EKONOMICKÁ UNIVERZITA V BRATISLAVE
NÁRODOHOSPODÁRSKA FAKULTA

Evidenčné číslo: 101006/B/2019/36100138876091140

Aplikácia programovacieho jazyka Python vo finančnej analýze

Bakalárska práca

2019 **Filip Frola**

EKONOMICKÁ UNIVERZITA V BRATISLAVE
NÁRODOHOSPODÁRSKA FAKULTA

Aplikácia programovacieho jazyka Python vo finančnej analýze

Bakalárska práca

Študijný program: Financie, bankovníctvo a investovanie
Študijný odbor: Financie, bankovníctvo a investovanie
Školiace pracovisko: Katedra financií NHF
Vedúci záverečnej práce: Mgr. Marek Káčer, PhD.

Bratislava 2018 **Filip Frola**

Poďakovanie

Touto cestou chcem poďakovať vedúcemu mojej záverečnej práce Mgr. Marekovi Káčerovi, PhD., za pomoc, vedenie a rady, ktoré mi poskytol počas tvorby bakalárskej práce.

ABSTRAKT

FROLA, Filip: *Aplikácia programovacieho jazyka Python vo finančnej analýze* – Ekonomická univerzita v Bratislave. Národohospodárska fakulta; Katedra financií. – Vedúci záverečnej práce: Mgr. Marek Káčer, PhD. – Bratislava: NHF EU, 2019, 68 s.

Cieľom tejto práce je implementovať dve rozdielne koncepcie technickej analýzy na forexových trhoch vo forme dvoch plne automatizovaných algoritmov vytvorených v programovacom jazyku Python a ich následnú komparáciu za účelom porovnania dosiahnutého zisku v jednotkách pohybu ceny. Obe stratégie boli úspešne uvedené na forexový trh. Výsledkom vykonanej implementácie a optimalizácie je porovnanie oboch algoritmov. Automatizovaný obchodný systém naprogramovaný primárne pomocou knižnice Pandas a API brány u forex brokera Oanda založený na identifikácií distribučného pásma sprevádzaného agresívnym opustením ceny vo forme voidu a následnom zadaní limitnej objednávky na úroveň návratu dosiahol na 15 minútovom časovom rámci lepšie výsledky ako algoritmus založený na krížení EMA hodnôt. Rozdiel medzi počtom dosiahnutých ziskových a stratových jednotkách pohybu ceny je 3688 jednotiek v prospech zisku pri RRR 3,34:1 proti 19 jednotkám RRR 1,18:1 v prípade druhého funkčného algoritmu.

Kľúčové slová: Python, Forex, Algorithmic trading, Market Range, Technická analýza

ABSTRACT

FROLA, Filip: *Financial analysis with Python programming Language* – University of Economics in Bratislava. Faculty of National Economy; Department of finance. – Thesis supervisor: Mgr. Marek Káčer, PhD. – Bratislava: NHF EU, 2019, 68 p.

The main objective of this bachelor's thesis is to design, implement and compare two different concepts of technical analysis approach to forex markets using two automated algorithms written in Python programming language. Outcome of this comparison is a result on whether an algorithm based on EMA crossover or one based on the identification of market range followed by explosive void penetration outside the range achieves better trading results in terms of absolute difference between won and lost number of pips. The comparison was made after a significant timeframe, trading hours, stop-loss, take-profit and currency selection optimization. Market range based trading algorithm with limit orders for entries on 15 minutes timeframe managed to overcome the EMA crossover trading algorithm with the difference of 3688 between profit and loss number of pips in favor of profit compared to the other algorithm's difference of 19 pips. The EMA crossover algorithm uses just-in-time market execution. Both trading systems are written using standard python libraries, Pandas, Oanda API and Datetime library.

Key words: Python, Forex, Algorithmic trading, Market Range, Technical analysis

OBSAH

Úvod

Automatizácia zasiahla životy každého z nás. Prebieha vo všetkých odvetviach od poľnohospodárstva a ťažby uhlia až po komunikáciu so zákazníkmi a diagnostiku pacientov. Výnimkou nie sú ani forexové trhy, na ktorých denne algoritmy vykonávajú viac ako 80 % denného objemu celkových obchodov.

V bakalárskej práci sa venujeme obchodovaniu na forexových trhoch a jej automatizácií vo forme obchodných algoritmov vytvorených v programovacom jazyku Python. Hlavným cieľom je porovnať výkonnosť dvoch stratégií založených na rozdielnych konceptoch technickej analýzy.

Pri spracovaní danej témy čerpáme prevažne z anglických knižných zdrojov a článkov, technickej dokumentácie ku knižniciam programovacieho jazyka Python a internetových zdrojov.

Bakalárska práca pozostáva z troch kapitol. V prvej kapitole sa budeme venovať aktuálnej situácií na trhoch a teoretickým východiskám pre vytvorenie automatizovaného obchodného systému. V tejto kapitole sa taktiež nachádza definícia hlavných a vedľajších cieľov.

V druhej časti práce si opíšeme metodiku práce a použité metódy skúmania. Vymedzíme si základný rámec a ukazovatele pri porovnávaní dvoch algoritmov a zoznámime sa so základnými nástrojmi a konceptami využívanými pri programovaní a vyhodnocovaní dvoch obchodných systémov.

Tretia kapitola popisuje funkcionality a proces získavania údajov, analýzy, vstupovania do pozícií, vyhodnocovania a optimalizácie oboch algoritmov. Taktiež v tretej kapitole nájdeme analýzu a rozbor výsledkov, podstúpené kroky pri optimalizácií algoritmov a záverečné porovnanie.

Záver práce pojednáva o súlade s vytýčenými cieľmi a popisuje výsledky práce. Porovnáva konkrétne výstupy s konkrétnymi cieľmi a ponúka záverečný komentár k problematike.

1 Súčasný stav riešenej problematiky doma a v zahraničí

1.1 Úvod do problematiky a rozdielnosti technickej a fundamentálnej analýzy

Forexové trhy sú nevyspytateľné. Nepredvídateľné. Volatilné. Neúprosné. Každá pozícia, ktorá je zisková pre jedného obchodníka, končí stratou pre toho druhého. Pri obchodovaní na forexovom trhu, rovnako ako na akciovom, komoditnom či opčnom sú víťazi a porazení. Na to do akej miery je subjekt ziskový vplýva niekoľko významných faktorov. Medzi tieto faktory patrí risk management, risk-reward ratio (RRR) pozícií, podiel úspešných a neúspešných obchodov, objem finančných prostriedkov s ktorým disponuje na účte, money management[1] a v neposlednom rade samotné rozhodovanie obchodníka a stratégia. V priebehu niekoľkých desiatok rokov sa do popredia dostali dva hlavné tábory, ktoré sa postupne zdokonaľovali. Tieto dva tábory však nie je možné chápať čierno-bielo. Vzniká medzi nimi prienik a sám obchodník si určí v akom pomere ich bude využívať a to hlavne podľa spôsobu, akým pristupuje k analýze trhov.

Tieto dva tábory sú fundamentálna a technická analýza. Z počiatku vo všeobecnosti obchodníci obchodovali pomocou stratégií, ktoré sa opierajú vo veľkom o fundament. Do tejto analýzy zahŕňame sledovanie udalostí ako počasia, správ, vyhlásení a pohľadov rôznych odborníkov. V rámci fundamentálnej analýzy na forexových trhoch sa vo veľkom kladie dôraz na trhové úrokové miery, politický vývoj krajín, kvartálne, polročné a ročné štatistické vyhlásenia, vyhlásenia funkcionárov centrálnych bánk. Obchodník si pomocou tejto analýzy vytvorí pohľad na možné smerovanie ceny a podľa toho vyhodnocuje svoje ďalšie pozície. Ide vo väčšine o strednodobý a dlhodobý pohľad na vývoj trhu. Medzi ďalšie významné prvky fundamentálnej analýzy môže patriť taktiež predpokladaná reakcia ďalších obchodníkov na trhu, nazývaná tiež sentiment. Takáto analýza si vyžaduje pokročilú znalosť bankovníctva, financií, politiky, ekonómie a vo veľkej miere aj medzinárodných vzťahov. Z tohto dôvodu sa rozličné analýzy môžu v značnej miere medzi subjektami líšiť. Pokiaľ obchodník vstupuje do pozícií na základe inštinktu, riadi sa vo väčšine fundamentálnou analýzou. Tento spôsob obchodovania je však v praxi veľmi komplexný a líši sa od zvoleného menového páru, obdobia, obchodných hodín a vo veľkej miere je úzko viazaný na schopnosť rýchleho vyhodnocovania a reakcie. Veľké spoločnosti zúčastňujúce sa na

[1] Kathy Lien – Day Trading and Swing Trading the Currency Market – 2nd edition, John Wiley and Sons Inc, 2009, New Jersey, ISBN: 978-0-470-37736-9, s 105 – 108, 290s

forexovom obchodovaní sú preto vo veľkej konkurenčnej výhode keďže majú často prístup ku kvalitnejším a rýchlejším informáciám ako bežný retailový obchodníci. [2] [3]

Druhým táborom a alternatívou k fundamentálnemu obchodovaniu sa stala technická analýza. Jej rozvoj nabral na intenzite vďaka napredovaniu výpočtovej technológie, matematiky, kvantitatívnych modelov, štatistiky, programovania a prieniku exaktných vied s finančnými trhmi. Pri tejto stratégií predpokladania budúcej ceny sa obchodník opiera o rôzne nástroje, ktoré využívajú geometriu, štatistiku, koreláciu, matematické výpočty a vo veľkej miere sú založené na predošlom cenovom vývoji. Tieto postupy sa vo veľkom využívajú na nízkych časových rámcoch a sú vhodné taktiež pre intradenné, strednodobé a dlhodobé analýzy. V posledných desaťročiach investovali najväčšie finančné inštitúcie nemalé finančné prostriedky na rozvoj stratégií zameraných práve na obchodovanie pomocou technickej analýzy. [4] [5]

Rozdiel medzi dvoma hlavnými prístupmi je viac než zrejmý, no väčšina obchodníkov využíva prvky z oboch obchodných prístupov a snaží sa docieliť čo najoptimálnejšie využitie ich nástrojov na naplnenie svojich úloh či už ide o uloženie, zhodnotenie alebo zabezpečenie svojich finančných aktív.[6] Hlavné koncepčné rozdiely medzi fundamentálnou a technickou analýzou tkvejú v časových rámcoch, risk managemente, v rozdieloch medzi úrovňami vstupu a výstupu z pozície a spôsobe akým sa pristupuje k spracovaniu a analýze dát. Technická analýza poskytuje presne stanovené úrovne vstupov a výstupov spolu s definovanou úrovňou rizika, ktoré sú založené prevažne na kvantitatívnych údajoch z forexového trhu a historických dátach. Pri komplexnejších stratégiách sa využívajú aj údaje z iných ako forexových trhov. Medzi tieto patrí napríklad akciový, opčný, indexový či futures trh.[7] Pri technickom prístupe niekedy obchodník vykonáva mnohonásobne viac obchodov ako subjekt opierajúci sa o fundament. Fundamentálna analýza naopak poskytuje informácie, ktoré by pomocou technickej analýzy neboli dostupné ako napríklad očakávaný menový vývoj, ktorý nemá opodstatnenie z pohľadu historických cien. Medzi takéto situácie môže patriť vystúpenie Anglicka

[2] Ludvik Turek – Jak na Forex Czechwealth, 2009, Praha, s 27 – 32, 104s.
[3] Brian Dolan - Currency Trading for Dummies 2nd edition, Wiley Publishing, Inc., New Jersey, ISBN: 978-1-118-01851-4, s 119 – 140, 332s
[4] Ludvik Turek – Manuál technické analýzy, 2008, Praha, Czechwealth, s 7-18, 272s.
[5] Brian Dolan - Currency Trading for Dummies 2nd edition, Wiley Publishing, Inc., New Jersey, ISBN: 978-1-118-01851-4, s 187 – 189, 332s
[6] Brian Dolan - Currency Trading for Dummies 2nd edition, Wiley Publishing, Inc., New Jersey, ISBN: 978-1-118-01851-4, 203 – 252, 332s
[7] Dr. Alexander Elder – Come Into My Trading Room, 2002, John Wiley and Sons Inc., ISBN 0-471-22534-7, New York, s 15 – 23, 313s

z Európskej Únie alebo rozhodnutie centrálnych bánk o výške depozít alebo úrokovej miere. Oba prístupy sa postupom času zdokonaľujú a ďalej členia na podkategórie podľa nástrojov, ktoré obchodník vo svojej stratégií využíva.

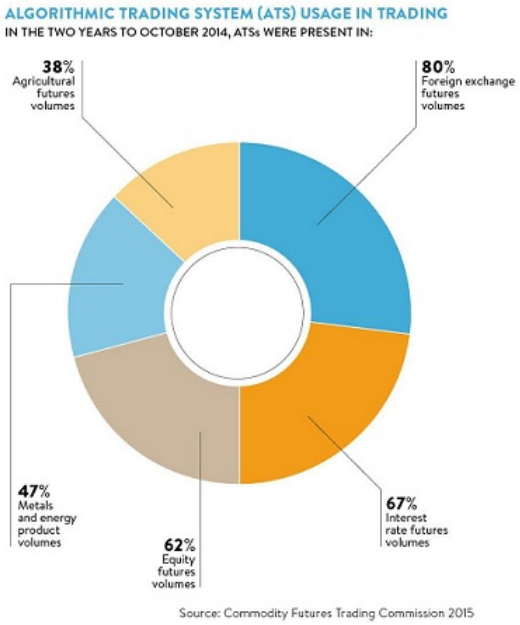
1.2 Systém Vstup a vláda algoritmov na trhu

Algoritmické obchodovanie, taktiež známe pod pojmom high-frequency trading je činnosť automatizovaného obchodovania na finančných trhoch pomocou algoritmu napísaného v programovacom jazyku s predom zadanými inštrukciami podľa ktorých obchodovanie prebieha.[8] Ide o činnosť vykonávanú bez ľudskej interakcie. Tento typ obchodovania sa dostáva do popredia koncom osemdesiatych a začiatkom deväťdesiatych rokov minulého storočia a prvý krát ho implementovali finančné inštitúcie na Wall Street v USA. Momentálny stav na finančných trhoch odzrkadľuje výrok bývalého predsedu burzy NASDAQ, Roberta Greifielda, v Apríli 2011 – „Je koniec. Obchodovanie, ktoré existovalo naprieč storočiami je mŕtve. Dnes máme elektronické trhy. To je súčasnosť."

Pravdivosť tohto výroku je možné dokázať prostredníctvom číselných dát, ktoré máme dnes k dispozícií z rôznych zdrojov. Z údajov komisie pre obchodovanie komodít a futures (Commodity Futures Trading Commision) z roku 2015 vyplýva, že najväčší podiel na objeme celkového množstva obchodov majú algoritmy na trhu forexových futurít a to až 80 percent. Druhý najväčší výskyt automatizovaných obchodných systémov môžeme nájsť na trhu futurít úrokových mier s podielom 67 percent viď. Obrázok č. 1.

[8] David Easley, Marcos López De Prado and Maureen O´Hara – High-Frequency Trading, Incisive Media, London, ISBN 978-1-78272-009-6, s 15 - 20, 236s

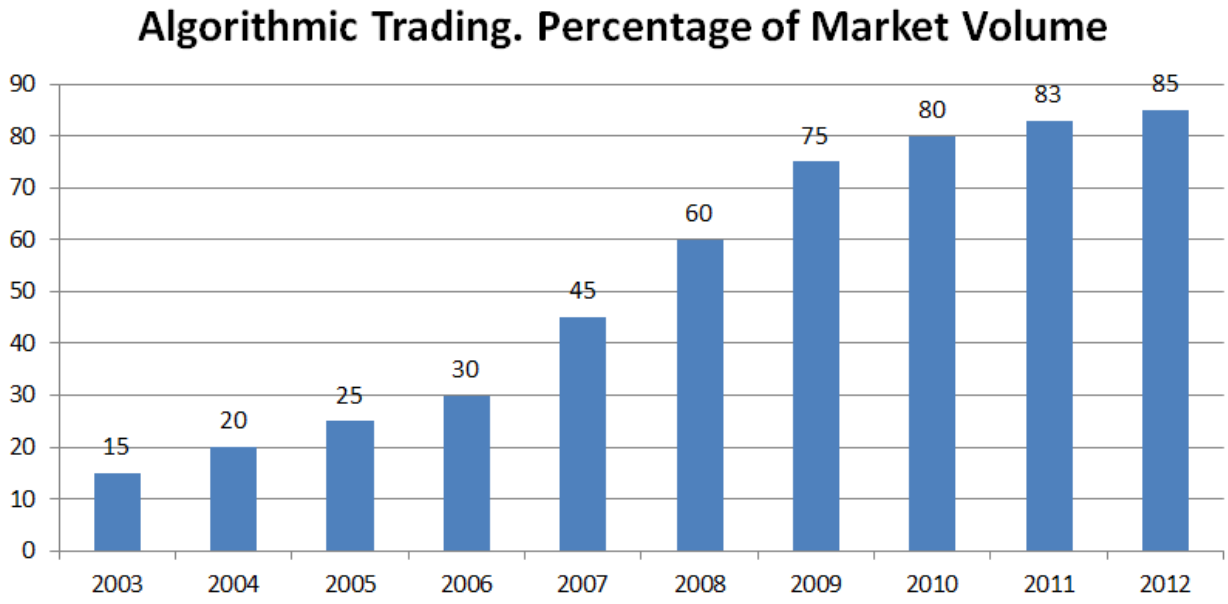
Obrázok č. 1 Používanie algoritmických systémov pri obchodovaní v roku 2014



Zdroj: Commodity Futures Trading Comission 2015

Na základe dát od autorov finančnej literatúry Roberta Kissella a Mortona Glantza bol podiel obchodov vykonaných pomocou algoritmov 85 percent z celkového objemu transakcií v roku 2012 na finančných trhoch. Tento podiel má podľa predpokladov autorov v rokoch 2016 až 2020 rásť o 10,3 percenta. Viď obrázok č. 2.

Obrázok č. 2 Podiel algoritmického obchodovania na celkovom objeme obchodov



Zdroj: 11 Morton Glantz, Robert Kissell

Algoritmické obchodovanie so sebou prinieslo na trh razantne zvýšenú likviditu a taktiež volatilitu, čo spôsobilo, že obchodné stratégie ktoré v minulosti fungovali, prestali

dosahovať predošlú mieru ziskovosti a subjekty, ktoré sa chcú úspešne zúčastňovať na procese obchodovania sú nútené vylepšovať a prispôsobovať svoje stratégie novým pravidlám. Obchodné algoritmy operujú v časových rámoch, v ktorých je manuálne obchodovanie nemožné a preto sa väčšina takýchto systémov využíva na obchody vo veľmi krátkych časových úsekoch, ide o tzv. Skalping.

Vo všeobecnosti môžeme tento typ obchodovania rozdeliť do dvoch majoritných skupín. Prvou skupinou sú algoritmy, ktoré sa zameriavajú na operácie v milisekundách a na ich konkurencieschopnosť vplýva rýchlosť internetového pripojenia, poloha dátového centra z ktorého algoritmus obchoduje, technické prevedenie, ktoré je jedným z determinantov rýchlosti a efektivity a množstvo iných faktorov. Takéto obchodovanie sa tiež označuje ako kvantitatívne, alebo jeho odnož obchodovanie s vysokou frekvenciou. Medzi najčastejšie implementácie patria systémy, ktoré sledujú vývoj diania vo svete a na základe dát vyhodnocujú a predpokladajú nasledovný vývoj podkladového aktíva. Taktiež sa vo veľkej miere využívajú dáta z korelačných aktív a na základe vzájomného vplyvu podkladových aktív sa podľa jedného aktíva predpokladá pohyb toho druhého. Ďalšou veľmi obľúbenou metódou je arbitráž, pri ktorej sa sleduje cena konkrétneho aktíva obchodovaného na rozdielnych burzách a pri odchýlke sa algoritmus snaží dosiahnuť zisk odkúpením a následným predajom u iného subjektu so ziskom.[9]

Druhú skupinu tvoria programy, ktoré sa snažia napodobniť konanie reálneho obchodníka, ktorý operuje na vyšších časových rámoch od 1 minúty po dňové až mesačné periódy. Vo väčšine sa však využíva minútový, päť minútový, pätnásť minútový a hodinový časový rámec na vstup a výstup z pozícií v závislosti, či sa do analýzy zahŕňa viacero časových rámcov pri rovnakom menovom páre, alebo sa obchoduje a rovnako aj vyhodnocuje pozícia vo fixnom časovom rámci. Voľba optimálnej periódy na analýzu a dĺžka obchodu má v konečnom dôsledku priamy vplyv na výsledky z obchodovania. Pri týchto metódach sa využívajú bežné nástroje technickej analýzy a na funkčnú implementáciu je potrebné zvoliť optimálne riadenie peňažných prostriedkov (Money management) a zvážiť využitie stop-loss a take-profit objednávok, ktoré budú nasledovať predom určené pravidlá. Táto práca sa zameriava práve na túto skupinu obchodných systémov. [10]

[9] Aldridge I., 2013, High – Frequency Trading – A Practical Guide to Algorithmic Strategies and Trading Systems, 2nd ed., Hoboken: John Wiley & Sons, Inc.
[10] Chan P. E., 2013 Algorithmic Trading – Winning Strategies and Their Rationale, Hoboken: John Wiley & Sons, Inc.

K tomu, aby bol systém vytvorený a schopný obchodovať bez zásahu je potrebné naplniť niekoľko predpokladov. Prvým je technická zručnosť, vďaka ktorej sa koncept pretransformuje na funkčný celok. Druhým predpokladom je výber správneho poskytovateľa dát a zároveň brokera u ktorého sa obchody budú vykonávať. Nie je podmienkou, aby bol poskytovateľ dát a broker rovnakým subjektom, avšak aby sa predišlo komplikáciám a nadmernej komplexnosti z technického hľadiska ide o preferovaný spôsob vývoja. Ďalšími predpokladmi sú dostatočne rýchle internetové pripojenie a dostupnosť počítača, na ktorom sa obchodovanie bude vykonávať. Taktiež je potrebné zriadiť demo účet u brokera aby sa predišlo prípadnej strate finančných prostriedkov v testovacej fáze. Poskytovateľ dát musí disponovať minimálne minútovými otváracími (open), najvyššími (high), najnižšími (low) a uzatváracími (close) cenami, keďže tieto tvoria takzvané sviečkové (candlestick) dáta. Je nutné aby broker podporoval funkcionalitu vstupu a výstupu z pozícií prostredníctvom zautomatizovaných brán, tzv. API (Application programming interface), ktoré môže program využiť na otváranie pozícií bez potreby zásahu obchodníka.

2 Cieľ práce, metodika práce a metódy skúmania

2.1 Cieľ práce

Prístupov k obchodovaniu je mnoho, avšak začínajúci obchodník si musí vybrať cestu, ktorá mu je sympatická, prípadne na základe porovnania dostupných konceptov a ich výsledkov. Hlavným cieľom tejto práce je vybrať dva obchodné koncepty v rámci technickej analýzy. Každý z týchto konceptov je potrebné dôsledne navrhnúť, vytvoriť, optimalizovať, aplikovať a následne vyhodnotiť pomocou vopred stanovených kritérií a štatistický nástrojov a metód. Hlavné výstupy tejto práce sú teda dve stratégie inšpirované rozdielnymi koncepciami technickej analýzy, ktoré sú implementované na forexové trhy pomocou programovacieho jazyka Python a ich vzájomné porovnanie. Okrem týchto hlavných cieľov môžeme stanoviť taktiež množstvo sprostredkujúcich kritérií, ktoré musia byť v súlade so zadaním na úplne naplnenie podstaty práce.

