_price = float(data_from_latest_candle[2])
224             self.stop_loss_value = round(averages[0]["wma30"] * 0.995, 1)
225             print("Last complete candle closed in between indicators")
226             time.sleep(2)
227             current_price = float(self.get_current_value(data))
228             time.sleep(2)
229             if current_price >= self.long_buy_price and trend_is_strong:
230                 print(f"Setting buy on {data_from_latest_candle[2]}")
231                 print(f"With STOP LOSS of {data_from_latest_candle[3]}")
232                 volume = self.available_funds * (1 / current_price)
233                 self.set_buy_order(current_price, volume)
234             else:
235                 print(f"current price {current_price}")
236                 print(f"trend is strong = {trend_is_strong} strength =  {trend_strength}")
237         elif self.stop_loss_value != 0 and self.long_buy_price != 0 and trend_is_strong:
238             time.sleep(2)
239             current_price = float(self.get_current_value(data))
240             time.sleep(2)
241             if current_price >= self.long_buy_price:
242                 print(f"Setting buy on {current_price}")
243                 print(f"With STOP LOSS of {self.stop_loss_value}")
244                 volume = self.available_funds * (1 / current_price)
245                 self.set_buy_order(current_price, volume)
246             else:
247                 print(f"current price {current_price}")
248                 print(f"trend is strong = {trend_is_strong} strength =  {trend_strength}")
249         else:
250             print(f"trend is strong = {trend_is_strong} strength =  {trend_strength}")
```

Kuva 14. Ostopaikan tunnistaminen nousutrendissä.

Kuvassa 14 rivillä 222 tarkastellaan, onko edellinen palkki sulkeutunut indikaattoreiden väliin. Voi olla, että oston kriteerit täyttyvät siinä mielessä, että palkki on sulkeutunut indikaattoreiden väliin, mutta nykyinen hinta ei ole vielä noussut kynttilät huippuun. Tässä tapauksessa hinnat otetaan muistiin ja niitä tarkastellaan tulevilla kierroksilla. Tämä tarkastelu tapahtuu rivillä 237.

```
241     def set_buy_order(self, buy_price, volume):
242         while True:
243             try:
244                 print("Trying to buy 1")
245                 api_call_data= {
246                                 "symbol" : self.currency,
247                                 "quantity" : self.truncate(volume, 5),
248                                 "price" : float(round(buy_price, 5))
249                                 }
250                 self.buy_order_api_call(api_call_data)
251                 data = None
252                 with open("trades.json", "r", encoding="utf-8") as f:
253                     data = json.loads(f.read())
254                 data["trades"].append(api_call_data)
255                 with open("trades.json", "w", encoding="utf-8") as f:
256                     f.write(json.dumps(data, indent=4))
257                 open_position = self.get_latest_order_spot() #Last closed order from api
258                 self.long_buy_price = 0
259                 self.last_bought_price = float(open_position["price"])
260                 self.target_price =  round(self.last_bought_price * self.win_target, 1)
261                 #self.traded_this_trend = True
262                 print("set_buy_orderPrint")
263                 return
264             except Exception as e:
265                 time.sleep(1)
266                 print(f"{e}")
267                 pass
```

Kuva 15. Rajapintakyselyn sisällön tuottaminen ja oston jälkeisten toimenpiteiden tekeminen nousutrendissä

Kuvassa 14 (s. 25) metodia set_buy_order (Kuva 15) kutsutaan, kun nykyinen hinta täyttää strategian määräävän tason. Itse ostaminen tapahtuu buy_order_api_call -metodissa (Kuva 16, s. 27). Buy_order_api_call -metodin jälkeen asetetaan tavoitehinta viimeisimmän tapahtuman hinnan mukaan.

```
269     def buy_order_api_call(self, data):
270         """
271         This is a separate method for handling buying via the api.
272         """
273         while True:
274             try:
275                 api_callback_buy = self.api.order_limit_buy(**data)
276                 open_order = self.get_open_order(api_callback_buy["clientOrderId"])
277                 if open_order == True:
278                     # If trade was accepted we have to wait for it to close
279                     while True:
280                         open_order = self.get_open_order(api_callback_buy["clientOrderId"])
281                         if open_order == True:
282                             print("waiting for trade to finish4")
283                             time.sleep(1)
284                         else:
285                             break
286                 if open_order == False:
287                     print("Buy order was made!!")
