LAPIN AMK
Lapland University of Applied Sciences

ALGORITMISEN KAUPANKÄYNNIN STRATEGIATUTKIMUS PYTHON-YMPÄRISTÖSSÄ

Jokiaho Pekka

Opinnäytetyö

Tieto- ja viestintätekniikka
Insinööri (AMK)

2025

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tieto- ja viestintätekniikka
Insinööri (AMK)

Tekijä	Pekka Jokiaho	**Vuosi**	2025
Ohjaaja	Erkki Mattila		
Toimeksiantaja	LapinAMK		
Työn nimi	Algoritmisen kaupankäynnin strategiatutkimus Python-ympäristössä		
Sivumäärä	47 + 14		

Tämä opinnäytetyö keskittyy algoritmisen kaupankäynnin strategian kehittämiseen, testaamiseen ja validointiin Python-ohjelmointiympäristössä. Työssä toteutetaan kvantitatiivinen analyysi, jossa tutkitaan Mean Reversion -strategian toimivuutta historiallisessa Bitcoin/EUR-markkinadatassa vuosilta 2020–2024. Strategia perustuu teknisiin indikaattoreihin, kuten Bollinger Bands ja Relative Strength Index (RSI), ja sen suorituskyky arvioidaan Backtrader-kirjaston avulla.

Tutkimuksessa hyödynnetään useita validointimenetelmiä, kuten parametrien optimointia, robustisuustestausta ja walk-forward-analyysiä. Näiden avulla pyritään minimoimaan ylioptimoinnin riskiä ja varmistamaan strategian kyky yleistyä erilaisissa markkinaolosuhteissa.

Työ osoittaa, että Python tarjoaa tehokkaan ja joustavan alustan algoritmisen kaupankäynnin tutkimukselle ja kehitykselle. Tulosten perusteella voidaan todeta, että huolellisesti suunniteltu ja validoitu strategia voi tuottaa kilpailukykyistä suorituskykyä simuloidussa ympäristössä.

Avainsanat algoritminen kaupankäynti, Python, backtestaus, Mean Reversion, strategiaoptimointi, robustisuustestaus, walk-forward-analyysi

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Study Programme in Information and Communication Technology
Bachelor of Engineering

Author	Pekka Jokiaho	**Year**	2025
Supervisor	Erkki Mattila		
Commissioned by	LapinAMK		
Title	Algorithmic Trading Strategy Research in Python Environment		
Number of pages	47 + 14		

The aim of this thesis study was to explore the role of back testing in the development of algorithmic trading strategies within Python-based environment.

The study focused on a Mean Reversion strategy that utilizes the Bollinger Bands and Relative Strength Index (RSI) indicators. The strategy was implemented using the Backtrader framework and tested on historical Bitcoin market data from 2020 to 2024. The research followed a systematic development line including parameter optimization, robustness testing, and walk-forward analysis. The goal was to create a strategy that performs reliably under varying market conditions without being overfitted to historical data.

The results show that the final strategy yielded an 84% cumulative return over four years and demonstrated resilience in multiple validation tests. The findings highlight the importance of a well-structured back testing process in reducing overfitting risk and increasing confidence in strategy performance before deployment on live markets.

Keywords algorithmic trading, backtesting, Python, Backtrader, strategy optimization, walk-forward analysis, robustness testing

SISÄLLYS

1 JOHDANTO

Algoritminen kaupankäynti on mullistanut rahoitusmarkkinoiden toiminnan viime vuosikymmeninä (Treleaven, Galas & Lalchand 2013) Tämä kaupankäynnin muoto, jossa tietokoneohjelmat toteuttavat automaattisesti ennalta määriteltyjä strategioita (Narang 2013), on nykyään keskeinen osa moderneja rahoitusmarkkinoita. Institutionaaliset toimijat, kuten pankit ja hedge-rahastot käyttävät algoritmeja suurten toimeksiantojen toteuttamiseen ja kvantitatiivisten strategioiden hyödyntämiseen. Samalla yhä useammat yksityissijoittajat ottavat käyttöön työkaluja omien strategioidensa rakentamiseen.

Erityisesti suurilla markkinapaikoilla algoritmien osuus kaupankäynnistä on kasvanut merkittävästi (European Securities and Markets Authority 2020), mikä on johtanut markkinoiden tehokkuuden ja likviditeetin paranemiseen (Hendershott, Jones & Menkveld 2011). Algoritmien rooli ei kuitenkaan rajoitu vain kaupankäyntiin vaan niitä käytetään yhä enemmän myös riskienhallintaan, arbitraasitilanteiden hyödyntämiseen ja markkinatiedon analysointiin.

Algoritmisen kaupankäynnin sovelluskohteet vaihtelevat laajasti. Institutionaaliset sijoittajat hyödyntävät algoritmeja esimerkiksi suurten volyymien hallintaan, kun taas kvantitatiiviset rahastot rakentavat monimutkaisia malleja, jotka analysoivat tuhansia signaaleja eri markkinaolosuhteissa. Teknologian kehityksen ja avoimen lähdekoodin ratkaisujen ansiosta myös yksityissijoittajat voivat nykyisin toteuttaa strategioita, jotka aiemmin olivat vain suurten toimijoiden ulottuvilla.

Python-ohjelmointikieli on noussut keskeiseen asemaan näissä tehtävissä sen selkeän syntaksin, laajan kirjastovalikoiman ja aktiivisen kehittäjäyhteisön ansiosta. Python tarjoaa tehokkaat työkalut datan käsittelyyn (pandas, NumPy), visualisointiin (matplotlib), koneoppimiseen (scikit-learn) ja strategioiden backtestaukseen (Backtrader, Zipline, QuantConnect). Tämä tekee siitä houkuttelevan vaihtoehdon myös kaupankäynnin automatisoinnin ja tutkimuksen välineenä. (López de Prado 2018.)

Erityisen tärkeää on ymmärtää, miten backtestauksen avulla voidaan arvioida ja parantaa strategioiden suorituskykyä ennen niiden käyttöönottoa reaalimarkkinoilla (Chan 2013). Tehottomasti toteutettu backtestaus voi johtaa virheellisiin johtopäätöksiin, etenkin jos testidata sisältää vinoumia tai jos strategia on optimoitu liiallisesti historialliseen dataan (Bailey, Borwein, López de Prado & Zhu 2014).

Tämän työn keskiössä on Mean Reversion -strategia, joka perustuu oletukseen siitä, että markkinahinnat palaavat pitkän aikavälin keskiarvoonsa (Gatev, Goetzmann & Rouwenhorst 2006; López de Prado 2018). Strategian toteutus yhdistää RSI-indikaattorin (Relative Strength Index) ja Bollinger Bandsin signaalit. Näiden avulla pyritään tunnistamaan ylilyöntitilanteita, joissa markkinahinta on poikennut normaalitasostaan.

Tutkimuksen tavoitteena on rakentaa, testata ja validoida kyseinen strategia Python-ympäristössä käyttäen historiallista Bitcoin/EUR-dataa vuosilta 2020–2024. Työssä pyritään vastaamaan seuraaviin tutkimuskysymyksiin:

- Kuinka tehokkaasti Mean Reversion -strategia voidaan optimoida ja validoida Python-ympäristössä
- Mitä rajoitteita liittyy strategian siirtämiseen reaalimaailmaan?

2 ALGORITMISEN KAUPANKÄYNNIN PERUSTEET JA BACKTESTAUS

2.1 Algoritminen kaupankäynti ja strategiat

Algoritminen kaupankäynti on muuttanut rahoitusmarkkinoiden toimintaa merkittävästi 2000-luvun alusta lähtien. Tietokoneohjelmien toteuttama automatisoitu kaupankäynti, jossa osto- ja myyntipäätökset perustuvat ennalta määriteltyihin sääntöihin, on noussut keskeiseksi osaksi erityisesti suurten markkinapaikkojen infrastruktuuria (Narang 2013; Treleaven ym. 2013). Tämä murros on lisännyt kaupankäynnin tehokkuutta, parantanut markkinalikviditeettiä ja laskenut kaupankäyntiin liittyviä transaktiokustannuksia (Hendershott ym. 2011; European Securities and Markets Authority 2020).

Vaikka keskeiset akateemiset lähteet ovat pääosin yli vuosikymmenen takaa, niiden esittämät perusmekanismit ovat edelleen merkityksellisiä algoritmisen kaupankäynnin ymmärtämiseksi. Teknologian nopean kehityksen vuoksi käytännön toteutukset ovat kuitenkin kehittyneet merkittävästi. Python-ohjelmointikieli on vakiinnuttanut asemansa suosituimpana välineenä strategiakehityksessä ja backtestauksessa, erityisesti sen kattavan kirjastovalikoiman, sekä laajan käyttäjäyhteisön ansiosta. (Chan 2013; McKinney 2017.)

Esimerkiksi kirjastot kuten Backtrader, Zipline ja QuantConnect mahdollistavat strategioiden simuloinnin, optimoinnin ja analysoinnin käyttäen historiallista dataa. Nämä kirjastot ovat saatavilla sekä institutionaalisille että yksityisille toimijoille, mikä on madaltanut kynnystä algoritmisen kaupankäynnin kokeilulle ja laajentanut sen käyttäjäkuntaa. Lisäksi Pythonin ekosysteemi tarjoaa sujuvan yhteensopivuuden koneoppimiskirjastojen (kuten scikit-learn tai TensorFlow) kanssa, mikä mahdollistaa yhä monimutkaisempien datalähtöisten strategioiden kehittämisen. (López de Prado 2018.)

Kehityksen myötä algoritminen kaupankäynti ei ole enää vain suurten pankkien ja hedge-rahastojen etuoikeus, vaan se on laajentunut kattamaan myös

yksityissijoittajat, jotka voivat rakentaa ja testata strategioita samoilla periaatteilla kuin ammattimaiset toimijat. Tämä demokratisoituminen on lisännyt markkinoiden älykkyyttä ja tarjonnut vaihtoehtoisia tapoja lähestyä sijoittamista sekä kaupankäynnin automatisointia.

2.1.1 Määritelmä ja peruskäsitteet

Algoritminen kaupankäynti tarkoittaa järjestelmällistä ja automaattista kaupankäyntiprosessia, jossa sekä kaupankäyntipäätökset että toimeksiantojen toteutus tehdään tietokoneohjelmien avulla. Algoritmit noudattavat ennalta määriteltyjä sääntöjä, jotka voivat perustua teknisiin indikaattoreihin, tilastollisiin malleihin, sääntöpohjaisiin logiikoihin tai kehittyneisiin koneoppimisalgoritmeihin (Chan 2013; López de Prado 2018). Tämän lähestymistavan etuna on kyky käsitellä suuria tietomääriä ja reagoida markkinamuutoksiin erittäin nopeasti ja johdonmukaisesti (Narang 2013). Erityisesti automatisoinnin avulla voidaan hallita inhimillisiä virheitä ja tunteiden vaikutusta kaupankäyntipäätöksiin, mikä parantaa johdonmukaisuutta ja skaalautuvuutta (Kissell 2013).

Algoritmisen kaupankäynnin strategian kehitysprosessi voidaan jäsentää kolmeen keskeiseen vaiheeseen (Bailey ym. 2014; López de Prado 2018). Ensimmäinen vaihe on markkinadatan keruu ja esikäsittely. Tässä vaiheessa kerätään historiallista hintadataa, volyymitietoa, uutisdataa ja muita mahdollisia muuttujia (esim. tekniset indikaattorit), minkä jälkeen data puhdistetaan ja muunnetaan analyysikelpoiseen muotoon. Tämä vaihe voi sisältää myös poikkeamien, puuttuvien arvojen ja datan laadun tarkastelun.

Toisessa vaiheessa kehitetään kaupankäyntilogiikka eli signaalien generointi. Tässä vaiheessa määritellään säännöt, joiden perusteella algoritmi antaa osto- tai myyntisignaaleja. Signaalit voivat perustua yksinkertaisiin ehtolauseisiin (esim. RSI < 30), tilastollisiin malleihin (esim. regressioanalyysi) tai kehittyneempiin koneoppimismalleihin (esim. XGBoost tai LSTM).

Kolmannessa vaiheessa määritellään toimeksiantojen toteutusmekanismi, jossa otetaan huomioon toteutuksen kustannukset, kuten bid-ask spreadit, slippage, latenssi ja mahdolliset transaktiokulut (Hasbrouck 2007; Kissell 2013).

Edistyneemmissä toteutusmekanismeissa voidaan käyttää algoritmeja, jotka hajauttavat toimeksiantoja eri ajankohtiin tai markkinapaikkoihin minimoidakseen markkinavaikutuksen (Almgren & Chriss 2000).

Python-ympäristö tarjoaa joustavat välineet koko strategiaprosessin toteuttamiseen. Esimerkiksi pandas-kirjasto mahdollistaa tehokkaan datan esikäsittelyn (McKinney 2017), NumPy tukee vektoripohjaista laskentaa ja scikit-learn tai TensorFlow mahdollistavat kehittyneiden koneoppimisstrategioiden käyttöönoton. Backtraderin avulla voidaan toteuttaa strategiasimulaatioita sekä backtestauksia, ja matplotlib tai seaborn tarjoavat visuaalisia keinoja tulosten analysointiin. Tämä kokonaisuus mahdollistaa empiiristen strategioiden nopean prototypoinnin, testaamisen ja arvioinnin yhdessä ohjelmointiympäristössä (Kissell 2013; López de Prado 2018).

2.1.2 Strategiatyypit ja niiden ominaispiirteet

Algoritmiset kaupankäyntistrategiat voidaan luokitella useisiin päätyyppeihin niiden toimintaperiaatteen perusteella. Jokaisella strategiaperheellä on omat vahvuutensa ja rajoitteensa, ja ne toimivat parhaiten tietyissä markkinaympäristöissä tai aikahorisonteissa. (Chan 2013; López de Prado 2018.)

Mean Reversion -strategiat perustuvat oletukseen siitä, että omaisuuserän hinta pyrkii palautumaan pitkän aikavälin keskiarvoonsa. Strategia tunnistaa ylilyöntejä, joissa hinta poikkeaa normaalista vaihteluvälistään, ja pyrkii hyötymään tästä poikkeamasta ostamalla ”liian halpaa” tai myymällä ”liian kallista”. Tyypillisiä indikaattoreita ovat Bollinger Bands ja RSI (Bollinger, 2001; Gatev ym. 2006). Mean Reversion soveltuu erityisesti rauhallisiin, vaihteleviin markkinoihin, joissa likviditeetti on hyvä ja trendit eivät ole vahvoja.