Práca je rozdelená na tri hlavné celky, z ktorých každý si vyžaduje osobitný prístup a rozdielne prístupy k programovaniu. Aj napriek faktu, že ide o rozdielne prístupy k obchodovaniu, oba algoritmy musia spĺňať niekoľko spoločných podmienok a výhod. Samostatnosť algoritmu je jedna z kľúčových vlastností, ktorá prináša výhodu nízkej časovej záťaže pre obchodníka, taktiež ho odbremení od neželaných psychických vnemov z neúspešných pozícií. Operácia v reálnom čase je schopnosť algoritmov jednať na základe bezprostredne aktuálnych informácií o pohybe ceny a na základe vopred stanovených pravidiel a podmienok vie takéto informácie vyhodnotiť takmer okamžite a podstúpiť potrebné kroky na realizáciu pozície či už na nákup alebo predaj. Ďalším spoločným kritériom je potreba funkcionality na automatické vyhodnocovanie stop-loss a take-profit objednávok. Tieto objednávky nám pri obchodovaní umožňujú zostať konzistentný a lepšie riadiť riziko a možný zisk, ktorý chceme v pozícií dosiahnuť. Úrovne stop-loss a take-profit objednávok musia byť určené na základe predošlého vývoju na trhu a prípadne použitím swing low a swing high cenových úrovní. Stop-loss objednávky sa nastavujú pri zadávaní obchodu a v prípade, že sa naša pozícia nevyvíja podľa očakávaní, uzavrie obchod so stratou, ktorá je v súlade s manažmentom peňažných prostriedkov. Take-profit je naopak limitná objednávka, ktorá udáva hranicu, na ktorej algoritmus uzavrie pozíciu v prípade pozitívneho vývoja so ziskom. Ďalším podstatným kritériom je zabránenie obchodovaniu v nevhodných situáciách a časoch. Na základe analýzy údajov nadobudnutých obchodovaním je možné stratégie optimalizovať za účelom predchádzania obchodovaniu v nevhodných časoch ako

napríklad v noci a podobne. V neposlednom rade majú byť algoritmy schopné operovať ako na viacerých časových pásmach bez významného zásahu do kódu, tak i na rôznych menových pároch nakoľko táto funkcionalita významne rozširuje možnosti pre analýzu potenciálnych vstupov a má priamy vplyv na zväčšovanie štatistickej vzorky potrebnej na vyhodnocovanie úspešnosti.

Prvý z algoritmov bude mať za úlohu získavať dáta vo frekvencií jednej minúty a na základe predom stanovených podmienok vyhodnocovať kríženie exponenciálnych kĺzavých priemerov, konkrétne jednej krátkej a jednej dlhej periódy. Následne má byť schopný určiť limitné stop-loss a take-profit objednávky v súlade s predom stanovenými kritériami a vstúpiť do dlhej alebo krátkej pozície. Druhý z algoritmov je založený na koncepte „Price Action“ a na rozdiel od prvej stratégie sa nespolieha výstredne na EMA indikátory ale na identifikáciu pásiem distribúcie a následných pokusov o výstup ceny z tohto pásma vo forme tzv. „void“ pohybov. Pri výbere tejto stratégie som sa inšpiroval vlastnými skúsenosťami z obchodovania na forexových trhoch. Hlavnou myšlienkou distribučných pásiem je, že v nich dochádza k výberu ziskov a strát subjektov, ktorý sa zúčastňovali na cenovom pohybe. Tieto distribučné pásma sa vyznačujú pohybom ceny v menšom rozpätí, ktorá osciluje v rozpätí do strany. Vyznačuje sa pomyselnou stropovou a podlahovou cenou tvoriacou dno, ktoré sú zväčša identifikovateľné prostredníctvom swing high a swing low patternov. Na potvrdenie takéhoto rozpätia je taktiež vhodné použiť EMA indikátor a to pri jeho sploštení, respektíve pohybe do strany. Z takéhoto rozpätia zväčša cena prerazí do trendu či rastúceho alebo klesajúceho. Pri niektorých únikoch ceny z daných pasiem dochádza k explozívnemu pohybu a často nedochádza k uspokojeniu obchodníkov na úrovniach, ktoré cena prejde rýchlo. Následne je predpoklad, že sa cena do týchto pásiem vráti (pullback) a bude pokračovať v trende reakciou buď na strop cene predošlého distribučného pásma v prípade dlhej (long) pozície alebo na cene predošlého dna v prípade short (krátkej) pozície. Tieto úrovne sú preto ihneď po explozívnom pohybe ceny von z pásma označené a vytvorí sa limitná objednávka. Pri pullbacku sa tak objednávka vyplní so stop-loss a take-profit cenami podľa predošlých swing high alebo swing low cien. Ďalšou podstatnou rozdielnosťou je forma vstupu do pozícií a to vo forme limitných cien miesto aktuálnych (just-in-time). Táto funkcionalita zabezpečuje väčší priestor na manipuláciu risk-reward-ratia tak aby bolo v súlade s našou stratégiou. Obe stratégie majú svoje rozdielnosti a predpokladá sa, že druhá z nich bude koncepčne zložitejšia na spracovanie ako i na optimalizáciu. Na základe skúseností z obchodovania na forexových trhoch je možné odhadovať, že obe stratégie bude veľmi náročné naprogramovať tak, aby dosahovali pozitívne zhodnotenie obchodného účtu

a očakávame, že porovnanie týchto stratégií nám poskytne obraz o vhodnosti a explicitne poukáže na stratégiu, ktorá je vhodnejšia na obdobné aplikácie v budúcnosti.

2.2 Metodika práce a metódy skúmania práce

2.2.1 Spoločné nástroje a metódy využívané pri oboch algoritmoch

Hlavným základom dát pre spracovanie sú otváracie (open), zatváracie (close), najvyššie (high) a najnižšie (low) ceny v rámci jednej periódy časového pásma, taktiež nazývané aj OHLC dáta. Pre príklad u 1 minútového časového rámca tieto dáta tvoria tzv. sviečku. Viacero takýchto sviečok tvorí sviečkové (candlesticks) vyobrazenie pohybu ceny a na jeho základe môžeme aplikovať ďalšie nástroje a hľadať spoločné znaky v podobe tzv. patternov. Po získaní požadovaných candlestick dát určujeme veľkosť jednotlivých sviečok, čo nám v budúcnosti pomôže identifikovať silu pohybu ceny. Niekoľko po sebe nasledujúcich sviečok vytvára pattern, ktorý nazývame swing high a swing low. Ide o body v pohybe ceny, ktoré tvoria tzv. krátkodobé body maxima a minima. Poznáme dva najpoužívanejšie spôsoby výpočtu týchto patternov. V prvom spôsobe bod maxima, respektíve minima vypočítame z troch sviečok a v tom druhom počítame s piatimi sviečkami. Bod maxima je identifikovaný vtedy, keď stredná sviečka má najvyššiu high cenu spomedzi všetkých v skupine. Bod minima je naopak vyhodnotený, keď low cena stredovej sviečky dosiahne najnižšiu hodnotu zo skupiny sviečok vo výpočte. Tieto úrovne sa ďalej využívajú rôzne, v našich stratégiách slúžia na odvodzovanie stop-loss a take-profit objednávok, nakoľko sa očakáva odpor pri opätovnom kontakte s takouto úrovňou a slúžia ako relatívne silné indície o tom, kam by mohol budúci cenový vývoj smerovať.

Pomocou historických OHLC dát sa kalkuluje nástroj technickej analýzy, ktorý sa rôzne, no aktívne využíva v oboch obchodných algoritmoch. Je ním exponenciálny kĺzavý priemer EMA a v jeho výpočte sa zohľadňuje x jednotiek časového rámca odvíjajúc sa od zvolenej periódy, z ktorých sa jeho hodnota neustále priebežne upravuje na základe aktuálnych cien po vstupe nových dát do systému. Výpočet týchto hodnôt zabezpečuje knižnica v programovacom jazyku Python – Ta-Lib. Existujú rôzne typy a periodicity kĺzavých a jednoduchých priemerov v závislosti od preferovanej stratégie a typu obchodovania. Ide o univerzálny nástroj, ktorý sa v praxi využíva pomerne často a dáva obchodníkovi prehľad o aktuálnom trende trhu.

Výpočet exponenciálneho kĺzavého priemeru EMA:

1. Je potrebné vyjadriť jednoduchý kĺzavý priemer SMA vzorcom:

SMA = jednoduchý kĺzavý priemer

p = zvolená časová perióda

pv = uzatváracia cena v jednotke periódy

$$SMA = p - \frac{\sum_{j=1}^{p} pv}{p} \quad (1)$$

2. Pokračujeme vyjadrením násobiteľa pre váženie exponenciálneho kĺzavého priemeru:

   p = zvolená časová perióda

   k = násobiteľ

$$k = \frac{2}{p+1} \quad (2)$$

3. Na záver vyjadríme exponenciálny kĺzavý priemer vzorcom, pričom prvá EMA je rovná SMA:

   EMA = exponenciálny kĺzavý priemer

   t = dnešný obchodný deň

   y = včerajší obchodný deň

   k = násobiteľ

$$EMA = Cena(t) * k + EMA(y) * (1 - k) \quad (3)$$

Čas je neodmysliteľnou súčasťou každej obchodnej stratégie a na to aby sme ho mohli využívať v našich analýzach a podmienkach, ktoré sú nastavené v zdrojovom kóde algoritmov, je potrebné aby bol v univerzálnom formáte 'YYYY-MM-dd HH:mm:ss'. Pomocou knižnice Datetime je možná filtrácia štatistickej vzorky vykonaných obchodov a následná optimalizácia pomocou jednoduchých metód ako .day, .date, .hour a podobne. Umožňuje nám rozlišovať, v ktorých časoch je konkrétna stratégia úspešnejšia a v ktorých naopak menej úspešná. Vďaka správnej interpretácií máme možnosť jednotlivé stratégie optimalizovať, aby vynechali jednotlivé časové úseky ako napríklad nočné hodiny, v ktorých môže systém chybne identifikovať nákupné alebo predajné signály. Taktiež sme v prípade potreby schopný zakázať algoritmom obchodovať v piatok po skončení európskych

obchodných hodín za účelom predchádzania stratovým pozíciám v dôsledku víkendových gapov, ak tieto podmienky vyhodnotíme ako menej ziskové v zrovnaní s priemerom.

Výber vhodných menových párov je signifikantným faktorom priamo ovplyvňujúcim ziskovosť jednotlivej stratégie a preto je dôležité venovať mu náležitú pozornosť. Za hlavné sme zvolili tzv. ‚major' menové páry, ktoré nemajú veľké špecifiká v kótovaní kurzov nakoľko sú v oboch obchodných algoritmoch zakomponované funkcionality, ktoré si vyžadovali explicitne zadefinovanie kurzového pohybu. Z týchto dôvodov nie je do obchodovania zaradený napríklad menový pár USD/JPY. Množina párov pozostáva z EUR/USD, NZD/USD, USD/CHF, GPB/USD, AUD/USD, USD/CAD a GBP/CHF. Tie môžu byť na základe výstupu z analýz zredukované o najmenej ziskové. Výsledná skupina sa teda môže u jednotlivých stratégií líšiť a bude uvedená vo výsledkoch práce spolu s opodstatnením.

Ďalším faktorom, ktorý priamo vplýva na ziskovosť obchodovania je voľba časového rámca. Ako bolo uvedené v úvode práce, primárnym cieľom v súvislosti s výberom časového rámca je operácia v kategórií intradenného obchodovania. Od výberu rámca závisí dĺžka jednotlivých sviečok a taktiež štruktúra dát. Prvou možnosťou je 1 minútový časový rámec, ktorý sa aktívne využíva pri tzv. skalpingu. Pozícia sa začína a rovnako aj končí pri väčšine prípadov v rámci niekoľkých minút. Obchodovanie v tak krátkom horizonte je často vystavované vysokej volatilite a taktiež používané indikátory nemusia vždy spoľahlivo identifikovať očakávaný smer pohybu. Výhodou však je vysoká frekvencia obchodov a v prípade úspešnej stratégie teda dochádza k rýchlejšiemu nárastu obchodného účtu, rovnako sa však negatívne odzrkadľuje účinok spreadu na vykonané obchody. Spread je rozdiel medzi ponúkanou (bid) a požadovanou (ask) cenou, v praxi závisí od výberu brokera a každý poskytuje iné veľkosti spreadov. Všeobecne však platí, čím vyšší spread, tým horšie má obchodník podmienky a ťažšie dosiahne so svojou obchodnou stratégiou ziskovosť. Výhoda 5 minútového rámca je rovnako relatívne vysoká frekvencia obchodov, jemná redukcia tzv. šumu v pohybe ceny a taktiež v malej miere redukuje negatívny dopad spreadu nakoľko sa pozície otvárajú v absolútnom vyjadrení na väčší pohyb ceny. Posledným vhodným časovým rámcom je 15 minútový, pri ktorom síce pozície trvajú dlhšie, avšak poskytuje značnú redukciu šumu a taktiež spread je relatívne k veľkosti pozície menší. Optimálne časové pásmo, ktoré sa bude využívať v stratégií je najlepšie získať pomocou uskutočnených obchodov na viacerých pásmach, porovnaním indikátorov ako pomer ziskových k stratovým pozíciám a rozdiel medzi absolútnymi hodnotami dosiahnutých stratových a ziskových jednotiek pohybu ceny v pipoch. Taktiež bude vhodné porovnať

dĺžku získavania vzorky obchodov a priemernú dĺžku trvania jednej pozície. Nakoľko sa jednotlivé stratégie správajú rozdielne v rôznych rámcoch, najväčší rozdiel ziskovosti sa očakáva medzi 1 minútovým a 15 minútovým rámcom.

Za účelom dosiahnutia prehľadnosti a zabezpečenia procesného spôsobu komunikácie medzi jednotlivými modulmi sa vo vytvorených algoritmoch používa taktiež tabuľkový procesor Excel. Prostredníctvom neho docielime funkcionalitu ukladania vyhodnotených dát a ich následné spracovanie v ďalšom module ako napríklad pri signáloch. Po úspešnom vyhodnotení vstupnej ceny pre pozíciu, stop-loss a take-profit úrovní sa tieto údaje spolu s inými dátami ako s časom, pozíciou sviečky na ktorej sa signál vyskytol a typom objednávky vo formáte BUY alebo SELL uložia do prislúchajúceho riadku, ktorý môže byť v objednávkových moduloch analyzovaný a ďalej upravený pomocou knižníc ako Pandas alebo xlrd. Po vstupe do pozície sa údaje o otvorenej pozícií vo formáte, ktorý závisí od zdrojového kódu konkrétnej stratégie, uloží do ďalšieho zošitu vo formáte .xlsx a slúži na zamedzenie objednávok, ktoré by mohli byť opätovne vyhodnotené bezprostredne po objednávke už zrealizovanej. Tento problém je nutné ošetriť podobným spôsobom ako u algoritmu založenom na krížení hodnôt EMA, tak i u algoritmu, ktorý sa opiera o limitné objednávky na úrovniach prieniku ceny z distribučných zón. Je potrebné aby systém vedel vyhodnotiť a predísť takýmto zdvojeným objednávkam na rovnakých menových pároch v aktuálnom čase.

2.2.2 Náležitosti vyhodnocovania a optimalizácie algoritmu založeného na krížení EMA

Pri prvej stratégií nás po úspešnom naprogramovaní a uvedení na demo účet bude čakať niekoľko výziev, ktoré je potrebné riešiť. Jednou z nich je zozbieranie dostatočnej vzorky na vyhodnocovanie. Veľkosť vzorky je stanovená na minimum 200 obchodov, ktoré by mali byť dostatočne veľký počet na identifikáciu slabých stránok stratégie a určenie nedostatkov, ktoré budú predmetom optimalizácie. Predmetom optimalizácie môžu byť dáta ako obchodné hodiny, obmedzenie konkrétnych dní v týždni, obmedzenie menových párov a podobne. Hlavný ukazovateľ, ktorý určuje schopnosť stratégie byť ziskovou je pomer ziskových pozícií na celkovom počte vykonaných obchodov. Tento ukazovateľ by bol veľmi podstatný v prípade, že by pozície mali priemerný RRR (risk-reward-ratio) 1:1, avšak v tejto konkrétnej stratégii sa využíva pomer vyšší v prospech ziskových pozícií. V praxi to znamená, že aj pri 40% ziskových pozícií je možné dosahovať zisk nakoľko ziskové pozície sú väčšie v absolútnom vyjadrení pohybu jednotiek ceny oproti tým stratovým. Preto

potrebujeme vypočítať aj rozdiel v celkových počtoch jednotiek pohybu pre jednotlivé ziskové a stratové pozície. Na základe takýchto informácií vyhodnotíme RRR. Následne je potrebné výsledky interpretovať a postupovať na body časovej a menovej optimalizácie. Po vyhodnotení najvhodnejšej úpravy algoritmu sa výsledky po optimalizácií opätovne vyjadria v rovnakých ukazovateľoch, interpretujú a porovnajú s druhou obchodnou stratégiou.

2.2.3 Náležitosti vyhodnocovania a optimalizácie algoritmu založeného na identifikácií distribúcie

Pri programovaní a vyhodnocovaní druhej stratégie je preferované zvoliť rozdielny prístup nakoľko je potrebné otestovať rozdielne časové rámce, prístupov k zadávaniu take-profit a stop-loss objednávok a taktiež otestovať schopnosti stratégie bez a vrátane spread-u. Takéto rozsiahle testovanie a vyhodnocovanie si vyžaduje pri obchodoch v reálnom čase niekoľko dní, prípadne týždňov v závislosti od jednotlivých parametrov pre každú testovaciu verziu. Toto špecifikum sa prejavuje hlavne pri vyšších časových rámcoch ako 5M a 15M, kde je priemerný čas trvania pozície razantne dlhší ako pri pozíciách vykonaných na 1M. Aby sme mohli vykonať porovnanie naprieč niekoľkými verziami s rozličnými podmienkami, pri zachovanej veľkosti štatistickej vzorky minimálne v početnosti 200 obchodov pre každú verziu, je nutné využiť testovanie na historických dátach. To je však možné až po vytvorení funkčného algoritmu, ktorý je schopný operovať v reálnych podmienkach a následne vytvoriť takzvaný ‘backtesting’ modul. Takýto modul je schopný vyhodnotiť úspešnosť stratégie v pomerne veľmi krátkom čase pri zachovaní veľkosti štatistickej vzorky. Ďalšou výhodou analýzy historických dát je nemennosť podmienok pri vykonávaní obchodov na rovnakom časovom rámci pri zmenených take-profit a stop-loss objednávkach a taktiež nám poskytne dáta na porovnanie vplyvu spreadu pri danej obchodnej stratégii. Pomocou neho môžeme vyvodiť závery o účinnosti zmien, ktoré sme vykonali v rámci optimalizácie keďže časové obdobie a podmienky na trhu sú nezmenené za predpokladu porovnávania rovnakého časového rámca.

2.2.4 Náležitosti vyhodnocovania a optimalizácie algoritmu založenom na identifikácií distribúcie

Pri programovaní a vyhodnocovaní druhej stratégie je preferované zvoliť rozdielny prístup nakoľko je potrebné otestovať rozdielne časové rámce, prístupov k zadávaniu take-profit a stop-loss objednávok a taktiež otestovať schopnosti stratégie bez a vrátane spread-u. Takéto rozsiahle testovanie a vyhodnocovanie si vyžaduje pri obchodoch v reálnom čase

niekoľko dní, prípadne týždňov v závislosti od jednotlivých parametrov pre každú testovaciu verziu. Toto špecifikum sa prejavuje hlavne pri vyšších časových rámcoch ako 5M a 15M, kde je priemerný čas trvania pozície razantne dlhší ako pri pozíciách vykonaných na 1M. Aby sme mohli vykonať porovnanie naprieč niekoľkými verziami s rozličnými podmienkami, pri zachovanej veľkosti štatistickej vzorky minimálne v početnosti 200 obchodov pre každú verziu, je nutné využiť testovanie na historických dátach. To je však možné až po vytvorení funkčného algoritmu, ktorý je schopný operovať v reálnych podmienkach a následne vytvoriť takzvaný ‘backtesting’ modul. Takýto modul je schopný vyhodnotiť úspešnosť stratégie v pomerne veľmi krátkom čase pri zachovaní veľkosti štatistickej vzorky. Ďalšou výhodou analýzy historických dát je nemennosť podmienok pri vykonávaní obchodov na rovnakom časovom rámci pri zmenených take-profit a stop-loss objednávkach a taktiež nám poskytne dáta na porovnanie vplyvu spreadu pri danej obchodnej stratégii. Pomocou neho môžeme vyvodiť závery o účinnosti zmien, ktoré sme vykonali v rámci optimalizácie keďže časové obdobie a podmienky na trhu sú nezmenené za predpokladu porovnávania rovnakého časového rámca.

Modul na backtest však pracuje v historických cenách a na svoje fungovanie nepotrebuje funkcionalitu obchodovania v reálnom čase. Za týmto účelom je potrebné najskôr vytvoriť funkčnú verziu algoritmu, ktorý obsahuje spomenutú funkcionalitu tzv. live trading a až následne aplikovať backtesting na vyhodnotené signály. Za účelom zefektívnenia vyhodnocovacieho procesu je vhodné vytvoriť samostatný Python script (súbor obsahujúci spustiteľný kód v programovacom jazyku – pozn. autora), ktorý využije excel zošit vo formáte .xlsx obsahujúci údaje o zrealizovaných obchodoch v chronologickom usporiadaní. Na základe týchto dát automaticky spracuje informácie a vyhodnotí požadované ukazovatele ako priemerné RRR, priemerná dĺžka trvania pozície, percentuálnu úspešnosť stratégie, počet dosiahnutých ziskových a stratových jednotiek pohybu v pipoch, rozbor ziskovosti podľa obchodných hodín, dní, párov a podobne. Následne sme schopný výsledky interpretovať a vyvodiť vhodné kroky na dosiahnutie optimalizácie. Po úspešnom porovnaní, výbere vhodnej verzie a dokončení optimalizácie sme schopný porovnať algoritmus založený na identifikácií distribúcie s prvým, ktorý sa opiera o kríženie EMA indikátorov a vyvodiť záver, ktorý z nich dosahuje lepšie výsledky v obchodovaní na forex trhu.

3 Výsledky práce a diskusia

V tejto časti bakalárskej práce sa podrobnejšie pozrieme na spread a konkrétne na to, ako vplýva na ziskovosť naprogramovanej stratégie. Nakoľko na výšku spreadu vplývajú rôzne faktory ako výber brokera alebo momentálny čas, v ktorom vstupujeme do pozície, vybrali sme si priemernú výšku spreadu u brokera Oanda, ktorý aplikujeme vo veľkosti 0.00015 jednotiek ceny pre krátke – short objednávky, resp. pri vstupe do predajnej pozície.