288                     return
289             except Exception as e:
290                 time.sleep(1)
291                 print(f"{e}")
292                 pass
```

Kuva 16. Rajapintakyselyn käsittely ostotapahtumissa.

Kuvassa 16 kuvataan ostotapahtuman toteuttaminen rajapinnan kautta. Niin ostamisessa, kuin myymisessäkin rajapintakäsittelyt toteutetaan erillisissä metodeissa. Täysin erillisten metodien tekeminen auttaa osto- ja myyntitoiminnallisuuksien testaamisessa, mahdollistaa saman asian tekemisen toisaalla, sekä pitää koodin siistinä.

4.2.5 Myyminen nousumarkkinassa

Ostamisen yhteydessä asetetaan kaupalle tavoitehinta, joka on ostohinta + yksittäisen kaupan prosentuaalinen tavoitehinta. Tavoitehinta voi vaihdella paljonkin riippuen aikavälistä, joka historiadatalle on asetettu. Mitä pidempi aikaväli, sitä isompaa prosentuaalista tuottoa voidaan odottaa. Stop loss-arvo seuraa WMA30-indikaattoria strategian mukaisesti.

```
142     def check_if_should_sell(self, data):
143         current_price = float(self.get_current_value())
144         if current_price >= self.target_price:
145             # Profit
146             self.set_sell_order(current_price, "limit")
147         elif current_price < self.stop_loss_value:
148             # Loss
149             self.set_sell_order(current_price, "market")
150 
151     def set_sell_order(self, price, ordertype):
152         while True:
153             try:
154                 api_call_data= {
155                                 "symbol" : self.currency,
156                                 "quantity" : self.truncate(self.available_btc, 5),
157                                 "price" : float(round(price, 5))
158                                 }
159                 data = None
160                 with open("trades.json", "r", encoding="utf-8") as f:
161                     data = json.loads(f.read())
162                 data["trades"].append(api_call_data)
163                 with open("trades.json", "w", encoding="utf-8") as f:
164                     f.write(json.dumps(data, indent=4))
165                 self.sell_order_api_call(api_call_data)
166                 print("SELL WAS MADE!")
167                 self.stop_loss_value = 0
168                 self.target_price = 0
169                 time.sleep(1)
170                 break
171             except Exception as e:
172                 time.sleep(1)
173                 print(f"{e}")
174                 pass
```

Kuva 17. Metodit myyntipaikan tunnistamiselle, sekä rajapinnalle lähetettävän datan muodostamiselle ja myynnin jälkitoimenpiteille.

Kuvan 13 (s. 25) check_if_should_sell -metodia kutsutaan, kun halutaan tietää, onko myyntimahdollisuutta auennut. Metodi vertaa nykyhetken hintaa ostamisen yhteydessä määriteltyyn tavoitehintaan ja hintakehitystä seuraavaan stop loss-arvoon. Set_sell_order -metodi puolestaan koostaa rajapinnalle lähetettävän datan ja kutsuu rajapintakäsittelyn tekevää metodia sell_order_api_call, jonka jälkeen tavoite- ja minimihinta nollataan.

```
176     def sell_order_api_call(self, data):
177         """
178         This is a separate method for handling selling via the api.
179         """
180         api_callback_sell = None
181         while True:
182             try:
183                 api_callback_sell = self.api.order_limit_sell(**data)
184                 open_order = self.get_open_order(api_callback_sell["clientOrderId"])
185                 if open_order == True:
186                     # If trade was accepted we have to wait for it to close
187                     while True:
188                         open_order = self.get_open_order(api_callback_sell["clientOrderId"])
189                         if open_order == True:
190                             print("waiting for trade to finish4")
191                             time.sleep(1)
192                         else:
193                             break
194                 if open_order == False:
195                     print("Sell order was made!!")