Mean Reversion -strategioissa yksityissijoittajalla voi olla suhteellisesti paremmat mahdollisuudet kuin monissa muissa algoritmisen kaupankäynnin muodoissa. Koska strategia perustuu tekniseen analyysiin ja keskihajonnasta poikkeavien hintaliikkeiden tunnistamiseen, sen toteuttaminen ei vaadi äärimmäisen nopeaa infrastruktuuria tai suuria pääomia. Vaikka suurin osa

sijoittajista häviää myös mean reversion -strategioissa, yksityissijoittaja voi hyödyntää rauhallisempia markkinaolosuhteita ja käyttää yksinkertaisia indikaattoreita, kuten RSI:tä ja Bollinger Bandseja, ilman että hänen tarvitsee kilpailla millisekuntien tasolla suurten toimijoiden kanssa (Chan 2013). Näin ollen strategia voi tarjota realistisemman lähtökohdan yksityiselle toimijalle, vaikka riskit ja tappion mahdollisuus ovat edelleen merkittävät.

RSI (Relative Strength Index) on J. Welles Wilderin (1978) kehittämä momentti-indikaattori, joka mittaa hintaliikkeiden vahvuutta suhteessa nousu- ja laskupäivien määrään. Sen arvo vaihtelee välillä 0–100. Perinteisesti yli 70:n arvoa pidetään merkkinä yliostetusta markkinasta ja alle 30:n arvoa ylimyydystä markkinasta. RSI:n taustalla on ajatus, että voimakkaan nousun jälkeen markkina on lyhyellä aikavälillä todennäköisesti saavuttanut huippunsa, kun taas voimakkaan laskun jälkeen on todennäköistä, että hinta korjaa ylöspäin. Tämän vuoksi RSI on suosittu väline mean reversion -strategioissa, sillä se auttaa tunnistamaan tilanteet, joissa hinnat voivat palautua kohti keskiarvoaan. (Wilder 1978.)

Bollinger Bands puolestaan perustuvat liukuvaan keskiarvoon ja sen ympärille muodostettuihin ylä- ja alarajoihin, jotka lasketaan tyypillisesti kahden keskihajonnan etäisyydelle keskiarvosta (Bollinger 2001). Kun hinta nousee nauhojen yläpuolelle, markkinoiden katsotaan olevan yliostetussa tilassa, ja kun hinta laskee alapuolelle, markkinat nähdään ylimyytyinä. Bollinger Bandsin taustalla on tilastollinen oletus, että suurin osa havainnoista osuu tietyn keskihajonta-alueen sisälle. Kun hinta ylittää tämän alueen, on todennäköistä, että se palaa takaisin kohti keskiarvoa. Tämän vuoksi indikaattori tukee mean reversion -logiikkaa erityisen hyvin.

RSI:n ja Bollinger Bandsin yhdistäminen tarjoaa sijoittajalle kaksinkertaisen varmistuksen: kun molemmat indikaattorit viittaavat markkinoiden yli- tai alilyönteihin, voidaan muodostaa vahvempi kaupankäyntisignaali. Tämä parantaa todennäköisyyttä, että havaittu poikkeama todella korjaantuu, mikä tekee yhdistelmästä hyödyllisen lähtökohdan mean reversion -strategioiden toteuttamiseen.

Momentum-strategiat lähtevät siitä oletuksesta, että hintaliikkeillä on taipumus jatkua samaan suuntaan tietyn ajan. Strategiat pyrkivät seuraamaan trendejä ostamalla omaisuuseriä, jotka ovat nousussa, ja myymällä niitä, jotka ovat laskussa (Narang 2013). Indikaattoreina voidaan käyttää esimerkiksi liukuvia keskiarvoja (Moving Averages), MACD:tä tai suoraa hinnan kehitystä. Momentum-strategiat sopivat erityisesti volatiliteettia sisältäviin ja trendikkäisiin markkinaympäristöihin.

Arbitraasistrategiat hyödyntävät hintaeroja eri markkinapaikkojen, instrumenttien tai johdannaisten välillä. Esimerkiksi samaa osaketta voidaan ostaa yhdeltä pörssiltä ja myydä toisella, mikäli hinnat poikkeavat toisistaan. Arbitraasi vaatii nopeaa toteutusta, alhaisia transaktiokustannuksia ja matalaa latenssia (Chan 2013). Tällaiset strategiat ovat yleisiä korkeataajuuskaupankäynnissä ja institutionaalisten toimijoiden käytössä.

Market Making -strategiat perustuvat likviditeetin tarjoamiseen markkinoilla. Markkinatakaaja asettaa samanaikaisesti osto- ja myyntitarjouksia ja ansaitsee voittoa bid-ask-spreadin kautta. Näissä strategioissa riskienhallinta on tärkeää, sillä varaston hallinta (*inventory management*) ja hinnan nopea liike voivat aiheuttaa tappioita (Hasbrouck 2007). Strategiat vaativat jatkuvaa läsnäoloa markkinoilla ja soveltuvat parhaiten korkealiikkeisiin ja likvideihin instrumentteihin.

Yksityissijoittajien mahdollisuudet menestyä momentum-, arbitraasi- ja market making -strategioissa ovat rajalliset verrattuna institutionaalisiin toimijoihin. Suuret pankkiiriliikkeet ja hedge-rahastot panostavat huomattavia resursseja algoritmien kehittämiseen sekä tietojärjestelmien ja tiedonsiirron nopeuteen. Kilpailu on äärimmäisen kovaa, ja etu on usein sillä toimijalla, jolla on lyhin ja nopein tiedonsiirtoreitti pörssin palvelimille. Esimerkiksi mikroaaltolinkkejä on rakennettu pelkästään sen vuoksi, että valo kulkee tyhjiössä hieman nopeammin kuin valokuidussa (Aldridge 2013, 25-27, 175–176) Tällaisessa ympäristössä yksityissijoittajalla on vaikeaa saavuttaa kilpailuetua, sillä infrastruktuuri-investoinnit ja alhainen latenssi muodostavat merkittävän kynnyksen. Tämä ei kuitenkaan tarkoita, etteikö yksityissijoittaja voisi oppia ja

kokeilla näitä strategioita tutkimus- ja harjoittelutarkoituksessa, mutta realistiset tuotto-odotukset on syytä pitää maltillisina tai varautua jopa koko pääoman menettämiseen.

News Trading -strategiat hyödyntävät äkillisiä markkinaliikkeitä, joita syntyy uutisten, taloustietojen tai muiden ulkoisten tapahtumien seurauksena. Strategioissa voidaan käyttää koneoppimista ja NLP-menetelmiä uutisvirran analysointiin ja signaalien muodostamiseen (López de Prado 2018). Toteutus vaatii tehokasta datan reaaliaikaista käsittelyä ja nopeaa reagointia, mikä tekee strategiasta teknisesti haastavan, mutta potentiaalisesti erittäin tuottoisan.

Python-ympäristö tarjoaa työkalut kaikkien näiden strategiatyyppien kehittämiseen ja testaamiseen. Kirjastot kuten Backtrader, Zipline ja QuantConnect mahdollistavat sekä signaalien muodostuksen että simulaation historiallisella datalla. Kehittäjä voi yhdistellä useita lähestymistapoja rakentaakseen monimutkaisia, monistrategisia järjestelmiä, joita voidaan optimoida ja validoida tehokkaasti.

2.1.3 Markkinoiden mikrorakenne

Algoritmisen kaupankäynnin menestyksekäs toteutus edellyttää syvällistä ymmärrystä markkinoiden mikrorakenteesta, eli miten kaupat toteutuvat, miten hinnat muodostuvat ja kuinka markkinat reagoivat eri tyyppisiin toimeksiantoihin. Mikrorakenneanalyysi keskittyy markkinoiden teknisiin ja toiminnallisiin yksityiskohtiin, joita ovat muun muassa tarjouskirjat, hinnanmuodostuksen logiikka, toimeksiantojen toteutusalgoritmit ja markkinasyvyys. (O'Hara 1995; Hasbrouck 2007.)

Tarjouskirja (*order book*) sisältää reaaliaikaisen näkymän kaikista avoimista osto- ja myyntitarjouksista. Sen rakenne vaikuttaa suoraan siihen, kuinka likvidi markkina on ja kuinka suuret toimeksiannot vaikuttavat hintaan. Bid-ask-spread (osto- ja myyntihinnan välinen ero) toimii kaupankäynnin kustannuksen mittarina. Mitä pienempi spread, sitä edullisemmin kaupat voivat toteutua. Slippage puolestaan viittaa tilanteeseen, jossa toimeksianto toteutuu eri hinnalla kuin oli tarkoitus, mikä voi johtua nopeista hinnanmuutoksista tai

riittämättömästä likviditeetistä. Tämä on keskeinen ongelma erityisesti strategioissa, jotka toimivat pienillä aikajänteillä tai käyttävät suuria kaupankäyntimääriä. (Hasbrouck 2007; Kissell 2013.)

Toinen olennainen tekijä on latenssi (*latency*), eli viive datan saapumisen, algoritmin päätöksenteon ja toimeksiannon toteutuksen välillä. Viiveet voivat johtaa siihen, että algoritmi tekee päätöksiä vanhentuneen tiedon perusteella; mikä on kriittinen haaste erityisesti korkeataajuuskaupankäynnissä (*High Frequency Trading*, HFT), jossa nanosekunnin erot voivat ratkaista voiton tai tappion. (Hasbrouck 2007.)

Python-ympäristössä voidaan rakentaa realistisia simulaatioita, joissa mikrorakenteen eri osa-alueet huomioidaan. Esimerkiksi tick-tason data mahdollistaa yksittäisten hintamuutosten ja volyymien seuraamisen tarkasti. Vielä tarkempaa mallintamista tarjoaa Level II -data, joka sisältää tietoa eri hintatasoilla olevista volyymeistä ja toimeksiannoista. Näiden avulla voidaan analysoida, kuinka strategia käyttäytyisi todellisessa markkinatilanteessa, mukaan lukien viiveet, likviditeettipuutteet ja slippage.

Ymmärrys markkinoiden mikrorakenteesta ei ainoastaan paranna strategian testaamisen laatua vaan se on edellytys sille, että simuloitu suorituskyky voisi vastata käytännön kaupankäyntiä. Tämän teoreettisen pohjan avulla voidaan siirtyä empiiriseen osuuteen, jossa kehitetään, optimoidaan ja validoidaan konkreettinen Mean Reversion -strategia Python-ympäristössä. Sen rakenne ja syvyys vaikuttavat kaupankäynnin likviditeettiin. Bid-ask spread, eli oston ja myynnin välinen hintaero määrittää toimeksiantojen kustannuksia. Slippage kuvaa toteutushinnan poikkeamaa suunnitellusta hinnasta, mikä voi johtua markkinoiden nopeista liikkeistä tai riittämättömästä likviditeetistä. Viiveet (*latency*) puolestaan voivat aiheuttaa sen, että algoritmi reagoi vanhentuneeseen tietoon, mikä on erityisen ongelmallista korkeataajuuskaupankäynnissä (HFT). (O'Hara 1995; Hasbrouck 2007; Kissell 2013.)

Python-ympäristössä voidaan rakentaa simulaatioita, joissa nämä tekijät huomioidaan realistisesti. Tick-data tai Level II -data mahdollistaa tarkan

simuloinnin, mukaan lukien hintatasot ja volyymit, joilla toimeksiannot oikeasti toteutuisivat.

Bid-ask-spreadin merkitys sijoittajalle perustuu siihen, että kaupat toteutuvat aina osto- ja myyntihinnan välissä olevaan kohtaan. Jos sijoittaja ostaa arvopaperin välittömästi markkinahinnalla, hän maksaa myyntilaidassa olevan hinnan (*ask*). Jos hän taas haluaa myydä heti, hän saa ostolaidassa olevan hinnan (*bid*). Erotus näiden kahden hinnan välillä muodostaa välittömän kustannuksen: sijoittaja on heti kaupanteon jälkeen tappiolla spreadin verran. Mitä kapeampi spread on, sitä pienempi on tämä alkuvaiheen tappio ja sitä kustannustehokkaampaa kaupankäynti on. Laaja spread puolestaan kasvattaa kustannuksia ja voi tehdä lyhyen aikavälin strategioista kannattamattomia, sillä jokainen kauppa alkaa suuremmalla negatiivisella tuotolla. Tästä syystä bid-ask-spreadiä pidetään likviditeetin mittarina: likvidillä markkinalla spread on kapea ja kaupankäynti edullista, kun taas epälikvidillä markkinalla spread levenee ja kustannukset kasvavat. (O'Hara, 1995; Hasbrouck, 2007.)

Slippage tarkoittaa tilannetta, jossa toimeksianto toteutuu eri hintaan kuin sijoittaja alun perin odotti. Tämä johtuu tyypillisesti nopeista hinnanmuutoksista tai siitä, ettei markkinoilla ole riittävästi vastapuolia toimeksiannon toteuttamiseen halutulla tasolla. Esimerkiksi markkinatoimeksianto toteutetaan aina parhaaseen saatavilla olevaan hintaan, mutta jos tarjouskirjassa ei ole riittävää volyymiä, osa kaupasta toteutuu seuraaville hintatasoille. Tällöin toteutushinta poikkeaa suunnitellusta ja kaupankäynnin kustannukset kasvavat. Slippage on erityisen merkittävä ongelma strategioissa, jotka toimivat pienillä aikajänteillä tai suurilla volyymeilla, koska hinnat voivat liikkua jo muutamien millisekuntien sisällä. Tämän riskin hallintaan voidaan käyttää limiittitoimeksiantoja, joissa sijoittaja määrittelee hinnan, jolla on valmis käymään kauppaa. Limiittitoimeksianto voi kuitenkin jäädä toteutumatta, jos markkinahinta ei saavuta määriteltyä tasoa. Slippage muodostaa siten keskeisen kustannustekijän algoritmisessa kaupankäynnissä, ja sen mallintaminen realistisesti on olennainen osa strategian testausta ja arviointia. (Kissell 2013; López de Prado 2018.)

2.2 Python-alustainen kaupankäyntikehys

Python on nykyaikainen, tehokas ja laajasti käytetty ohjelmointikieli, joka on saavuttanut keskeisen aseman rahoitusalan sovelluksissa, erityisesti algoritmisessa kaupankäynnissä (Chan 2013; López de Prado 2018). Sen avoin lähdekoodi, selkeä syntaksi, matala oppimiskynnys ja laaja kirjastoekosysteemi tekevät siitä houkuttelevan työkalun tutkijoille ja sijoitusalan ammattilaisille.

Python tukee laajasti eri omaisuusluokkia, kuten osakkeita, valuuttapareja, futuureja ja kryptovaluuttoja, ja tarjoaa kattavat työkalut koko kaupankäyntiputken toteuttamiseen. Kirjastot kuten yfinance, ccxt ja alpaca-trade-api mahdollistavat datan keruun eri markkinoilta, kun taas pandas, NumPy, TA-Lib, Backtrader ja Pandas TA tukevat datan käsittelyä, teknistä analyysiä ja strategian testaamista. Tulosten visualisointiin voidaan käyttää esimerkiksi matplotlib-, seaborn- ja plotly-kirjastoja. (McKinney 2017.)