Kapitola výsledky práce sa delí na päť častí, ktoré odkazujú na zdrojové kódy v prílohách. V prvej časti si predstavíme funkcionality prvého algoritmu s vizuálnymi ukážkami, jeho hlavné moduly a popis zdrojového kódu vytvoreného vo vývojárskom prostredí PyCharm. Druhá časť sa zaoberá spracovaním štatistických dát a vyhodnocovaním úspešnosti. Predmetom tretej kapitoly je naprogramovanie druhého, zložitejšieho algoritmu na obchodovanie distribúcií a taktiež si popíšeme jeho funkcionality s názornými ukážkami spolu s backtestingovým a štatistickým modulom. Štvrtá kapitola obsahuje porovnanie výsledkov rôznych verzií, vrátane optimalizovanej, ktorá bude zároveň vhodným adeptom na záverečné porovnanie s prvým algoritmom. V záverečnej – piatej časti sa nachádza finálne porovnanie oboch algoritmov a výber najlepšej stratégie vzhľadom na dosiahnuté hodnoty hlavných ukazovateľov ziskovosti. Naprieč všetkými časťami sú postupne vymedzené najväčšie výzvy, vrátane riešení s ktorými sa obchodník môže stretnúť pri vytváraní obdobných automatizovaných obchodných systémov.

3.1 Algoritmus zameraný na kríženie hodnôt krátkej a dlhej periódy indikátoru EMA

Kostrou tejto obchodnej stratégie je využitie exponenciálnych kĺzavých priemerov EMA s periódami 13 a 50 jednotiek z časového rámca a ich vzájomné kríženie. Podmienky pre vstup tvorí prekríženie (cross) krátkej a dlhej periódy a následná voľba vstupu do krátkej alebo dlhej pozície podľa bezprostredne predošlých hodnôt týchto priemerov. V prípade, že ide o prekríženie a predošlá hodnota EMA 13 bola vyššia ako EMA 50, ide o vstup do krátkej pozície a naopak pri predošlej hodnote EMA 50, ktorá bola vyššia ako EMA 13 ide o dlhú pozíciu a predpokladáme nárast ceny v krátkom časovom horizonte. Tieto vstupy nie sú orientované na dlhodobé držanie pozície pretože dáta z minútového časového rámca nemajú dostatočnú analytickú hodnotu pre vyhodnocovanie dlhodobej zmeny trendu. Vzťah je znázornený na Obrázku č. 3

Obrázok č. 3 Kríženie EMA 13 a EMA 50

Zdroj: Vlastné spracovanie

Ide o pomerne jednoduchý obchodný koncept, avšak účelom je využitie malých, častých a konzistentných obchodov pri nízkej miere rizika podstupovaného pri jednotlivých obchodoch a tým je docielená nenáročnosť na pomer výnosu a rizika stratégie. To znamená, že aj pri častých stratových pozíciách chceme docieliť kladné zhodnotenie obchodného účtu za predpokladu kladného celkového pomeru ziskových a stratových obchodov. Za hlavnú prekážku je možné považovať výšku spreadu, čo je rozdiel medzi bid (predajnou) a ask (nákupnou) cenou, ktorú poskytuje broker. Ďalším negatívnym faktorom pri častých obchodoch s malou investíciou je komisia, ktorú sme povinný platiť pri každej otvorenej pozícií. Tieto dva faktory majú najviac negatívny vplyv na popisovanú obchodnú stratégiu.

Jadro programu sa nazýva zdrojový kód v ktorom sú naprogramované kroky pomocou príkazov v akých má algoritmus postupovať. Syntax znamená v terminológií programovania úroveň náročnosti čítania a zrozumiteľnosti zdrojového kódu. Python disponuje veľmi zrozumiteľným a jasným syntaxom, čo ho stavia do silnej pozície oproti iným programovacím jazykom a to je aj dôvod prečo sa stáva vhodnou voľbou. Knižnice sú balíky funkcií vytvorené komunitou programátorov a je možné ich implementovať do zdrojového kódu. Prostredníctvom knižnice Pandas je možné pracovať s číselnými dátami vo forme tabuľky, ktorá sa dá následne zapísať do Excel súboru a je dostupná k ďalšiemu spracovaniu či už v ďalšej reprodukcií kódu alebo prostredníctvom samotného Excelu. Ďalšou veľmi

užitočnou knižnicou pre Python je Ta-lib. Obsahuje množstvo funkcií, indikátorov a nástrojov pre prácu s forexovými dátami. Nachádzajú sa v nej príkazy, ktoré slúžia na využitie nástrojov technickej analýzy, ktoré sú dostupné pre príklad aj v platforme Meta Trader 4.

Zdrojový kód algoritmu je rozdelený do dvoch modulov. Úloha prvého pasívneho modulu je zabezpečiť pripojenie na poskytovateľa dát prostredníctvom brány, tzv. API a využíva k tomu identifikačné číslo a prístupový kľúč účtu. Následne si vyžiada dáta z trhu podľa predom nastavených parametrov, v ktorých je možné zvoliť požadované menové páry, požadovaný počet a dĺžku trvania (periodicitu) jednotlivých sviečok. Zo sviečok extrahuje údaje otváracích, najvyšších, najnižších a zatváracích cien a pomocou knižnice Pandas ich zapíše do premennej. Po tom ako máme požadované údaje vytriedené, dochádza k identifikácií swing high a swing low bodov prostredníctvom porovnávania cien. Tieto údaje slúžia na identifikáciu limitných stop loss a take profit možností v prípade vstupu do pozície. Program taktiež vyhodnotí veľkosť sviečky v pipoch (jednotka pohybu ceny) a či je rastúca, klesajúca alebo neutrálna. Následne pristúpi k využitiu Ta-lib knižnice a pomocou nástroja EMA vykalkuluje kĺzavé priemery pre dané ceny v periódach 13 a 50 sviečok podľa zadaného parametra. V záverečnej časti sa porovnávajú jednotlivé priemery a program hľadá prekríženie ich hodnôt, ak prekríženie nastalo, získa potrebné dáta na zápis podmienok pre možný vstup do pozície. Tieto údaje obsahujú:

- stop loss a take profit ceny na základe predošlých swing high a swing low úrovní
- typ možnej objednávky (short alebo long pozícia) na základe predošlých hodnôt EMA
- menový pár

Proces sa zopakuje pre všetky menové páry ktoré sme zvolili na začiatku a nájdené vstupy sa zapíšu k prislúchajúcej sviečke na ktorej nastali. Pokiaľ nedôjde ku problému, modul by sa mal úspešne ukončiť a uložiť dáta vo formáte xlsx (Excel súbor). Popisovaný zdrojový kód prvého modulu je dostupný v prílohe č. 1 spoločne s čiastočnými komentármi jednotlivých častí.

Druhý modul je aktívny a na svoje fungovanie potrebuje dáta v podobe Excel súboru. Po jeho spustení sa aktivuje funkcia, ktorá vyvolá prvý modul, zabezpečí potrebné dáta a aktivuje časovač. Program počká, kým budú zanalyzované dáta dostupné na použitie. Pre efektívne fungovanie algoritmu potrebujeme aktuálne dáta z trhu a preto je nutné aby bol Excel súbor aktualizovaný každú minútu v prípade obchodovania na minútovom časovom rámci, respektíve každých päť minút pre obchodovanie na päť minútovom rámci. V prípade,

že je nájdená dostupná pozícia na poslednej aktuálnej sviečke, vyextrahuje potrebné dáta na vstup do pozície a odošle ich do funkcie, ktorá slúži na vytvorenie pozície na trhu s parametrami zo signálu prostredníctvom brány API u brokera. Táto pozícia sa uloží do dočasného súboru, ktorý v prípade identifikácie signálu na rovnakom menovom páre v krátkom časovom úseku od prvotnej pozície zabráni vytvoreniu zdvojenej objednávky. Dočasný súbor sa po uplynutí 10 minút vymaže aby sa predišlo neželanému zabráneniu legitímnej pozície. Keď je objednávka úspešne vytvorená, program pokračuje ďalej bez zastavenia až kým ho užívateľ neukončí. Zdrojový kód druhého modulu je dostupný v prílohe č. 2 s detailným popisom jednotlivých častí.

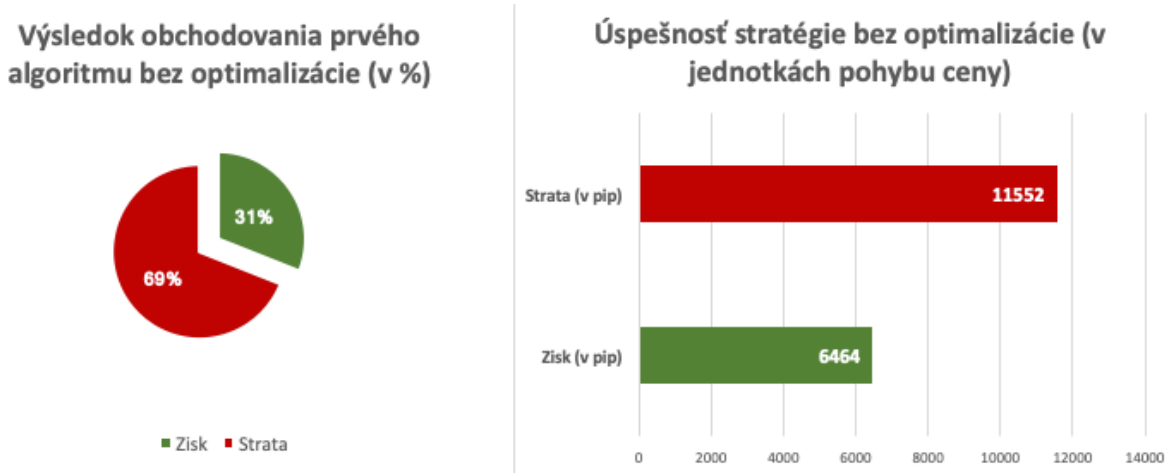
Po úspešnom uzavretí viac ako 200 pozícií vytvoríme excel zošit vo formáte .xlsx, ktorý obsahuje obchodnú históriu a následne môže byť spracovaná v štatistickom module. Obsahuje naprogramované vzorce a operácie pre výpočet požadovaných ukazovateľov pre prvý algoritmus z kapitoly metodika práce a metódy skúmania. Pri programovaní týchto funkcií sa využíva formátovanie hodnôt a funkcie prostredníctvom knižnice Pandas. Jednotlivé vstupy vo forme riadkov z obchodnej histórie sú rozdelené štrukturálne do stĺpcov, na ktorých vieme uplatniť štatistické metódy ako napríklad výpočet priemeru z hodnôt, filtrovanie dát, viacúrovňové zoskupovanie na základe zadaného parametru, časové triedenie pomocou Datetime knižnice a podobne. Výstupy zo štatistického modulu sú po spracovaní uložené do preddefinovaných súborov či už vo formáte .txt alebo .xlsx a môžu byť graficky spracované, porovnávané a interpretované. Štatistický modul je dostupný v prílohe č.3. [11] [12] [13] [14] [15] [16]

[11] www.buildmedia.readthedocs.org, Oanda V-20 Rest API official documentation [online]. Dostupné na: https://buildmedia.readthedocs.org/media/pdf/oanda-api-v20/latest/oanda-api-v20.pdf
[12] www.pandas.pydata.org, Pandas library official documentation [online]. Dostupné na: https://pandas.pydata.org/pandas-docs/stable/#
[13] wwwmrjbq7.github.io, TA-Lib library official documentation [online]. Dostupné na: https://pandas.pydata.org/pandas-docs/stable/#
[14] www.docs.python.org, Datetime library official documentation [online]. Dostupné na: https://docs.python.org/3/library/datetime.html
[15] www.xlrd.readthedocs.io, xlrd library official documentation [online]. Dostupné na: https://xlrd.readthedocs.io/en/latest/
[16] Swigart A., 2015, Automate the Boring stuff with Python, San Francisco: No Starch Press, 978-1-59327-599-0, 481s

3.2 Vyhodnotenie a optimalizácia algoritmu založeného na krížení hodnôt EMA

Štatistická vzorka obsahuje 243 obchodov, ktoré sa uskutočnili na všetkých menových pároch, počas všetkých hodín s výnimkou víkendov, kedy je forexový trh uzatvorený a nie je možné zadávať pozície. Prvotné výsledky bez optimalizácie nie sú uspokojivé a pomer ziskových ku stratovým obchodom je 31 % (76) ku 69% (169) s priemerným RRR na úrovni 1,23:1 a v celkovom vyjadrení v jednotkách pohybu ceny strata prevyšuje zisk o 5088 pipov, teda zisk v pipoch sa musí zvýšiť o 78,7 % aby sa vyrovnal strate. Viď. graf č. 1.

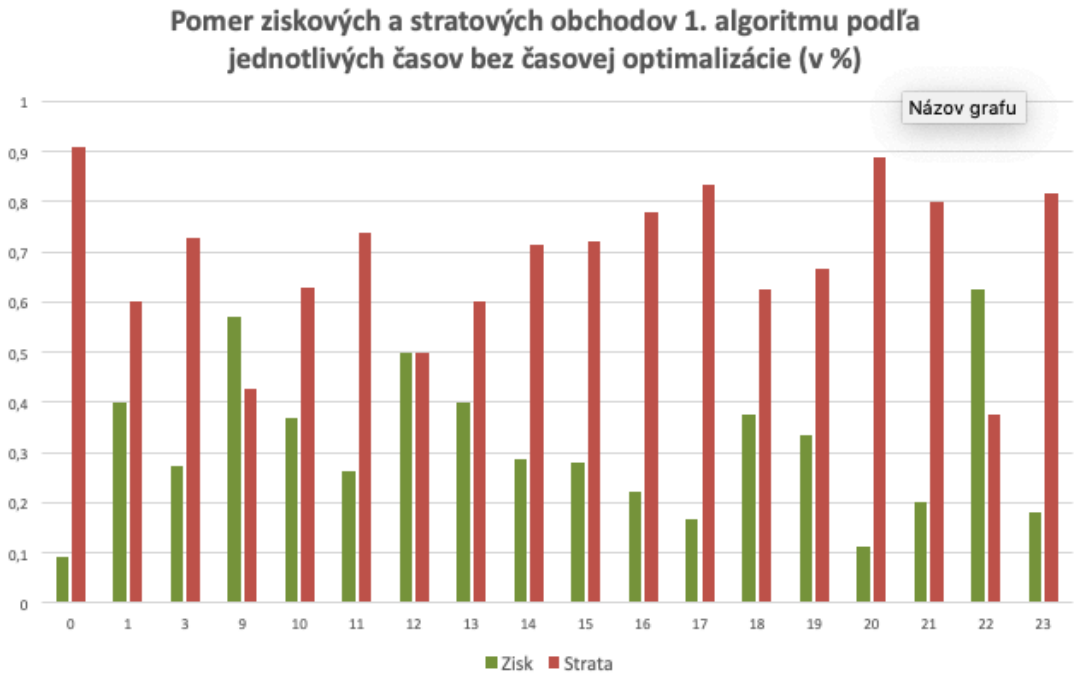
Graf č. 1



Zdroj: Vlastné spracovanie

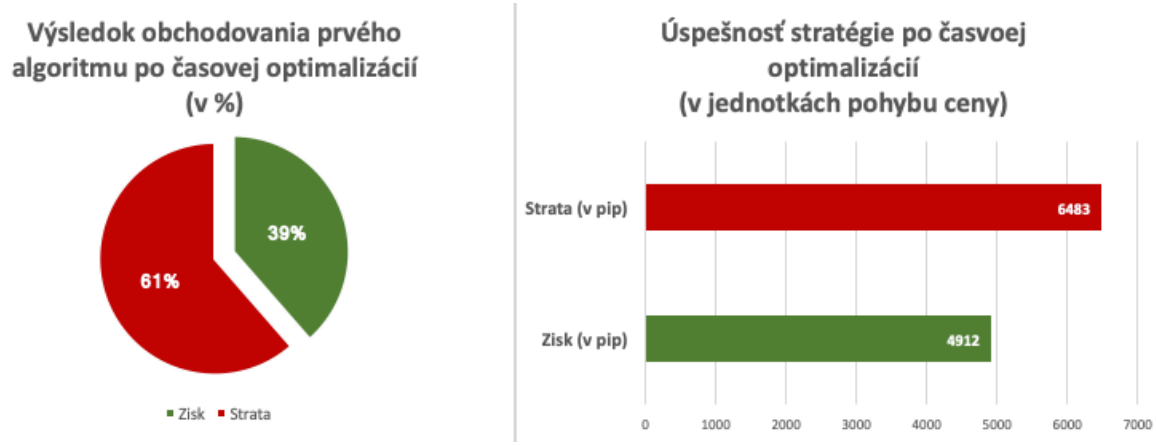
Následnou analýzou obchodných časov sa vieme dopracovať k identifikácií najmenej ziskových časov. Tieto časy sa vzťahujú na vstup do pozícií. V prípade prvého algoritmu sú najmenej profitabilné časy v hodinách 00:00, 15:00, 16:00, 17:00, 20:00, 23:00. Časy 11:00 a 14:00 sú blízko otváracím dobám pre Európu a USA a preto ich zahrnieme do obchodných aj po optimalizácií a ostatné najmenej ziskové vylúčime. Viď obrázok 1 Pozitívnym výsledkom je fakt, že pomocou tejto metódy sme zvýšili pomer ziskových k stratovým pozíciám z 31% na 39% pri 1,21:1 RRR a v celkovom vyjadrení v jednotkách pohybu ceny zisk zaostáva o 1517 pipov. Na dorovnanie straty potrebujeme zvýšiť zisk o 32% v porovnaní s predošlými 78,7%. Jednou veľkou slabou stránkou je zníženie počtu obchodov z pôvodných 243 na 153. Viď. graf č. 2. a graf č. 3.

Graf č. 2



Zdroj: Vlastné spracovanie

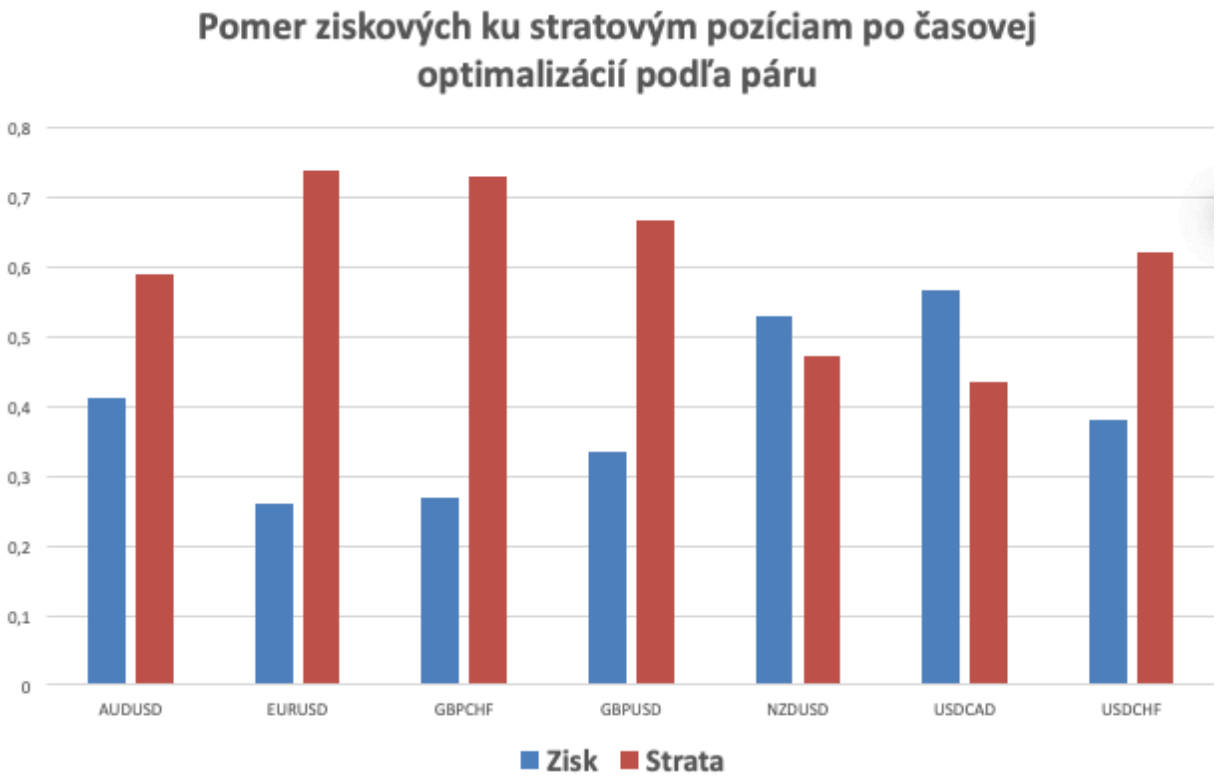
Graf č. 3



Zdroj: Vlastné spracovanie

Po časovej pristúpime k optimalizácií podľa menových párov nakoľko algoritmus sa správa rozdielne v závislosti od obchodovaného páru. Je nutné podotknúť, že štatistická vzorka sa značne znížila z 243 obchodov na 153. To prikladá ďalším optimalizáciám nižšiu štatistickú hodnotu. Z analýzy dát môžeme pozorovať výrazné rozdiely v ziskovosti na základe zvoleného menového páru. Najlepšie výsledky dosahuje algoritmus na USD/CAD a NZD/USD. Z týchto dôvodov ich zahrnieme do ďalšej verzie. Viď graf č. 4.

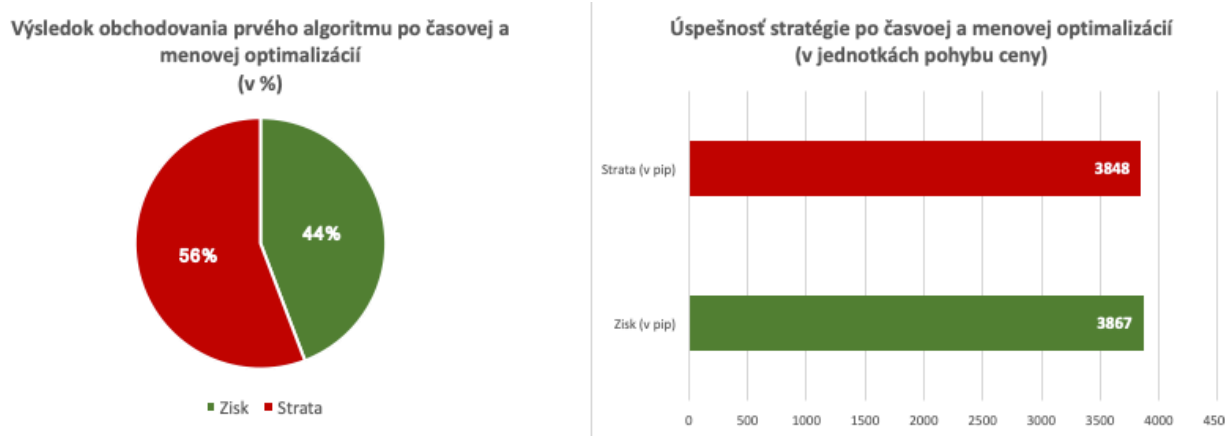
Graf č. 4



Zdroj: Vlastné spracovanie

Finálna verzia 1. algoritmu dosahuje po filtrovaní neželaných časov a menových párov úspešnosť 55% pri RRR 1,20:1 a celkový počet dosiahnutých pipov je 1806 v zisku proti 1177 v strate. Výsledok je teda pozitívny, avšak treba vziať do úvahy radikálne zníženie štatistickej vzorky a to na 40 obchodov. Z tohto dôvodu nie je možné jednoznačne tvrdiť ani predpokladať, že algoritmus sa bude obdobne správať aj na inej, väčšej, vzorke obchodnej histórie. Následne môžeme vyradiť menové páry EURUSD a GBPCHF nakoľko ponúkajú najnižší potenciál pre dosahovanie zisku spomedzi ostatných menových párov. Štatistická vzorka v tomto prípade je 104 obchodov, čo je relatívne dostatočné množstvo. Finálne výsledky 1. algoritmu sú pomer ziskových ku stratovým pozíciám 44% pri RRR 1,18:1 a celkový počet dosiahnutých pipov zisku je o 19 jednotiek vyšší ako straty. V prípade prvej verzie bolo potrebné navýšiť zisk o 78,7% na dosiahnutie vyrovnaného výsledku zisku a straty. Viď obrázok graf č. 5.