196                     return
197             except Exception as e:
198                 time.sleep(1)
199                 print(f"{api_callback_sell}")
200                 print(f"{e}")
201                 pass
```

Kuva 18. Rajapintakyselyn käsittely myyntitapahtumissa

4.2.6 Myyntiposition avaaminen lyhyeksi myyntiä varten

Lyhyeksi myynnissä puhutaan position avaamisesta, koska tuotetta ei varsinaisesti osteta, vaan sitä lainataan toiselta pörssissä toimivalta henkilöltä. Kun positio suljetaan voitollisesti, palautetaan lainatut Bitcoinit lainaajalle ja voitto on lainaus- ja palautushinnan erotus. Kun positio suljetaan tappiolla, Bitcoinit palautetaan ja tappio on lainaus- ja palautushinnan erotus.

Strategia-luvussa todettiin, että laskumarkkina tarkoittaa sitä, että nopeampi indikaattori on hitaamman indikaattorin alapuolella. Kuten nousumarkkinassa, myös laskumarkkinassa ensin tarkastellaan indikaattoreiden tilannetta (Kuva 19, s. 30, rivi 105). Toimintalogiikan mukainen edellisen tapahtuman tarkastelu tapahtuu metodissa check_latest_order_in_downtrend (Kuva 20, s. 30).

```
105             elif averages[1]["ema9"] < averages[1]["wma30"]:
106                 if self.downtrend:
107                     print("CONTINUING DOWNTREND")
108                     self.check_latest_order_in_downtrend(averages, one_before_newest, data, last_closed_order, last_closed_margin_order)
109                 else:
110                     print("TREND CHANGE! DOWNTREND!")
111                     self.long_buy_price = 0
112                     self.short_buy_price = 0
113                     self.check_latest_order_in_downtrend(averages, one_before_newest, data, last_closed_order, last_closed_margin_order)
114                     self.downtrend = True
115                     self.uptrend = False
```

Kuva 19. Laskutrendin tunnistaminen ja siinä toimiminen.

```
129     def check_latest_order_in_downtrend(self, averages, one_before_newest, data, last_closed_order, last_closed_margin_order):
130         if self.loan_open == False:
131             # and self.traded_this_trend == False:
132             print("Checking if should short!2")
133             self.check_if_should_short(averages, one_before_newest, data)
134         elif self.loan_open == True:
135             self.stop_loss_value =  round(averages[1]["wma30"] * 1.005, 1) # Moving stop loss value
136             print("Checking if should sell short position!2")
137             self.check_if_should_sell_short()
```

Kuva 20. Edellisen tapahtuman tarkastelu laskumarkkinassa.

```
353     def check_if_should_short(self, averages, data_from_latest_candle, data):
354         trend_strength = round(averages[0]["ema9"] / averages[0]["wma30"], 4)
355         trend_is_strong = False
356         if trend_strength <= 0.997:
357             trend_is_strong = True
358         current_price = 0
359         if float(data_from_latest_candle[4]) < averages[1]["wma30"] and float(data_from_latest_candle[4]) > averages[1]["ema9"]:
360             self.short_buy_price = float(data_from_latest_candle[3])
361             self.stop_loss_value = round(averages[0]["wma30"] * 1.005, 1)
362             print("Last complete candle closed in between indicators SHORT")
363             current_price = float(self.get_current_value())
364             if current_price <= self.short_buy_price and trend_is_strong:
365                 print(f"Setting short on {data_from_latest_candle[3]}")
366                 print(f"With STOP LOSS of {averages[0]['wma30']}")
367                 volume = 0
368                 while True:
369                     try:
370                         max_loan = self.api.get_max_margin_loan(asset='BTC')
371                         volume = self.truncate(float(max_loan["amount"]) * 0.5, 5)
372                         break
373                     except Exception:
374                         time.sleep(1)
375                         pass
376                 self.set_short_order(current_price, volume)
377             else:
378                 print(f"current price {current_price}")
379                 print(f"trend is strong = {trend_is_strong} strength =  {trend_strength}")
380         elif self.stop_loss_value != 0 and self.short_buy_price != 0 and trend_is_strong:
381             current_price = float(self.get_current_value())
382             if current_price <= self.short_buy_price:
383                 print(f"Setting short on {current_price}")
384                 print(f"With STOP LOSS of {self.stop_loss_value}")
385                 volume = 0
386                 while True:
387                     try:
388                         max_loan = self.api.get_max_margin_loan(asset='BTC')
389                         volume = self.truncate(float(max_loan["amount"]) * 0.5, 5)
390                         break
391                     except Exception:
392                         time.sleep(1)
393                         pass
394                 self.set_short_order(current_price, volume)
395         else:
396             print(f"current price was not set")
397             print(f"trend is strong = {trend_is_strong} strength =  {trend_strength}")
```

Kuva 21. Lyhyeksi myynnin kriteerien tarkastelu.