Pythonin joustavuus mahdollistaa kokonaan kustomoitujen kaupankäyntijärjestelmien rakentamisen ilman tarvetta kaupallisille ohjelmistoille. Tämä sisältää automaattisen datanlatauksen, strategialogiikan, backtestauksen, riskienhallinnan ja raportoinnin. Lisäksi Python integroituu hyvin pilvipohjaisiin ympäristöihin ja tukee koneoppimisen sekä syväoppimisen sovelluksia esimerkiksi scikit-learn-, XGBoost- ja TensorFlow-kirjastojen kautta. (López de Prado 2018.)

2.2.1 Pythonin ohjelmointikirjastot automatisoituun kaupankäyntiin

Pythonin käyttö automatisoidussa kaupankäynnissä perustuu laajaan ja kasvavaan valikoimaan erikoistuneita kirjastoja. Nämä mahdollistavat strategian suunnittelun, datankeruun, testaamisen, optimoinnin ja suoran markkinainteraktion. Yhdessä nämä työkalut muodostavat perustan kokonaisvaltaiselle strategiaputkelle: datan hakemisesta ja esikäsittelystä kaupankäyntisignaalien tuottamiseen ja tulosten analysointiin. Pythonin vahvuus piilee juuri sen ekosysteemissä, joka mahdollistaa näiden osien saumattoman yhdistämisen.

Backtrader on erityisen suosittu Python-kirjasto, joka tarjoaa joustavan arkkitehtuurin strategiakehitykseen ja takautuvaan testaukseen. Sen avulla voidaan yhdistää useita datasarjoja, toteuttaa kaupankäyntisignaaleja useiden indikaattoreiden avulla, asettaa tilaustyyppejä ja hallita positioita dynaamisesti. Backtrader tukee myös visualisointia ja useita aikakehyksiä, mikä tekee siitä soveltuvan sekä yksinkertaisiin että monimutkaisiin strategioihin. (López de Prado 2018.)

Zipline on toinen laajalti käytetty kirjasto, jota erityisesti Quantopian ympäristössä hyödynnettiin. Se soveltuu erityisesti osakemarkkinoiden backtestaukseen ja on rakennettu yhteistyöhön Pandasin ja NumPyn kanssa. Ziplineä käytetään yhä akateemisissa projekteissa, ja se on tunnettu yksinkertaisuudestaan. (Chan, 2013.)

QuantConnect, joka toimii Lean-moottorilla, tarjoaa Python API:n pilvipohjaiseen backtestaukseen ja reaaliaikaiseen kaupankäyntiin. Se tukee useita omaisuusluokkia ja mahdollistaa strategioiden testaamisen yli vuosikymmenen pituisella datalla. (QuantConnect 2023.)

Ccxt on avoimen lähdekoodin kirjasto, jonka avulla voidaan muodostaa yhteyksiä satoihin kryptopörsseihin. Se mahdollistaa kaupankäyntistrategioiden toteuttamisen ja testaamisen kryptomarkkinoilla aidon rajapintadataliikenteen avulla. (GitHub, Inc 2023.)

2.2.2 Backtestaus ja optimointityökalut Pythonissa

Python tarjoaa tehokkaita työkaluja kaupankäyntistrategioiden kehittämiseen, testaamiseen ja optimointiin hyödyntämällä laajaa ja integroitua kirjastoekosysteemiä. Toteumatestaus eli backtestaus on keskeinen osa tätä prosessia, sillä sen avulla voidaan arvioida strategian suorituskykyä historiallisessa markkinaympäristössä ennen sen mahdollista käyttöönottoa. (Bailey ym. 2014.)

Backtestaus alkaa yleensä datan esikäsittelyllä. Pandas ja NumPy tarjoavat laajan valikoiman työkaluja datan siistimiseen, aggregointiin, normalisointiin ja

analysointiin. Näiden kirjastojen avulla voidaan esimerkiksi laskea teknisiä indikaattoreita, poistaa poikkeamia ja yhdistää useita datasarjoja. Tämän jälkeen strategia voidaan koodata esimerkiksi Backtrader-ympäristöön, joka tukee kaupankäyntilogiikan testaamista useilla eri aikajänteillä ja instrumenteilla. (López de Prado 2018.)

Visualisointi on tärkeä osa strategian tulkintaa ja kommunikointia. Matplotlib ja seaborn mahdollistavat laajasti mukautettavien kaavioiden luonnin, joilla voidaan kuvata esimerkiksi hintakehitystä, kaupankäyntisignaaleja, tuottokäyriä ja riskimittareita. Tämä auttaa ymmärtämään, missä tilanteissa strategia toimii ja missä se epäonnistuu.

Strategioiden optimointi tapahtuu usein parametrien säätämisen kautta. Python tarjoaa tähän useita lähestymistapoja. Yksinkertaisimmillaan voidaan käyttää silmukoita ja itertools.product-metodia kaikkien mahdollisten parametriyhdistelmien läpikäymiseen. Kehittyneempiä ratkaisuja ovat esimerkiksi Optuna-kirjasto, joka tukee bayesilaista optimointia ja hyperparametrien tehokasta säätöä, sekä GridSearchCV, joka on erityisesti koneoppimismallien parametritutkimukseen kehitetty työkalu. (Bailey ym. 2014.)

Optimoinnissa voidaan myös hyödyntää geneettisiä algoritmeja, kuten DEAP-kirjastoa, tai käyttää satunnaistettuja algoritmeja, jotka etsivät tehokkaasti globaalisti hyviä ratkaisupisteitä suurista parametriavaruuksista. Tällainen lähestymistapa vähentää riskiä joutua paikalliseen optimiin ja parantaa mahdollisuuksia löytää yleistettäviä strategioita. (López de Prado 2018.)

Python-ympäristön vahvuus on siinä, että koko kehitysprosessi aina datan esikäsittelystä backtestaukseen, visualisointiin ja optimointiin voidaan toteuttaa yhtenä kokonaisuutena. Tämä mahdollistaa toistettavat ja dokumentoitavat strategiatutkimukset, jotka tukevat avoimuutta ja tutkimuksellista luotettavuutta.

Pythonin itertools.product on sisäänrakennettu menetelmä, joka muodostaa karteesisen tulon eli kaikkien annettujen arvojoukkojen kaikki mahdolliset yhdistelmät. Tämä tekee siitä yksinkertaisen mutta tehokkaan työkalun parametrien optimointiin. Kun kaupankäyntistrategiassa halutaan testata

esimerkiksi eri arvoja RSI-indikaattorin pituudelle, Bollinger Bands -indikaattorin pituudelle sekä RSI:n ala- ja ylärajoille, itertools.product tuottaa näistä jokaisen mahdollisen yhdistelmän. Näin varmistetaan, että kaikki vaihtoehdot käydään systemaattisesti läpi ilman, että yksikään mahdollisuus jää huomaamatta. (Python Software Foundation 2023.)

Esimerkiksi, jos RSI:n pituudeksi valitaan arvot [10, 14] ja Bollinger Bands -indikaattorin pituudeksi arvot [15, 20], itertools.product muodostaa neljä eri yhdistelmää: (10,15), (10,20), (14,15) ja (14,20). Optimointivaiheessa kukin näistä yhdistelmistä ajetaan erikseen backtestin läpi, minkä jälkeen voidaan vertailla lopputuloksia ja löytää tuottoisin parametrikokonaisuus. Tämä lähestymistapa tunnetaan nimellä brute force -optimointi, koska se käy läpi kaikki mahdolliset vaihtoehdot. Menetelmä on laskennallisesti yksinkertainen ja soveltuu hyvin tilanteisiin, joissa parametrien määrä ja arvojoukot ovat rajallisia (Bailey ym. 2014).

2.3 Backtestaus menetelmänä

2.3.1 Backtestauksen periaatteet

Backtestaus on metodologinen lähestymistapa kaupankäyntistrategioiden testaamiseen historiallisella markkinadatalla. Sen perusperiaatteena on simuloida kaupankäyntistrategian toimintaa menneillä markkinaliikkeillä, mikä mahdollistaa strategian toimivuuden arvioinnin erilaisissa markkinaolosuhteissa. (Chan, 2013; López de Prado 2018.)

Backtestauksen avulla voidaan analysoida strategian tuottoja, riskejä ja käyttäytymistä markkinoiden eri vaiheissa. Prosessi sisältää strategian määrittelyn, historiallisen datan keräämisen, kaupankäyntisääntöjen implementoinnin ja tulosten analysoinnin (Bailey ym. 2014). Python-ympäristössä tämä prosessi voidaan toteuttaa esimerkiksi Backtrader-kirjastolla, joka mahdollistaa testien toistettavuuden, yksityiskohtaisen lokituksen ja visuaalisen tarkastelun. Backtestauksen keskeinen tavoite on tuottaa tilastollisesti merkittävää tietoa strategian toimivuudesta ja auttaa

tunnistamaan mahdolliset heikkoudet ennen strategian käyttöönottoa reaalimarkkinoilla (López de Prado 2018).

2.3.2 Datan laatu ja saatavuus

Backtestauksen luotettavuus riippuu merkittävästi käytetyn markkinadatan laadusta ja kattavuudesta. Laadukas markkinadata sisältää tarkat hinta- ja volyymitiedot sekä huomioi markkinoiden mikrorakenteen, kuten bid-ask-spreadit ja markkinasyvyyden (O'Hara 1995; Hasbrouck 2007). Pythonissa dataa voidaan hakea avoimista lähteistä, kuten yfinance, ccxt, Binance API tai Polygon.io, ja analysoida sitä pandas- ja NumPy-kirjastojen avulla.

Historiallisen datan tulee olla riittävän pitkältä aikajaksolta ja sisältää erilaisia markkinaolosuhteita, kuten nousevat ja laskevat trendit sekä sivuttaisliikkeet. Datan granulariteetin merkitys korostuu erityisesti lyhyen aikavälin strategioissa, joissa tarvitaan tick-tason tai minuutti-tason dataa kauppojen täsmälliseen mallintamiseen (López de Prado 2018). Korporaatiotapahtumat, kuten osingot, splitit ja fuusiot, tulee olla asianmukaisesti huomioitu datassa, jotta backtestauksen tulokset olisivat mahdollisimman todenmukaisia.

2.3.3 Backtestauksen rajoitteet

Backtestauksen tuloksia tulkittaessa on välttämätöntä ymmärtää menetelmän rajoitteet ja potentiaaliset harhat. Merkittävä rajoite on niin kutsuttu look-ahead bias (ennakointiharha), jossa strategia saattaa tahattomasti hyödyntää tulevaisuuden tietoa, joka ei olisi ollut saatavilla todellisessa kaupankäyntitilanteessa. (Bailey ym. 2014.)

Toinen keskeinen rajoite liittyy markkinavaikutuksen mallintamisen haasteisiin: backtestaus ei välttämättä pysty täysin huomioimaan, miten strategian omat kaupat vaikuttaisivat markkinahintoihin todellisessa tilanteessa(López de Prado 2018). Esimerkiksi pienivaihtoisten yhtiöiden osakkeissa, kuten First North -markkinapaikan yrityksissä yksittäinen suurempi toimeksianto voi merkittävästi liikuttaa hintaa, mikä johtaisi siihen, että backtestin oletus "kauppa toteutuu markkinahintaan" ei pidä todellisuudessa paikkaansa. Lisäksi historiallisen

datan perusteella saadut tulokset eivät takaa strategian menestystä tulevaisuudessa, sillä markkinaolosuhteet ja -dynamiikka muuttuvat jatkuvasti.

Kaupankäyntikustannusten, kuten komissioiden ja markkinaliukuman (*slippage*), tarkka mallintaminen Python-ympäristössä on haastavaa ja voi johtaa liian optimistisiin tuloksiin, ellei kustannusmallia määritellä realistisesti. (Chan 2013.)

2.4 Ylioptimoinnin riskit ja hallinta

2.4.1 Ylioptimoinnin tunnistaminen

Ylioptimointi on yksi algoritmisen kaupankäynnin keskeisimmistä riskeistä, sillä se voi johtaa harhaanjohtavan positiivisiin testituloksiin, jotka eivät toistu reaalimaailmassa (Bailey ym. 2014; López de Prado 2018). Ylioptimointi tarkoittaa tilannetta, jossa strategian parametrit on sovitettu liian tarkasti johonkin tiettyyn historiallisten tietojen jaksoon, eikä strategia näin ollen kykene reagoimaan muuttuviin markkinaolosuhteisiin tai uuteen dataan. Tämä voi aiheuttaa merkittäviä tappioita, jos strategia siirretään suoraan tuotantoon ilman riittävää validointia.

Ylioptimointi voi ilmetä monin eri tavoin. Yleisiä merkkejä ovat esimerkiksi epärealistisen korkeat tuotot testidatassa, strategian heikko toimivuus uusilla tai muuttuneilla datajaksoilla sekä liiallinen monimutkaisuus, jossa liian monta parametria tai sääntöä on viritetty juuri tiettyyn datan piirteeseen. Erityisen hälyttävää on, jos pienet muutokset parametrien arvoissa johtavat huomattaviin muutoksiin strategian suorituskyvyssä, mikä viittaa epävakauteen ja liialliseen hienosäätöön. (Bailey ym. 2014.)

Python-ympäristössä ylioptimoinnin tunnistaminen ja hallinta on teknisesti mahdollista useilla menetelmillä. Yksi keskeisimmistä on datan jakaminen in-sample- ja out-of-sample-osioihin. In-sample-datalla strategia optimoidaan ja sen jälkeen suorituskykyä arvioidaan out-of-sample-osiossa, jota ei ole käytetty parametrien säätöön. Tämän lähestymistavan avulla voidaan tarkastella strategian kykyä yleistää aiemmista havainnoista uusiin tilanteisiin. (Chan 2013.)

Toinen olennainen väline ylioptimoinnin tunnistamiseen on herkkyysanalyysi (*sensitivity analysis*). Herkkyysanalyysissä strategian keskeisiä parametreja muutetaan systemaattisesti ja analysoidaan, kuinka nämä muutokset vaikuttavat suorituskykyyn. Jos strategia tuottaa johdonmukaisia tuloksia laajalla parametrialueella, voidaan olettaa sen olevan vähemmän altis ylioptimoinnille. Lisäksi voidaan hyödyntää Monte Carlo -simulaatioita, joissa kaupankäyntitapahtumien järjestys satunnaistetaan ja mitataan strategian tuottojakauman stabiilisuutta. (López de Prado 2018.)