Graf č. 5

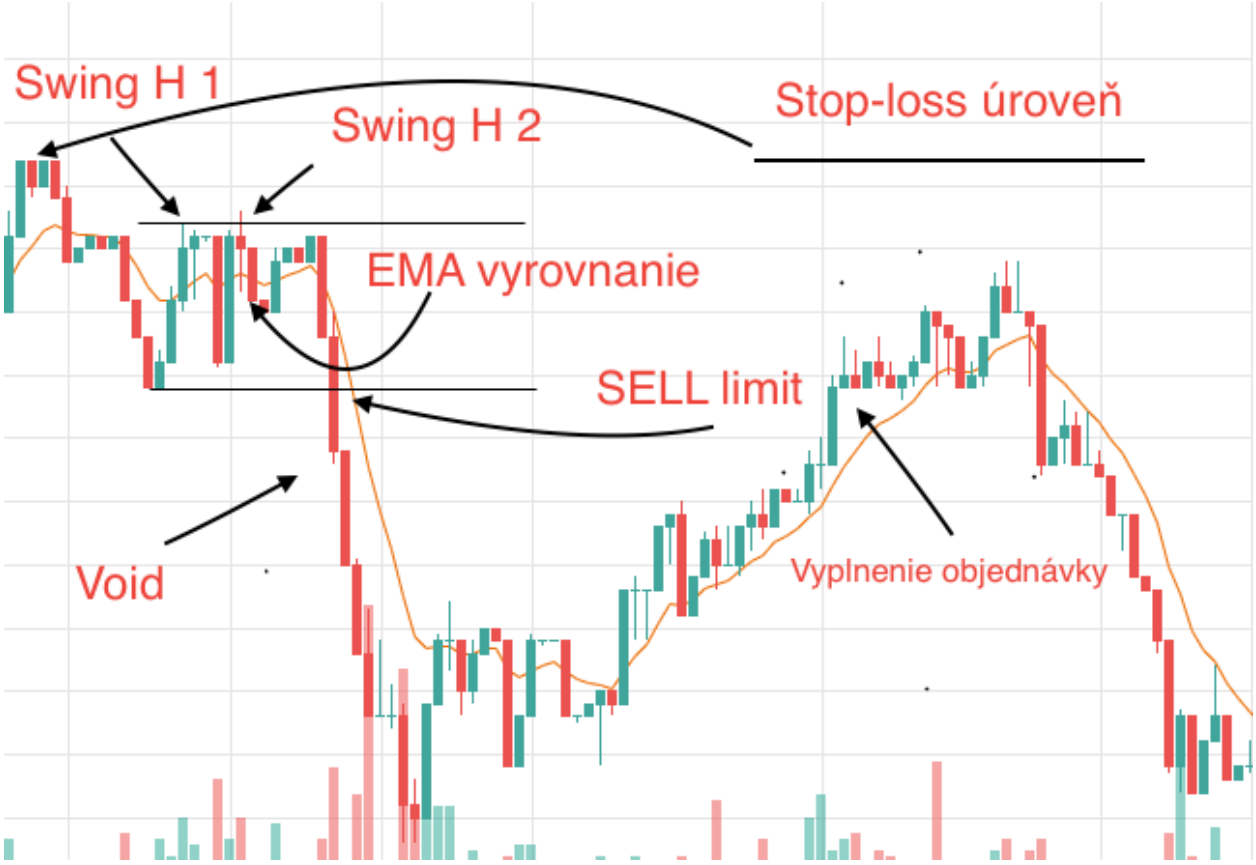


Zdroj: Vlastné spracovanie

3.3 Algoritmus zameraný na identifikáciu a obchodovanie prienikov z distribučných zón

V poradí druhá obchodná stratégia implementovaná vo forme automatizovaného obchodného systému využíva niektoré už uvedené koncepty. Miesto analýzy kríženia dvoch EMA hodnôt s kratšou a dlhšou periódou ako v minulej stratégii však pri tomto algoritme využijeme len jedno hodnotu EMA s periódou 10 sviečok. Táto perióda určuje ako rýchlo dokáže algoritmus reagovať na vyrovnávanie takéhoto kĺzavého priemeru nakoľko nižšia perióda odzrkadlí zmenu trendu, respektíve pohyb do strany s menším omeškaním ako kým sa vyrovná EMA s dlhšou periódou. Swing high a swing low využijeme okrem určovania stop-loss a take-profit objednávok taktiež k identifikácií hraníc pre distribučné pásmo a teda pre výpočet vrcholu a dna, na ktorých podľa stratégie chceme umiestňovať vstupy do long a short pozícií. Viď. obrázok č. 4.

Obrázok č. 4 Proces identifikácie limitnej objednávky



Zdroj: Vlastné spracovanie

Algoritmus sa skladá primárne z troch modulov pre obchodovanie v reálnom čase, jedného samostatného modulu, ktorý slúži na backtesting a optimalizáciu a štatistického modulu, pomocou ktorého je možné spracovať, formátovať a vykonávať matematické operácie na vzorke obchodnej histórie za účelom získania dát vhodných pre interpretáciu úspešnosti a výsledkov zmien nastavení. V dôsledku komplexnosti riešenia sú využívané Excel zošity formátu .xlsx pre presun dát medzi jednotlivými celkami.

Prvou časťou je pasívny script, ktorý v úvode vykoná obdobné operácie ako v prvom algoritme. Po overení účtu u brokera na API bráne získa dáta potrebné pri ďalšom formátovaní. Využitím modulu ExcelWriter z knižnice Pandas spracuje a vytvorí pre preddefinované menové páry samostatné stĺpce v DataFrame objekte ako napríklad OHLC ceny, čas sviečky, typ vo forme rastúca/klesajúca a označí či ide o úroveň swing high prípadne swing low na základe kalkulácie. Každý menový pár uloží do inej strany v Excel zošite. Všetky číselné a textové dáta sú uložené vo formáte ‘string’ a preto je pre niektoré číselné stĺpce potrebné využiť formátovanie na typ ‘float‘ a zaokrúhliť matematicky na 5 desatinných miest.

Pre identifikáciu distribučných zón sa využíva metóda porovnávania swing high a swing low úrovní. Pokiaľ je na aktuálnej sviečke zaznamenaná swing high cena, zapíše sa do premennej vo forme prvku v ‘list‘ objekte, následne sa kontroluje predošlá úroveň swing high a pomocou fixne zadaného rozptylu 0.00009 pipov algoritmus overí, či ide o rovnakú úroveň. Pri situácií kedy máme dve po sebe nasledujúce swing high úrovne na rovnakých cenách v úzkom časovom rozptyle script túto informáciu zaznamená a pristúpi k porovnávaniu x EMA hodnôt v tomto rozpätí. Pokiaľ je zmena hodnôt EMA minimálna, respektíve nulová, dochádza k splneniu druhej podmienky pre identifikáciu distribučného pásma kde úroveň swing high sa považuje za stropnú cenu a swing low úroveň naopak za dno pásma. Knižnica Pandas využíva na indexovanie metódy .loc, .iloc a procesný pohyb dát je zabezpečený objektmi typu Series, čo je jeden stĺpec objektu DataFrame. Pre iteráciu sa využíva slučka for a je vhodné využiť ukladanie dát do ‘generator‘ objektu, ktorý je následne spracovaný pomocou iterácie. Takýto prístup umožňuje programu pracovať bez ukladania všetkých hodnôt v pamäti a ponúka výrazné zrýchlenie kódu.

Po úspešnej identifikácií distribučného pásma, jeho stropu a dna program vytvorí nový stĺpec s údajom o priemernej veľkosti 10 predchádzajúcich sviečok. Tieto hodnoty sa využijú pri výpočte, respektíve identifikácií tzv. void pohybov. Void pohyb predstavuje explozívne opustenie distribučného pásma a na jeho overenie slúžia podmienky splnenia fixného násobku priemernej veľkosti distribučného pásma spolu s overením, že je close cena void sviečky mimo distribučného pásma. Identifikácia void pohybu je umiestnená v samostatnej funkcii. Niektoré komponenty, ktoré si vyžadujú zmenu určitých parametrov sú programované vo forme funkcií pre ich jednoduchšiu manipuláciu v prípade implementácie určitých zmien ako napríklad zmena násobiteľa, počtu predošlých sviečok, ktoré slúžia na overenie podmienok a podobne. Vďaka funkčným parametrom je možné zmeniť napríklad veľkosť zadávaných objednávok alebo časový rámec pre požadované dáta bez zásahu do zdrojového kódu jednoduchou zmenou parametra pri vyvolaní funkcie.

Ďalším krokom po vyhodnotení distribučného pásma a prieniku ceny vo forme voidu program na základe predošlých swing high a swing low cien stanoví take-profit a stop-loss ceny a spoločne ich zaznamená. Údaje ako čas identifikácie signálu, index sviečky na ktorej sa signál vyskytol, cenu pre zadanie limitnej objednávky, take-profit a stop-loss cenu sa uložia do zošitu df_hotovy.xlsx v novom stĺpci do riadku kde sa nachádza vstupná cena a modul čaká na opätovnú žiadosť o dáta. Celá popisovaná funkcionalita je súčasťou funkcie a modul obsahuje taktiež druhú pomocnú funkciu na vymazanie všetkých aktívnych

objednávok cez API bránu, táto je využívaná len manuálne a nie je súčasťou automatizovanej stratégie. Zdrojový kód k 1. pasívnemu modulu je dostupný v prílohe č. 4.

Prvým spustiteľným scriptom je modul na aktualizáciu dát. Komunikuje s prvým pasívnym analytickým modulom a v minútovej periodicite si žiada dáta vo forme df_hotovy.xlsx Excel zošitu. Následne dochádza k identifikácií riadkov so signálmi, spracovaniu dát, formátovaniu a rozčleneniu. Vo funkcií cistic() dochádza k prečisteniu duplicitných signálov. Ďalej sa dáta uložia do ďalšieho zošitu ako signals.xlsx a sú dostupné k použitiu. Po týchto operáciách modul čaká v prípade 1M časového rámca zvyšný počet sekúnd do novej minúty a proces sa opakuje pre zabezpečenie aktuálnych signálov v reálnom čase. Zdrojový kód je dostupný v prílohe č.5

Simultánne s aktualizáciou dát je potrebné spustiť tretí – objednávkový modul. Jeho úlohou je prijať výstup v podobe signals.xlsx a takzvaným ‘parsovaním’ priradiť hodnoty zo signálu do premenných, ktoré môžu byť následne použité v tele objednávky. Tento proces je zabezpečený ukladaním dát priebežne do objektu generátora za účelom zrýchlenia procesu. V prípade nevhodnej priradenej hodnoty sa takáto hodnota upraví. Tento jav sa využíva pri nízkych stop-loss, respektíve take-profit hodnotách a zabezpečuje, aby pozícia prebiehala vo väčšom cenovom rozpätí nakoľko pri vysokej volatilite na trhu často dochádza ku uzatvorení objednávky na nízko položenej stop-loss limitnej objednávke. Ihneď po úprave tela objednávky sa odošle požiadavka na zadanie limitnej pozície a v prípade, že sa cena dostane na túto úroveň, vstúpime automaticky na trh. Pre zamedzenie zdvojených objednávok je čakajúca limitná pozícia uložená do premennej typu ‘list’ a pri spracovaní ďalšieho signálu sa vstupná cena signálu porovná s existujúcimi čakajúcimi objednávkami. Pokiaľ je výsledkom porovnania zhoda, pozícia sa zahodí a algoritmus pokračuje na ďalší signál. Súčasne sa aktualizujú dáta vďaka druhému modulu a vždy keď dôjde k pridaniu nového signálu do zošita signals.xlsx, dochádza k okamžitému spracovaniu v objednávkovom module. Zdrojový kód objednávkového modulu je dostupný v prílohe č. 6

Okrem schopnosti obchodovania v reálnom čase pomocou obchodnej verzie však potrebujeme samostatný modul na testovanie vyhodnotených pozícií na historických cenách. Backtesting modul je samostatný script, ktorý danú funkcionalitu zabezpečuje. Na zber a vyhodnocovanie pozícií používa rovnaký zdrojový kód ako prvý modul obchodnej verzie a signály uložené v príslušnom riadku ukladá do . zošita df_backtest_feed.xlsx. Na rozdiel od druhého modulu nepotrebuje zbierať signály v aktuálnom čase a preto využíva funkciu vytvor_signal_excel_file(). Funkcia vyhodnotí a uloží signály do ďalšieho zošita. Následne

pomocou funkcie itteruj(), ktorá využíva generátor spracuje signál a pomocou funkcie vyhodnotenie_signalu overí, či došlo ku vyplneniu stop-loss alebo take-profit ceny a teda či bola pozícia zisková, alebo stratová. Funkcia vyhodnotenie_signalu pracuje s parametrami data_df, signal_type, signal_price, signal_sl, signal_tp, signal_index, singal_identified_index a sheet_name. Pomocou daných parametrov docielime schopnosť vykonať operáciu na rôznych hodnotách a veľkých dátových setoch. Signály sú porovnávané s historickým vývojom v čase kedy nastali. Algoritmus overí, či v prípade long objednávky došlo po vyplnení vstupu k situácií, kedy bol high z OHLC dát pre každú nasledujúcu sviečku vyšší ako take-profit cena, alebo či skôr došlo k situácií kedy bola low cena z OHLC dát pre každú nasledujúcu sviečku nižšia ako stop-loss cena. Pri short objednávkach funguje proces antagonisticky. Po ukončení pozície sa výsledok uloží do premennej vo funkcií itteruj() do riadku objektu Series. Po vyhodnotení všetkých identifikovaných signálov sa výstup uloží v podobe obchodnej histórie spolu s výsledkom pozície. Zdrojový kód backtesting modulu je dostupný v prílohe č. 7

Piaty – záverečný modul vyžaduje výstup z backtesting scriptu vo forme obchodnej histórie formátu .xlsx. Pomocou zadefinovaných operácií vyhodnotí štatistické ukazovatele, ktoré budú následne použité pri optimalizácií a na porovnanie s prvým algoritmom. Script pracuje nezávisle od časového rámca, menových párov, zmien v stop-loss a take-profit cenách a je plne dynamický, čo zaručuje jednoduchý prístup k okamžitým výsledkom po zmene parametrov v backtesting module. Hlavnou motiváciou pre vytvorenie programovateľnej kalkulácie ukazovateľov úspešnosti bola praktickosť a časová úspora pri porovnávaní jednotlivých verzií pri optimalizácií. Modul využíva niektoré jednoduché funkcie zo základného balíka Python knižníc ako napríklad sum() pre súčet, count() pre počet a mean() ako stredovú hodnotu z deskriptívnej štatistiky. Z knižnice pandas sú použité metódy na filtrovanie a viacúrovňové zoskupovanie ako .groupby(), .loc, .iloc a podobne. Časová analýza a formátovanie je vykonané pomocou datetime knižnice a jeho metód ako

.date(), .day() a v neposlednom rade .hour(). Zdrojový kód štatistického modulu je dostupný v prílohe č. 8. [17] [18] [19] [20] [21] [22]

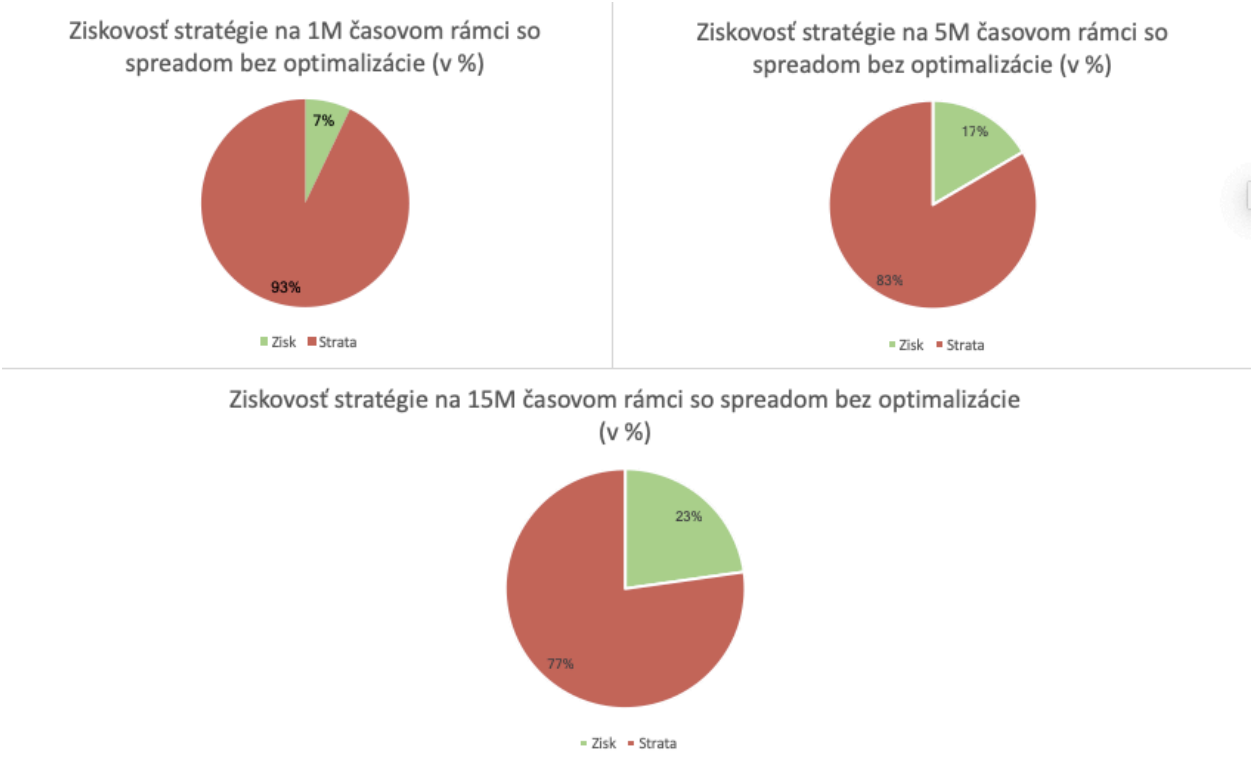
3.4 Vyhodnotenie a optimalizácia algoritmu zameraného na identifikáciu a obchodovanie prienikov z distribučných zón

V prvom kole testovania bolo potrebné získať časový rámec, na ktorom algoritmus dosahoval najlepšie výsledky. Do testovania boli zahrnuté tri verzie a to konkrétne 1M, 5M a 15M. Všetky verzie obchodovali v rovnakých podmienkach. Do testovania bolo zahrnutých 5000 posledných sviečok, respektíve jednotiek časového rámca so zarátaným spreadom. Take-profit násobiteľ na hodnote 3 a stop-loss upravený o 0,00010 jednotiek pohybu. Na základe výsledkov môžeme 15 minútový časový rámec považovať za optimálnu voľbu pri ďalšom bádaní a optimalizácií.

Na základe údajov z grafu č. 6 je možné pozorovať úspešnosť jednotlivých verzií. Najlepšie skončila verzia obchodujúca na 15 minútovom časovom rámci a to s úspešnosťou 23% oproti 17% pri 5M, respektíve 7% pri 1M verzií.

[17] www.buildmedia.readthedocs.org, Oanda V-20 Rest API official documentation [online]. Dostupné na: https://buildmedia.readthedocs.org/media/pdf/oanda-api-v20/latest/oanda-api-v20.pdf
[18] www.pandas.pydata.org, Pandas library official documentation [online]. Dostupné na: https://pandas.pydata.org/pandas-docs/stable/#
[19] wwwmrjbq7.github.io, TA-Lib library official documentation [online]. Dostupné na: https://pandas.pydata.org/pandas-docs/stable/#
[20] www.docs.python.org, Datetime library official documentation [online]. Dostupné na: https://docs.python.org/3/library/datetime.html
[21] www.xlrd.readthedocs.io, xlrd library official documentation [online]. Dostupné na: https://xlrd.readthedocs.io/en/latest/
[22] Swigart A., 2015, Automate the Boring stuff with Python, San Francisco: No Starch Press, 978-1-59327-599-0, 481s

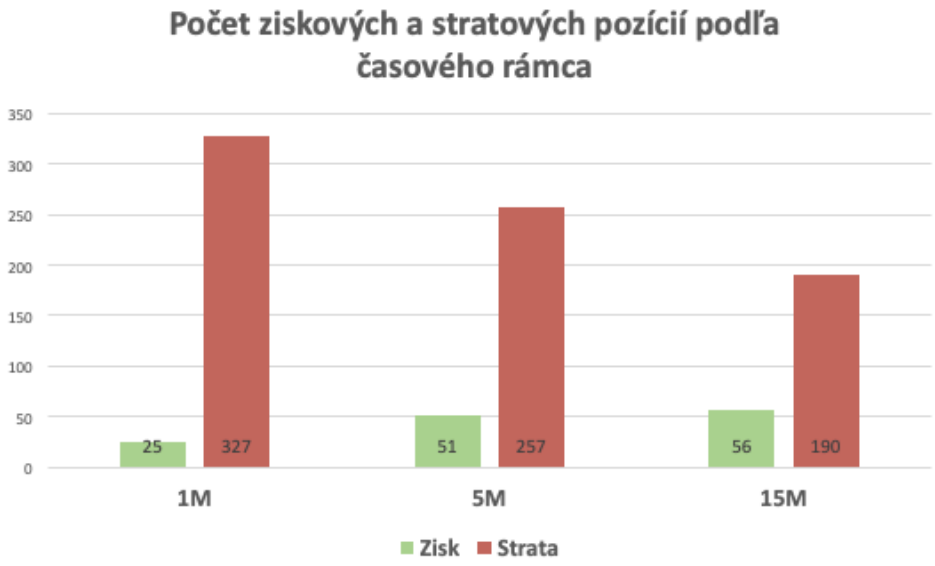
Graf č. 6



Zdroj: Vlastné spracovanie

Najväčší počet ziskových pozícií sa vyskytuje opäť na 15M časovom rámci s hodnotou 56 voči 190 stratovým a ide o najväčšiu hodnotu spomedzi všetkých verzií v porovnaní s 51 pre 5M a 25 pri 1M rámci. Štatistická vzorka sa pohybuje od 246 pri 15M až po 352 obchodov pri 1M časovom rámci. Viď graf č. 7.