Samalla tavoin, kuin nousumarkkinassa, hinnan tulee sulkea indikaattoreiden väliin (Kuva 21 rivi 359). Mikäli tämä tapahtuu, mutta nykyinen hinta ei ole strategian

vaatimalla tasolla, otetaan hintatasot muistiin ja niitä tarkastellaan tulevilla kierroksilla kuvan 21 (s. 30) rivillä 380.

Metodi check_if_should_short kutsuu metodia set_short_order, kun hinta alittaa short_buy_price-muuttujan arvon. Set_short_order -metodi tuottaa rajapintakyselyn sisällön ja kutsuu kyselyn tekevää metodia short_order_api_call (Kuva 23, s. 32). Kun rajapintakysely on toteutettu, tavoitehinta asetetaan edellisen toteutuneen tapahtuman hinnan perusteella.

```
399     def set_short_order(self, price, volume):
400         while True:
401             try:
402                 print("Trying to short 1")
403                 api_call_data= {
404                     "symbol" : "BTCBUSD",
405                     "side" : "SELL",
406                     "type" : "LIMIT",
407                     "quantity" : volume,
408                     "price" : price,
409                     "sideEffectType" : "MARGIN_BUY",
410                     "timeInForce" : "GTC"
411                     }
412                 data = None
413                 with open("trades.json", "r", encoding="utf-8") as f:
414                     data = json.loads(f.read())
415                 data["trades"].append(api_call_data)
416                 with open("trades.json", "w", encoding="utf-8") as f:
417                     f.write(json.dumps(data, indent=4))
418                 self.short_order_api_call(api_call_data)
419                 self.target_price = round(price * self.short_win_target, 1)
420                 self.shorted_amount = self.get_borrowed_btc_margin()
421                 self.loan_open = True
422                 #self.traded_this_trend = True
423                 print(f"Short order set. Target price : {self.target_price}")
424                 return
425             except Exception as e:
426                 time.sleep(1)
427                 print(f"{e}")
428                 pass
```

Kuva 22. Rajapintakyselyn sisällön tuottaminen ja lyhyeksi myynnin jälkeisten toimenpiteiden tekeminen.

```
430     def short_order_api_call(self, data):
431         """
432         This is a separate method for handling buying via the api.
433         """
434         while True:
435             try:
436                 api_callback_buy = self.api.create_margin_order(**data)
437                 print("API CALL BACK WAS RECEIVED")
438                 closed_order = self.get_latest_order_margin(api_callback_buy["clientOrderId"])
439                 if closed_order == True:
440                     return closed_order
441             except Exception as e:
442                 time.sleep(1)
443                 print(f"{e}")
444                 pass
```

Kuva 23. Rajapintakyselyn tekeminen lyhyeksi myytäessä.

4.2.7 Myyntiposition sulkeminen laskumarkkinassa

Position avaamisen yhteydessä asetetaan tavoitehinta, jonka alittuessa positio suljetaan. Stop loss-arvo seuraa WMA30-indikaattoria myös laskumarkkinassa ja sen ylittyessä positio suljetaan. Kuvassa 24 (s. 33) metodi check_if_should_sell_short tarkastelee nykyistä hintaa tavoitehintaa, sekä stop loss-arvoa vasten. Jos positio tulee sulkea, kutsutaan metodia set_sell_short_order, joka tuottaa rajapintakyselyssä tarvittavan datan. Rajapintakäsittely position sulkemista varten tehdään metodissa short_sell_order_api_call (Kuva 25, s. 34). Position sulkemisen jälkeen hinnan tarkasteluun käytettyjen muuttujien arvot nollataan.