Ylioptimoinnin tunnistaminen ja hallinta eivät ole vain teknisiä tehtäviä, vaan myös metodologinen haaste, joka vaatii kurinalaista lähestymistapaa, systemaattista testausta ja kriittistä arviointia. Ilman tätä riski siitä, että strategia toimii vain historiassa mutta epäonnistuu todellisessa markkinassa, kasvaa merkittävästi.

2.4.2 Walk-forward-analyysi

Walk-forward-analyysi on yksi tehokkaimmista keinoista torjua ylioptimointia ja arvioida strategian todellista suorituskykyä (Bailey ym. 2014; López de Prado, 2018). Menetelmä perustuu ajatukseen, että strategian parametrit optimoidaan aiemmalla historiallisella datalla (in-sample), jonka jälkeen niiden toimivuutta testataan seuraavalla, erillisellä datalla (out-of-sample). Tämä prosessi toistetaan systemaattisesti käyttäen liukuvaa ikkunaa, jolloin koko aikajakso katetaan vaiheittain etenevällä optimointi- ja testausprosessilla.

Tällainen iteratiivinen lähestymistapa jäljittelee todellista kaupankäyntiympäristöä, jossa strategiaa päivitetään vain aiemman tiedon perusteella ja sen toimivuutta testataan tulevissa, tuntemattomissa markkinaolosuhteissa. Näin saadaan tarkempi ja realistisempi kuva strategian yleistettävyydestä ja robustisuudesta. (Bailey ym. 2014.)

Python-ohjelmointiympäristössä walk-forward-analyysi voidaan toteuttaa monin eri tavoin. Yksi yleisimmistä lähestymistavoista on rakentaa manuaalisesti niin kutsuttu sliding window -mekanismi, jossa optimointi- ja testijaksot liikkuvat eteenpäin askel askeleelta. Toinen vaihtoehto on hyödyntää valmiita työkaluja,

kuten scikit-learnin TimeSeriesSplit-luokkaa, joka mahdollistaa aikajärjestyksen säilyttävän ja toistettavan train/test-jakamisen. Lisäksi voidaan rakentaa mukautettuja funktioita tai kehikkoja, jotka optimoivat parametrit vain historiadatalla ja estävät datavuodon testijaksolle. (López de Prado 2018.)

Walk-forward-analyysin etuna on, että se antaa käyttäjälle mahdollisuuden tarkastella strategian suorituskykyä yksityiskohtaisesti eri ajanjaksoilla, erityisesti vaihtuvissa markkinaolosuhteissa. Samalla se paljastaa, kuinka hyvin strategia kestää ajallista siirtymää ja kuinka sen parametrit sopeutuvat muuttuviin tilanteisiin. Tämä tekee walk-forward-analyysistä erittäin hyödyllisen työkalun niin kaupallisen käyttöönoton kuin tutkimuksellisen validoinnin näkökulmasta. (Bailey ym. 2014; López de Prado, 2018.)

2.4.3 Robustisuustestit

Robustisuustestit, eli vankkuustestit ovat olennainen osa algoritmisten strategioiden validointia. Niiden tarkoituksena on mitata strategian kykyä säilyttää suorituskykynsä muuttuvissa olosuhteissa ja pienillä muutoksilla parametrien arvoissa. (Chan 2013.)

Parametrien herkkyysanalyysi perustuu siihen, että strategian keskeisiä parametreja muunnellaan askelittain ja jokaisella versiolla suoritetaan uusi simulaatio. Tarkoituksena on selvittää, kuinka vakaasti strategia toimii pienillä muutoksilla, eli onko suorituskyky riippuvainen tarkasta parametrivalinnasta vai säilyykö se kohtuullisena laajemmalla alueella. Herkkyysanalyysi auttaa paljastamaan, onko strategia liian hienosäädetty. (López de Prado & Zhu 2014; López de Prado, 2018.)

Markkinaskenaarioiden simuloinnissa strategiaa testataan useilla eri historiallisilla ajanjaksoilla, omaisuuslajeilla tai volatiliteettitasoilla. Näin voidaan arvioida, toimiiko strategia vain tietyssä markkinaympäristössä vai säilyttääkö se toimivuutensa myös olosuhteiden muuttuessa. Tämä tarjoaa arvokasta tietoa strategian yleistettävyydestä. (Chan 2013; López de Prado 2018.)

Slippage- ja spread-mallinnuksessa simuloidaan kaupankäyntikustannuksia, kuten slippagea (kaupan toteutuksen viive ja hintapoikkeama) sekä bid-ask-spreadiä (osto- ja myyntihinnan ero). Tarkoituksena on arvioida, miten kustannukset vaikuttavat strategian kannattavuuteen. Jos strategia toimii vain täydellisissä olosuhteissa ilman kitkatekijöitä, sen käytännön hyöty jää kyseenalaiseksi.

Tällaiset testit auttavat arvioimaan strategian resilienssiä, ja niiden tulokset voivat paljastaa, onko strategia todella käyttökelpoinen reaalimarkkinoilla. Vahvat vankkuustulokset vähentävät riskiä siitä, että strategia hajoaa ensi kosketuksessa uusiin markkinaolosuhteisiin. (López de Prado 2018.)

2.5 Markkinatehokkuus ja algoritminen kaupankäynti

2.5.1 Vaikutukset markkinoiden likviditeettiin

Algoritmisen kaupankäynnin yleistyminen on merkittävästi muuttanut rahoitusmarkkinoiden likviditeettirakenteita ja hinnanmuodostuksen mekanismeja. Suurten volyymien markkinoilla, kuten suurten yhtiöiden osakkeissa tai likvideissä futuurisopimuksissa algoritmit tarjoavat jatkuvaa ja nopeaa likviditeettiä. Ne reagoivat millisekunneissa tarjouksiin ja hintamuutoksiin, mikä pienentää tehokkaasti osto- ja myyntihintojen välistä eroa (bid-ask spread) ja nopeuttaa hintojen sopeutumista uuteen informaatioon. (Hendershott ym. 2011.)

Algoritmit, jotka toimivat markkinatakaajina, pyrkivät maksimoimaan voittonsa asettamalla osto- ja myyntitarjouksia samaan aikaan useilla hintatasoilla. Ne käyttävät hyväkseen tilapäisiä epätasapainotiloja kysynnän ja tarjonnan välillä sekä reagoivat kilpailijoiden liikkeisiin lähes reaaliaikaisesti (Treleaven ym. 2013). Tällainen toiminta ei ainoastaan lisää markkinoiden tehokkuutta vaan myös vakauttaa lyhyen aikavälin volatiliteettia.

Kuitenkin vaikutukset eivät ole täysin yksiselitteisiä. Pienemmissä ja vähemmän vaihdetuissa arvopapereissa algoritmien osuus kaupankäynnistä on pienempi, ja likviditeetti voi olla huomattavasti herkempi yksittäisten toimijoiden

vetäytymiselle. Tämä voi johtaa tilanteisiin, joissa markkinat ovat syviä vain näennäisesti, ja todelliset kaupankäyntimahdollisuudet voivat kadota nopeasti, esimerkiksi volatiliteetin äkillisesti kasvaessa (O'Hara 1995). Lisäksi on esitetty huolta siitä, että likviditeetti, jota algoritmit tarjoavat, voi olla vähemmän "luotettavaa" verrattuna perinteisiin toimijoihin, sillä se voi kadota hetkessä markkinahäiriön aikana.

Tutkimusten mukaan algoritmien likviditeettitarjonta on tehokkainta vakiintuneilla ja säännellyillä markkinoilla, mutta uusien tai epävakaiden instrumenttien kohdalla vaikutus voi olla heikompi tai jopa haitallinen. Tämän vuoksi on tärkeää erottaa eri markkinoiden rakenteet ja ymmärtää, millaisissa olosuhteissa algoritmien toiminta todella lisää markkinoiden tehokkuutta. (O'Hara 1995; Hendershott ym. 2011.)

2.5.2 Flash crash -ilmiöt

Flash crash -ilmiöt ovat algoritmisen kaupankäynnin aikakaudelle tyypillisiä äkillisiä ja jyrkkiä markkinahintojen romahduksia, jotka korjaantuvat yleensä nopeasti. Ne heijastavat algoritmisten järjestelmien välistä vuorovaikutusta ja markkinarakenteen haavoittuvuutta, erityisesti korkeataajuisessa kaupankäynnissä. Näiden tapahtumien seuraukset voivat olla merkittäviä, sillä ne voivat horjuttaa sijoittajien luottamusta markkinoiden vakauteen. (Kirilenko, Kyle, Samadi & Tuzun 2017.)

Flash crash -tilanteet johtuvat usein algoritmien synkronoituneesta toiminnasta: yhden algoritmin käynnistämä myynti aiheuttaa reaktioita muissa algoritmeissa, mikä johtaa spiraalimaisesti kiihtyvään hinnanlaskuun. Tämä ketjureaktio voi syntyä sekunneissa ja aiheuttaa huomattavaa hintavolatiliteettia. Esimerkiksi vuoden 2010 Yhdysvaltain flash crashissa Dow Jones -indeksi laski lähes tuhat pistettä muutamassa minuutissa, ennen kuin palautui lähes yhtä nopeasti. (Kirilenko ym. 2017.)

Näiden ilmiöiden ehkäisemiseksi tarvitaan kehittyneitä algoritmeja, jotka osaavat tunnistaa markkinadatan poikkeavuudet ja reagoida viiveellä tai säädellysti äärimmäisiin hintaliikkeisiin. Lisäksi tarvitaan markkinarakenteeseen

sisäänrakennettuja suojamekanismeja, kuten kaupankäynnin automaattisia keskeytyksiä. (López de Prado 2018.)

Python-ympäristö tarjoaa joustavan alustan flash crash -ilmiöiden tutkimiseen. Tällaisia tilanteita voidaan simuloida mallintamalla algoritmien käyttäytymistä historiallisissa korkean volatiliteetin jaksoissa – esimerkiksi käyttämällä satunnaistettua agenttipohjaista simulaatiota tai toistamalla tapahtumaketjuja, jotka ovat aiemmin johtaneet markkinahäiriöihin. Nämä simulaatiot mahdollistavat sekä algoritmien reagointilogiikan että markkinarakenteen vaikutusten analysoinnin realistisissa olosuhteissa.

2.5.3 Sääntelynäkökulmat

Algoritmisen kaupankäynnin lisääntyminen on tuonut mukanaan merkittäviä sääntelyhaasteita ja -vaatimuksia. Sääntelykehykset ovat kehittyneet vastaamaan uudenlaisten markkinatoimijoiden toimintaa, jossa nopeus, automaatio ja massiivinen datankäsittely ovat keskiössä. Viranomaisten keskeinen tavoite on varmistaa markkinoiden vakaus, tehokkuus ja sijoittajansuoja nopeasti muuttuvassa algoritmisessa ympäristössä. (European Securities and Markets Authority 2020.)

Yksi sääntelyn kulmakivistä on algoritmien ennakkotestaus ennen käyttöönottoa. Algoritmeille asetetaan vaatimuksia dokumentoidusta testauksesta, jossa tulee osoittaa järjestelmän toimivan myös poikkeusolosuhteissa. Tämä koskee erityisesti korkean taajuuden kaupankäyntiä (HFT), jossa virheellisesti toimiva algoritmi voi vaikuttaa merkittävästi markkinoiden toimintaan. (European Securities and Markets Authority 2020.)

Sääntely edellyttää myös tehokkaita riskienhallintamekanismeja. Tällaisia ovat esimerkiksi limiittijärjestelmät, jotka estävät yksittäisten algoritmien aiheuttamia liiallisia liikkeitä, sekä järjestelmien valvontatyökalut, joilla voidaan havaita virheelliset toiminnot ajoissa. (Kirilenko ym. 2017.)

Lisäksi sääntelyssä korostetaan läpinäkyvyyttä ja vastuun kohdentamista. Algoritmisilla toimijoilla tulee olla valmiudet jäljittää ja dokumentoida algoritmiensa toimintaa sekä vastata sen aiheuttamista markkinavaikutuksista. Tämä tarkoittaa käytännössä esimerkiksi lokien säilyttämistä ja dokumentoidun kehitysprosessin esittämistä viranomaisille tarvittaessa. (European Securities and Markets Authority 2020.)

Viranomaiset ovat ottaneet käyttöön myös rakenteellisia suojaelementtejä, joilla pyritään ehkäisemään algoritmien aiheuttamia markkinahäiriöitä. Näitä ovat esimerkiksi kaupankäynnin keskeytysmekanismit (*circuit breakers*), jotka aktivoituvat tietyn prosentuaalisen hinnanmuutoksen jälkeen. Ne mahdollistavat tilanteen rauhoittamisen ja estävät hintojen syöksykierteitä, joita on havaittu muun muassa vuoden 2010 flash crash -tapauksessa. (Kirilenko ym. 2017; European Securities and Markets Authority 2020.)

2.6 Algoritmisen kaupankäynnin haasteet ja kriittinen tarkastelu

2.6.1 Algoritmisen kaupankäynnin varjopuolet

Algoritminen kaupankäynti liitetään usein tehokkuuteen, tarkkuuteen ja mahdollisuuteen hyödyntää suuria datamääriä nopeasti. Näiden etujen rinnalla on kuitenkin huomattavia varjopuolia, jotka tekevät toiminnasta monille sijoittajille riskialtista. Ensinnäkin kaupankäynnin luonne on usein nollasummapeli: jokainen voitettu euro on toiselta markkinaosapuolelta hävitty euro. Tämä eroaa perinteisestä sijoittamisesta, jossa pitkän aikavälin arvonmuodostus perustuu yritysten reaalitaloudelliseen toimintaan ja tuoton luomiseen. Useiden tutkimusten mukaan valtaosa algoritmista kaupankäyntiä harrastavista yksityissijoittajista jää tappiolle pitkällä aikavälillä, sillä heillä ei ole käytössään samoja resursseja, tietoliikenneyhteyksiä ja infrastruktuuria kuin institutionaalisilla toimijoilla. (Chan 2013; López de Prado 2018.)

Toinen merkittävä varjopuoli liittyy markkinoiden vakauteen. Algoritmit voivat aiheuttaa äkillisiä ja hallitsemattomia hintaliikkeitä, jotka eivät perustu fundamentteihin, vaan algoritmien väliseen vuorovaikutukseen. Tunnetuin

esimerkki tästä on vuoden 2010 Yhdysvaltain flash crash, jossa Dow Jones -indeksi romahti lähes tuhat pistettä muutamassa minuutissa ja palautui sen jälkeen yhtä nopeasti (Kirilenko ym. 2017). Tällaiset tapahtumat horjuttavat sijoittajien luottamusta markkinoiden toimintaan ja osoittavat, että algoritmien yleistyminen ei aina lisää tehokkuutta, vaan voi myös kasvattaa systeemistä riskiä.