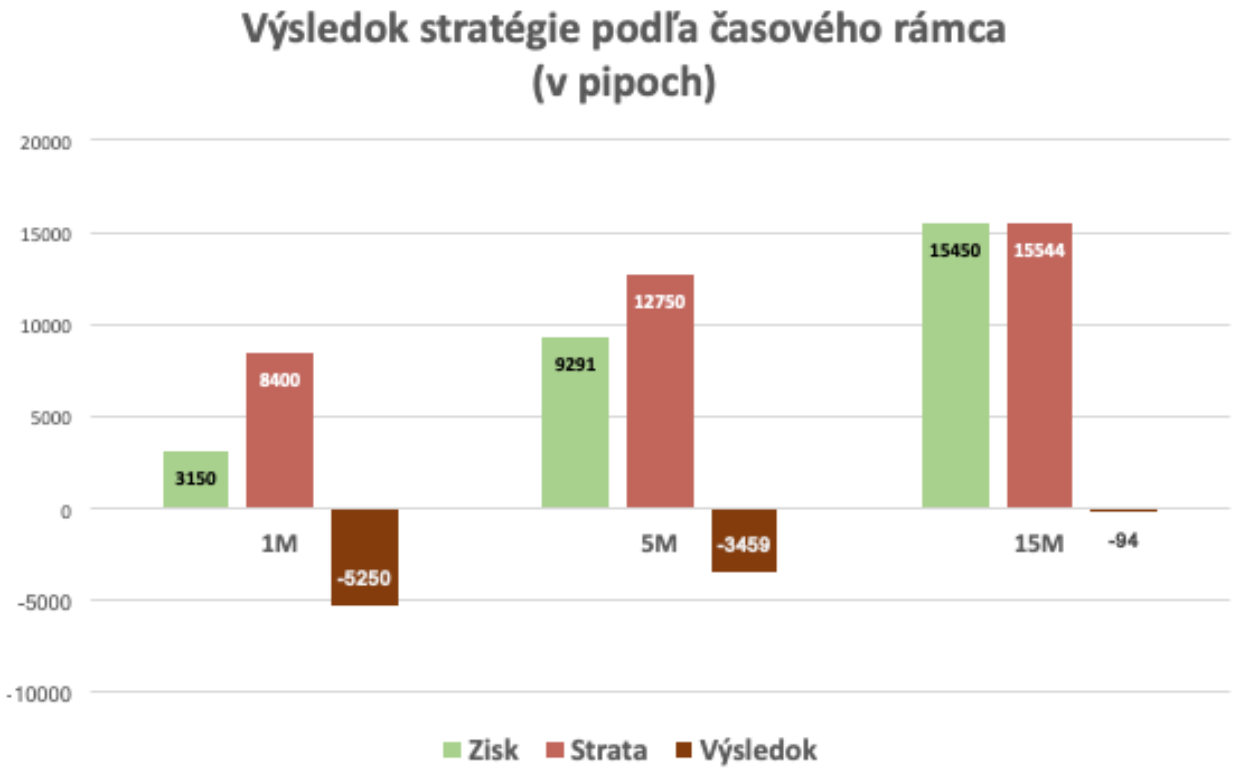
Graf č. 7



Zdroj: Vlastné spracovanie

V absolútnom vyjadrení v jednotkách pohybu pre ziskové a stratové pozície vedie znovu 15M verzia. Množstvo pipov sa navyšuje podľa dĺžky časového rámca a to z dôvodu väčších stop-loss a take-profit úrovní v absulútnom vyjadrení. 15M verzia dosiahla takmer vyrovnaný stav ziskových a stratových pipov s rozdielom 94 pipov v prospech straty, čo je v porovnaní s 3459 pipovým rozdielom pre 5M a 5250 pipovým rozdielom pri 1M verzií veľmi dobrý výsledok bez časovej a menovej optimalizácie s udržaním nezmenenej štatistickej vzorky. Viď graf č. 8.

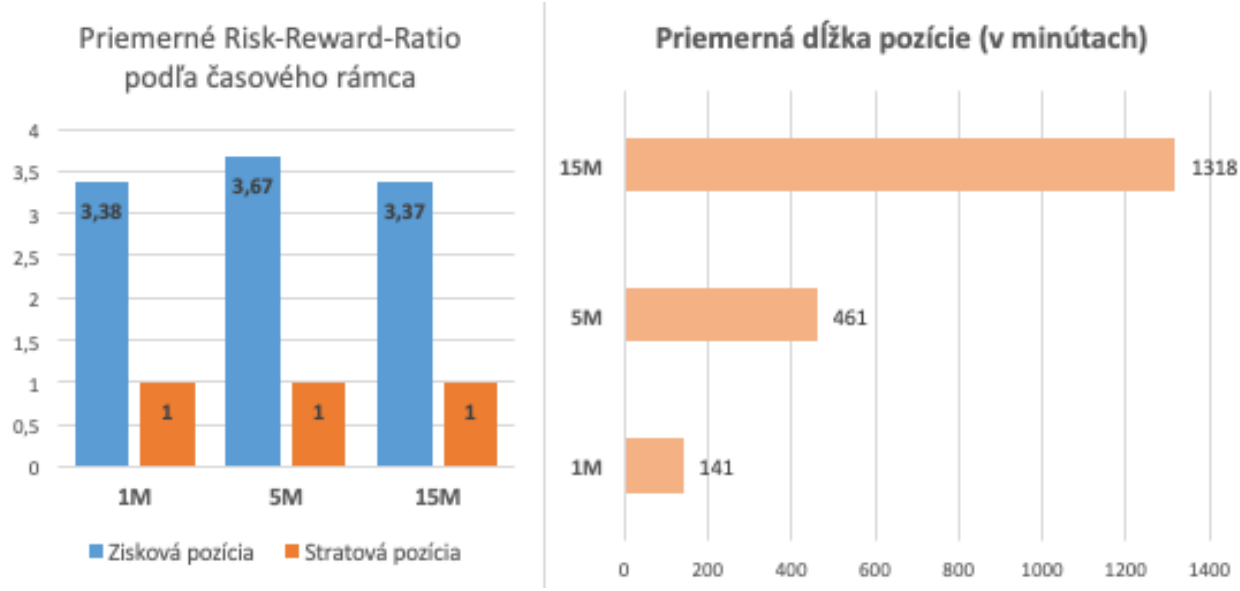
Graf č. 8



Zdroj: Vlastné spracovanie

Priemerná potenciálna výška zisku proti výške straty je hlavným dôvodom, prečo sú stratégie s nízkym pomerom ziskových obchodov schopné udržať si v prípade 15M verzie neutrálny absolútny výsledok vyjadrený v jednotkách pohybu ceny. Všetky testované verzie si udržali RRR v rozmedzí 3,37 až 3,67. Rozdielne výsledky však môžeme spozorovať pri priemernom trvaní pozície. V prípade 15M vezie je jedna pozícia v priemere otvorená po dobu 1318 minút, pri 5M verzií je táto hodnota 461 minút a najkratší interval môžeme pozorovať na grafe č. 9 pri 1M časovom rámci s hodnotou 141 minút.

Graf č. 9



Zdroj: Vlastné spracovanie

Pomocou grafu č. 10 je možné pozorovať, aký vplyv má spread na schopnosť algoritmu vykonávať ziskové obchody. Na rovnakej vzorke pri rovnakých pozíciách na 15M verzii je výsledok obchodovania algoritmu bez spreadu +2808 pipov v prospech zisku zatiaľ čo so započítaným spreadom je výsledok -94 pipov v prospech straty. Na základe údajov je možné preto tvrdiť, že spread by mal byť jedným z hlavných faktorov pri výbere brokera nakoľko priamo vplýva na ziskovosť stratégie či automatizovanej alebo manuálnej.

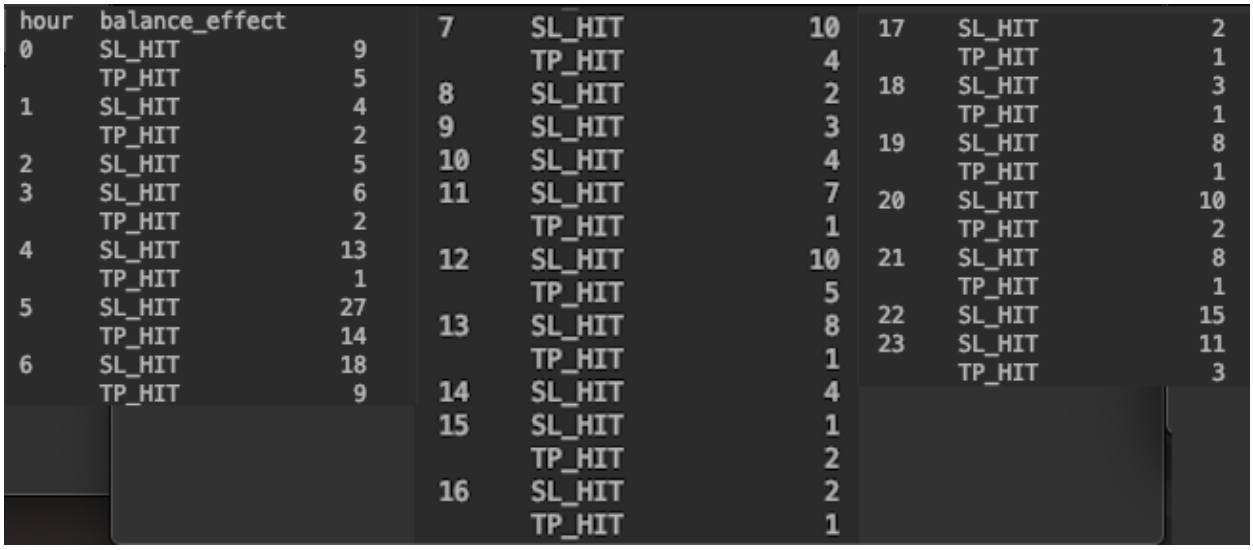
Graf č. 10

Zdroj: Vlastné spracovanie

Po zvolení optimálneho rámca je možné vylúčiť niektoré obchodné hodiny počas ktorých algoritmus dosahoval najhoršie výsledky. Vytvorením filtrovaného výberu

pomocou hodín a úspešnosti je možné vyvodiť záver. Obchodné hodiny, ktoré sa vylúčia sú 2:00, 8:00 – 10:00, 14:00 a 22:00 nakoľko počas nich algoritmus nedosiahol žiaden úspešný obchod. Viď záznamy z python konzoly na obrázku č. 5.

Obrázok č. 5 Záznam z Python konzoly

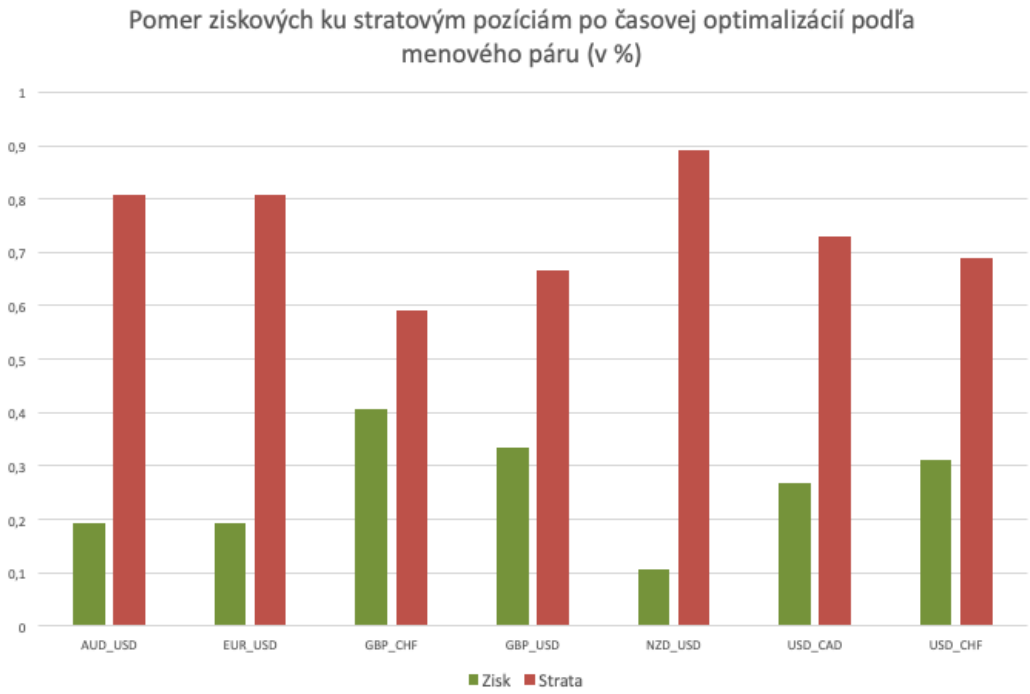


hour	balance_effect	
0	SL_HIT	9
	TP_HIT	5
1	SL_HIT	4
	TP_HIT	2
2	SL_HIT	5
3	SL_HIT	6
	TP_HIT	2
4	SL_HIT	13
	TP_HIT	1
5	SL_HIT	27
	TP_HIT	14
6	SL_HIT	18
	TP_HIT	9
7	SL_HIT	10
	TP_HIT	4
8	SL_HIT	2
9	SL_HIT	3
10	SL_HIT	4
11	SL_HIT	7
	TP_HIT	1
12	SL_HIT	10
	TP_HIT	5
13	SL_HIT	8
	TP_HIT	1
14	SL_HIT	4
15	SL_HIT	1
	TP_HIT	2
16	SL_HIT	2
	TP_HIT	1
17	SL_HIT	2
	TP_HIT	1
18	SL_HIT	3
	TP_HIT	1
19	SL_HIT	8
	TP_HIT	1
20	SL_HIT	10
	TP_HIT	2
21	SL_HIT	8
	TP_HIT	1
22	SL_HIT	15
23	SL_HIT	11
	TP_HIT	3

Zdroj: Vlastné spracovanie

Následne máme možnosť získať údaje o úspešnosti obchodovania podľa menového páru. Menový pár s najnižšou úspešnosťou vyplývajúc z grafu je NZD/USD a preto bude vhodné ho vynechať z obchodovania za predpokladu, že značne neovplyvní veľkosť štatistickej vzorky, ktorá je po časovej optimalizácií 213 obchodov. Odstránením sa vzorka zmenší o 28 jednotiek, čo je prijateľné vzhľadom na zmenu, ktorú optimalizácia prinesie. Viď. graf č. 11.

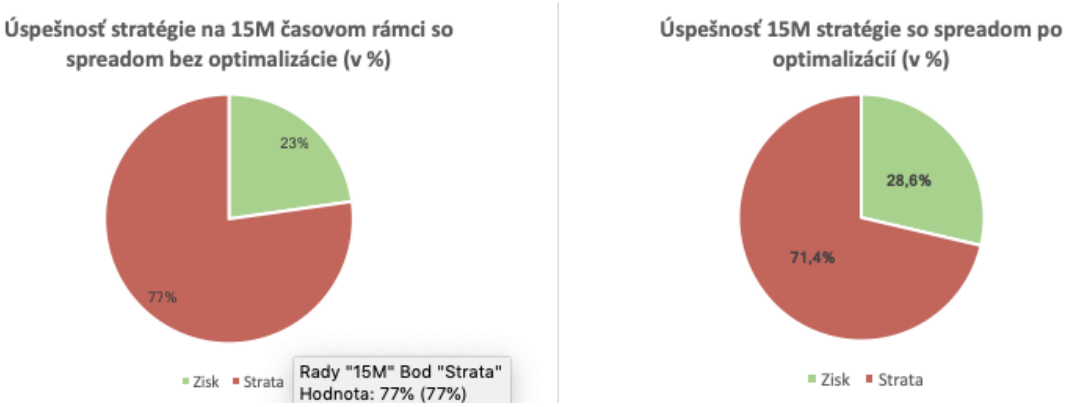
Graf č. 11



Zdroj: Vlastné spracovanie

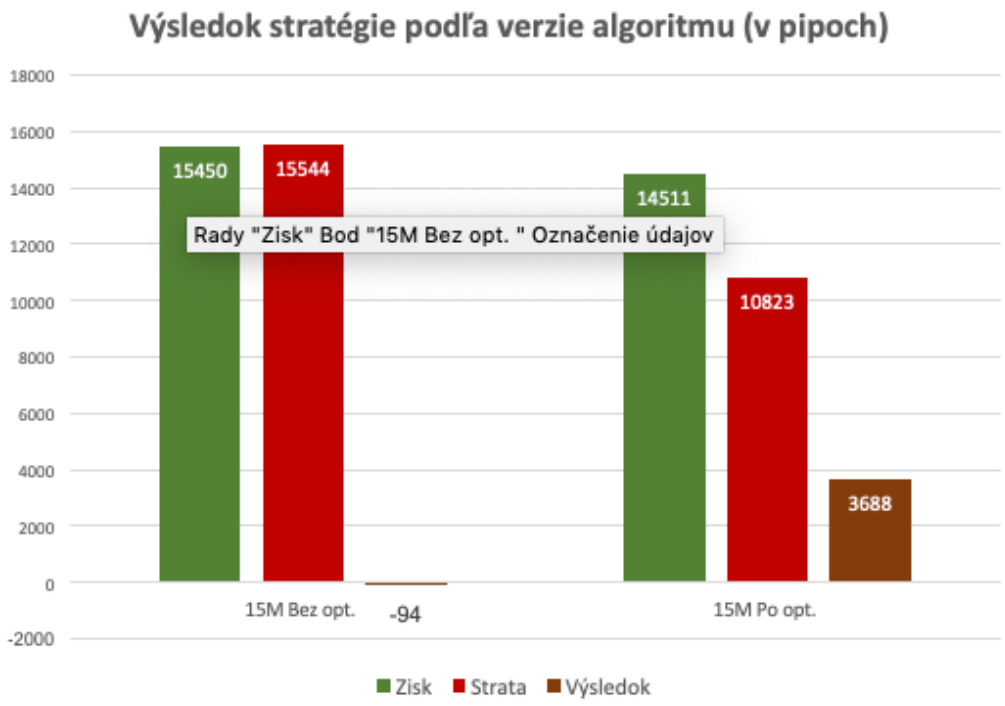
Vďaka časovej a menovej optimalizácií sa testovacia vzorka zmenšila o 61 obchodov z 246 na 185. Pomer ziskových ku stratovým pozíciám sa zvýšil o 5,6 % z pôvodných 23% na 28,6% pri takmer totožnom RRR na úrovni 3,34:1 z pôvodných 3,37:1. Najväčšiu výpovednú hodnotu má však v našom prípade počet dosiahnutých pipov v absolútnom vyjadrení. V pôvodnej verzií pred optimalizáciou bol výsledok -94 pipov v strate. Po vynechaní niektorých obchodných hodín a menového páru NZD/USD stratégia dosiahla 3688 pipov zisku ako rozdiel medzi stratou so 10823 jednotkami pohybu a ziskom so 14551 jednotkami. Pokročilá optimalizácia nie je vhodná nakoľko by sa štatistická vzorka naďalej výrazne zmenšovala. Porovnanie výsledkov prvej, neoptimalizovanej verzie a finálnej, po optimalizácií, je možné vidieť na grafoch č. 12 a 13. nižšie spolu s prílohou č. 1.

Graf č. 12



Zdroj: Vlastné spracovanie

Graf č. 13



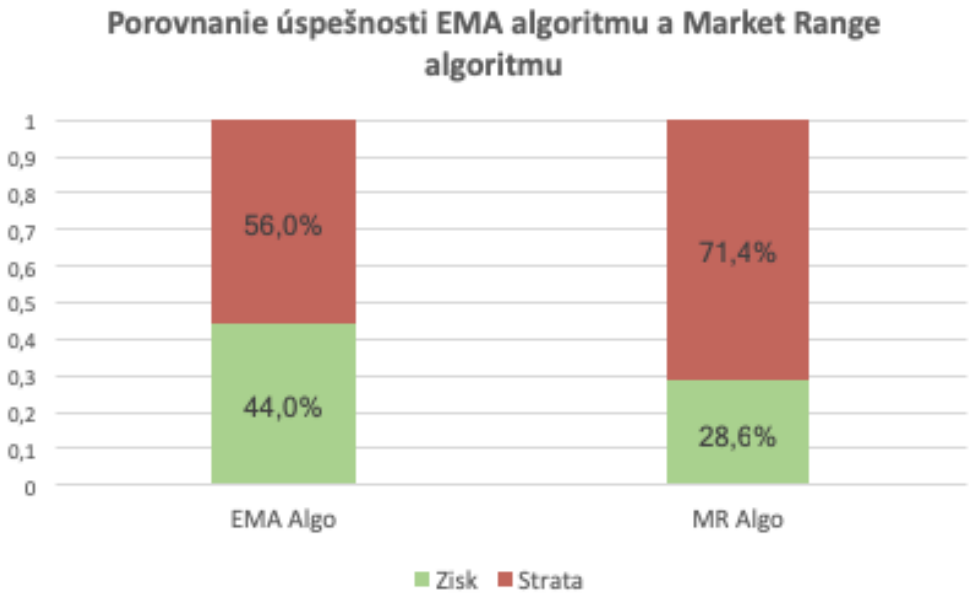
Zdroj: Vlastné spracovanie

3.5 Porovnanie finálnych verzií EMA algoritmu a Market Range algoritmu

V záverečnej časti tejto kapitoly sa pozrieme na porovnanie oboch obchodných prístupov a ich implementácie v programovacom jazyku Python. Oba algoritmy majú svoje špecifiká a to najmä v časovom rámci, na ktorom operujú. Pre prvý EMA algoritmus je to kratší 1M časový rámec a pri druhom algoritme sme zvolili 15M verziu, ktorá spomedzi

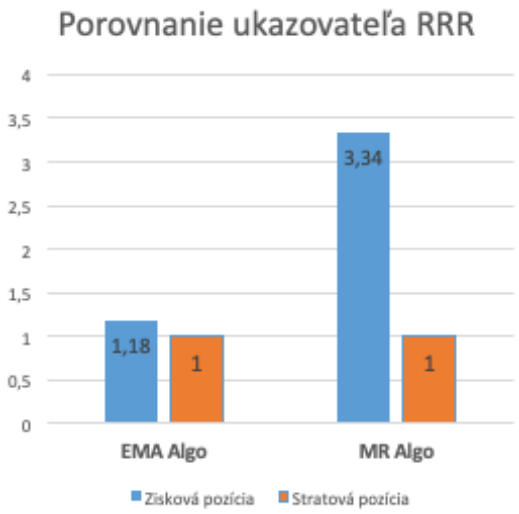
všetkých testovaných verzií priniesla najlepšie výsledky. Prvý z algoritmov ponúka úspešnosť na úrovni 44% oproti tomu druhému s úspešnosťou 28,6%. Viď graf č. 14. Napriek vyššiemu pomeru ziskových ku stratovým pozíciám u stratégie založenej na krížení hodnôt EMA ponúka druhý algoritmus výrazne vyššie priemerné RRR na úrovni 3,34:1 oproti 1,18:1. Viď graf č. 15. To je hlavným dôvodom prečo druhý algoritmus založený na obchodovaní distribučných pásiem výrazne vedie v celkovom počte dosiahnutých jednotiek pohybu ceny. Na základe dát zo 104 obchodov pri EMA stratégii a 185 uzavretých obchodov pri druhom algoritme môžeme preukázateľne interpretovať pomocou grafu č. 16, že pri využití druhého algoritmu sme dosiahli lepšie výsledky a to konkrétne o 3679 pipov. V prípade prvého algoritmu nenastane takmer žiadne zhodnotenie obchodného účtu.