```
297     def check_if_should_sell_short(self):
298         current_price = float(self.get_current_value())
299         if current_price <= self.target_price:
300             self.set_sell_short_order(current_price)
301         elif current_price >= self.stop_loss_value:
302             self.set_sell_short_order(current_price)
303 
304     def set_sell_short_order(self, price):
305         while True:
306             try:
307                 volume = self.shorted_amount
308                 api_call_data= {
309                     "symbol" : "BTCBUSD",
310                     "side" : "BUY",
311                     "type" : "LIMIT",
312                     "quantity" : bot.truncate(volume, 5),
313                     "price" : price,
314                     "sideEffectType" : "AUTO_REPAY",
315                     "timeInForce" : "GTC"
316                     }
317                 self.short_sell_order_api_call(api_call_data)
318                 data = None
319                 with open("trades.json", "r", encoding="utf-8") as f:
320                     data = json.loads(f.read())
321                 data["trades"].append(api_call_data)
322                 with open("trades.json", "w", encoding="utf-8") as f:
323                     f.write(json.dumps(data, indent=4))
324                 print("SELL WAS MADE!")
325                 self.stop_loss_value = 0
326                 self.target_price = 0
327                 self.shorted_amount = 0
328                 self.loan_open = False
329                 time.sleep(1)
330                 break
331             except Exception as e:
332                 time.sleep(1)
333                 print(f"{e}")
334                 pass
```

Kuva 24. Metodit position sulkupaikan tunnistamiselle, sekä rajapinnalle lähetettävän datan muodostamiselle ja sulun jälkitoimenpiteille.

```
336     def short_sell_order_api_call(self, data):
337         """
338         This is a separate method for handling short selling via the api.
339         """
340         api_callback_sell = None
341         while True:
342             try:
343                 api_callback_sell = self.api.create_margin_order(**data)
344                 closed_order = self.get_latest_order_margin(api_callback_sell["clientOrderId"])
345                 if closed_order == True:
346                     return closed_order
347             except Exception as e:
348                 print(data)
349                 time.sleep(1)
350                 print(f"{e}")
351                 pass
```

Kuva 25. Rajapintakäsittely myyntiposition sulkemiselle.

5 TESTAAMINEN JA TULOSTEN ARVIOINTI

Tämä luku käsittelee robotin testausta ja tuloksia.

5.1 Testaaminen, robotin toimita ja tulosten mittaaminen

Robottia testattiin käyttäen 60 minuutin aikaväliä hintadatassa. Mittausjakson kesto oli noin 7 päivää (29.12.2021-5.1.2022). Robotti oli testausjaksolla päällä vuorokauden ympäri. Funktionaalisesti robotti toimi täysin virheettömästi ja strategian mukaisesti, eli robotti kykeni toteuttamaan strategiaa mittausjaksolla virheettömästi ja keskeytyksettä.

Tulosten mittaaminen on haastavaa muutamasta syystä. Binancessa normaalit kaupat, eli suorat osto- ja myyntitapahtumat tapahtuvat eri tilillä, kuin velkavivulliset kaupat eli lyhyeksi myynnit. Rahan joutuu siis jakamaan kahden tilin kesken. Tämän lisäksi kaupat eivät aina toteudu täysimääräisesti, joka yhdessä erotettujen tilien kanssa johtaa siihen, että kahdella tilillä voi olla samanaikaisesti kahta eri valuttaa. Bitcoinin volatiliteetti tuo laskuihin oman haastavuutensa, sillä hinta heitteli tarkastelujaksolla erittäin voimakkaasti.

5.2 Tulokset

Robotti ei kyennyt tuottamaan voittoa. Testausjakson aikana robotti teki seitsemän kauppaa, joista yksi oli voitollinen. Testausjakson alussa normaalilla tilillä oli 53.88 BUSD-vakaavaluutta, jonka arvo on sidottu Yhdysvaltain dollariin, sekä 0.00000275 Bitcoinia. Tilillä, jota käytetiin lyhyeksi myyntiin oli 55.09 BUSD-vakaavaluuttaa, eikä yhtään Bitcoinia. Testausjakson loputtua normaalilla tilillä oli 52.39 BUSD-vakaavaluutta, sekä 0.00000029 Bitcoinia. Toisella tilillä oli testausjakson loputtua 53.31 BUSD-vakaavaluuttaa, eikä yhtään Bitcoinia. Näiden lukujen perusteella robotin tulos oli noin 3 prosenttia tappiota.