Lisäksi algoritmien yleistyminen voi johtaa siihen, että markkinoiden dynamiikka muuttuu entistä vaikeammaksi ennustaa. Kun suuri osa kaupankäynnistä perustuu toisiinsa reagoiviin ohjelmistoihin, voi syntyä tilanteita, joissa perinteinen analyysi menettää merkitystään. Tämä voi heikentää yksityissijoittajien mahdollisuuksia menestyä, sillä markkinat eivät enää käyttäydy rationaalisesti tai selkeästi fundamenttien ohjaamina. (O'Hara, 1995; Hasbrouck, 2007.)

2.6.2 Sijoittamisen ja uhkapelin välimaasto

Algoritmisen kaupankäynnin luonnetta voidaan tarkastella sijoittamisen ja uhkapelaamisen välimaastossa. Perinteinen sijoittaminen perustuu yritysten tuottamaan kassavirtaan, osinkoihin ja pitkän aikavälin arvonmuodostukseen. Algoritmisessa kaupankäynnissä horisontti on usein hyvin lyhyt, ja strategiat pohjautuvat markkinahintojen tilapäisiin liikkeisiin. Tällöin menestys riippuu enemmän markkinapsykologiasta, hinnoittelun epätasapainoista ja toisten sijoittajien käyttäytymisestä kuin todellisesta arvonmuodostuksesta. (Murphy 1999; Chan 2013.)

Tämä näkökulma korostuu erityisesti kryptovaluuttamarkkinoilla, joissa omaisuuserien arvo ei perustu reaalitalouden tuottoon vaan pitkälti spekulaatioon. Kryptovaluuttojen arvo määräytyy kysynnän ja tarjonnan vaihtelun sekä sijoittajien riskinottohalukkuuden perusteella. Tätä voidaan verrata klassiseen esimerkkiin Hollannin 1600-luvun tulppaanimaniasta, jossa hintojen nousu perustui pelkästään siihen, että löytyi aina uusi ostaja, joka oli valmis maksamaan enemmän. Hinnat romahtivat kun ostajia ei enää ollut (Kindleberger 2000). Tällaisessa ympäristössä algoritmisen kaupankäynnin

voidaan nähdä muistuttavan uhkapeliä: se ei luo uutta arvoa, vaan jakaa olemassa olevaa varallisuutta uudelleen markkinaosapuolten kesken.

On kuitenkin tärkeää huomioida, että osa algoritmisista strategioista, kuten arbitraasi ja market making, voi lisätä markkinoiden tehokkuutta ja likviditeettiä. Näissä tapauksissa kyse ei ole pelkästään voiton tavoittelusta toisten kustannuksella, vaan markkinoiden toiminnan parantamisesta. Silti valtaosa yksityissijoittajien käyttämistä lyhyen aikavälin teknisistä strategioista sijoittuu harmaalle alueelle sijoittamisen ja uhkapelin välillä.

2.6.3 Yksityissijoittajan mahdollisuudet suuria toimijoita vastaan

Avoimen lähdekoodin työkalut, kuten Backtrader, Zipline ja QuantConnect, sekä helposti saatavilla olevat markkinadatapalvelut, kuten yfinance ja ccxt, ovat madaltaneet kynnystä algoritmisen kaupankäynnin aloittamiseen. Tämä kehitys on tehnyt mahdolliseksi sen, että myös yksityissijoittajat voivat toteuttaa strategioita, jotka aiemmin olivat vain suurten pankkien ja hedge-rahastojen ulottuvilla. (McKinney 2017; López de Prado 2018.)

Käytännössä yksityissijoittajan mahdollisuudet menestyä suuria institutionaalisia toimijoita vastaan ovat kuitenkin rajalliset. Suuret pankkiiriliikkeet investoivat miljardeja euroja algoritmien kehittämiseen, tietoliikenneyhteyksiin ja huippunopeaan infrastruktuuriin. Esimerkiksi valokaapeliyhteydet pörsseihin ja mikroaaltolinkit, jotka välittävät signaaleja hieman valokuitua nopeammin, tarjoavat suurille toimijoille ratkaisevaa etumatkaa erityisesti korkeataajuuskaupankäynnissä (Aldridge 2013). Yksityissijoittajilla ei ole realistisia mahdollisuuksia kilpailla tällaisilla alueilla, koska etumatka perustuu suoraan teknologisiin resursseihin ja nopeuteen.

Sen sijaan yksityissijoittajilla on paremmat mahdollisuudet hyötyä strategioista, jotka perustuvat tekniseen analyysiin ja pidempiin aikahorisonteihin. Mean reversion -strategiat, jotka hyödyntävät esimerkiksi RSI- ja Bollinger Bands -indikaattoreita, eivät vaadi mikrosekuntitason reagointia. Näissä menetelmissä yksityissijoittaja voi kilpailla paremmin, koska onnistuminen riippuu enemmän luovuudesta, parametrivalinnoista ja strategian kurinalaisesta testaamisesta

kuin kalliin teknologian omistamisesta. Tämä ei silti tarkoita, että yksityissijoittaja olisi keskimäärin voitolla: tutkimusten mukaan suurin osa aktiivisista kaupankävijöistä jää edelleen tappiolle, vaikka käytössä olisi algoritmeja. (Barber & Odean 2013.)

3 ALGORITMISEN KAUPANKÄYNTISTRATEGIAN KEHITYS

3.1 Datan valinta ja esikäsittely

Tässä tutkimuksessa tarkasteltavana kohdeinstrumenttina toimi Bitcoin/USDT-valuuttapari. Pari valittiin analyysin kohteeksi sen korkean volatiliteetin ja likviditeetin vuoksi, mikä tarjoaa erityisen hedelmällisen alustan algoritmisten kaupankäyntistrategioiden testaamiseen. Historiallista hintadataa ladattiin viidellä eri aikavälin resoluutiolla: minuutti (1m), tunti (1h), päivä (1d), viikko (1w) ja kuukausi (1M). Näiden resoluutioiden avulla pyrittiin selvittämään, millä aikajänteellä strategian suorituskyky on optimaalisin. Datan ajallinen kattavuus ulottui vuodesta 2020 vuoden 2024 loppuun saakka, mikä mahdollisti useiden erilaisten markkinaympäristöjen analysoinnin saman datasetin puitteissa. (Liite 1.)

Data ladattiin CSV-muodossa, ja sitä ennen suoritettiin huolellinen esikäsittely, jolla pyrittiin varmistamaan analyysin luotettavuus ja eheys. Ensimmäisessä vaiheessa tarkasteltiin aikaleimojen oikeellisuus, ja ne muunnettiin datetime-muotoon, mikä mahdollisti ajallisesti järjestetyn analyysin. Tämän jälkeen poistettiin kaikki rivit, joissa oli puuttuvia arvoja tai nollia keskeisissä kentissä, kuten sulkuhinta, volyymi tai korkein ja alin hinta. Lopuksi varmistettiin, että kaikki sarakkeet sisälsivät oikeantyyppistä dataa: numeerisiksi tarkoitetut muuttujat muunnettiin `float`-muotoon ja aikakentät ajalliseksi indeksiksi.

Datan resoluution valinta perustui empiiriseen vertailuun eri aikavälien suorituskyvystä. Kullekin resoluutiolle ajettiin sama yksinkertainen kaupankäyntistrategia, jonka jälkeen verrattiin saavutettua tuottoa ja toteutuneiden kauppojen määrää. Näin saatiin vertailukelpoiset mittarit eri datatiheyksien soveltuvuudesta algoritmiseen kaupankäyntiin. Tulokset osoittivat selvästi, että yhden minuutin ja yhden tunnin datassa strategia kävi kauppaa liian tiheästi, mikä johti heikkoon kokonaistuottoon. Viikko- ja kuukausidatan tapauksessa signaaleja ei muodostunut riittävästi, jolloin strategian potentiaali jäi kokonaan hyödyntämättä.

Päivätason (1d) data osoittautui näiden kokeilujen perusteella parhaaksi kompromissiksi. Se tarjosi riittävän määrän kauppoja, mutta ei ylisuorittanut siten, että kulut tai slippage olisivat merkittävästi heikentäneet tuloksia. Tämän vuoksi päätettiin käyttää jatkokehityksessä yksinomaan 1 päivän resoluution dataa. Vertailun tueksi esitetään seuraava Taulukko 1, joka havainnollistaa eri resoluutioiden vaikutusta sekä pääoman kehitykseen että kauppojen määrään. Päätös tehtiin 1 päivän resoluution hyödyksi, koska se tarjosi parhaan yhdistelmän kaupankäyntitehokkuutta ja hallittua riskitasoa.

Taulukko 1. Resoluutioiden vertailu (Loppupääoma ja kauppojen määrä)

Resoluutio	Loppupääoma	Kauppoja
1m	15 898,20	40293
1h	76 402,17	270
1d	130 805,08	11
1w	102 067,10	1
1M	100 000,00	0

3.2 Alkuperäinen strategia

Strategian taustalla oli kahden teknisen analyysin työkalun yhdistelmä: suhteellisen voimaindeksin (RSI) ja Bollingerin nauhojen. Nämä indikaattorit toimivat yhdessä siten, että strategia pyrki tunnistamaan tilanteet, joissa markkina oli ylireagoinut lyhyellä aikavälillä, ja odotettavissa oli palautuminen kohti keskiarvoa. Kaupankäyntisignaali muodostui, kun sulkuhinta laski Bollingerin alempien nauhojen alapuolelle ja RSI oli matalalla tasolla, mikä viittasi tekniseen ylimyyntiin. Tällöin syntyi osto-operaation edellytys. Myyntisignaali puolestaan syntyi, kun hinta nousi Bollingerin ylärajan yläpuolelle tai RSI kohosi korkealle, merkkinä mahdollisesta yliostetusta tilanteesta. (Liite 2.)

Strategian alkuperäiset parametrit perustuivat yleisesti hyväksyttyihin arvoihin. RSI-indikaattorin pituudeksi valittiin 14 päivää, joka on yksi käytetyimmistä arvoista teknisessä analyysissä. Bollingerin nauhojen pituudeksi asetettiin 20 päivää, joka vastaa noin yhtä kaupankäyntikuukautta. RSI\:n alarajaksi valittiin 30 ja ylärajaksi 70, joiden sisällä markkinan katsotaan olevan neutraali ja joiden ulkopuolella ylilyönnit ovat mahdollisia.

Tällä lähtökohtaisella strategialla saavutettiin testijaksoilla maltillisia positiivisia tuottoja. Se antoi selkeitä signaaleja ja pysyi poissa markkinoilta, kun selkeitä yli- tai alilyöntejä ei esiintynyt. Kuitenkin jo varhaisissa testeissä havaittiin, että strategia oli varsin herkkä parametrien muutoksille. Esimerkiksi pienetkin muutokset RSI\:n pituudessa tai raja-arvoissa johtivat merkittäviin muutoksiin kaupankäynnin määrässä ja tuoton volatiliteetissa. Lisäksi markkinasuunta vaikutti selvästi strategian toimivuuteen: nousevassa trendissä strategia saattoi jäädä kokonaan pois kaupankäynnistä, kun taas laskevassa tai sivuttaisessa markkinassa se aktivoitui useammin. Nämä havainnot toivat esiin tarpeen jatkotestaukselle ja strategian hienosäädölle, jotta se voisi toimia mahdollisimman johdonmukaisesti eri markkinaympäristöissä.

3.3 Parametrien optimointi

Optimointivaiheessa käytettiin systemaattista lähestymistapaa, jonka tarkoituksena oli löytää tehokkaimmat mahdolliset yhdistelmät strategian muuttujille. Käytössä oli Pythonin itertools-kirjaston product-menetelmä, joka mahdollistaa kaikkien mahdollisten parametrikombinaatioiden läpikäynnin ennalta määritellyistä arvojoukoista. Optimointi kohdistui erityisesti RSI:n pituuteen, Bollingerin nauhojen pituuteen sekä RSI:n ala- ja ylärajoihin, joiden vaikutusta strategian kannattavuuteen haluttiin tutkia tarkemmin.

Parametrien testivälit valittiin aiempien tutkimusten ja käytännön kokemuksen perusteella. Esimerkiksi RSI:n pituuksia testattiin välillä 10–20, Bollingerin nauhoja 15–25 päivän jaksoilla, ja alarajaksi harkittiin arvoja 25–35 sekä ylärajaksi 65–75. Näiden yhdistelmien avulla muodostettiin laaja matriisi, jossa oli kymmeniä mahdollisia strategiavariaatioita. Jokainen yhdistelmä testattiin

erikseen käyttäen samaa datasarjaa ja kaupankäyntilogiikkaa, jolloin tuloksia voitiin vertailla luotettavasti. (Liite 3.)

Tämän lähestymistavan avulla saatiin esiin parametriyhdistelmät, jotka tuottivat korkeimman loppupääoman annetulla historiallisella aineistolla. Samalla nähtiin, miten herkästi strategian suoritus muuttui yksittäisten muuttujien arvojen muuttuessa. Laajan testiaineiston ansiosta oli mahdollista tehdä perusteltu valinta parhaista parametreista jatkotestejä ja käyttöönottovaihetta varten. Optimoinnin tuloksena havaittiin, että useat parametriyhdistelmät tuottivat samankaltaisia tuloksia, mutta Taulukon 2 perusteella yhdistelmä RSI=14, BB=25, lower=30 ja upper=70 erottui selvästi parhaaksi useilla mittareilla mitattuna.