Graf č. 14



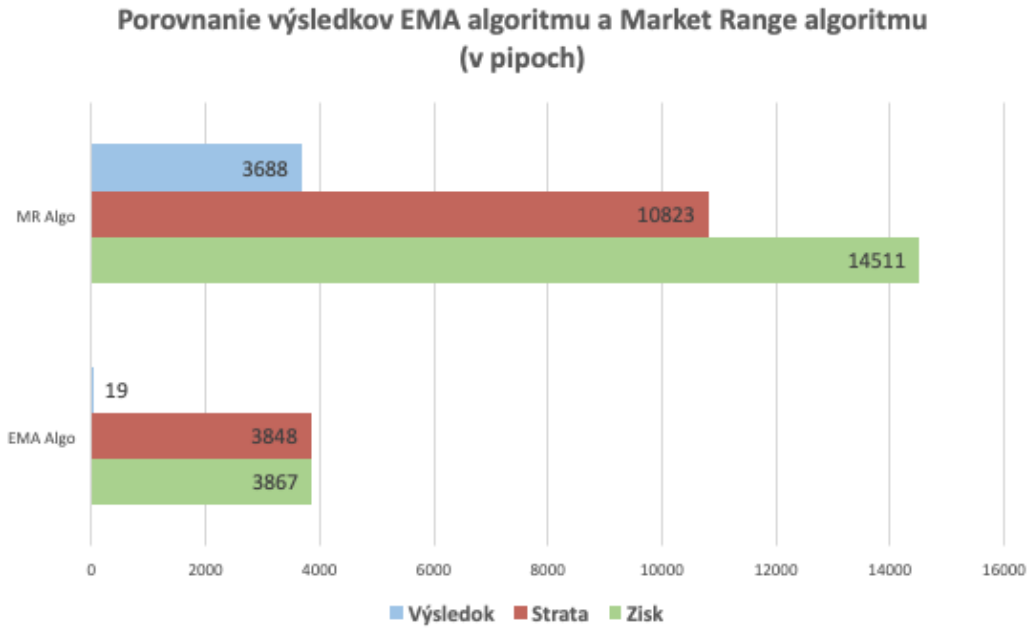
Zdroj: Vlastné spracovanie

Graf č. 15



Zdroj: Vlastné spracovanie

Graf č. 16



Zdroj: Vlastné spracovanie

Záver

Účelom tejto bakalárskej práce bolo vytvoriť automatizované obchodné systémy založené na rozdielnych konceptoch a to konkrétne obchodovanie kríženia EMA hodnôt a obchodovanie prienikov z distribučných pásiem a ich následné porovnanie za účelom preskúmania možnosti dosiahnutia lepších obchodných výsledkov na forexových trhoch. Vytvorené obchodné systémy spĺňajú všetky špecifiká definované v cieľoch práce vrátane plnej automatizácie, schopnosti operovať v reálnom čase bez zásahu, analyzovať vývoj cien a na základe vopred nadefinovaných pravidiel vstupovať do pozícií spolu so zadanými stop-loss a take-profit objednávkami. V prípade druhého algoritmu bola taktiež úspešne implementovaná funkcionalita testovania na historických cenách. Obe stratégie boli upravené pre najlepšie výsledky pomocou časovej a menovej optimalizácie. Výsledky dosiahnuté pri oboch systémoch boli porovnané a výstupom je tvrdenie, že obchodný algoritmus zameraný na obchodovanie prienikov z distribučných pásiem aj napriek nižšej úspešnosti ponúka vyššie Risk-Reward-Ratio a tým signifikantne vyšší potenciál na zhodnotenie obchodného účtu na rozdiel od algoritmu založeného na krížení EMA hodnôt. Toto tvrdenie je dokázané výstupmi z vyhodnocovaných dát.

Algoritmické obchodovanie pomocou technickej analýzy je veľmi dobrý nástroj pomocou ktorého sa môže obchodník bezstarostne zúčastňovať na dianí či už forexového, akciového, komoditného alebo iného trhu a to bez fyzického zásahu. Ponúka mnoho možností vytvárania a zlepšovania obchodných stratégií a rozvíja technické znalosti. Prvotná časová investícia na vývoj je z počiatku vysoká, no po úspešnom nastavení a optimalizácií dochádza k značnej úspore času a pri efektívnom fungovaní ponúka automatizované obchodovanie atraktívny nástroj na zvýšenie pasívnych príjmov. Existuje mnoho spôsobov a metód ako pristupovať k podobnému obchodovaniu a je len na obchodníkovi aký spôsob sa rozhodne využiť. Na záver mám potrebu zdôrazniť, že algoritmické obchodovanie je a naďalej aj bude súčasťou každodenného diania na finančných trhoch, no pri jeho využívaní je potrebné byť obozretný a venovať značnú časť testovaniu stratégie na cvičnom účte. V prípade nesprávne fungujúceho zdrojového kódu a zapojením reálneho účtu je veľmi pravdepodobná strata značnej časti obchodného účtu za veľmi krátky časový úsek.

Zoznam použitej literatúry

1. Ludvik Turek – Jak na Forex Czechwealth, 2009, Praha, s 27 – 32, 104s.

2. Ludvik Turek – Manuál technické analýzy, 2008, Praha, Czechwealth, s 7-18, 272s.

3. Dr. Alexander Elder – Come Into My Trading Room, 2002, John Wiley and Sons Inc., ISBN 0-471-22534-7, New York, s 15 – 23, 313s

4. Brian Dolan - Currency Trading for Dummies 2nd edition, Wiley Publishing, Inc., New Jersey, ISBN: 978-1-118-01851-4, s 119 – 140, s 187 – 189, 203 – 252, 332s

5. Kathy Lien – Day Trading and Swing Trading the Currency Market – 2nd edition, John Wiley and Sons Inc, 2009, New Jersey, ISBN: 978-0-470-37736-9, s 105 – 108, 290s

6. https://stackoverflow.com/ - vybrané problémy

7. David Easley, Marcos López De Prado and Maureen O´Hara – High-Frequency Trading, Incisive Media, London, ISBN 978-1-78272-009-6, s 15 - 20, 236s

8. Data from Commodity Futures Trading Commision, 2015

9. Morton Glantz, Robert Kissell, published on Wikimedia, GNU Free documentation license

10. Chan P. E., 2013 Algorithmic Trading – Winning Strategies and Their Rationale, Hoboken: John Wiley & Sons, Inc.

11. Aldridge I., 2013, High – Frequency Trading – A Practical Guide to Algorithmic Strategies and Trading Systems, 2nd ed., Hoboken: John Wiley & Sons, Inc.

12. Swigart A., 2015, Automate the Boring stuff with Python, San Francisco: No Starch Press, 978-1-59327-599-0, 481s

13. www.buildmedia.readthedocs.org, Oanda V-20 Rest API official documentation [online]. Dostupné na: https://buildmedia.readthedocs.org/media/pdf/oanda-api-v20/latest/oanda-api-v20.pdf

14. www.pandas.pydata.org, Pandas library official documentation [online]. Dostupné na: https://pandas.pydata.org/pandas-docs/stable/#

15. wwwmrjbq7.github.io, TA-Lib library official documentation [online]. Dostupné na: https://pandas.pydata.org/pandas-docs/stable/#

16. www.docs.python.org, Datetime library official documentation [online]. Dostupné na: https://docs.python.org/3/library/datetime.html

17. www.xlrd.readthedocs.io, xlrd library official documentation [online]. Dostupné na: https://xlrd.readthedocs.io/en/latest/

Prílohy

Príloha č. 1 Algoritmus založený na krížení EMA hodnôt – 1. modul

```
1.  import oandapyV20
2.  from oandapyV20 import API
3.  import pandas as pd
4.  import oandapyV20.endpoints.pricing as pricing
5.  import configparser
6.  import talib
7.  from pandas import ExcelWriter
8.  import oandapyV20.endpoints.instruments as instruments
9.
10. accountID = "101-004-8194699-
    002" # define accID, standalone variables so that we dont need to create them eve
    ry time
11. access_token = "0a504d77a452765e89fef14a396edbb5-
    926aa0288f617853dc8468faa10fe46d" # define api_token
12. menove_pary = ["EUR_USD", "NZD_USD","USD_CHF","GBP_USD","AUD_USD","USD_CAD","GBP_
    CHF"]
13. api = API(access_token = access_token) # initialize API
14.
15. def aktualna_cena_pre(par):  # najde akutalnu cenu pre dany menovy par, sluzi na
    TP a SL
16.
17.     import oandapyV20.endpoints.pricing as pricing
18.     params = {"instruments": str(par)}
19.     data_filter = pricing.PricingInfo(accountID=accountID,
20.                                       params=params)  # specifies values to call
    from request
21.     req_prices = api.request(data_filter)  # calls request, fetches data
22.     fetched_data = data_filter.response  # assigns fetched data to variable actua
    l price, not needed
23.     cena = fetched_data["prices"][0]["bids"][0]["price"]
24.     return float(cena)
25.
26. def fire_up(acc_id, access_t):
27.     df_hotovy = pd.DataFrame()
28.     accountID = acc_id # define accID
29.     access_token = access_t # define api_token
30.
31.     params = {"count":100, # sets parameters for query
32.               "granularity": "M1"}
33.     writer = pd.ExcelWriter("/Users/Cappucinoes/PycharmProjects/Bakalarka_Indic/s
    econd_algo/df_hotovy.xlsx")
34.     for par in menove_pary:
35.         data_set = instruments.InstrumentsCandles(instrument = par,params = param
    s) # prepares the request
36.         candle_data_request = api.request(data_set) #initialize request
37.         df = pd.DataFrame(data_set.response) #store fetched data into dataframe
38.         ohlc_list = ["o", "h", "l", "c"]  # support variable
39.         df_hotovy = pd.DataFrame(columns=["o", "h", "l", "c", "time", "type", "EM
    A_13", "EMA_50", "candle_size_pip"],
40.                                  index=[i for i in range(df.candles.shape[0])])
41.
42.         for i in range(df.candles.shape[0]):
43.             for w in ohlc_list:  # for every open, high, low, close in i row
44.                 df_hotovy[w][i] = df.candles[i].get("mid").get(w)  # ohlc columns
     into df
45.                 df_hotovy["time"][i] = df.candles[i].get("time")  # time column i
    nto df
46.
47.
```

```
48.             if df_hotovy["c"][i] > df_hotovy["o"][i]:
49.                 df_hotovy["type"][i] = "bullish"  # if cena close > open = bullis
    h
50.
51.             elif df_hotovy["c"][i] < df_hotovy["o"][i]:
52.                 df_hotovy["type"][i] = "bearish"  # opak vrchneho if
53.
54.             else:
55.                 df_hotovy["type"][i] = "neutral"
56.
57.             df_hotovy["candle_size_pip"][i] = abs(
58.                 float(df_hotovy["h"][i]) - float(df_hotovy["l"][i]))  # calculate
     size of the bullish candle
59.         swing_col = pd.Series(index=range(1, df_hotovy.shape[0] - 2))
60.
61.         for i in range(2, df_hotovy.shape[0] - 2):
62.
63.             if df_hotovy.h[i] > df_hotovy.h[i + 1] and df_hotovy.h[i] > df_hotovy
    .h[i - 1] and df_hotovy.h[i] > \
64.                     df_hotovy.h[
65.                         i - 2] and df_hotovy.h[i] > df_hotovy.h[i + 2]:
66.                 swing_col[i] = "SWING_HIGH"
67.             elif df_hotovy.l[i] < df_hotovy.l[i + 1] and df_hotovy.l[i] < df_hoto
    vy.l[i - 1] and df_hotovy.l[i] < \
68.                     df_hotovy.l[
69.                         i - 2] and df_hotovy.l[i] < df_hotovy.l[i + 2]:
70.                 swing_col[i] = "SWING_LOW"
71.             else:
72.                 pass
73.         df_hotovy["SWING"] = swing_col
74.
75.         df_hotovy["EMA_13"] = talib.EMA(df_hotovy["c"], timeperiod=13)
76.         df_hotovy["EMA_50"] = talib.EMA(df_hotovy["c"], timeperiod=50)
77.
78.         cross_index = []
79.         cross_value = []
80.         for i,a in df_hotovy.iterrows():
81.
82.             if (a["EMA_13"] < a["EMA_50"]) and (df_hotovy["EMA_13"][i-
    1] > df_hotovy["EMA_50"][i-1]):
83.                 cena = aktualna_cena_pre(par)
84.                 cross_index.append(i)
85.                 cross_value.append({"SL":cena+0.00050,
86.                                     "TP":cena-0.00100,
87.                                     "TYPE":"SELL",
88.                                         "PAIR":par})
89.
90.             elif (a["EMA_13"] > a["EMA_50"]) and (df_hotovy["EMA_13"][i-
    1] < df_hotovy["EMA_50"][i-1]):
91.                 cena = aktualna_cena_pre(par)
92.                 cross_index.append(i)
93.                 cross_value.append({"SL":cena-0.00050,
94.                                     "TP":cena+0.00100,
95.                                     "TYPE":"BUY",
96.                                         "PAIR":par})
97.
98.             else:
99.                 pass
100.               cross_ser = pd.Series(cross_value, index=cross_index)
101.
102.               df_hotovy["EMA_CROSS"] = cross_ser
103.               df_hotovy.to_excel(writer, par)
104.           writer.save()
105.           print("Data ulozene!")
106.
107.       #saved on 12th of april, paths for MAC
```

Príloha č. 2 Algoritmus založený na krížení EMA hodnôt – 2. modul

```
1.  import datetime
2.  import time
3.  import forex_bakalarka # initializes connection gate with 1M feed
4.  import pandas as pd
5.  import xlrd
6.  from oandapyV20 import API
7.  import ast
8.  import json
9.
10. accountID = forex_bakalarka.accountID
11. access_token = forex_bakalarka.access_token
12. second_algo_historia = []           # file do ktoreho sa ulozia objednavky aby ne
    mal zdvojene
13.
14. def spusti_range_signal(): #if this function is called, request for 5M data df is
     sent
15.     forex_bakalarka.fire_up(accountID, access_token)
16.
17. def open_trade(SL, TP, typ, mena, units=2000):
18.     typ = typ.replace("''"", "")
19.     mena = mena.replace("''", "")
20.     SL = str(round(float(SL), 5))
21.     TP = str(round(float(TP), 5))
22.     try:
23.         if typ == "SELL":
24.             units = -units
25.
26.         elif typ == "BUY":
27.             units = units
28.
29.         order_data = {
30.             "order": {
31.                 "stopLossOnFill": {
32.                     "timeInForce": "GTC",
33.                     "price": str(SL)
34.                 },
35.                 "takeProfitOnFill": {
36.                     "timeInForce": "GTC",
37.                     "price": str(TP)
38.                 },
39.                 "timeInForce": "FOK",
40.                 "instrument": str(mena),
41.                 "units": str(units),
42.                 "type": "MARKET",
43.                 "positionFill": "DEFAULT"
44.             }
45.         }
46.
47.         r = orders.OrderCreate(accountID=accountID, data=order_data)
48.         client.request(r)
49.         print("Objednavka otvorena {}".format(mena))
50.         second_algo_historia.append(mena)
51.         vsetky_identifikovane_objednavky = open("/Users/Cappucinoes/PycharmProjec
    ts/Bakalarka_Indic/second_algo/vsetky_identifikovane_objednavky.txt", "a+")
52.         vsetky_identifikovane_objednavky.write("{} {} \n".format(mena, datetime.d
    atetime.now()))
53.         vsetky_identifikovane_objednavky.close()
54.         return second_algo_historia
55.
```

```
    except Exception as Error:
        print(Error)
        print("TP je {}".format(TP), "Format TP je {}".format(type(TP)))
        pass

starttime = time.time()
import xlrd
forex_bakalarka.fire_up(accountID,access_token)
xlsx = xlrd.open_workbook("/Users/Cappucinoes/PycharmProjects/Bakalarka_Indic/second_algo/df_hotovy.xlsx")

import oandapyV20.endpoints.orders as orders  # sluzi na otvaranie pozicii
import oandapyV20.endpoints.trades as trades  # sledovanie aktualnych pozicii
client = API(access_token=access_token)
print("Refreshing active for every 60 seconds.")

while True:
    now = datetime.datetime.now().second
    minutes = datetime.datetime.now().minute

    if minutes in [a for a in range(0,60,10)]:
        second_algo_historia = []         # premaze file kazdych 10 minut, az potom vie otvorit poziciu na rovnakom pare
    else:
        pass

    if now == 1: # cakat tri sekundy, aktualne sviecky, aktualna minuta, je treba zmenit pri vyssom ramci
        spusti_range_signal() #asks for data from 5M range_signal modul
        print("Spusteny modul forex_bakalarka")

        try:
            for sheet_name in xlsx.sheet_names():
                print("Vyhladavam signaly pre {}.".format(sheet_name))
                signals = pd.read_excel("/Users/Cappucinoes/PycharmProjects/Bakalarka_Indic/second_algo/df_hotovy.xlsx",sheet_name)["EMA_CROSS"]

                if pd.isnull(signals[signals.index[-1]]) == False:
                    print("MAME SIGNAL NA {} {}".format(sheet_name, signals[signals.index[-1]]))

                    signal_str = signals[signals.index[-1]] # z tohto stringu potrebujeme dictionary aby sme ho mohli dat do jsonu a poslat na objednavku
                    signal_str = signal_str[1:-1].replace(":", "").strip().split(",")
                    values = [i.split()[1] for i in signal_str]
                    SL,TP,typ,mena = [i for i in values]
                    mena = sheet_name
                    print(type(SL),type(TP),type(typ),type(mena))

                    if mena not in second_algo_historia:
                                open_trade(SL,TP,typ,mena,units = 1000)
                                print("Aktualna cena je {}".format(forex_bakalarka.aktualna_cena_pre(mena)))
                            else:
                                print("ZABRANILO SA DVOJITEMU OBCHODU")

                        else:
                            pass

            except (ConnectionError) as error:
                print(error)
                pass
            print("Hladanie skoncilo")
```

```
112.              #time.sleep(60.0 - ((time.time() 0- starttime)) % 60.0) # refresh
    every 60 seconds, needs to be 5mins
113.
114.       #saved on 12th of april, paths for mac
```

Príloha č. 3 Algoritmus založený na krížení EMA hodnôt – 3. script – štatistika

```
1.  import pandas as pd
2.  import datetime
3.  import numpy as np
4.
5.  pd.set_option('display.width', 320)
6.  pd.set_option("display.max_columns", 20)
7.  columns = ["Date_open", "Order_type", "Size", "Pair","Open_price", "SL", "TP", "D
    ate_closed", "Closed_price"]
8.  trading_history = pd.read_csv("/Users/Cappucinoes/PycharmProjects/Bakalarka_Indic
    /second_algo/AccountHistory.csv",sep=";", usecols= [i for i in range(14)])
9.
10. trading_history.loc[trading_history.shape[0]] = trading_history.columns
11. profit_col = trading_history[trading_history.columns[12]]
12. trading_history = trading_history[[i for i in trading_history.columns[1:10]]]
13. trading_history.columns = columns
14. trading_history["Result"] = profit_col
15.
16. trading_history.Result.loc[244] = float(trading_history.Result.loc[244])
17. #print(trading_history.Result.apply(lambda x: abs(x)).mean())
18. trading_history["Result_string"]= trading_history.Result.apply(lambda x: "LOOSE"
    if x < 0 else "WIN")
19. #print(trading_history.groupby(by = trading_history.Result_string).count().Result
    )
20. trading_history.to_excel("/Users/Cappucinoes/PycharmProjects/Bakalarka_Indic/seco
    nd_algo/data_graf.xlsx")
21.
22. trading_history.Date_closed = pd.to_datetime(trading_history.Date_closed)
23. trading_history.sort_values(by= "Date_closed", inplace= True)
24.
25. trading_history.drop(trading_history.index[0], inplace=True)
26.
27. trading_history.to_excel("/Users/Cappucinoes/PycharmProjects/Bakalarka_Indic/seco
    nd_algo/data_graf.xlsx")
28. trading_history.drop(trading_history.index[243], inplace= True)
29. trading_history.Open_price = trading_history.Open_price.apply(lambda x: float(x))

30. trading_history.Closed_price= trading_history.Closed_price.apply(lambda x: float(
    x))
31. trading_history["Pips"]= trading_history.apply(lambda x: abs(x.Open_price - x.Clo
    sed_price), axis = 1)
32.
33. bez_opt_avg_win = trading_history[trading_history.Result_string == "WIN"].Pips.me
    an()
34.
35. bez_opt_avg_loose = trading_history[trading_history.Result_string == "LOOSE"].Pip
    s.mean()
36. print(bez_opt_avg_win, bez_opt_avg_loose)
37. print(f"RRR bez optimalizacie je {bez_opt_avg_win/bez_opt_avg_loose}")
38.
39. trading_history.Open_price = trading_history.Open_price.apply(lambda x: float(x))

40. trading_history.Closed_price= trading_history.Closed_price.apply(lambda x: float(
    x))
41. trading_history["Pips"]= trading_history.apply(lambda x: abs(x.Open_price - x.Clo
    sed_price), axis = 1)
42. print(trading_history.groupby("Result_string").Pips.sum())
43.
```

```
trading_history["hour"] = trading_history.Date_closed.apply(lambda x: x.hour)

trading_history.to_excel("/Users/Cappucinoes/PycharmProjects/Bakalarka_Indic/second_algo/data_graf.xlsx")
hour_result_group = trading_history.groupby(by=["hour","Result_string"]).count().Result
test = trading_history.groupby(by=["hour", "Result_string"])[["Result_string"]] # single bracket = series, double = Dataframe
test_count_grouped = test.count()
hour_result_group.to_excel("/Users/Cappucinoes/PycharmProjects/Bakalarka_Indic/second_algo/data_graf.xlsx")

df_percenta = pd.DataFrame(columns=["WIN", "LOOSE"])
for i in test_count_grouped.index.levels[0]:
    try:
        win_percento = test_count_grouped.loc[i].loc["WIN"] / (test_count_grouped.loc[i].loc["WIN"]+test_count_grouped.loc[i].loc["LOOSE"])
        loose_percento = test_count_grouped.loc[i].loc["LOOSE"] / (test_count_grouped.loc[i].loc["WIN"] + test_count_grouped.loc[i].loc["LOOSE"])
        dictionary = dict(zip(["WIN","LOOSE"], [win_percento[0],loose_percento[0]]))
        df_percenta.loc[i] = dictionary
    except (IndexError,KeyError):
        pass
df_percenta.to_excel("/Users/Cappucinoes/PycharmProjects/Bakalarka_Indic/second_algo/percenta.xlsx")
bad_hours = [0,6,7,15,16,17,20,23]

def vymaz_hodin(hodina):

    if hodina in bad_hours:
        return np.nan
    else:
        return int(hodina)
#Vysledok podla parov bez odstranenia hodin

bez_opt_pair_df = trading_history.groupby(["Pair","Result_string"]).count()[["Result"]]
bez_opt_pary_excel = pd.Series()
index = []
for i in bez_opt_pair_df.index.levels[0]:
    index.append(i)
    win_ratio = bez_opt_pair_df.loc[i].loc["WIN"][0]/(bez_opt_pair_df.loc[i].loc["LOOSE"][0] + bez_opt_pair_df.loc[i].loc["WIN"][0])
    bez_opt_pary_excel[i] = win_ratio
bez_opt_pary_excel.to_excel("/Users/Cappucinoes/PycharmProjects/Bakalarka_Indic/second_algo/bez_opt_pary.xlsx")

trading_history.hour = trading_history.hour.apply(vymaz_hodin)
trading_history.dropna(how="any",axis = 0, inplace = True)
print(trading_history.sort_values(by = "hour").groupby(by = "Result_string").Pips.sum())
cas_opt_avg_win = trading_history[trading_history.Result_string == "WIN"].Pips.mean()
cas_opt_avg_loose = trading_history[trading_history.Result_string == "LOOSE"].Pips.mean()
print(cas_opt_avg_win/abs(cas_opt_avg_loose))
print(trading_history.groupby(["Result_string", "Pair"]).count().Result)

ciastocna_optimalizacia = trading_history[["Result","Result_string"]]
ciastocna_optimalizacia.to_excel("/Users/Cappucinoes/PycharmProjects/Bakalarka_Indic/second_algo/ciastocna_optimalizacia.xlsx")

#Vysledok podla parov s optimalizaciou hodin
po_opt_pair_df = trading_history.groupby(["Pair","Result_string"]).count()[["Result"]]
```

```
po_opt_pary_excel = pd.Series()
index = []
for i in po_opt_pair_df.index.levels[0]:
    index.append(i)
    win_ratio = po_opt_pair_df.loc[i].loc["WIN"][0]/(po_opt_pair_df.loc[i].loc["LOOSE"][0] + po_opt_pair_df.loc[i].loc["WIN"][0])
    po_opt_pary_excel[i] = win_ratio
      po_opt_pary_excel.to_excel("/Users/Cappucinoes/PycharmProjects/Bakalarka_Indic/second_algo/po_opt_pary.xlsx")
      pairs = ["NZDUSD","USDCAD"]
      trading_history["Pips"]= trading_history.apply(lambda x: abs(x.Open_price - x.Closed_price), axis = 1)

      opt_pair = trading_history[trading_history.Pair.isin(pairs)]
      # print(opt_pair.groupby("Result_string").count().Result)
      # print(opt_pair[opt_pair.Result_string == "WIN"].Result.mean())
      # print(opt_pair[opt_pair.Result_string == "LOOSE"].Result.mean())
      # print(opt_pair.groupby("Result_string").Pips.mean())
      # print(opt_pair.groupby("Result_string").Pips.sum())
      # print(trading_history.groupby("Result_string").Pips.sum())
      #results saved with graphs

      pairs = ["NZDUSD","USDCAD", "AUDUSD","USDCHF", "GBPUSD"]
      bad_hours = [0,6,7,20,23,15,16,17]

      opt_pair_time = trading_history[trading_history.Pair.isin(pairs)]
      print(opt_pair_time.groupby("Result_string").count().Result)
      print(opt_pair_time.groupby("Result_string").Pips.sum())
      avg_win = opt_pair_time[opt_pair_time.Result_string == "WIN"].Result.mean()
      avg_loose = opt_pair_time[opt_pair_time.Result_string == "LOOSE"].Result.mean()
      print(avg_win/abs(avg_loose))
```