Kun verrataan kaupattavan instrumentin hintakehitystä, niin robotti pärjäsi kohtalaisesti. Testausjaksolla Bitcoinin arvo laski noin 12 prosenttia. Bitcoinia testausjakson yli omistaneen henkilön omistuksen arvon kehitys olisi siis ollut 4 kertaa huonompaa, kuin robotilla kauppaa tekevän. Toisaalta robotin tulisi yrittää hyötyä arvon laskusta lyhyeksi myymällä, joten tyytyväinen tulokseen ei voi olla. Tähän seikkaan paneudutaan päätännössä.

6 PÄÄTÄNTÖ

Opinnäytetyön vaatimuksena oli, että robotti kykenee toteuttamaan ennalta määrättyä strategiaa lasku- ja nousumarkkinassa siten, että sen toiminta ei keskeydy. Tavoitteena oli, että toiminta on voitollista. Edellisen kappaleen pohjalta voidaan todeta, että vaatimukset täyttyivät, mutta tavoitteeseen ei päästy.

Tutkimuskysymykset olivat ”Kykeneekö yksinkertaisin indikaattorein ja vähällä datalla toteutettu robotti arvonluontiin?” ja ”Onko kannattavaa tuottaa robotti siinä tarkoituksessa, että sen varaan voisi tulevaisuudessa luottaa osan sijoituspääomasta?”. Tämän työn perusteella vastaus molempiin kysymyksiin on kielteinen. Seuraavissa kappaleissa käydään läpi osatekijöitä, jotka vaikuttivat robotin tulokseen ja joita tarkastelemalla voisi olla mahdollista muuttaa tutkimuskysymysten vastauksia.

Syitä siihen, että tavoitteeseen ei päästy, on monia. Mittausjaksolla Bitcoinin hinnan heilunta oli erittäin voimakasta. Vaikka luvussa 2 mainitsin, että Bitcoinin volatiliteetti mahdollistaa isommat kaupat, koitui se isoksi ongelmaksi strategian valinnan takia. Bitcoinin volatiliteetti aiheuttaa sen, että hinta leikkaa indikaattorit todella usein. Tämä johtaa siihen, että kauppoja sulkeutuu useammin, silloinkin, kun ei ehkä pitäisi. Volatiliteetti johtaa myös jyrkkiin hinnan nousu- ja laskukäyriin, jolloin strategian mukaisia ostopaikkoja ei ilmaannu. Tämä on syy sille, miksi tarkastelujakson aikana oli niin vähän kauppoja. Tarkastelujakso oli myös äärimmäisen lyhyt. Luotettavamman tuloksen saamiseksi tarvittaisiin huomattavasti pidempi testausjakso, mutta valitettavasti se ei työn puitteissa ole mahdollista.

Vaikka robotti ei kykene ainakaan toteutetun testin perusteella tekemään voittoa, koodissa on paljon sellaista, jota voi käyttää uudelleen tulevissa projekteissa. Uskon, että strategia on toimiva, mutta ei sovi yhteen valitun instrumentin kanssa. Robottia voisi siis testata vähemmän volatiileilla instrumenteilla, mutta markkinoiden toiminnasta johtuen voi voittojen saaminen silti olla vaikeaa. Jos työ pitäisi tehdä uudelleen siten, että pääsisin tavoitteeseen, pyrkisin rakentamaan kevyen

tekoälyn, joka tunnistaa tuki- ja vastustustasoja, jotka toimisivat osto- tai myyntipaikkoja. Tämä olisi hintavälin perusteella sijoittamista, jota käsiteltiin luvussa 2.

Kokonaisuutena olen lopputulokseen tyytyväinen. En tiennyt mitään päiväkaupankäynnistä työtä aloittaessa, joten henkilökohtainen tavoitteeni oppia aiheesta täyttyi. Kykenen soveltamaan oppimaani ja edellä mainittu ideani tekoälyn kehityksestä on sellainen, jonka aion toteuttaa.