Taulukko 2. Top 5 optimointitulosta

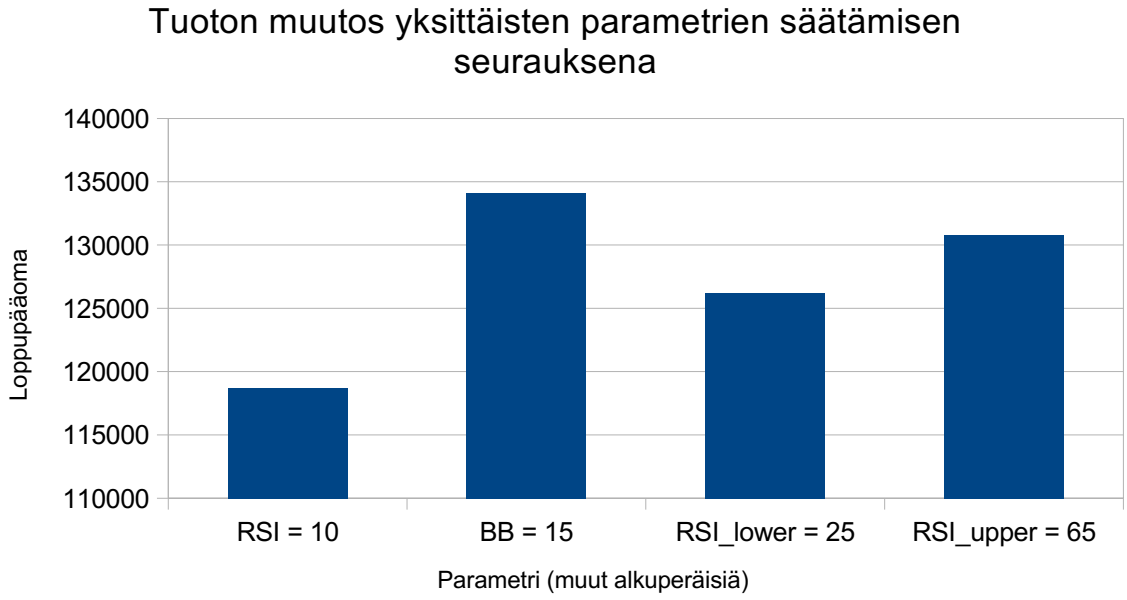
RSI	BB	RSI_lower	RSI_upper	Final_Value
10	15	25	65	196 156,66
18	15	25	70	196 156,66
18	15	35	75	196 156,66
18	15	35	70	196 156,66
18	15	35	65	196 156,66

3.4 Herkkyysanalyysi

Strategian herkkyysanalyysivaiheessa keskityttiin siihen, kuinka yksittäisten parametrien muuttaminen vaikuttaa strategian kokonaissuorituskykyyn. Tarkoituksena oli pitää kaikki muut parametrit vakioina ja muuttaa vain yhtä muuttujaa kerrallaan, jotta saataisiin selville, miten herkästi strategia reagoi yksittäisiin säätöihin. Tällainen lähestymistapa auttaa ymmärtämään, mitkä parametrit ovat kaikkein kriittisimpiä ja mitkä sietävät enemmän vaihtelua ilman merkittävää vaikutusta lopputulokseen.

Käytännön toteutuksessa analysoitiin RSI-indikaattorin pituuden vaikutusta vaihteluvälillä 10–20, Bollingerin nauhojen pituuden vaikutusta välillä 15–25 sekä RSI:n raja-arvojen vaikutusta alarajan liikkuessa välillä 25–35 ja ylärajan välillä 65–75. Jokaisessa tapauksessa tarkasteltiin, miten nämä yksittäiset muutokset vaikuttivat sekä strategian tuottamaan loppupääomaan että toteutuneiden kauppojen määrään.

Kuvio 1 perusteella havaittiin, että tietyt parametrit vaikuttivat selvästi enemmän kuin toiset. Esimerkiksi RSI:n pituuden lyhentäminen johti selkeään muutokseen kaupankäynnin aktiivisuudessa, mikä vaikutti suoraan tuottoihin. Vastaavasti Bollingerin nauhojen lyhentäminen toi enemmän signaaleja, mutta samalla myös nosti riskitasoa. Herkkyysanalyysi tarjosi arvokasta tietoa siitä, missä rajoissa parametreja voidaan säätää ilman, että strategia menettää tehokkuuttaan. Tämä analyysi muodosti tärkeän pohjan myöhemmille robustisuustesteille ja strategiapäivityksille. Herkkyysanalyysin perusteella strategia osoitti kohtalaista robustisuutta: yksittäiset parametrit eivät romahduttaneet suorituskykyä, mutta optimaalinen yhdistelmä oli selkeä.



Kuvio 1. Tuoton muutos suhteessa yksittäisiin parametreihin

3.5 Markkinaskenaarioiden simulointi

Strategian toimivuutta arvioitiin myös erilaisten markkinaskenaarioiden yhteydessä, jotta saataisiin selville, kuinka hyvin strategia mukautuu vaihteleviin markkinaolosuhteisiin. Tätä varten valittiin kolme tyypillistä markkinaympäristöä: nouseva trendi, laskeva trendi ja sivuttaisliikkeeseen painottuva markkina. Kunkin skenaarion dataosuudet valittiin käsin, jotta ne edustaisivat mahdollisimman selkeästi kyseistä markkinatyyppiä ilman ristikkäisiä trendejä tai epäselviä suuntia.

Nousevan markkinan testijakso sisälsi pitkän ja suhteellisen tasaisen hinnannousun, jossa mahdolliset korjausliikkeet olivat pieniä ja lyhytaikaisia. Laskeva markkina puolestaan muodostui jaksosta, jossa hinnat laskivat tasaisesti pidemmän ajan kuluessa, mikä loi otollisen ympäristön mean reversion -strategialle löytää ylimyytyjä tilanteita. Sivuttaisliikettä edustava jakso sisälsi vähäisiä hintavaihteluita, ilman selkeää suuntaa ylös- tai alaspäin. Tällaisessa ympäristössä mean reversion -strategioiden voidaan olettaa toimivan erityisen hyvin, koska hinnat taipuvat palaamaan kohti keskiarvoaan ilman trendien aiheuttamaa vinoumaa.

Kullekin markkinatyypille ajettiin sama strategia samoilla parametreilla ja kaupankäyntilogiikalla, minkä ansiosta tulokset olivat vertailukelpoisia. Tulosten perusteella havaittiin selkeä ero strategian tehokkuudessa eri markkinaympäristöissä. Parhaiten strategia toimi sivuttaisessa markkinassa, jossa loppupääoma kasvoi merkittävästi ja signaaleja syntyi tehokkaasti. Laskevassa markkinassa strategia menestyi kohtuullisesti, mutta nousevassa trendissä tuotto jäi vaatimattomaksi, ja kauppoja syntyi vähemmän.

Tämä testi osoitti, että strategian suorituskyky on vahvasti sidoksissa markkinarakenteeseen. Mean reversion -lähestymistavan luonteeseen kuuluu, että se toimii parhaiten tilanteissa, joissa ei ole selkeää suuntaa vaan toistuvaa liikehdintää tietyssä hintahaarukassa. Nämä havainnot tukevat jatkokehityksen tarvetta suodattaa strategian toimintaa eri markkinatilanteiden mukaan, esimerkiksi käyttämällä trendisuodattimia tai volatiliteetti-indikaattoreita. Taulukko 3 huomataan strategian toimivan parhaiten sivuttaismarkkinassa, joka

tukee mean reversion -logiikkaa. Nousevassa markkinassa suorituskyky oli selkeästi heikompi.

Taulukko 3. Markkinaskenaarioiden vaikutus

Skenaario	Loppupääoma	Kauppoja
Nouseva	109 213,30	6
Laskeva	117 110,45	7
Sivuttaisliike	126 45,99	5

3.6 Slippage- ja kulumallinnus

Strategian realistisuuden arvioimiseksi suoritettiin useita kaupankäyntikustannuksia jäljitteleviä simulaatioita. Näissä skenaarioissa pyrittiin jäljentämään mahdollisimman todenmukaisesti markkinoilla esiintyviä transaktiokustannuksia, kuten slippagea ja kaupankäyntikuluja, joita algoritmistrategia saattaa kohdata reaaliaikaisessa kaupankäynnissä.

Ensimmäisessä skenaariossa käytettiin realistisia oletuksia, joissa slippageksi asetettiin 0,05 prosenttia ja kaupankäyntikuluiksi 0,1 prosenttia transaktiota kohden. Tämä vastaa tyypillistä kustannustasoa monilla kryptovaluuttapörsseillä, joissa kaupankäyntivolyymit ovat korkeita ja spreadit matalia.

Toisessa, niin sanotussa korkeiden kulujen skenaariossa nostettiin kustannuksia huomattavasti. Tässä mallissa slippageksi määriteltiin 0,2 prosenttia ja kaupankäyntikuluiksi 0,25 prosenttia. Tämän tason kustannukset simuloivat tilannetta, jossa likviditeetti on rajallista, ja toimeksiantoja ei voida toteuttaa tarkalleen toivotuilla hinnoilla.

Kolmannessa, niin kutsutussa stressiskenaariossa arvioitiin strategian suorituskyky tilanteessa, jossa sekä slippage että kulut ovat poikkeuksellisen korkeat: molemmat 0,5 prosenttia. Tämä tilanne voi syntyä esimerkiksi erittäin

volatiilissa markkinatilanteessa tai huonosti likvidissä kohdeinstrumentissa, jossa jokainen toimeksianto aiheuttaa merkittäviä kustannuksia.

Näiden kolmen kustannustason avulla saatiin muodostettua käsitys strategian robustisuudesta suhteessa kaupankäynnin teknisiin esteisiin ja todellisiin kustannustekijöihin. Taulukko 4 osoittaa, että vaikka korkeammat kulut heikensivät tuottoja jonkin verran, strategian ydinlogiikka säilytti toimivuutensa kaikissa kolmessa skenaariossa. Strategia kesti hyvin kaupankäyntikulut, mikä osoittaa, ettei se ole riippuvainen sadasosan tarkkuudella tehdystä ajoituksesta.

Taulukko 4. Kulujen vaikutus tuottoon

Kustannustaso	Loppupääoma
Realistinen	130 805,08
Korkea	128299,62
Stressi	123683,14

3.7 Walk-forward-analyysi

Walk-forward-analyysissä koko historiallinen data jaettiin toistuviin koulutus- ja testijaksoihin, joiden tavoitteena oli simuloida strategian käyttöä oikeassa ajassa etenevässä ympäristössä. Tämä analyysimenetelmä perustuu siihen, että strategia optimoidaan ensin tietyn jakson niin sanotun koulutusjakson datalla, minkä jälkeen sen suorituskykyä mitataan erillisellä, tulevaisuutta simuloivalla testijaksolla. (Liite 4.)

Tässä tutkimuksessa data pilkottiin kolmeen koulutusjaksoon, joiden kesto oli 1,5 vuotta kukin. Näitä seurasivat aina kuuden kuukauden pituiset testijaksot, jotka vastasivat reaalimaailman kaupankäyntiä ilman tietoa tulevasta. Koulutusjaksot valittiin siten, että ne sisälsivät monipuolisesti erilaisia markkinaolosuhteita: nousuja, laskuja ja sivuttaisliikkeitä. Tämä mahdollisti sen, että strategiaa ei ylisovitettu yksittäiseen markkinatrendiin.

Walk-forward-testauksen avulla voitiin arvioida, kuinka hyvin strategia yleistyy uusiin, sille tuntemattomiin tilanteisiin. Samojen parametrien käyttäminen eri testijaksoilla mahdollisti myös arvioinnin siitä, ovatko valitut arvot aidosti robustit vai pelkästään kyseiseen dataan sopivia. Taulukko 5 osoittaa, että valitut parametrit (RSI=12, BB=18, lower=25, upper=65) säilyttivät tehokkuutensa läpi kaikkien testijaksojen, mikä antaa vahvan viitteen strategian yleistettävyydestä ja reaalimaailman käyttökelpoisuudesta. Testi osoitti, että strategia ei ollut ylisovitettu, ja samat parametrit toimivat useilla jaksoilla.

Taulukko 5. Markkinaskenaarioiden vaikutus

Jakso	Testin tuotto	Parametrit
2021-07–2021-12	100 000,00	12,18,25,65
2023-01–2023-06	110 266,38	12,18,25,65
2024-01–2024-12	140 199,15	12,18,25,65

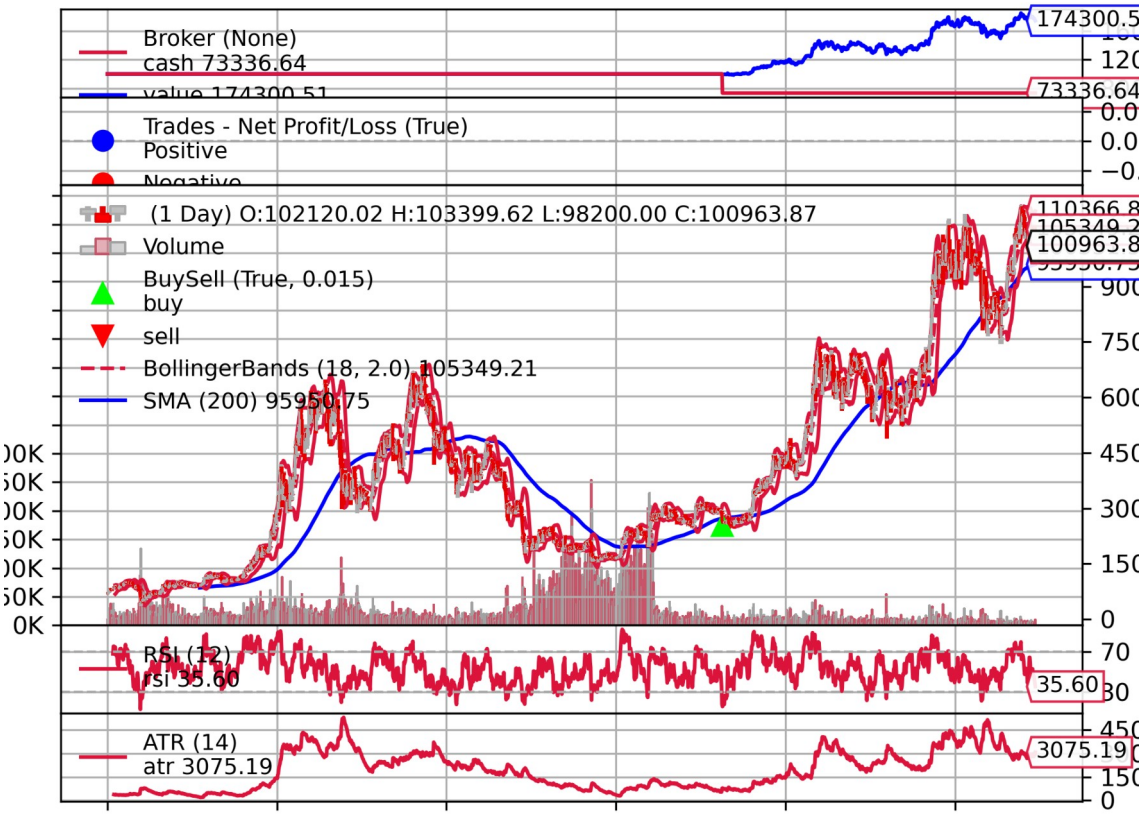
3.8 Lopullinen strategiapäivitys

Walk-forward-analyysin ja robustisuustestien perusteella strategiaa päätettiin päivittää merkittävästi. Ensinnäkin, parametriarvoihin tehtiin hienosäätöä. RSI-indikaattorin pituudeksi valittiin 12 päivää ja Bollingerin nauhojen pituudeksi 18 päivää. Näiden lisäksi RSI:n alarajaksi asetettiin 25 ja ylärajaksi 65. Tämä parametriyhdistelmä osoitti walk-forward-testeissä johdonmukaista suorituskykyä eri ajanjaksoilla ja markkinaympäristöissä. (Liite 5.)