Príloha č. 4 Algoritmus založený na identifikácií distribučných pásiem – 1. modul obchodovania v reálnom čase

```
import oandapyV20
from oandapyV20 import API
import pandas as pd
import oandapyV20.endpoints.orders as orders
import oandapyV20.endpoints.instruments as instruments
import talib
import oandapyV20.endpoints.pricing as pricing
import configparser

accountID = "101-004-8194699-002" # define accID, stanalone variables so that we dont need to create them every time
access_token = "0a504d77a452765e89fef14a396edbb5-926aa0288f617853dc8468faa10fe46d" # define api_token
api = API(access_token) # initialize API
menove_pary = ["EUR_USD", "NZD_USD","USD_CHF","GBP_USD","AUD_USD","USD_CAD","GBP_CHF"] #list of currencies we wish to use

def fire_up(acc_id, access_t):
    accountID = acc_id # define accID
    access_token = access_t # define api_token

    params = {"count":300, # sets parameters for query
                  "granularity": "M15"}
    writer = pd.ExcelWriter("/Users/Cappucinoes/PycharmProjects/Bakalarka_PA/range_market_multi/df_hotovy.xlsx")

```

```
    for par in menove_pary: # bellow is a list of operations every sheet does

        data_set = instruments.InstrumentsCandles(instrument = par,params = params) # prepares the request
        api.request(data_set) #initialize request
        df = pd.DataFrame(data_set.response) #store fetched data into dataframe

        ohlc_list = ["o", "h", "l", "c"]  # support variable
        df_hotovy = pd.DataFrame(columns=["o", "h", "l", "c", "time", "type", "EMA_20", "candle_size_pip"],
                                 index=[i for i in range(df.candles.shape[0])])

        for i in range(df.candles.shape[0]):
            for w in ohlc_list:  # for every open, high, low, close in i row
                df_hotovy[w][i] = df.candles[i].get("mid").get(w)  # ohlc columns into df
                df_hotovy["time"][i] = df.candles[i].get("time")  # time column into df

            if df_hotovy["c"][i] > df_hotovy["o"][i]:
                df_hotovy["type"][i] = "bullish"  # if cena close > open = bullish

            elif df_hotovy["c"][i] < df_hotovy["o"][i]:
                df_hotovy["type"][i] = "bearish"  # opak vrchneho if
            else:
                df_hotovy["type"][i] = "neutral"

            df_hotovy["candle_size_pip"][i] = abs(
                float(df_hotovy["h"][i]) - float(df_hotovy["l"][i]))  # calculate size of the bullish candle

        swing_col = pd.Series(index=range(1, df_hotovy.shape[0] - 2))

        for i in range(2, df_hotovy.shape[0] - 2):

            if df_hotovy.h[i] > df_hotovy.h[i + 1] and df_hotovy.h[i] > df_hotovy.h[i - 1] and df_hotovy.h[i] > df_hotovy.h[
                i - 2] and df_hotovy.h[i] > df_hotovy.h[i + 2]:
                swing_col[i] = "SWING_HIGH"
            elif df_hotovy.l[i] < df_hotovy.l[i + 1] and df_hotovy.l[i] < df_hotovy.l[i - 1] and df_hotovy.l[i] < df_hotovy.l[
                i - 2] and df_hotovy.l[i] < df_hotovy.l[i + 2]:
                swing_col[i] = "SWING_LOW"
            else:
                pass

        df_hotovy["EMA_20"] = talib.EMA(df_hotovy["c"], timeperiod=10)  # upraveny period z 20

        pd.set_option("display.float_format",
                      lambda x: "%.5f" % x)  # sets pandas number format for rounding to 0.000000

        df_hotovy["swing_col"] = swing_col

        # RANGE IDENTIFIER
        # pocitanie vzdialenosti medzi swingami
        cislo_sviecky = 0
        highs = []
        lows = []
        indexy_pre_swing_high = []
        indexy_pre_swing_low = []

        for swing in df_hotovy["swing_col"]:
            if swing == "SWING_HIGH" or swing == "SWING_LOW":
```

```
79.
80.                 if swing == "SWING_HIGH":
81.                     highs.append(df_hotovy.h[cislo_sviecky])  # prida cenu high s
    wingu do zoznamu 1. upper border
82.                     indexy_pre_swing_high.append(cislo_sviecky)
83.
84.
85.                 elif swing == "SWING_LOW":
86.                     lows.append(df_hotovy.l[cislo_sviecky])  # prida cenu low swi
    ngu do zoznamu 2. bottom border
87.                     indexy_pre_swing_low.append(cislo_sviecky)
88.
89.             cislo_sviecky += 1
90.         indexy_pre_swing_high = indexy_pre_swing_high[2:]  # sluzia na najdenie i
    ndexu riadka a nasledny concat s df_hotovy
91.         indexy_pre_swing_low = indexy_pre_swing_low[2:]
92.
93.         # H a H swingove rozdiely
94.         highs_float = [float(x) for x in highs]
95.         highs_reversed_float = highs_float[-1::-
    1]  # od najnovsej ceny high_swingu po najstarsiu
96.         pd.Series(highs_float, index=[i for i in range(len(highs))]).plot()
97.
98.         cena_high_swingu = []
99.         rozdiel_1_predosla_h = []
100.                rozdiel_2_predosla_h = []
101.
102.                for cena_high in range(len(highs_reversed_float) - 2):
103.                    cena_high_swingu.append(highs_reversed_float[cena_high])
104.
105.                    calc = highs_reversed_float[cena_high] - highs_reversed_float[
    cena_high + 1]
106.                    rozdiel_1_predosla_h.append(abs(calc))
107.
108.                    calc = highs_reversed_float[cena_high] - highs_reversed_float[
    cena_high + 2]
109.                    rozdiel_2_predosla_h.append(abs(calc))
110.
111.                rozdiel_1_predosla_h = [round(x, 5) for x in rozdiel_1_predosla_h]

112.                rozdiel_2_predosla_h = [round(x, 5) for x in rozdiel_2_predosla_h]

113.
114.                cena_high_swingu = pd.Series(cena_high_swingu)
115.                rozdiel_1_predosla = pd.Series(rozdiel_1_predosla_h)
116.                rozdiel_2_predosla = pd.Series(rozdiel_2_predosla_h)
117.                H_porovnanie_odnajnovsej = pd.concat(
118.                    [cena_high_swingu, rozdiel_1_predosla, rozdiel_2_predosla, pd.
    Series(indexy_pre_swing_high[-1::-1])], axis=1)
119.                H_porovnanie_odnajnovsej.columns = ["cena_high_swingu", "rozdiel_1
    _predosla_h", "rozdiel_2_predosla_h",
120.                                                    "indexy_pre_swing_high"]
121.                H_porovnanie_odnajstarsej = H_porovnanie_odnajnovsej.iloc[-1::-
    1]
122.                H_porovnanie_odnajstarsej.set_index("indexy_pre_swing_high", drop=
    True,
123.                                                    inplace=True)  # porovnanie H
    swingov v rovnakom formate ako df_hotovy
124.                # H_porovnanie_odnajstarsej hotovy dataframe pre concat s df_hotov
    y
125.
126.                df_hotovy = pd.concat([df_hotovy, H_porovnanie_odnajstarsej], axis
    =1)
127.
128.                # L a L rozdiely
129.                cena_low_swingu = []
```

```
130.                rozdiel_1_predosla_l = []
131.                rozdiel_2_predosla_l = []
132.                lows_reversed_float = [float(x) for x in lows[-1::-1]]
133.
134.                for cislo_operacie in range(len(lows_reversed_float) - 2):
135.                    cena_low_swingu.append(lows_reversed_float[cislo_operacie])
136.
137.                    calc = abs(lows_reversed_float[cislo_operacie] - lows_reversed
    _float[cislo_operacie + 1])
138.                    rozdiel_1_predosla_l.append(calc)
139.
140.                    calc = abs(lows_reversed_float[cislo_operacie] - lows_reversed
    _float[cislo_operacie + 2])
141.                    rozdiel_2_predosla_l.append(calc)
142.
143.                cena_low_swingu = pd.Series(cena_low_swingu)
144.                rozdiel_1_predosla_l = pd.Series(rozdiel_1_predosla_l)
145.                rozdiel_2_predosla_l = pd.Series(rozdiel_2_predosla_l)
146.
147.                L_porovnanie_odnajnovsej = pd.concat(
148.                    [cena_low_swingu, rozdiel_1_predosla_l, rozdiel_2_predosla_l,
    pd.Series(indexy_pre_swing_low[-1::-1])], axis=1)
149.                L_porovnanie_odnajnovsej.columns = ['cena_low_swingu', 'rozdiel_1_
    predosla_l', 'rozdiel_2_predosla_l',
150.                                                    "indexy_pre_swing_low"]
151.                L_porovnanie_odnajnovsej.rozdiel_1_predosla_l.plot()
152.                L_porovnanie_od_najstarsej = L_porovnanie_odnajnovsej.iloc[-1::-
    1]
153.                L_porovnanie_od_najstarsej.set_index("indexy_pre_swing_low", drop=
    True,
154.                                                     inplace=True)  # rozdiel medz
    i swing lowami, zoradene podla df_hotovy
155.                # L_porovnanie_od_najstarsej hotovy df na concat s df_hotovy
156.                df_hotovy = pd.concat([df_hotovy, L_porovnanie_od_najstarsej], axi
    s=1)
157.
158.                # Tento modul pocita rozdiely medzi H swingom a L swingom v pipoch
    
159.                # vacsia hodnota rozdielu = silnejsi trendovy pohyb
160.                highs_reversed = highs[-1::-1]
161.                lows_reversed = lows[-1::-1]
162.
163.                list_rozdielov = []
164.                if len(highs_reversed) > len(lows_reversed):  # pripad kedy mame v
    iac swing highov
165.                    for i in range(len(lows_reversed)):
166.                        rozdiel = abs(float(lows_reversed[i]) - float(highs_revers
    ed[i]))
167.                        list_rozdielov.append(rozdiel)
168.
169.                elif len(highs_reversed) < len(lows_reversed):  # pripad kedy mame
     viac swing lowov
170.                    for i in range(len(highs_reversed)):
171.                        rozdiel = abs(float(lows_reversed[i]) - float(highs_revers
    ed[i]))
172.                        list_rozdielov.append(rozdiel)
173.
174.                list_rozdielov = [round(x, 5) for x in list_rozdielov]
175.                import openpyxl
176.                # AVERAGE SIZE OF LAST 10 CANDLES
177.                avg_last_10 = []
178.                ceny = []
179.                for avg_period in range(10, df_hotovy.shape[0], 10):
180.                    avg_period = int(avg_period)
181.                    data = df_hotovy.candle_size_pip[(avg_period - 10): (avg_perio
    d)]
```

```
                    avg = sum([i for i in dict(data).values()]) / len([i for i in dict(data).values()])
                    avg_last_10.append(avg)
                    ceny.append(df_hotovy.o[avg_period])

                avg_last_10 = pd.Series(avg_last_10, index=[i for i in range(10, df_hotovy.shape[0], 10)])
                df_hotovy["avg_last_10"] = avg_last_10
                index = []
                EMA_status = []
                for i in range(
                        df_hotovy.shape[0] - (int(df_hotovy.columns[6][4:6]) + 1)):  # +1 alebo nie? porovnat hodnotu rozdielu v df

                    rozdiel = abs(df_hotovy.EMA_20[df_hotovy.shape[0] - (1 + i)] - df_hotovy.EMA_20[df_hotovy.shape[0] - (2 + i)])
                    if rozdiel <= 0.00002:
                        EMA_status.append("EMA FLAT")
                        index.append(df_hotovy.shape[0] - (1 + i))

                ema_series = pd.Series(EMA_status, index=index)[-1::-1]

                df_hotovy["FLAT_INDIC"] = ema_series
                # RANGE CEILING
                test_df = df_hotovy[["cena_high_swingu", "rozdiel_1_predosla_h", "rozdiel_2_predosla_h"]]
                test_df = test_df.iloc[-1::-1]

                roof_price = []
                roof_price_index = []
                pocitadlo = test_df.shape[0]

                for prvok in test_df.rozdiel_1_predosla_h:
                    if pd.isnull(prvok) == False and prvok < 0.00009:
                        roof_price.append(prvok)
                        roof_price_index.append(pocitadlo - 1)
                    pocitadlo -= 1

                roof_price_series = pd.Series(roof_price, index=roof_price_index)

                pocitadlo_2 = test_df.shape[0]

                for rozdiel in test_df.rozdiel_2_predosla_h:
                    if pd.isnull(rozdiel) == False and rozdiel < 0.00009:

                        if (pocitadlo_2 - 1) in roof_price_index and roof_price_series[pocitadlo_2 - 1] > rozdiel:
                            roof_price_series[pocitadlo_2 - 1] = rozdiel
                        elif (pocitadlo_2 - 1) in roof_price_index and roof_price_series[pocitadlo_2 - 1] < rozdiel:
                            pass

                        else:
                            roof_price_series[pocitadlo_2 - 1] = rozdiel
                    pocitadlo_2 -= 1
                test_df["roof_price"] = roof_price_series

                def roof_price(index):
                    test_df.roof_price[index] = test_df.cena_high_swingu[index]

                roof_price([i for i in roof_price_series.index])
                test_df = test_df["roof_price"]
                df_hotovy["roof_price"] = test_df

                # RANGE FLOOR MODUL
```

```
240.                test_df = df_hotovy[["cena_low_swingu", "rozdiel_1_predosla_l", "r
    ozdiel_2_predosla_l"]]
241.                test_df = test_df.iloc[-1::-1]
242.
243.                floor_price = []
244.                floor_price_index = []
245.                pocitadlo = test_df.shape[0]
246.
247.                for prvok in test_df.rozdiel_1_predosla_l:
248.                    if pd.isnull(prvok) == False and prvok < 0.00013:
249.                        floor_price.append(prvok)
250.                        floor_price_index.append(pocitadlo - 1)
251.                    pocitadlo -= 1
252.
253.                floor_price_series = pd.Series(floor_price, index=floor_price_inde
    x)
254.                pocitadlo_2 = test_df.shape[0]
255.
256.                for rozdiel in test_df.rozdiel_2_predosla_l:
257.                    if pd.isnull(rozdiel) == False and rozdiel < 0.00013:
258.
259.                        if (pocitadlo_2 - 1) in floor_price_index and floor_price_
    series[pocitadlo_2 - 1] > rozdiel:
260.                            floor_price_series[pocitadlo_2 - 1] = rozdiel
261.                        elif (pocitadlo_2 - 1) in floor_price_index and floor_pric
    e_series[pocitadlo_2 - 1] < rozdiel:
262.                            pass
263.
264.                        else:
265.                            floor_price_series[pocitadlo_2 - 1] = rozdiel
266.                    pocitadlo_2 -= 1
267.                test_df["floor_price"] = floor_price_series
268.
269.                def floor_price(index):
270.                    test_df.floor_price[index] = test_df.cena_low_swingu[index]
271.
272.                floor_price([i for i in floor_price_series.index])
273.                test_df = test_df["floor_price"]
274.                df_hotovy["floor_price"] = test_df
275.
276.                # PREDPOKLAD PRE VOID SVIECKU
277.                test_df = df_hotovy
278.                indexy = []
279.                hodnoty = []
280.                for i in range(test_df.shape[0]):
281.                    if pd.isnull(test_df.swing_col[i]) == False:
282.                        indexy.append(i)
283.                        hodnoty.append(test_df.swing_col[i])
284.
285.                swing_ser = pd.Series(hodnoty, index=indexy)
286.                indexy = indexy[-1::-1]
287.                swing_ser = swing_ser[-1::-
    1]  # otocit na lepsiu manipulaciu v loope (od najnovsieho po najstarsie)
288.                spodna_hranica = []
289.                index_pre_spodnu_hranicu = []
290.
291.                for index_indexov in range(len(indexy)):
292.                    if df_hotovy.roof_price[indexy[index_indexov]] > 0:
293.                        for i in range(1, 6):
294.
295.                            try:
296.                                if df_hotovy.swing_col[indexy[index_indexov + i]]
    == "SWING_LOW":
297.                                    spodna_hranica.append(df_hotovy.l[indexy[index
    _indexov + i]])
```

```
                                    index_pre_spodnu_hranicu.append(indexy[index_indexov])
                                    break
                            except IndexError:
                                spodna_hranica.append("NaN")
                                index_pre_spodnu_hranicu.append(indexy[index_indexov])
                                break

                spodna_hranica_ser = pd.Series(spodna_hranica, index=index_pre_spodnu_hranicu)
                df_hotovy["spodna_hranica"] = spodna_hranica_ser
                horna_hranica = []
                index_pre_hornu_hranicu = []
                for index_indexov in range(len(indexy)):  # can be improved using functions together with the block above
                    if df_hotovy.floor_price[indexy[index_indexov]] > 0:
                        for i in range(1,6):
                            try:
                                if df_hotovy.swing_col[indexy[index_indexov + i]] == "SWING_HIGH":
                                    horna_hranica.append(df_hotovy.h[indexy[index_indexov + i]])
                                    index_pre_hornu_hranicu.append(indexy[index_indexov])
                                    break
                            except IndexError:
                                horna_hranica.append("NaN")
                                index_pre_hornu_hranicu.append(indexy[index_indexov])
                                break
                horna_hranica_ser = pd.Series(horna_hranica, index=index_pre_hornu_hranicu)
                df_hotovy["horna_hranica"] = horna_hranica_ser

                swing_col = df_hotovy["swing_col"][pd.isnull(df_hotovy["swing_col"]) == False][-1::-1]

                def prva_ema_rozdiel(h,l):
                    rozdiel = h - l
                    return abs(rozdiel)

                def spojovak(hranica, price):
                    for ind in df_hotovy[pd.isnull(df_hotovy[hranica]) == False][price].index:
                        df_hotovy.loc[ind, price] = float(df_hotovy.loc[ind, hranica])

                spojovak("horna_hranica", "roof_price")
                spojovak("spodna_hranica", "floor_price")
                df_hotovy.drop(["spodna_hranica",'horna_hranica'], axis=1, inplace = True)

                def ema_rozdiel(krok, indx_h, indx_l):
                    if indx_h[krok - 1] < indx_l[krok - 1]:
                        rozdiel = indx_h[krok - 1] - indx_l[krok]
                        return abs(rozdiel)
                    else:
                        rozdiel = indx_l

                def closest_opposite_swing(ind, checked_candles):
                    try:
                        for next_candle in range(1, checked_candles + 1):
                            if swing_col[swing_col.index[ind + next_candle]] != swing_col[swing_col.index[ind]]:
```

```
                                ema_range = abs(swing_col.index[ind + next_candle] - swing_col.index[ind])
                                return ema_range
                                break

                            else:
                                pass
                    except IndexError:
                        pass

                def ema_flatter(ema_range):
                    if pd.isnull(ema_range) == False and ema_range > 2:  # tu sa upravuje kolko sviecom musi mat range minimalne
                        for pokus in range(ema_range + 1):
                            if pd.isnull(df_hotovy.FLAT_INDIC[swing_col.index[ind] - pokus]) == False:
                                validate = "RANGE_VALID"
                                validate_index = swing_col.index[ind]
                                return [validate, validate_index]
                                break

                def signal(valid_range_index, range_size, void_magnitude):  # doro bit void indic
                    try:
                        for i in range(1, 10):

                            if float(df_hotovy.c[valid_range_index + i]) > df_hotovy.roof_price[valid_range_index]:  # BUY SIGNAL
                                for potencial_void in range(5):
                                    if df_hotovy.candle_size_pip[valid_range_index + i + potencial_void] > (
                                            range_size * void_magnitude):
                                        signal = "BUY {} SL {} TP {}".format(df_hotovy.roof_price[valid_range_index], df_hotovy.floor_price[valid_range_index] - 0.00020,
                                                                             df_hotovy.roof_price[valid_range_index] + (abs(df_hotovy.roof_price[valid_range_index] - (df_hotovy.floor_price[valid_range_index] - 0.00020))) * 2.5) #RRR set to 2.5:1
                                        return [valid_range_index, signal]
                                        break

                            elif float(df_hotovy.c[valid_range_index + i]) < df_hotovy.floor_price[valid_range_index]:
                                for potencial_void in range(5):
                                    if df_hotovy.candle_size_pip[valid_range_index + i + potencial_void] > (
                                            range_size * void_magnitude):
                                        signal = "SELL {} SL {} TP {}".format(df_hotovy.floor_price[valid_range_index], df_hotovy.roof_price[valid_range_index] + 0.00020,
                                                                              df_hotovy.floor_price[valid_range_index] - (abs(df_hotovy.floor_price[valid_range_index] - (df_hotovy.roof_price[valid_range_index] + 0.00020)))*2.5) #RRR set to 2.5:1
                                        return [valid_range_index, signal]
                                        break

                    except (IndexError, KeyError):
                        pass

                signal_value = []
                signal_index = []
                for ind in range(len(swing_col.index)):
                    if float(df_hotovy.roof_price[swing_col.index[ind]]) > 0:
```

```
                        ema_range = closest_opposite_swing(ind, checked_candles=5)

                        validate = ema_flatter(ema_range)

                        if isinstance(validate, list) == True:
                            range_size = float(df_hotovy.roof_price[validate[1]])
    - float(df_hotovy.floor_price[validate[1]])

                            signalik = signal(validate[1], range_size, 1.15)
                            if isinstance(signalik, list):
                                signal_value.append(signalik[1])
                                signal_index.append(signalik[0])

                signal_ser = pd.Series(signal_value, index=signal_index)
                df_hotovy["signal"] = signal_ser
                df_hotovy.to_excel(writer,par)
            writer.save()# NOT INDENTED

            print("Success! Multiple currency DF saved.")

        def cancel_all_pending_orders():
            r = orders.OrdersPending(accountID)
            api.request(r)
            pending_orders = r.response
            pending_orders_ids = [i["id"] for i in pending_orders["orders"]]
            for i in pending_orders_ids:
                r = orders.OrderCancel(accountID= accountID, orderID=i)
                api.request(r)

            return print("All orders were cancelled")
        #fire_up(acc_id= accountID, access_t= access_token)

        #cancel_all_pending_orders()
```