LÄHTEET

Mordor Intelligence. 2021. Algorithmic trading market – growth, trends, covid-19 impact, and forecast (2021-2026). https://www.mordorintelligence.com/industry-reports/algorithmic-trading-market

Visma. 2021. Pörssi – Mikä on pörssi? https://www.visma.fi/epasseli/kirjanpidon-sanakirja/p/porssi/

Visma. 2021. Osake – Mikä on osake? https://www.visma.fi/epasseli/kirjanpidon-sanakirja/o/osake/

Saastamoinen, A. Junttila, J. Kurki, R. 2014. Mikä on bitcoin? https://yle.fi/aihe/artikkeli/2014/01/30/bitcoin-paljon-puhuttu-huonosti-tunnettu

Joutsen, N. 2021. Kryptovaluutta – Mitä se on ja miten se toimii käytännössä? https://www.vertaaensin.fi/blog/kryptovaluutta

Reed, E. 2021. Crypto vs. Stocks: Which is better? https://finance.yahoo.com/news/crypto-vs-stocks-better-201720889.html?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAAL1lDF6dJJ8MVHM0a51c16ObTIXpQzJYc-98zV0SokBepH4m4bByLmYJ47Kvo9Dzr21s6NqTts2EaZIaNW5WEAlho6Bu9BVpnG5ommmjffVzXrHt5rv9S38ZMJL3U-QlAOMBivFSK7EsbSkGLmhkfqHKZo1l6hQ67h5Sggh2hNLr

Sijoitustieto. 2018. Volatiliteetti – Markkinoiden pelkokerroin. https://www.sijoitustieto.fi/volatiliteetti

Heikinheimo, H. 2021. Tekninen analyysi – 3 tärkeintä indikaattoria. https://www.sijoittaja.fi/276210/tekninen-analyysi/

Mitchell, C. 2022. Trend Trading: The 4 most common indicators. https://www.investopedia.com/articles/active-trading/041814/four-most-commonlyused-indicators-trend-trading.asp

Hayes, A. 2021. Trading strategy. https://www.investopedia.com/terms/t/trading-strategy.asp

Viitala, T. 2022. Päiväkauppa – Aloita Treidaus 2022. https://sijoitusrahastot.org/paivakauppa/

CMC Markets. 2022. Trading strategies every trader should know. https://www.cmcmarkets.com/en/trading-guides/trading-strategies

FxPro-Finland. 2022. Skalppaus(Scalping). https://www.fxpro-finland.com/help-section/traders-glossary/scalping

Nordnet. 2022. Mitä lyhyeksi myynti tarkoittaa? https://www.nordnet.fi/faq/666-mitae-lyhyeksi-myynti-tarkoittaa

Trading Strategy Guides. 2020. 9/30 Trading Strategy – Pro Traders Want To Hide This Setup from You. https://tradingstrategyguides.com/9-30-trading-strategy/

Arora, S. 2022. Python Programming Language: Step- by-Step Guide 2020. https://hackr.io/blog/python-programming-language

Beal, V. 2022. High-Level Programming Language. https://www.webopedia.com/definitions/high-level-language/

Dev. 2020. Why python Programming Language Is So Popular In 2020. https://www.howtopython.org/why-python-programming-language-popular-2020/

Peters, K. 2021. Binance Exchange. https://www.investopedia.com/terms/b/binance-exchange.asp

Chen, J. 2021. Fiat Money. https://www.investopedia.com/terms/f/fiatmoney.asp

Hayes, A. 2022. Stablecoin. https://www.investopedia.com/terms/s/stablecoin.asp

Hoffman, C. 2021. What Is an API, and How Do Developers Use Them? https://www.howtogeek.com/343877/what-is-an-api/

Kohti taloudellista riippumattomuutta. 2010. Johdatus Tekniseen Analyysiin Osa 1: Liukuvat keskiarvot. http://www.taloudellinenriippumatto-muus.com/2010/05/johdatus-tekniseen-analyysiin-osa-1.html