Toiseksi, strategiaan lisättiin yksinkertainen mutta tehokas trendisuodatin 200 päivän liukuvan keskiarvon (SMA200) muodossa. Tämän suodattimen tarkoituksena oli parantaa strategian kontekstisidonnaisuutta: kauppoja sallittiin ainoastaan tilanteissa, joissa hinta oli kyseisen keskiarvon alapuolella. Tämä rajasi strategian toimintaa laskeviin trendeihin, joissa mean reversion -logiikka toimii tyypillisesti paremmin.

Kolmantena komponenttina lisättiin volatiliteettisuodatin, joka perustui ATR-indikaattorin (Average True Range) arvoon. Mikäli ATR ylitti viisi prosenttia hinnasta, kaupankäynti estettiin tänä ajankohtana. Tämä lisäys suojasi strategiaa liialliselta markkinamelulta ja esti kaupankäyntiä tilanteissa, joissa hintaliikkeet olivat poikkeuksellisen suuria ja mahdollisesti epäluotettavia.

Tällä lopullisella versiolla strategia osoitti äärimmäistä selektiivisyyttä: koko testijaksolla toteutui vain yksi kauppa. Kuitenkin tämä kauppa oli erittäin onnistunut ja nosti pääoman merkittävästi. Loppupääoma nousi 174 300,51 euron tasolle, Kuvio 2 osoittaa, että strategian signaalien harvinaisuudesta huolimatta valitut ehdot mahdollistavat erittäin tehokkaan kaupankäynnin tietyissä olosuhteissa. Yksi valittu signaali tuotti huomattavan tuoton. Strategia on äärimäisen selektiivinen, mutta potentiaalisesti äärimäisen tehokas.



Kuvio 2. Mean Reversion -strategian optimoitu tulos

4 POHDINTA

Tämä tutkimus osoitti, että algoritmisen kaupankäynnin strategioiden kehitys, testaus ja arviointi voidaan toteuttaa systemaattisesti Python-ympäristössä hyödyntäen avointa lähdekoodia ja kvantitatiivisia analyysimenetelmiä. Backtraderin kaltaiset kirjastot mahdollistavat monipuolisen strategioiden rakentamisen, simuloimisen ja analysoinnin, mikä tekee niistä erittäin hyödyllisiä sekä akateemisiin että käytännön sovelluksiin. Valittu lähestymistapa mahdollisti tehokkaan ja toistettavan tutkimusprosessin, jossa voitiin hyödyntää suuria datamääriä, säilyttää analyysin läpinäkyvyys sekä dokumentoida vaiheet yksityiskohtaisesti.

Tutkimuksen kohteena ollut Mean Reversion -strategia perustui kahteen tunnettuun tekniseen indikaattoriin: Bollinger Band -indikaattoriin ja Relative Strength Indexiin (RSI). Näiden yhdistelmä tarjosi selkeitä sisään- ja ulostulokohtia erityisesti silloin, kun markkinat liikkuivat vaakasuunnassa. Strategia osoitti parhaimman suorituskykynsä sivuttaisliikkeeseen painottuvissa markkinoissa, joissa hinta heiluu tietyn tason sisällä ilman voimakasta trendiä. Tämä on loogista, sillä mean reversion -lähestymistapa perustuu oletukseen, että hinta palautuu keskiarvonsa lähelle. Tulokset korostivat myös analyysin tarkkuuden merkitystä: liian hienojakoinen data, kuten minuutti- tai tuntitasoinen, johti ylikaupankäyntiin ja heikompaan tuottoon. Toisaalta liian harva data, kuten viikko- tai kuukausitasoinen, ei tarjonnut riittävästi signaaleja. Päivätason data osoittautui optimaalisen tasapainoiseksi, mikä mahdollisti hallitun kauppatahdin ja riittävän signaalien määrän.

Tutkimuksessa toteutettu parametrien optimointi, herkkyysanalyysi ja walk-forward-analyysi mahdollistivat strategian robustisuuden arvioinnin monesta eri näkökulmasta. Näiden analyysien avulla saatiin kattava kuva siitä, kuinka strategia käyttäytyy erilaisissa markkinaolosuhteissa sekä kuinka vakaa se on parametrien muutoksille. Herkkyysanalyysi osoitti, että pieniä muutoksia parametrien arvoihin voitiin tehdä ilman merkittävää heikkenemistä tuloksissa, mikä viittaa siihen, ettei strategia ollut liialti hienosäädetty. Walk-forward-analyysi puolestaan antoi realistisen kuvan strategian suorituskyvystä uusilla,

ennennäkemättömillä markkinajaksoilla, mikä on tärkeää todellisen kaupankäynnin näkökulmasta.

Lisäksi toteutetut slippage- ja spread-simuloinnit toivat esiin kaupankäyntikustannusten vaikutuksen strategian kannattavuuteen. Vaikka korkeat kulut laskivat absoluuttista tuottoa, strategian looginen rakenne säilytti arvonsa. Tämän ansiosta voidaan olettaa, että strategia voisi olla toimiva myös todellisessa kaupankäyntiympäristössä. Lopullinen strategiapäivitys, johon lisättiin trendisuodatin (SMA200) ja volatiliteettisuodatin (ATR), johti erittäin valikoivaan, mutta samalla tuottavaan kaupankäyntikäyttäytymiseen. Vaikka strategia toteutti vain yhden kaupan neljän vuoden aikana, oli tämä kauppa poikkeuksellisen tuottoisa. Tämä osoittaa, että hyvin määritellyt signaaliehdot voivat vähentää melua ja lisätä signaalien laatua merkittävästi.

On kuitenkin tärkeää suhteuttaa saavutettua tuottoa muihin mahdollisiin sijoitusstrategioihin. Esimerkiksi buy & hold -lähestymistavalla, jossa sijoittaja olisi yksinkertaisesti ostanut Bitcoinia vuoden 2020 alussa ja pitänyt sijoituksensa tutkimusajanjakson loppuun saakka, olisi kokonaistuotto ollut noin 1 286 680 euroa. Tämä ylittää selvästi strategian 174 300 euron lopputuloksen. Näin ollen vaikka kehitetty strategia menestyi validoiduissa testeissä ja osoitti kykyä hallita riskejä valikoivalla kaupankäynnillä, sen tuottopotentiaali jäi merkittävästi jälkeen passiivisesta sijoitusstrategiasta kyseisellä aikavälillä. Tämä havainto korostaa sitä, että strategian tehokkuutta tulee aina arvioida suhteessa yksinkertaisiin vertailustrategioihin, erityisesti markkinoilla, jotka ovat historiallisesti olleet vahvassa nousutrendissä.

Bitcoinin ja muiden kryptovaluuttojen pitkäaikainen holdaaminen voidaan nähdä passiivisen sijoittamisen sijaan myös uhkapelinä. Perinteisten osakkeiden arvo perustuu yritysten liiketoimintaan, omaisuuteen ja vuosittain syntyvään kassavirtaan, kun taas kryptovaluuttojen arvo määräytyy lähes yksinomaan kysynnän ja tarjonnan vaihtelun kautta. Tämä tekee niistä erityisen alttiita spekulaatiolle ja hintakuplille. Historiallisesti ilmiötä voidaan verrata Hollannin 1600-luvun tulppaanimaniaan, jossa sipuleiden hinnat nousivat irti fundamentaalisista perusteista ja romahtivat nopeasti, kun ostajia ei enää

löytynyt (Kindleberger 2000). Kryptovaluuttojen kohdalla tilanne on samankaltainen: arvo perustuu pitkälti siihen, että löytyy aina uusi sijoittaja, joka on valmis maksamaan enemmän. Lisäksi kryptot ovat houkuttelevia rikollisille, sillä niiden avulla voidaan siirtää varoja vaikeammin jäljitettävällä tavalla (European Securities and Markets Authority 2020). Näistä syistä Bitcoinin holdaaminen voidaan kriittisesti arvioiden nähdä enemmän spekulatiivisena toimintana kuin sijoittamisena perinteisessä merkityksessä, vaikka jotkut ovatkin onnistuneet tekemään sillä huomattavia voittoja.

Laajempana johtopäätöksenä voidaan todeta, että yksinkertaisistakin teknisistä indikaattoreista rakennettu strategia voi saavuttaa vaikuttavia tuloksia, mikäli se rakennetaan ja testataan kurinalaisesti ja järjestelmällisesti. Strategian kehitysprosessin monivaiheisuus osoittautui avaintekijäksi sen luotettavuuden ja yleistettävyyden arvioinnissa. Ilman herkkyysanalyysia, robustisuustestejä ja walk-forward-validointia olisi ollut suuri riski ylisovitukseen ja siten heikkoon suorituskykyyn reaaliaikaisessa kaupankäynnissä.

Tulevaisuuden kehitystyössä olisi syytä hyödyntää tutkimuksen tuloksia ja laajentaa strategian käyttöä paper trading -vaiheeseen, jota seuraa mahdollinen live-kaupankäynti. Lisäksi strategian soveltuvuutta tulisi tutkia muilla markkinoilla, kuten osake- tai hyödykemarkkinoilla, sekä lyhyemmillä aikajänteillä. Toinen kiinnostava kehityspolku olisi yhdistää tämä strategia muihin erilaisten logiikkojen strategioihin portfoliomuodossa, jolloin yksittäisten strategioiden heikkouksia voidaan kompensoida ja hajautus parantaa kokonaisuuden riskikorjattua tuottoa.

Lopuksi on syytä todeta, että koneoppimismenetelmien käyttö voisi tarjota seuraavan kehitysaskeleen. Vaikka tämä tutkimus nojasi sääntöpohjaiseen lähestymistapaan, tulevaisuudessa voitaisiin kehittää dynaamisempia, datasta oppivia strategioita. Tällaiset menetelmät voisivat mukautua muuttuvaan markkinaympäristöön tehokkaammin ja mahdollisesti löytää yhteyksiä, joita ihmisen on vaikea havaita. Tämä tutkimus muodostaa vahvan ja systemaattisen pohjan jatkotutkimukselle ja tarjoaa selkeät suuntaviivat strategian viemiseksi seuraavalle tasolle.

LÄHTEET

Aldridge, I. 2013. High-frequency trading: A practical guide to algorithmic strategies and trading systems. Wiley. Viitattu 15.8. 2025 https://ebookcentral-proquest-com.ez.lapinamk.fi/lib/ulapland-ebooks/reader.action?docID=1164806&c=RVBVQg&ppg=1

Almgren, R. & Chriss, N. 2000. Optimal execution of portfolio transactions. Journal of Risk, 3(2), 5–39. Viitattu 20.8.2025 https://www.smallake.kr/wp-content/uploads/2016/03/optliq.pdf

Bailey, D. H., Borwein, J. M., López de Prado, M., & Zhu, Q. J. 2014. The probability of backtest overfitting. Journal of Computational Finance, 20(4), 39–69. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2326253

Barber, B. M. & Odean, T. 2013. The behavior of individual investors. Handbook of the Economics of Finance. Viitattu 10.8.2025 https://faculty.haas.berkeley.edu/odean/papers%20current%20versions/behavior%20of%20individual%20investors.pdf

Bollinger, J. 2001. Bollinger on Bollinger Bands. McGraw-Hill.

GitHub, Inc. 2023. CryptoCurrency eXchange Trading Library. Viitattu 10.7.2025 https://github.com/ccxt/ccxt

Chan, E. P. 2013. Algorithmic trading: Winning strategies and their rationale. Wiley. Viitattu 10.7.2025 https://ebookcentral-proquest-com.ez.lapinamk.fi/lib/ulapland-ebooks/reader.action?docID=1204071&c=RVBVQg&ppg=1

European Securities and Markets Authority. 2020. Report on trends, risks and vulnerabilities. European Securities and Markets Authority. https://www.esma.europa.eu/

Gatev, E., Goetzmann, W. N. & Rouwenhorst, K. G. 2006. Pairs trading: Performance of a relative-value arbitrage rule. The Review of Financial Studies, 19(3), 797–827. Viitattu 10.7.2025 https://doi.org/10.1093/rfs/hhj020

Hasbrouck, J. 2007. Empirical market microstructure: The institutions, economics, and econometrics of securities trading. Oxford University Press. Viitattu 10.7.2025 https://ebookcentral-proquest-com.ez.lapinamk.fi/lib/ulapland-ebooks/reader.action?docID=415271&c=RVBVQg&ppg=1

Hendershott, T., Jones, C. M., & Menkveld, A. J. 2011. Does algorithmic trading improve liquidity? The Journal of Finance. Viitattu 10.7.2025 https://onlinelibrary.wiley.com/doi/epdf/10.1111/j.1540-6261.2010.01624.x

Kindleberger, C. P. 2000. Manias, panics, and crashes. Wiley. Viitattu 10.7.2025 https://delong.typepad.com/manias.pdf

Kirilenko, A. A., Kyle, A. S., Samadi, M., & Tuzun, T. 2017. The flash crash: High-frequency trading in an electronic market. The Journal of Finance. Viitattu 20.8.2025 https://www.repository.cam.ac.uk/bitstream/1810/304244/1/KirilenkoFlashCrash.pdf

Kissell, R. 2013. The science of algorithmic trading and portfolio management. Academic Press. Viitattu 10.7.2025 https://storage.sandtears.com/06_Book/The%20Science%20of%20Algorithmic%20Trading%20and%20Portfolio%20Management%2C%20Robert%20Kissell.pdf

López de Prado, M. 2018. Advances in financial machine learning. Wiley. Viitattu 20.8.2025 https://agorism.dev/book/finance/ml/Marcos%20Lopez%20de%20Prado%20-%20Advances%20in%20Financial%20Machine%20Learning-Wiley%20%282018%29.pdf

McKinney, W. 2017. Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. O'Reilly Media.