Príloha č. 5 Algoritmus založený na identifikácií distribučných pásiem – 2. modul obchodovania v reálnom čase

```
#RUN THIS MODUL SEPARATELY THAN RUN ORDER_MODUL
print("..............: INITIALIZING :..............")
import datetime
import time
import range_signal # initializes connection gate with 5M feed
import pandas as pd
import xlrd
from oandapyV20.endpoints import orders
accountID = range_signal.accountID
access_token = range_signal.access_token

def spusti_range_signal(): #if this function is called, request for 5M data df is
 sent
    range_signal.fire_up(accountID, access_token)

starttime = time.time()

def cistic(data_frame):

    data_frame = data_frame
    data_frame_ceny = data_frame.PRICE
    for i in range(df.shape[0]):

        try:
```

```
            if float(data_frame_ceny[i]) == float(data_frame_ceny[i+1]):
                data_frame.drop(data_frame.index[i], inplace= True)
        except :
            print("Signaly pre {} vycistene od duplicitnych objednavok, ulozene do df".format(sheet_name))
    return data_frame
if __name__ == "__main__":

    print("Refreshing active for every 60 seconds.")
    xlsx = xlrd.open_workbook("/Users/Cappucinoes/PycharmProjects/Bakalarka_PA/range_market_multi/df_hotovy.xlsx")
    while True:
        now = datetime.datetime.now().second
        writer_signals = pd.ExcelWriter("/Users/Cappucinoes/PycharmProjects/Bakalarka_PA/range_market_multi/signals.xlsx")
        if now == 1: #cakat sekundu na nahranie dat
            print("\n FIREEEEEEEEEEEEEEEEEEEEEEEEEEEE \n")
            spusti_range_signal()  # asks for data from 5M range_signal modul
            try:
                for sheet_name in xlsx.sheet_names(): # pre kazdy par v data feed exceli
                    print("Vyhladavam ordery pre {}".format(sheet_name))
                    signals = pd.read_excel("/Users/Cappucinoes/PycharmProjects/Bakalarka_PA/range_market_multi/df_hotovy.xlsx",sheet_name = sheet_name)["signal"] # najde stlpec signal pre aktualny par
                    signals = signals[pd.isnull(signals) == False]
                    columns = ["typ", "PRICE", "STOP_LOSS", "TAKE_PROFIT"]
                    data = [i.split(" ") for i in signals]
                    for i in data:
                        i.remove("SL")
                        i.remove("TP")
                    df = pd.DataFrame(data, columns= columns)
                    df_signaly_pre_par = cistic(data_frame = df)
                    df_signaly_pre_par.to_excel(writer_signals, sheet_name)
                writer_signals.save()
            except (ConnectionError) as error:
                print(error)
                pass

            print("Dataframe saved! Ready to be used with signals.xlsx, next data available in {} seconds".format(60 - datetime.datetime.now().second))
```

Príloha č. 6 Algoritmus založený na identifikácií distribučných pásiem – 3. modul obchodovania v reálnom čase

```
#SCRIPT OD ORDER_MODUL
print("Order modul bezi.")
import oandapyV20
from oandapyV20 import API
import pandas as pd
import oandapyV20.endpoints.pricing as pricing
import configparser
from range_signal import accountID, access_token
import datetime
import xlrd
api = API(access_token=access_token)
open_file = open("/Users/Cappucinoes/PycharmProjects/Bakalarka_PA/range_market_multi/obchodna_historia.txt", "a")
open_file.close()

def get_actual_price(menovy_par):
    params = {"instruments": menovy_par}
    r = pricing.PricingInfo(accountID= accountID, params= params)
    rv = api.request(r)
```

```
    return rv["prices"][0]['bids'][0]["price"]

def order_ident(df_to_parse): # vezme df z aktualneho sheetu z for loopu
    order_type = []
    order_price = []
    order_sl = []
    order_tp = []
    for i in df_to_parse.iterrows():
        if "BUY" in i[1]["typ"]:

            if abs(float(i[1]["PRICE"]) - float(i[1]["STOP_LOSS"])) < 0.00030: #ak je stoploss prilis maly
                i[1]["STOP_LOSS"] = str(float(i[1]["STOP_LOSS"]) - 0.00030) #zvacsime stoploss o hodnotu

            if abs(float(i[1]["PRICE"]) - float(i[1]["TAKE_PROFIT"])) < 0.00030: #ak je take profit prilis maly
                i[1]["TAKE_PROFIT"] = str(float(i[1]["TAKE_PROFIT"]) + 0.00050) # zvacsime ho o ciastku

            order_type.append("BUY") # priradi hodnotu do listu
            order_price.append(i[1]["PRICE"])
            order_sl.append(i[1]["STOP_LOSS"])
            order_tp.append(i[1]["TAKE_PROFIT"])

        elif "SELL" in i[1]["typ"]:

            if abs(float(i[1]["PRICE"]) - float(i[1]["STOP_LOSS"])) < 0.00030:
                i[1]["STOP_LOSS"] = str(float(i[1]["STOP_LOSS"]) + 0.00030)

            if abs(float(i[1]["PRICE"]) - float(i[1]["TAKE_PROFIT"])) < 0.00030:

                i[1]["TAKE_PROFIT"] = str(float(i[1]["TAKE_PROFIT"]) - 0.00050)

            order_type.append("SELL")
            order_price.append(i[1]["PRICE"])
            order_sl.append(i[1]["STOP_LOSS"])
            order_tp.append(i[1]["TAKE_PROFIT"])
    return order_type,order_price,order_sl,order_tp #returns 4 lists with values


import oandapyV20.endpoints.orders as orders #sluzi na otvaranie pozicii
import oandapyV20.endpoints.trades as trades #sledovanie aktualnych pozicii

def get_actual_price(menovy_par):
    params = {"instruments": menovy_par}
    r = pricing.PricingInfo(accountID= accountID, params= params)
    rv = api.request(r)
    return rv["prices"][0]['bids'][0]["price"]

def open_limit_order(units,buy_or_sell, price, stop_loss, take_profit,instrument):
    if buy_or_sell == "SELL":
        units = -units

    order_data = {
        "order": {
            "price": str(price),
            "timeInForce": "GTC",
            "instrument": str(instrument),
            "units": str(units),
            "clientExtensions": {
                "comment": "Algo rulez bitch!"
            },
            "stopLossOnFill": {
                "timeInForce": "GTC",
```

```
                "price": str(stop_loss)
            },
            "takeProfitOnFill": {
                "price": str(take_profit)
            },
            "type": "MARKET_IF_TOUCHED",
            "positionFill": "DEFAULT"
        }
    }
    r = orders.OrderCreate(accountID=accountID, data = order_data)
    api.request(r)
    print(r.response, "\n VYTVORENA POZICIA")
    history_file.write(price)

def round_up(list_to_round): # funkcia pre zaokruhlenie
    rounded_list = []
    for i in list_to_round:
        rounded_list.append(str(round(float(i), 5)))
    return rounded_list
xlsx = xlrd.open_workbook("/Users/Cappucinoes/PycharmProjects/Bakalarka_PA/range_market_multi/signals.xlsx") # assigns opened excel with signals on all sheets

while True:
          for sheet_name in xlsx.sheet_names():
              df_for_order_ident = pd.read_excel("/Users/Cappucinoes/PycharmProjects/Bakalarka_PA/range_market_multi/signals.xlsx", sheet_name = sheet_name) #assigns one sheet to df so that order_ident can parse it
              order_type, order_price, order_sl, order_tp = order_ident(df_to_parse=df_for_order_ident) # vyziada si data vo forme listov / napr order_type je list so vsetkymi typmi #aktualne refreshovany every 5 mins
              actual_price = get_actual_price(menovy_par= sheet_name)
              order_price = round_up(order_price)
              order_sl = round_up(order_sl)
              order_tp = round_up(order_tp)
              order_instrument = sheet_name # sets currency for order

              with open("/Users/Cappucinoes/PycharmProjects/Bakalarka_PA/range_market_multi/obchodna_historia.txt", "r+") as history_file:
                  obchodna_historia = history_file.read()

                  for i in range(len(order_type)):
                      if (order_type[i] == "BUY") and (order_price[i] not in obchodna_historia) and (float(actual_price) >= float(order_price[i])):
                          open_limit_order(2000,buy_or_sell= order_type[i],price = order_price[i], stop_loss= order_sl[i], take_profit= order_tp[i], instrument= order_instrument)

                      elif (order_type[i] == "SELL") and (order_price[i] not in obchodna_historia) and (float(actual_price) <= float(order_price[i])): # PRIDAT PODMIENKU ACTUAL PRICE < ORDER_PRICE[I]
                          open_limit_order(2000, buy_or_sell= order_type[i],price = order_price[i], stop_loss= order_sl[i], take_profit= order_tp[i], instrument= order_instrument)

                      else:
                          pass
```

Príloha č. 7 Algoritmus založený na identifikácií distribučných pásiem – Backtesting úprava vo forme dodatku k 1. modulu obchodovania v reálnom čase

```
def cistic(data_frame,sheet_name):
```

```
    data_frame = data_frame
    print("Zahajujem odstranovanie duplicit")
    data_frame_ceny = data_frame.PRICE
    for i in range(data_frame_ceny.shape[0]):
        try:
            if data_frame_ceny[data_frame_ceny.index[i]] == data_frame_ceny[data_frame_ceny.index[i+1]]:
                data_frame.drop(data_frame_ceny.index[i+1], inplace= True)

        except IndexError:
            print("Duplicity odstranene!")

            #print("Signaly pre {} vycistene od duplicitnych objednavok, ulozene do df".format(sheet_name))

    return data_frame
import xlrd

def vytvor_signal_excel_file(): #creates supp files
    xlsx = xlrd.open_workbook("/Users/Cappucinoes/PycharmProjects/Python-trading-Algo/Backtesting/df_backtest_feed.xlsx")
    writer_signals = pd.ExcelWriter("/Users/Cappucinoes/PycharmProjects/Python-trading-Algo/Backtesting/signals.xlsx")
    try:
        for sheet_name in xlsx.sheet_names():  # pre kazdy par v data feed exceli

            print("Vyhladavam ordery pre {}".format(sheet_name))
            signals = pd.read_excel("/Users/Cappucinoes/PycharmProjects/Python-trading-
Algo/Backtesting/df_backtest_feed.xlsx", sheet_name=sheet_name)["signal"]  # najde stlpec signal pre aktualny par
            signals = signals[pd.isnull(signals) == False]
            columns = ["typ", "PRICE", "STOP_LOSS", "TAKE_PROFIT", "IDENTIFICATOR_CANDLE"]
            data = [i.split(" ") for i in signals]
            for i in data:
                i.remove("SL")
                i.remove("TP")
                i.remove("SC")
            df = pd.DataFrame(data, columns=columns, index= signals.index)
            df_signaly_pre_par = cistic(data_frame=df,sheet_name= sheet_name)
            df_signaly_pre_par.to_excel(writer_signals, sheet_name)
        writer_signals.save()
    except (ConnectionError) as error:
        print(error)
        pass
    print("Dataframe saved! Ready to be used with signals.xlsx")
import operator

def vyhodnotenie_signalu(data_df, signal_type,signal_price,signal_sl,signal_tp,signal_index, signal_identified_index,sheet_name):
    data_df_iter = data_df.iterrows()
    for row in data_df_iter:
        data_df_row_index = row[0]
        index_condition = operator.gt(data_df_row_index,signal_index) # hlada az od indexu orderu

        if index_condition: #ak je index iteracie od orderu mozeme pokracovat
            if signal_type == "BUY": #urci kontrolu TP a SL pre buy order
                check_tp = operator.ge(data_df["h"][data_df_row_index], signal_tp)
                check_sl = operator.le(data_df["l"][data_df_row_index], signal_sl)
                def check_open():
                    for index_kontrola_open in range(signal_identified_index+1, data_df_row_index):
```

```
                        if data_df["l"][index_kontrola_open] <= signal_price:
                            return True

            elif signal_type == "SELL": #urci kontrolu TP a SL pre sell order
                check_tp = operator.le(data_df["l"][data_df_row_index], signal_tp)#spread je +0.00015
                check_sl = operator.ge(data_df["h"][data_df_row_index], signal_sl)#spread je +0.00015
                def check_open():
                    for index_kontrola_open in range(signal_identified_index+1, data_df_row_index):
                        if data_df["h"][index_kontrola_open] >= signal_price:
                            return True
# funkcia check_open() skontroluje ci bol obchod vobed otvoreny v range LIMIT - SL/TP, ak ano vrati True

            if check_tp and check_open(): #ak cena dosiahla TP (resp. ak by bola pozicia uzavreta) a ak poziciu otvorilo

                vysledok = "{} {} {} {} {} TP_HIT {} {} {}".format(data_df["time"][signal_index],sheet_name,signal_type, signal_index,signal_price, data_df_row_index, signal_tp, data_df["time"][data_df_row_index])
                print("Cena otvorenia {} {} TP_HIT {}".format(signal_type,signal_price, signal_tp))
                return vysledok
                break

            elif check_sl and check_open(): #ak cena dosiahla SL (resp. ak by bola pozicia uzavreta) a ak poziciu otvorilo
                vysledok = "{} {} {} {} {} SL_HIT {} {} {}".format(data_df["time"][signal_index],sheet_name,signal_type, signal_index, signal_price, data_df_row_index,signal_sl, data_df["time"][data_df_row_index])
                print("Cena otvorenia {} {} SL_HIT {}".format(signal_type, signal_price, signal_sl))
                return vysledok
                break
            else:
                pass

def itteruj():
    vysledky_sorted = pd.DataFrame()
    writer_vysledky = pd.ExcelWriter("/Users/Cappucinoes/PycharmProjects/Python-trading-Algo/Backtesting/vysledky.xlsx")
    xlsx = xlrd.open_workbook("/Users/Cappucinoes/PycharmProjects/Python-trading-Algo/Backtesting/df_backtest_feed.xlsx")
    for sheet_name in xlsx.sheet_names():
        vysledok = []
        signal_df = pd.read_excel("/Users/Cappucinoes/PycharmProjects/Python-trading-Algo/Backtesting/signals.xlsx", sheet_name)
        data_df = pd.read_excel("/Users/Cappucinoes/PycharmProjects/Python-trading-Algo/Backtesting/df_backtest_feed.xlsx", sheet_name)
        signal_generator = signal_df.iterrows()
        for signal_row in signal_generator:
            signal_index = signal_row[0]
            signal_type = signal_row[1]["typ"]
            signal_price = signal_row[1]["PRICE"]
            signal_sl = signal_row[1]["STOP_LOSS"]
            signal_tp = signal_row[1]["TAKE_PROFIT"]
            signal_identified_index = signal_row[1]["IDENTIFICATOR_CANDLE"]

            vysledok.append(vyhodnotenie_signalu(data_df=data_df, signal_type = signal_type, signal_price = signal_price,
                                        signal_sl = signal_sl, signal_tp = signal_tp,signal_index = signal_index, signal_identified_index = signal_identified_index,
                                                        sheet_name=sheet_name))
            ser = pd.Series(vysledok)
```

```
                ser.dropna(inplace=True)
                ser.to_excel(writer_vysledky, sheet_name)
                vysledky_sorted = pd.concat([vysledky_sorted, ser], ignore_index=True)
            writer_vysledky.save()
            vysledky_sorted.to_excel("/Users/Cappucinoes/PycharmProjects/Python-trading-Algo/Backtesting/vysledky_sorted.xlsx")

        fire_up(accountID,access_token) # fetches data into file
        vytvor_signal_excel_file() # creates supp files
        itteruj() # does condition checking

        def plus_minus(riadok):
            for i in riadok.split():
                if i == "SL_HIT" or i == "TP_HIT":
                    return i

        def extract_price(riadok, index_ceny): # index ceny je index riadku.split napr. pre open je 3
            return riadok.split()[index_ceny]


        def vysledok_v_pip(riadok):
            vysledok_v_pip = abs(riadok["Open_price"] - riadok["Closed_price"])
            if riadok["balance_effect"] == "TP_HIT":
                return vysledok_v_pip
            else:
                return -vysledok_v_pip

        def spracuj_vysledky():
            # columns = ["Date_open", "Order_type", "Size", "Pair","Open_price", "SL", "TP", "Date_closed", "Closed_price"]
            vysledky_df = pd.read_excel("/Users/Cappucinoes/PycharmProjects/Python-trading-Algo/Backtesting/vysledky_sorted.xlsx")
            vysledky_df.columns = ["data"]
            vysledky_df["Date_open"] = pd.to_datetime(vysledky_df.data.apply(lambda x: x.split()[0].replace("T"," ")[:-11]))
            vysledky_df["Order_type"] = vysledky_df.data.apply(lambda x: x.split()[2])
            vysledky_df["Pair"] = vysledky_df.data.apply(lambda x: x.split()[1])
            vysledky_df["Open_price"] = vysledky_df.data.apply(extract_price,index_ceny=4) # index ceny = na akej pozicii indexu sa nachadza pozadovana cena
            vysledky_df["Date_closed"] = pd.to_datetime(vysledky_df.data.apply(lambda x: x.split()[-1].replace("T"," ")[:-11]))
            vysledky_df["Closed_price"] = vysledky_df.data.apply(extract_price,index_ceny=7)
            vysledky_df["Closed_price"] = vysledky_df["Closed_price"].apply(lambda x: float(x))
            vysledky_df["balance_effect"] = vysledky_df.data.apply(plus_minus)
            vysledky_df["Open_price"] = vysledky_df["Open_price"].apply(lambda x: float(x))
            vysledky_df["Result"] = vysledky_df.apply(vysledok_v_pip, axis= 1)
            return vysledky_df

        import numpy as np
        desired_width = 320
        pd.set_option('display.width',desired_width)
        np.set_printoptions(linewidth=desired_width)
        pd.set_option('display.max_columns',10)
        vysledky_df = spracuj_vysledky()
        vysledky_df.to_excel("/Users/Cappucinoes/PycharmProjects/Python-trading-Algo/Backtesting/vysledky_stats.xlsx")
```

Príloha č. 8 Algoritmus založený na identifikácií distribučných pásiem – Štatistický script

```
import pandas as pd
import datetime
import oandapyV20.endpoints.forexlabs as labs
from oandapyV20 import API
pd.set_option('display.width', 320)
pd.set_option("display.max_columns", 20)

trading_history = pd.read_excel("/Users/Cappucinoes/PycharmProjects/Python-trading-Algo/Backtesting/vysledky_stats.xlsx")
trading_history = trading_history.loc[:,"Date_open":]
print(trading_history.head())
# ak dojde k odstraneniu casov alebo parov je potrebne ich zadat pred spustenim funkcie!!!
def uloz_df_na_graf(df,file_name):
    df.to_excel("/Users/Cappucinoes/PycharmProjects/Python-trading-Algo/Backtesting/{}.xlsx".format(file_name))
def vytvor_statistiku(trading_history):
    zoradene_datum_otvorenia = trading_history.sort_values(by= "Date_open")
    zoradene_datum_zatvorenia = trading_history.sort_values(by = "Date_closed")

    #WIN RATIO CELEJ STRATEGIE
    print("Celkovy pocet pipov pre L/W",trading_history.groupby(by = "balance_effect").Result.sum()) # pocet pipov W/L
    print(trading_history.groupby(by = "balance_effect").count().Result) # pomer lose win

    #currency win ratio
    stat = trading_history.groupby(by = ["Pair", "balance_effect"]).count().Result
    uloz_df_na_graf(stat, "neoptimalizovana_nospread_curr")
    print("Vysledne data pre graf na win ratio podla paru ulozene")

    #win ratio by time
    trading_history["hour"] = trading_history.Date_open.apply(lambda x: x.hour)
    stat = trading_history.groupby(by = ["hour", "balance_effect"]).count().Result
    uloz_df_na_graf(stat, "neoptimalizovana_casy_stats")
    print("Vysledne data pre graf na statistiku podla casu ulozene")

    #average win/loss in pips and AVG RRR
    stat = trading_history.groupby("balance_effect").mean().Result
    print("Priemerny pocet pipov pri stratovom obchode je ", stat[0])
    print("Priemerny pocet pipov pri ziskovom obchode je ", stat[1])
    avg_rrr = abs(stat[1]/stat[0])
    print("Priemerne risk-reward ratio je",avg_rrr,": 1")

    #WIN ratio podla buy/sell typu
    stat = trading_history.groupby(by = ["Order_type", "balance_effect"]).count().Result
    # print("WIN RATIO PRE BUY JE ", (stat["BUY"]["TP_HIT"] / stat["BUY"]["SL_HIT"])*100, " percent")
    # print("WIN RATIO PRE SELL JE ", (stat["SELL"]["TP_HIT"] / stat["SELL"]["SL_HIT"])*100, " percent")
    print(stat)

    #average hold time
    dates_df = trading_history.apply(lambda x: x.Date_closed - x.Date_open , axis = 1)
    avg_hold_time = divmod(dates_df.mean().total_seconds(), 60)
    print("Average position hold time is {} minutes and {} seconds".format(str(avg_hold_time[0])[:-2], str(avg_hold_time[1])[:2]))

    #WIN RATIO PODLA DNI + pocet dni potrebnych na obchodovanie
```

```
52.     day_col = trading_history.Date_open.apply(lambda x: x.day)
53.     trading_history["day"] = day_col
54.     prvy_den = trading_history.day.iloc[0]
55.     posledny_den = trading_history.day.iloc[trading_history.shape[0]-1]
56.     print("Na obchodovanie bolo treba {} dni".format(posledny_den - prvy_den))
57.     results_by_day = trading_history.groupby(["day", "balance_effect"]).count().R
    esult
58.     results_by_day_df = pd.DataFrame(columns=["WIN_RATIO", "LOOSE_RATIO"])
59.     for i in results_by_day.index.levels[0]:
60.         try:
61.             win_percento = results_by_day.loc[i].loc["TP_HIT"] / (
62.                         results_by_day.loc[i].loc["TP_HIT"] + results_by_day.loc[
    i].loc["SL_HIT"])
63.             loose_percento = results_by_day.loc[i].loc["SL_HIT"] / (
64.                         results_by_day.loc[i].loc["TP_HIT"] + results_by_day.loc[
    i].loc["SL_HIT"])
65.             dictionary = dict(zip(["WIN_RATIO", "LOOSE_RATIO"], [win_percento, lo
    ose_percento]))
66.             results_by_day_df.loc[i] = dictionary
67.         except (IndexError, KeyError):
68.             pass
69.
70.     uloz_df_na_graf(results_by_day_df.WIN_RATIO, "WIN_RATIO_BY_DAY")
71.     print("Win ratio podla obchodnych dni ulozene.")
72. vytvor_statistiku(trading_history)
```

Všetky zdrojové kódy sú taktiež dostupné v GitHub repozitári na adrese:

https://github.com/Cappucinoes/Python-trading-Algo