Murphy, J. J. 1999. Technical analysis of the financial markets. New York Institute of Finance. Viitattu 10.7.2025 https://dl.fxf1.com/books/english/Murphy%20-%20Tech%20Analysis%20Of%20The%20Financial%20Markets.pdf

Narang, R. K. 2013. Inside the black box: A simple guide to quantitative and high-frequency trading. Wiley. Viitattu 10.7.2025 https://ebookcentral-proquest-com.ez.lapinamk.fi/lib/ulapland-ebooks/reader.action?docID=1143552&c=RVBVQg&ppg=1

O'Hara, M. 1995. Market microstructure theory. Blackwell. Viitattu 10.7.2025 https://www.scribd.com/document/349585448/Market-Microstructure-Theory-pdf

Python Software Foundation. 2023. itertools — Functions creating iterators for efficient looping. Python 3.11. Dokumentaatio. Viitattu 20.8.2025 https://docs.python.org/3/library/itertools.html

QuantConnect. 2023. Lean Algorithm Framework Documentation. Viitattu 20.8.2025 https://www.quantconnect.com/docs

Treleaven, P., Galas, M., & Lalchand, V. 2013. Algorithmic trading review. Communications of the ACM. Viitattu 20.8.2025 https://www.researchgate.net/publication/262239006_Algorithmic_Trading_Review

Wilder, J. W. 1978. New concepts in technical trading systems. Trend Research. Viitattu 10.7.2025 https://www.scribd.com/document/523273115/New-Concepts-in-Technical-Trading-Systems

LIITTEET

Liite 1. datan_noutaminen.py

Liite 2. mean_reversion_backtesti.py

Liite 3. mean_reversion_backtesti_optimointi.py

Liite 4. walk_forward.py

Liite 5. mean_reversion_backtesti_optimoitu.py

Liite 1. datan_noutaminen.py

```
import ccxt
import pandas as pd
from datetime import datetime, timedelta
import time

exchange = ccxt.binance()

symbol = 'BTC/USDT'
timeframe = '1d'
since = exchange.parse8601('2020-01-01T00:00:00Z')
end_date = exchange.parse8601('2024-12-31T00:00:00Z')
all_data = []

limit = 1000

print("Haetaan dataa...")
while since < end_date:
    try:
            ohlcv = exchange.fetch_ohlcv(symbol, timeframe=timeframe, since=since,
limit=limit)
        if not ohlcv:
            break
        all_data.extend(ohlcv)

        last_timestamp = ohlcv[-1][0]
        since = last_timestamp + 3600000

        time.sleep(exchange.rateLimit / 1000)

    except Exception as e:
        print(f"Virhe haussa: {e}")
        time.sleep(10)

df = pd.DataFrame(all_data, columns=['timestamp', 'open', 'high', 'low', 'close',
'volume'])
```

```
df['datetime'] = pd.to_datetime(df['timestamp'], unit='ms')
df.set_index('datetime', inplace=True)

filename = 'BTCUSDT_1d_2020_2024.csv'
df.to_csv(filename)
print(f"Tallennettu: {filename}")
```

Liite 2: mean_reversion_backtesti.py

```
import backtrader as bt
import pandas as pd
import matplotlib.pyplot as plt


df = pd.read_csv("BTCUSDT_1d_2020_2024.csv")

df["datetime"] = pd.to_datetime(df["datetime"], errors="coerce")
df.set_index("datetime", inplace=True)


for col in ["open", "high", "low", "close", "volume"]:
    df[col] = pd.to_numeric(df[col], errors="coerce")


df.replace([float("inf"), float("-inf")], pd.NA, inplace=True)


df = df[(df[["open", "high", "low", "close", "volume"]] != 0).all(axis=1)]


df.dropna(inplace=True)


print("Rivejä jäljellä:", len(df))
print("Mahdollisia nollia:", (df[["open", "high", "low", "close", "volume"]] == 0).sum())


class PandasData(bt.feeds.PandasData):
    params = {
        "datetime": None,
        "open": "open",
        "high": "high",
        "low": "low",
        "close": "close",
```

```
        "volume": "volume",
        "openinterest": -1
    }


class MeanReversionStrategy(bt.Strategy):
    params = (
        ("rsi_period", 14),
        ("bb_period", 20),
        ("rsi_lower", 30),
        ("rsi_upper", 70),
    )

    def __init__(self):
        self.rsi = bt.indicators.RSI(self.data.close, period=self.params.rsi_period)
        self.bb = bt.indicators.BollingerBands(period=self.params.bb_period)
        self.order = None
        self.trade_count = 0


        self.addminperiod(max(self.params.rsi_period, self.params.bb_period))

        self.min_period = max(self.params.rsi_period, self.params.bb_period)

    def notify_order(self, order):
        if order.status in [order.Completed, order.Canceled, order.Margin]:
            self.order = None

    def next(self):

        if len(self.data) < self.min_period:
            return

        if self.order:
            return
```

```
        if not self.position:
            if self.data.close[0] < self.bb.lines.bot[0] and self.rsi[0] < self.params.rsi_lower:
                self.order = self.buy()
                self.trade_count += 1
                print(f"{self.data.datetime.date(0)} >> Osto: Close={self.data.close[0]:.2f}")
        else:
            if self.data.close[0] > self.bb.lines.top[0] or self.rsi[0] > self.params.rsi_upper:
                self.order = self.close()
                print(f"{self.data.datetime.date(0)} >> Myynti: Close={self.data.close[0]:.2f}")


data = PandasData(dataname=df)
cerebro = bt.Cerebro()
cerebro.adddata(data)
cerebro.addstrategy(MeanReversionStrategy)
cerebro.broker.set_cash(100000)
cerebro.broker.setcommission(commission=0.005)
cerebro.broker.set_slippage_perc(perc=0.005)


print("Alkupääoma: %.2f" % cerebro.broker.getvalue())
strategies = cerebro.run()
print("Loppupääoma: %.2f" % cerebro.broker.getvalue())
strategy = strategies[0]
print("Kauppoja tehty: %d" % strategy.trade_count)


fig = cerebro.plot(style='candlestick', use='mpl')[0][0]
fig.savefig("strategia_tulos.png", dpi=300)
```

Liite 3. mean_reversion_backtesti_optimointi.py

```
import backtrader as bt
import pandas as pd
import itertools

df = pd.read_csv("BTCUSDT_1d_2020_2024.csv")
df["datetime"] = pd.to_datetime(df["datetime"], errors="coerce")
df.set_index("datetime", inplace=True)

for col in ["open", "high", "low", "close", "volume"]:
    df[col] = pd.to_numeric(df[col], errors="coerce")

df.replace([float("inf"), float("-inf")], pd.NA, inplace=True)
df = df[(df[["open", "high", "low", "close", "volume"]] != 0).all(axis=1)]
df.dropna(inplace=True)

class PandasData(bt.feeds.PandasData):
    params = {
        "datetime": None,
        "open": "open",
        "high": "high",
        "low": "low",
        "close": "close",
        "volume": "volume",
        "openinterest": -1
    }

class MeanReversionStrategy(bt.Strategy):
    params = (
        ("rsi_period", 14),
        ("bb_period", 20),
        ("rsi_lower", 30),
        ("rsi_upper", 70),
    )

    def __init__(self):
```

```
        self.rsi = bt.indicators.RSI(self.data.close, period=self.params.rsi_period)
        self.bb = bt.indicators.BollingerBands(period=self.params.bb_period)
        self.order = None

    def next(self):
        if self.order:
            return

        if not self.position:
            if self.data.close[0] < self.bb.lines.bot[0] and self.rsi[0] < self.params.rsi_lower:
                self.order = self.buy()
        else:
            if self.data.close[0] > self.bb.lines.top[0] or self.rsi[0] > self.params.rsi_upper:
                self.order = self.close()

rsi_period_range = range(10, 21, 2)
bb_period_range = range(15, 26, 2)
rsi_lower_range = range(25, 36, 5)
rsi_upper_range = range(65, 76, 5)

results = []

for rsi_p, bb_p, rsi_l, rsi_u in itertools.product(rsi_period_range, bb_period_range,
rsi_lower_range, rsi_upper_range):
    cerebro = bt.Cerebro()
    cerebro.broker.set_cash(100000)
    cerebro.broker.setcommission(commission=0.001)
    cerebro.broker.set_slippage_perc(perc=0.0005)
    data = PandasData(dataname=df)
    cerebro.adddata(data)
    cerebro.addstrategy(
        MeanReversionStrategy,
        rsi_period=rsi_p,
        bb_period=bb_p,
        rsi_lower=rsi_l,
        rsi_upper=rsi_u
```

```
    )
    try:
        strategies = cerebro.run()
        final_value = cerebro.broker.getvalue()
        results.append((rsi_p, bb_p, rsi_l, rsi_u, final_value))
    except Exception as e:
        print(f"Virhe yhdistelmällä ({rsi_p}, {bb_p}, {rsi_l}, {rsi_u}): {e}")

df_results = pd.DataFrame(results, columns=["RSI", "BB", "RSI_lower", "RSI_upper",
"Final_Value"])
df_results.sort_values("Final_Value", ascending=False, inplace=True)
df_results.to_csv("optimointitulokset.csv", index=False)
print(df_results.head())
```

Liite 4. walk_forward.py

```
import pandas as pd
import itertools
import backtrader as bt
from datetime import timedelta

df = pd.read_csv("BTCUSDT_1d_2020_2024.csv")
df["datetime"] = pd.to_datetime(df["datetime"], errors="coerce")
df.set_index("datetime", inplace=True)
for col in ["open", "high", "low", "close", "volume"]:
    df[col] = pd.to_numeric(df[col], errors="coerce")
df.dropna(inplace=True)

class PandasData(bt.feeds.PandasData):
    params = dict(datetime=None, open='open', high='high', low='low',
                  close='close', volume='volume', openinterest=-1)

class MeanReversionStrategy(bt.Strategy):
    params = (("rsi_period", 14), ("bb_period", 20), ("rsi_lower", 30), ("rsi_upper", 70))
    def __init__(self):
        self.rsi = bt.indicators.RSI(self.data.close, period=self.params.rsi_period)
        self.bb = bt.indicators.BollingerBands(period=self.params.bb_period)
        self.order = None
    def next(self):
        if self.order: return
        if not self.position:
            if self.data.close[0] < self.bb.lines.bot[0] and self.rsi[0] < self.params.rsi_lower:
                self.order = self.buy()
        else:
            if self.data.close[0] > self.bb.lines.top[0] or self.rsi[0] > self.params.rsi_upper:
                self.order = self.close()

train_period = timedelta(days=365 * 1.5)
test_period = timedelta(days=180)
start_date = df.index[0]
results = []
```

```
while start_date + train_period + test_period <= df.index[-1]:
    train_end = start_date + train_period
    test_end = train_end + test_period
    train_data = df[start_date:train_end]
    test_data = df[train_end:test_end]

    param_combinations = list(itertools.product([12, 14], [18, 20], [25, 30], [65, 70]))
    best_value = -float('inf')
    best_params = None

    for rsi_p, bb_p, rsi_l, rsi_u in param_combinations:
        cerebro = bt.Cerebro()
        cerebro.broker.set_cash(100000)
        cerebro.broker.setcommission(commission=0.001)
        cerebro.broker.set_slippage_perc(perc=0.0005)
        data = PandasData(dataname=train_data)
        cerebro.adddata(data)
        cerebro.addstrategy(MeanReversionStrategy,
                            rsi_period=rsi_p, bb_period=bb_p,
                            rsi_lower=rsi_l, rsi_upper=rsi_u)
        try:
            cerebro.run()
            value = cerebro.broker.getvalue()
            if value > best_value:
                best_value = value
                best_params = (rsi_p, bb_p, rsi_l, rsi_u)
        except:
            continue

    cerebro = bt.Cerebro()
    cerebro.broker.set_cash(100000)
    cerebro.broker.setcommission(commission=0.001)
    cerebro.broker.set_slippage_perc(perc=0.0005)
    data = PandasData(dataname=test_data)
    cerebro.adddata(data)
```

```
    cerebro.addstrategy(MeanReversionStrategy,
                        rsi_period=best_params[0], bb_period=best_params[1],
                        rsi_lower=best_params[2], rsi_upper=best_params[3])
    try:
        cerebro.run()
        test_value = cerebro.broker.getvalue()
        results.append({
            "Train Start": start_date.date(),
            "Train End": train_end.date(),
            "Test End": test_end.date(),
            "RSI": best_params[0],
            "BB": best_params[1],
            "RSI_lower": best_params[2],
            "RSI_upper": best_params[3],
            "Test Final Value": test_value
        })
    except:
        continue

    start_date = train_end

df_results = pd.DataFrame(results)
df_results.to_csv("walk_forward_results.csv", index=False)
print(df_results)
```

Liite 5. mean_reversion_backtesti_optimoitu.py

```
import backtrader as bt
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("BTCUSDT_1d_2020_2024.csv")
df["datetime"] = pd.to_datetime(df["datetime"], errors="coerce")
df.set_index("datetime", inplace=True)

for col in ["open", "high", "low", "close", "volume"]:
    df[col] = pd.to_numeric(df[col], errors="coerce")

df.replace([float("inf"), float("-inf")], pd.NA, inplace=True)
df = df[(df[["open", "high", "low", "close", "volume"]] != 0).all(axis=1)]
df.dropna(inplace=True)

print("Rivejä jäljellä:", len(df))
print("Mahdollisia nollia:", (df[["open", "high", "low", "close", "volume"]] == 0).sum())

class PandasData(bt.feeds.PandasData):
    params = {
        "datetime": None,
        "open": "open",
        "high": "high",
        "low": "low",
        "close": "close",
        "volume": "volume",
        "openinterest": -1
    }

class UpdatedMeanReversionStrategy(bt.Strategy):
    params = (
        ("rsi_period", 12),
        ("bb_period", 18),
        ("rsi_lower", 25),
```

```
        ("rsi_upper", 65),
        ("sma_period", 200),
        ("atr_period", 14),
        ("atr_threshold", 0.05),
    )

    def __init__(self):
        self.rsi = bt.indicators.RSI(self.data.close, period=self.params.rsi_period)
        self.bb = bt.indicators.BollingerBands(period=self.params.bb_period)
        self.sma = bt.indicators.SMA(self.data.close, period=self.params.sma_period)
        self.atr = bt.indicators.ATR(period=self.params.atr_period)
        self.order = None
        self.trade_count = 0

    def next(self):
        if self.order:
            return

        if self.atr[0] > self.data.close[0] * self.params.atr_threshold:
            return

        if self.data.close[0] >= self.sma[0]:
            return

        if not self.position:
            if self.data.close[0] < self.bb.lines.bot[0] and self.rsi[0] < self.params.rsi_lower:
                self.order = self.buy()
                self.trade_count += 1
                print(f"{self.data.datetime.date(0)} >> Osto: Close={self.data.close[0]:.2f}")
        else:
            if self.data.close[0] > self.bb.lines.top[0] or self.rsi[0] > self.params.rsi_upper:
                self.order = self.close()
                print(f"{self.data.datetime.date(0)} >> Myynti: Close={self.data.close[0]:.2f}")

data = PandasData(dataname=df)
```

```
cerebro = bt.Cerebro()
cerebro.adddata(data)
cerebro.addstrategy(UpdatedMeanReversionStrategy)
cerebro.broker.set_cash(100000)
cerebro.broker.setcommission(commission=0.001)
cerebro.broker.set_slippage_perc(perc=0.0005)

print("Alkupääoma: %.2f" % cerebro.broker.getvalue())
strategies = cerebro.run()
print("Loppupääoma: %.2f" % cerebro.broker.getvalue())

strategy = strategies[0]
print("Kauppoja tehty: %d" % strategy.trade_count)

fig = cerebro.plot(style='candlestick', use='mpl')[0][0]
fig.savefig("updated_strategy_result.png", dpi=300)
```